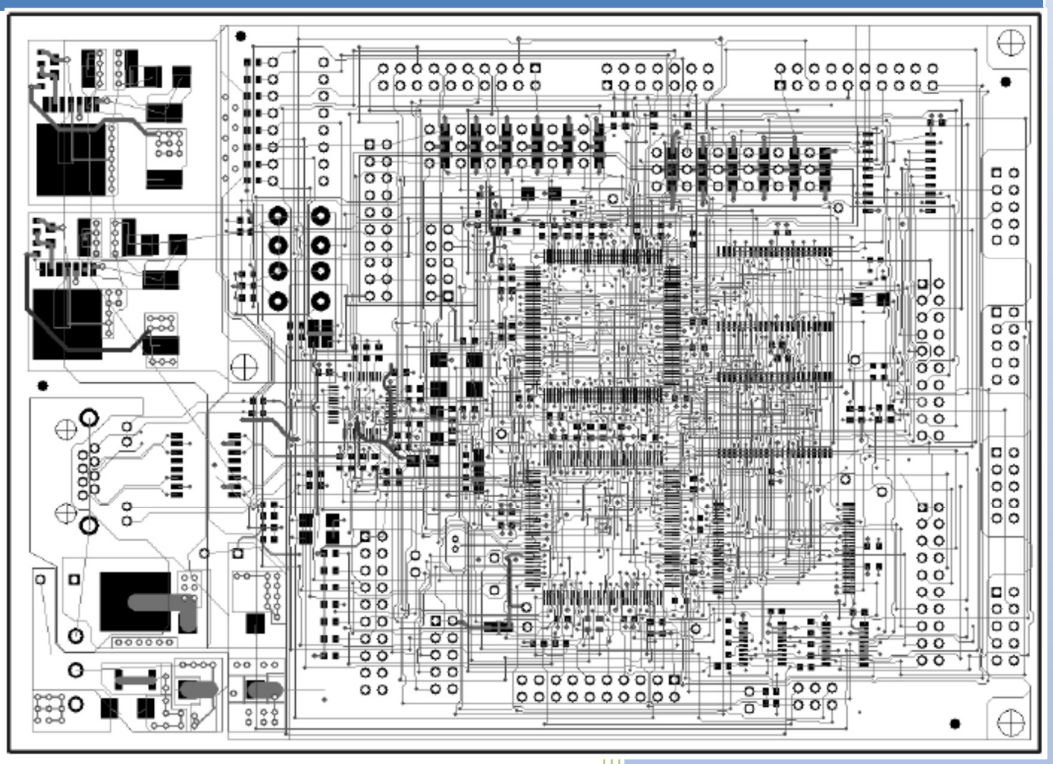
Εύρωστος Έλεγχος Στην Πλοήγηση Σκαφών Επιφανείας



Ελευθέριος Κ. Λόγης

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛ/ΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ & ΡΟΜΠΟΤΙΚΗΣ

ΕΥΡΩΣΤΟΣ ΕΛΕΓΧΟΣ ΣΤΗΝ ΠΛΟΗΓΗΣΗ ΣΚΑΦΩΝ ΕΠΙΦΑΝΕΙΑΣ

ΕΛΕΥΘΕΡΙΟΣ ΚΩΝΣΤΑΝΤΙΝΟΥ ΛΟΓΗΣ

Η ΕΠΤΑΜΕΛΗΣ ΕΠΙΤΡΟΠΗ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ	ΥΠΟΓΡΑΦΗ
1. Κουσιουρής Τρύφων	Καθηγητής	
2. Γεωργίου Ιωάννης	Καθηγητής	
3. Γρηγορόπουλος Γρηγόριος	Καθηγητής	
4. Καϊκτσής Λάμπρος	Καθηγητής	
5. Μαράτος Νικόλαος	Καθηγητής	
6. Ξηρός Νικόλαος	Καθηγητής	
7. Ψυλλάκης Χαράλαμπος	Λέκτορας	

Πίνακας Περιεχομένων

1 Εισαγωγή

Η παρούσα διδακτορική διατριβή ασχολείται με το σχεδιασμό των ηλεκτρονικών, τη μοντελοποίηση και τον έλεγχο ενός σκάφους επιφανείας, που δημιουργήθηκε για να κατευθύνεται σε ένα σύνολο προκαθορισμένων στόχων σε έναν διδιάστατο χάρτη. Αναπτύχθηκε σε τηλεχειριζόμενο μοντέλο σκάφους τύπου Ferry Boat, που συναρμολογήθηκε στο ΕΜΠ, στο οποίο δόθηκε και δυνατότητα αυτοκατεύθυνσης. Με εξαίρεση τα πλαστικά του σκάφους, το χειριστήριο τηλεκατεύθυνσης και τον δέκτη τηλεκατεύθυνσης, όλα τα υπόλοιπα ηλεκτρονικά συστήματα τα οποία χρειάστηκαν, κατασκευάστηκαν από τον συγγραφέα της παρούσας διατριβής.

Οι λεπτομέρειες της ανάπτυξης των ηλεκτρονικών, καθώς και η μεθοδολογία που ακολουθήθηκε για τη μοντελοποίηση και τον έλεγχο του σκάφους, παρουσιάζονται στη διατριβή.

Μετά την κατασκευή του σκάφους, συλλέχθηκαν δεδομένα για την απόκρισή του, πραγματοποιώντας χειροκίνητους ελιγμούς από τη μονάδα ραδιοελέγχου. Τα δεδομένα που προέκυψαν, χρησιμοποιήθηκαν για την εκπαίδευση ενός νευρωνικού δικτύου, για την πρόβλεψη της απόκρισης του σκάφους στα χειριστήρια ώσης και διεύθυνσης. Η προσέγγιση της μοντελοποίησης βασίστηκε στη λογική μαύρου κουτιού (black box modelling). Το μοντέλο που προέκυψε χρησιμοποιήθηκε για τη διεξαγωγή προσομοιώσεων σχετικά με τις μεθόδους ελέγχου που αναπτύχθηκαν για το σκάφος.

Για να μπορέσει το σκάφος να φτάσει σε καθορισμένους στόχους στο χάρτη, συντέθηκαν δύο τύποι εποπτικών ελεγκτών. Αυτοί οι αλγόριθμοι παρακολουθούν τη θέση του σκάφους σε σχέση με τον στόχο του και δημιουργούν εντολές μεταφορικής και γωνιακής ταχύτητας. Οι εντολές αυτές αποτελούν τις τιμές αναφοράς για τους αντίστοιχους βρόχους ελέγχου.

Για την αποτελεσματική αντιμετώπιση των μη γραμμικοτήτων του συστήματος, αναπτύχθηκε μια μεθοδολογία βασισμένη στο μοντέλο νευρωνικών δικτύων, η οποία επιπροσθέτως αποσυνέζευξε τους βρόχους ελέγχου ταχύτητας και κατεύθυνσης. Χρησιμοποιήθηκαν επίσης τεχνικές σθεναρού ελέγχου, για τη μείωση της επίδρασης των ανακριβειών πρόβλεψης του μοντέλου και της κβαντοποίησης της δράσης ελέγχου, προκειμένου να αμβλυνθεί το προκύπτον σφάλμα.

Στη συνέχεια, η απόκριση του σκάφους προσομοιώθηκε χρησιμοποιώντας διάφορα σενάρια, ώστε να εξεταστούν οι ικανότητες ελιγμών του. Πραγματοποιήθηκε επίσης σύγκριση των δύο εποπτικών ελεγκτών.

Τέλος, προσομοιώθηκε ένα μικρό σμήνος τριών σκαφών, για τη διερεύνηση της επέκτασης των μεθόδων που περιγράφονται στη διατριβή, σε σύνολα σχετιζομένων σκαφών.

1.1 Καινοτομία και Συνεισφορά στην Επιστημονική Έρευνα

Η καινοτομία της παρούσας διατριβής σχετίζεται με τα ακόλουθα σημεία.

Πλαίσιο μοντελοποίησης του σκάφους.

Η μοντελοποίηση ενός σκάφους επιφανείας με αποκλειστική χρήση νευρωνικών δικτύων δεν συνιστά συνήθη πρακτική στη σχετική βιβλιογραφία. Στην παρούσα διατριβή, ο λόγος για αυτήν την επιλογή ήταν διττός. Πρώτον, το μέγεθος του θεωρούμενου σκάφους είναι μικρό σε σχέση με ακόμη και τις παραμικρές διαταραχές από την επιφάνεια του νερού, και έτσι η υιοθέτηση άλλων προσεγγίσεων, όπως, για παράδειγμα, η χρήση των εξισώσεων Nomoto, οδηγεί σε λύσεις λιγότερο ακριβείς. Δεύτερον, από ακαδημαϊκή άποψη, είναι σημαντικό να ελεγχθεί εάν η χρήση μιας προσέγγισης μοντελοποίησης «μαύρου κουτιού» (“black box”) μπορεί να είναι αποτελεσματική για τη θεωρούμενη κατηγορία προβλημάτων.

Έτσι, το πλαίσιο μοντελοποίησης της παρούσας εργασίας διαφέρει ουσιαστικά από αυτό αντίστοιχων μελετών της βιβλιογραφίας, οι οποίες βασίζονται είτε σε φυσικά μοντέλα είτε στις εξισώσεις Nomoto [1]–[3].

Υλοποίηση αλγορίθμου PID με χρήση της αθροιστικής προσέγγισης του ολοκληρώματος.

Στην πλειονότητα των σχετικών εργασιών της βιβλιογραφίας, η υλοποίηση του ελέγχου PID βασίζεται στην ψηφιοποίηση του PID μέσω τραπεζοειδούς ολοκλήρωσης - βλέπε βιβλίο του Ogata [4] και το σχετικό παράρτημα του αγγλικού κειμένου της διατριβής (κείμενο στην αγγλική ΠΑΡΑΡΤΗΜΑ 5: «Alternative Implementation of PID Controllers (trapezoidal Integration)»). Δεδομένου ότι οι προσομοιώσεις μας χρησιμοποίησαν προσεγγίσεις αθροίσματος για να υπολογίσουν επιταχύνσεις και ταχύτητες, επιλέξαμε να ψηφιοποιήσουμε το PID χρησιμοποιώντας την προσέγγιση άθροισης, προκειμένου να ενοποιηθεί η προσέγγισή μας αναφορικά με την ολοκλήρωση. Η όλη διαδικασία περιγράφεται στην παρούσα διατριβή, ενώ επίσης έχει δημοσιευθεί στην εργασία [5].

Εισαγωγή της μονάδας “Model Based Controller” (MBC).

Η μονάδα MBC, η οποία εισήχθη στην παρούσα μελέτη, εκφράζει τη σχέση μεταξύ των ζητούμενων επιταχύνσεων και της απαιτούμενης δράσης ελέγχου, βελτιστοποιώντας την επιλογή της με βάση τις εφικτές αποκρίσεις, κάνοντας χρήση προβλέψεων από τη διαδοχική εφαρμογή πιθανών ενεργοποιήσεων στο νευρωνικό μοντέλο του σκάφους. Εξ όσων γνωρίζουμε, δεν έχει χρησιμοποιηθεί αντίστοιχη

προσέγγιση στη διεθνή βιβλιογραφία. Λεπτομερής περιγραφή παρουσιάζεται στην ενότητα 5.4.1.

Ο αλγόριθμος μη γραμμικού εποπτικού ελέγχου.

Οι μη γραμμικές εξισώσεις που χρησιμοποιήθηκαν εδώ για τη δημιουργία των νόμων βάσει των οποίων το σκάφος οδηγείται στον στόχο του δεν έχουν αναφερθεί, εξ όσων γνωρίζουμε, στη διεθνή βιβλιογραφία, και παρουσιάζονται για πρώτη φορά στην παρούσα διατριβή. Όπως αποδεικνύεται στη διατριβή, το σχετικό σφάλμα τείνει ασυμπτωτικά στο μηδέν. Έτσι, προϊόντος του χρόνου, οι δράσεις ελέγχου που παράγονται από τις εν λόγω μη γραμμικές εξισώσεις οδηγούν το σκάφος στον στόχο.

1.2 Ευχαριστίες

Από τα έως τώρα βιώματα, έχω την πεποίθηση ότι η προσωπική προσπάθεια του καθενός αυξάνεται τόσο αθροιστικά όσο και πολλαπλασιαστικά με συντελεστές που έχουν να κάνουν με τους ανθρώπους που τον περιβάλλουν. Εάν οι άνθρωποι αυτοί λειτουργούν υποστηρικτικά, τότε η ενέργεια του αυξάνεται και φτάνει στους στόχους του πολύ πιο εύκολα.

Για το λόγο αυτό νιώθω την υποχρέωση να ευχαριστήσω τους ανθρώπους που με στήριξαν τόσο στο γνωστικό, όσο και στο ηθικό μέρος, προκειμένου να φτάσω στο τέλος αυτής της διατριβής. Στήριξη που παρείχαν, ακόμα και σε περιόδους όπου δεν είχα εμφανή πρόοδο.

Για τους παραπάνω αλλά και πολλούς ακόμα λόγους, θα ήθελα να ευχαριστήσω τον καθηγητή κύριο Ξηρό Νικόλαο, ο οποίος με ενέπνευσε να ξεκινήσω στον τομέα αυτό, με δέχθηκε σα φοιτητή του και στάθηκε δίπλα μου καθόλη τη διάρκεια της πορείας μου. Θα ήθελα επίσης να ευχαριστήσω τον καθηγητή κύριο Κουσιουρή Τρύφωνα, που ανέλαβε διοικητικά αλλά και εκπαιδευτικά την ευθύνη της διατριβής από την σχολή των Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του ΕΜΠ. Ένα ευχαριστώ επίσης και στον καθηγητή κύριο Νικόλαο Μαράτο για τη στήριξη που παρείχε σε όλα τα διοικητικά θέματα.

Στο σημείο αυτό θεωρώ υποχρέωσή μου να αφιερώσω μια ξεχωριστή παράγραφο στη σύζυγό μου Ιωάννα Κουκοπούλου, η οποία έχει υπομείνει ατελείωτες ώρες απομόνωσής μου από εκείνη, προκειμένου να μελετήσω, να υλοποιήσω και να καταγράψω τμήματα της εργασίας. Όλα αυτά πολλές φορές και κατά τη διάρκεια των διακοπών μας, ή για να το πω πιο σωστά.. κυρίως κατά τη διάρκεια των διακοπών μας!!). Χωρίς την πολύτιμη στήριξη και κατανόησή της, η ολοκλήρωση του δρόμου αυτού θα ήταν αδύνατη.

Για επίσης πολλούς λόγους θα ήθελα να ευχαριστήσω τον αδερφό μου Δημήτριο Λόγη. Είναι ο άνθρωπος που μου γνώρισε τον κόσμο της ηλεκτρονικής και μετέτρεψε το ενδιαφέρον μου σε πάθος για σχεδίαση και προγραμματισμό ηλεκτρονικών. Εκτός

των άλλων βοήθησε και στην εργασία αυτή κατά τη λήψη των μετρήσεων για τη μοντελοποίηση, ενώ έδωσε ιδέες σε διάφορες φάσεις της διατριβής.

Μιλώντας για μέλη οικογενείας, θα πρέπει να ευχαριστήσω τη θεία μου Δέσποινα Γκουρβέλου, μαθηματικό, η οποία κατέβαλε κάθε δυνατή προσπάθεια ώστε να αποκτήσω όσο καλύτερο μαθηματικό υπόβαθρο μπορούσα, από τα πρώτα στάδια της μάθησής μου έως και την αποφοίτηση από το Πολυτεχνείο.

Παραμένοντας στα οικογενειακά πλαίσια, θα πρέπει να ευχαριστήσω τους γονείς μου Μαρίνα Γκουρβέλου που μου έμαθε να εργάζομαι υπεύθυνα με επιμονή και υπομονή και τον πατέρα μου και καθηγητή ιατρικής Κωνσταντίνο Λόγη που «άνοιξε το δρόμο» αναφορικά με την ακαδημαϊκή πορεία και μας έφερε σε επαφή με τις απαιτήσεις της.

Ο Δρ. Κωνσταντίνος Ξάνθος είναι επίσης ένας ακόμα μαθηματικός τον οποίο πρέπει να ευχαριστήσω. Με βοήθησε σε προχωρημένα θέματα μαθηματικών σχετιζόμενα με τη διατριβή και μάλιστα αφιλοκερδώς!

Θα ήθελα επίσης να ευχαριστήσω τον Δρ. Νικόλαο Κουβαρά Ναυπηγό Μηχανολόγο Μηχανικό, που ανέλαβε εξολοκλήρου το στάδιο της συγκόλλησης-συναρμολόγησης των πολυεστερικών και της τοποθέτησης των κινητήρων του σκάφους.

Τέλος, θα ήθελα να ευχαριστήσω τη διοίκηση και τους συναδέλφους μου στην εταιρία που εργάζομαι, οι οποίοι με υποστήριξαν σε κάθε περίπτωση που χρειάστηκε να φύγω από την εταιρία, προκειμένου να εξυπηρετήσω κάποια υποχρέωσή μου σχετική με τη διατριβή μου.

1.3 Σύμβολα, Μεταβλητές και Συντομογραφίες

Η παράγραφος αυτή έχει ως στόχο τη συλλογή όλων των συμβόλων που χρησιμοποιήθηκαν, προκειμένου να διευκολυνθεί η κατανόηση του κειμένου.

Λίστα Συμβόλων	
Σύμβολο	**Ορισμός**
f_c	Νευρωνικό μοντέλο σκάφους «συνεχούς χρόνου».
y_T	Κανόνας πρόωσης προς μηχανές σκάφους (γκάζι).
δ_R	Κανόνας στροφής.
g	New system that creates remote control outputs (y_T, δ_R), based on $\vec{\boldsymbol{v}} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, in such a way as to provide linear system response with respect to $\vec{\mathbf{v}}$.
A	Η Ιακωβιανή που αφορά στην απόκριση μηδενικής εισόδου.
B	Η Ιακωβιανή που αφορά στην απόκριση μηδενικής κατάστασης υπό την επίδραση εισόδου.

u_I	Διάνυσμα εισόδου στο γραμμικοποιημένο σύστημα.
u	Μεταφορική ταχύτητα σκάφους σε Km/h.
ω	Γωνιακή ταχύτητα στροφής (αλλαγής πορείας) σκάφους σε °/sec.
x	Μεταβλητές κατάστασης συστήματος, δηλαδή $\begin{bmatrix} u \\ \omega \end{bmatrix}$.
α	Μεταφορική επιτάχυνση ($= \dot{u}$) σε m/(sec)^2.
$\dot{\omega}$	Γωνιακή επιτάχυνση σε °/(sec)^2.
$y_{\%}$	Κανονικοποιημένη εντολή «γκαζιού» (ισχύος στους κινητήρες), για χρήση στο μοντέλο του συστήματος μέσω νευρωνικών δικτύων. Στηρίζεται στον τύπο: $$y_{\%} = \frac{y_T}{256}$$
$\delta_{\%}$	Κανονικοποιημένη εντολή αλλαγής πορείας («τιμόνι» ή για την ακρίβεια εντολή στροφής ακροφυσίων waterjets), για χρήση στο μοντέλο του συστήματος μέσω νευρωνικών δικτύων. Στηρίζεται στον τύπο: $$\delta_{\%} = \frac{\delta_R - 121}{256}$$
$u_{\%}$	Κανονικοποιημένη μεταφορική ταχύτητα, για χρήση στο μοντέλο του συστήματος μέσω νευρωνικών δικτύων. Στηρίζεται στον τύπο: $$u_{\%} = \frac{u}{30\text{Km/h}}$$
$\omega_{\%}$	Κανονικοποιημένη γωνιακή ταχύτητα για χρήση στο μοντέλο του συστήματος μέσω νευρωνικών δικτύων. Στηρίζεται στον τύπο: $$\omega_{\%} = \frac{\omega_T}{50^o/\text{sec}}$$
$\alpha_{\%}$ or $\dot{u}_{\%}$	Κανονικοποιημένη μεταφορική επιτάχυνση για χρήση στο μοντέλο του συστήματος μέσω νευρωνικών δικτύων. Στηρίζεται στον τύπο: $$\alpha_{\%} = \frac{\alpha}{3.4\text{m/(sec)}^2}$$
$\dot{\omega}_{\%}$	Κανονικοποιημένη γωνιακή επιτάχυνση για χρήση στο μοντέλο του συστήματος μέσω νευρωνικών δικτύων. Στηρίζεται στον τύπο: $$\dot{\omega}_{\%} = \frac{\dot{\omega}}{130^o/\text{sec}^2}$$
y_C	Κανόνας ισχύος από ελεγκτή ανάδρασης (Feedback Controller ή FC).
δ_C	Κανόνας στροφής από FC.
Ox_0y_0	Σταθερό σύστημα αναφοράς για τον δισδιάστατο χώρο στον οποίο μελετάται η κίνηση του σκάφους και οι θέσεις των στόχων.
$O'x_Ry_R$	Σύστημα αναφοράς παράλληλο στο Ox_0y_0, το οποίο ακολουθεί το κέντρο μάζας του σκάφους.

O'xy	Σωματόδετο σύστημα αναφοράς, το οποίο είναι δεμένο στο κέντρο μάζας του σκάφους και ακολουθεί τις κινήσεις του.
$\vec{R}$	Διάνυσμα που ενώνει το O με το O'.
$\vec{r}$	Διάνυσμα που συνδέει το O' με τον στόχο.
$\vec{T}$	Διάνυσμα που συνδέει το O με τον στόχο.
(x_t, y_t)	Συντεταγμένες στόχου με αναφορά στο Ox_0y_0.
θ	Γωνία μεταξύ πλώρης σκάφους και διανύσματος $\vec{r}$. Θετική ανθωρολογιακά.
φ	Γωνία μεταξύ πλώρης σκάφους και άξονα x_R. Θετική ανθωρολογιακά.
φ_r	Γωνία μεταξύ $\vec{r}$ και x_R ή x_0 άξονα. Θετική ανθωρολογιακά.
T_s	Συχνότητα δειγματοληψίας για όλες τις λειτουργίες του συστήματος. Η συχνότητα δειγματοληψίας που χρησιμοποιήθηκε στην εργασία αυτή είναι 0.03s.

Table 1: Singals, units and notations.

Λίστα Συντομογραφιών	
Σύμβολο	**Ορισμός**
μC	Μονάδα μικροελεγκτή. Αναφέρεται στο αναπτυξιακό σύστημα, το οποίο είναι το κέντρο λήψεως αποφάσεων του σκάφους.
SC	Εποπτικός Ελεγκτής (Supervisory Controller)
FC	Ελεγκτής Ανατροφοδότησης (Feedback Controller)
MBC	Model Based Controller

Λίστα Συμβόλων	
Σύμβολο	**Ορισμός**
bold italics	Σηματοδοτεί ότι πρόκειται για κώδικα (C, VHDL ή Matlab). Για παράδειγμα ***plot(X,Y)***.

1.4 Παράγοντες Κανονικοποίησης

Παράγων Κανονικοποίησης	**Τιμή**	**Μονάδες**
Thrust Vector	255	Άνευ
Steer Vector	Steer vector has middle point (no steering) at 121 units and is normalized by 128. Range: [-0.9453125, 1.046875] but we keep it up to 1, so the input must be up to 249.	Άνευ
SpeedX	30	Km/h

Yaw Rate	50	o/sec
Acceleration X	3.4	m/sec^2
Yaw Acceleration	130	o/sec^2

1.5 Οριακές Τιμές

Μεταβλητή	Κάτω Όριο	Πάνω Όριο	Μονάδες
Thrust Vector	0	255	Άνευ
Steer Vector	0	249	Άνευ
SpeedX	0	30	Km/h
Yaw Rate	-50	50	o/sec
Acceleration X	-3.4	3.4	m/sec^2
Yaw Acceleration	-130	130	o/sec^2

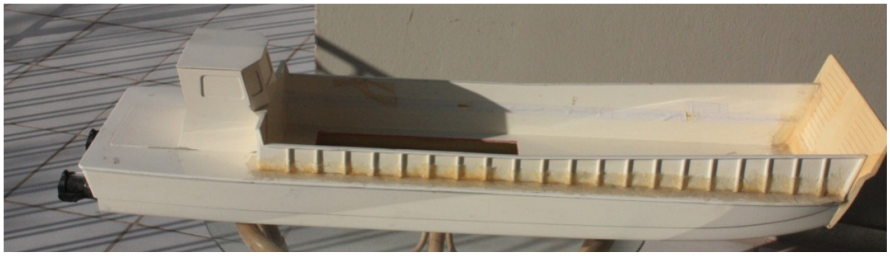
1.6 Κανονικοποιημένα Όρια κατά την Εκπαίδευση των Νευρωνικών Δικτύων

Variable	Low Limit	High Limit	Units
Thrust Vector	0	1	Unitless
Steer Vector	-1	1	Unitless
SpeedX	0	1	Km/h
Yaw Rate	-1	1	o/sec
Acceleration X	-1	1	m/sec^2
Yaw Acceleration	-1	1	o/sec^2

Equation Section (Next)

2 Σκάφος Επιφανείας

Το σκάφος που χρησιμοποιήθηκε στη διατριβή εικονίζεται στην Εικόνα 1:



Εικόνα 1. Σκάφος που χρησιμοποιήθηκε για τη διατριβή.

Είναι ένα πλοίο τύπου ferry boat, επίπεδου κύτους, εξοπλισμένο με δύο εκτοξευτήρες νερού (waterjets) για πρόωση [6]. Αυτός ο τύπος πρόωσης επιτρέπει τη στροφή ακόμη και από μηδενική ταχύτητα.

Το τμήμα των εκτοξευτών νερού που εκτείνεται πέρα από το κύτος του σκάφους φαίνεται στην Εικόνα 2. Ακροφύσια εκτοξευτών νερού.. Αυτά τα ακροφύσια συνδέονται σε μια σφαιρική άρθρωση που τους επιτρέπει να περιστρέφονται. Έτσι, αλλάζοντας την κατεύθυνση της εκροής του νερού, δημιουργούνται δυνάμεις που περιστρέφουν το σκάφος.

Οι εκτοξευτές νερού οδηγούνται από δύο ηλεκτρικούς κινητήρες [7] (κάθε εκτοξευτής έχει το δικό του κινητήρα). Οι παράμετροι που ελέγχονται για τον κάθε εκτοξευτή είναι:

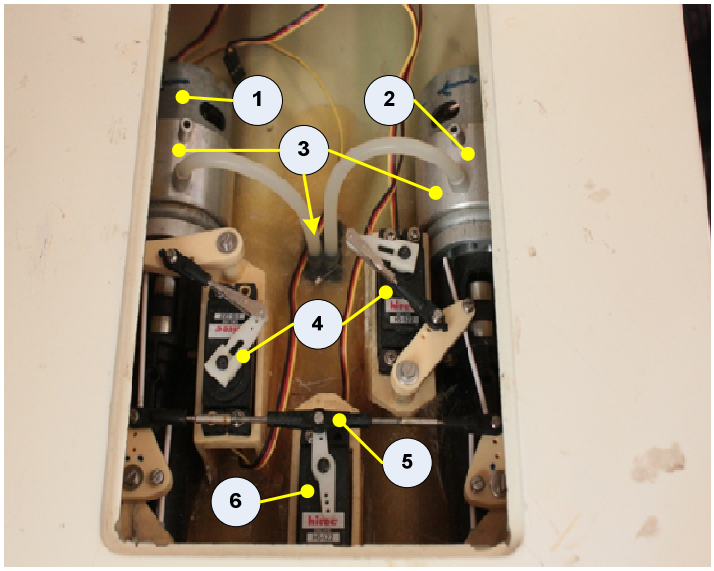
- Οι στροφές ανά λεπτό της εσωτερικής πτερωτής.
- Η γωνία εκτόξευσης νερού.
- Η κατεύθυνση της πορείας (πρόσω ή όπισθεν).

Όλες οι παραπάνω παράμετροι ελέγχονται ταυτόχρονα και για τους δύο εκτοξευτές νερού. Η εσωτερική δομή που υποστηρίζει αυτές τις κινήσεις φαίνεται στην Εικόνα 3.

Εικόνα 2. Ακροφύσια εκτοξευτών νερού.

Εικόνα 3. Σερβομηχανισμοί ελέγχου εκτοξευτών και ηλεκτροκινητήρες πρόωσης.

Η Εικόνα 3 δίνει μια σαφή εικόνα του μηχανισμού πρόωσης και στροφής του σκάφους. Η Εικόνα 4 προσθέτει κάποια σημάδια για να βοηθήσει στην επεξήγηση των εξαρτημάτων.



Εικόνα 4. Εικόνα επεξήγησης στοιχείων ελέγχου πρόωσης.

Αναφερόμενοι στην Εικόνα 4, το σύστημα στροφής και πρόωσης αποτελείται από τα ακόλουθα μέρη:

1. Ηλεκτροκινητήρες. Αυτοί οι κινητήρες παρέχουν την ώθηση για τους έλικες υδροβολής.
2. Εξωτερικός δακτύλιος ψύξης που περικλείει τους κινητήρες. Καθώς η μεταφορική ταχύτητα του σκάφους αυξάνεται, μια ποσότητα νερού πιέζεται στους σωλήνες σιλικόνης που φαίνονται με τον αριθμό (3). Το νερό κυκλοφορεί στον εξωτερικό δακτύλιο ψύχοντας τους κινητήρες και απορρίπτεται από έναν άλλο σωλήνα

σιλικόνης που δεν είχε συνδεθεί ακόμη τη στιγμή που τραβήχτηκε αυτή η φωτογραφία.

3. Σωλήνες σιλικόνης και είσοδος νερού για ψύξη κινητήρα.
4. Σέρβομηχανισμοί που ελέγχουν το πίσω πτερύγιο που αλλάζει την κατεύθυνση της πρόωσης (κίνηση εμπρός – πίσω).
5. Άξονας που ενώνει τα ακροφύσια για ταυτόχρονη στροφή (αριστερά δεξιά).
6. Ο σερβοκινητήρας που χρησιμοποιείται για την περιστροφή των ακροφυσίων.

2.1 Γρήγορη εισαγωγή στον ηλεκτρονικό εξοπλισμό του σκάφους

Για την υποστήριξη όλων των λειτουργιών που απαιτούνται για την παροχή δεδομένων για μοντελοποίηση και για τον αυτόνομο έλεγχο, το σκάφος είναι εξοπλισμένο με τις ηλεκτρονικές συσκευές που αναφέρονται παρακάτω:

No	Electronic Device	Notes
1	Πλακέτα ανάπτυξης με μικροελεγκτή ARM	σχεδιάστηκε και αναπτύχθηκε ειδικά για αυτό το σκάφος. Ξεχωριστή ενότητα θα είναι αφιερωμένη στις δυνατότητες υλικού και τη δομή αυτής της πλακέτας, αλλά για τις ανάγκες αυτής και της επόμενης ενότητας, αρκεί να αναφέρουμε ότι παρέχει όλες τις διεπαφές, τις RAM και τις μνήμες Flash που συλλέγουν και αποθηκεύουν δεδομένα ενώ ο επεξεργαστής ARM μπορεί να τα επεξεργάζεται σε πραγματικό χρόνο όπως απαιτείται.
2	Δέκτης τηλεχειρισμού	Δέκτης της μονάδας τηλεχειριστηρίου. Μεταφράζει το ραδιοσήμα σε εξόδους PWM που ελέγχουν τους σερβοκινητήρες.
3	Μονάδα μέτρησης αδράνειας	Συμβολίζεται ως IMU. Ανήκει στους αισθητήρες 6-DoF (βαθμοί ελευθερίας). Βοήθησε στην καταγραφή της μεταφορικής επιτάχυνσης και του ρυθμού στροφής (αλλαγής πορείας) όπως απαιτείται από την εφαρμογή μας.
4	Ηλεκτρονικό μαγνητόμετρο	3D αισθητήρας του μαγνητικού πεδίου της Γης.
5	Ασύρματη ζεύξη τύπου Zigbee.	Σειριακή θύρα που εκπέμπεται μέσω ραδιοζεύξης. Χρησιμοποιώντας αυτή τη συσκευή, συνδέουμε την πλακέτα ανάπτυξης στον υπολογιστή. Έχει αναπτυχθεί ένα CLI για την αποστολή εντολών και την εκτέλεση συγκεκριμένων λειτουργιών ή λήψη δεδομένων.

6	Μονάδα GPS.	Μονάδα εντοπισμού θέσης.
7	Δύο αισθητήρες RPM για τους κινητήρες	Οπτικοί αισθητήρες ανίχνευσης της περιστροφής των κινητήρων.
8	Smart camera	Μονάδα που εντοπισμού φωτεινών στόχων. Αρχικά είχε ως στόχο τη δημιουργία μιας εντολής θέσης, αλλά εγκαταλείφθηκε ως ιδέα στην πορεία.

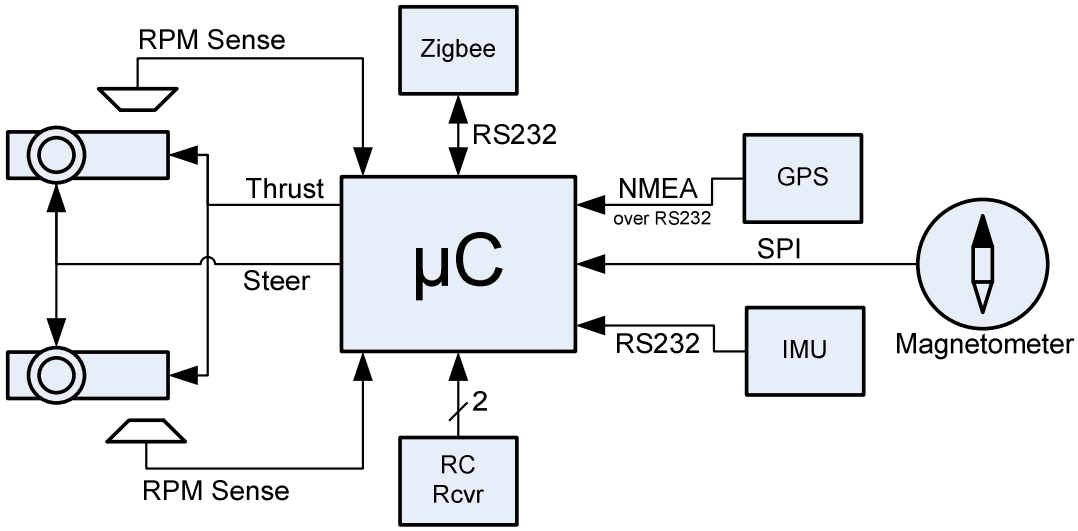
Χρησιμοποιώντας τον παραπάνω εξοπλισμό, συλλέξαμε μετρήσεις για να μοντελοποιήσουμε μαθηματικά τη συμπεριφορά του σκάφους.

Να σημειωθεί ότι αυτή η ενότητα είναι μόνο μια σύντομη εισαγωγή στα ηλεκτρονικά του σκάφους. Περισσότερες λεπτομέρειες υπάρχουν σε μια ενότητα αφιερωμένη στο υλικό στην αγγλική έκδοση του κειμένου.

Equation Section (Next)

3 Στοιχεία και Δομή Συστήματος Ελέγχου

Το παρακάτω σχήμα δίνει μια γρήγορη επισκόπηση του συστήματος ελέγχου.



Σχήμα 1. Δομή συστήματος ελέγχου.

Η πλακέτα ανάπτυξης που επισημαίνεται ως μC λαμβάνει δεδομένα από αισθητήρες RPM, GPS, IMU, μαγνητόμετρο και δημιουργεί αποφάσεις ώσης και διεύθυνσης για τους εκτοξευτές νερού.

Για τη διαδικασία ταυτοποίησης του συστήματος χρησιμοποιήθηκε το τηλεχειριστήριο για τη δημιουργία των σημάτων ώσης και διεύθυνσης και ο μC κατέγραφε τις εντολές του μαζί με την έξοδο των αισθητήρων.

Equation Section (Next)

4 Ταυτοποίηση Συστήματος Σκάφους

Προκειμένου να δημιουργηθεί μοντέλο για τον έλεγχο του σκάφους, έγιναν μετρήσεις των αποκρίσεων σε κινήσεις που υπαγορεύθηκαν μέσω τηλεχειρισμού. Οι μετρήσεις χρησιμοποιήθηκαν μετά από επεξεργασία αποθορυβοποίησης και κανονικοποίησης, ώστε να εκπαιδευτούν τα νευρωνικά δίκτυα που θα αποτελέσουν το μαθηματικό μοντέλο του σκάφους.

Οι ενότητες που ακολουθούν περιγράφουν τη διαδικασία.

Equation Section (Next)

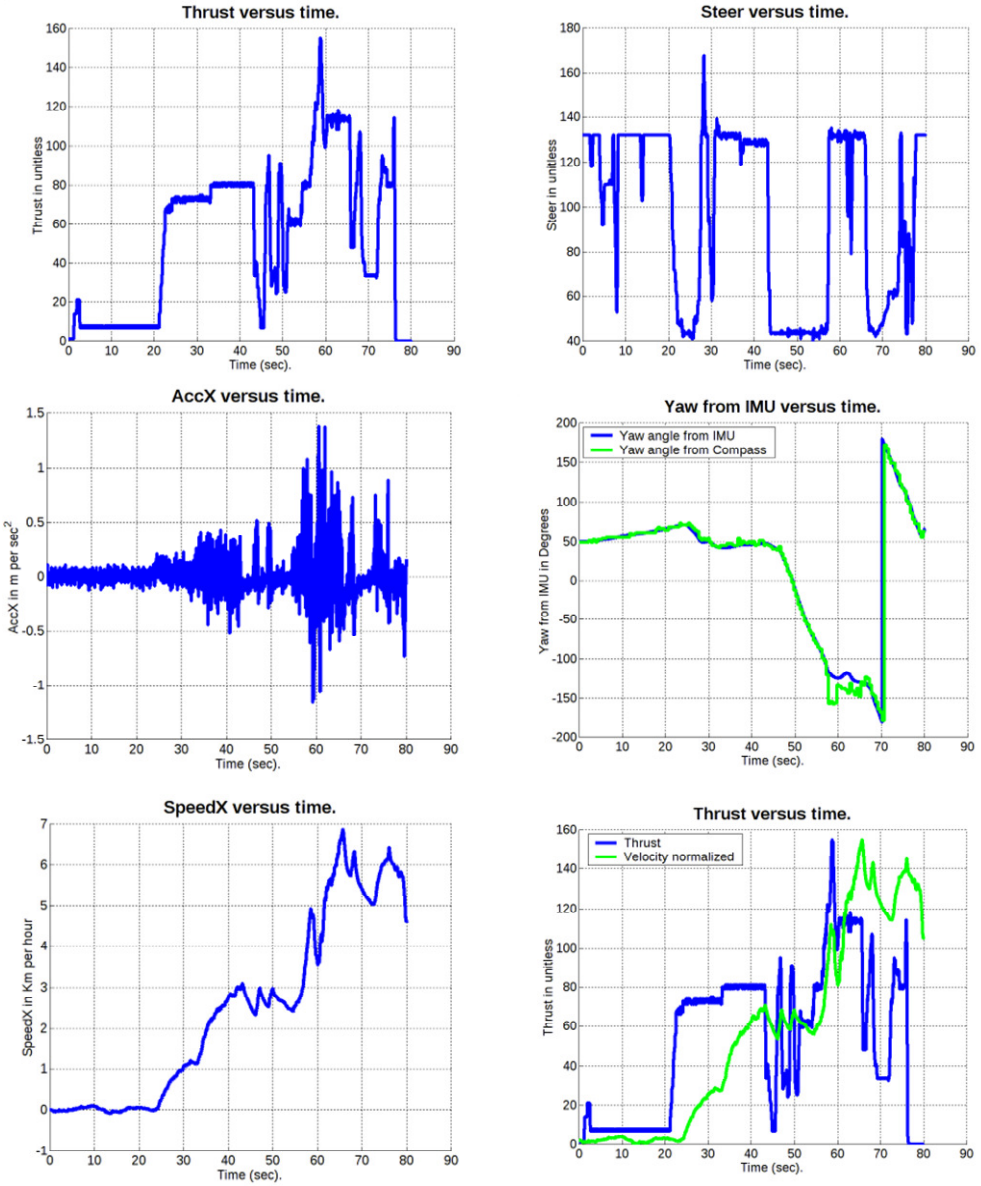
4.1 Μετρήσεις

Όλες οι μετρήσεις επί του πεδίου έγιναν από τα ηλεκτρονικά του σκάφους, τα οποία κατέγραφαν στις SRAM τις κινήσεις του χειριστή, τις επιταχύνσεις σε έξι άξονες, τον προσανατολισμό σε σχέση με τον Βορρά, τις στροφές των μηχανών και τη θέση στο χάρτη (GPS). Μετά την ολοκλήρωση κάθε δοκιμής, τα δεδομένα αποστέλλωντο μέσω σειριακής σε υπολογιστή με τη χρήση εντολών CLI που αναπτύχθηκαν για τον επεξεργαστή του σκάφους.

4.1.1 Δοκιμή Επιτάχυνσης No1

Το σχετιζόμενο με τη δοκιμή αυτή αρχείο είναι το “Verde_2_4_ForMatlab.csv”. Ο αρχικός προγραμματισμός για τη δοκιμή αυτή ήταν να ξεκινήσει το σκάφος από στάση και να κινηθεί ευθεία στην αρχή με τη μισή ρύθμιση πρόωσης και μετά με μέγιστη πρόωση. Εξαιτίας του περιορισμένου χώρου του πεδίου δοκιμών, χρειάστηκε να πραγματοποιηθούν στροφές τύπου U στα όρια της δεξαμενής. Επιπροσθέτως, δεν ήταν δυνατό να χρησιμοποιηθεί πλήρης πρόωση.

Τα διαγράμματα που ακολουθούν παραθέτουν τα αποτελέσματα.



Εικόνα 5. Καταγραφές από τη δοκιμή επιτάχυνσης No1.

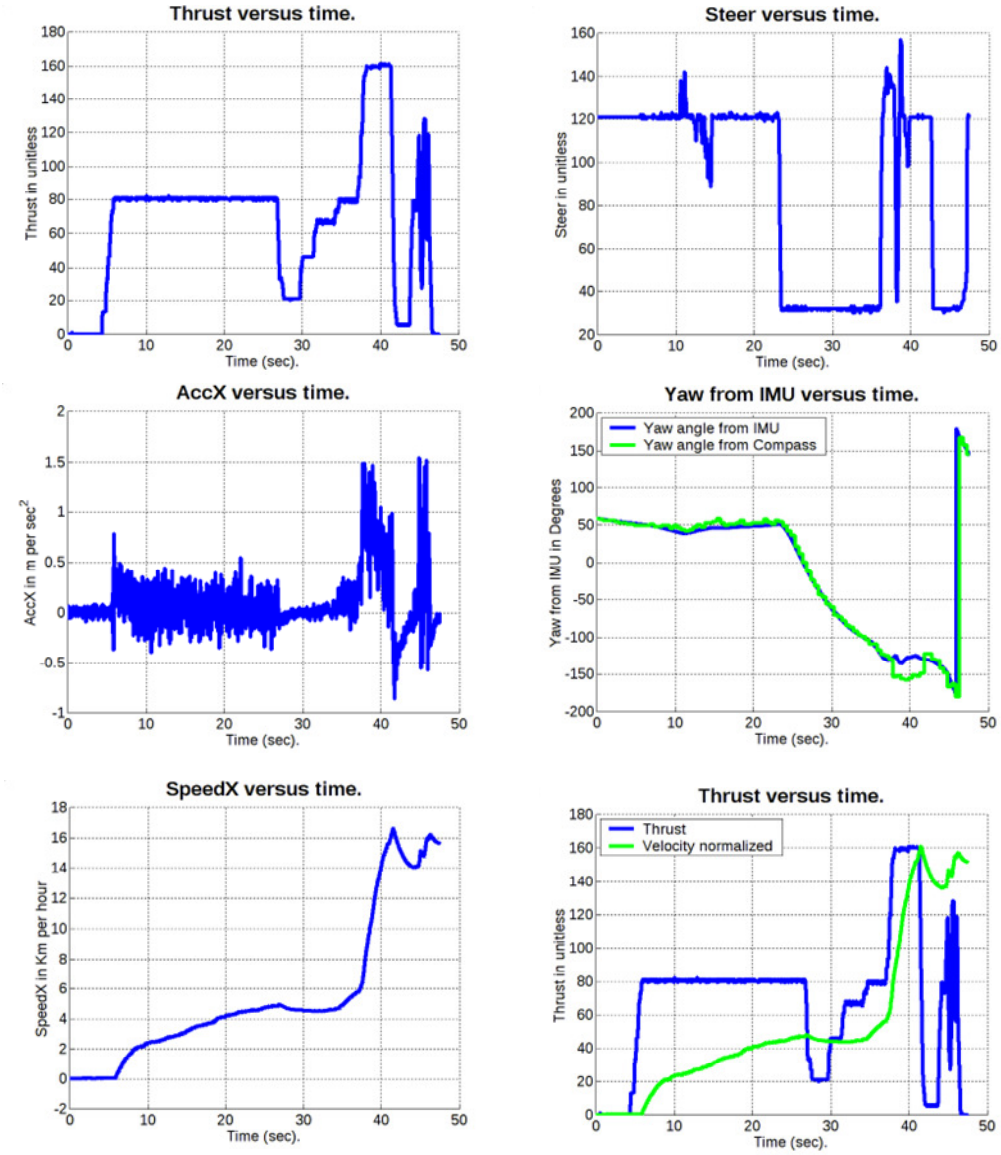
Ένα από τα πράγματα που ήταν αξιοπρόσεκτο στις μετρήσεις μας, ήταν η σχεδόν απόλυτη ταύτιση των ενδείξεων από την ολοκλήρωση των μετρήσεων του

επιταχυνσιομέτρου κατεύθυνσης (yaw), με τις ενδείξεις του μαγνητομέτρου που έβρισκε τη θέση σε σχέση με τον βορρά. Αυτό φαίνεται στο τμήμα "Yaw from IMU versus time" τμήματος της Εικόνα 5.

4.1.2 Δοκιμή Επιτάχυνσης No2

Το σχετιζόμενο αρχείο με τη μέτρηση αυτή είναι το "Verde_2_6_ForMatlab.csv". Σε αυτό το σύνολο μετρήσεων έχουμε καταγράψει την απόκριση του σκάφους όταν χρησιμοποιούμε μισή ώθηση και πλήρη (μετά την αναστροφή στα όρια της λίμνης).

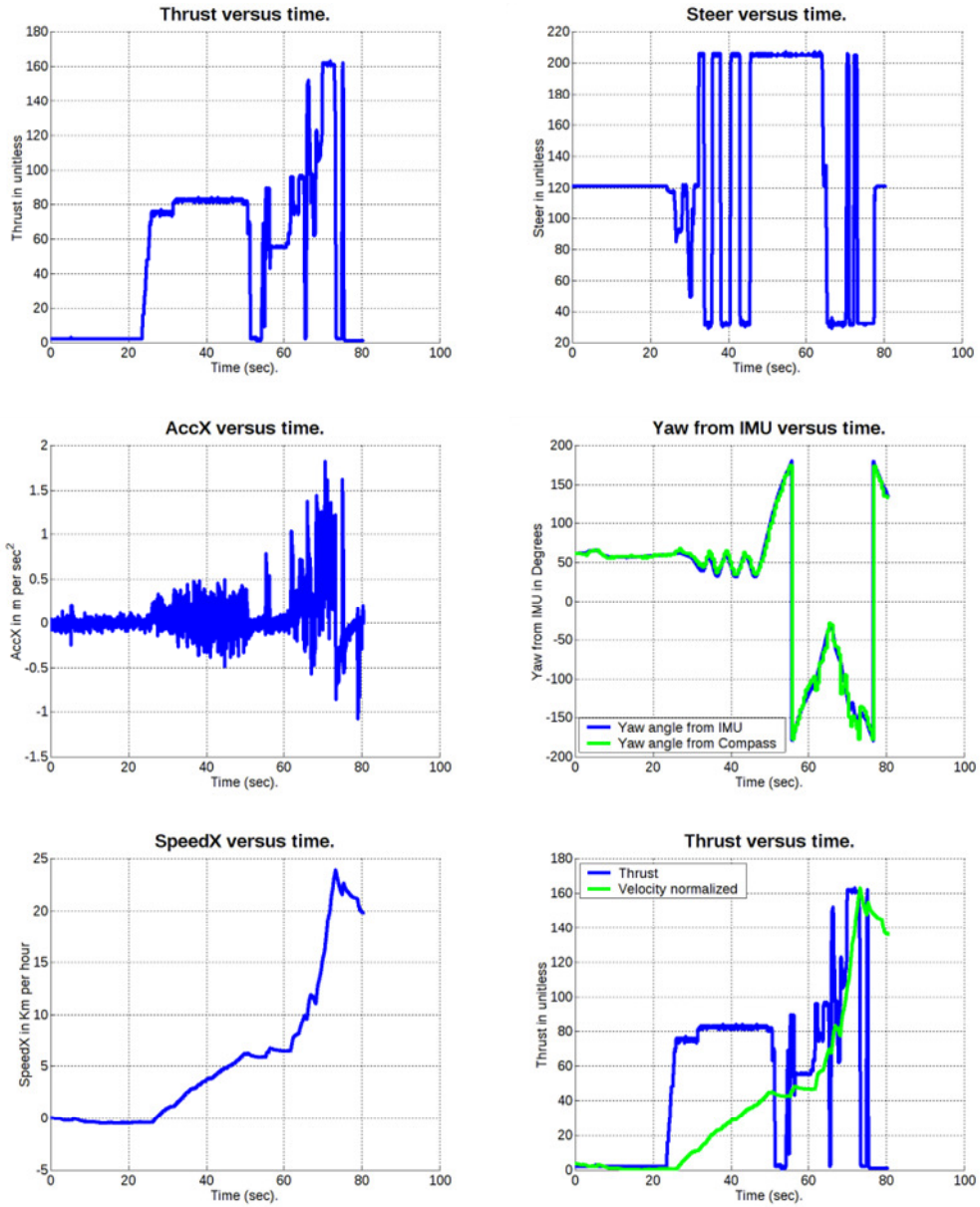
Τα αποτελέσματα παρατίθενται στα παρακάτω διαγράμματα.



Εικόνα 6. Καταγραφές από τη δοκιμή επιτάχυνσης No2.

4.1.3 Δοκιμή Ικανότητας Ελιγμών

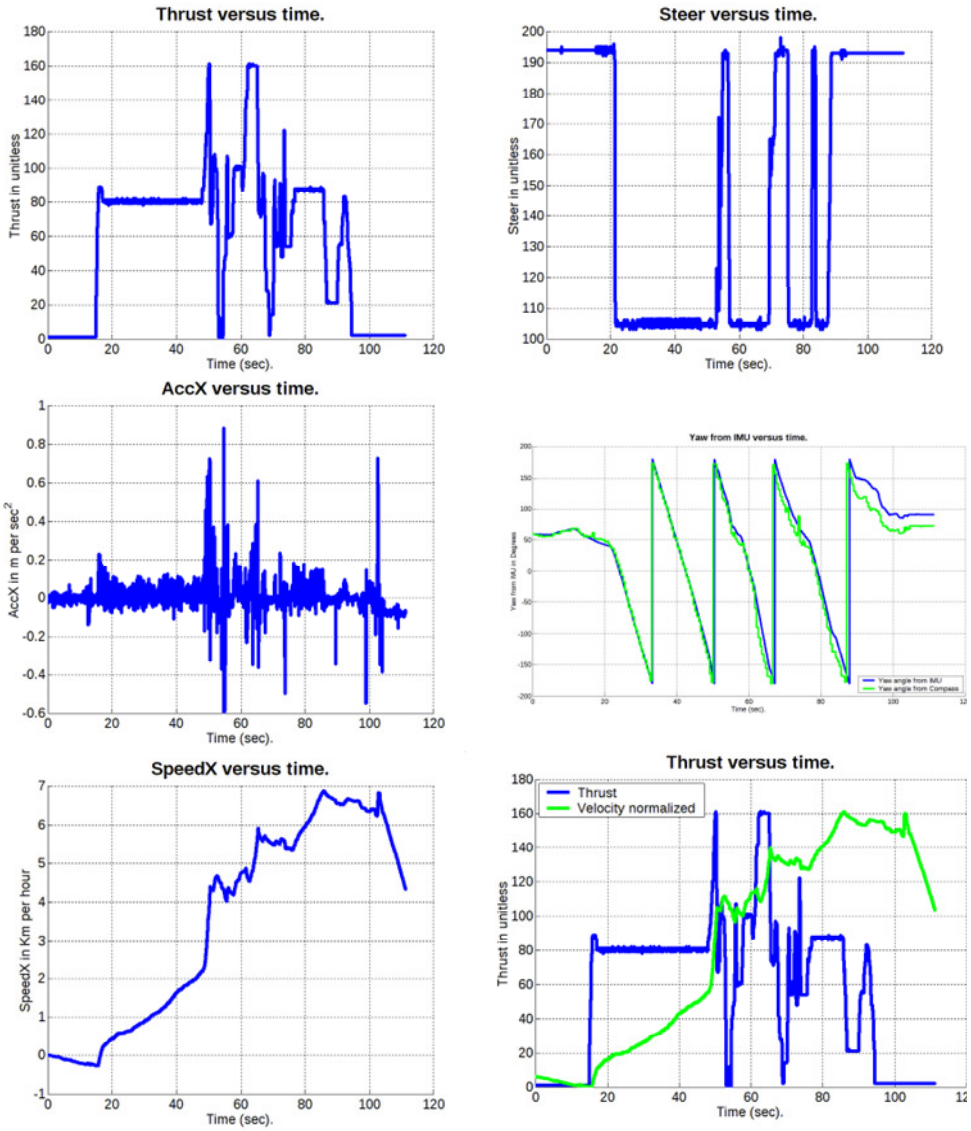
Το αρχείο που σχετίζεται με αυτήν τη μέτρηση είναι "Verde_2_7_ForMatlab.csv". Σε αυτό το σετ μέτρησης καταγράψαμε ελιγμούς τύπου Z.



Εικόνα 7. Καταγραφές από τη δοκιμή ελιγμών τύπου Z.

4.1.4 Δοκιμές Κυκλικής Κίνησης (Ωρολογιακή Φορά)

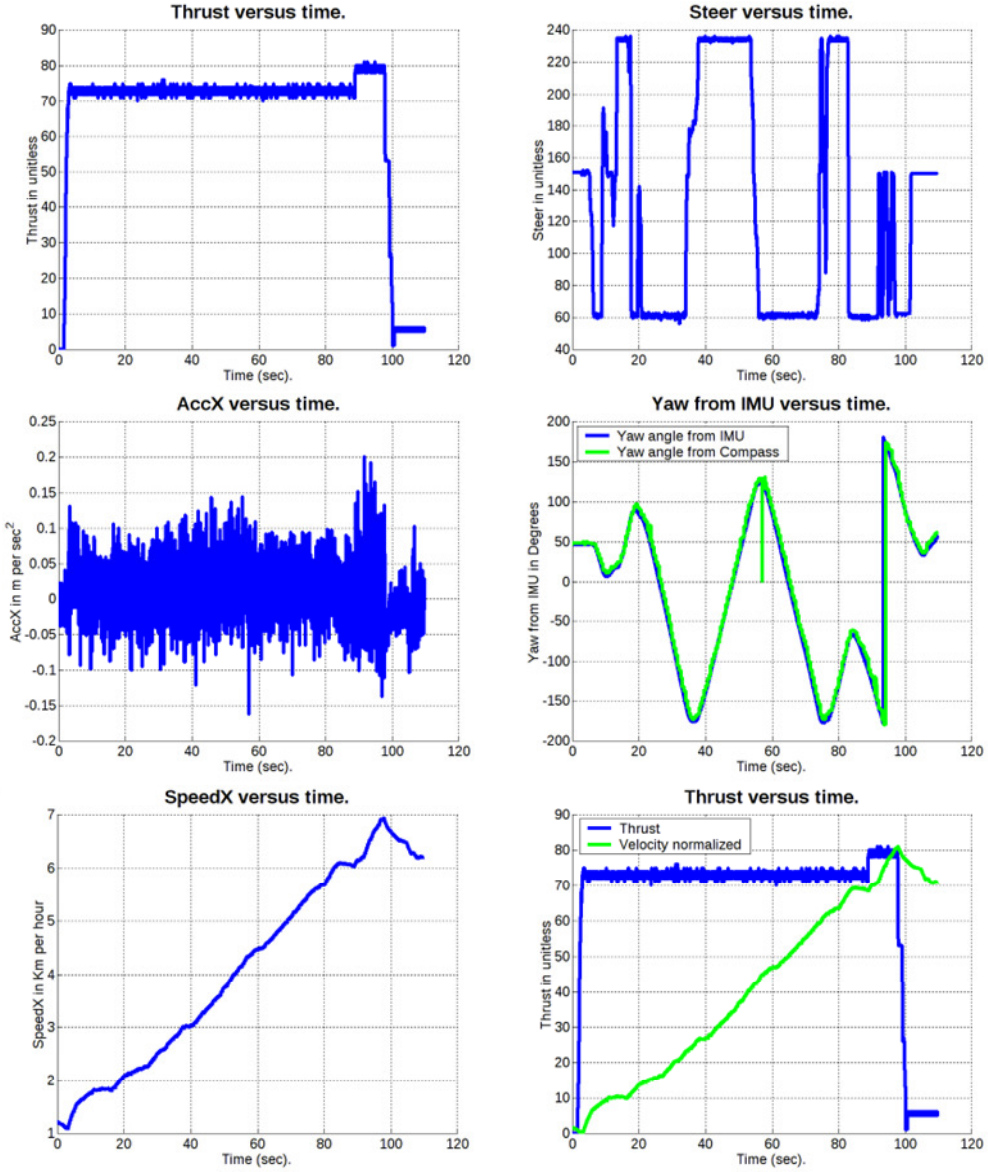
Το αρχείο που σχετίζεται με αυτήν τη μέτρηση είναι το "Verde_2_8_ForMatlab.csv". Σε αυτές τις μετρήσεις καταγράψαμε δεξιόστροφο ελιγμό.



Εικόνα 8. Καταγραφές από τις δοκιμές κυκλικής κίνησης.

4.1.5 Ελιγμοί Τύπου 8

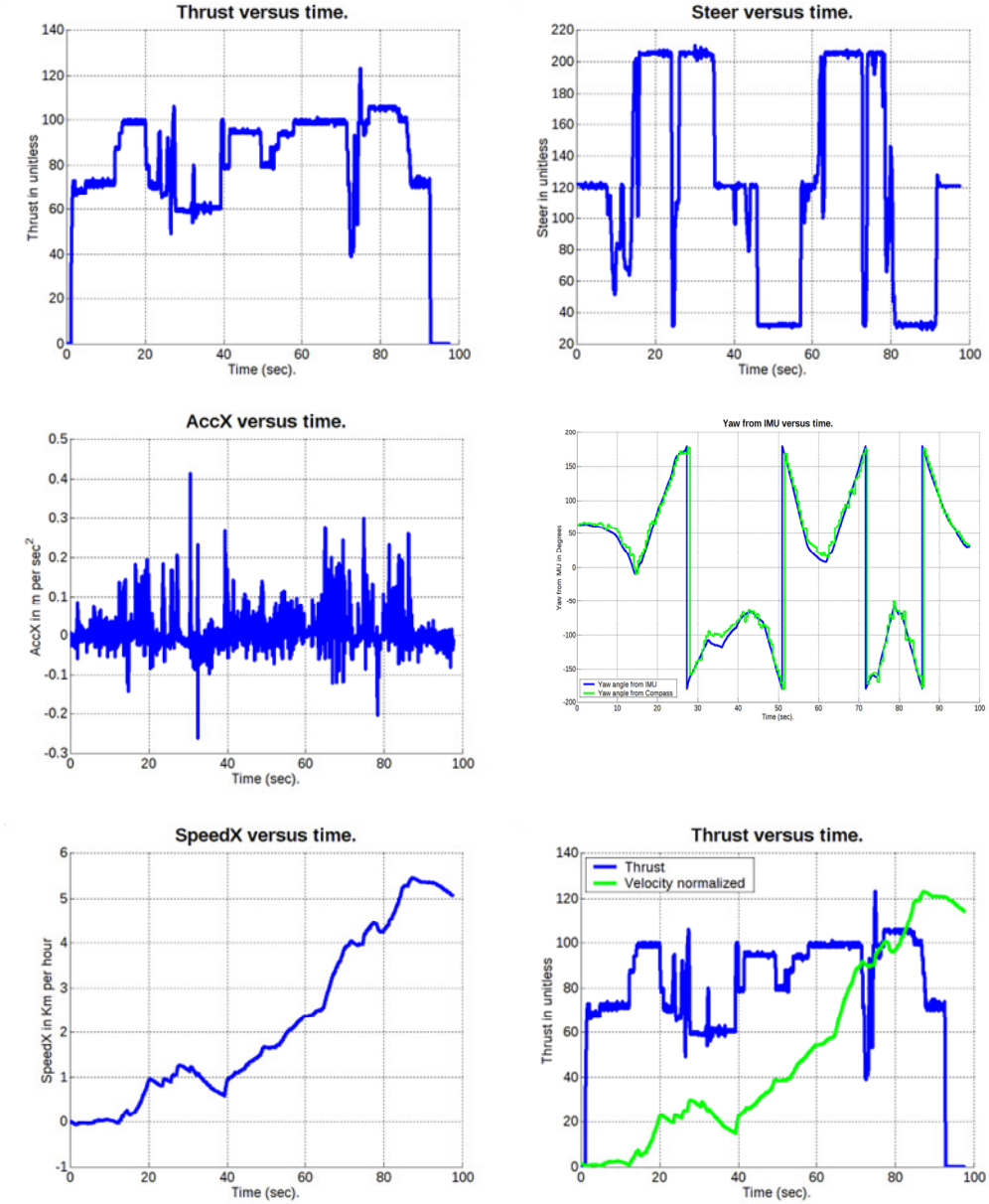
Το αρχείο που σχετίζεται με αυτήν τη μέτρηση είναι το "Verde_2_10_ForMatlab.csv". Σε αυτό το σετ μετρήσεων καταγράψαμε ελιγμούς τύπου 8.



Εικόνα 9. Καταγραφές από τους ελιγμούς τύπου 8.

4.1.6 Δεύτερη Μέτρηση με Ελιγμούς Τύπου 8

Το αρχείο που σχετίζεται με αυτήν τη μέτρηση είναι το “Verde_2_11_ForMatlab.csv”. Σε αυτό το σετ μετρήσεων καταγράψαμε ελιγμούς τύπου 8 για μία ακόμα φορά.



Εικόνα 10. Καταγραφές από τη δεύτερη προσπάθεια ελιγμών τύπου 8.

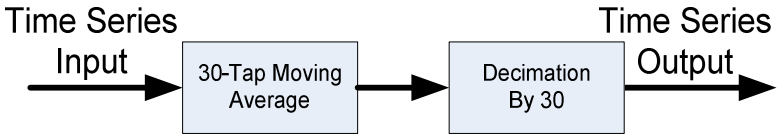
4.2 Επεξεργασία Μετρήσεων

Τόσο για το συνεχές όσο και για το διακριτό μοντέλο, απαιτήθηκε ανάλυση και επεξεργασία σήματος προκειμένου να είναι δυνατή η χρήση των πειραματικών δεδομένων που προέκυψαν από τη μέτρηση, προκειμένου να εκπαιδευθούν τα νευρωνικά δίκτυα που εμφανίζονται στον πυρήνα του μοντέλου [8], [9].

Για το μοντέλο συνεχούς χρόνου απαιτήθηκε να δημιουργηθεί ένα ομαλό σήμα γωνιακής επιτάχυνσης για την αλλαγή πορείας. Η προεπεξεργασία σήματος που χρησιμοποιήθηκε για την απόκτησή του φαίνεται στο Σχήμα 2.

Σχήμα 2. Επεξεργασία σήματος για τη δημιουργία ομαλού σήματος γωνιακής επιτάχυνσης για την εκπαίδευση του νευρωνικού δικτύου συνεχούς χρόνου

Για το μοντέλο διακριτού χρόνου χρειαζόταν να διατηρήσουμε τον αριθμό των εισόδων στα νευρωνικά δίκτυα του πυρήνα στο ελάχιστο. Για το σκοπό αυτό, απαιτήθηκε το decimation μετά από φιλτράρισμα των δεδομένων στις χρονικές κλίμακες που εμφανίζονται στο σύστημα. Σύμφωνα με την τυπική πρακτική, η συχνότητα δειγματοληψίας των μετρήσεων ήταν υψηλός κάτι που επέτρεψε να εξασφαλιστεί η πιστότητα. Το decimation των χρονοσειρών δεδομένων πραγματοποιήθηκε με τη διάδοση των καταγεγραμμένων σημάτων δεδομένων μέσω της αρχιτεκτονικής φιλτραρίσματος που φαίνεται στο Σχήμα 3.



Σχήμα 3. Decimation που εφαρμόστηκε για την ελαχιστοποίηση των δειγμάτων στην είσοδο του νευρωνικού διακριτού χρόνου.

4.3 Μοντελοποίηση με Νευρωνικά Δίκτυα

Για την ταυτοποίηση του σκάφους μέσω νευρωνικών δικτύων αναπτύχθηκαν δύο τύποι νευρωνικών:

Νευρωνικό «Συνεχούς Χρόνου» όπως το ονομάσαμε, με χρήση των νευρωνικών δικτύων

- AccX network
- Wdot network

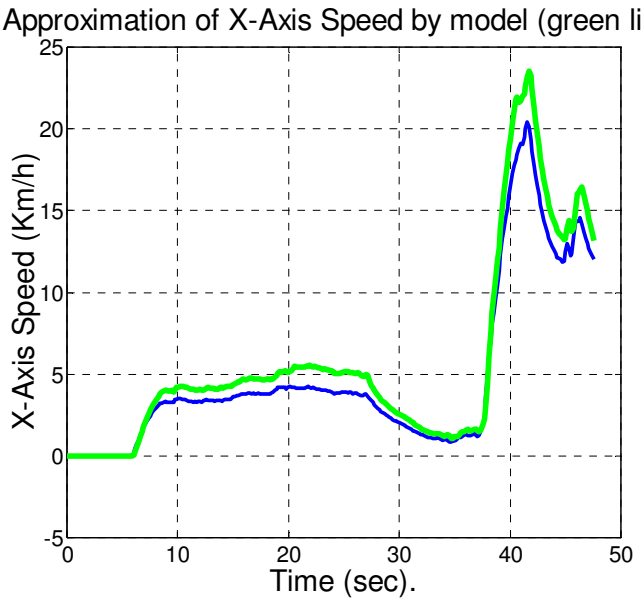
για την πρόβλεψη μεταφορικής (κατά το διαμήκη άξονα) και γωνιακής (yaw) επιτάχυνσης αντιστοίχως.

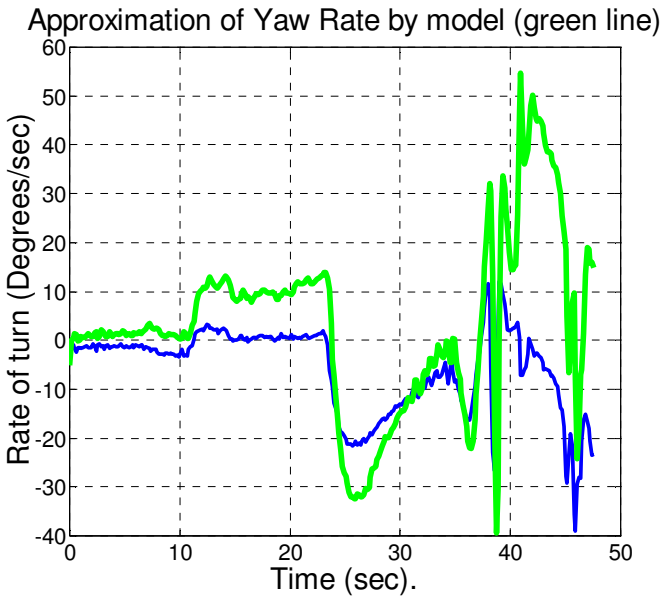
Μοντέλο «διακριτού χρόνου» όπως ονομάστηκε με χρήση των νευρωνικών

- SpeedX
- W

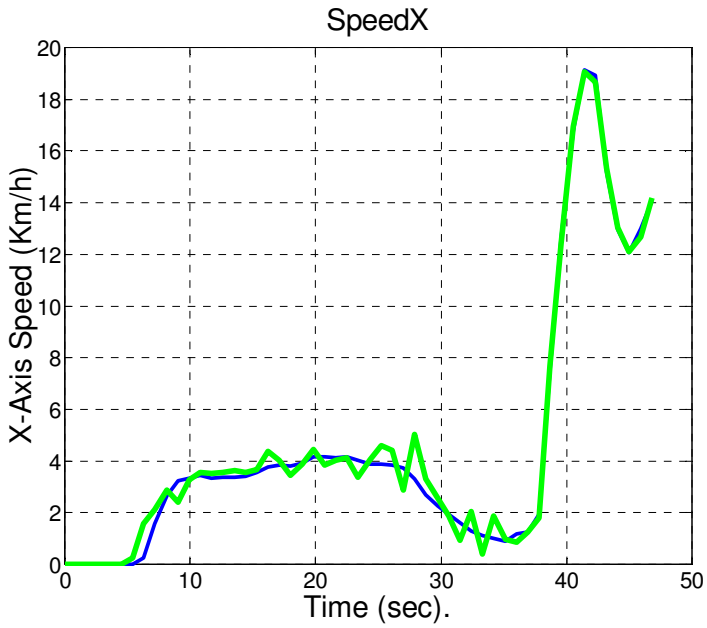
για την πρόβλεψη μεταφορικής και γωνιακής ταχύτητας αυτή τη φορά και όχι επιτάχυνσης.

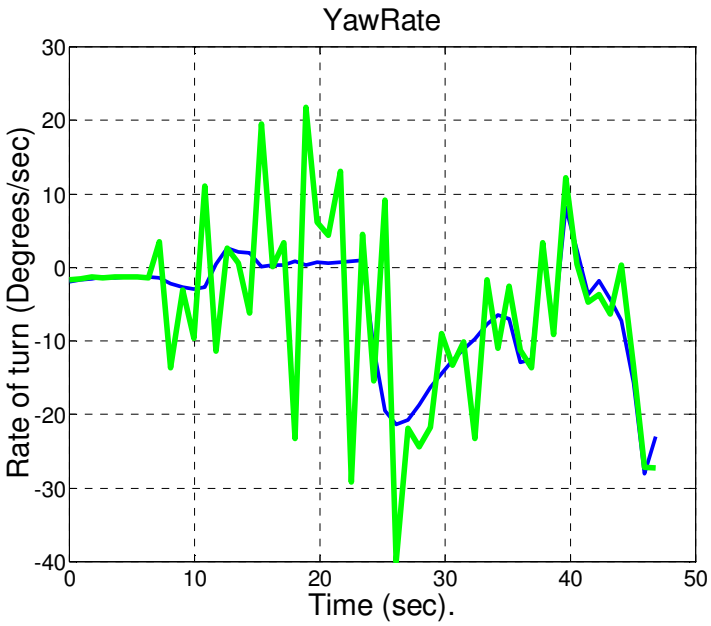
Το αποτέλεσμα της προσέγγισης των κινήσεων του σκάφους από το μοντέλο συνεχούς χρόνου φαίνεται στην παρακάτω εικόνα:





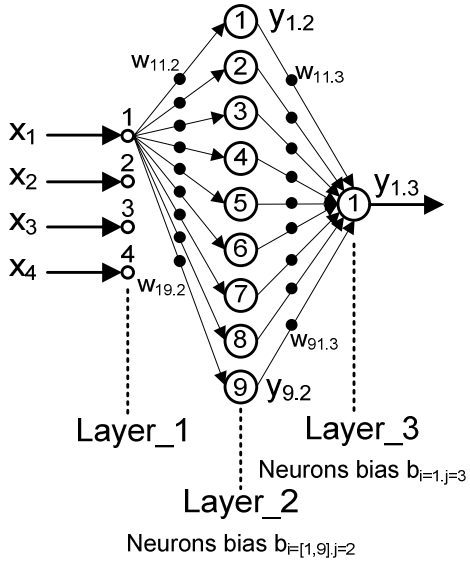
Για την περίπτωση του μοντέλου «διακριτού χρόνου» οι εικόνες που ακολουθούν δίνουν την προσέγγιση του για τις κινήσεις του σκάφους:





Οι παράγραφοι που ακολουθούν έχουν ως στόχο να δώσουν τους μαθηματικούς τύπους που χρησιμοποιούνται για την παραγωγή της πρόβλεψης των νευρωνικών.

4.3.1 Νευρωνικό Τύπου Perceptron



Σχήμα 4. Αποτύπωση ενός νευρωνικού δικτύου τύπου perceptron.

Το νευρωνικό τύπου Perceptron [10], [11] που φαίνεται στην Σχήμα 4. Αποτύπωση ενός νευρωνικού δικτύου τύπου, είναι ακριβώς αυτό που χρησιμοποιήθηκε στη διαδικασία μοντελοποίησης που εφαρμόσαμε για τη σύνθεση του μοντέλου «συνεχούς» χρόνου. Το πρώτο επίπεδο είναι μόνο οι είσοδοι, άρα οι κόμβοι δεν είναι

νευρώνες, απλώς buffers. Οι είσοδοι επισημαίνονται ως xi όπου i=[1,4]. Κάθε είσοδος πολλαπλασιαζόμενη με ένα βάρος (wij.k), συνδέεται σε όλους τους νευρώνες του εσωτερικού επιπέδου 2. Αναφορικά με τους δείκτες κάθε βάρους, i=[1,4] που σημαίνει την είσοδο στην οποία αντιστοιχεί, j=[1,9] και αναφέρεται στο νευρώνα στον οποίο καταλήγει και τέλος k=[1,3] το οποίο συμβολίζει το επίπεδο απόληξης της σύνδεσης.

Κάθε νευρώνας έχει μια έξοδο που ονομάζεται yi.j όπου i είναι ο δείκτης του νευρώνα στο επίπεδο j. Για κάθε νευρώνα υπάρχει επίσης ένα bias bi.j όπου i είναι ο αριθμός του νευρώνα στο επίπεδο j. Έτσι η έξοδος για το δίκτυο στο Σχήμα 17 είναι:

$$
\begin{aligned}
&y_{1.3} = \left(\sum_{i=1}^{9} y_{i.2} w_{i1.3} \right) + b_{1.3} \\
&y_{i.2} = \text{Tansig}\left[\left(\sum_{k=1}^{4} x_k w_{ki.2} \right) + b_{i.2} \right] \\
&\text{which gives:} \\
&y_{1.3} = \left[\sum_{i=1}^{9} \left[\text{Tansig}\left[\left(\sum_{k=1}^{4} x_k w_{ki.2} \right) + b_{i.2} \right] w_{i1.3} \right] \right] + b_{1.3}
\end{aligned}
\tag{4.1}
$$

Μια πολύ σημαντική σημείωση εδώ είναι η ακόλουθη: αν εξετάσετε τις μονάδες των παραμέτρων στην είσοδο του Νευρωνικού Δικτύου, θα διαπιστώσετε ότι (όπως φαίνεται από τον "Πίνακα 1. Singals, units and notations.") γT και δR δεν έχουν μονάδες, αλλά το u είναι σε Km/h και το ω σε rad/sec. Πώς συνδυάζονται όλα αυτά τα σήματα στην εξίσωση του νευρωνικού δικτύου; Μια απάντηση είναι ότι η πραγματική είσοδος στο δίκτυο είναι σήματα γ%, δ%, u% και ω%, τα οποία είναι οι κανονικοποιημένες εκδόσεις των σημάτων και ως εκ τούτου είναι χωρίς μονάδα (έχουν διαιρεθεί με μια σταθερά με τις ίδιες μονάδες). Μια άλλη απάντηση είναι ότι ακόμα κι αν τα σήματα χρησιμοποιήθηκαν ως έχουν, το δίκτυο λειτουργεί ως ένα είδος LUT και ως εκ τούτου οι μονάδες δεν έχουν καμία σημασία για αυτό, εφόσον είναι ίδιες με αυτές που παρουσιάζονται στο δίκτυο τη στιγμή. της εκπαίδευσης. Ωστόσο, η προσωπική μου προτίμηση είναι να χρησιμοποιήσω την κανονικοποιημένη έκδοση χωρίς μονάδες.

4.3.2 Μαθηματικοί Τύποι Συναρτήσεων Νευρώνων

Ανάλογα με τον τύπο του νευρώνα, η συνάρτηση μεταφοράς διαφέρει. Οι τρεις συναρτήσεις μεταφοράς που χρησιμοποιούνται για τους νευρώνες των υλοποιημένων δικτύων είναι:

- Logsig σιγμοειδής συνάρτηση. Ο τύπος για τη logsig είναι:

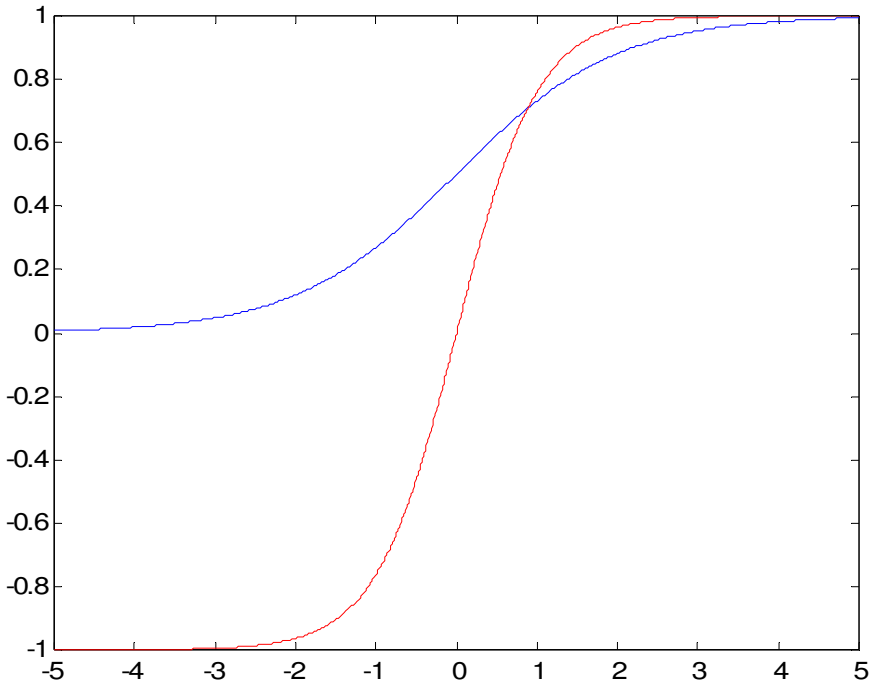
$$
\begin{aligned}
&\text{logsig(n) = 1 / (1 + exp(-n))} \\
&logsig(n) = \frac{1}{1+e^{-n}}
\end{aligned}
\tag{4.2}
$$

- Tansig σιγμοειδής συνάρτηση. Ο τύπος για την tansig είναι:

tansig(n) = 2/(1+exp(-2*n))-1

$$tansig(n) = \frac{2}{1+e^{-n}} - 1 \quad (4.3)$$

- Συνάρτηση Purelin, που είναι απλά μια γραμμική άθροιση.



Σχήμα 5. συνάρτηση logsig (μπλε) σε αντιπαράβολή με την tansig (κόκκινη).

Όπως φαίνεται από τη συνάρτηση logsig έχει μόνο θετική έξοδο, ενώ η συνάρτηση tansig μπορεί επίσης να εξάγει και αρνητικές τιμές.

4.3.3 Παράγωγοι Συναρτήσεων Νευρώνων

4.3.3.1 ΠΑΡΑΓΩΓΟΣ ΣΥΝΑΡΤΗΣΗΣ LOGSIG

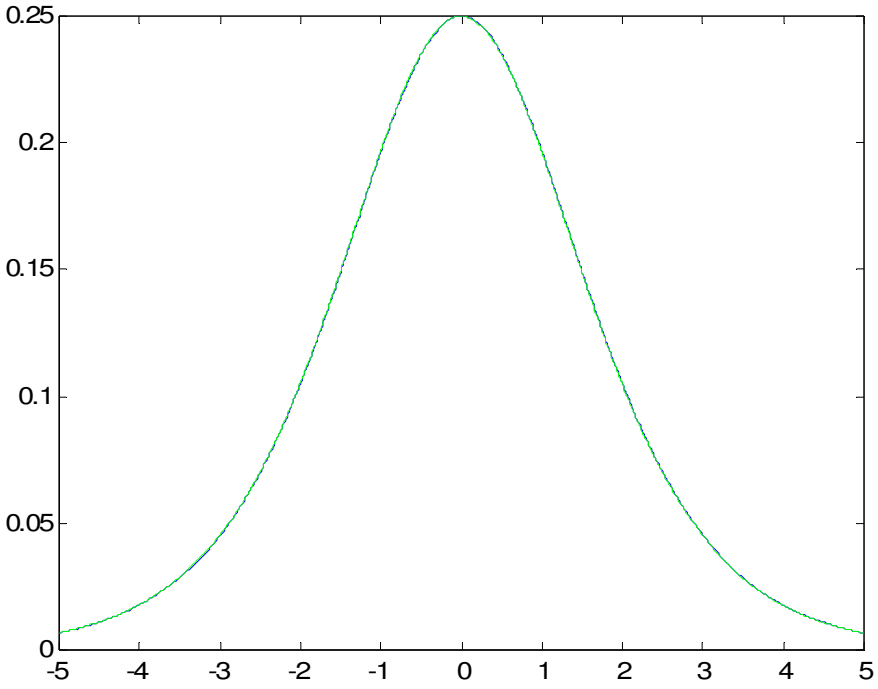
Η παράγωγος της συνάρτησης logsig είναι η εξής:

$$\frac{d}{dn} logsig(n) = \frac{e^{-n}}{(1+e^{-n})^2} \quad (4.4)$$

Η λειτουργία έχει επαληθευτεί στο Matlab με n = [-5:0.01:5] χρησιμοποιώντας διαφορές των επόμενων δειγμάτων διαιρεμένες με το βήμα δειγμάτων ως εξής:

$$Deriv(i) = \frac{logsig(n(i+1)) - logsig(n(i))}{n(i+1) - n(i)} \quad (4.5)$$

Τα αποτελέσματα φαίνονται στην Σχήμα 6.



Σχήμα 6. Επαλήθευση της παραγώγου logsig.

Η έξοδος της παραγώγου της συνάρτησης είναι με μπλε χρώμα, ενώ η προσέγγιση της διαφοράς είναι με πράσινο. Η μέγιστη απόκλιση μεταξύ των δύο είναι 4.8112e-004.

Το αρχείο με τον κώδια Matlab που χρησιμοποιήθηκε είναι στο E:\Work\PHD\MatlabCodes\logsig_der.m.

4.3.3.2 TANSIG FUNCTION DERIVATIVE

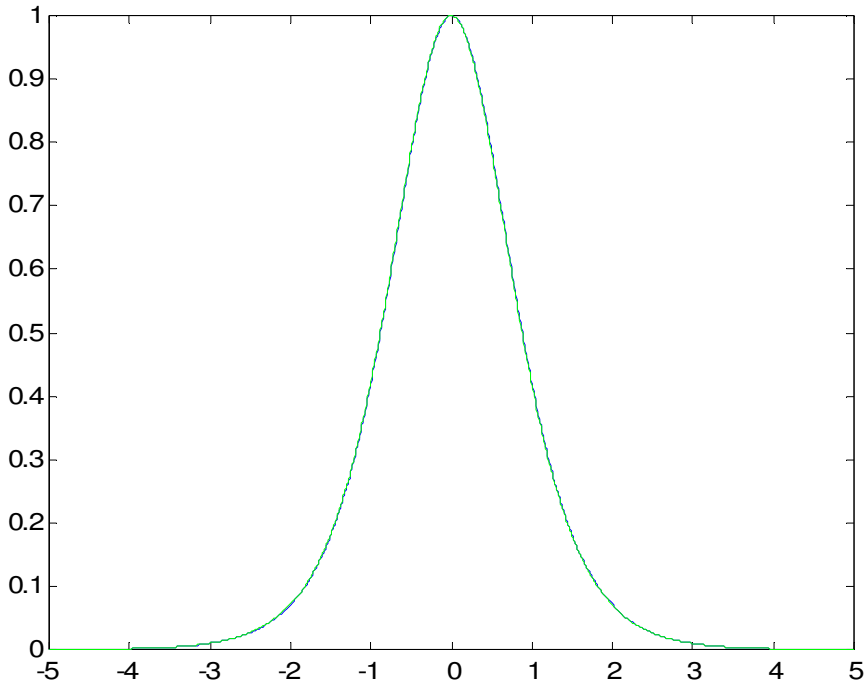
Η παράγωγος της συνάρτησης tansig είναι η εξής:

$$\frac{d}{dn}tansig(n) = \frac{4e^{-2n}}{(1+e^{-2n})^2} \quad (4.6)$$

Η λειτουργία έχει επαληθευτεί στο Matlab με n = [-5:0.01:5] χρησιμοποιώντας διαφορές των επόμενων δειγμάτων διαιρεμένες με το βήμα δειγμάτων ως εξής:

$$Deriv(i) = \frac{tansig(n(i+1)) - tansig(n(i))}{n(i+1) - n(i)} \quad (4.7)$$

Τα αποτελέσματα παρουσιάζονται στην Σχήμα 7.



Σχήμα 7. Επιβεβαίωση παραγώγου tansig.

Η έξοδος της παραγώγου της συνάρτησης είναι με μπλε χρώμα, ενώ η προσέγγιση της διαφοράς είναι με πράσινο. Η μέγιστη απόκλιση μεταξύ των δύο είναι 0.0038.
Ο κώδικας που χρησιμοποιήθηκε για τη δοκιμή είναι στο αρχείο E:\Work\PHD\MatlabCodes\tansig_der.m.

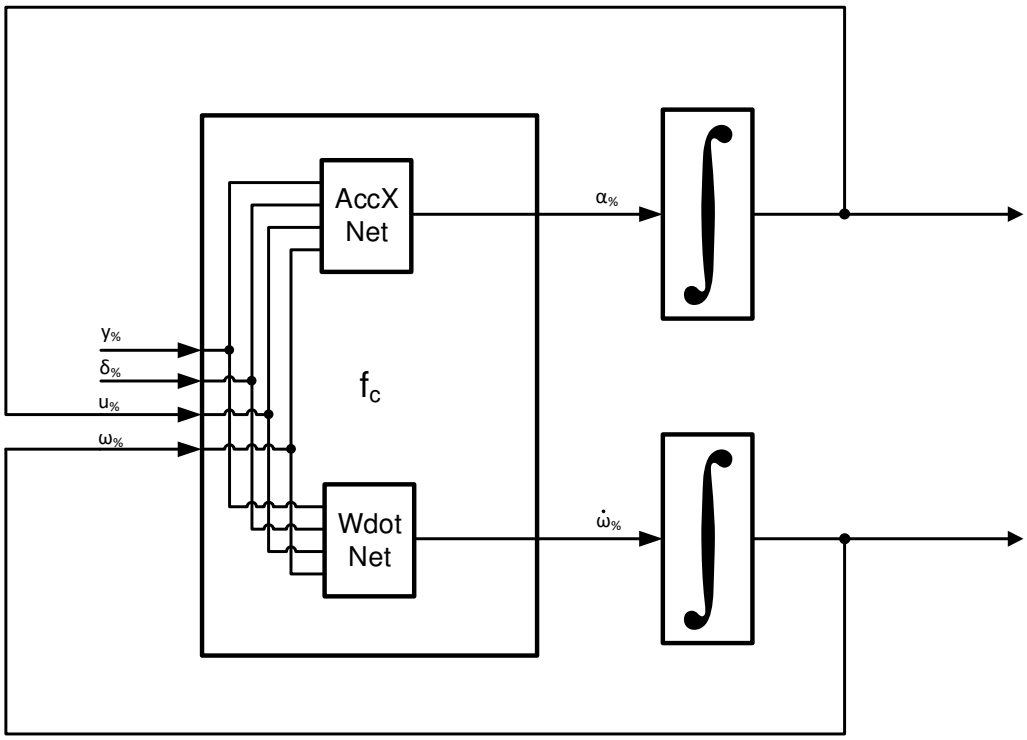
5 Σύνθεση του Πλήρους Μοντέλου και Ελεγκτές

5.1 Εισαγωγή στην Ενότητα

Μετά την αξιολόγηση της μεθόδου χρήσης εξισώσεων στη θέση της υλοποίησης των νευρωνικών δικτύων Matlab, πρέπει να εφαρμοστεί και να αξιολογηθεί το πλήρες μοντέλο των διασυνδεδεμένων νευρωνικών δικτύων χρησιμοποιώντας τα δεδομένα από τα πειράματα. Αυτή η ενότητα περιγράφει τη διαδικασία και την εσωτερική δομή του μοντέλου που χρησιμοποιήσαμε.

5.2 Πλήρες Μοντέλο Σκάφους

Όπως αναφέρεται στην παράγραφο 4.2.1, για τη μοντελοποίηση του σκάφους χρησιμοποιούνται δύο νευρωνικά δίκτυα. Ένα που εξάγει δείγματα μεταφορικής επιτάχυνσης και ένα που εξάγει δείγματα γωνιακής επιτάχυνσης. Η έξοδος κάθε δικτύου ολοκληρώνεται στο χρόνο με τον επιλεγμένο αλγόριθμο ολοκλήρωσης (βλ. ενότητα 5.5.2.1) και ανατροφοδοτείται και στα δύο νευρωνικά δίκτυα, προκειμένου να δημιουργηθεί το επόμενο δείγμα όπως φαίνεται στο Διάγραμμα 1 [12].



Διάγραμμα 1. Το πλήρες μοντέλο σκάφους μέσω νευρωνικών.

Το σύστημα των διασυνδεδεμένων νευρωνικών δικτύων μαζί με τους ολοκληρωτές είναι αυτό που χρησιμοποιείται σε όλες τις εξομοιώσεις για την απόκριση του σκάφους. Χρησιμοποιείται επίσης από τον ελεγκτή MBC, όπως θα παρουσιαστεί στην παράγραφο 5.5.1, για τη λήψη απόφασης σχετικά με την καταλληλότερη ενέργεια ελέγχου που πρέπει να χρησιμοποιηθεί.

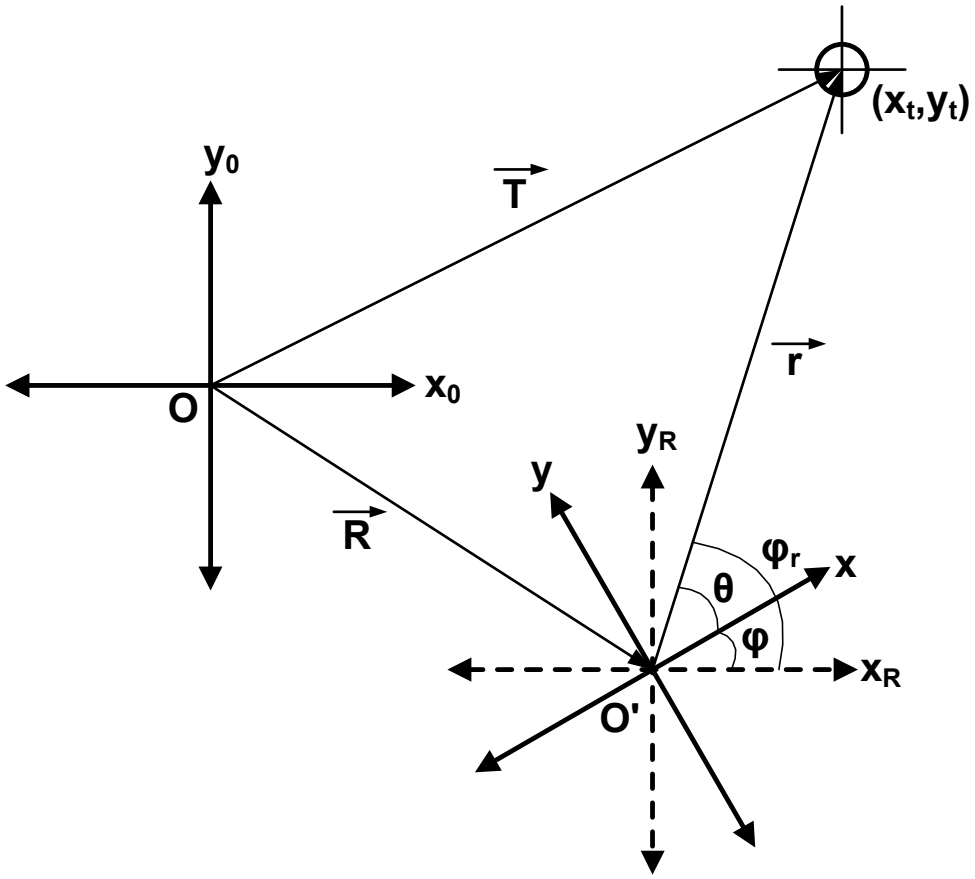
Αν και το μοντέλο ονομάζεται "Συνεχούς χρόνου" στην ενότητα 4.2.1, ο διακριτός χρόνος είναι κρυμμένος στην προσέγγιση ολοκλήρωσης. Αυτό το γεγονός επιτρέπει τη χρήση αυτής της μοντελοποίησης σε περιβάλλοντα προσομοίωσης διακριτού χρόνου.

Πρέπει επίσης να σημειωθεί ότι τα νευρωνικά δίκτυα έχουν εκπαιδευτεί και με δεδομένα διακριτού χρόνου, που δημιουργήθηκαν κάθε Ts, δηλαδή κάθε 30 ms.

5.3 Δομή Αλγορίθμων Ελέγχου

Πριν περιγραφούν οι αλγόριθμοι ελέγχου, είναι απαραίτητο να παρουσιαστούν οι μεταβλητές που χρησιμοποιούνται. Η ακόλουθη υποενότητα παρουσιάζει τον τρόπο με τον οποίο αξιολογούνται οι θέσεις και οι ταχύτητες, καθώς και τα συστήματα αξόνων αναφοράς που απαιτούνται για το σκοπό αυτό.Equation Section (Next)

5.3.1 Συστήματα Αξόνων Αναφοράς



Σχήμα 8. Συστήματα αξόνων αναφοράς κινήσεων.

Αναφερόμενοι στο Σχήμα 8, έχουμε τρία συστήματα αξόνων:

- Ox_0y_0
- $O'x_Ry_R$
- O'xy

Το Ox_0y_0 είναι το σταθερό σύστημα αναφοράς του δισδιάστατου χώρου όπου υπάρχουν τα σημεία του σκάφους και του στόχου.

Το $O'x_Ry_R$ είναι το σύστημα αξόνων που δένεται στη θέση του κέντρου μάζας του σκάφους και είναι παράλληλο στο Ox_0y_0. Το διάνυσμα $\vec{R}$ συνδέει το σημείο O με το O' (ουσιαστικά $\vec{R} = \overrightarrow{OO'}$).

O'xy είναι το σωματόδετο σύστημα αναφοράς [13] το οποίο ακολουθεί τις αλλαγές πορείας του σκάφους, ορίζοντας ως άξονα x τον διαμήκη άξονα του σκάφους, με τον θετικό ημιάξονα να εκτείνεται από το κέντρο μάζας του σκάφους προς την πλώρη.

Ο παρακάτω πίνακας συνοψίζει όλους τους όρους που απαιτούνται για την κατανόηση των συστημάτων αναφοράς:

Κατάλογος Συμβόλων Συστημάτων Αξόνων Αναφοράς	
Σύμβολο	**Ορισμός**
Ox_0y_0	Σταθερό σύστημα αναφοράς για τον δισδιάστατο χώρο στον οποίο μελετάται η κίνηση του σκάφους και οι θέσεις των στόχων.
$O'x_Ry_R$	Σύστημα αναφοράς παράλληλο στο Ox_0y_0, το οποίο ακολουθεί το κέντρο μάζας του σκάφους.
O'xy	Body-fixed reference frame which follows the vessel's angle of turn
$\vec{R}$	Διάνυσμα που ενώνει το O με το O'.
$\vec{r}$	Διάνυσμα που συνδέει το O' με τον στόχο.
$\vec{T}$	Διάνυσμα που συνδέει το O με τον στόχο.
(x_t, y_t)	Συντεταγμένες στόχου με αναφορά στο Ox0y0.
θ	Γωνία μεταξύ πλώρης σκάφους και διανύσματος . Θετική ανθωρολογιακά.
φ	Γωνία μεταξύ πλώρης σκάφους και άξονα xR. Θετική ανθωρολογιακά.
φ_r	Γωνία μεταξύ και xR ή x0 άξονα. Θετική ανθωρολογιακά.

Αυτά τα σύμβολα θα χρησιμοποιηθούν στις παραγράφους που ακολουθούν, μαζί με τα ακόλουθα:

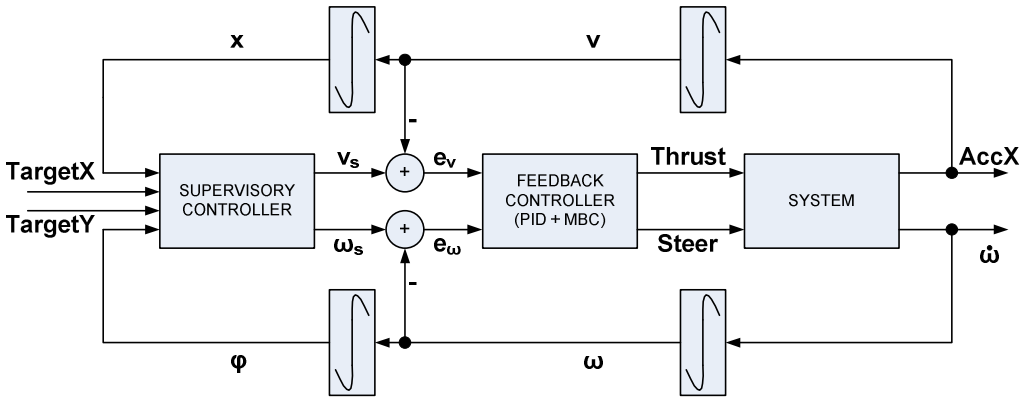
Section Symbols List (repeated from section "1.3 Σύμβολα, Μεταβλητές και Συντομογραφίες")	
Σύμβολο	**Ορισμός**
y_T	Κανόνας πρόωσης προς μηχανές σκάφους (γκάζι).
δ_R	Κανόνας στροφής.
y_{TA}	Auto-generated thrust command.
δ_{RA}	Auto-generated steering command.
u	Translational speed of the vessel in Km/h.
ω	Angular speed of the vessel in °/sec.
α	Translational acceleration of vessel ($=\dot{u}$) in m/(sec)^2.
$\dot{\omega}$	Angular acceleration in °/(sec)^2.
T_s	Sampling rate for all operations of the system. The sampling rate used is 0.03s.

Για να βοηθηθεί η κατανόηση των ακόλουθων ενοτήτων προστίθενται ορισμένες επεξηγήσεις συντομογραφιών-συντμήσεων:

Λίστα Συντομογραφιών	
Σύμβολο	**Ορισμός**
μC	Μονάδα μικροελεγκτή. Αναφέρεται στο αναπτυξιακό σύστημα, το οποίο είναι το κέντρο λήψεως αποφάσεων του σκάφους.
SC	Εποπτικός Ελεγκτής (Supervisory Controller)
FC	Ελεγκτής Ανατροφοδότησης (Feedback Controller)
MBC	Model Based Controller
Zoh	Zero-order hold

5.3.2 Επισκόπηση του Τρόπου Ελέγχου του Συστήματος

Για τον έλεγχο της θέσης του σκάφους σε έναν διδιάστατο χώρο, αποφασίστηκε το ακόλουθο σχήμα ελέγχου:



Διάγραμμα 2. Πλήρες μοντέλο ελέγχου συστήματος.

Αυτό το διάγραμμα είναι μια επισκόπηση του πραγματικού σχήματος ελέγχου. Τα κρίσιμα μέρη θα αναλυθούν περαιτέρω στις ενότητες που ακολουθούν.

Σα σύστημα ο Supervisory Controller (SC) παίρνει ως είσοδο τη θέση του στόχου στο σύστημα αξόνων O'xy, το οποίο είναι το σωματόδετο σύστημα αναφοράς. Ο στόχος του είναι να μηδενίσει το μέτρο του διανύσματος $\vec{r}$ το οποίο συνδέει το στόχο με το κέντρο μάζας του σκάφους. Για να το επιτύχει αυτό, ο SC δημιουργεί επιθυμητές τιμές ταχύτητας τόσο μεταφορικής όσο και γωνιακής.

Η έξοδος SC λαμβάνεται στη συνέχεια από τον ελεγκτή ανάδρασης (FC), ο οποίος λειτουργεί παραγωγός χειρισμών ώθησης και στροφής για το σύστημα, προκειμένου να αναγκάσει το σκάφος να επιτύχει τις απαιτούμενες ταχύτητες.

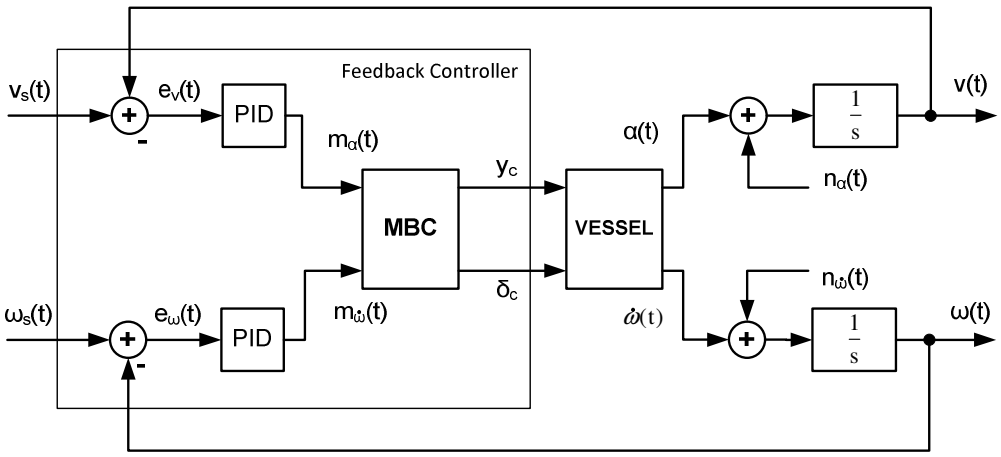
Οι έξοδοι του FC λαμβάνονται από το σύστημα, το οποίο με τη σειρά του αποκρίνεται με έξοδο μεταφορικής επιτάχυνσης και έξοδο γωνιακής επιτάχυνσης.

5.4 Feedback Controller (FC)

Η μονάδα FC αποτελείται από τα ακόλουθα μέρη:

- PIDu: Ελεγκτής PID για τη ρύθμιση της μεταφορικής ταχύτητας του σκάφους (u).
- PIDω: Ελεγκτής PID για τη ρύθμιση της γωνιακής ταχύτητας αλλαγής πορείας του σκάφους (u).
- MBC: Model Based Controller που λαμβάνει ως είσοδο τα αιτήματα από τους δύο ελεγκτές PID που αναφέρονται παραπάνω και αναζητά τις καταλληλότερες εντολές ώθησης και διεύθυνσης για το σκάφος, προκειμένου να παράγει τις ζητούμενες επιταχύνσεις από τους PID.

Το MBC λειτουργεί επίσης ως αποζεύκτης για το αίτημα επιταχύνσεων και έτσι οι βρόχοι ανάδρασης μπορούν να διαχωριστούν και να αντιμετωπιστούν ως SISO. Τυχόν ανακρίβειες στις επιταχύνσεις του συστήματος και στην απόζευξη που επιτυγχάνει το MBC προστίθενται ως θόρυβοι στις επιταχύνσεις εξόδου του συστήματος. Το Διάγραμμα 3 αποτυπώνει σχηματικά τα ανωτέρω.



Διάγραμμα 3. Ελεγκτής Ανατροφοδότησης (FC) και η λειτουργία του στο σύστημα.

5.4.1 Model Based Controller (MBC)

Ο σκοπός του ελεγκτή MBC είναι να λειτουργεί ως συσκευή που θα έχει τη λειτουργία αντίστροφης συνάρτησης μεταφοράς του σκάφους, έτσι ώστε να περνάει τα σήματα ρύθμισης απευθείας στην έξοδο του συστήματος. Εκτός από την αναστροφή συστήματος, αποσυνδέει τις αιτήσεις για ταχύτητα και γωνιακή ταχύτητα, για να επιτρέψει τη χρήση ξεχωριστών ελεγκτών PID για καθεμία από τις δύο μεταβλητές.
Η πραγματική υλοποίηση του MBC δεν μπορεί να εκτελέσει τα καθήκοντά της 100%. Για το λόγο αυτό, οι ανακρίβειες στις ζητούμενες εξόδους μαζί με την ανεπάρκεια αποσύνδεσης μοντελοποιούνται ως θόρυβος σε μεταφορικές και γωνιακές επιταχύνσεις. Για την περιστολή αυτού του θορύβου, τα PID που δημιουργούν αιτήματα ενεργειών ελέγχου στο MBC έχουν σχεδιαστεί για ελαχιστοποίηση της H_∞ της συνάρτησης μεταφοράς από το θόρυβο στο σφάλμα.

Ο MBC λειτουργεί λαμβάνοντας ως εισόδους:

- Την έξοδο του PID για επιθυμητή μεταφορική επιτάχυνση
- Την έξοδο του PID για επιθυμητή γωνιακή επιτάχυνση
- Την τρέχουσα μεταφορική ταχύτητα.
- Την τρέχουσα γωνιακή ταχύτητα.

Εκμεταλλευόμενος τα μοντέλα νευρωνικών δικτύων του σκάφους για πρόγνωση μεταφορικών και γωνιακών επιταχύνσεων, χρησιμοποιεί την τρέχουσα τιμή μεταφορικής ταχύτητας και την τιμή της τρέχουσας γωνιακής ταχύτητας ως σταθερές και παράγει όλες τις πιθανές εξόδους για μεταφορικές και γωνιακές επιταχύνσεις μεταβάλλοντας τις εισόδους ώθησης και διεύθυνσης των νευρωνικών δικτύων. Δεδομένου ότι οι τιμές ώθησης και διεύθυνσης είναι διακριτές και περιορισμένες (0 έως 255 για την ώθηση και 0 έως 249 για τη διεύθυνση), το ενδιάμεσο αποτέλεσμα του MBC είναι ένας πίνακας στοιχείων 256 x 250 πιθανών διανυσματικών αποτελεσμάτων, όπου το πρώτο στοιχείο του διανύσματος είναι η μεταφορική

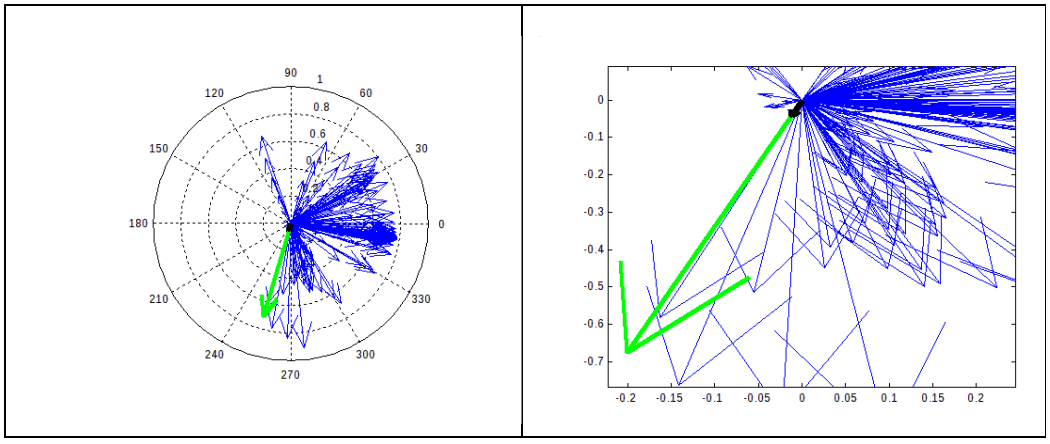
επιτάχυνση και το δεύτερο η γωνιακή επιτάχυνση. Όλα τα πιθανά διανύσματα ανήκουν σε ένα διδιάστατο επίπεδο «φάσης» όπου στον άξονα x είναι οι πιθανές τιμές μεταφορικής επιτάχυνσης και στον άξονα y οι πιθανές τιμές γωνιακής επιτάχυνσης. Η λέξη «φάση» βρίσκεται σε εισαγωγικά γιατί οι επιταχύνσεις δεν είναι καταστάσεις του συστήματος (οι αντίστοιχες ταχύτητες είναι..). Για το υπόλοιπο της παραγράφου αυτό το διδιάστατο επίπεδο θα ονομάζεται επίπεδο επιταχύνσεων.

Ο MBC έχει δύο τρόπους λειτουργίας, που επιλέγονται από τις παραμέτρους προγραμματισμού του:

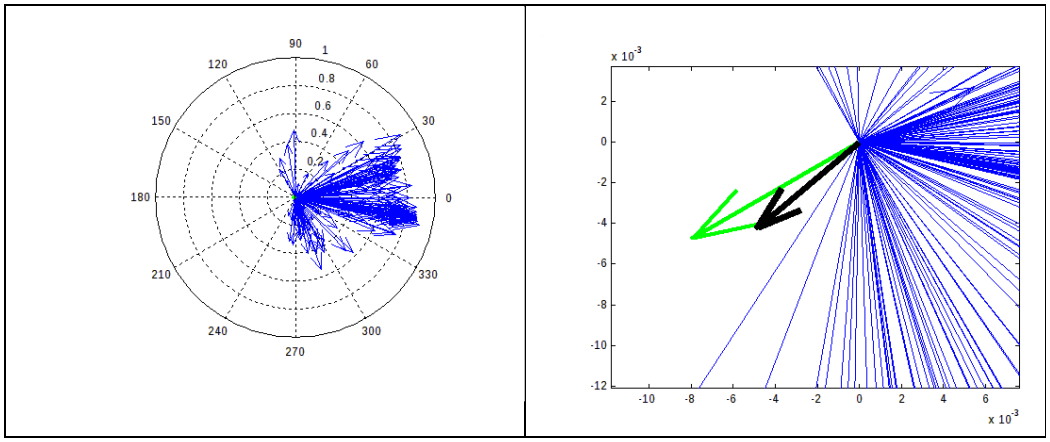
- Στη λειτουργία «BestAngleOnly», αναζητά στο επίπεδο επιταχύνσεων το διάνυσμα που ταιριάζει καλύτερα στη γωνία του διανύσματος επιταχύνσεων που ζητήθηκε. Αυτός ο τρόπος επιλογής δημιουργεί έναν θόρυβο σχετικά με το μέγεθος των εξόδων επιταχύνσεων, αλλά αυτός ο θόρυβος εξασθενείται από την πολύ συχνή επανεκτίμηση νέων επιθυμητών διανυσμάτων (κάθε 30 ms), τις συναρτήσεις μεταφοράς των ελεγκτών PID και γενικά την επιμονή του εποπτικού ελεγκτή για να κρατήσει το σκάφος σε μια πορεία προς το σημείο στόχο του. Είναι επίσης η πρώτη λειτουργία που λειτούργησε με επιτυχία.

- Στη λειτουργία 'BestAngleAndMag', η απόφαση για το καλύτερο διάνυσμα αξιολογείται μεταξύ ενός συνόλου πιθανών διανυσμάτων εξόδου που έχουν αποδεκτές διαφορές γωνίας σε σχέση με τις ζητούμενες από τους ελεγκτές PID. Η καλύτερη αντιστοίχιση είναι αυτή που έχει το μέγεθος πιο κοντά στο ζητούμενο, μεταξύ του συνόλου των διανυσμάτων εγκεκριμένης γωνίας. Αυτή η λειτουργία είναι μια λογική βελτίωση της προηγούμενης, επιστρέφοντας καλύτερα αποτελέσματα όπως θα φανεί στις ακόλουθες παραγράφους.

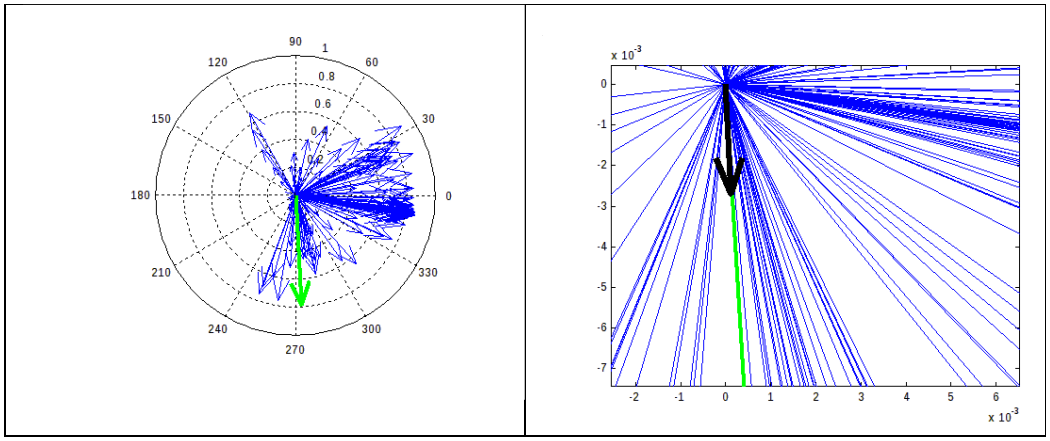
Σε γραφική αναπαράσταση, η λειτουργία των δύο επιλογών του MBC φαίνεται στα σχήματα που ακολουθούν. Τα μπλε βέλη είναι τα εφικτά στοιχεία από το μοντέλο. Το μαύρο βέλος είναι το ζητούμενο από τα PID και το πράσινο το επιλεγμένο. Στα σχήματα που ακολουθούν, το δεξί είναι μια μεγεθυσμένη έκδοση του αριστερού.



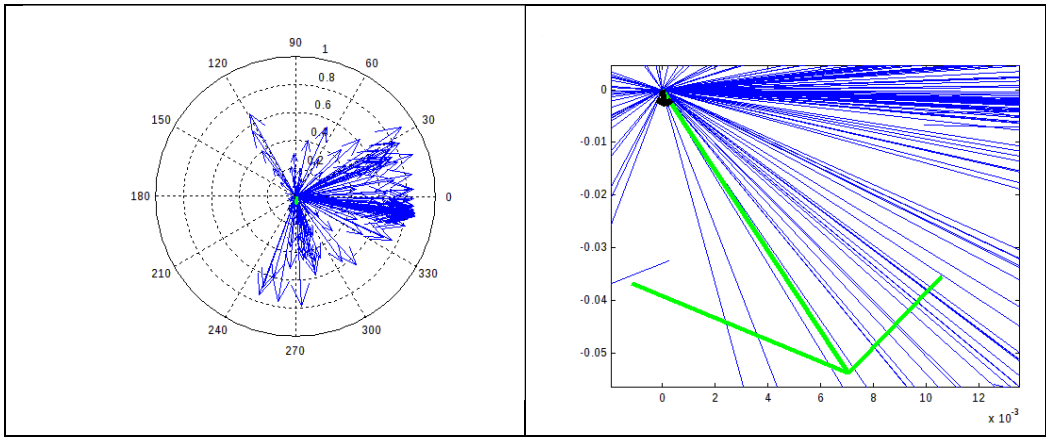
Σχήμα 9. 'BestAngleOnly' μέθοδος επιλογής κατάλληλου διανύσματος εξόδου (πράσινο) σε σχέση με αίτημα (μαύρο).



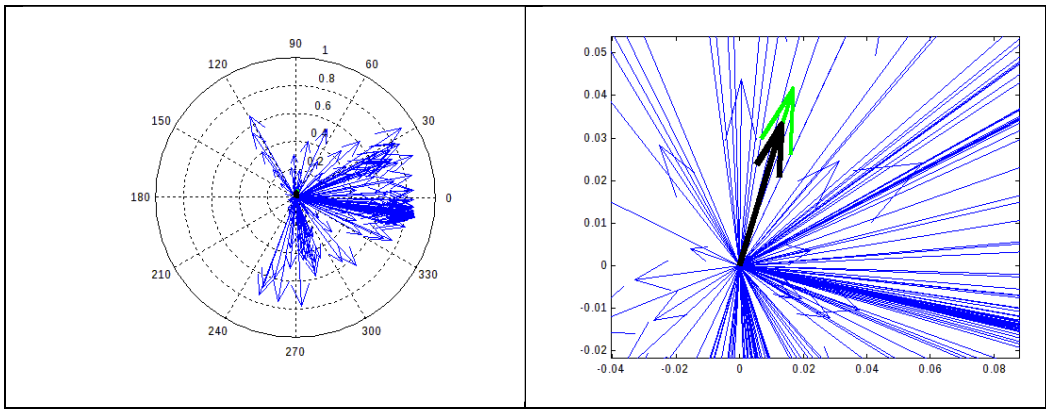
Σχήμα 10. 'BestAngleOnly' μέθοδος επιλογής κατάλληλου διανύσματος εξόδου (πράσινο) σε σχέση με αίτημα (μαύρο). Μια δεύτερη περίπτωση.



Σχήμα 11. 'BestAngleOnly' μέθοδος επιλογής κατάλληλου διανύσματος εξόδου (πράσινο) σε σχέση με αίτημα (μαύρο). Μια τρίτη περίπτωση.



Σχήμα 12. ' BestAngleAndMag ' μέθοδος επιλογής κατάλληλου διανύσματος εξόδου (πράσινο) σε σχέση με αίτημα (μαύρο). Το αποτέλεσμα για την αντίστοιχη προηγούμενη περίπτωση.



Σχήμα 13. ' BestAngleAndMag ' μέθοδος επιλογής κατάλληλου διανύσματος εξόδου (πράσινο) σε σχέση με αίτημα (μαύρο). Ένα ακόμη παράδειγμα.

5.4.1.1 ΔΟΚΙΜΕΣ ΕΞΟΜΟΙΩΣΗΣ ΤΟΥ MBC

Για την αξιολόγηση του ελεγκτή MBC, σχεδιάστηκε ένα περιβάλλον εξομοίωσης, στο οποίο ο MBC καλείται να επιτύχει ζητούμενες ταχύτητες. Στις εικόνες που θα ακολουθήσουν, καταγράφονται οι επιδόσεις του ελεγκτή.

Μπαίνοντας στη λεπτομέρεια της λειτουργίας του περιβάλλοντος εξομοίωσης :

- Λαμβάνει ένα αρχείο εισόδου που περιέχει τα σημεία ρύθμισης της ταχύτητας, της γωνιακής ταχύτητας και τις επαναλήψεις προσομοίωσης για κάθε σημείο ρύθμισης.

- Κάθε βήμα προσομοίωσης απέχει ένα Ts (που σημαίνει 30 ms) από το προηγούμενο. Ο λόγος για αυτό είναι ότι λειτουργεί συνδέοντας το MBC στο μοντέλο νευρωνικού δικτύου του συστήματος. Έτσι, οι έξοδοι MBC εφαρμόζονται ως είσοδοι στο μοντέλο. Η εκπαίδευση του μοντέλου έχει γίνει με ρυθμό Ts. Αυτό υπαγορεύει το Ts ως το προτεινόμενο χρονικό βήμα για τις προσομοιώσεις.

- Για κάθε καταχώρηση επιθυμητών ταχυτήτων, ξεκινά ένας βρόχος, η λειτουργία του οποίου είναι η εξής:

    - Διατηρεί ένα ιστορικό τριών δειγμάτων για την ταχύτητα, τη γωνιακή ταχύτητα και τα σφάλματα μεταξύ του σημείου ρύθμισης και της τρέχουσας τιμής της ταχύτητας και της γωνιακής ταχύτητας.

    - Μια βάση δεδομένων με τις μεταβλητές που περιγράφηκαν στο προηγούμενο βήμα, ανατίθεται στον MBC.

    - Ο MBC δημιουργεί ενέργεια ώσης και διεύθυνσης για το μοντέλο ως είσοδο. το μοντέλο λαμβάνει ως είσοδο επίσης τις τρέχουσες ταχύτητες και παράγει τις επιταχύνσεις (μεταφορικές και γωνιακές) που προκύπτουν από τις μεταβλητές εισόδου, προσεγγίζοντας κατά το δυνατό την επιθυμητή απόκριση του σκάφους.

- Οι επιταχύνσεις που προκύπτουν από το προηγούμενο βήμα ενσωματώνονται, χρησιμοποιώντας την προσέγγιση ολοκλήρωσης της παραγράφου 5.5.2.1, και υπολογίζονται οι αντίστοιχες ταχύτητες.
- Αφού αποθηκευτούν οι υπολογισμένες και οι προκύπτουσες τιμές του τρέχοντος βήματος, ο βρόχος ξεκινά ξανά από την αρχή.

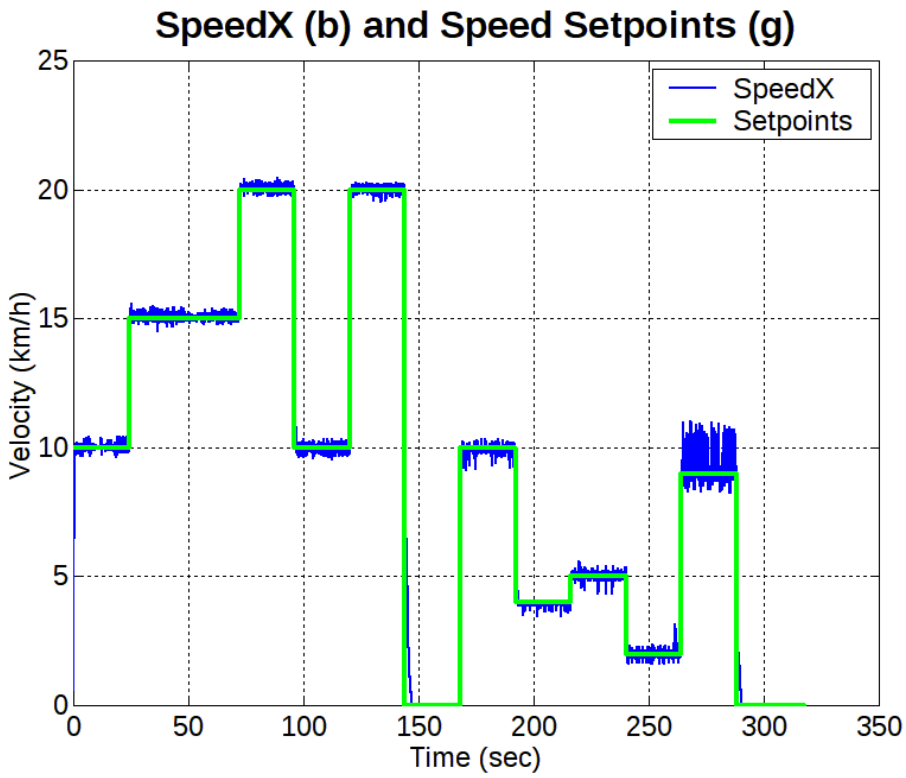
- Ενώ εκτελείται κάθε βήμα, τα αποτελέσματα αποθηκεύονται σε αρχεία και εμφανίζονται με τη μορφή γραφημάτων στο τέλος όλων των επαναλήψεων.

Τα σχήματα που ακολουθούν παρουσιάζουν τα αποτελέσματα της προσομοίωσης για τα σημεία ρύθμισης και τις επαναλήψεις προσομοίωσης που αναφέρονται στον Πίνακας 1.

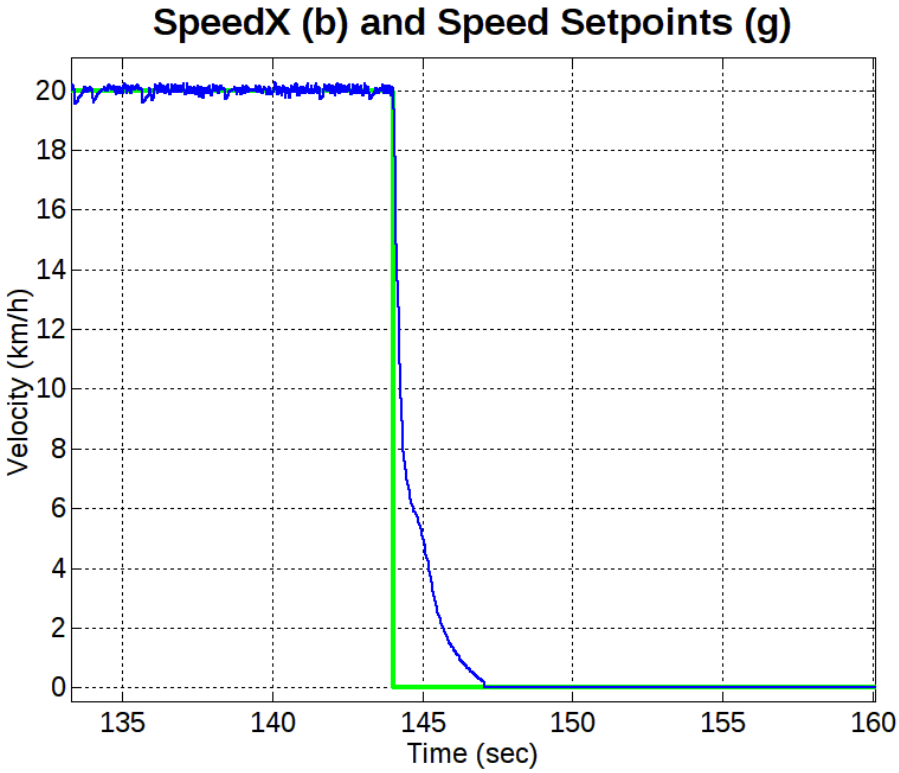
No	Velocity (km/h)	W (o/sec)	Simulation Reps
1	10	0	800
2	15	7	800
3	15	-3	800
4	20	-5	800
5	10	-25	800
6	20	0	800
7	0	0	800
8	10	0	800
9	4	0	800
10	5	7	800
11	2	10	800
12	9	20	800
13	0	0	1000

Πίνακας 1. Επιλογές ταχυτήτων για την εξομοίωση του MBC.

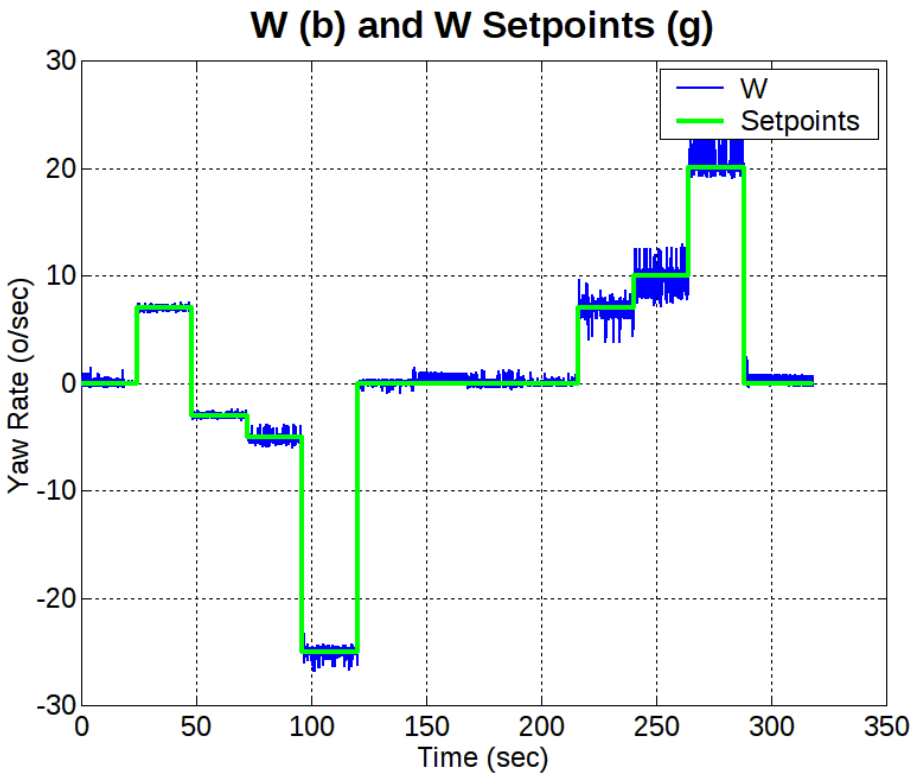
Τα αποτελέσματα από την εκτέλεση της προσομοίωσης για αυτά τα σημεία ρύθμισης ακολουθούν από το Σχήμα 30 έως το Σχήμα 35



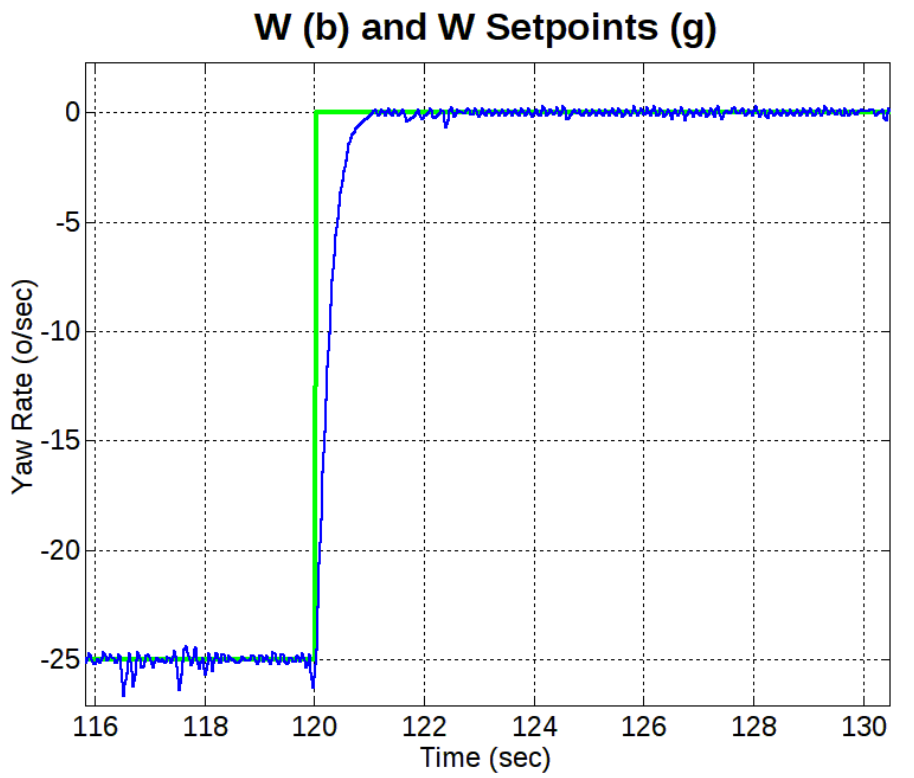
Σχήμα 14. Μεταφορική ταχύτητα σε αντιπαράθεση με τα αιτήματα προς τον MBC.



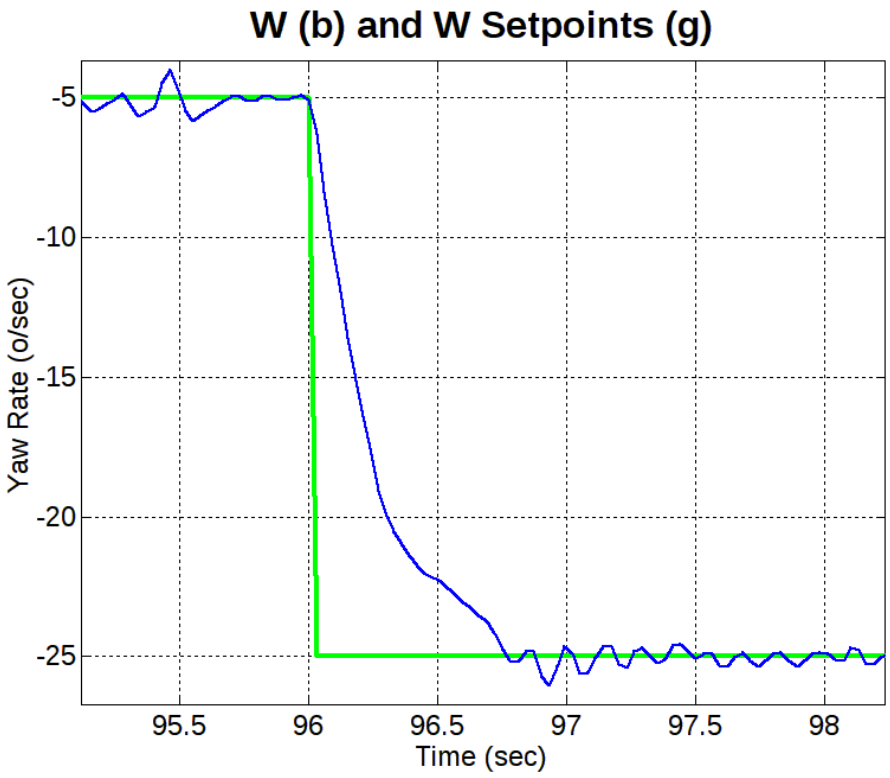
Σχήμα 15. Εστίαση σε συγκεκριμένη περιοχή αλλαγής του αιτήματος μεταφορικής ταχύτητας.



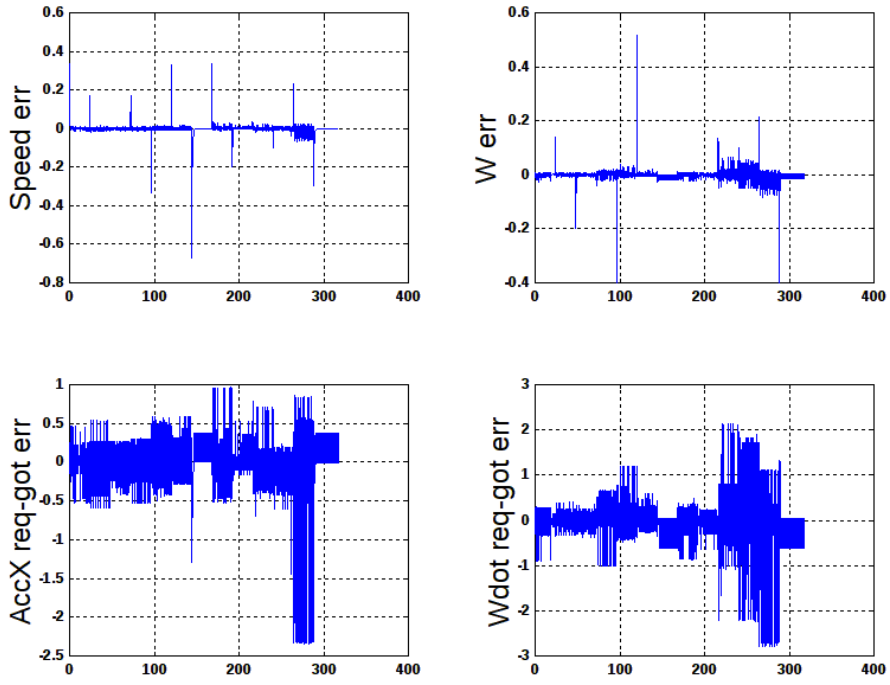
Σχήμα 16. Γωνιακή ταχύτητα σε αντιπαράθεση με το αντίστοιχο αίτημα προς MBC.



Σχήμα 17. Εστίαση σε συγκεκριμένη περιοχή αλλαγής του αιτήματος γωνιακής ταχύτητας.



Σχήμα 18. Εστίαση σε διαφορετική περιοχή αλλαγής του αιτήματος γωνιακής ταχύτητας.

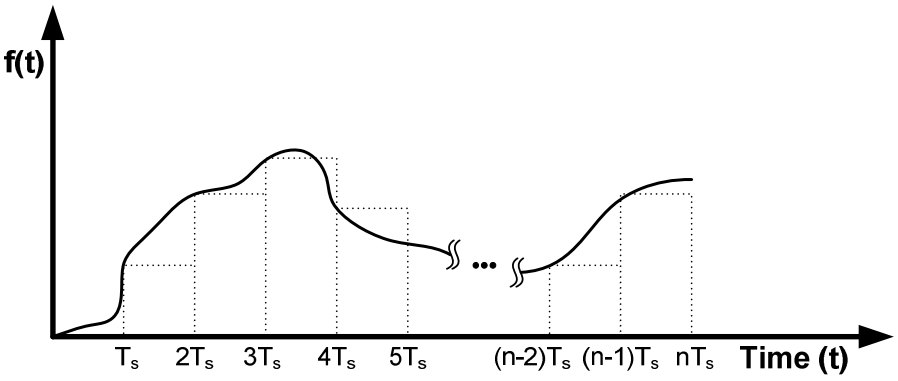


Σχήμα 19. Καταγραφές σφαλμάτων κατά την εξομοίωση.

5.4.2 Υλοποίηση των PID Controllers (Προσέγγιση Ολοκληρώματος μέσω Άθροισης Παραλληλογράμων)

5.4.2.1 ΠΡΟΣΕΓΓΙΣΗ ΑΘΡΟΙΣΗΣ ΣΤΟ ΟΛΟΚΛΗΡΩΜΑ

Ένας τρόπος προσέγγισης των ολοκληρωμάτων είναι η χρήση άθροισης παραλληλογράμμων [14]–[16]. Αυτό μοιάζει πολύ με την προσέγγιση που χρησιμοποιείται στον Zero Order Hold (ZoH) όπως θα δούμε στις επόμενες ενότητες. Για την προσέγγιση του ολοκληρώματος ενός σήματος με άθροισμα, ακολουθείται η μέθοδος που απεικονίζεται στο επόμενο σχήμα (Σχήμα 20). [17]:



Σχήμα 20. Προσεγγίζοντας το ολοκλήρωμα μέσω άθροισης.

Η χρήση του Ts υπονοεί την περίοδο δειγματοληψίας.

Όπως φαίνεται στο Σχήμα 20, το ολοκλήρωμα του αυθαίρετου σήματος προσεγγίζεται από την περιοχή των παραλληλογράμμων με ύψος ίσο με το προηγούμενο στιγμιαίο δείγμα και πλάτος ίσο με Ts. Τέτοια παραλληλόγραμμα δίνουν εμβαδόν f(k) x T_s για ένα αυθαίρετο $k \in \mathbb{N}$.

- Για τη χρονική στιγμή T_s: $\int_0^{T_s} f(t)dt \approx f(0)\cdot T_s$
- Για τη χρονική στιγμή $2T_s$: $\int_0^{2T_s} f(t)dt \approx f(0)\cdot T_s + f(1)\cdot T_s$
- Για τη χρονική στιγμή nT_s: $\int_0^{nT_s} f(t)dt \approx f(0)\cdot T_s + f(1)\cdot T_s + \ldots + f(n-1)\cdot T_s$

Αν χρησιμοποιήσουμε το $y[nT_s]$ ως σύμβολο για την προσέγγιση ενός ολοκληρώματος στη χρονική στιγμή nT_s, τότε από τις παραπάνω εξισώσεις έχουμε:

$$\int_0^{nT_s} f(t)dt \approx y[nT_s] = f(0)\cdot T_s + f(1)\cdot T_s + \ldots + f(n-1)\cdot T_s \Rightarrow$$

$$y\left[nT_s\right] = \sum_{k=1}^{n} f\left[\left(k-1\right)T_s\right]\cdot T_s \qquad (5.1)$$

Δεδομένου ότι όλα τα σήματα με τα οποία έχουμε να κάνουμε είναι αιτιατά, οι αρνητικές χρονικές τιμές τους είναι μηδέν; Για το λόγο αυτό (και επειδή $f\left[\left(-1\right)T_s\right] = 0$) μπορούμε να γράψουμε (5.1):

$$\boxed{y[nT_s] = \sum_{k=0}^{n} f\left[(k-1)T_s\right] \cdot T_s} \tag{5.2}$$

Από (5.2) παρατηρούμε επίσης ότι:

$$y[T_s] = f[0 \cdot T_s] \cdot T_s = f(0) \cdot T_s$$

$$y[2T_s] = f(0) \cdot T + f(T_s) \cdot T_s = y[T_s] + f(T_s) \cdot T_s$$

πράγμα που σημαίνει ότι για τη γενική περίπτωση

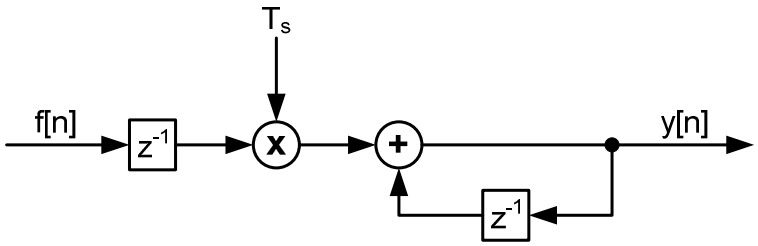
$$y[nT_s] = y\left[(n-1)T_s\right] + f\left[(n-1)T_s\right] \cdot T_s \tag{5.3}$$

Εφαρμόζοντας μετασχηματισμό Z στην εξίσωση (5.3) έχουμε ότι

$$Y(z) = z^{-1}Y(z) + z^{-1}T_s F(z) \Leftrightarrow$$

$$\boxed{Y(z) = \frac{z^{-1}T_s}{1 - z^{-1}} F(z)} \tag{5.4}$$

Εάν θέλουμε να αναπαραστήσουμε την (5.4) σε μορφή block διαγράμματος, τότε θα έχουμε

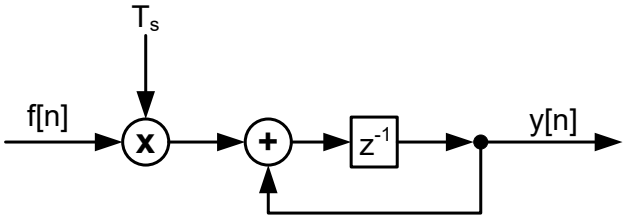


Διάγραμμα 4. Διαγραμματική απεικόνιση της προσέγγισης του ολοκληρώματος.

Ένα μπλοκ διάγραμμα που είναι πιο αποτελεσματικό, αναφορικά με τη χρήση στοιχείων μνήμης, μπορεί να προκύψει εάν απλοποιήσουμε το παραπάνω διάγραμμα με χειρισμό στοιχείων μπλοκ διαγράμματος ή λαμβάνοντας υπόψη ότι:

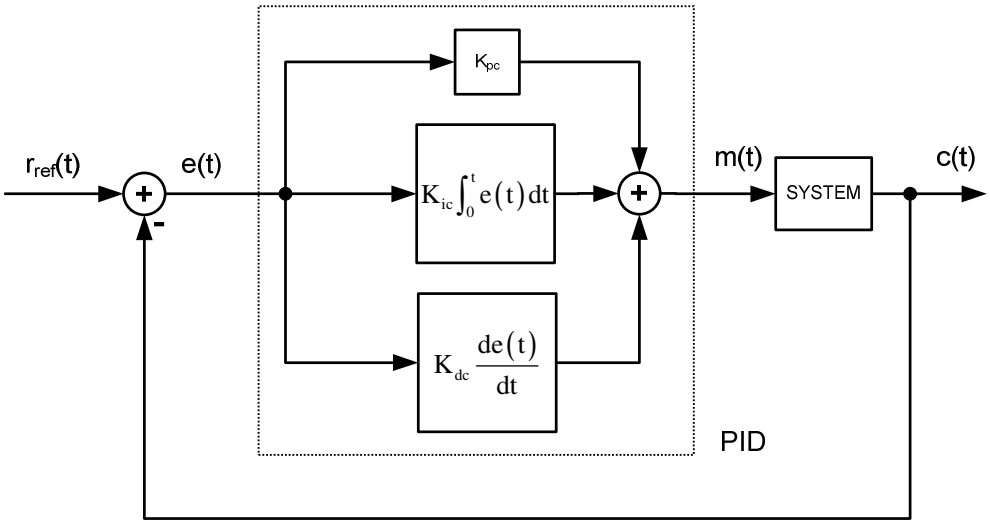
$$Y(z) = z^{-1}Y(z) + z^{-1}T_s F(z) = z^{-1}\left[Y(z) + T_s F(z)\right]$$

Αυτό σημαίνει ότι το Διάγραμμα 4 μπορεί να θεωρηθεί ισοδύναμο του



Διάγραμμα 5. Βελτιωμένη έκδοση (ως προς στοιχεία μνήμης) της προσέγγισης ολοκληρώματος.

5.4.2.2 DIGITIZING THE ANALOG PID FUNCTIONALITY



Εικόνα 11. Αναλογικός PID.

Για να μετατραπεί ο αναλογικός ελεγκτής PID σε ψηφιακό [8], εξετάζουμε κάθε όρο ξεχωριστά, κάτι που είναι σωστό, αφού τα γραμμικά συστήματα έχουν την ιδιότητα της υπέρθεσης.

P-term:

Ο όρος P είναι ο απλούστερος από όλους. Ενισχύει οποιοδήποτε εισερχόμενο σήμα με ένα κέρδος K_{pc}, όπου το c σημαίνει "συνεχές", υποδηλώνοντας το κέρδος συνεχούς συστήματος. Αυτό έγινε για να αποφευχθεί η ανάμειξη αυτών των σταθερών με το K_p που αφορά σε ψηφιακούς ελεγκτές. Η σήμανση c ακολουθείται για όλα τα κέρδη όρου PID.

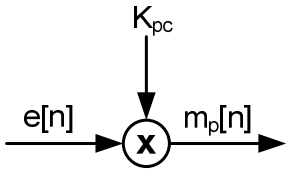
Η είσοδος του ψηφιακού P-όρου PID είναι το τρέχον δείγμα σφάλματος $e[nT_s]$. Η έξοδος του P-όρου είναι:

$$m_p[nT_s] = K_{pc}e[nT_s] \tag{5.5}$$

ή σε μορφή μετασχηματισμού Z

$$M_p(z) = K_{pc}E(z) \tag{5.6}$$

Το μπλοκ διάγραμμα του P-όρου είναι το ακόλουθο



Εικόνα 12. Μπλοκ διάγραμμα ψηφιακού P-όρου.

Θα πρέπει να σημειωθεί ότι με τον συμβολισμό e[n] εννοούμε $e[nT_s]$ (κάτι που ισχύει για κάθε σήμα διακριτού χρόνου) και θα χρησιμοποιούμε εναλλακτικά και τους δύο συμβολισμούς στο κείμενο.

I-term:

Για τον όρο I χρησιμοποιούμε την προσέγγιση που αναλύθηκε στην ενότητα "Προσέγγιση άθροισης στο ολοκλήρωμα". Αυτό έχει ως αποτέλεσμα τις ακόλουθες εξισώσεις:

$$K_{ic}\int_0^t e(\tau)d\tau = K_{ic}\int_0^{nT_s} e(\tau)d\tau \approx K_{ic}\sum_{k=0}^{n} e\left[(k-1)T_s\right]\cdot T_s$$

$$\boxed{m_i[nT_s] = K_{ic}T_s\sum_{k=0}^{n} e\left[(k-1)T_s\right]} \quad (5.7)$$

ή κατόπιν μετασχηματισμού Z

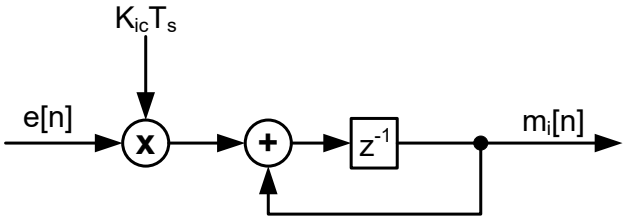
$$Z\{m_i[nT_s]\} = K_{ic}T_s Z\left\{\sum_{k=0}^{n} e\left[(k-1)T_s\right]\right\} \Rightarrow$$

$$\boxed{M_i(z) = K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}E(z)} \quad (5.8)$$

Για το μπλοκ διάγραμμα του I-όρου μπορούμε να απαλείψουμε τον παρονομαστή στην εξίσωση (5.8) και να έχουμε

$$M_i(z) = z^{-1}\left[M_i(z) + K_{ic}T_s E(z)\right]$$

Οπότε και θα έχουμε



Εικόνα 13. Ψηφιακός I όρος.

D-term:

Για τον όρο D προσεγγίζουμε την παράγωγο με διαφορές ως εξής:

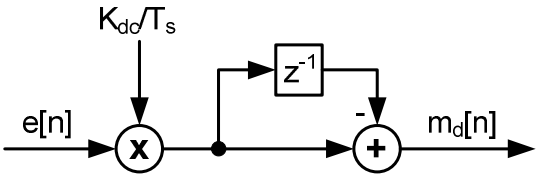
$$m_d(t) = K_{dc}\frac{de(t)}{dt} \approx K_{dc}\frac{\Delta e(t)}{\Delta t} \Rightarrow$$

$$\boxed{m_d[nT_s] = K_{dc}\frac{e[nT_s] - e\left[(n-1)T_s\right]}{T_s}} \quad (5.9)$$

Ή στα πλαίσια μετασχηματισμού Z

$$\boxed{M_d(z) = \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)E(z)} \quad (5.10)$$

Το διάγραμμα μπλοκ της (5.10) είναι



Εικόνα 14. Μπλοκ διάγραμμα ψηφιακού D-όρου.

Combining all terms:

Αφού αναλύσαμε κάθε όρο ξεχωριστά, μπορούμε τώρα να καταλήξουμε στην εξίσωση για τη συνολική έξοδο του ψηφιακού ελεγκτή PID [5]:

$$m[nT_s] = m_p[nT_s] + m_i[nT_s] + m_d[nT_s] \Rightarrow$$

$$\boxed{m[nT_s] = K_{pc}e[nT_s] + K_{ic}T_s\sum_{k=0}^{n} e[(k-1)T_s] + K_{dc}\frac{e[nT_s]-e[(n-1)T_s]}{T_s}} \quad (5.11)$$

ή με όρους μετασχηματισμού Z:

$$M(z) = M_p(z) + M_i(z) + M_d(z) \Rightarrow$$

$$M(z) = K_{pc}E(z) + K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}E(z) + \frac{K_{dc}}{T_s}(1-z^{-1})E(z) \Leftrightarrow$$

$$\boxed{M(z) = \left[K_{pc} + K_{ic}\frac{z^{-1}T_s}{1-z^{-1}} + \frac{K_{dc}}{T_s}(1-z^{-1})\right]E(z)} \quad (5.12)$$

Αυτό που έχει ιδιαίτερη σημασία στη διακριτοποίηση του αναλογικού PID είναι ότι εισάγεται η περίοδος δειγματοληψίας T_s ως πολλαπλασιαστής στους όρους I και D. Αυτό αποκαλύπτει την ύψιστη σημασία της επιλογής της περιόδου δειγματοληψίας, αφού μια λάθος επιλογή μπορεί να καταστήσει ένα σύστημα ασταθές!

5.4.2.3 BLOCK DIAGRAMS FOR DIGITAL PID

Η εξίσωση (5.12) μπορεί επίσης να μετατραπεί σε διάγραμμα μπλοκ κατόπιν κάποιων μαθηματικών χειρισμών. Απαλείφοντας τον όρο $(1-z^{-1})$ από τον παρονομαστή, έχουμε

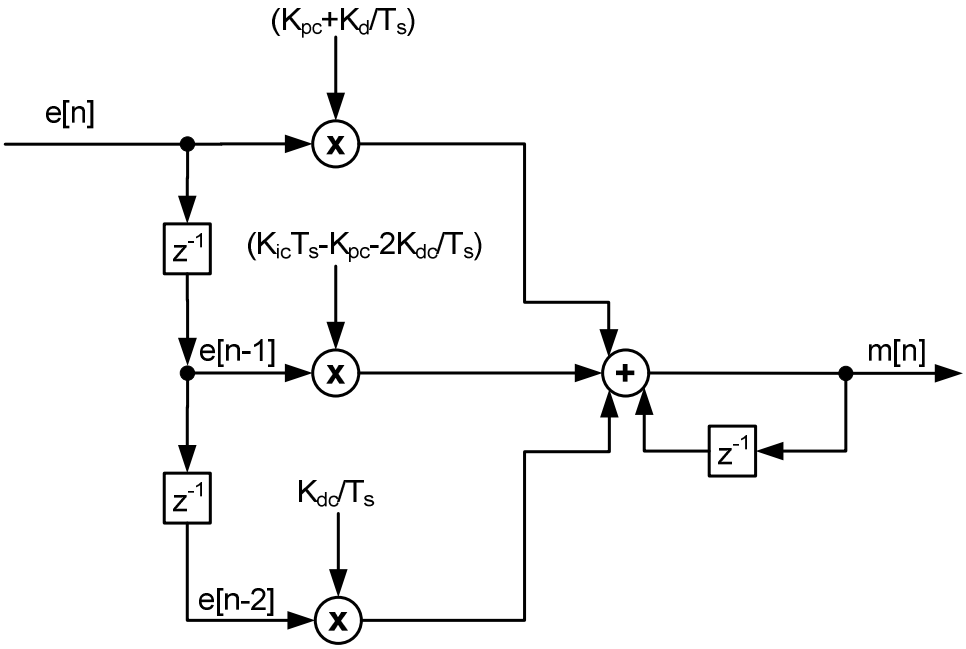
$$(1-z^{-1})M(z) = \left[(1-z^{-1})K_{pc} + K_{ic}T_s z^{-1} + \frac{K_{dc}}{T_s}(1-z^{-1})^2\right]E(z) \Leftrightarrow$$

$$(1-z^{-1})M(z) = \left[K_{pc} - z^{-1}K_{pc} + K_{ic}T_s z^{-1} + \frac{K_{dc}}{T_s}(1-2z^{-1}+z^{-2})\right]E(z) \Leftrightarrow$$

$$\boxed{M(z) = z^{-1}M(z) + \left[\left(K_{pc} + \frac{K_{dc}}{T_s}\right) + \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)z^{-1} + \frac{K_{dc}}{T_s}z^{-2}\right]E(z)} \quad (5.13)$$

$$\begin{aligned} m[nT_s] &= m[(n-1)T_s] + \left(K_{pc} + \frac{K_{dc}}{T_s}\right) e[nT_s] + \\ &+ \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right) e[(n-1)T_s] + \\ &+ \frac{K_{dc}}{T_s} e[(n-2)T_s] \end{aligned} \tag{5.14}$$

Οι (5.13) ή (5.14) δίνουν το ακόλουθο μπλοκ διάγραμμα



Εικόνα 15. Διάγραμμα μπλοκ της υλοποίησης ψηφιακού PID.

5.4.2.4 ΕΝΑΛΛΑΚΤΙΚΗ ΥΛΟΠΟΙΗΣΗ PID (VELOCITY FORM)

Υπάρχει επίσης μια εναλλακτική μορφή του ψηφιακού PID, η οποία εστιάζει στη μείωση του αντίκτυπου των ξαφνικών αλλαγών στην είσοδο. Η μαθηματική παράκαμψη που υλοποιεί την παραπάνω πρόταση θα φανεί στην εξαγωγή των εξισώσεων που ακολουθούν.

Αυτή η μορφή μπορεί επίσης να χρησιμοποιηθεί στην εφαρμογή μας. Στη συνέχεια θα υπάρξουν και σχετικά αποτελέσματα.

Για να εξαγάγουμε τις εξισώσεις για τη velocity form ξεκινάμε από την εξίσωση (5.11) της προηγούμενης ενότητας και την επεξεργαζόμαστε:

$$m[nT_s] = K_{pc}e[nT_s] + K_{ic}T_s\sum_{k=0}^{n} e[(k-1)T_s] + K_{dc}\frac{e[nT_s] - e[(n-1)T_s]}{T_s}$$

Θέλουμε να βρούμε τη διαφορά μεταξύ διαδοχικών δειγμάτων του ελεγκτή, τα οποία θα συμβολίζονται ως $\nabla m[nT_s]$.

$$\nabla m[nT_s] = m[nT_s] - m[(n-1)T_s] \Rightarrow$$

$$\begin{aligned}\nabla m[nT_s] = \quad & K_{pc}\{e[nT_s] - e[(n-1)T_s]\} + \\ & K_{ic}T_s\left\{\sum_{k=0}^{n} e[(k-1)T_s] - \sum_{k=0}^{n-1} e[(k-1)T_s]\right\} + \\ & \frac{K_{dc}}{T_s}\{e[nT_s] - e[(n-1)T_s] - e[(n-1)T_s] + e[(n-2)T_s]\} \Leftrightarrow\end{aligned}$$

$$\begin{aligned}\nabla m[nT_s] = \quad & K_{pc}\{e[nT_s] - e[(n-1)T_s]\} + \\ & K_{ic}T_s e[(n-1)T_s] + \\ & \frac{K_{dc}}{T_s}\{e[nT_s] - 2e[(n-1)T_s] + e[(n-2)T_s]\} \Leftrightarrow\end{aligned}$$

$$\begin{aligned}\nabla m[nT_s] = \quad & \left(K_{pc} + \frac{K_{dc}}{T_s}\right)e[nT_s] + \\ & \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)e[(n-1)T_s] + \\ & \frac{K_{dc}}{T_s}e[(n-2)T_s]\end{aligned} \qquad (5.15)$$

Αντικαθιστώντας το $e[nT_s]$ με το ισοδύναμό του δηλαδή $r_{ref}[nT_s] - c[nT_s]$, η (5.15) μετατρέπεται σε

$$\begin{aligned}\nabla m[nT_s] = \quad & \left(K_{pc} + \frac{K_{dc}}{T_s}\right)r_{ref}[nT_s] + \\ & \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)r_{ref}[(n-1)T_s] + \\ & \frac{K_{dc}}{T_s}r_{ref}[(n-2)T_s] - \\ & \left(K_{pc} + \frac{K_{dc}}{T_s}\right)c[nT_s] - \\ & \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)c[(n-1)T_s] - \\ & \frac{K_{dc}}{T_s}c[(n-2)T_s]\end{aligned} \qquad (5.16)$$

Η μαθηματική παράκαμψη που χρησιμοποιείται στην εξαγωγή της velocity form είναι η επιβολή ισότητας των τριών τελευταίων δειγμάτων εισόδου αναφοράς: $r_{ref}[nT_s] = r_{ref}[(n-1)T_s] = r_{ref}[(n-2)T_s]$ [4].

Η ιδέα για την αλλαγή αυτή προήλθε από τους όρους της r_{ref}, και τις μετατροπές που χρειάζονται προκειμένου να απαλειφθούν οι παράγοντες που εισάγουν την επίδραση του r_{ref} στους I και D όρους, μειώνοντας αποτελεσματικά την επίδραση αλλαγών της r_{ref} στην έξοδο. Με αυτή την υπόθεση η (5.16) αλλάζει σε

$$\begin{aligned}\nabla m[nT_s] = m[nT_s] - m[(n-1)T_s] = \\ K_{ic}T_s r_{ref}[nT_s] - \\ \left(K_{pc} + \frac{K_{dc}}{T_s}\right)c[nT_s] - \\ \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)c[(n-1)T_s] - \\ \frac{K_{dc}}{T_s}c[(n-2)T_s] \Leftrightarrow\end{aligned}$$

$$\boxed{\begin{aligned}m[nT_s] = m[(n-1)T_s] + \\ K_{ic}T_s r_{ref}[nT_s] - \\ \left(K_{pc} + \frac{K_{dc}}{T_s}\right)c[nT_s] - \\ \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)c[(n-1)T_s] - \\ \frac{K_{dc}}{T_s}c[(n-2)T_s]\end{aligned}} \quad (5.17)$$

Από την (5.17) μπορούμε να προχωρήσουμε σε μετασχηματισμό Z και να πάρουμε τον τύπο με όρους του z.

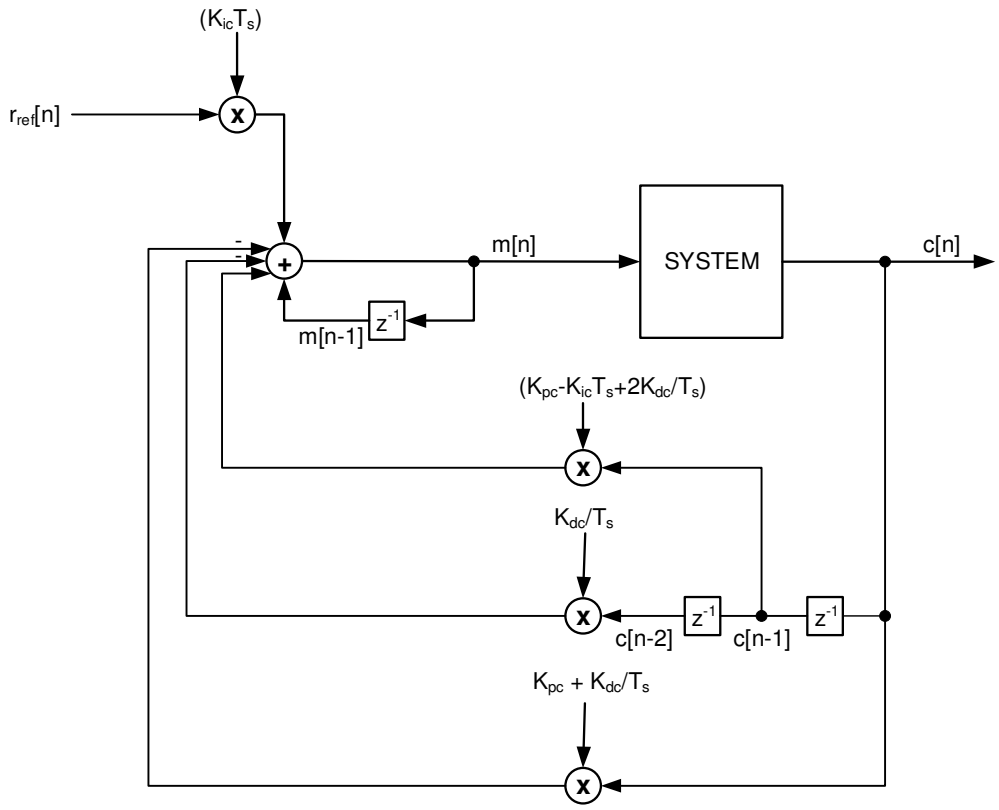
$$\begin{aligned}M(z) = z^{-1}M(z) + K_{ic}T_s R_{ref}(z) - \left(K_{pc} + \frac{K_{dc}}{T_s}\right)C(z) \\ - \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)z^{-1}C(z) - z^{-2}\frac{K_{dc}}{T_s}C(z) \Leftrightarrow\end{aligned}$$

$$\begin{aligned}(1-z^{-1})M(z) = K_{ic}T_s\left[R_{ref}(z) - z^{-1}C(z)\right] - K_{pc}(1-z^{-1})C(z) - \\ \frac{K_{dc}}{T_s}(1-2z^{-1}+z^{-2})C(z) \Leftrightarrow\end{aligned}$$

$$\left(1-z^{-1}\right)M\left(z\right)=K_{ic}T_s\left[R_{ref}\left(z\right)-z^{-1}C\left(z\right)\right]-K_{pc}\left(1-z^{-1}\right)C\left(z\right)-$$
$$\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)^2 C\left(z\right)\Leftrightarrow$$

$$\boxed{M\left(z\right)=-K_{pc}C\left(z\right)+K_{ic}T_s\frac{\left[R_{ref}\left(z\right)-z^{-1}C\left(z\right)\right]}{\left(1-z^{-1}\right)}-\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)C\left(z\right)} \quad (5.18)$$

Η υλοποίηση σε μορφή μπλοκ διαγράμματος της (5.17) η (5.18) παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα 16. Μπλοκ διάγραμμα της velocity form υλοποίησης του PID.

5.5 Εισάγοντας τους PID στο Σύστημα

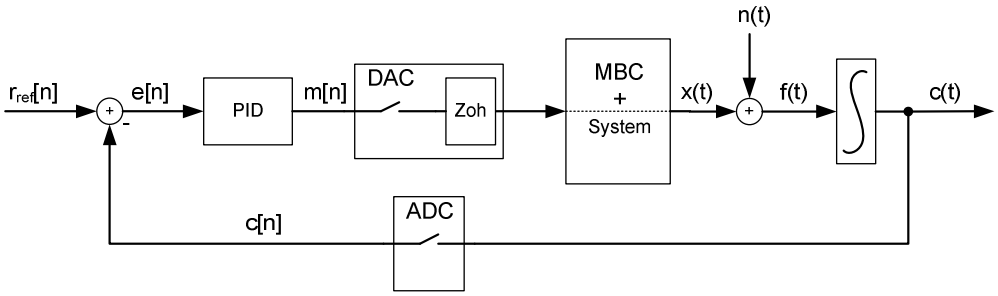
Σε αυτό το σημείο είναι σημαντικό να δείξουμε πώς το σύστημα προορίζεται να ελέγχεται με τη χρήση ελεγκτών PID.

Η χρήση ελεγκτών MBC κάνει αποσύζευξη στο σύστημα, επομένως μπορούν να σχηματιστούν δύο ξεχωριστοί βρόχοι ελέγχου με χρήση PID. Ένας για τη γωνιακή ταχύτητα και ένας για τη μεταφορική ταχύτητα. Το επόμενο σημαντικό βήμα είναι να προσδιοριστεί ποια μέρη του βρόχου ανήκουν στον ψηφιακό κόσμο και ποια μέρη ανήκουν στον αναλογικό. Αυτό θα καθορίσει τη σωστή θέση και τον αριθμό των

μετατροπέων σήματος από αναλογικών σε ψηφιακό (A/D ή ADC) και ψηφιακού σε αναλογικό (D/As ή DAC) που χρειάζονται.

Αναφερόμενοι στο «Διάγραμμα 2. Πλήρες μοντέλο ελέγχου συστήματος.» και «Διάγραμμα 3. Ελεγκτής Ανατροφοδότησης (FC) και η λειτουργία του στο σύστημα.». τα μέρη που ανήκουν στο ψηφιακό μέρος είναι όλα εκτός από το ίδιο το σύστημα. Αυτό σημαίνει ότι τα μόνα σημεία όπου χρειάζονται μετατροπείς που διασυνδέουν τον ψηφιακό με τον αναλογικό κόσμο είναι τα όρια του συστήματος.

Το μπλοκ διάγραμμα που περιγράφει τη διεπαφή του συστήματος φαίνεται στην Εικόνα 17.



Εικόνα 17. Βρόχος ελέγχου με PID και μετατροπείς σήματος.

Επειδή ο MBC λειτουργεί σχεδόν σαν αντίστροφο σύστημα, η έξοδος του PID περνάει μέσω του D/A + Zoh απευθείας στο $\dot{c}(t)$. Στο σημείο αυτό προστίθεται θόρυβος εξαιτίας

- Ανακριβειών στη μοντελοποίηση
- Πιθανών αστοχιών του MBC στο να επιτύχει την πλήρη αντιστροφή του συστήματος
- Πιθανής αναποτελεσματικότητας του MBC στο να αποσυζεύξει τουυς δύο βρόχους ελέγχου
- Εξωτερικών επιδράσεων στο σκάφος όπως άνεμος και κυματισμός.

Όλα τα παραπάνω μοντελοποιούνται σα θόρυβος που προστίθεται πριν το σήμα f(t).

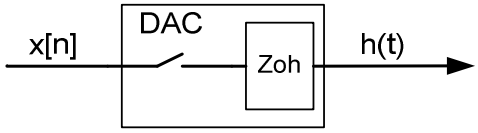
5.5.1 Μαθηματική Ανάλυση στο Διακριτό Χρόνο

Το πρώτο πράγμα που πρέπει να ειπωθεί, σχετικά με την ανάλυση του συστήματος σε διακριτό χρόνο, είναι ότι οι πλασματικοί κρουστικοί δειγματολήπτες [4], [9], [16] που αντιπροσωπεύουν τον μισό DAC (το άλλο μισό της αναπαράστασης του DAC είναι ο Zoh) και ο ADC πρέπει να είναι απολύτως συγχρονισμένοι. Αυτό σημαίνει ότι πρέπει να έχουν και την ίδια συχνότητα και την ίδια φάση. Με τον τρόπο αυτό απλοποιούνται αρκετά οι αναλύσεις στον μαθηματικό τομέα, δεν χρειάζονται μετατροπείς ρυθμού δειγματοληψίας, ενώ τα διαθέσιμα σήματα για επεξεργασία αναφέρονται στην ίδια στιγμή δειγματοληψίας. Επίσης, οποιαδήποτε εσωτερική επεξεργασία πρέπει να ξεκινάει μετά τη δειγματοληψία και των δύο μετατροπέων και πρέπει να εκτελείται σε χρόνο μικρότερο από την περίοδο δειγματοληψίας.

Έχοντας ήδη αναλύσει τον διακριτό PID (αναφορά στις εξισώσεις (5.12) με (5.14)) το επόμενο βήμα είναι να αναλυθεί η επίδραση της ύπαρξης του Zoh.

5.5.1.1 ΜΑΘΗΜΑΤΙΚΗ ΑΝΑΛΥΣΗ ZOH

Αυτό που κάνει ο zero order hold είναι να διατηρεί σταθερή την τιμή στην έξοδό του μεταξύ των στιγμών δειγματοληψίας. Μαθηματικά και με αναφορά στο παρακάτω σχήμα [4]:



Εικόνα 18. Εικόνα αναφοράς για την ανάλυση του Zoh.

$$h\left(nT_s + t\right) = x\left(nT_s\right), t \in \left[0, T_s\right)$$

Αυτό σημαίνει ότι

$$\begin{aligned} h(t) = {} & x(0)\left[U(t) - U\left(t - T_s\right)\right] + x\left(T_s\right)\left[U\left(t - T_s\right) - U\left(t - 2T_s\right)\right] + \ldots \\ & + x\left(nT_s\right)\left[U\left(t - nT_s\right) - U\left[t - (n+1)T_s\right]\right] \end{aligned} \tag{5.19}$$

Παίρνοντας μετασχηματισμός Laplace της (5.19) έχουμε ότι

$$\begin{aligned} L\{h(t)\} = {} & x(0)\left(\frac{1}{s} - e^{-sT_s}\frac{1}{s}\right) + x\left(T_s\right)\left(e^{-sT_s}\frac{1}{s} - e^{-s2T_s}\frac{1}{s}\right) + \ldots \\ & + x\left(nT_s\right)\left[e^{-snT_s}\frac{1}{s} - e^{-s(n+1)T_s}\frac{1}{s}\right] \Rightarrow \end{aligned}$$

$$H(s) = \sum_{n=0}^{\infty}\left[x\left(nT_s\right)e^{-snT_s}\frac{\left(1 - e^{-sT_s}\right)}{s}\right] \Leftrightarrow$$

$$H(s) = \frac{\left(1 - e^{-sT_s}\right)}{s}\sum_{n=0}^{\infty}x\left(nT_s\right)e^{-snT_s} \tag{5.20}$$

Το άθροισμα της εξίσωσης (5.20) είναι ο starred Laplace transform του x(t) (ισούται με τον μετασχηματισμό Z με $z = e^{sT_s}$), που σημαίνει ότι η (5.20) γίνεται

$$H(s) = G_{Zoh}(s)X^*(s) \tag{5.21}$$

Που σημαίνει ότι

$$\boxed{G_{Zoh}(s) = \frac{\left(1 - e^{-sT_s}\right)}{s}} \tag{5.22}$$

Εφόσον η ανάλυση του συστήματος σε διακριτό χρόνο περιλαμβάνει τον μετασχηματισμό Z, είναι σημαντικό να βρεθεί ο μετασχηματισμός Z μιας συνάρτησης μεταφοράς συνεχούς χρόνου όταν πολλαπλασιάζεται με τη συνάρτηση μεταφοράς του Zoh.

Για να γίνει αυτό, υποθέτουμε ότι είναι ένα αυθαίρετο σήμα Y(s).

$$Y(s) = G_{Zoh}(s)G(s) \Rightarrow$$

$$Y(s) = \frac{1-e^{-sT_s}}{s}G(s) = \frac{G(s)}{s} - e^{-sT_s}\frac{G(s)}{s} \tag{5.23}$$

if $\frac{G(s)}{s} = G_1(s)$ τότε η (5.23) γίνεται

$$Y(s) = G_1(s) - e^{-sT_s}G_1(s) \Rightarrow$$

$$L^{-1}\{Y(s)\} = L^{-1}\{G_1(s)\} - L^{-1}\{e^{-sT_s}G_1(s)\} \Rightarrow$$

$$y(t) = g_1(t) - \int_0^t \delta(t - T_s - \tau)g_1(\tau)\,d\tau \Rightarrow$$

$$y(t) = g_1(t) - g_1(t - T_s) \Rightarrow$$

$$Y(z) = G_1(z) - z^{-1}G_1(z) \Leftrightarrow$$

$$\boxed{Y(z) = (1-z^{-1})G_1(z) \text{ where } G_1(z) = Z\left\{\frac{G(s)}{s}\right\}} \tag{5.24}$$

5.5.1.2 ΕΦΑΡΜΟΖΟΝΤΑΣ ΤΑ ΠΡΟΗΓΟΥΜΕΝΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟΝ ΚΛΕΙΣΤΟ ΒΡΟΧΟ

Μετά την ολοκλήρωση της ανάλυσης τοποθέτησης των δειγματοληπτών στον βρόχο, το αποτέλεσμα του συνδυασμού MBC + σκάφος και την μαθηματική επεξήγηση της συμπεριφοράς του Zoh μπορούμε να προχωρήσουμε στην ανάλυση του πλήρους βρόχου.

Ξεκινώντας από το τμήμα συνεχούς χρόνου, έχουμε (στο s):

$$F(s) = X(s) + N(s) \tag{5.25}$$

$$X(s) = M^*(s)\frac{1-e^{-sT_s}}{s} \tag{5.26}$$

Με το σύμβολο * εννοούμε τον starred Laplace transform δηλαδή τον μετασχηματισμό Laplace ενός κρουστικά δειγματοληπτημένου σήματος.

$$C(s) = F(s)\frac{1}{s} \tag{5.27}$$

Θεωρώντας προσωρινά N(s) = 0, έχουμε από τις (5.25), (5.26) και (5.27) ότι

$$C(s) = M^*(s)\frac{(1-e^{-sT_s})}{s}\cdot\frac{1}{s} \Rightarrow$$

$$C(z)=Z\left\{M^{*}(s)\frac{\left(1-e^{-sT_s}\right)}{s}\cdot\frac{1}{s}\right\}=M(z)\left(1-z^{-1}\right)Z\left\{\frac{1}{s^2}\right\}=M(z)\cancel{\left(1-z^{-1}\right)}\frac{T_s z^{-1}}{\left(1-z^{-1}\right)^{\cancel{2}}}\Leftrightarrow$$

$$C(z)=M(z)\frac{T_s z^{-1}}{\left(1-z^{-1}\right)} \tag{5.28}$$

Αντικαθιστώντας το M(z) από την (5.12), η (5.28) γίνεται

$$C(z)=\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}E(z) \tag{5.29}$$

Λαμβάνοντας υπ 'όψιν ότι

$$E(z)=R_{ref}(z)-C(z) \tag{5.30}$$

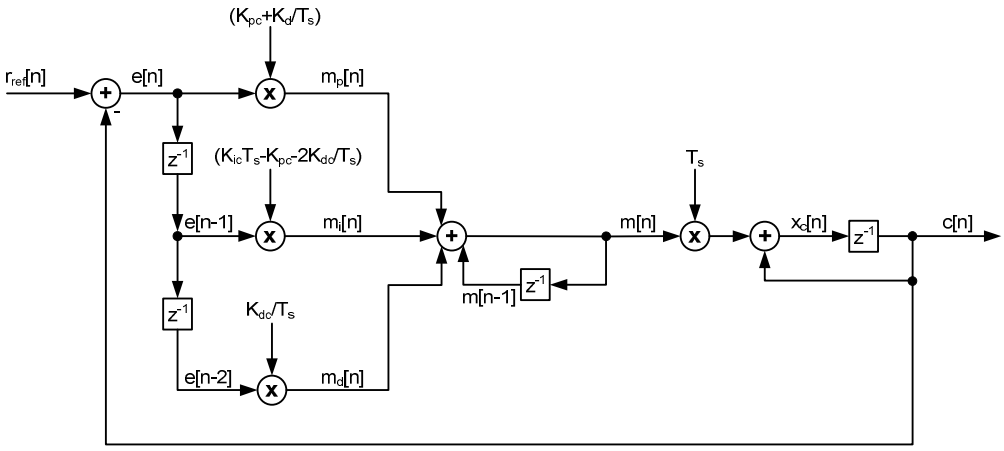
Έχουμε ότι η (5.29) είναι

$$C(z)=\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}\left[R_{ref}(z)-C(z)\right]\Leftrightarrow$$

$$\boxed{C(z)=\frac{\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}}{1+\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}}R_{ref}(z)} \tag{5.31}$$

Η εξίσωση (5.31) είναι η pulse transfer function κλειστού βρόχου από την είσοδο $R_{ref}(z)$ στην έξοδο C(z).

Το μπλοκ διάγραμμα της (5.31) ακολουθεί



Εικόνα 19. Διάγραμμα μπλοκ για την ανάλυση κλειστού βρόχου στον διακριτό χρόνο.

Εμβαθύνοντας στην (5.31) μπορούμε να βρούμε εξίσωση που να παράγει ισοδύναμο αποτέλεσμα με το παραπάνω block διάγραμμα, προκειμένου να δίνεται δυνατότητα διασταύρωσης των αποτελεσμάτων.

$$C(z)=\frac{\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]T_s z^{-1}}{\left(1-z^{-1}\right)+\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]T_s z^{-1}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{\left[\frac{K_{pc}\left(1-z^{-1}\right)}{\left(1-z^{-1}\right)}+K_{ic}\frac{z^{-1}T_s}{\left(1-z^{-1}\right)}+\frac{K_{dc}}{T_s}\frac{\left(1-z^{-1}\right)^2}{\left(1-z^{-1}\right)}\right]T_s z^{-1}}{\left(1-z^{-1}\right)+\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{\left(1-z^{-1}\right)}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]T_s z^{-1}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{\left[K_{pc}\left(1-z^{-1}\right)+K_{ic}z^{-1}T_s+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)^2\right]T_s z^{-1}}{\left(1-z^{-1}\right)^2+\left[K_{pc}\left(1-z^{-1}\right)+K_{ic}z^{-1}T_s+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)^2\right]T_s z^{-1}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{K_{pc}T_s z^{-1}-K_{pc}T_s z^{-2}+K_{ic}z^{-2}T_s^{\,2}+K_{dc}z^{-1}\left(1-2z^{-1}+z^{-2}\right)}{1-2z^{-1}+z^{-2}+K_{pc}T_s z^{-1}-K_{pc}T_s z^{-2}+K_{ic}z^{-2}T_s^{\,2}+K_{dc}z^{-1}\left(1-2z^{-1}+z^{-2}\right)}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{K_{pc}T_s z^{-1}-K_{pc}T_s z^{-2}+K_{ic}z^{-2}T_s^{\,2}+K_{dc}z^{-1}-2K_{dc}z^{-2}+K_{dc}z^{-3}}{1-2z^{-1}+z^{-2}+K_{pc}T_s z^{-1}-K_{pc}T_s z^{-2}+K_{ic}z^{-2}T_s^{\,2}+K_{dc}z^{-1}-2K_{dc}z^{-2}+K_{dc}z^{-3}}R_{ref}(z)\Leftrightarrow$$

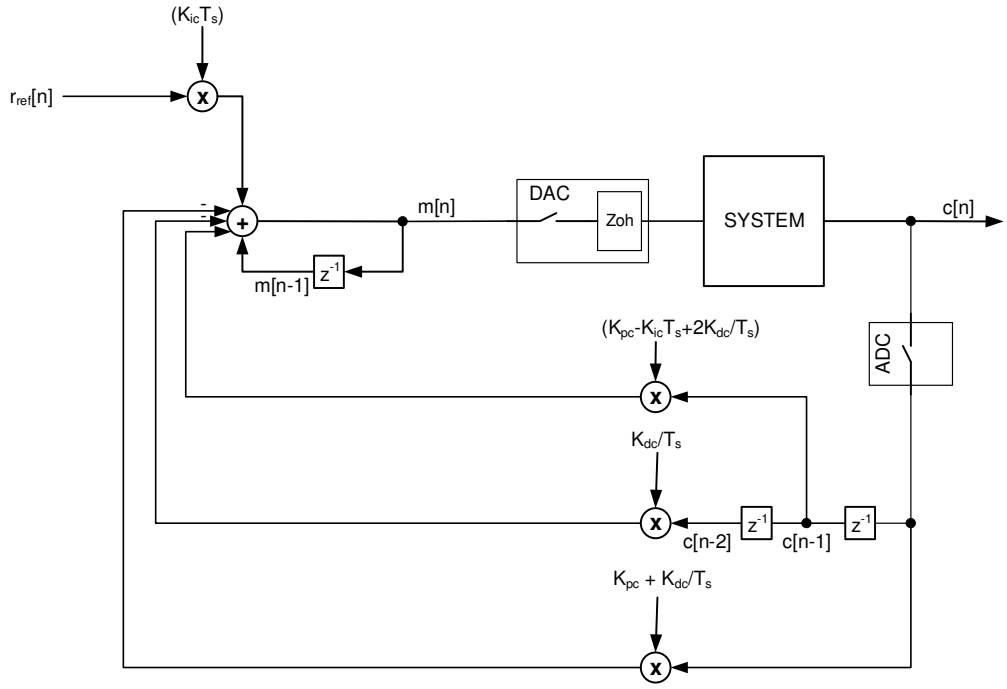
$$C(z)=\frac{\left(K_{pc}T_s+K_{dc}\right)z^{-1}+\left(K_{ic}T_s^{\,2}-K_{pc}T_s-2K_{dc}\right)z^{-2}+K_{dc}z^{-3}}{1+\left(K_{pc}T_s-2+K_{dc}\right)z^{-1}+\left(1-K_{pc}T_s+K_{ic}T_s^{\,2}-2K_{dc}\right)z^{-2}+K_{dc}z^{-3}}R_{ref}(z)\Rightarrow$$

$$\begin{aligned}c[n]=&\left(K_{pc}T_s+K_{dc}\right)r_{ref}\left[(n-1)T_s\right]+\\&+\left(K_{ic}T_s^{\,2}-K_{pc}T_s-2K_{dc}\right)r_{ref}\left[(n-2)T_s\right]+\\&+K_{dc}r_{ref}\left[(n-3)T_s\right]+\\&-\left(K_{pc}T_s-2+K_{dc}\right)c\left[(n-1)T_s\right]+\\&-\left(1-K_{pc}T_s+K_{ic}T_s^{\,2}-2K_{dc}\right)c\left[(n-2)T_s\right]+\\&-K_{dc}c\left[(n-3)T_s\right]\end{aligned}\qquad(5.32)$$

5.5.1.3 ΑΝΑΛΥΟΝΤΑΣ ΤΟΝ ΚΛΕΙΣΤΟ ΒΡΟΧΟ ΜΕ VELOCITY FORM PID

Το διάγραμμα συστήματος που χρησιμοποιεί τη velocity form του PID γίνεται όπως φαίνεται στο Σχήμα 48. Το διάγραμμα παρουσιάζει επίσης τις πραγματικές θέσεις των μετατροπέων στο σύστημα και όπως συζητήθηκε στην προηγούμενη ενότητα, η ανάλυση λαμβάνει ως γεγονός ότι οι μετατροπείς είναι πλήρως συγχρονισμένοι.

Για να αναλύσουμε στο Z, πρέπει να αντικαταστήσουμε το πλαίσιο που αναφέρεται στο σύστημα με την προσέγγισή μας για το σύστημα, η οποία είναι ένας Zoh ακολουθούμενος από τη συνάρτηση 1/s, η οποία δίνει την εξίσωση (5.28).

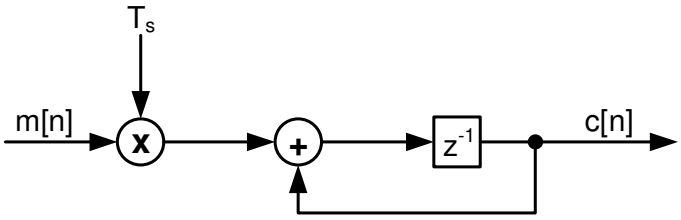


Εικόνα 20. Χρησιμοποιώντας τη velocity form για τον έλεγχο του συστήματος. Στο διάγραμμα εμφανίζονται και οι μετατροπείς.

Επαναλαμβάνοντας για ευκολία την (5.28) σε αυτό το σημείο του κειμένου έχουμε:

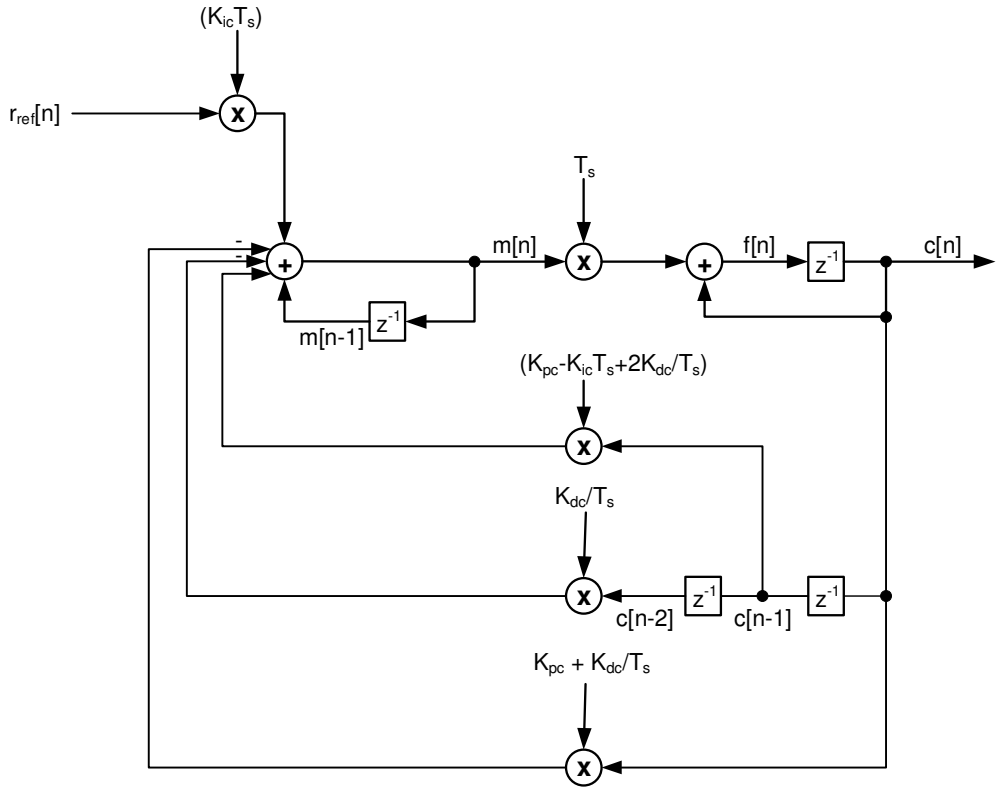
$$C(z) = M(z)\frac{T_s z^{-1}}{(1-z^{-1})} \Leftrightarrow C(z) = z^{-1}\left[C(z) + T_s M(z)\right]$$

Που σε μορφή μπλοκ διαγράμματος αντιστοιχεί σε αυτό της Εικόνα 21.



Εικόνα 21. Αναπαράσταση του συνόλου σύστημα και Zoh.

Επομένως αντικαθιστώντας το πλαίσιο “System” της Εικόνα 20 με την προσέγγισή του που φαίνεται στην Εικόνα 21 θα έχουμε το αποτέλεσμα της Εικόνα 22.



Εικόνα 22. Υλοποίηση του συστήματος με velocity form PID έτοιμη για ανάλυση στο Z.

Το διάγραμμα μπορεί πλέον να αναλυθεί στο Z.

Ξεκινώντας από την εξίσωση για τη velocity form έχουμε

$$\left.\begin{aligned} M(z) &= -K_{pc}C(z) + K_{ic}T_s\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)C(z) \\ C(z) &= M(z)\frac{T_s z^{-1}}{\left(1-z^{-1}\right)} \end{aligned}\right\} \Rightarrow$$

$$C(z) = \left\{-K_{pc}C(z) + K_{ic}T_s\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)C(z)\right\}\frac{T_s z^{-1}}{\left(1-z^{-1}\right)} \Leftrightarrow$$

$$C(z)\left(1-z^{-1}\right) = -K_{pc}T_s z^{-1}C(z) + K_{ic}T_s^2 z^{-1}\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - K_{dc}z^{-1}\left(1-z^{-1}\right)C(z) \Leftrightarrow$$

$$C(z)\left(1-z^{-1}\right)^2 = -K_{pc}T_s z^{-1}\left(1-z^{-1}\right)C(z) + K_{ic}T_s^2 z^{-1}R_{ref}(z) - K_{ic}T_s^2 z^{-2}C(z) - K_{dc}z^{-1}\left(1-z^{-1}\right)^2 C(z) \Leftrightarrow$$

$$\left[\left(1-z^{-1}\right)^2 + K_{pc}T_s z^{-1}\left(1-z^{-1}\right) + K_{ic}T_s^2 z^{-2} + K_{dc}z^{-1}\left(1-z^{-1}\right)^2\right]C(z) = K_{ic}T_s^2 z^{-1}R_{ref}(z) \Leftrightarrow$$

$$\left[1 + \left(K_{pc}T_s - 2 + K_{dc}\right)z^{-1} + \left(1 - K_{pc}T_s + K_{ic}T_s^2 - 2K_{dc}\right)z^{-2} + K_{dc}z^{-3}\right]C(z) = K_{ic}T_s^2 z^{-1}R_{ref}(z) \Rightarrow$$

$$\begin{aligned} c[nT_s] = {} & K_{ic}T_s^2 r_{ref}[(n-1)T_s] - \\ & (K_{pc}T_s - 2 + K_{dc})c[(n-1)T_s] - \\ & (1 - K_{pc}T_s + K_{ic}T_s^2 - 2K_{dc})c[(n-2)T_s] - \\ & K_{dc}c[(n-3)T_s] \end{aligned} \tag{5.33}$$

Η εξίσωση (5.33) δίνει την απόκριση του συστήματος υπό τον έλεγχο του velocity form PID [5].

5.5.1.4 ΣΥΓΚΡΙΝΟΝΤΑΣ ΜΕ ΤΟΝ ΠΛΗΡΩΣ ΑΝΑΛΟΓΙΚΟ ΒΡΟΧΟ PID

Μια σημαντική σύγκριση για να ελέγξουμε την ποιότητα της υλοποίησής μας και να επιβεβαιώσουμε ότι είναι προς τη σωστή κατεύθυνση, είναι να τη συγκρίνουμε με έναν πλήρως αναλογικό βρόχο, σαν το σύστημά μας να ελεγχόταν από έναν αναλογικό ελεγκτή PID.

Όπως θα αποδειχθεί στην παρακάτω παράγραφο “Επιλέγοντας τα PID Gains”, η συνάρτηση μεταφοράς ενός αντίστοιχου πλήρως αναλογικού συστήματος είναι

$$G_{rc}(s) = \frac{\frac{G_p(s)}{s}}{1 + \frac{G_p(s)}{s}} \tag{5.34}$$

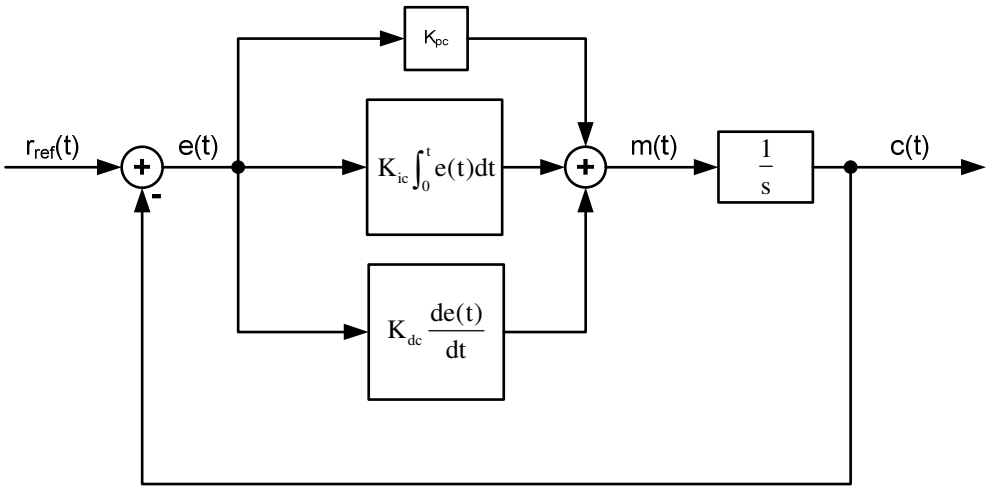
όπου $G_p(s)$ είναι η συνάρτηση μεταφοράς του PID block, δηλαδή

$$G_p(s) = K_{pc} + K_{ic}\frac{1}{s} + sK_{dc} \tag{5.35}$$

$$G_{rc}(s) = \frac{K_{pc} + K_{ic}\frac{1}{s} + sK_{dc}}{s + K_{pc} + K_{ic}\frac{1}{s} + sK_{dc}} = \frac{K_{pc}s + K_{ic} + K_{dc}s^2}{(1+K_{dc})s^2 + K_{pc}s + K_{ic}} \Leftrightarrow$$

$$G_{rc}(s) = \frac{\frac{K_{dc}}{(1+K_{dc})}s^2 + \frac{K_{pc}}{(1+K_{dc})}s + \frac{K_{ic}}{(1+K_{dc})}}{s^2 + \frac{K_{pc}}{(1+K_{dc})}s + \frac{K_{ic}}{(1+K_{dc})}} \tag{5.36}$$

Αυτή η συνάρτηση μεταφοράς αναφέρεται σε ένα σύστημα με τη δομή που φαίνεται στην Εικόνα 23.



Εικόνα 23. Ανάλογο συνεχούς χρόνου για το σύστημα ελέγχου.

Συγκρίνοντας τον παρονομαστή της (5.36) με τον παρονομαστή του πρότυπου δευτεροβάθμιου συστήματος δηλαδή

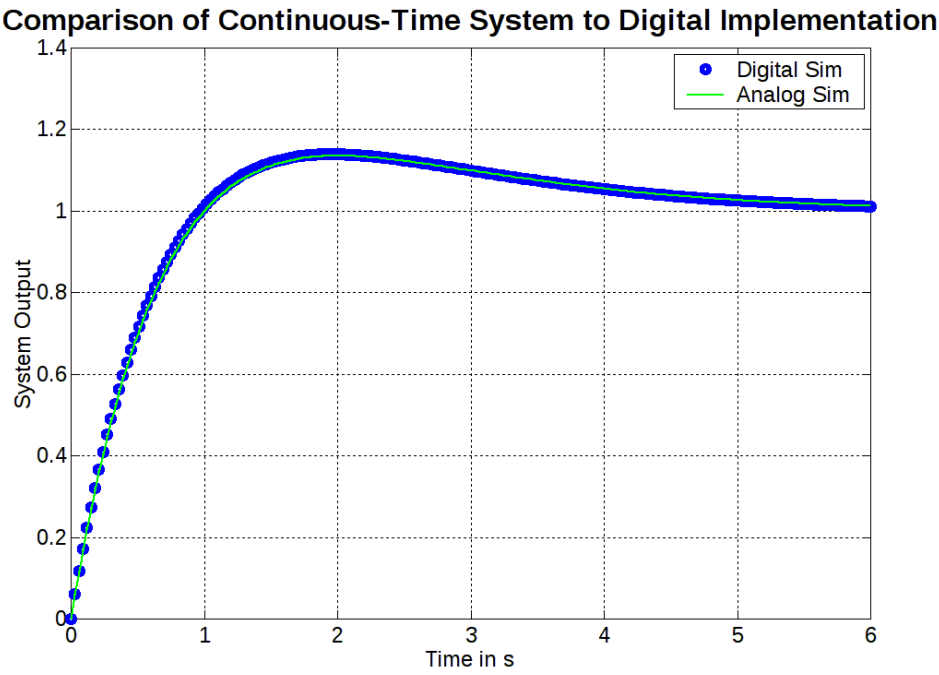
$$s^2 + 2\zeta\omega_n s + \omega_n^2$$

τότε για να έχουμε ένα ζ και ω_n κοντά στο 1 επιλέγουμε $K_{pc} = 2$, $K_{ic} = 1$. Θέλοντας να αποφύγουμε τον D όρο ει δυνατόν (για να μειωθεί ο θόρυβος που μπορεί να δημιουργήσει), για το παράδειγμά μας θα χρησιμοποιήσουμε $K_{dc} = 0$.

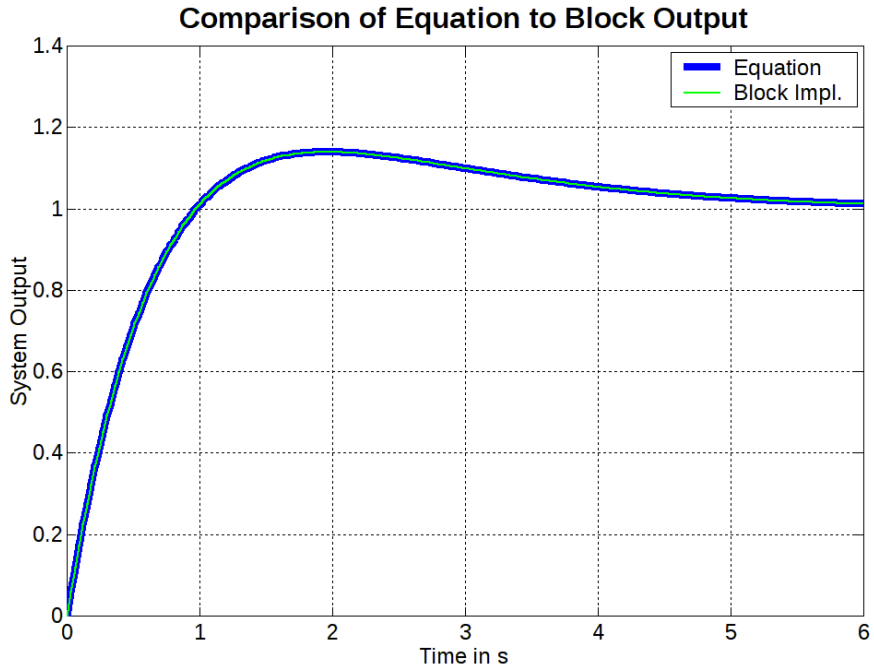
Τα αποτελέσματα της προσομοίωσης του αναλογικού συστήματος, η υλοποίηση μπλοκ του ψηφιακού συστήματος και η εξίσωση (5.32) φαίνονται στα σχήματα «Εικόνα 52. Σύγκριση συστήματος συνεχούς χρόνου με ψηφιακή υλοποίηση» και «Εικόνα 53. Σύγκριση υλοποίησης εξίσωσης με υλοποίηση μπλοκ.

Στην περίπτωση της εικόνας «Εικόνα 24. Σύγκριση απόκρισης συνεχούς χρόνου με ψηφιακή υλοποίηση.» παρατηρούμε ότι η ψηφιακή υλοποίηση δίνει αποτελέσματα πολύ κοντά σε αυτά του συστήματος συνεχούς χρόνου και ως εκ τούτου θεωρούμε την προσέγγιση επιτυχή.

Στην περίπτωση της Εικόνα 25, το αποτέλεσμα είναι απόλυτη ταύτιση (ως όφειλε) αποδεικνύοντας την ακρίβεια και των δύο υλοποιήσεων (μπλοκ και εξίσωσης).



Εικόνα 24. Σύγκριση απόκρισης συνεχούς χρόνου με ψηφιακή υλοποίηση.



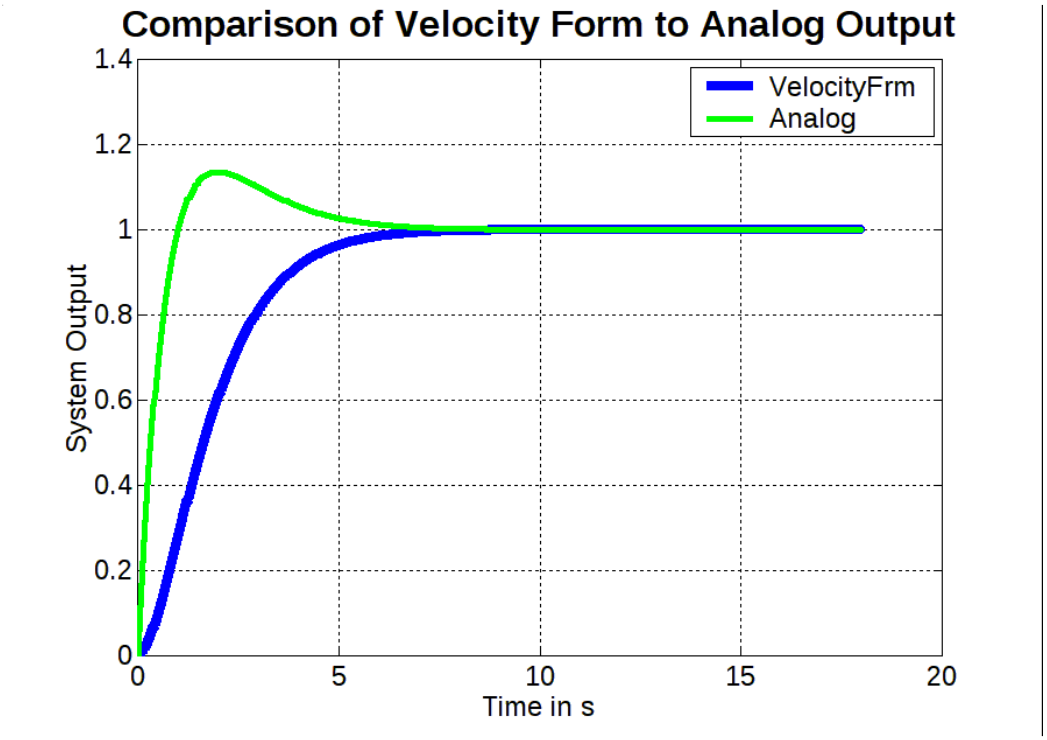
Εικόνα 25. Σύγκριση αποκρίσεων υλοποίησης με εξισώσεις διαφορών και υλοποίησης με block διάγραμμα.

5.5.1.4.1 Απόκριση Πλήρους Συστήματος με PID Τύπου Velocity Form

Χρησιμοποιώντας τα ίδια PID gains με την προηγούμενη ενότητα, η ανάλυση μπορεί να επαναληφθεί με τη velocity form του PID.

Η υλοποίηση με velocity form αναμένεται να είναι πιο νωχελική, μια και έχει υπάρξει μαθηματικός χειρισμός ο οποίος περιορίζει τη δράση της αλλαγής της εισόδου.

Τα αποτελέσματα της ανάλυσης του συστήματος χρησιμοποιώντας το μπλοκ διάγραμμα της εικόνας Εικόνα 22 αποτυπώνονται στην Εικόνα 26.

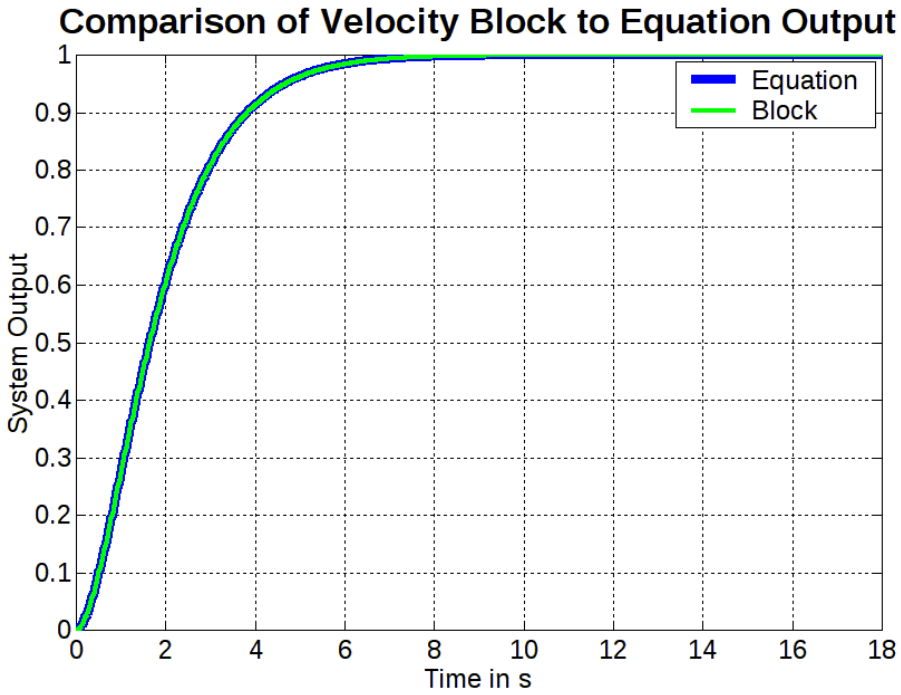


Εικόνα 26. Απόκριση υλοποίησης velocity form PID σε αντιπαράθεση με υλοποίηση συνεχούς χρόνου.

Όπως ήταν αναμενόμενο, η απόκριση χρησιμοποιώντας τη φόρμα ταχύτητας είναι πιο αργή σε σύγκριση με τον αναλογικό ελεγκτή. Στη συγκεκριμένη περίπτωση δείχνει και να ταιριάζει καλύτερα στην εφαρμογή, καθώς εξομαλύνει το overshoot.

Για να επιβεβαιώσουμε την ορθότητα της υλοποίησης, έγινε διασταύρωση με τα αποτελέσματα που παρήχθησαν από την εξίσωση (5.33) που περιγράφει το σύστημα.

Το αποτέλεσμα της σύγκρισης φαίνεται στην Εικόνα 27. Όπως φαίνεται από το σχήμα, τα αποτελέσματα είναι απόλυτη ταύτιση, γεγονός που αποδεικνύει την ισοδυναμία των δύο μεθόδων εξαγωγής αποτελεσμάτων.



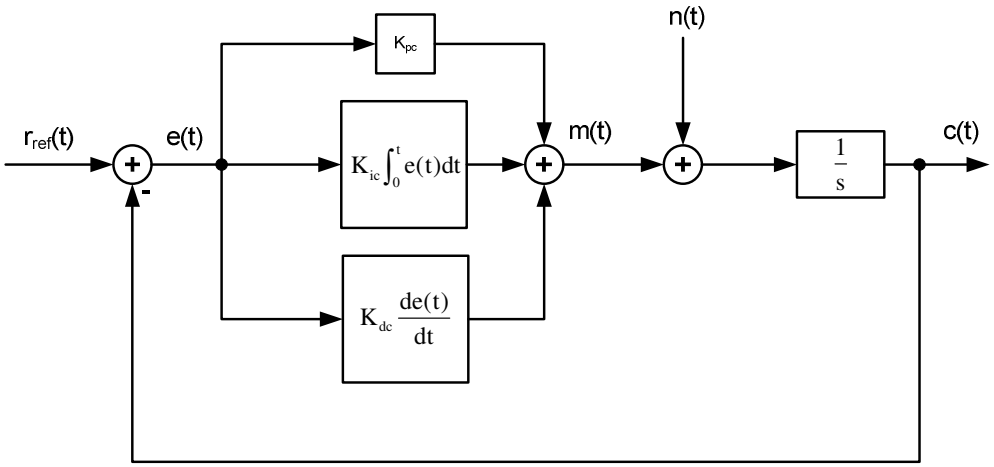
Εικόνα 27. Σύγκριση αποκρίσεων velocity form υλοποίησης με εξισώσεις διαφορών και υλοποίησης με block διάγραμμα

5.5.2 Επιλέγοντας τα PID Gains

Όπως αναφέρεται στην ενότητα “Feedback Controller (FC)”, ο MBC λειτουργεί ως αποζεύκτης και μετατροπέας συστήματος, επομένως οι βρόχοι για τη μεταφορική ταχύτητα και τη γωνιακή ταχύτητα μπορούν να διαχωριστούν και να αντιμετωπιστούν σαν να λείπουν το MBC και το σκάφος. Οι ανακρίβειες που προκύπτουν από αυτήν την προσέγγιση αντιμετωπίζονται ως θόρυβος που προστίθεται στις επιταχύνσεις. Άλλες πηγές θορύβου για τις επιταχύνσεις είναι τα κύματα και ο άνεμος.

Η όλη επεξεργασία της επιλογής κέρδους θα πραγματοποιηθεί σε συνεχή χρόνο και η ακρίβεια των αποτελεσμάτων θα επαληθευτεί συγκρίνοντας τα αποτελέσματα διακριτού χρόνου στη συνέχεια.

Ξεκινώντας σε συνεχή χρόνο, μπορούμε να απεικονίσουμε κάθε έναν από τους δύο βρόχους του συστήματός μας ως εξής:



Εικόνα 28. Αναπαράσταση συνεχούς χρόνου του βρόχου του συστήματος.

Αναφερόμαστε σε δύο βρόχους αφού υπάρχει ένας για τη μεταφορική ταχύτητα του σκάφους και ένας για τη γωνιακή ταχύτητα.

Σε αυτό το σημείο πρέπει να καταστεί σαφές, ότι αυτός ο τύπος προσέγγισης του συστήματος πρέπει να λαμβάνει υπόψη την αδράνεια του συστήματος. Προσπαθώντας, για παράδειγμα, να αναγκάσουμε το σκάφος να επιτύχει πολλαπλά Hertz στην αλλαγή γωνιακής ταχύτητας, θα αποτύχουμε λόγω της αδράνειας του σκάφους. Επομένως, το μοντέλο δε θα επιτύχει μια ρεαλιστική προσέγγιση του σκάφους. Έχοντας αυτό υπόψη, προχωράμε στην ανάλυση του βρόχου. Ξεκινάμε με n(t) = 0:

$$E(s) = R_{ref}(s) - C(s)$$

$$C(s) = G_p(s) \cdot \frac{1}{s} E(s)$$

$$C(s) = \frac{G_p(s)}{s}\left[R_{ref}(s) - C(s)\right] \Leftrightarrow$$

$$C(s) = \frac{\frac{G_p(s)}{s}}{1 + \frac{G_p(s)}{s}} R_{ref}(s)$$

$$\boxed{G_{rc}(s) = \frac{\frac{G_p(s)}{s}}{1 + \frac{G_p(s)}{s}}} \qquad (5.37)$$

Στη σχέση (5.37) ο δείκτης “rc” υπονοεί από την είδοσο “r_{ref}” στην έξοδο “c”.

Αναφορικά με το θόρυβο: η συνάρτηση μεταφοράς από την είσοδο του θορύβου στην έξοδο είναι η εξής:

$$C(s) = \left[N(s) + M(s)\right]\frac{1}{s} \qquad (5.38)$$

$$M(s) = G_p(s) \cdot \left[-C(s)\right] \Leftrightarrow M(s) = -G_p(s)C(s) \Rightarrow$$

$$C(s) = \left[N(s) - G_p(s)C(s)\right]\frac{1}{s} \Leftrightarrow$$

$$sC(s) = N(s) - G_p(s)C(s) \Leftrightarrow$$

$$C(s) = \frac{N(s)}{s + G_p(s)} \Rightarrow$$

$$G_{nc}(s) = \frac{1}{s + G_p(s)} \qquad (5.39)$$

Η συνάρτηση μεταφοράς $G_p(s)$ του PID είναι:

$$G_p(s) = K_{pc} + K_{ic}\frac{1}{s} + sK_{dc} \qquad (5.40)$$

Επομένως από (5.39) και (5.40) θα έχουμε ότι

$$G_{nc}(s) = \frac{1}{s + K_{pc} + K_{ic}\frac{1}{s} + sK_{dc}} \Leftrightarrow$$

$$G_{nc}(s) = \frac{s}{s^2 + K_{pc}s + K_{ic} + s^2K_{dc}} \Leftrightarrow$$

$$G_{nc}(s) = \frac{s}{(1 + K_{dc})s^2 + K_{pc}s + K_{ic}} \Leftrightarrow$$

$$G_{nc}(s)=\frac{\frac{s}{(1+K_{dc})}}{s^2+\frac{K_{pc}}{(1+K_{dc})}s+\frac{K_{ic}}{(1+K_{dc})}} \tag{5.41}$$

Για να γίνουν οι εξισώσεις λίγο μικρότερες, το σύμβολο α_d ορίζεται ως:

$$\alpha_d=\frac{1}{(1+K_{dc})} \tag{5.42}$$

Με χρήση της (5.42) η (5.41) μεταβάλλεται σε

$$G_{nc}(s)=\frac{\alpha_d s}{s^2+\alpha_d K_{pc}s+\alpha_d K_{ic}}\Rightarrow$$

$$G_{nc}(j\omega)=\frac{\alpha_d j\omega}{-\omega^2+\alpha_d K_{pc}j\omega+\alpha_d K_{ic}}\Leftrightarrow$$

$$G_{nc}(j\omega)=\frac{\alpha_d j\omega}{\alpha_d K_{ic}-\omega^2+\alpha_d K_{pc}j\omega}\Rightarrow$$

$$|G_{nc}(j\omega)|=\frac{|\alpha_d\omega|}{\sqrt{\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2}} \tag{5.43}$$

Για να διευκολυνθεί ο υπολογισμός της ακραίας τιμής για την (5.43), υπολογίζεται η παράγωγος του $|G_{nc}(j\omega)|^2$. Με αυτόν τον τρόπο αποφεύγεται η απόλυτη τιμή και η τετραγωνική ρίζα.

$$|G_{nc}(j\omega)|^2=\frac{(\alpha_d\omega)^2}{\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2}\Rightarrow$$

$$\frac{d|G_{nc}(j\omega)|^2}{d\omega}=\frac{2\alpha_d{}^2\omega\left[\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2\right]-\alpha_d{}^2\omega^2\left[2\left(\alpha_d K_{ic}-\omega^2\right)(-2\omega)+2\left(\alpha_d K_{pc}\right)^2\omega\right]}{\left[\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2\right]^2}=$$

$$\frac{2\alpha_d{}^2\omega\left[\left(\alpha_d K_{ic}\right)^2-2\alpha_d K_{ic}\omega^2+\omega^4+\left(\alpha_d K_{pc}\right)^2\omega^2\right]-\alpha_d{}^2\omega^2\left[-4\omega\left(\alpha_d K_{ic}-\omega^2\right)+2\left(\alpha_d K_{pc}\right)^2\omega\right]}{\left[\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2\right]^2}=$$

$$\frac{2\alpha_d{}^4K_{ic}{}^2\omega-4\alpha_d{}^3K_{ic}\omega^3+2\alpha_d{}^2\omega^5+2\alpha_d{}^3K_{pc}{}^2\omega^3+4\alpha_d{}^2\omega^3\left(\alpha_d K_{ic}-\omega^2\right)-2\alpha_d{}^3K_{pc}{}^2\omega^3}{\left[\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2\right]^2}=$$

$$\frac{2\alpha_d{}^4K_{ic}{}^2\omega-4\alpha_d{}^3K_{ic}\omega^3+2\alpha_d{}^2\omega^5+2\alpha_d{}^3K_{pc}{}^2\omega^3+4\alpha_d{}^3K_{ic}\omega^3-4\alpha_d{}^2\omega^5-2\alpha_d{}^3K_{pc}{}^2\omega^3}{\left[\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2\right]^2}=$$

$$\frac{2\alpha_d{}^4K_{ic}{}^2\omega+\left(-4\alpha_d{}^3K_{ic}\omega^3+2\alpha_d{}^3K_{pc}{}^2\omega^3+4\alpha_d{}^3K_{ic}\omega^3-2\alpha_d{}^3K_{pc}{}^2\omega^3\right)-2\alpha_d{}^2\omega^5}{\left[\left(\alpha_d K_{ic}-\omega^2\right)^2+\left(\alpha_d K_{pc}\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{nc}(j\omega)\right|^2}{d\omega}=\frac{2\alpha_d{}^4K_{ic}{}^2\omega-2\alpha_d{}^2\omega^5}{\left[\left(\alpha_dK_{ic}-\omega^2\right)^2+\left(\alpha_dK_{pc}\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{nc}(j\omega)\right|^2}{d\omega}=\frac{2\alpha_d{}^2\omega\left(\alpha_d{}^2K_{ic}{}^2-\omega^4\right)}{\left[\left(\alpha_dK_{ic}-\omega^2\right)^2+\left(\alpha_dK_{pc}\omega\right)^2\right]^2} \quad (5.44)$$

Αναζητώντας τα ακρότατα, αν εξισώσουμε την παράγωγο με το μηδέν, τότε έχουμε:

$$\frac{d\left|G_{nc}(j\omega)\right|^2}{d\omega}=0\Rightarrow$$

$$\frac{2\alpha_d{}^2\omega\left(\alpha_d{}^2K_{ic}{}^2-\omega^4\right)}{\left[\left(\alpha_dK_{ic}-\omega^2\right)^2+\left(\alpha_dK_{pc}\omega\right)^2\right]^2}=0\Rightarrow$$

$\omega=0$ or $\omega^4=\alpha_d{}^2K_{ic}{}^2\Leftrightarrow\omega^2=\left|\alpha_dK_{ic}\right|\Leftrightarrow$

$$\omega=\sqrt{\left|\alpha_dK_{ic}\right|} \quad (5.45)$$

Επειδή $\alpha_d>0,K_{ic}>0$ η (5.45) γίνεται:

$$\boxed{\omega=\sqrt{\alpha_dK_{ic}}} \quad (5.46)$$

Η τιμή του $\left|G_{nc}(j\omega)\right|$ στο $\omega=\sqrt{\alpha_dK_{ic}}$ είναι (χρησιμοποιώντας τις (5.43) και (5.46)) :

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{\left|\alpha_d\sqrt{\alpha_dK_{ic}}\right|}{\sqrt{\left(\alpha_dK_{ic}-\left(\sqrt{\alpha_dK_{ic}}\right)^2\right)^2+\left(\alpha_dK_{pc}\sqrt{\alpha_dK_{ic}}\right)^2}}\Leftrightarrow$$

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{\left|\alpha_d\sqrt{\alpha_dK_{ic}}\right|}{\sqrt{\left(\alpha_dK_{ic}-\left|\alpha_dK_{ic}\right|\right)^2+\left(\alpha_dK_{pc}\sqrt{\alpha_dK_{ic}}\right)^2}}\overset{K_{pc}>0,K_{ic}>0,\alpha_d>0}{\Rightarrow}$$

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{\alpha_d\sqrt{\alpha_dK_{ic}}}{\alpha_dK_{pc}\sqrt{\alpha_dK_{ic}}}\Leftrightarrow$$

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{1}{K_{pc}}$$

Εφόσον η συνάρτηση μεταφοράς (5.41) έχει ένα μηδενικό στο 0, και δύο πόλους στο $\omega\neq0$, η καμπύλη αρχίζει να ανέρχεται από το μηδέν και γυρίζει πάλι προς τα κάτω μετά τον δεύτερο πόλο (τον πιο απομακρυσμένο από το μηδέν). Αυτό σημαίνει ότι το ακρότατο που βρήκαμε είναι το μέγιστο. Έτσι, η παραπάνω εξίσωση είναι το $\left\|G_{nc}\right\|_\infty$, αλλά αυτό θα επαληθευτεί και από το διάγραμμα bode της G_{nc}.

5.5.2.1 ΣΥΝΑΡΤΗΣΗ ΜΕΤΑΦΟΡΑΣ ΑΠΟ ΤΗΝ ΕΙΣΟΔΟ ΘΟΡΥΒΟΥ ΣΤΟ ΣΦΑΛΜΑ

Πιο σημαντική ακόμα συνάρτηση για την εφαρμογή τεχνικών Robust Control είναι η συνάρτηση μεταφοράς από την είσοδο θορύβου στο σφάλμα e(t). Υπολογίζοντας το E(s), έχουμε:

Αρχίζοντας από τον υπολογισμό του E(s), θεωρώντας $r_{ref}(t) = 0$ ($R_{ref}(s) = 0$).

$$E(s) = -C(s) = -\left[N(s) + M(s)\right]\frac{1}{s} \Rightarrow$$

$$E(s) = -\left[N(s) + G_p(s)E(s)\right]\frac{1}{s} \Leftrightarrow$$

$$\left[1 + \frac{G_p(s)}{s}\right]E(s) = -\frac{N(s)}{s} \Leftrightarrow$$

$$E(s) = -\frac{N(s)}{s + G_p(s)} \Leftrightarrow$$

$$G_{ne}(s) = -\frac{1}{s + G_p(s)} \tag{5.47}$$

σχέση που είναι μια ολισθημένη κατά π έκδοση της (5.39). Έτσι το υπόλοιπο της προηγούμενης ανάλυσης εφαρμόζεται απευθείας και επομένως

$$G_{ne}(s) = -\frac{\dfrac{s}{(1+K_{dc})}}{s^2 + \dfrac{K_{pc}}{(1+K_{dc})}s + \dfrac{K_{ic}}{(1+K_{dc})}} \tag{5.48}$$

Το αναμενόμενο ακρότατο είναι και πάλι στο

$$\omega = \sqrt{\alpha_d K_{ic}} \tag{5.49}$$

Μη τη μέγιστη τιμή να είναι:

$$\left|G_{ne}(j\sqrt{\alpha_d K_{ic}})\right| = \frac{1}{K_{pc}} \tag{5.50}$$

Η επιλογή των κερδών ξεκινά επίσης με την θεώρηση ότι χρειαζόμαστε μια ελαφρώς αργή απόκριση από τους ελεγκτές PID, για να αποφύγουμε μεγάλα overshoots στις επιταχύνσεις που απαιτούνται από το σύστημά μας. Για το λόγο αυτό, είναι πιο πιθανό να καταλήξουμε σε ένα $\zeta \geq 1$ με αναφορά στο τυπικό σύστημα 2ης τάξης (δες εξίσωση (5.51)).

$$G_{2nd}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.51}$$

Για να εφαρμόσουμε τα παραπάνω συμπεράσματα για τα κέρδη PID, συγκρίνουμε την (5.48) με την εξίσωση για το τυπικό σύστημα 2ας τάξεως $G_{2nd}(s)$.

Αφήνοντας κατά μέρος το μηδενικό στον αριθμητή της (5.48) προς το παρόν, αναγνωρίζουμε τις ακόλουθες εξισώσεις συγκρίνοντας την εξίσωση (5.48) με την (5.51).

$$\omega_n^2 = \frac{K_{ic}}{(1+K_{dc})} \tag{5.52}$$

$$2\zeta\omega_n = \frac{K_{pc}}{(1+K_{dc})} \tag{5.53}$$

Από (5.52) έχουμε ότι:

$$\boxed{\omega_n = \sqrt{\frac{K_{ic}}{(1+K_{dc})}}} \tag{5.54}$$

Με χρήση της (5.54) μαζί με την (5.53) έχουμε ότι

$$2\zeta\sqrt{\frac{K_{ic}}{(1+K_{dc})}} = \frac{K_{pc}}{(1+K_{dc})} \Leftrightarrow$$

$$\boxed{\zeta = \frac{K_{pc}}{2\sqrt{K_{ic}(1+K_{dc})}}} \tag{5.55}$$

Κάτι αξιοσημείωτο είναι ότι συγκρίνοντας την (5.49) (σημείο του ακρότατου) με την (5.54) (τη σχέση του ω_n με τα κέρδη του συστήματος) το αποτέλεσμα είναι:

$$\omega = \sqrt{\alpha_d K_{ic}} = \sqrt{\frac{K_{ic}}{1+K_{dc}}} = \omega_n$$

Το παραπάνω αποτέλεσμα σημαίνει ότι το ακρότατο βρίσκεται στη φυσική συχνότητα του συστήματος.

Ένα αρχικό σύνολο υποθέσεων που μπορεί να εφαρμοστεί για τον βρόχο είναι:

- $\zeta \geq 1$
- K_{pc} όσο το δυνατόν μεγαλύτερο, για να μειώσει την επίδραση του θορύβου, η οποία με τη σειρά της μειώνει την επίδραση των αβεβαιοτήτων μοντελοποίησης, που προκύπτουν από την υπόθεση ότι ο MBC δρα ως τέλειος αποζεύκτης.
- Η αύξηση του K_{pc} έρχεται με κόστος τα περιθώρια ευστάθειας, και αυτό είναι κάτι που θα πρέπει να ληφθεί υπόψη.
- K_{dc} να κρατηθεί μικρό, γιατί ενισχύει την επίδραση του θορύβου.

Οι παραπάνω προτάσεις μπορούν να αποδειχθούν λειτουργώντας αρχικά στην εξίσωση (5.51) (πρότυπο σύστημα 2ας τάξεως [18], [19]) αλλά αυτή τη φορά με μηδενικό στον αριθμητή, προκειμένου να πάρουμε ακριβέστερη συμπεριφορά:

$$G_{2nd\_s}(s) = \frac{\omega_n{}^2 s}{s^2 + 2\zeta\omega_n s + \omega_n{}^2} \tag{5.56}$$

Η παρουσία ενός μηδενικού στο s = 0 στον αριθμητή συνεπάγεται μια απόκριση "AC-coupled", με την έννοια ότι ξεκινά από το μηδέν στις χαμηλές συχνότητες και αυξάνεται κατά 20dB/dec μέχρι να βρει τον πόλο ή τους πόλους του συστήματος. Το σύμβολο – μπορεί να εισαχθεί μαζί με ένα κέρδος K, αλλά προς το παρόν θα προχωρήσουμε στην ανάλυση χωρίς αυτό.

Η ρίζες του παρονομαστή μπορούν να βρεθούν ως εξής:

$$\Delta = (2\zeta\omega_n)^2 - 4\omega_n{}^2 = 4\zeta^2\omega_n{}^2 - 4\omega_n{}^2 \Leftrightarrow$$

$$\boxed{\Delta = 4\omega_n{}^2(\zeta^2 - 1)} \tag{5.57}$$

Ακολουθεί ανάλυση των περιπτώσεων με βάση τις τιμές του ζ και ειδικά τη σχέση τους με το 0 και το 1:

Case 1: ζ < 0

Αυτή η περίπτωση δεν είναι εφικτή αφού τα κέρδη των PID μας είναι θετικά και δεδομένου ότι το ζ δίνεται από την (5.55). Ακόμα κι αν ήταν δυνατό, θα οδηγούσε σε αστάθεια αφού οι ρίζες θα ήταν:

$$s_{1,2} = \frac{2|\zeta|\omega_n \pm \sqrt{4\omega_n{}^2(\zeta^2 - 1)}}{2} = \frac{2|\zeta|\omega_n \pm 2|\omega_n|\sqrt{\zeta^2 - 1}}{2}$$

Εφόσον το ω_n είναι θετικό (είναι τετραγωνική ρίζα, βλέπε (5.54)), αυτό θα οδηγούσε σε τουλάχιστον μία ρίζα με θετικό πραγματικό μέρος, πράγμα που σημαίνει ότι αυτή η περίπτωση απορρίπτεται.

Case 2: 0 < ζ < 1

Στην περίπτωση αυτή έχουμε Δ < 0, που οδηγεί σε δύο μιγαδικές ρίζες με αρνητικό πραγματικό μέρος. Οι ρίζες είναι:

$$s_{1,2} = \frac{-2\zeta\omega_n \pm \sqrt{4\omega_n{}^2(\zeta^2 - 1)}}{2} = -\zeta\omega_n \pm j\omega_n\sqrt{1 - \zeta^2} \Rightarrow$$

$$s_{1,2} = -\zeta\omega_n \pm j\omega_d$$

όπου:

$$\omega_d = \omega_n\sqrt{1 - \zeta^2} \tag{5.58}$$

Για να βρούμε τη βηματική απόκριση του συστήματος εργαζόμαστε ως ακολούθως:

$$Y(s) = G_{2nd\_s}(s) \cdot U(s) = \frac{\omega_n{}^2 \cancel{s}}{(s + \zeta\omega_n - j\omega_d)(s + \zeta\omega_n + j\omega_d)} \cdot \frac{1}{\cancel{s}} \Leftrightarrow$$

$$Y(s) = \frac{\omega_n{}^2}{(s + \zeta\omega_n - j\omega_d)(s + \zeta\omega_n + j\omega_d)} \tag{5.59}$$

Χρησιμοποιώντας την ανάπτυξη σε απλά κλάσματα:

$$Y(s)=\omega_n^{\,2}\left[\frac{A}{(s+\zeta\omega_n-j\omega_d)}+\frac{B}{(s+\zeta\omega_n+j\omega_d)}\right]$$

όπου

$$A=\lim_{s\to(-\zeta\omega_n+j\omega_d)}\left[\cancel{(s+\zeta\omega_n-j\omega_d)}\,\frac{1}{\cancel{(s+\zeta\omega_n-j\omega_d)}\,(s+\zeta\omega_n+j\omega_d)}\right]\Leftrightarrow$$

$$A=\frac{1}{(-\zeta\omega_n+j\omega_d+\zeta\omega_n+j\omega_d)}=\frac{1}{2j\omega_d} \qquad (5.60)$$

και

$$B=\lim_{s\to(-\zeta\omega_n-j\omega_d)}\left[\cancel{(s+\zeta\omega_n+j\omega_d)}\,\frac{1}{(s+\zeta\omega_n-j\omega_d)\,\cancel{(s+\zeta\omega_n+j\omega_d)}}\right]\Leftrightarrow$$

$$B=\frac{1}{(-\zeta\omega_n-j\omega_d+\zeta\omega_n-j\omega_d)}=\frac{1}{-2j\omega_d} \qquad (5.61)$$

Που σημαίνει ότι

$$Y(s)=\frac{\omega_n^{\,2}}{2j\omega_d}\left[\frac{1}{(s+\zeta\omega_n-j\omega_d)}-\frac{1}{(s+\zeta\omega_n+j\omega_d)}\right]\Rightarrow$$

$$y(t)=L^{-1}\{Y(s)\}=\frac{\omega_n^{\,2}}{2j\omega_d}\left[e^{(-\zeta\omega_n+j\omega_d)t}-e^{(-\zeta\omega_n-j\omega_d)t}\right]\Leftrightarrow$$

$$y(t)=\frac{\omega_n^{\,2}e^{-\zeta\omega_n t}}{2j\omega_d}\left[e^{j\omega_d t}-e^{-j\omega_d t}\right]=\frac{\omega_n^{\,2}e^{-\zeta\omega_n t}}{2\omega_d}\left(\frac{e^{j\omega_d t}}{j}-\frac{e^{-j\omega_d t}}{j}\right)=\frac{\omega_n^{\,2}e^{-\zeta\omega_n t}}{2\omega_d}\left(\frac{e^{j\omega_d t}}{e^{j\frac{\pi}{2}}}+\frac{e^{-j\omega_d t}}{e^{-j\frac{\pi}{2}}}\right)\Leftrightarrow$$

$$y(t)=\frac{\omega_n^{\,2}e^{-\zeta\omega_n t}}{2\omega_d}\left[e^{j\left(\omega_d t-\frac{\pi}{2}\right)}+e^{-j\left(\omega_d t-\frac{\pi}{2}\right)}\right]\Rightarrow$$

$$y(t)=\frac{\omega_n^{\,2}e^{-\zeta\omega_n t}}{\cancel{2}\,\omega_d}\,\cancel{2}\,\mathrm{Re}\left[e^{j\left(\omega_d t-\frac{\pi}{2}\right)}\right]\Rightarrow$$

$$y(t)=\frac{\omega_n^{\,2}e^{-\zeta\omega_n t}}{\omega_d}\cos\left(\omega_d t-\frac{\pi}{2}\right)\Rightarrow$$

$$\boxed{y(t)=\frac{\omega_n^{\,2}}{\omega_d}e^{-\zeta\omega_n t}\sin(\omega_d t)} \qquad (5.62)$$

Αυτή είναι μια απόκριση με ταλάντωση που μειώνεται στο χρόνο. Επειδή δεν θέλουμε ταλαντώσεις, πρέπει να υπάρχει ένας πολύ καλός λόγος για να χρησιμοποιήσουμε $0 < \zeta < 1$. Μερικές αποκρίσεις που χρησιμοποιούν μια λογική προσέγγιση που αφορά στο ω_d, π.χ. σε ένα ω_d που παράγει μια ημιτονοειδή απόκριση περιόδου 4 δευτερολέπτων:

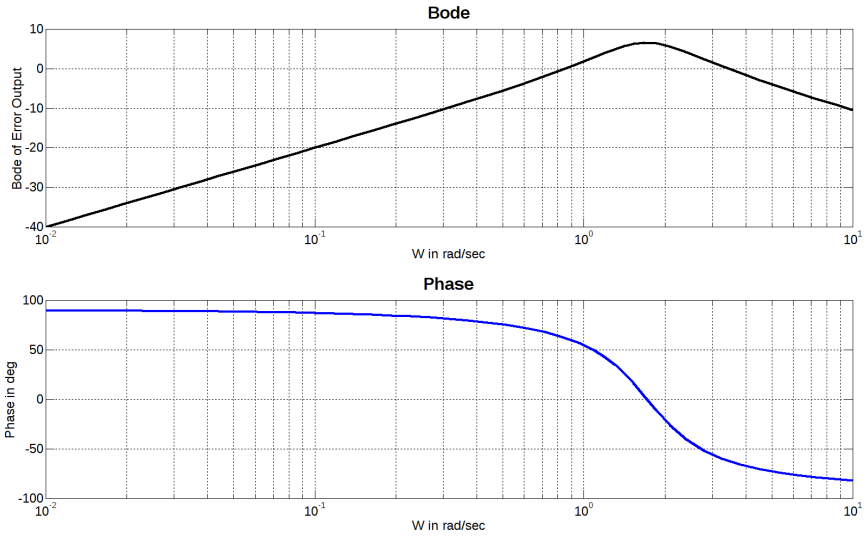
$$\omega_d = \frac{2\pi \text{ rad}}{4 \text{ sec}} = \frac{\pi}{2} \text{rad / sec}$$

Που σημαίνει ότι αν επιλέξουμε ζ = 0.4, τότε έχουμε ω_n :

$$\omega_d = \omega_n \sqrt{1-\zeta^2} \underset{\zeta=0.4}{\overset{\omega_d=\pi/2}{\Rightarrow}} \omega_n = \frac{\pi}{2\sqrt{1-0.4^2}} \Leftrightarrow$$

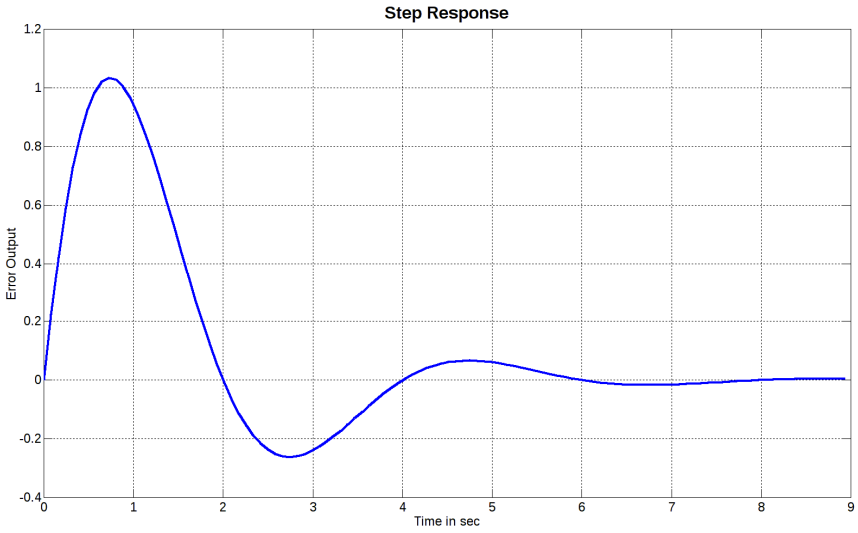
$$\omega_n = 1.7139 \text{ rad/sec}$$

Μερικά παραδείγματα ακολουθούν στις παρακάτω εικόνες.



Εικόνα 29. Bode διάγραμμα για ζ=0.4, ωn=1.7139, ωd=π/2.

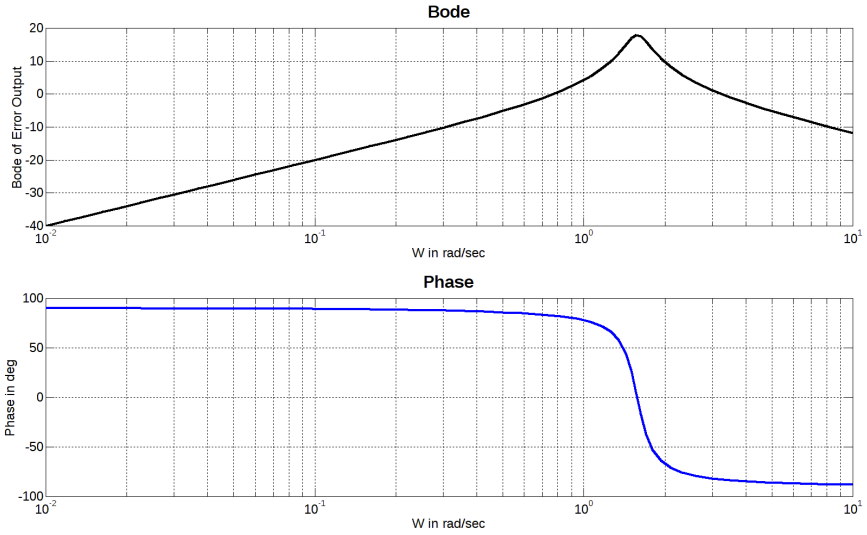
Το πρώτο που πρέπει να ειπωθεί για αυτήν την περίπτωση, είναι ότι έχει μια περιοχή όπου ενισχύει τον θόρυβο. Λόγω του γεγονότος ότι το πραγματικό μέρος των μιγαδικών ριζών είναι το ίδιο (ζεύγος μιγαδικών ριζών), το διάγραμμα Bode έχει ένα μέγιστο και αρχίζει να πέφτει μετά από αυτό.



Εικόνα 30. Βηματική απόκριση για ζ=0.4, ωn=1.7139, ωd=π/2.

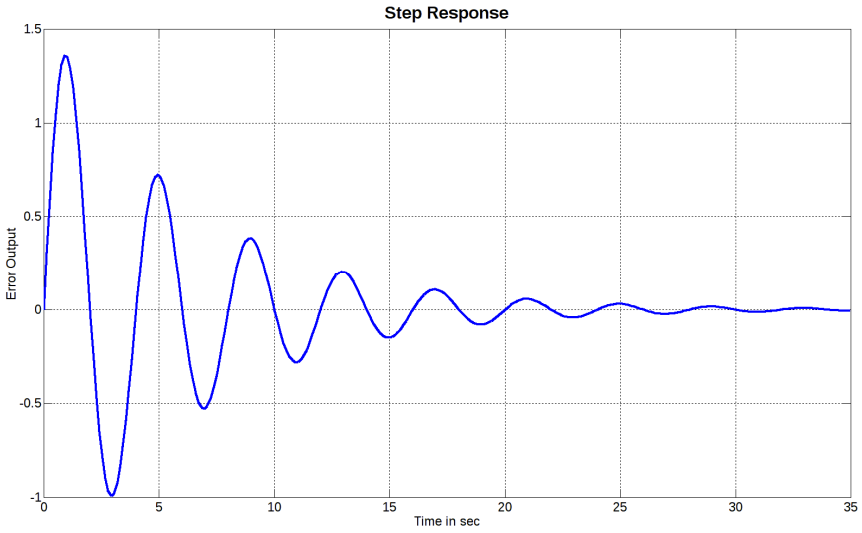
Η βηματική απόκριση πηγαίνει ελαφρώς πάνω από το 1 όπως αναμένεται από το διάγραμμα Bode και εμφανίζει κάποιες ταλαντώσεις.

Η βηματική απόκριση πηγαίνει ελαφρώς πάνω από το 1 όπως αναμενόταν από το διάγραμμα Bode και εμφανίζει κάποιες ταλαντώσεις.



Εικόνα 31. Bode διάγραμμα για ζ=0.1, ωn=1.7139, ωd=π/2.

Η ενίσχυση του θορύβου σε μια συγκεκριμένη κορυφή είναι ακόμη μεγαλύτερη και η βηματική απόκριση υφίσταται πολλές ταλαντώσεις.



Εικόνα 32. Βηματική απόκριση για ζ=0.1, ωn=1.7139, ωd=π/2.

Επομένως ακόμα και αν αποφασίσουμε να κρατήσουμε το ζ κάτω από 1, θα πρέπει να είμαστε κοντά σε αυτό.

Case 3: ζ = 1

Στην περίπτωση αυτή έχουμε:

$$G_{2nd}(s)=\frac{\omega_n^{\,2}s}{s^2+2\omega_n s+\omega_n^{\,2}} \Leftrightarrow G_{2nd}(s)=\frac{\omega_n^{\,2}s}{\left(s+\omega_n\right)^2}$$

Που σημαίνει δύο πραγματικές ρίζες μαζί στο ίδιο σημείο.

Η βηματική απόκριση ενός τέτοιου συστήματος μπορεί να βρεθεί ως εξής:

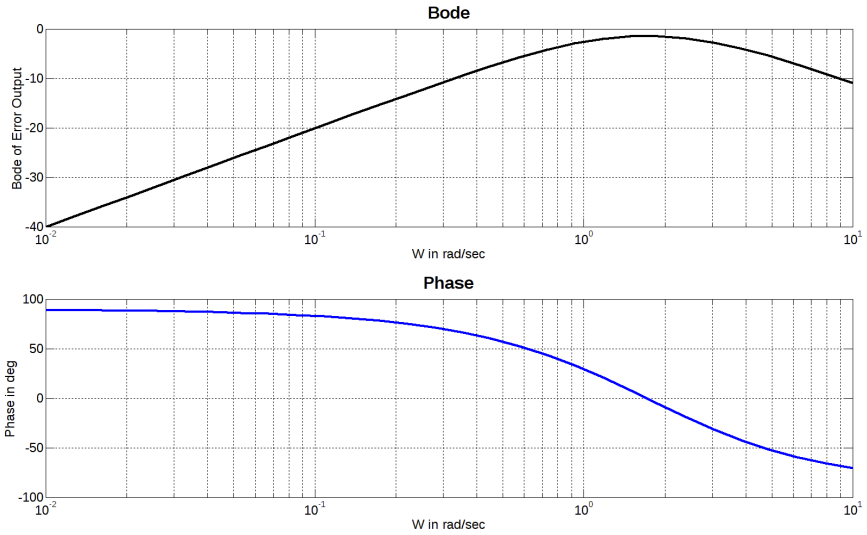
$$Y(s)=G_{2nd\_s}(s)\cdot U(s)=\frac{\omega_n^{\,2}\cancel{s}}{\left(s+\omega_n\right)^2}\cdot\frac{1}{\cancel{s}} \Leftrightarrow$$

$$Y(s)=\frac{\omega_n^{\,2}}{\left(s+\omega_n\right)^2} \Rightarrow$$

$$\boxed{y(t)=\omega_n^{\,2}te^{-\omega_n t}} \qquad (5.63)$$

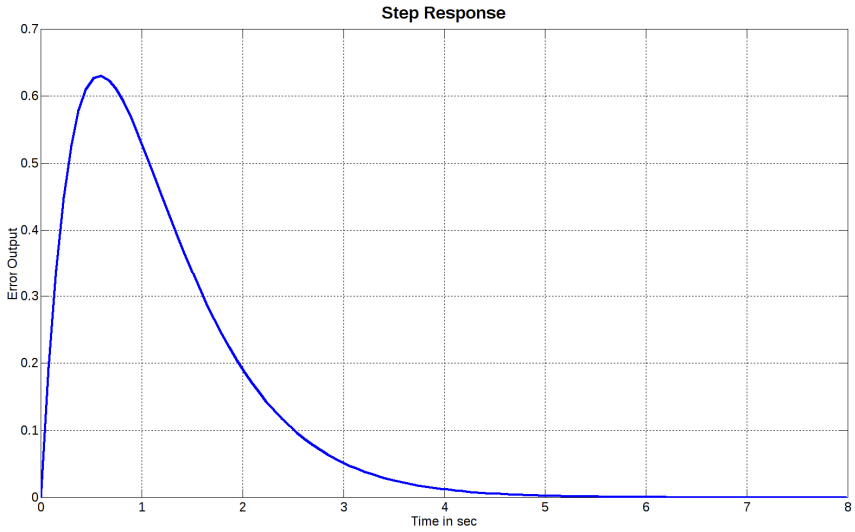
Αφού από τον αντίστροφο μετασχηματισμό Laplace έχουμε ότι:

$$L^{-1}\left\{\frac{n!}{(s-a)^{n+1}}\right\}=te^{at}, n=1,2,3,\ldots$$



Εικόνα 33. Bode διάγραμμα για ζ=1, ωn=1.7139.

Όπως φαίνεται από το διάγραμμα Bode, ο θόρυβος μόνο μειώνεται από το σύστημα στο ζ = 1. Δεν υπάρχει κέρδος σε καμία συχνότητα. Η βηματική απόκριση δεν έχει ταλαντώσεις και μηδενίζεται σε χρόνο κοντά στο ζ=0,4 (περίπου στα 6 δευτερόλεπτα η απόκριση είναι σχεδόν μηδενική).



Εικόνα 34. Βηματική απόκριση για ζ=1, ωn=1.7139.

Αυτή είναι μια αποδεκτή συμπεριφορά, η οποία ενισχύεται επίσης από το γεγονός ότι οι ρίζες του χαρακτηριστικού πολυωνύμου συγκεντρώνονται στον πραγματικό άξονα, δημιουργώντας έτσι εξασθένηση -40 dB/dec ακριβώς μετά από αυτές. Επομένως δεν υπάρχει επίπεδο στο μέγιστο της απόκρισης σφάλματος στο θόρυβο.

Case 4: ζ > 1

Στην περίπτωση αυτή έχουμε δύο πραγματικές ρίζες:

$$\Delta = 4\omega_n^2\left(\zeta^2-1\right) > 0$$

$$s_{1,2} = \frac{-2\zeta\omega_n \pm \sqrt{4\omega_n^2\left(\zeta^2-1\right)}}{2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2-1}$$

Είναι ενδιαφέρον να διερευνηθεί εάν μπορεί να βρεθεί ένα ζ για το οποίο μια ρίζα θα είναι θετική.

Αν υποθέσουμε ότι $s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2-1} > 0 \Leftrightarrow$

$\omega_n\sqrt{\zeta^2-1} > \zeta\omega_n \Leftrightarrow \sqrt{\zeta^2-1} > \zeta \overset{\zeta>0}{\Rightarrow} \zeta^2-1 > \zeta^2 \Leftrightarrow -1>0$ κάτι που είναι άτοπο.

Έτσι, έχουμε πάντα πραγματικές αρνητικές ρίζες. Στο θεωρητικό πολύ μεγάλο ζ, έχουμε ότι

$$s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2-1} \underset{\sqrt{\zeta^2-1}\approx\zeta}{\overset{\zeta>>1}{\approx}} -\zeta\omega_n + \omega_n\zeta = 0$$

Που σημαίνει ότι η ρίζα πλησιάζει στο μηδέν.

Η βηματική απόκριση είναι:

$$Y(s) = G_{2nd\_s}(s)\cdot U(s) = \frac{\omega_n^2}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\cdot\frac{1}{s} \Rightarrow$$

$$Y(s) = \omega_n^2\left[\frac{A}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)} + \frac{B}{\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right] \text{ όπου}$$

$$A = \lim_{s\to\left(-\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\left[\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)\frac{1}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right] \Leftrightarrow$$

$$A = \frac{1}{-\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}} \Leftrightarrow$$

$$A = \frac{1}{2\omega_n\sqrt{\zeta^2-1}}$$

$$B = \lim_{s\to\left(-\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)}\left[\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)\frac{1}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right] \Leftrightarrow$$

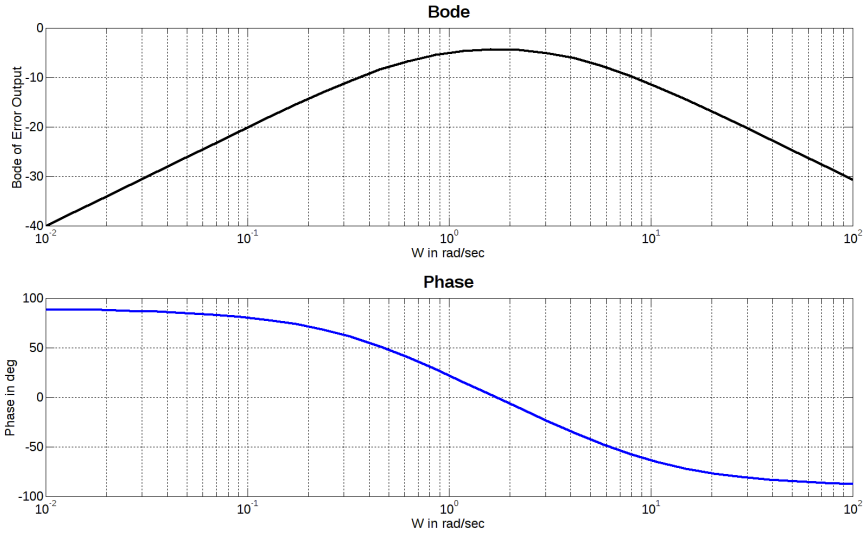
$$B = \frac{1}{-2\omega_n\sqrt{\zeta^2-1}}$$

Που μας οδηγεί στο

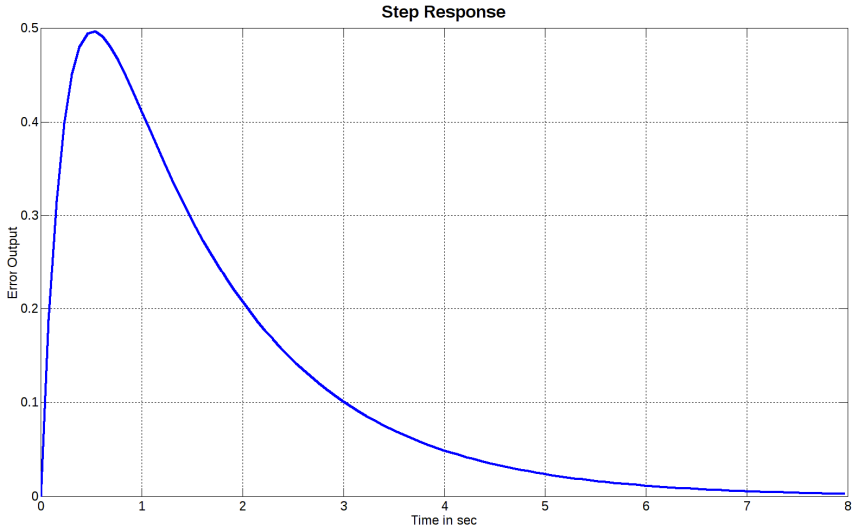
$$Y(s) = \frac{\omega_n^2}{2\omega_n\sqrt{\zeta^2-1}}\left[\frac{1}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)} - \frac{1}{\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right] \Rightarrow$$

$$y(t)=\frac{\omega_n}{2\sqrt{\zeta^2-1}}\left[e^{\left(-\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)t}-e^{\left(-\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)t}\right]\Leftrightarrow$$

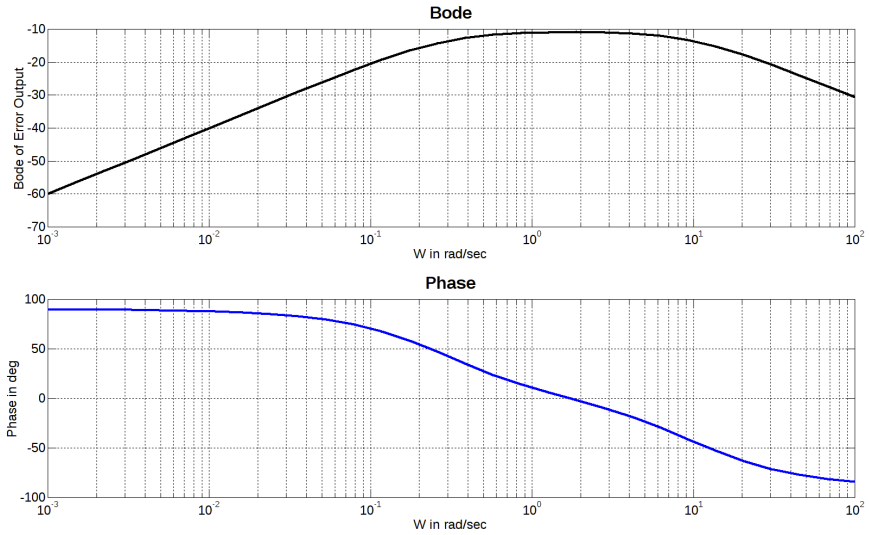
$$\boxed{y(t)=\frac{\omega_n}{2\sqrt{\zeta^2-1}}e^{-\zeta\omega_n t}\left[e^{\left(\omega_n\sqrt{\zeta^2-1}\right)t}-e^{-\left(\omega_n\sqrt{\zeta^2-1}\right)t}\right]} \quad (5.64)$$



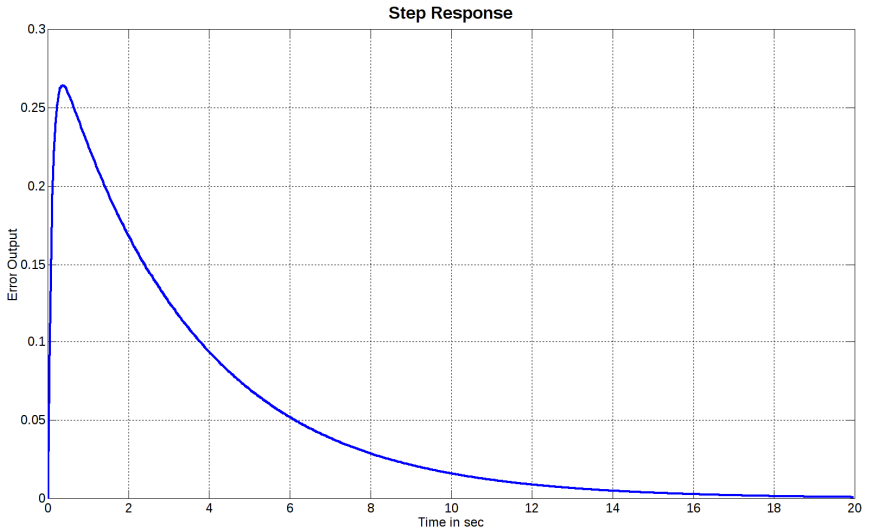
Εικόνα 35. Διάγραμμα Bode για ζ=1.4, ωn=1.7139.



Εικόνα 36. Βηματική απόκριση για ζ=1.4, ωn=1.7139.



Εικόνα 37. Bode διάγραμμα για ζ=3, ωn=1.7139.



Εικόνα 38. Βηματική απόκριση για ζ=3, ωn=1.7139

Σε αυτό το σημείο, πρέπει να ληφθεί μια απόφαση σχετικά με τη συνιστώμενη απόκριση, από την οποία θα εξαχθούν τα κέρδη PID.

Για να κάνουμε την ανάλυση ακριβή, επαναλαμβάνουμε εδώ την εξίσωση (5.48), μαζί με μία ακριβή παραλλαγή της (5.56) προσαρμοσμένη στην περίπτωσή μας:

$$G_{ne}(s) = -\frac{\dfrac{s}{(1+K_{dc})}}{s^2 + \dfrac{K_{pc}}{(1+K_{dc})}s + \dfrac{K_{ic}}{(1+K_{dc})}} \qquad (5.65)$$

$$G_{2nd\_s1}(s)=\frac{K\omega_n^{\,2}s}{s^2+2\zeta\omega_n s+\omega_n^{\,2}} \tag{5.66}$$

Όπου K είναι ένας αρνητικός αριθμός (K=-|K|).

$$\omega_n^{\,2}=\frac{K_{ic}}{(1+K_{dc})}\Leftrightarrow$$

$$\boxed{\omega_n=\sqrt{\frac{K_{ic}}{(1+K_{dc})}}} \tag{5.67}$$

$$2\zeta\omega_n=\frac{K_{pc}}{(1+K_{dc})}\Rightarrow 2\zeta\sqrt{\frac{K_{ic}}{(1+K_{dc})}}=\frac{K_{pc}}{(1+K_{dc})}\Leftrightarrow 2\zeta\frac{\sqrt{K_{ic}}}{\sqrt{(1+K_{dc})}}=\frac{K_{pc}}{(1+K_{dc})}\Leftrightarrow$$

$$2\zeta=\frac{K_{pc}}{\left(\sqrt{K_{ic}}\right)\cdot\sqrt{(1+K_{dc})}}\Leftrightarrow$$

$$\boxed{\zeta=\frac{K_{pc}}{2\sqrt{K_{ic}(1+K_{dc})}}} \tag{5.68}$$

$$K\omega_n^{\,2}=-\frac{1}{(1+K_{dc})}\Rightarrow K\frac{K_{ic}}{\cancel{(1+K_{dc})}}=-\frac{1}{\cancel{(1+K_{dc})}}\Leftrightarrow$$

$$\boxed{K=-\frac{1}{K_{ic}}} \tag{5.69}$$

Θα διερευνηθούν δύο περιπτώσεις: μία με ζ > 1 και μία άλλη με ζ = 1.
Κάνοντας μια ακριβή ανάλυση για την κατάσταση ισορροπίας της (5.66), έχουμε ότι:

$$G_{2nd\_s1}(s)=\frac{K\omega_n^{\,2}s}{s^2+2\zeta\omega_n s+\omega_n^{\,2}}\Rightarrow G_{2nd\_s1}(j\omega)=\frac{K\omega_n^{\,2}j\omega}{(j\omega)^2+2\zeta\omega_n j\omega+\omega_n^{\,2}}\Leftrightarrow$$

$$G_{2nd\_s1}(j\omega)=\frac{K\omega_n^{\,2}j\omega}{\omega_n^{\,2}-\omega^2+2j\zeta\omega_n\omega} \tag{5.70}$$

$$\left|G_{2nd\_s1}(j\omega)\right|=\frac{\sqrt{\left(K\omega_n^{\,2}\omega\right)^2}}{\sqrt{\left(\omega_n^{\,2}-\omega^2\right)^2+(2\zeta\omega_n\omega)^2}}\Leftrightarrow$$

$$\left|G_{2nd\_s1}(j\omega)\right|=\frac{\left|K\omega_n^{\,2}\omega\right|}{\sqrt{\left(\omega_n^{\,2}-\omega^2\right)^2+(2\zeta\omega_n\omega)^2}} \tag{5.71}$$

Μελετώντας το $\left|G_{2nd\_s1}(j\omega)\right|^2$ για ακρότατο όπως κάναμε προηγουμένως, θα έχουμε ότι

$$\left|G_{2nd\_s1}(j\omega)\right|^2=\frac{\left(K\omega_n^{\,2}\omega\right)^2}{\left(\omega_n^{\,2}-\omega^2\right)^2+(2\zeta\omega_n\omega)^2} \tag{5.72}$$

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=\frac{2\left(K\omega_n{}^2\omega\right)K\omega_n{}^2\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]-\left(K\omega_n{}^2\omega\right)^2\left[2\left(\omega_n{}^2-\omega^2\right)(-2\omega)+2\left(2\zeta\omega_n\omega\right)2\zeta\omega_n\right]}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}=$$

$$=\frac{2K^2\omega_n{}^4\omega\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]-\left(K\omega_n{}^2\omega\right)^2\left[2\left(\omega_n{}^2-\omega^2\right)(-2\omega)+2\left(2\zeta\omega_n\omega\right)2\zeta\omega_n\right]}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}=$$

$$=\frac{2K^2\omega_n{}^4\omega\left(\omega_n{}^4-2\omega_n{}^2\omega^2+\omega^4+4\zeta^2\omega_n{}^2\omega^2\right)-K^2\omega_n{}^4\omega^2\left[-4\omega_n{}^2\omega+4\omega^3+8\zeta^2\omega_n{}^2\omega\right]}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}=$$

$$=\frac{2K^2\omega_n{}^8\omega-\cancel{4K^2\omega_n{}^6\omega^3}+2K^2\omega_n{}^4\omega^5+\cancel{8K^2\zeta^2\omega_n{}^6\omega^3}+\cancel{4K^2\omega_n{}^6\omega^3}-4K^2\omega_n{}^4\omega^5-\cancel{8K^2\zeta^2\omega_n{}^6\omega^3}}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=\frac{2K^2\omega_n{}^8\omega-2K^2\omega_n{}^4\omega^5}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=\frac{2K^2\omega_n{}^4\omega\left(\omega_n{}^4-\omega^4\right)}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2} \qquad (5.73)$$

Για να βρούμε το ακρότατο (που ξέρουμε εκ των προτέρων ότι θα είναι το μέγιστο), έχουμε

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=0\Rightarrow$$

$$2K^2\omega_n{}^4\omega\left(\omega_n{}^4-\omega^4\right)=0\Rightarrow$$

$\omega=0$ or $\omega_n{}^4=\omega^4\Rightarrow$

$$\boxed{\omega_{max}=\omega_n} \qquad (5.74)$$

Από τη στιγμή που εξετάζουμε για θετικά ω. Η μέγιστη τιμή είναι [20]

$$\left\|G_{2nd\_sl}\right\|_\infty=\frac{\left|K\omega_n{}^3\right|}{\sqrt{\left(2\zeta\omega_n{}^2\right)^2}}=\frac{\left|K\omega_n{}^3\right|}{\left|2\zeta\omega_n{}^2\right|}=\frac{|K|\omega_n{}^3}{2\zeta\omega_n{}^2}\Leftrightarrow$$

$$\boxed{\left\|G_{2nd\_sl}\right\|_\infty=\frac{|K|\omega_n}{2\zeta}} \qquad (5.75)$$

Εξετάζοντας την (5.75) συμπεραίνουμε ότι για να εξασθενήσουμε το θόρυβο στο σύστημά μας, θα πρέπει να φροντίσουμε ώστε το ω_n να είναι μικρό και το ζ μεγάλο, αλλά κάθε επιλογή έχει και τα προβλήματά της. Μειώνοντας πολύ το ω_n θα έχουμε ένα πολύ αργής απόκρισης σύστημα, κάτι που γίνεται και για μεγάλο ζ. Την ίδια στιγμή, αφού οι ρίζες του χαρακτηριστικού πολυωνύμου είναι

$s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}$ για $\zeta \geq 1$, παρατηρούμε ότι περιορίζουμε τη σχετική ευστάθεια είτε μειώνοντας το ω_n είτε αυξάνοντας το ζ!

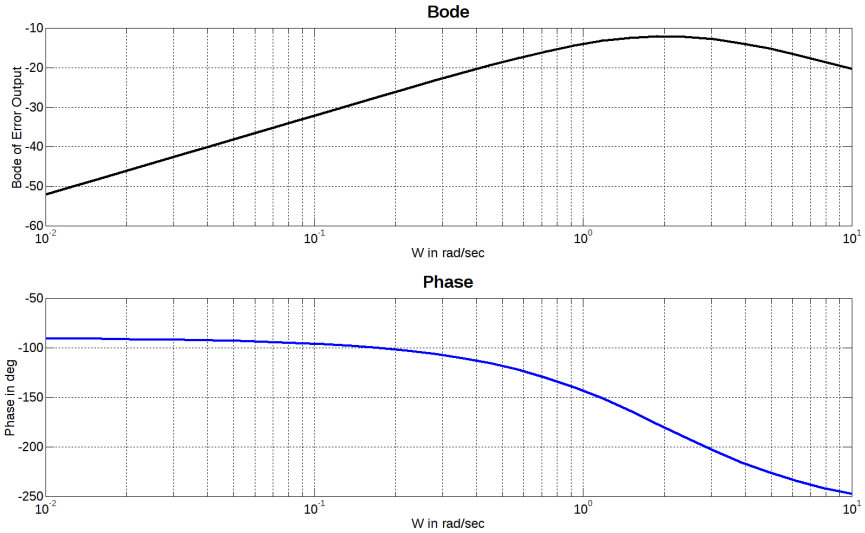
5.5.3 Περίπτωση ζ = 1 (για Supervisory Controller PID)

Εάν σταθεροποιήσουμε το ζ στο 1, μετά από πειραματισμούς με το ω_n, καταλήγουμε σε $\omega_n = 2$. Η συνάρτηση μεταφοράς από θόρυβο σε σφάλμα που προκύπτει από αυτή την επιλογή είναι:

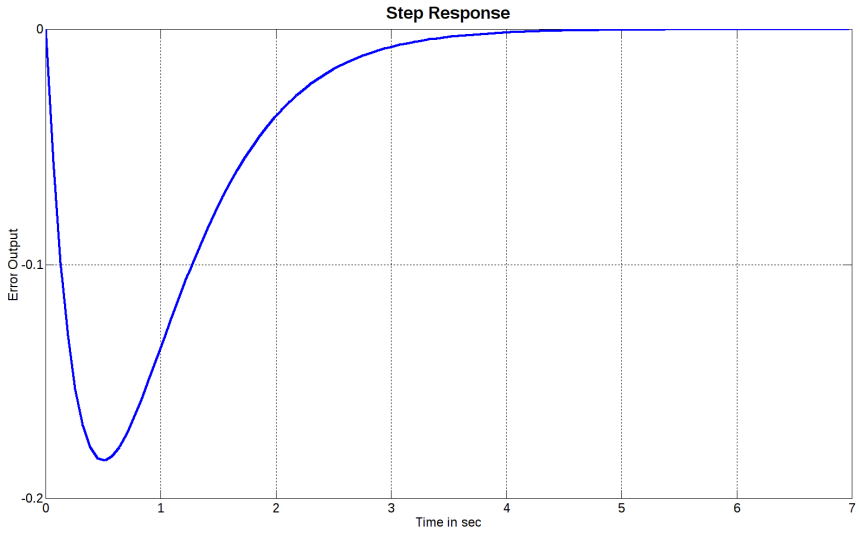
$$G_{ne(\zeta=1,\omega_n=2)}(s) = \frac{K\omega_n^2 s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \Rightarrow$$

$$G_{ne(\zeta=1,\omega_n=2)}(s) = -\frac{s}{s^2 + 4s + 4} \quad (5.76)$$

Οι επιλογές αυτές δίνουν τα παρακάτω αποτελέσματα αναφορικά με την απόκριση συχνότητας και τη βηματική:



Εικόνα 39. Bode διάγραμμα συνάρτησης μεταφοράς από θόρυβο σε σφάλμα για ζ = 1, ωn = 2 rad/sec.



Εικόνα 40. Βηματική απόκριση συνάρτησης μεταφοράς από θόρυβο στο σφάλμα για ζ = 1, ωn = 2 rad/sec.

Όπως αναμενόταν από την (5.74) το μέγιστο κέρδος της συνάρτησης μεταφοράς από το θόρυβο στο σφάλμα είναι για $\omega = \omega_n = 2\,rad/\sec$. Για να επιβεβαιώσουμε σχετικά με το ακρότατο της συνάρτησης μεταφοράς, θα πρέπει πρώτα να αποσαφηνιστούν όλες οι σταθερές που θα χρησιμοποιηθούν.

Είναι προτιμότερο για την περίπτωσή μας να αποφύγουμε τον D όρο, ει δυνατόν. Αυτό σημαίνει ότι θα ξεκινήσουμε με:

$K_{dc} = 0$

Χρησιμοποιώντας την (5.54) με $K_{dc} = 0$ έχουμε ότι:

$K_{ic} = \omega_n{}^2 = 4$

Με χρήση της (5.55) παίρνουμε:

$K_{pc} = 2\zeta\omega_n = 4$

Τελικά, χρησιμοποιώντας την (5.69) έχουμε ότι

$$K = -\frac{1}{K_{ic}} = -\frac{1}{4}$$

Επομένως από (5.75) έχουμε ότι:

$$\left\| G_{ne(\zeta=1,\omega_n=2)} \right\|_\infty = \frac{|K|\omega_n}{2\zeta} = \frac{\frac{1}{4}x2}{2} = \frac{1}{4} \Rightarrow$$

$$\left\| G_{ne(\zeta=1,\omega_n=2)} \right\|_\infty = 20\log\left(\frac{1}{4}\right) \simeq -12\text{dB}$$

Κάτι που φαίνεται από τις παραπάνω καμπύλες.

Με αυτά τα PID gains, η συνάρτηση μεταφοράς G_{rc} γίνεται (δες (5.36)):

$$G_{rc(\zeta=1,\omega_n=2)}(s)=\frac{4s+4}{s^2+4s+4}$$

Για αυτή τη συνάρτηση μεταφοράς έχουμε τα διαγράμματα Bode και βηματικής απόκρισης των Εικόνα 41. και Εικόνα 42..

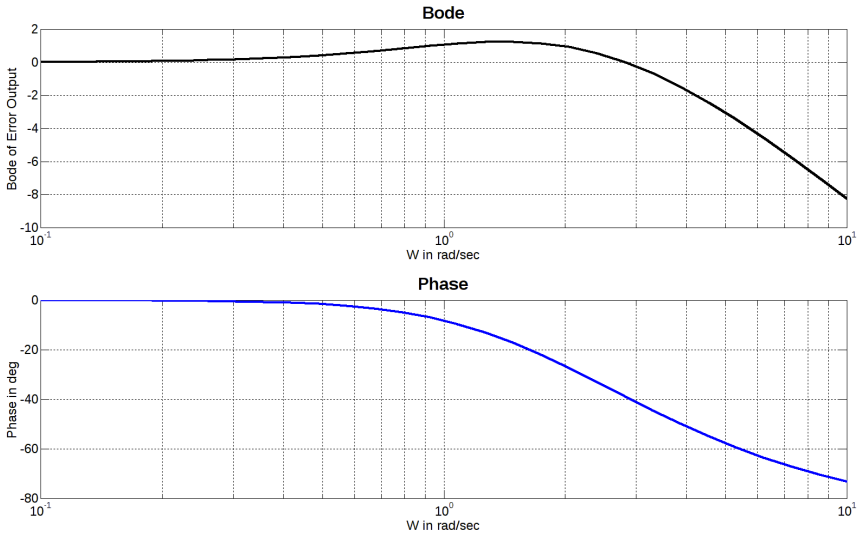
Υλοποιώντας και εξομοιώνοντας την $G_{rc(\zeta=1,\omega_n=2)}$ στην ψηφιακή της μορφή, θα έχουμε για τις ψηφιακές υλοποιήσεις που παρουσιάζονται σε αυτό το κείμενο (δηλαδή positional form και velocity form):

$$\mathrm{C_{pform1}}(z)=\frac{0.12z^{-1}-0.1164z^{-2}}{1-1.88z^{-1}+0.8836z^{-2}}\mathrm{R_{ref}}(z) \tag{5.77}$$

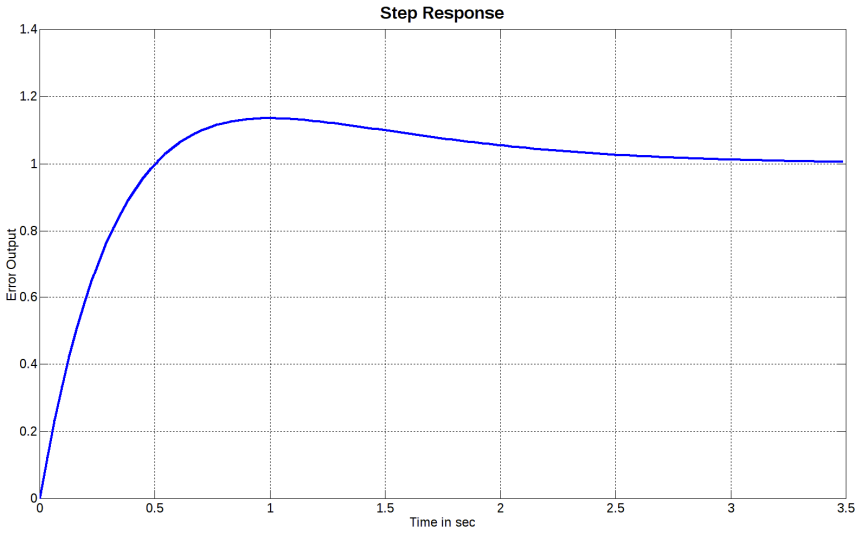
Με αντίστροσο μετασχηματισμό Z αυτό μεταφράζεται σε

$$\begin{aligned}\mathrm{c_{pform1}}[nT_s]=0.12r_{ref}[(n-1)T_s]-0.1164r_{ref}[(n-2)T_s]+\\ 1.88c[(n-1)T_s]-0.8836c[(n-2)T_s]\end{aligned} \tag{5.78}$$

Για το διακριτό χρόνο. Για να οδηγηθούμε στις εξισώσεις αυτές, βασιστήκαμε στην (5.32).



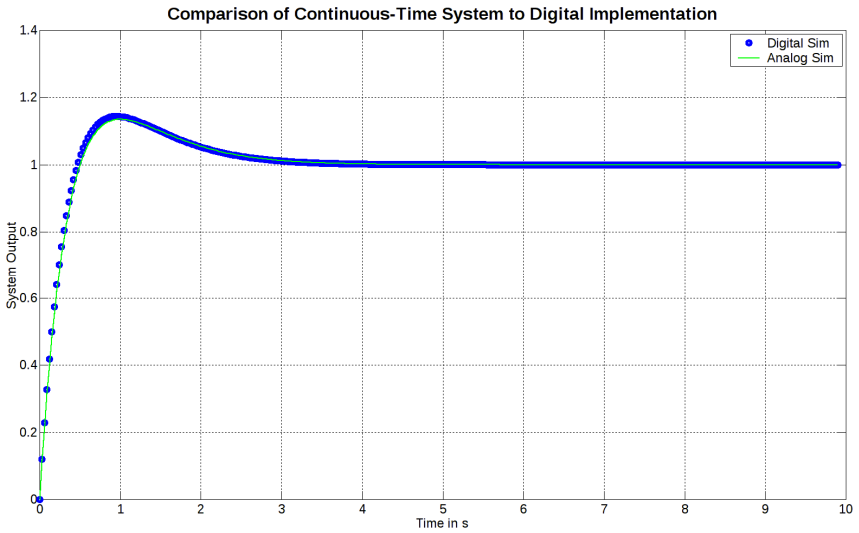
Εικόνα 41. Διάγραμμα Bode της συνάρτησης μεταφοράς από την είσοδο αναφοράς στην έξοδο για ζ = 1, ωn = 2 rad/sec.



Εικόνα 42. Βηματική απόκριση της συνάρτησης μεταφοράς από την είσοδο αναφοράς στην έξοδο για ζ = 1, ωn = 2 rad/sec.

Η απόκριση της υλοποίησης της positional form του συστήματος που περιγράφεται από την εξίσωσης (5.78) ακολουθεί στη Εικόνα 43.. Η ψηφιακή απόκριση παρουσιάζεται σε αντιπαράθεση με την αναλογική (πράσινη γραμμή). Οι δύο αποκρίσεις είναι σχεδόν ταυτόσημες. Ωστόσο και οι δύο υποφέρουν από overshoot, κάτι που θα αντιμετωπιστεί με τη χρήση της εφαρμογής velocity form.

Όπως αναφέρθηκε ήδη, οι υλοποιήση με velocity form παρουσιάζει μια πιο φιλτραρισμένη απόκριση σε σύγκριση με τις αντίστοιχες υλοποιήσεις positional form.



Εικόνα 43. Απόκριση της υλοποίησης με positional form PID του συστήματος με ζ = 1, ωn = 2 rad/sec.

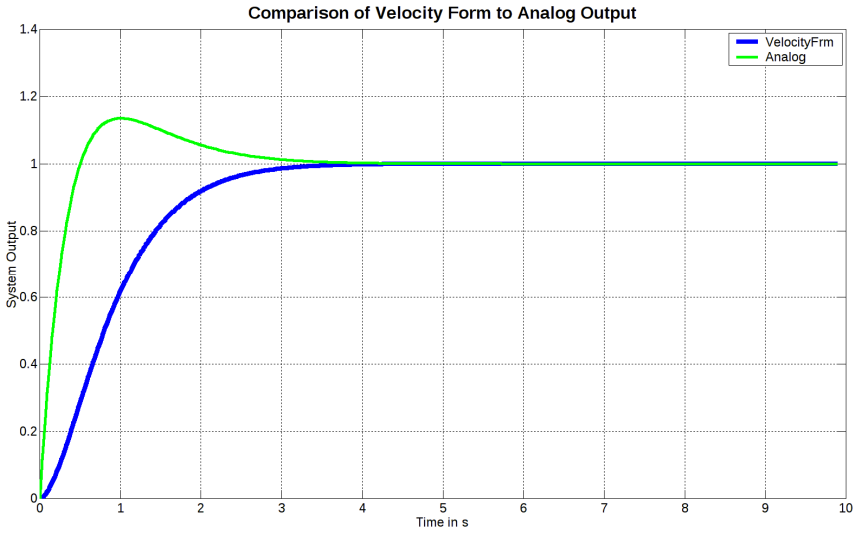
Η velocity form υλοποίηση χρησιμοποιεί τις εξισώσεις:

$$C_{vform1}(z) = \frac{0.0036z^{-1}}{1-1.88z^{-1}+0.8836z^{-2}} R_{ref}(z) \tag{5.79}$$

ή στο διακριστό χρόνο με εξισώσεις διαφορών (βασισμένη στην (5.33):

$$\begin{aligned} c_{vform1}[nT_s] = 0.0036 r_{ref}[(n-1)T_s] + \\ 1.88c[(n-1)T_s] - 0.8836c[(n-2)T_s] \end{aligned} \tag{5.80}$$

Οι παραπάνω εξισώσεις δίνουν τις αποκρίσεις της Εικόνα 44..



Εικόνα 44. Απόκριση της ψηφιακής velocity form υλοποίησης του συστήματος για ζ = 1, ωn = 2 rad/sec.

Όπως γίνεται αντιληπτό από αυτή την απόκριση, το overshoot έχει περισταλλεί. Έτσι, αυτός ο τύπος υλοποίησης θα χρησιμοποιηθεί για τον PID του εποπτικού ελεγκτή (supervisory controller).

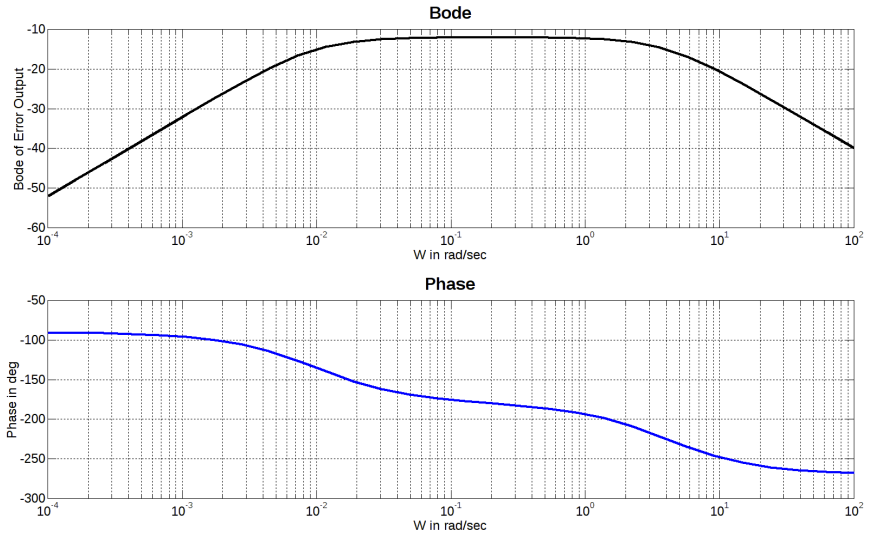
5.5.4 Περίπτωση ζ = 10 (για MBC)

Για να προσαρμοστεί γρήγορα στις εντολές του εποπτικού ελεγκτή, ο MBC χρειάζεται γρήγορη απόκριση, σε σύγκριση με τον SC, χωρίς overshoot. Για το λόγο αυτό, η επιλογή για βρόχους MBC PID έχει επιλεγεί με ζ = 10 και ω_n = 0,2. Η συνάρτηση μεταφοράς από θόρυβο σε σφάλμα που προκύπτει από αυτή την επιλογή είναι:

$$G_{ne(\zeta=10,\omega_n=0.2)}(s) = \frac{K\omega_n^2 s}{s^2+2\zeta\omega_n s+\omega_n^2} \Rightarrow$$

$$G_{ne(\zeta=10,\omega_n=0.2)}(s) = -\frac{s}{s^2+4s+0.04} \tag{5.81}$$

Αυτές οι επιλογές δίνουν τις ακόλουθες αποκρίσεις συχνότητας και σε βηματική είσοδο:



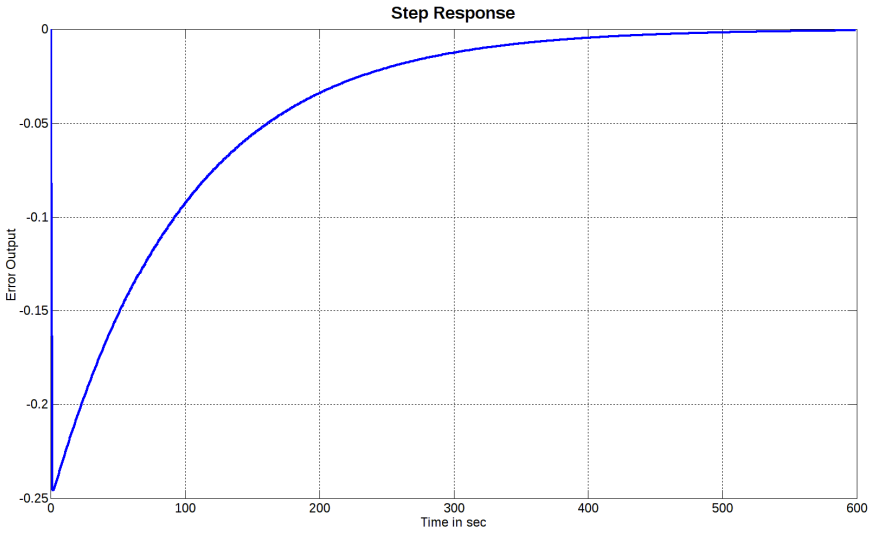
Εικόνα 45. Διάγραμμα Bode συνάρτησης μεταφοράς από το θόρυβο στο σφάλμα για ζ = 10, ωn = 0.2 rad/sec.

Από τη στιγμή που έχουμε ζ > 1, έχουμε δύο πόλους στον πραγματικό άξονα, οι οποίοι καταλήγουν σε μια επιπεδότητα στην κορυφή του γραφήματος απολαβής, μέχρι να βρεθεί ο δεύτερος πόλος. Μετά τη συχνότητα του δεύτερου πόλου, το κέρδος αρχίζει να πέφτει, κάτι που εκμεταλλευόμαστε για να μειώσουμε τον θόρυβο που προκύπτει από την προσέγγιση του ελεγκτή MBC στο σημείο ρύθμισης μας. Αυτός ο θόρυβος αναμένεται να είναι υψηλής συχνότητας σε σύγκριση με τις θέσεις των πόλων, καθώς η μεταβολή του σφάλματος θα είναι μία φορά ανά περίοδο δειγματοληψίας T_s.

Ακολουθώντας παρόμοια ανάλυση με την παράγραφο 5.5.3 έχουμε ότι προτιμάμε $K_{dc} = 0$

Χρησιμοποιώντας την (5.54) με $K_{dc} = 0$ έχουμε ότι:

$K_{ic} = \omega_n^{\,2} = 0.04$



Εικόνα 46. Βηματική απόκριση συνάρτησης μεταφοράς από το θόρυβο στο σφάλμα για ζ = 10, ωn = 0.2 rad/sec.

Χρησιμοποιώντας την (5.55) έχουμε:

$$K_{pc} = 2\zeta\omega_n = 4$$

Από (5.69) έχουμε ότι

$$K = -\frac{1}{K_{ic}} = -\frac{1}{0.04}$$

Από (5.75) έχουμε:

$$\left\| G_{ne(\zeta=10,\omega_n=0.2)} \right\|_\infty = \frac{|K|\omega_n}{2\zeta} = \frac{\frac{1}{0.04}\text{x}0.2}{2\text{x}10} = \frac{1}{4} \Rightarrow$$

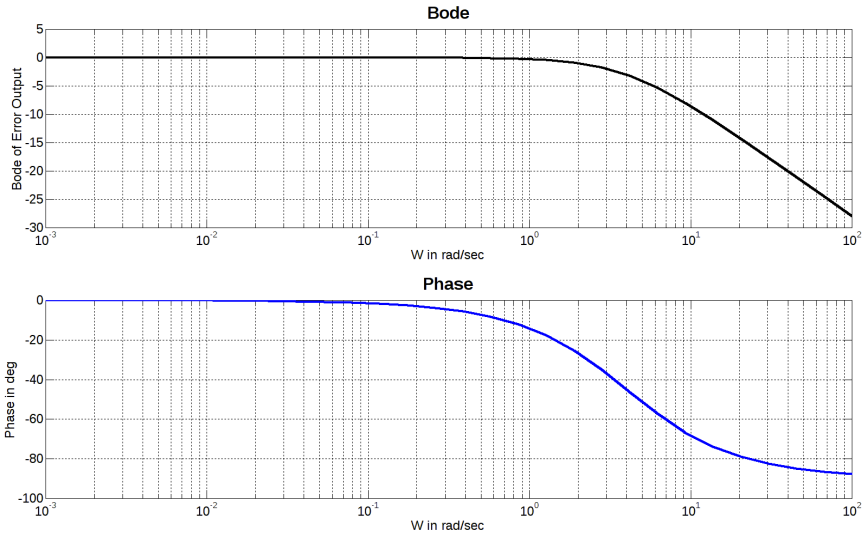
$$\left\| G_{ne(\zeta=10,\omega_n=0.2)} \right\|_\infty = 20\log\left(\frac{1}{4}\right) \simeq -12\text{dB}$$

Κάτι που επαληθεύεται και από τις παραπάνω καμπύλες που παρήχθησαν από το MATLAB.

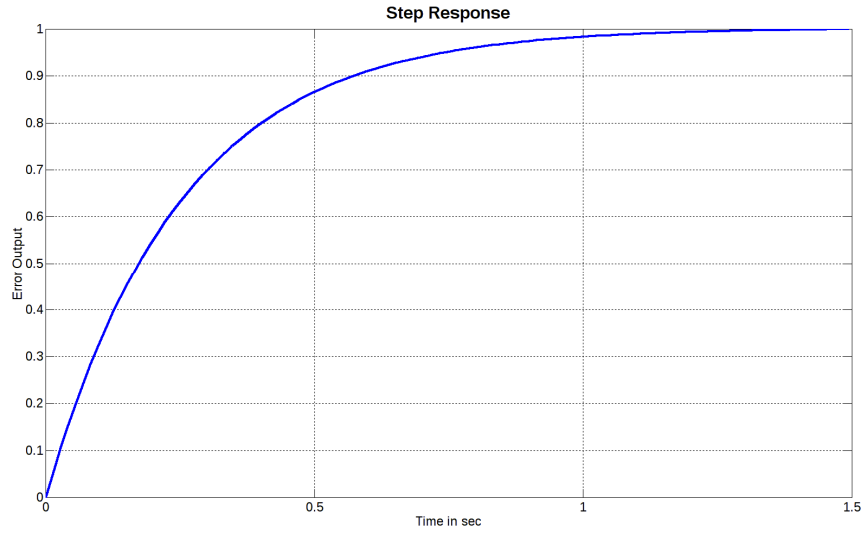
Με αυτές τις ρυθμίσεις για τα PID gains, η συνάρτηση μεταφοράς G_{rc} γίνεται (δες (5.36)):

$$G_{rc(\zeta=10,\omega_n=0.2)}(s) = \frac{4s+0.04}{s^2+4s+0.04} \qquad (5.82)$$

Για αυτή τη συνάρτηση μεταφοράς έχουμε τα διαγράμματα απόκρισης Bode και βήματος των Εικόνα 47. και Εικόνα 48..



Εικόνα 47. Διάγραμμα Bode για τη συνάρτηση μεταφοράς από την είσοδο αναφοράς στην έξοδο για ζ = 10, ωn = 0.2 rad/sec.



Εικόνα 48. Βηματική απόκριση της συνάρτησης μεταφοράς από την είσοδο αναφοράς στην έξοδο για ζ = 10, ωn = 0.2 rad/sec.

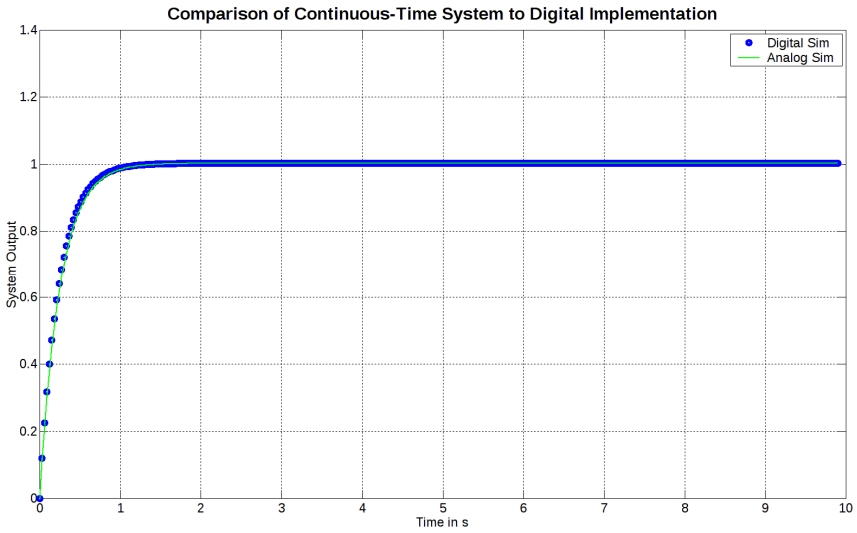
Υλοποιώντας και εξομοιώνοντας την $G_{rc(\zeta=10,\omega_n=0.2)}$ στον διακριτό χρόνο, έχουμε για τις υλοποιήσεις διακριτού χρόνου που παρουσιάστηκαν παραπάνω στο κείμενο (positional form και velocity form):

$$C_{pform2}(z)=\frac{0.12z^{-1}-0.12z^{-2}}{1-1.88z^{-1}+0.88z^{-2}}R_{ref}(z) \qquad (5.83)$$

Η (5.83) είναι στο μετασχηματισμό Z, που σε εξισώσεις διαφορών μεταφράζεται σε

$$c_{pform2}[nT_s] = 0.12r_{ref}[(n-1)T_s] - 0.12r_{ref}[(n-2)T_s] + 1.88c[(n-1)T_s] - 0.88c[(n-2)T_s] \quad (5.84)$$

Για να προκύψουν αυτές οι εξισώσεις έγινε χρήση της (5.32).



Εικόνα 49. Απόκριση της ψηφιακής positional form υλοποίησης του συστήματος για ζ = 10, ωn = 0.2

Το Εικόνα 49. δείχνει την απόκριση του (5.84). Είναι μια γρήγορη απόκριση, πολύ κοντά σε αυτό που μπορεί να επιτύχει το σκάφος, και περίπου τέσσερις φορές ταχύτερη από τις (5.80), που είναι κάτι που θέλουμε για να έχουμε τον εσωτερικό βρόχο. Να είναι δηλαδή πιο γρήγορος από τον εποπτικό. Αυτό είναι απαραίτητο για τη σταθερότητα του συστήματος.

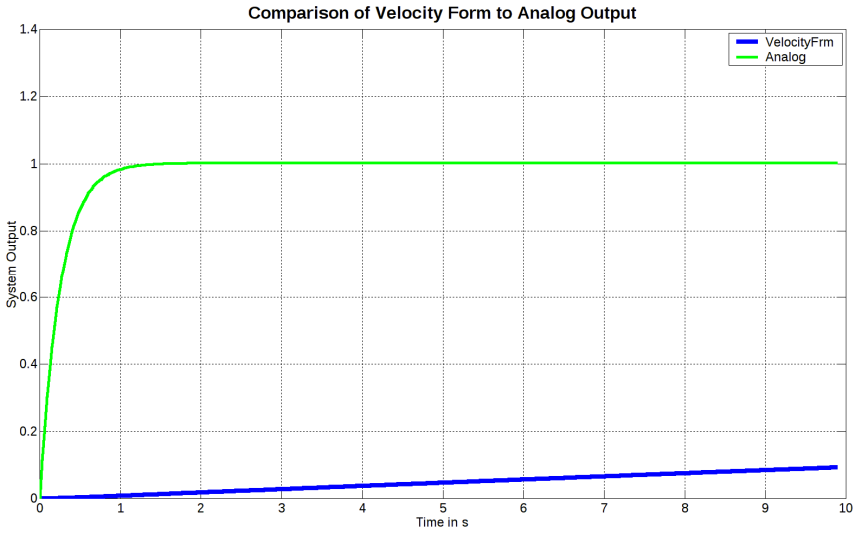
Από την άλλη πλευρά, η υλοποίηση της velocity form χρησιμοποιώντας Kpc = 4, Kic = 0.04 περιγράφεται από τις ακόλουθες εξισώσεις ((5.85) & (5.86)). Η απόκριση αυτών των εξισώσεων φαίνεται στην Εικόνα 50..

$$C_{vform2}(z) = \frac{3.6\cdot10^{-5}z^{-1}}{1-1.88z^{-1}+0.88z^{-2}}R_{ref}(z) \quad (5.85)$$

ή σε μορφή διακριτού χρόνου (στηριγμένη στην (5.33):

$$c_{vform2}[nT_s] = 3.6\cdot10^{-5}r_{ref}[(n-1)T_s] + 1.88c[(n-1)T_s] - 0.88c[(n-2)T_s] \quad (5.86)$$

Αυτή τη φορά η σταθερά πολλαπλασιασμού για την είσοδο αναφοράς είναι εξαιρετικά μικρή, με αποτέλεσμα μια απαράδεκτα αργή βηματική απόκριση. Επομένως, για αυτήν την περίπτωση η velocity form δεν είναι εφαρμόσιμή.



Εικόνα 50. Απόκριση συστήματος με PID τύπου velocity form για ζ = 10, ωn = 0.2

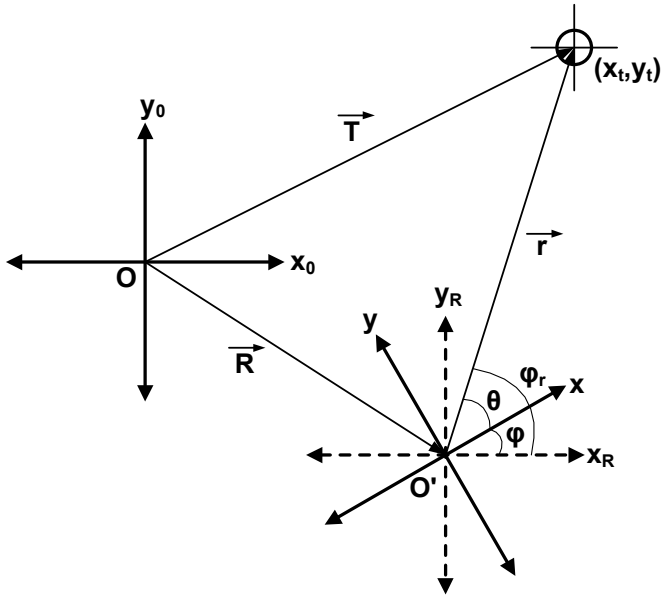
5.6 Εποπτικός Ελεγκτής (Supervisory Controller ή SC)

Ο εποπτικός ελεγκτής είναι βασικά ένα σύνολο δύο μαθηματικών εξισώσεων που αναγκάζουν το σύστημα να μηδενίσει την απόσταση και τη γωνία από τον στόχο του. Ο παρακάτω πίνακας παραθέτει τα σύμβολα που χρησιμοποιούνται σε αυτή την παράγραφο και βοηθά στην κατανόηση των ιδεών που χρησιμοποιούνται:

Symbol	Description
SC	Supervisory Controller.
v_s	Η ζητούμενη μεταφορική ταχύτητα από τον εποπτικό ελεγκτή. Το σχήμα ελέγχου που ακολουθεί προσπαθεί να αναγκάσει το σύστημα να επιτύχει αυτό το αίτημα.
ω_s	Το ζητούμενο ποσοστό στροφής από τον εποπτικό ελεγκτή. Το σχήμα ελέγχου που ακολουθεί προσπαθεί να αναγκάσει το σύστημα να επιτύχει αυτό το αίτημα.
$\vec{r}$	Το διάνυσμα που συνδέει το κέντρο μάζας του σκάφους με τον στόχο.
$\|\vec{r}\|$	Το μέτρο του $\vec{r}$ που ισοδυναμεί με την απόσταση από τον στόχο.
φ	The angle between the axis parallel to vessel's course and Ox_0y_0 or $O'x_Ry_R$ axes. Θετική κατά την ανθωρολογιακή φορά.
φ_r	Η γωνία μεταξύ $\vec{r}$ και Ox_0y_0 ή $O'x_Ry_R$ άξονες. Θετική ανθωρολογιακά.
θ	Η γωνία μεταξύ του άξονα παράλληλου προς την πορεία του σκάφους και του διανύσματος $\vec{r}$. Θετική ανθωρολογιακά.
R_s	Απόσταση μεταξύ σκάφους και στόχου από την οποία τα αιτήματα ταχύτητας του SC αλλάζουν μέθοδο υπολογισμού.

Οι έξοδοι του SC είναι η v_s και η ω_s, που είναι η μεταφορική ταχύτητα και η γωνιακή ταχύτητα αντίστοιχα, που εντέλλονται στο υπόλοιπο σύστημα.

Ως είσοδο παίρνει το διάνυσμα $\vec{r}$ (με αναφορά στο σχήμα "Σχήμα 8. Συστήματα αξόνων αναφοράς κινήσεων." ή την «Εικόνα 51. Επανάληψη εικόνας συστημάτων αξόνων.» είναι ή ίδια σε σμίκρυνση), που είναι το διάνυσμα που συνδέει το κέντρο μάζας του σκάφους με το σημείο στόχο, ειδωμένο στο σωματόδετο σύστημα αξόνων O'xy. Από το διάνυσμα αυτό, το μέτρο της απόστασης ($\|\vec{r}\|$) και η γωνία απόκλισης θ μπορούν να υπολογιστούν.



Εικόνα 51. Επανάληψη εικόνας συστημάτων αξόνων.

Εμβαθύνοντας αναφορικά με τη γωνία θ, θα πρέπει να αναφερθούμε επιπροσθέτως στο σύστημα αξόνων $O'x_Ry_R$. Αυτό το σύστημα αξόνων παραμένει σταθερό ως προς τη γωνία και παράλληλο με το Ox_0y_0, αλλά ακολουθεί το κέντρο μάζας του σκάφους. Όταν αναφερόμαστε στο ω_s, που είναι μια από τις εξόδους του supervisory controller, εννοούμε τον επιθυμητό ρυθμό μεταβολής της γωνίας φ.

Ορίζοντας το θ:

$$\boxed{\theta = \varphi_r - \varphi} \qquad (5.87)$$

Αν θεωρήσουμε το φ_r ως σταθερά (για σχετικά μεγάλες αποστάσεις αυτό είναι αληθές), τότε έχουμε

$$\frac{d\theta}{dt} = \dot{\theta} = \frac{d\varphi_r}{dt}^{0} - \frac{d\varphi}{dt} \Leftrightarrow$$

$$\boxed{\dot{\theta} = -\dot{\varphi} = -\omega_s} \tag{5.88}$$

5.6.1 Εξισώσεις που Γεννούν Setpoints

Υπολογισμοί Επιθυμητής Ταχύτητας

Για τη δημιουργία ενός v_s (επιθυμητή ταχύτητα) ο αλγόριθμος που χρησιμοποιείται είναι:

$$\begin{aligned} &\text{if } \left(\|\vec{r}\| > R_s\right) \text{ then} \\ &v_s = v_{mset} \\ &\text{else} \\ &v_s = v_{mset} \cdot \left(\frac{\|\vec{r}\|}{R_s}\right)^2 \end{aligned} \tag{5.89}$$

όπου R_s είναι η απόσταση μεταξύ σκάφους και στόχου, στην οποία βασίζεται η μέθοδος επιλογής επιθυμητής ταχύτητας.

Για να αποδείξουμε ότι το σύνολο των εξισώσεων (5.89) οδηγεί σε μηδενική απόσταση, διερευνούμε την περίπτωση όπου η γωνία θ είναι μηδέν. Δεν πρόκειται για αυθαίρετη απόφαση, καθώς όπως θα αποδειχθεί στην ενότητα «Υπολογισμοί γωνιακής ταχύτητας», ότι η γωνία θ τελικά και ευσταθώς θα καταλήξει στο μηδέν.

Πρώτα πρέπει να παρατηρήσουμε ότι όταν θ = 0, τότε

$$v_s = -\frac{d\|\vec{r}\|}{dt} \tag{5.90}$$

Εξετάζοντας την περίπτωση που $\|\vec{r}\| < R_s$ έχουμε (χρησιμοποιώντας την (5.89) και (5.90)) τότε

$$-\frac{d\|\vec{r}\|}{dt} = v_{mset} \cdot \left(\frac{\|\vec{r}\|}{R_s}\right)^2 \Leftrightarrow$$

$$\frac{d\|\vec{r}\|}{dt} = -\frac{v_{mset}}{R_s^{\,2}}\|\vec{r}\|^2 \Leftrightarrow$$

$$\frac{d\|\vec{r}\|}{\|\vec{r}\|^2} = -\frac{v_{mset}}{R_s^{\,2}}dt \Rightarrow$$

$$\int\frac{d\|\vec{r}\|}{\|\vec{r}\|^2} = -\int\frac{v_{mset}}{R_s^{\,2}}dt \Rightarrow$$

$$-\frac{1}{\|\vec{r}\|} + c_1 = -\frac{v_{mset}}{R_s^{\,2}}t \tag{5.91}$$

Για να διευκολυνθεί η γραφή εκχωρούμε το σύμβολο r(t) ως υποκατάστατο του για την περίπτωση που ερευνούμε (καμία αλλαγή γωνίας θ και θ = 0). Τότε η εξίσωση (5.91) γίνεται:

$$-\frac{1}{r(t)}+c_1=-\frac{v_{mset}}{R_s^{\,2}}t$$

Για να βρούμε τη σταθερά c_1 θέτουμε t = 0 και έχουμε:

$$c_1=\frac{1}{r(0)}\equiv\frac{1}{r_0}$$

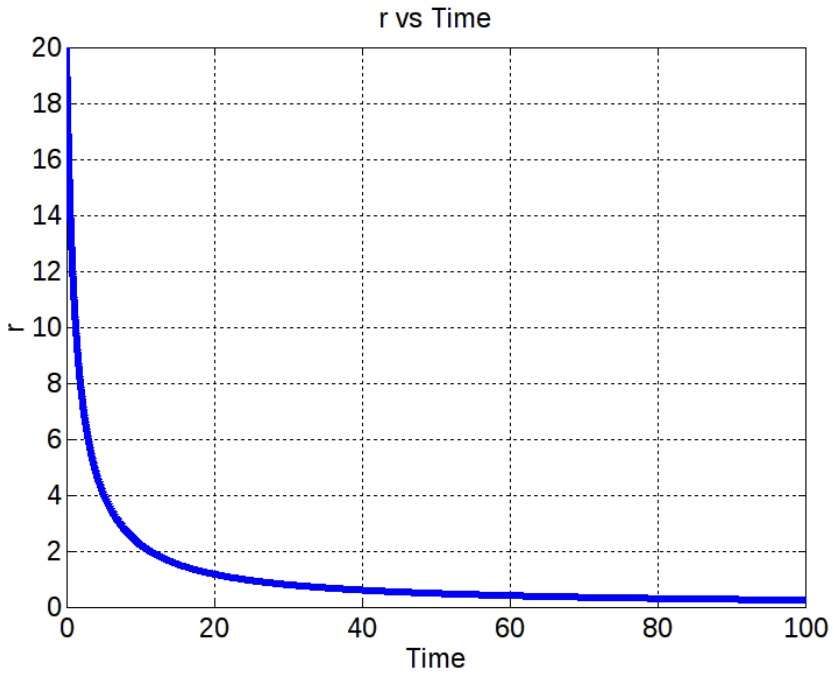
Επομένως η (5.91) γίνεται

$$-\frac{1}{r(t)}=-\frac{1}{r_0}-\frac{v_{mset}}{R_s^{\,2}}t \Leftrightarrow$$

$$r(t)=\frac{1}{\frac{1}{r_0}+\frac{v_{mset}}{R_s^{\,2}}t} \Leftrightarrow$$

$$\boxed{r(t)=\frac{r_0}{1+\frac{v_{mset}}{R_s^{\,2}}r_0 t}} \qquad (5.92)$$

Ένα παράδειγμα απόκρισης της συνάρτησης (5.92) με R_s = 20m, r_0 = 20m αποτυπώνεται στο παρακάτω διάγραμμα (Εικόνα 52. Απόσταση συναρτήσει χρόνου όπως διαμορφώνεται από τα αιτήματα ταχύτητας του SC.).



Εικόνα 52. Απόσταση συναρτήσει χρόνου όπως διαμορφώνεται από τα αιτήματα ταχύτητας του SC.

Δύο σημειώσεις για το διάγραμμα της Εικόνα 52:

- Αυτή δεν είναι η πραγματική απόκριση του σκάφους. Είναι η απάντηση που θα προέκυπτε εάν το σκάφος μπορούσε να ακολουθήσει αμέσως τις εντολές από το SC.
- Στην αρχή η απόκριση μπορεί να φαίνεται πολύ αργή, αλλά πρέπει να σημειωθεί ότι το «σκάφος φτάνει στο στόχο» δεν σημαίνει απαραίτητα ότι η απόσταση είναι μηδέν. Είναι μια αυθαίρετη επιλογή. Έτσι, εάν για παράδειγμα ορίσουμε ότι «σκάφος φτάνει στο στόχο» σημαίνει σκάφος σε απόσταση 2 μέτρων από τον στόχο, τότε ο πραγματικός χρόνος που απαιτείται για αυτό μειώνεται δραματικά.

Για να κλείσουμε την ενότητα, το όριο t → +∞ θα πρέπει να εφαρμοστεί στην (5.92), προκειμένου να αποδειχθεί ότι τελικά ο απόσταση θα είναι μηδέν.

$$\lim_{t\to\infty}\left[r(t)\right]=\lim_{t\to\infty}\left(\frac{r_0}{1+\frac{v_{mset}}{R_s^{\,2}}r_0 t}\right)=0$$

Αυτό αποδεικνύει ότι το σκάφος θα έχει τελικά μηδενική απόσταση από τον στόχο.

<u>Υπολογισμοί Γωνιακής Ταχύτητας</u>

Για τη δημιουργία δείγματος επιθυμητής γωνιακής ταχύτητας ω_s, πρώτα μια συνάρτηση $f_s(\theta)$ ορίζεται, όπου

$$f_s(\theta)\equiv \text{sign}(\theta)\cdot(1-\cos\theta) \tag{5.93}$$

$\theta\in(-\pi,\pi]$ που υπονοεί ότι από τη στιγμή που $-1\le\cos\theta\le 1$, θα έχουμε για την $f_s(\theta)$ ότι $-2\le f_s(\theta)\le 2$.

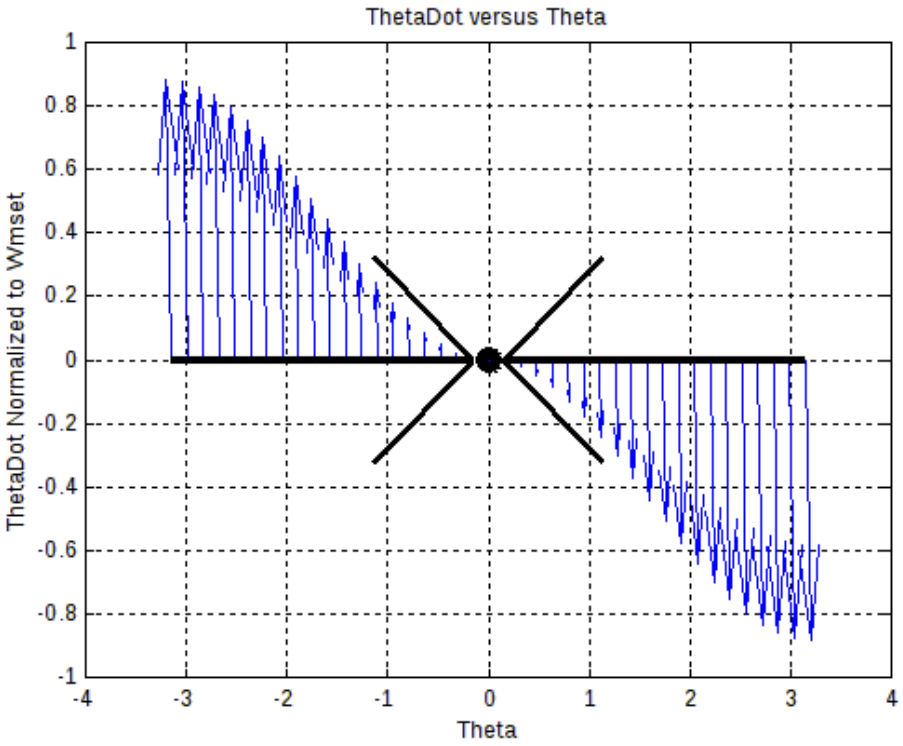
Επιστρέφοντας στη συνάρτηση παραγωγής ω_s

$$\omega_s=\frac{\omega_{mset}}{2}\cdot f_s(\theta) \tag{5.94}$$

Παρατηρώντας ότι $\omega_s=-\dot{\theta}$ και αντικαθιστώντας την $f_s(\theta)$ με την (5.93),

$$\boxed{\dot{\theta}=-\frac{\omega_{mset}}{2}\cdot\text{sign}(\theta)\cdot(1-\cos\theta)} \tag{5.95}$$

Αυτό είναι ένα μονοδιάστατο, πρώτου βαθμού, μη γραμμικό σύστημα, για το οποίο εάν σχεδιάσουμε το λεγόμενο «flow on the line», θα έχουμε το διάγραμμα που φαίνεται στην εικόνα:

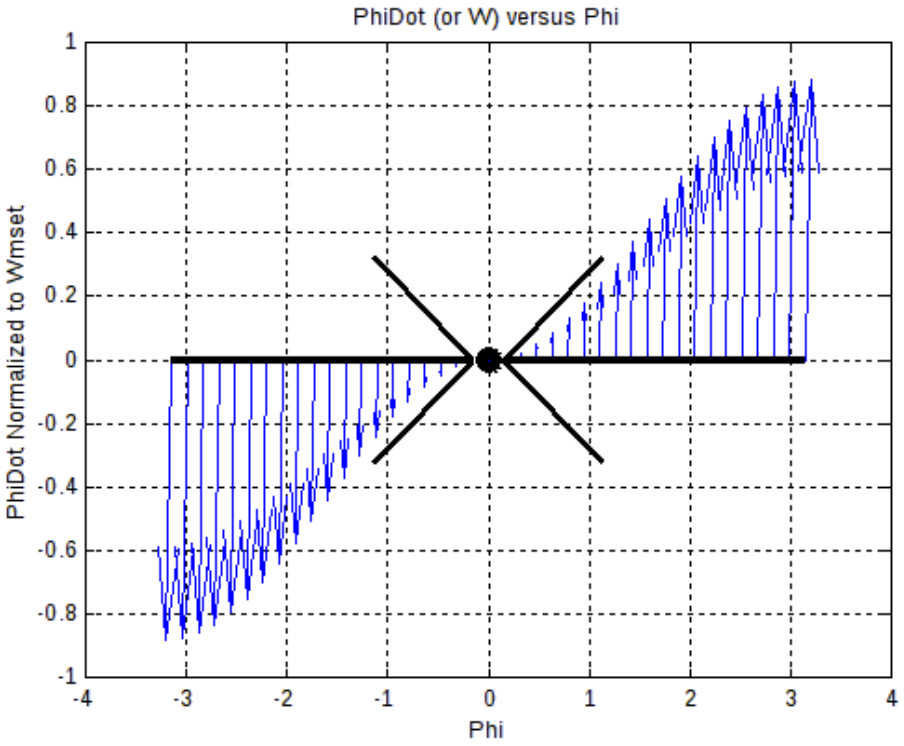


Εικόνα 53. Ροή αποτελεσμάτων για εξίσωση (5.95).

Στη Εικόνα 53. Ροή αποτελεσμάτων για εξίσωση ο Υ άξονας είναι κανονικοποιημένος ως προς ω_{mset} που είναι η μέγιστη γωνιακή ταχύτητα που επιτρέπεται (προγραμματιζόμενη σταθερά) να επιδιώξει ο εποπτικός ελεγκτής (SC). Η γωνία μεταβάλλεται από –π έως π.

Η Εικόνα 53. Ροή αποτελεσμάτων για εξίσωση αποτυπώνει τον τρόπο με τον οποί ο SC αποκρίνεται σε διάφορες τιμές του θ. Όταν θ < 0 παράγει εξόδους οι οποίες έχουν ως αποτέλεσμα θετική παράγωγο του θ ($\dot{\theta}$), με αποτέλεσμα να εξαναγκάζει το θ να γίνει μηδέν. Η απόδειξη ότι καταλήγει στο μηδέν θα ακολουθήσει. Όταν η γωνία θ είναι θετική, ο SC απαντά με ενέργειες που οδηγούν σε αρνητικό $\dot{\theta}$ που και πάλι εξαναγκάζει το σύστημα σε μηδενικό θ.

Είναι ενδιαφέρον να εξεταστεί η ανάλογη ροή της γωνίας φ η οποία είναι στενά συνδεδεμένη με τη γωνία θ. Η Εικόνα 54 εικονίζει το αποτέλεσμα.



Εικόνα 54. Ελκυστής για συνάρτηση γωνίας που προκύπτει από τη λειτουργία του SC.

Η Εικόνα 54 είναι και πάλι κανονικοποιημένη ως προς τον Y άξονα ως προς ω_{mset}. Η γωνία μεταβάλλεται από –π έως π.

Αυτό που υπονοεί η Εικόνα 54 είναι ότι όταν η γωνία μεταξύ της μεταφορικής ταχύτητας του σκάφους και του διανύσματος που συνδέει το κέντρο μάζας του σκάφους με τον στόχο είναι αρνητική (που σημαίνει ότι ο στόχος είναι στα δεξιά), τότε η γωνιακή ταχύτητα γίνεται αρνητική, αναγκάζοντας έτσι το σκάφος να στρίψει δεξιά, με αποτέλεσμα φθίνουσα φ και άρα φθίνουσα θ. Το αντίστοιχο συμβαίνει όταν η γωνία είναι θετική. Τότε και η γωνιακή ταχύτητα γίνεται θετική, μειώνοντας έτσι το θ για άλλη μια φορά.

Οι παραπάνω διαδικασίες δείχνουν ότι το σημείο μηδενικής γωνίας στον άξονα x της Εικόνα 53 και της Εικόνα 54 λειτουργεί ως ελκυστής ή σταθερό σημείο ισορροπίας, όπου ο όρος σημείο ισορροπίας υποδηλώνει ότι η παράγωγος της υπό εξέταση μεταβλητής είναι μηδέν.

Για να εμβαθύνουμε στο τί συμβαίνει σε γωνίες κοντά στο 0, χρησιμοποιείται η ανάπτυξη Taylor του cosθ γύρω από τη γωνία μηδέν (ή σειρά Maclaurin). Ο τύπος για ανάπτυγμα γύρω από το μηδέν είναι:

$$f(x)=f(0)+f'(0)x+\frac{f''(0)}{2}x^2+\ldots+\frac{f^{(n)}(0)}{n!}x^2+R_{(n+1)}$$

Εφαρμόζοντας για το cosθ έχουμε:

$$\cos(\theta)=\cos(0)+\frac{\mathrm{d}\cos\theta}{\mathrm{d}\theta}\Big|_{\theta=0}\cdot\theta+\frac{1}{2}\cdot\frac{\mathrm{d}^2\cos\theta}{\mathrm{d}\theta^2}\Big|_{\theta=0}\cdot\theta^2+\ldots\Leftrightarrow$$

$$\cos(\theta)=1-\sin(0)\cdot\theta+\frac{1}{2}\cdot\left(-\cos(0)\right)\cdot\theta^2+\ldots\Rightarrow$$

$$\cos(\theta)\approx 1-\frac{1}{2}\cdot\theta^2 \qquad (5.96)$$

Η (5.96) προκύπτει από την απαλοιφή των υψηλότερης τάξης όρων (στην οποία περιλαμβάνεται και το υπόλοιπο $R_{(n+1)}$), αφού η γωνία θ είναι κοντά στο μηδέν.

Εφαρμόζοντας την προσέγγιση (5.96) στην (5.95), έχουμε ότι για μικρές γωνίες:

$$\dot{\theta}=-\frac{\omega_{\mathrm{mset}}}{2}\cdot\mathrm{sign}(\theta)\cdot\left(1-1+\frac{1}{2}\cdot\theta^2\right)\Leftrightarrow$$

$$\boxed{\dot{\theta}=-\frac{\omega_{\mathrm{mset}}}{4}\cdot\mathrm{sign}(\theta)\cdot\theta^2} \qquad (5.97)$$

Στο σημείο αυτό γίνεται η υπόθεση ότι $\mathrm{sign}(\theta)=1$. Αυτό υπονοεί προσέγγιση από θετικές γωνίες. Τότε η (5.97) γίνεται

$$\dot{\theta}=-\frac{\omega_{\mathrm{mset}}}{4}\cdot\theta^2 \qquad (5.98)$$

Ξεκινώντας από την (5.98) έχουμε για τη θ(t):

$$\frac{1}{\left[\theta(\mathrm{t})\right]^2}\frac{\mathrm{d}\theta(\mathrm{t})}{\mathrm{dt}}=-\frac{\omega_{\mathrm{mset}}}{4}\Leftrightarrow$$

$$\frac{1}{\left[\theta(\mathrm{t})\right]^2}\mathrm{d}\theta(\mathrm{t})=-\frac{\omega_{\mathrm{mset}}}{4}\mathrm{dt}\Rightarrow$$

$$\int\frac{1}{\left[\theta(\mathrm{t})\right]^2}\mathrm{d}\theta(\mathrm{t})=-\int\frac{\omega_{\mathrm{mset}}}{4}\mathrm{dt}\Rightarrow$$

$$-\frac{1}{\theta(\mathrm{t})}+\mathrm{c}_0=-\frac{\omega_{\mathrm{mset}}}{4}\mathrm{t} \qquad (5.99)$$

Για να βρούμε την c_0 θέτουμε t = 0 και από (5.99) έχουμε:

$$-\frac{1}{\theta(0)}+\mathrm{c}_0=-\frac{\omega_{\mathrm{mset}}}{4}0\Leftrightarrow$$

$$\mathrm{c}_0=\frac{1}{\theta(0)}\equiv\frac{1}{\theta_0} \qquad (5.100)$$

Από (5.99) και (5.100) έχουμε ότι

$$-\frac{1}{\theta(\mathrm{t})}+\frac{1}{\theta_0}=-\frac{\omega_{\mathrm{mset}}}{4}\mathrm{t}\Leftrightarrow$$

$$\frac{1}{\theta(t)} = \frac{1}{\theta_0} + \frac{\omega_{mset}}{4} t \Leftrightarrow$$

$$\boxed{\theta(t) = \frac{1}{\frac{1}{\theta_0} + \frac{\omega_{mset}}{4} t}} \quad (5.101)$$

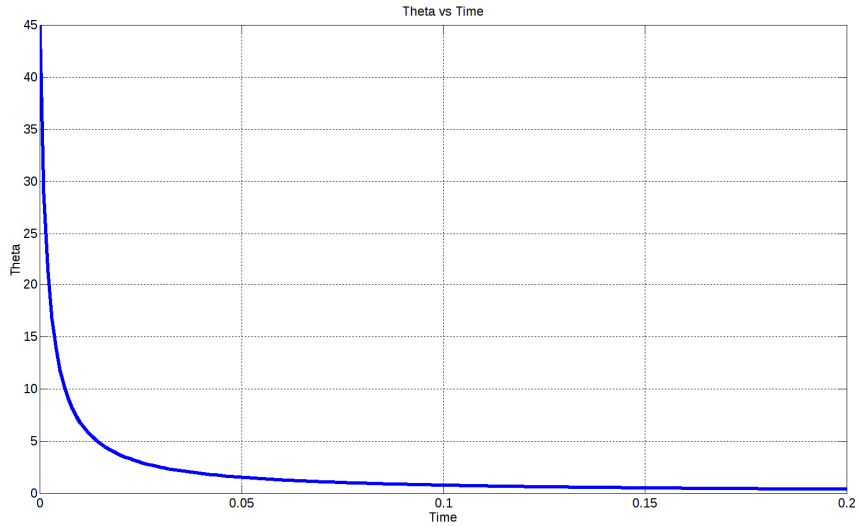
Η (από τη στιγμή που η θ_0 δεν είναι 0):

$$\boxed{\theta(t) = \frac{\theta_0}{1 + \frac{\omega_{mset}}{4} \theta_0 t}} \quad (5.102)$$

Παίρνοντας το όριο της (5.102) καθώς t → ∞ έχουμε ότι:

$$\lim_{t\to\infty}\left[\theta(t)\right] = \lim_{t\to\infty}\left(\frac{\theta_0}{1 + \frac{\omega_{mset}}{4} \theta_0 t}\right) = 0 \quad (5.103)$$

Το αποτέλεσμα (5.103) υπονοεί ότι καθώς ο χρόνος περνάει, η γωνία προσεγγίζει το μηδέν. Η γραφική αναπαράσταση της εξέλιξης της (5.103) στο χρόνο φαίνεται στην Εικόνα 55. θ(t) συναρτήσει χρόνου..



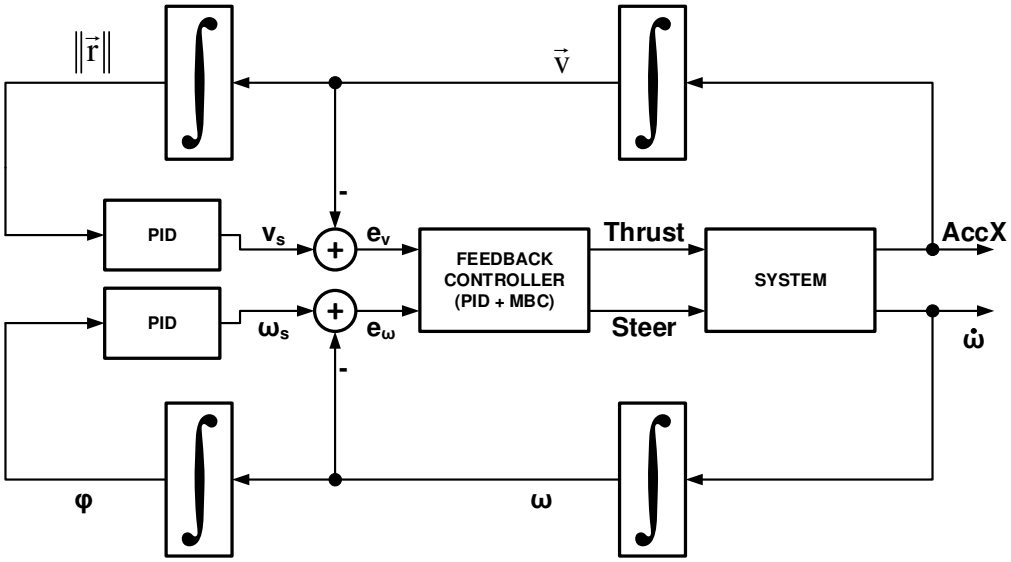
Εικόνα 55. θ(t) συναρτήσει χρόνου.

Πρέπει να ξεκαθαριστεί ότι η Εικόνα 55 δεν αναπαριστά την απόκριση του σκάφους, αλλά τη θεωρητική περίπτωση κατά την οποία το σκάφος θα ακολουθούσε αμέσως τις επιταγές του εποπτικού ελεγκτή SC.

5.7 PID ως Εποπτικός Ελεγκτής (SC)

Σε αυτή την ενότητα εξετάζεται η χρήση ενός ελεγκτή PID στη θέση του μη γραμμικού SC της ενότητας 5.6.

Λειτουργώντας ως SC, ο ελεγκτής PID έχει ως στόχο το μηδενισμό της τιμής του σφάλματος μεταξύ εισόδου και σήματος ανατροφοδότησης. Αν χρησιμοποιήσουμε ως ανατροφοδοτήσεις τα σήματα απόσταση από και γωνία ως προς τον στόχο, τότε και για τα δύο η επιθυμητή τελική τιμή τους είναι το 0. Ως εκ τούτου, εφόσον η αναφορά είναι μηδέν, το σήμα ανάδρασης είναι και το σφάλμα.



Εικόνα 56. Το πλήρες σύστημα με PID ως εποπτικούς ελεγκτές.

Επειδή θεωρούμε ότι ο ελεγκτής MBC διαχωρίζει τους βρόχους ελέγχου απόστασης και γωνίας, χρησιμοποιούμε δύο διαφορετικούς PID. Έναν για το μηδενισμό της απόστασης και έναν για το μηδενισμό της γωνίας.

Η Εικόνα 56 δείχνει το σχήμα ελέγχου με τους PID ως εποπτικούς ελεγκτές. Ένα σημείο που πρέπει να διευκρινιστεί, είναι ότι το $\|\vec{r}\|$ είναι το μέτρο του διανύσματος $\vec{r}$ με αναφορά στην Εικόνα 51. Επανάληψη εικόνας συστημάτων αξόνων.. Είναι το διάνυσμα που συνδέει το κέντρο μάζας του σκάφους με τον στόχο. Παρόλο που η Εικόνα 56 δείχνει διανύσματα ως μεγέθη για το βρόχο ελέγχου απόστασης από τον στόχο (επάνω), ο πραγματικός βρόχος PID βασίζεται στη βαθμωτή τιμή $\|\vec{r}\|$. Δεδομένου ότι η μεταφορική επιτάχυνση θεωρείται μόνο κατά μήκος του άξονα x του O'xy, αυτό δημιουργεί ένα λογικό ζήτημα, επειδή έχουμε ένα μείγμα διανυσμάτων και βαθμωτών μεγεθών. Αυτό επιλύεται από τον βρόχο γωνίας, ο οποίος ευθυγραμμίζει τον άξονα O'x στο $\vec{r}$, αναγκάζοντας τη λειτουργία του βρόχου απόστασης σε έναν μόνο άξονα

Η ενότητα 5.8.2 περιγράφει τα αποτελέσματα της λειτουργίας του PID SC στο μοντέλο του σκάφους. Τα επιλεγμένα κέρδη είναι $K_{pc} = K_{ic} = 4$ και $K_{dc} = 0$ και η υλοποίηση των PID ακολουθεί την positional form.

5.8 Απόκριση Σκάφους

5.8.1 Απόκριση με Χρήση του μη-γραμμικού Εποπτικού Ελεγκτή

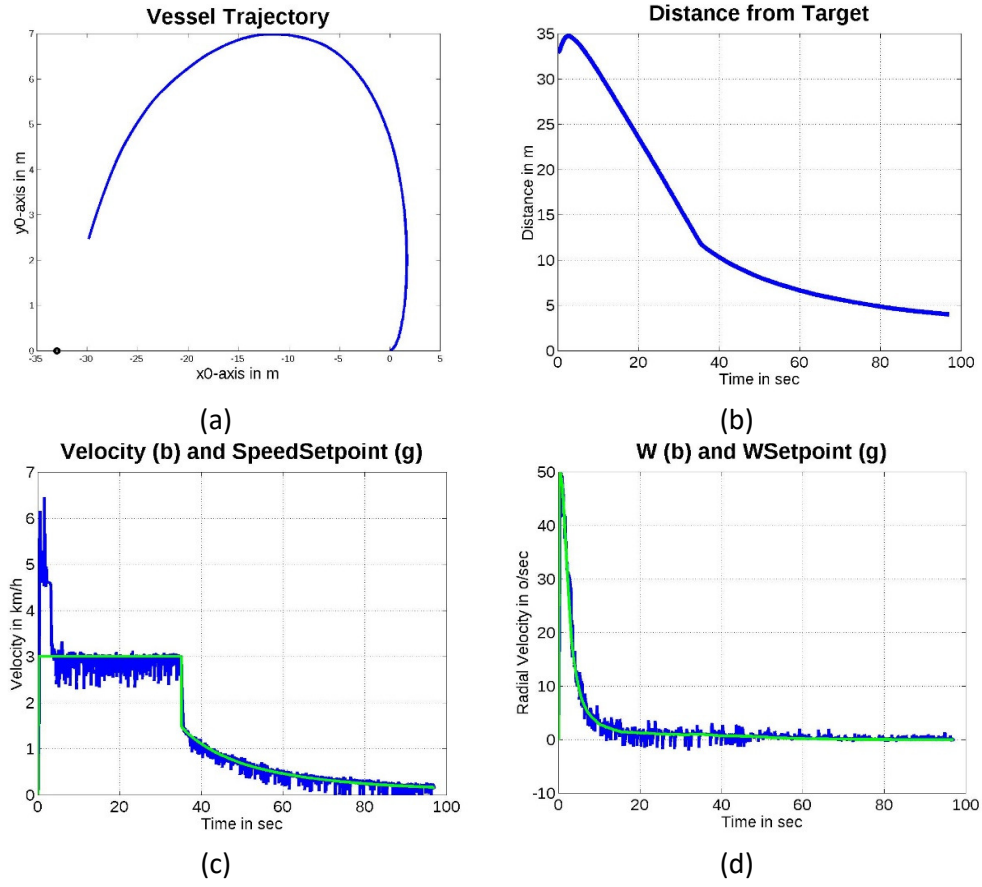
Στην ενότητα αυτή παρουσιάζονται τα αποτελέσματα της χρήσης του εποπτικού ελεγκτή που περιγράφεται στην ενότητα 5.6.

Δοκιμή Κίνησης Δυτικά

Η πρώτη δοκιμή της ενότητας είναι μια δοκιμή κίνησης δυτικά, ξεκινώντας με προσανατολισμό ανατολικά. Η θέση του στόχου είναι το σημείο (-33m, 0). Λόγω της κατασκευής του SC, το σκάφος θα στρίψει αριστερά για γωνίες 0 < θ < 180ο. Η δοκιμή σταματά όταν το σκάφος απέχει 4 μέτρα από τον στόχο.

Η Εικόνα 57(a) δείχνει την τροχιά που ακολουθεί το σκάφος μέχρι να φτάσει σε απόσταση 4 μέτρων από τον στόχο. Η Εικόνα 57(b) δείχνει πώς αυτή η απόσταση αλλάζει σε σχέση με το χρόνο, ενώ το σκάφος ακολουθεί τη διαδρομή της Εικόνα 57(a).

Οι εικόνες Εικόνα 57(c) & (d) δείχνουν τις επιθυμητές ταχύτητες που δημιουργήθηκαν από τον SC και πώς αυτές ακολουθήθηκαν από το σκάφος ως αποτέλεσμα των ενεργειών του MBC. Οι πράσινες γραμμές είναι οι επιθυμητές ταχύτητες και οι μπλε οι πραγματικές τιμές.

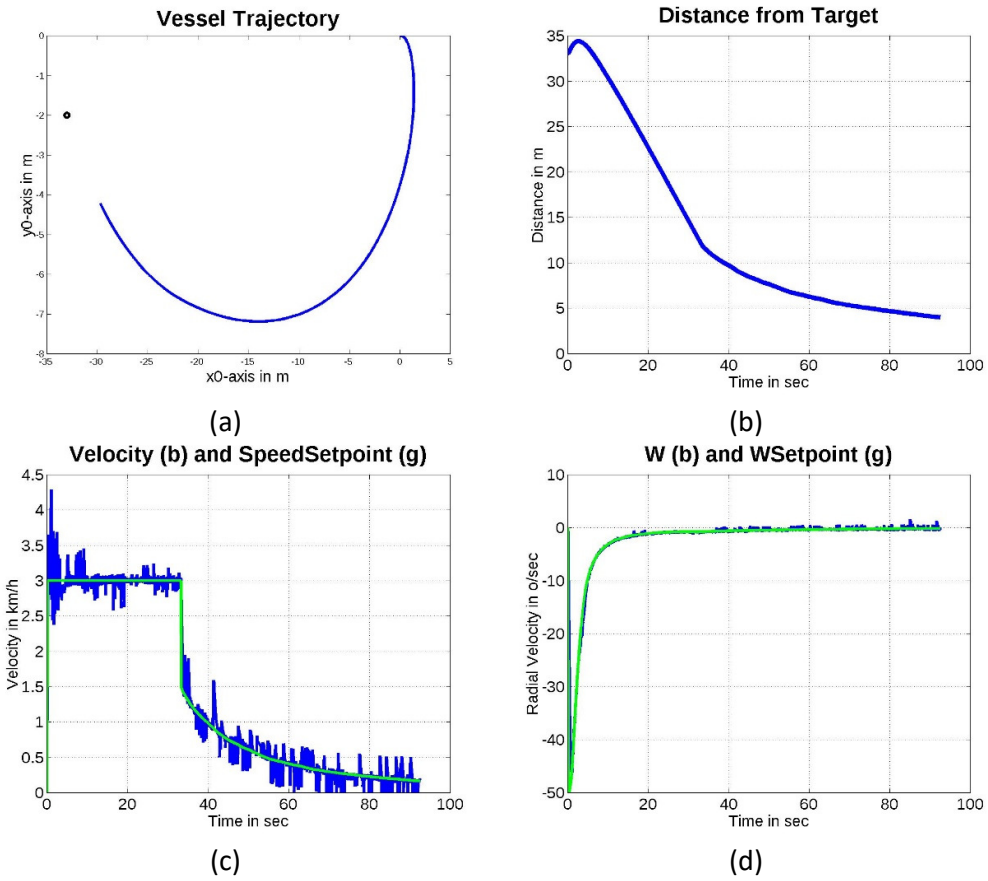


Εικόνα 57. Δοκιμή κίνησης δυτικά με τον SC της §5.6.

Δοκιμή Κίνησης Δυτικά με Δεξιά Στροφή

Το δεύτερο τεστ της ενότητας μοιάζει με το πρώτο, αλλά αυτή τη φορά ο στόχος είναι στο (-33, -2), για να αναγκάσει το SC να κάνει δεξιά στροφή στην αρχή της διαδρομής.

Τα αποτελέσματα, όσον αφορά στην τροχιά, δείχνουν ότι το σκάφος ξεκινά με μια δεξιά στροφή. Η Εικόνα 58(a) το αποδεικνύει. Τα υπόλοιπα γραφήματα ακολουθούν την περιγραφή της προηγούμενης ενότητας.



Εικόνα 58. Δοκιμή κίνησης δυτικά με δεξιά στροφή, με χρήση του SC της §5.7.

Δοκιμή Πολλαπλών Στόχων

Η τελική δοκιμή της ενότητας περιλαμβάνει ένα σενάριο που μπορεί να επεκταθεί, προκειμένου να οδηγήσουμε το σκάφος σε μια προκαθορισμένη τροχειά. Το σκάφος ακολουθεί μια λίστα σημείων στόχων. Η λίστα φαίνεται στον Πίνακας 2:

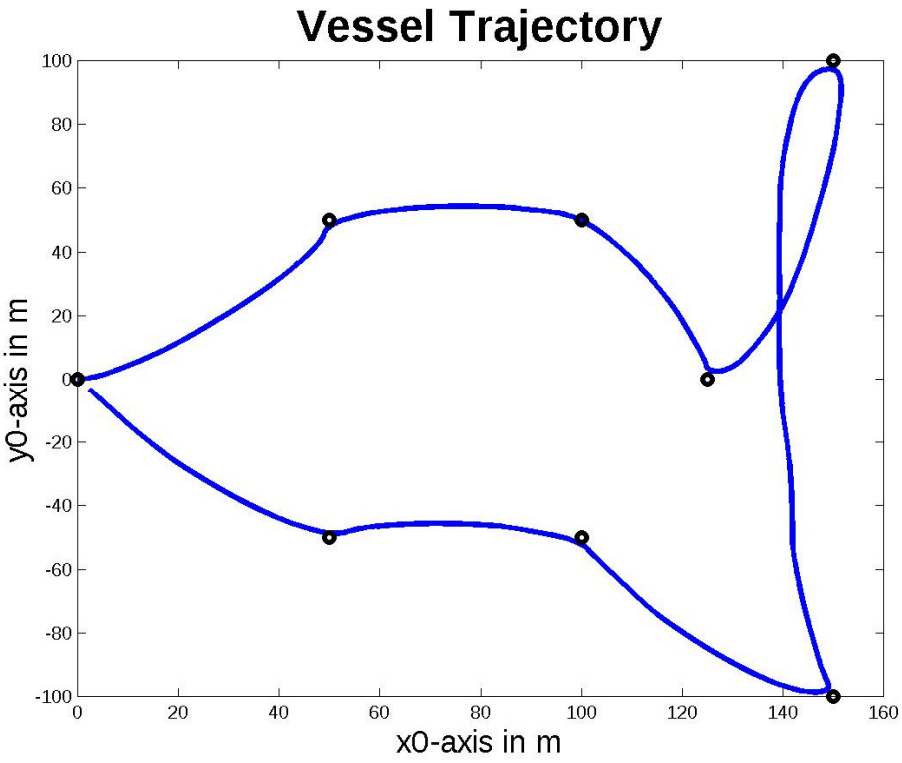
No	Target x	Target y
1	50	50
2	100	50
3	125	0
4	150	100
5	150	-100
6	100	-50
7	50	-50
8	0	0

Πίνακας 2. Λίστα θέσεων στόχων.

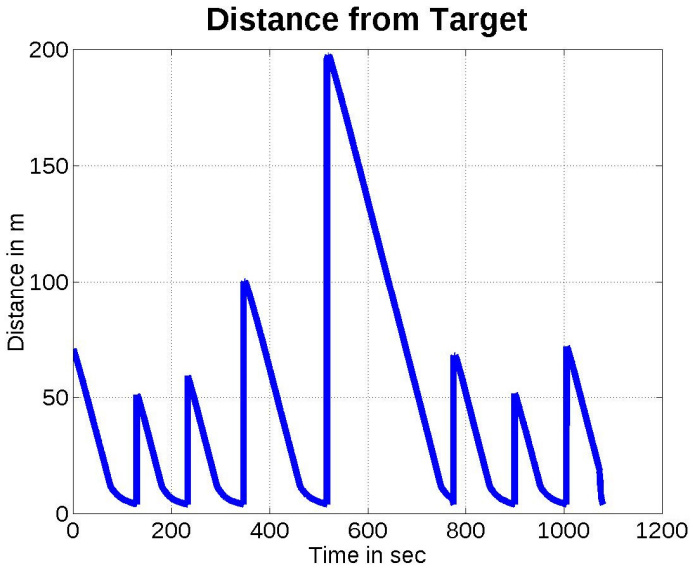
Η Εικόνα 59 δείχνει την τροχιά που ακολουθήθηκε κατά τη διάρκεια της δοκιμής. Η λογική επιλογής του στόχου ήταν να επιλέγει ο επόμενος κάθε φορά που το σκάφος είχε απόσταση 4 μέτρων από τον τρέχοντα στόχο. Τα σημεία στόχου εμφανίζονται με τη χρήση μιας μαύρης κουκκίδας.

Όπως φαίνεται ξεκάθαρα, το σκάφος περνά από κάθε στόχο με επιτυχία. Το αποτέλεσμα επαληθεύεται επίσης από την Εικόνα 60(a), η οποία δείχνει ότι η απόσταση από τον στόχο μειώνεται σε σχέση με το χρόνο. Κάθε φορά που εμφανίζεται μια νέα κορυφή στην καμπύλη απόστασης, είναι η επιλογή ενός νέου σημείου στόχου από τη λίστα.

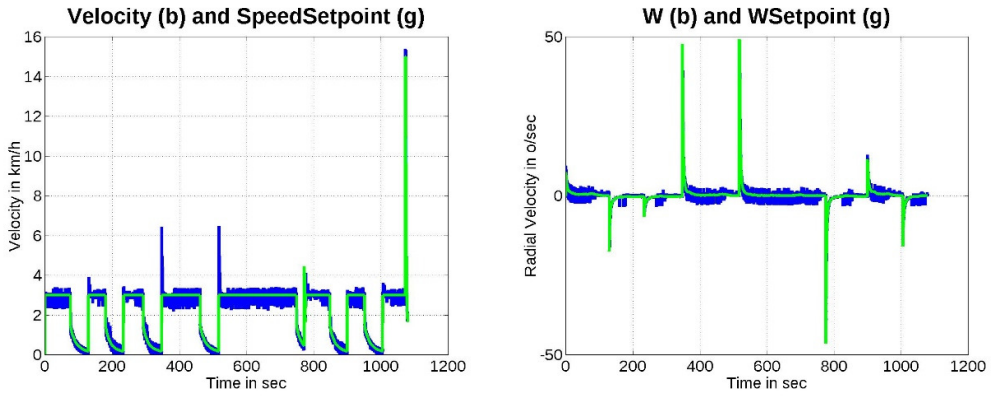
Τα σχήματα των Εικόνα 60(β) και (γ) εμφανίζουν τις καμπύλες ταχύτητας και γωνιακής ταχύτητας, αντίστοιχα, με τον χρόνου (μπλε γραμμές). Οι πράσινες γραμμές αποτυπώνουν τις εντολές του SC.



Εικόνα 59. Πορεία που ακολούθησε το σκάφος στη δοκιμή πολλαπλών στόχων.



(a)



(b) (c)

Εικόνα 60. Αποτελέσματα προσέγγισης πολλαπλών στόχων. (a) Απόσταση συναρτήσει χρόνου (b) Εντολές ταχύτητας (πράσινη γραμμή) και πραγματική ταχύτητα (μπλε γραμμή) (c) Εντολές γωνιακής ταχύτητας (πράσινη γραμμή) και πραγματική γωνιακή ταχύτητα (μπλε γραμμή).

5.8.2 Αποτελέσματα με Χρήση PID ως Εποπτικού Ελεγκτή (SC)

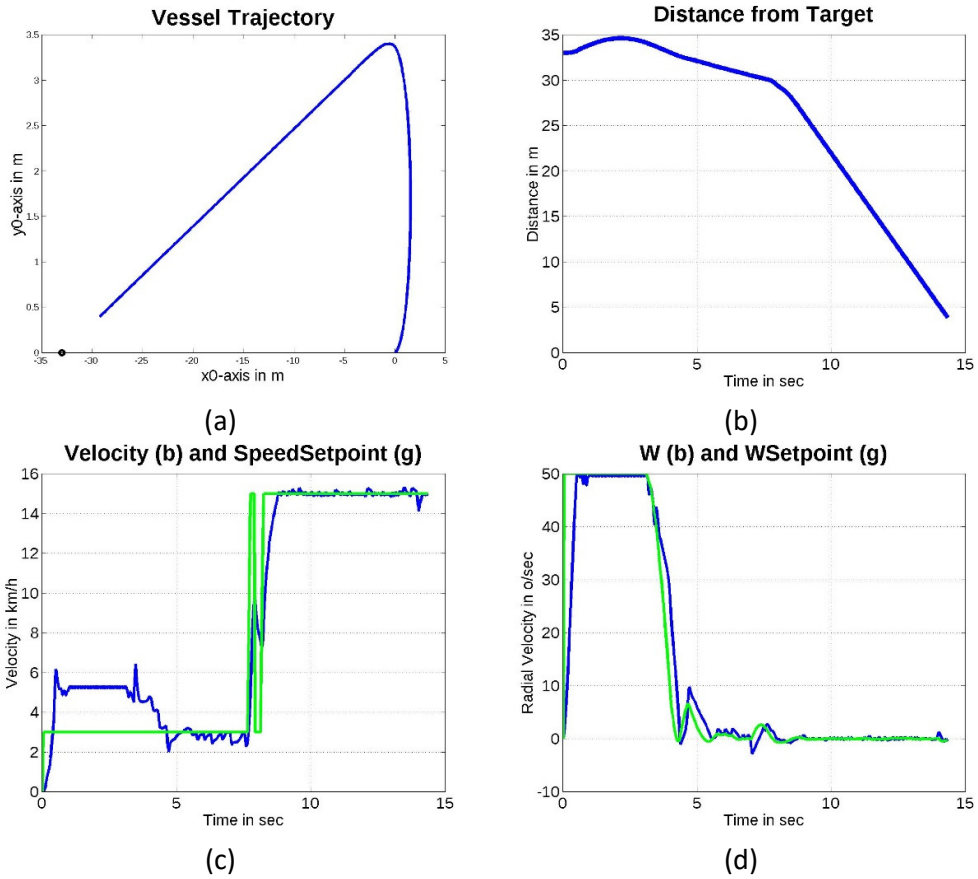
Σε αυτή την ενότητα, παρουσιάζονται τα αποτελέσματα από τη χρήση ενός ελεγκτή PID ως SC. Αντί να χρησιμοποιήσουμε το μη γραμμικό SC της ενότητας 5.7, το αντικαταστήσαμε με έναν PID που έχει 0 ως εισόδους αναφοράς (δείτε ενότητα 5.7 «PID ως Εποπτικός Ελεγκτής (SC)»).

<u>Δοκιμή Κίνησης Δυτικά</u>

Σε αυτή τη δοκιμή έχουμε ένα σημείο στόχο στο (-33, 0) και παρακολουθούμε την απόκριση του σκάφους καθώς προσπαθεί να βρει στόχο στις 180ο. Λόγω του σχεδιασμού του ελεγκτή SC το σκάφος αναμένεται να κάνει δεξιά στροφή για να κατευθυνθεί προς τον στόχο.

Η Εικόνα 61 δείχνει τα αποτελέσματα αυτής της δοκιμής. Η τροχιά του σκάφους φαίνεται στην Εικόνα 61(a). Σε σύγκριση με τον προηγούμενο σχεδιασμό SC, για τις τρέχουσες ρυθμίσεις και των δύο SC, η απόκριση του PID SC είναι πιο «απόλυτη». Δημιουργεί μεγάλες επιταχύνσεις φτάνοντας στον στόχο πολύ πιο γρήγορα.

Το εάν είναι προτιμητέα λύση ή όχι εξαρτάται από την εφαρμογή. Εάν επρόκειτο να χρησιμοποιηθεί για ανθρώπινη μεταφορά θα ήταν μάλλον προβληματική και θα έπρεπε τουλάχιστον να γίνουν κατάλληλες προσαρμογές στα κέρδη PID. Από την άλλη, αν η εφαρμογή ήταν UAV, τότε οι απότομες κινήσεις δεν είναι και τόσο πρόβλημα, ενδεχομένως να αποτελούν και πλεονέκτημα.



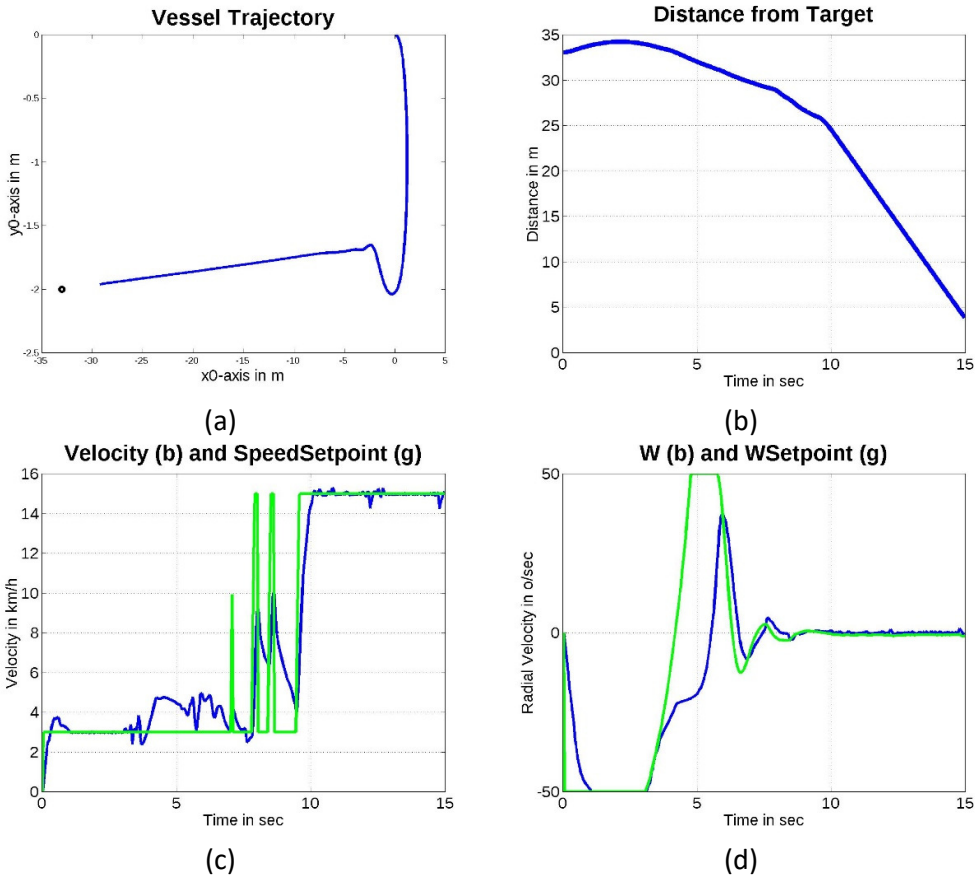
Εικόνα 61. Αποτέλεσμα δοκιμής κίνησης δυτικά με PID SC.

Δοκιμή Κίνησης Δυτικά με Δεξιά Στροφή

Αυτή η δοκιμή είναι σχεδόν η ίδια με την προηγούμενη παράγραφο, αλλά αυτή τη φορά ο στόχος είναι στο (-33, -2), αναγκάζοντας το σκάφος να κάνει δεξιά στροφή.

Αυτή τη φορά εμφανίζεται μια ιδιομορφία, η οποία είναι ένδειξη ότι το μοντέλο RC δεν έχει ακριβώς την ίδια συμπεριφορά στην αριστερή στροφή με τη δεξιά. Ωστόσο το σκάφος είναι και πάλι επιτυχές στην επίτευξη του στόχου του.

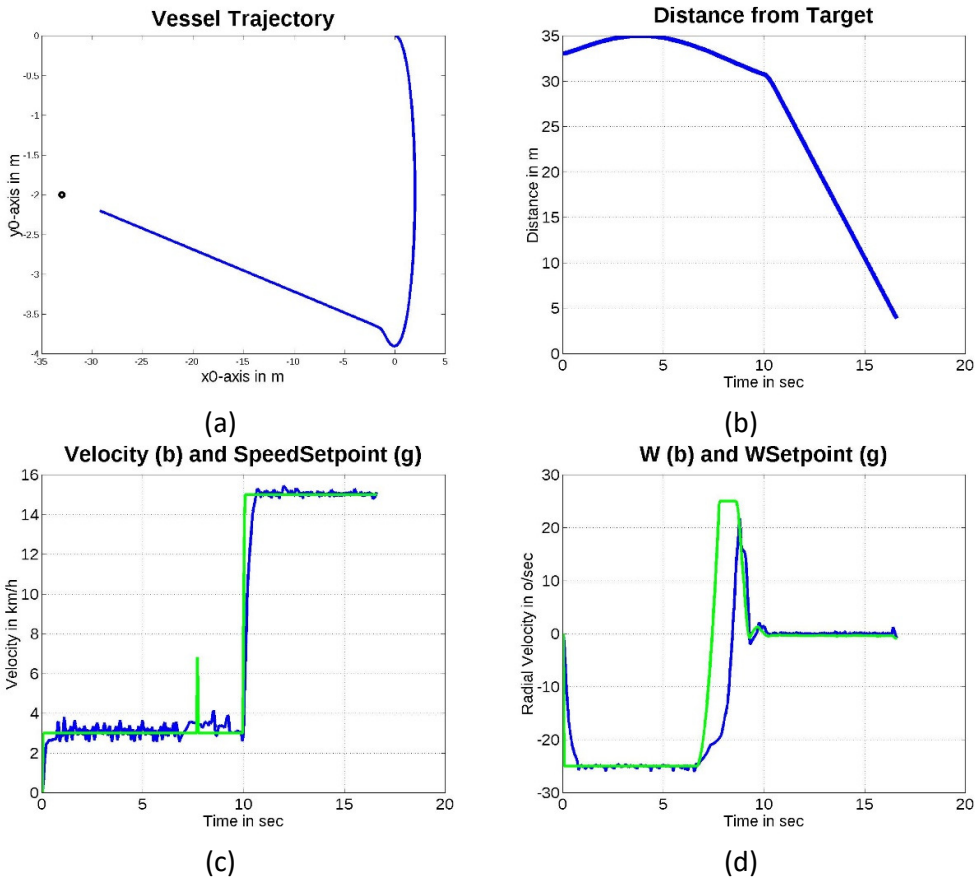
Η Εικόνα 62 δείχνει τα αποτελέσματα της δοκιμής. όπου φαίνεται το overshoot στη δεξιά στροφή. Ένας εύκολος τρόπος για να αντισταθμιστεί αυτή η υπέρβαση (είναι επίσης δυνατή η εκτέλεση βελτιστοποίησης των παραμέτρων PID) είναι να μειωθεί η μέγιστη γωνιακή ταχύτητα που ζητείται από τον SC. Τα αποτελέσματα της μείωσης της γωνιακής ταχύτητας φαίνονται στην επόμενη παράγραφο.



Εικόνα 62. Αποτέλεσμα δοκιμής κίνησης δυτικά με δεξιά στροφή και PID SC.

Δοκιμή No2 Κίνησης Δυτικά με Δεξιά Στροφή

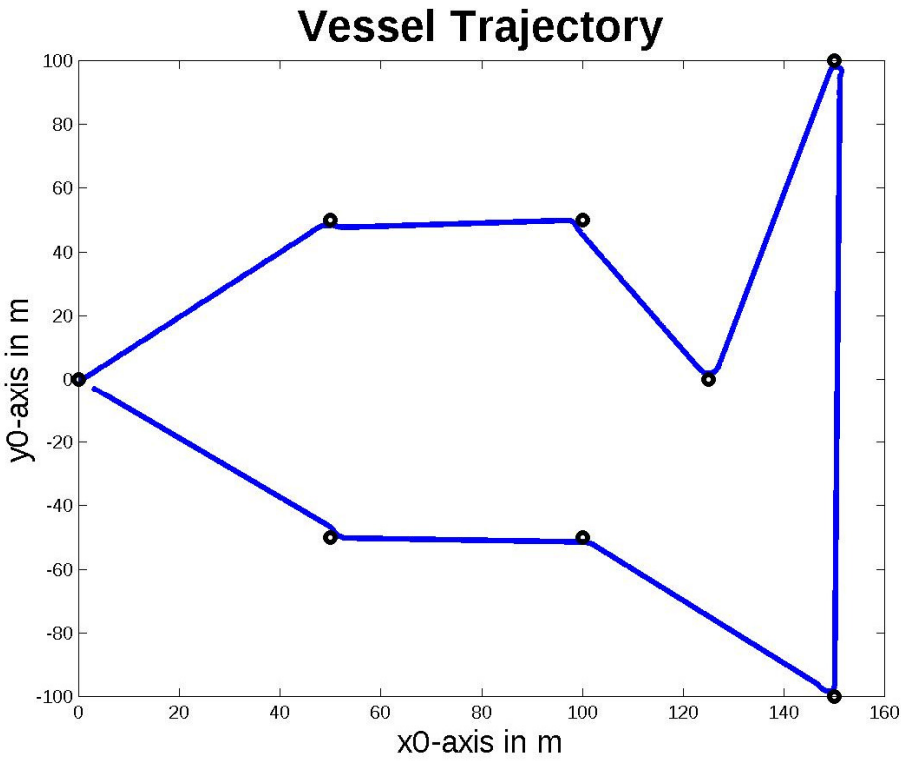
This test is exactly the same as the previous with the only exception that there is a yaw rate limiter at 25°/sec. This change fixes the overshoot a lot, as Εικόνα 63(a) shows.



Εικόνα 63. Αποτέλεσμα δοκιμής κίνησης δυτικά με δεξιά στροφή και PID SC με yaw rate limiting.

Δοκιμή Πολλαπλών Στόχων

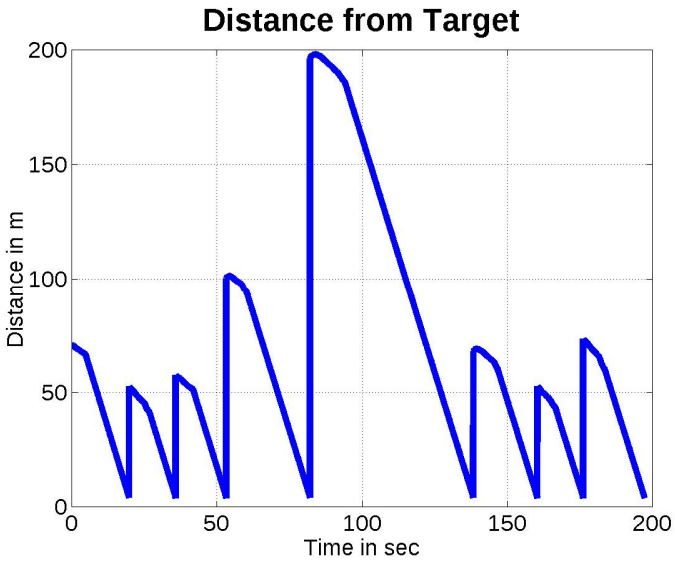
Όπως και με τον προηγούμενο τύπο εποπτικού ελεγκτή, η τελική δοκιμή είναι μια διαδρομή πολλαπλών σημείων που πρέπει να ακολουθηθεί.



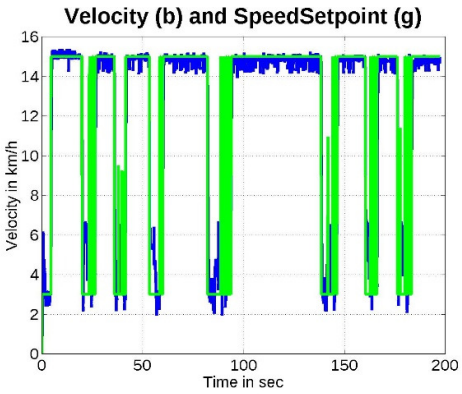
Εικόνα 64. Πορεία σκάφους στη δοκιμή προσέγγισης πολλαπλών στόχων με PID SC.

Σε σύγκριση με το μη γραμμικό SC της προηγούμενης ενότητας, όπως περιμέναμε και από τις προηγούμενες δοκιμές, η απόκριση είναι μια πιο αυστηρή διαδρομή με τη συντριπτική πλειοψηφία των μονοπατιών να είναι ευθείες. Αυτή η συμπεριφορά ήταν αναμενόμενη αφού είδαμε ανάλογη απόκριση του PID SC στις προηγούμενες δοκιμές.

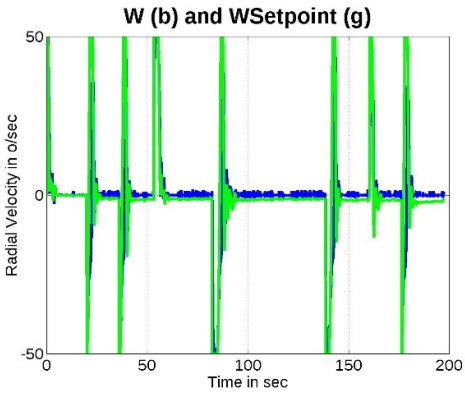
Όπως επίσης συζητήθηκε νωρίτερα, δεν υπάρχει «καλύτερη» και «χειρότερη» απόκριση μεταξύ των δύο τύπων SC. Για να έχει νόημα η σύγκριση, θα πρέπει να έχουμε γνώση των απαιτήσεων της χρήσης του σκάφους.



(a)



(c)



(d)

Εικόνα 65. Αποτέλεσμα δοκιμής προσέγγισης πολλαπλών στόχων με PID SC.

6 Επέκταση σε Σμήνη Σκαφών

Έχοντας δοκιμάσει την απόκριση του ενός σκάφους με ικανοποιητικά αποτελέσματα, το επόμενο βήμα είναι να προχωρήσουμε σε δοκιμές για σμήνη σκαφών.

Τα σενάρια μας θα έχουν ένα κύριο σκάφος ακολουθούμενο από σκάφη δορυφόρους σε προκαθορισμένο σχηματισμό.

Ξεκινάμε αυτές τις δοκιμές με την υπόθεση ότι το βασικό σκάφος θα πρέπει να χρησιμοποιήσει τον μη γραμμικό εποπτικό ελεγκτή της ενότητας 5.6. Αυτό του τύπου ο ελεγκτής παράγει ηπιότερες αποκρίσεις για το βασικό σκάφος και ως εκ τούτου είναι ευκολότερο να ακολουθηθεί. Οι δορυφόροι θα πρέπει να ανταποκρίνονται γρηγορότερα στις εντελλόμενες αλλαγές κατεύθυνσης, επομένως μια λογική εικασία είναι ότι ο εποπτικός ελεγκτής PID τους ταιριάζει περισσότερο.

6.1 Δοκιμή Κίνησης Δυτικά Σμήνους Τριών Σκαφών

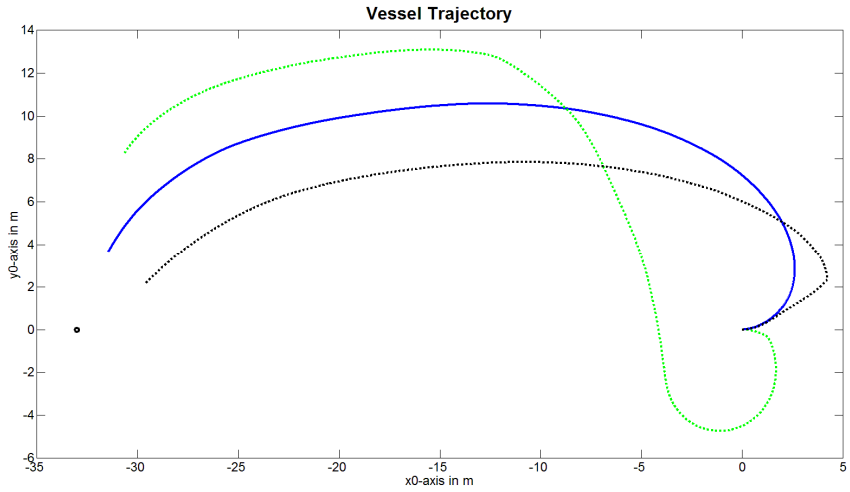
Σε αυτό το σενάριο θα χρησιμοποιήσουμε τρία σκάφη. Το σκάφος No 1 είναι το βασικό ή σκάφος οδηγός. Είναι αυτό στο οποίο ανατίθενται τα σημεία στόχοι.

Τα σκάφη No2 και 3 είναι δορυφόροι του 1, έχοντας απόσταση 3 μέτρα από το No1 σε γωνίες $+90^o$ και -90^o αντίστοιχα. Τα σημεία στόχου για κάθε έναν από τους δορυφόρους προσαρμόζονται δυναμικά, με βάση την τρέχουσα θέση του βασικού σκάφους και τους προκαθορισμένους στόχους του. Ο Πίνακας 3 περιγράφει τα χρώματα, τα σύμβολα και το τί αντιπροσωπεύουν στις παραστάσεις των τροχιών του σμήνους.

No	Σύμβολο - Χρώμα	Επεξήγηση
1	Μπλε	Πορεία σκάφους οδηγού.
2	Πράσινο	Πορεία σκάφους No2. Αφορά στο σκάφος με θέση 3m μακριά από το οδηγό σκάφος και σε γωνία $+90^o$ από αυτό.
3	Μαύρο	Πορεία σκάφους No3. Αφορά στο σκάφος με θέση 3m μακριά από το οδηγό σκάφος και σε γωνία -90^o από αυτό.
4	Έντονοι Κύκλοι	Σημεία στόχοι οδηγού σκάφους..
5	Μικροί Κύκλοι	Στόχοι πορείας δορυφόρων. Πράσινοι κύκλοι για το No2 και μαύροι για το No3.

Πίνακας 3: Συμβολισμοί και χρώματα σια τις τροχιές του σμήνους.

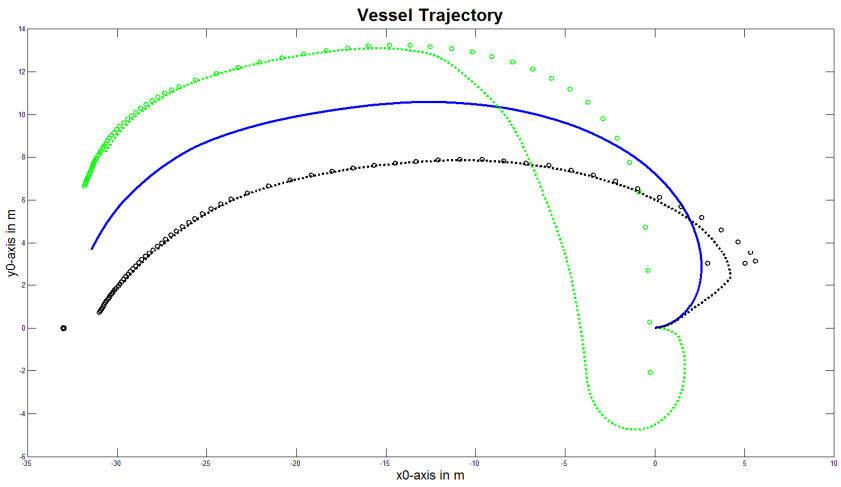
Τα αποτελέσματα αυτής της δοκιμής φαίνονται στην Εικόνα 66. Μόνο για λόγους προσομοίωσης, τα σκάφη θεωρούνται ως σημεία και όλα ξεκινούν από το (0,0). Το πράσινο σκάφος πρέπει να παραμείνει στις $+90^o$ της πορείας του οδηγού και το μαύρο στις -90^o και τα δύο σε απόσταση 3m.



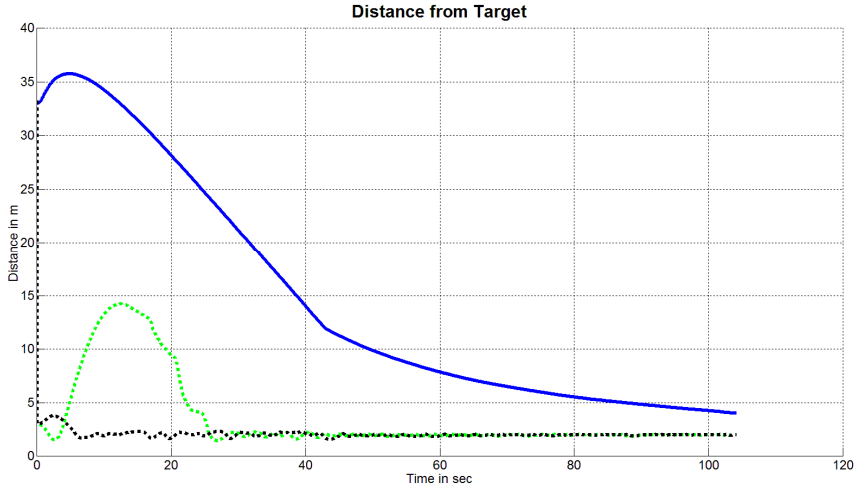
Εικόνα 66: Πορείες των τριών σκαφών κατά τη δοκιμή πορείας δυτικά.

Όπως φαίνεται στην Εικόνα 67, τα σκάφη μετά από ένα αρχικό μεταβατικό στάδιο που έχει να κάνει με το γύρισμα του πλοιάρχου, τα δορυφορικά σκάφη ακολουθούν την απαιτούμενη διαδρομή (όλα τα σχήματα που χρησιμοποιούνται σε αυτή την υποενότητα βρίσκονται στη θέση: “Matlab\Matlab_180910\ModelsRevisionAndControl\Code_synced_with_PHD_doc\FullSystemCode\SwarmTestBedIO\MiniSwarmTestBedResults_20230311_2228\ManualSave”).

Η Εικόνα 67 δείχνει τις τροχιές όλων των σκαφών, μαζί με επιλεγμένα στιγμιότυπα (ένα κάθε 50 περιόδους δειγματοληψίας) των σημείων - στόχων για τους δορυφόρους (μικροί κύκλοι).

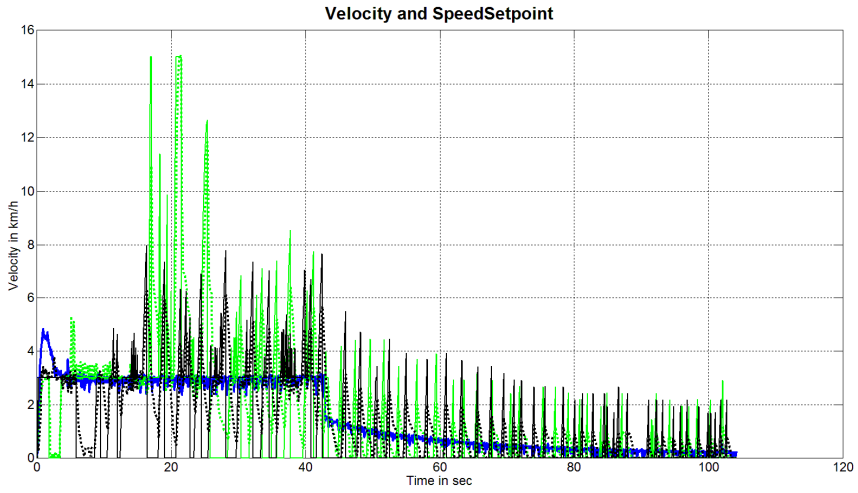


Εικόνα 67: Πορείες των τριών σκαφών, μαζί με στιγμιότυπα από τα σημεία στόχων για τους δορυφόρους.



Εικόνα 68: Απόσταση καθενός από τα σκάφη του σμήνους από το στόχο τους.

Όπως δείχνει η Εικόνα 68, η απόσταση όλων των σκαφών από τους στόχους τους μειώνεται από τη δράση ελέγχου.

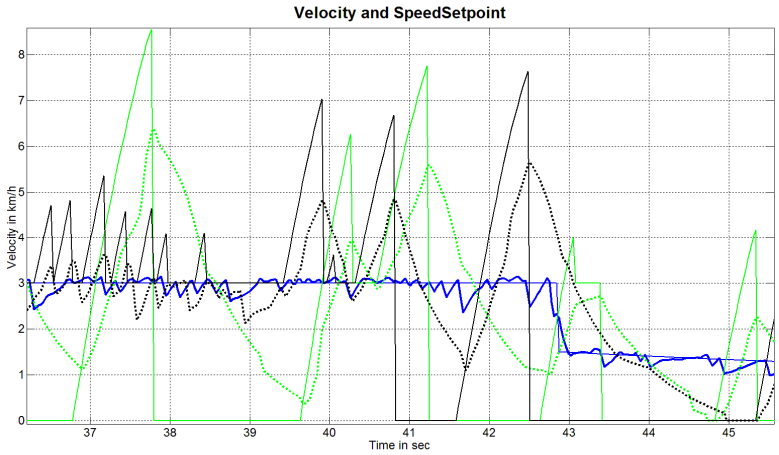


Εικόνα 69: Ταχύτητες σκαφών σε αντιπαραβολή με τις εντολές των εποπτικών ελεγκτών.

Η Εικόνα 69 περιέχει πολλές πληροφορίες και δεν είναι τόσο ευανάγνωστη. Δείχνει για κάθε σκάφος την ταχύτητα στόχο που ζητά ο εκάστοτε εποπτικός ελεγκτής, μαζί με την πραγματική ταχύτητα. Η Εικόνα 70 είναι μία εστίαση σε ένα μέρος της Εικόνα 69 και αποκαλύπτει ότι τα σκάφη ακολουθούν την ζητούμενη ταχύτητα με την έννοια ότι οι βρόχοι εσωτερικού ελέγχου μπορούν να διαχειριστούν τα αιτήματα των εποπτικών ελεγκτών. Ο Πίνακας 4 βοηθά στην κατανόηση των προαναφερθέντων σχημάτων.

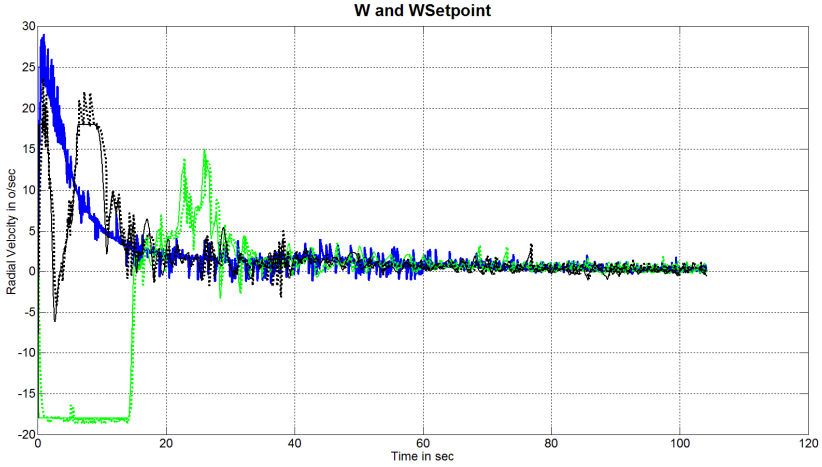
No	Σύμβολο ή/και Χρώμα	Επεξηγήσεις
1	Blue noisy curve	Master vessel velocity curve.
2	Blue smooth curve	Master supervisory requested velocity.
3	Green dotted curve	Satellite vessel 1 actual velocity curve.
4	Green smooth curve	Satellite vessel 1 supervisory requested velocity.
5	Black dotted curve	Satellite vessel 2 actual velocity curve.
6	Black smooth curve	Satellite vessel 2 supervisory requested velocity.

Πίνακας 4: Συμβολισμοί και χρώματα σχετιζόμενα με τις καμπύλες ταχυτήτων.

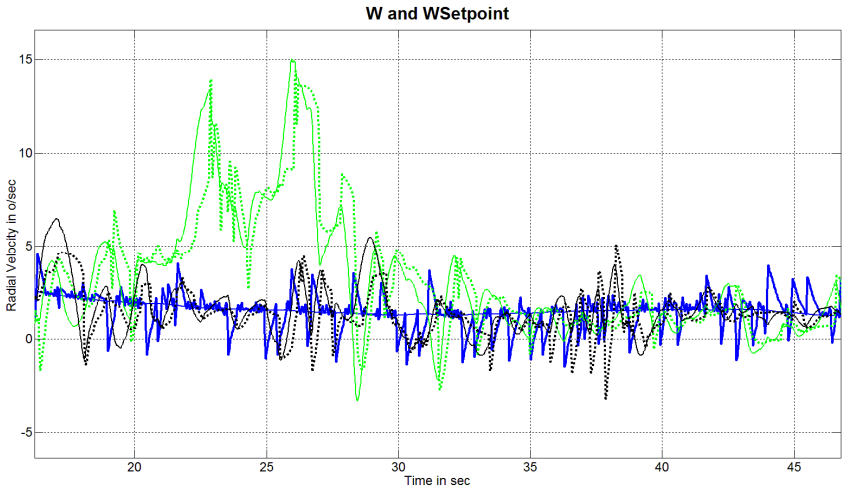


Εικόνα 70: Λεπτομέρεια της Εικόνα 69

Όλες οι πληροφορίες που παρέχονται από τις καμπύλες της Εικόνα 70 για την μεταφορική ταχύτητα, έχουν απόλυτη αντιστοιχία με τις καμπύλες της Εικόνα 71. Η Εικόνα 72 αποτελεί εστίαση σε συγκεκριμένη περιοχή της Εικόνα 71.



Εικόνα 71: Γωνιακή ταχύτητα για όλα τα σκάφη, μαζί με τις εντολές των εποπτικών ελεγκτών τους.



Εικόνα 72: Λεπτομέρεια της Εικόνα 71.

6.2 Δοκιμή Πολλαπλών Στόχων

Στη δοκιμή αυτή τα τρία σκάφη καλούνται να ακολουθήσουν μία διαδρομή, η οποία περνά από πολλούς στόχους, όπως στην υποενότητα **Error! Reference source not found.**.

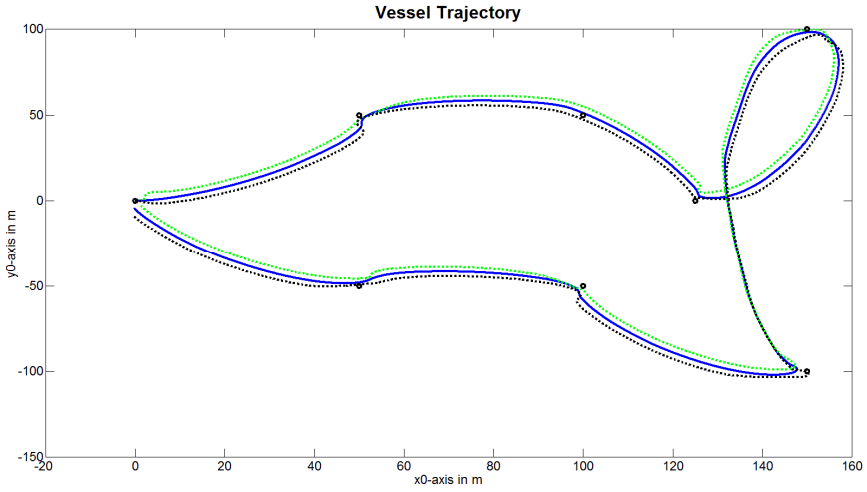
No	Target x	Target y
1	50	50
2	100	50
3	125	0
4	150	100
5	150	-100
6	100	-50
7	50	-50
8	0	0

Πίνακας 5: Κατάλογος σημείων - στόχων στο επίπεδο, για το οδηγό σκάφος.

Ο σχηματισμός, τα σύμβολα, τα χρώματα καμπυλών κ.λπ. είναι τα ίδια για τα τρία σκάφη, όπως συζητήθηκε στην προηγούμενη υποενότητα.

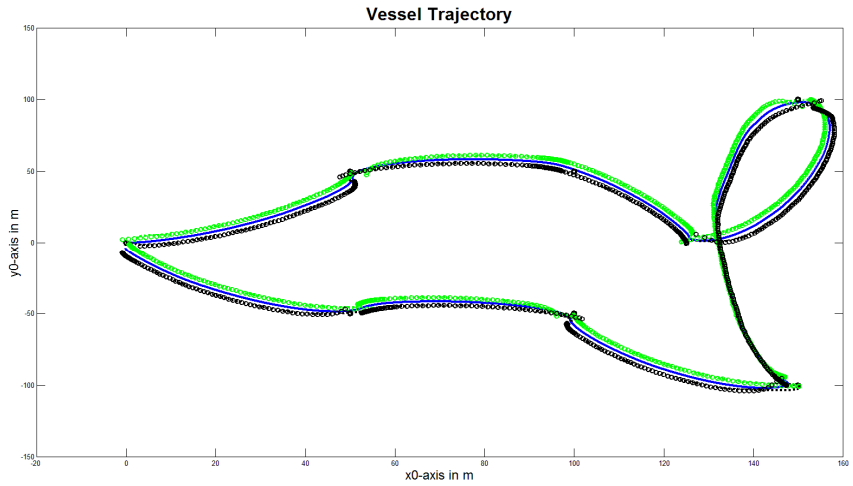
Το αποτέλεσμα αυτού του πειράματος προσομοίωσης ήταν επιτυχές, με τους δορυφόρους να ακολουθούν το οδηγό σκάφος, όπως φαίνεται στην .

Όλες οι εικόνες για την παράγραφο αυτή βρίσκονται στο φάκελο “Matlab\Matlab_180910\ModelsRevisionAndControl\Code_synced_with_PHD_doc\FullSystemCode\SwarmTestBedIO\MiniSwarmTestBedResults_20230311_1855\ManualSave”.



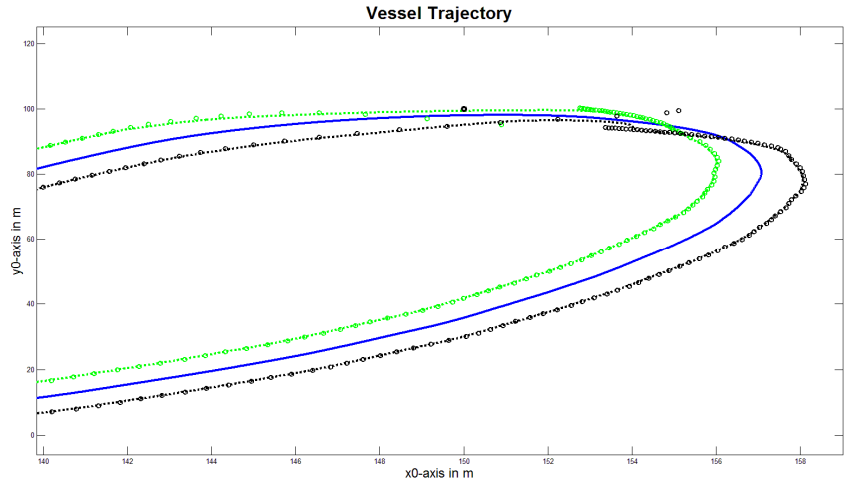
Εικόνα 73: Πορείες σμήνους σκαφών στο επίραμα πολλαπλών στόχων με μή γραμμικό εποπτικό ελεγκτή.

Error! Reference source not found. displays also snapshots of the targets of satellite vessels along the course of master vessel.



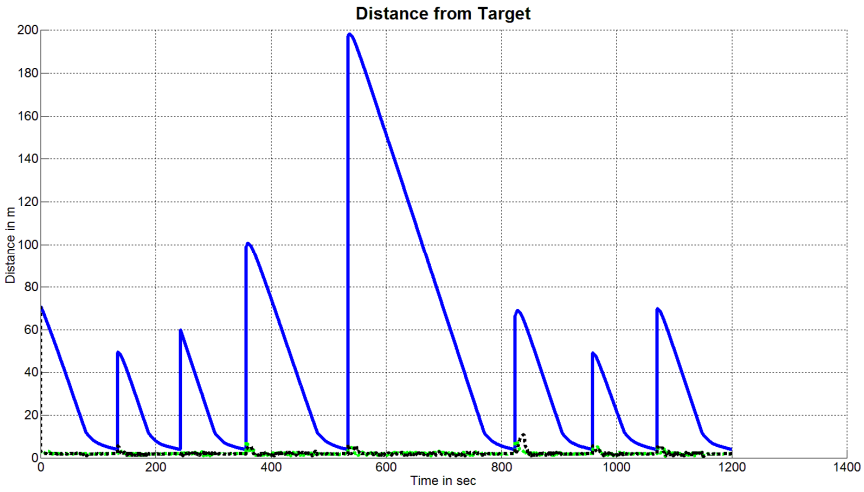
Εικόνα 74: Η Εικόνα 73 μαζί με τα σημεία - στόχους για κάθε δορυφόρο.

Η Εικόνα 75 είναι εστιασμένη σε ένα μικρό μέρος της Εικόνα 74 όπου φαίνεται η επιτυχία των δορυφόρων να ακολουθήσουν τα σημεία στα οποία εντέλλονται να βρίσκονται. Τα σημεία αυτά ξεχωρίζουν ως μικροί κύκλοι, ενώ οι πορείες των δορυφόρων είναι διάστικτες γραμμές.



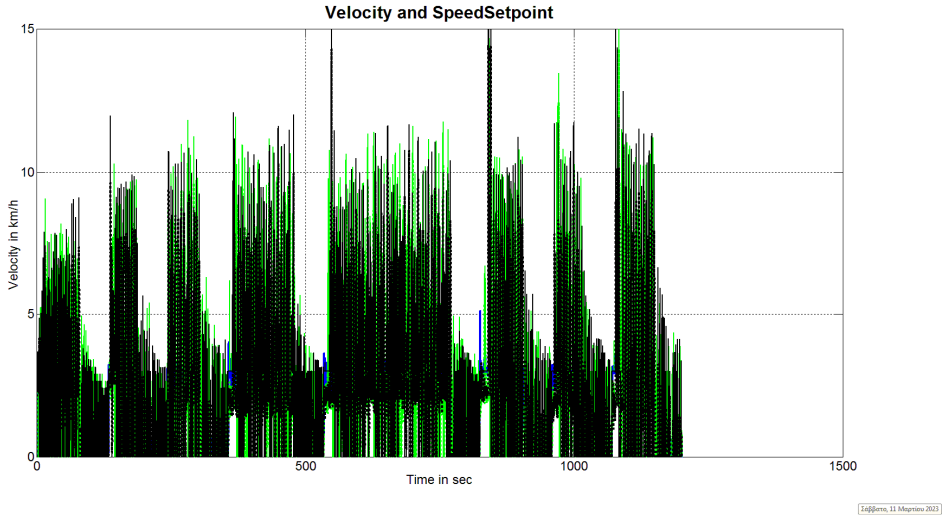
Εικόνα 75: Εστίαση σε λεπτομέρεια της Εικόνα 74.

Η Εικόνα 76 δείχνει πώς οι δράσεις του σχήματος ελέγχου μειώνουν την απόσταση από το στόχο για καθένα από τα σκάφη.

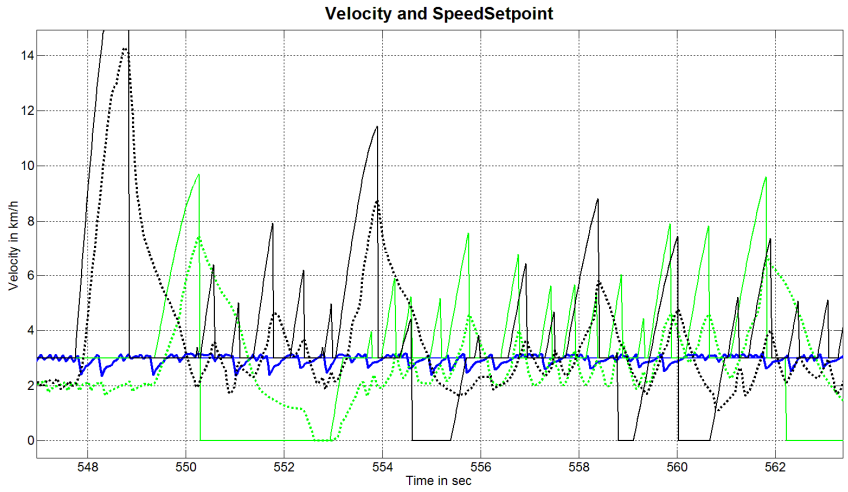


Εικόνα 76: Μεταβολή συναρτήσει του χρόνου της απόστασης από το στόχο για το κάθε σκάφος.

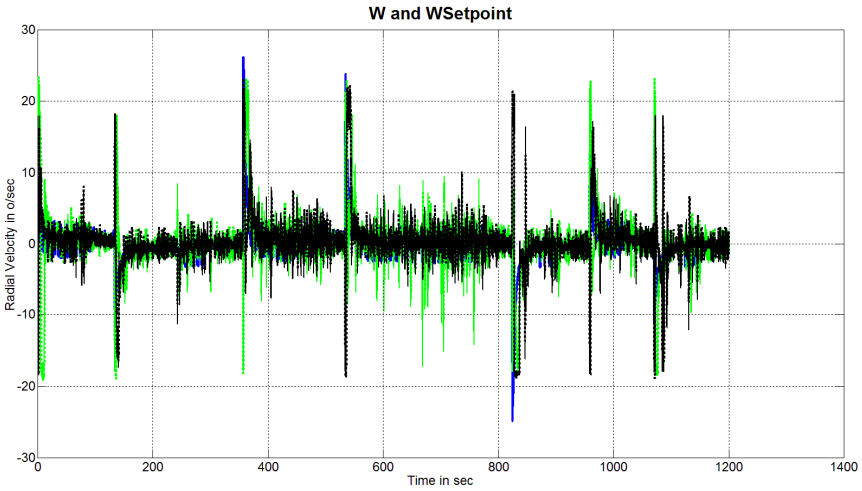
Τα σχήματα που ακολουθούν έχουν την ίδια περιγραφή που χρησιμοποιήθηκε στις προηγούμενες παραγράφους. Περιγράφουν ταχύτητες και για κάθε παράσταση ταχύτητας υπάρχει μια εστιασμένη εκδοχή της που ακολουθεί. Ο Πίνακας 4 ισχύει και για αυτή την υποενότητα.



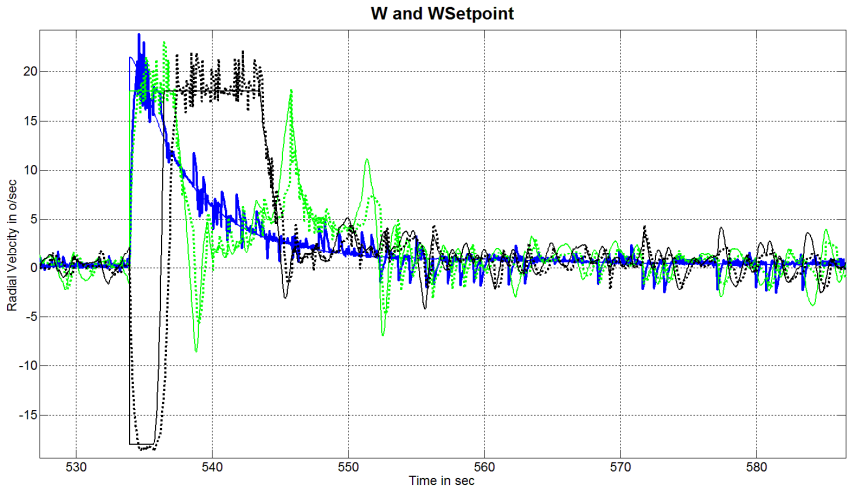
Εικόνα 77: Ταχύτητες και εντολές ταχύτητας για το σμήνος σκαφών.



Εικόνα 78: Λεπτομέρεια της Εικόνα 77.



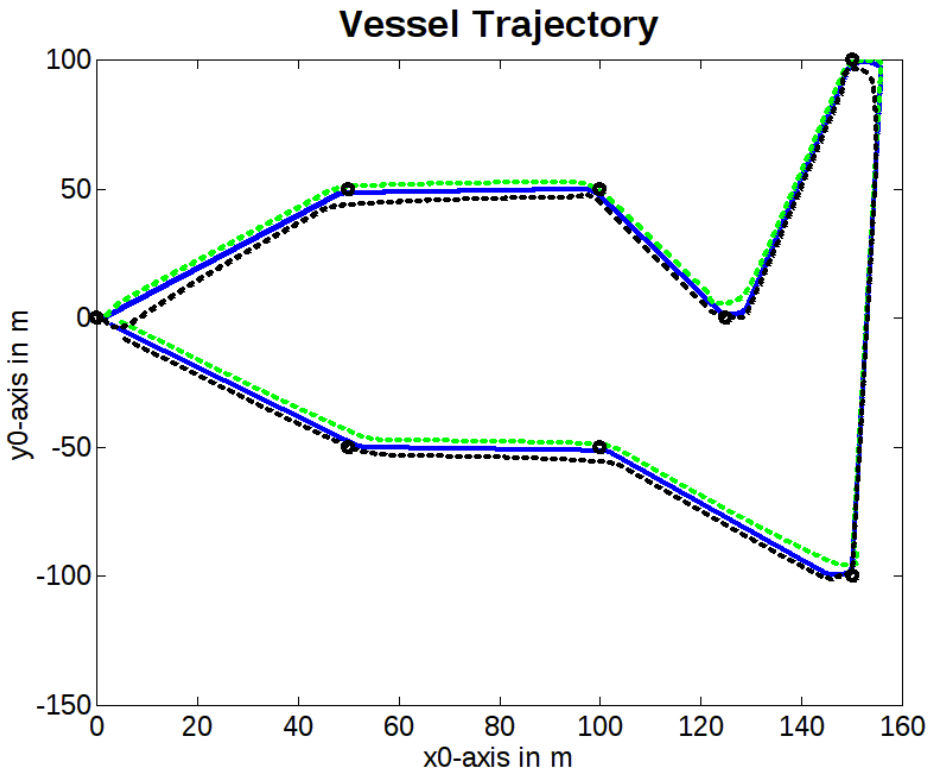
Εικόνα 79: Γωνιακές ταχύτητες και εντολές γωνιακών ταχυτήτων σμήνους.



Εικόνα 80: Λεπτομέρεια της Εικόνα 79.

6.3 Δοκιμή Πολλαπλών Στόχων με Εποπτικό Ελεγκτή Τύπου PID για το Σκάφος Οδηγό

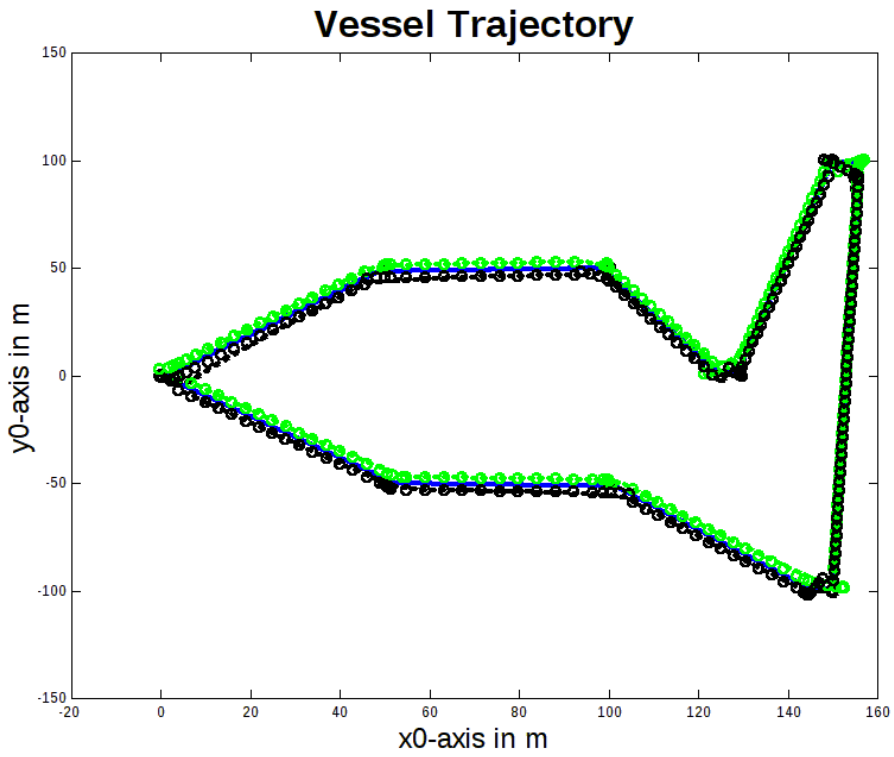
Όλες οι εικόνες της παραγράφου βρίσκονται στο φάκελο: "Matlab\Matlab_180910\ModelsRevisionAndControl\Code_synced_with_PHD_doc\FullSystemCode\SwarmTestBedIO\MiniSwarmTestBedResults_20230311_1855\ManualSave".



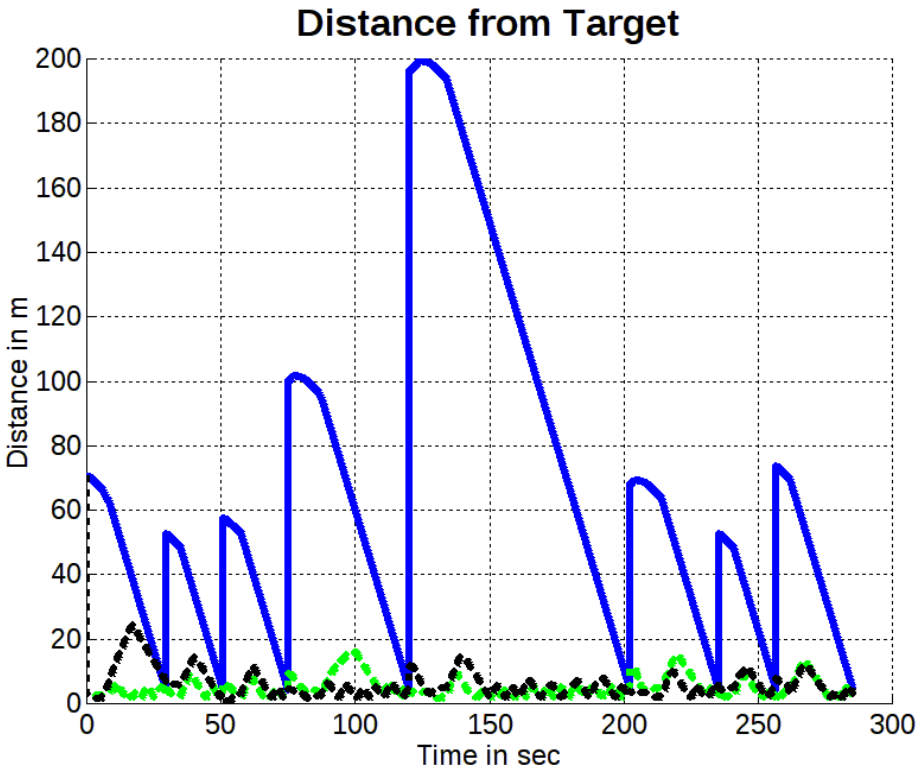
Εικόνα 81: Πορείες σκαφών σε δοκιμή πολλαπλών στόχων, με το οδηγό σκάφος να έχει εποπτικό ελεγκτή τύπου PID.

Αυτό το πείραμα δείχνει ότι ακόμη και αν χρησιμοποιήσουμε εποπτικό ελεγκτή τύπου PID για το σκάφος οδηγό, οι δορυφόροι επιτυγχάνουν να ακολουθήσουν. Ωστόσο, έχουμε τροποποιήσει τις ρυθμίσεις του σκάφους οδηγού, αναφορικά με τις μέγιστες επιτρεπόμενες ταχύτητες. Το σκάφος οδηγός περιορίζεται λίγο ως προς τις μέγιστες επιτρεπόμενες ταχύτητες, προκειμένου να επιτρέπει στους δορυφόρους να συγκλίνουν ταχύτερα. Αυτό διευκολύνει τους δορυφόρους να παρακολουθούν τις κινήσεις του οδηγού.

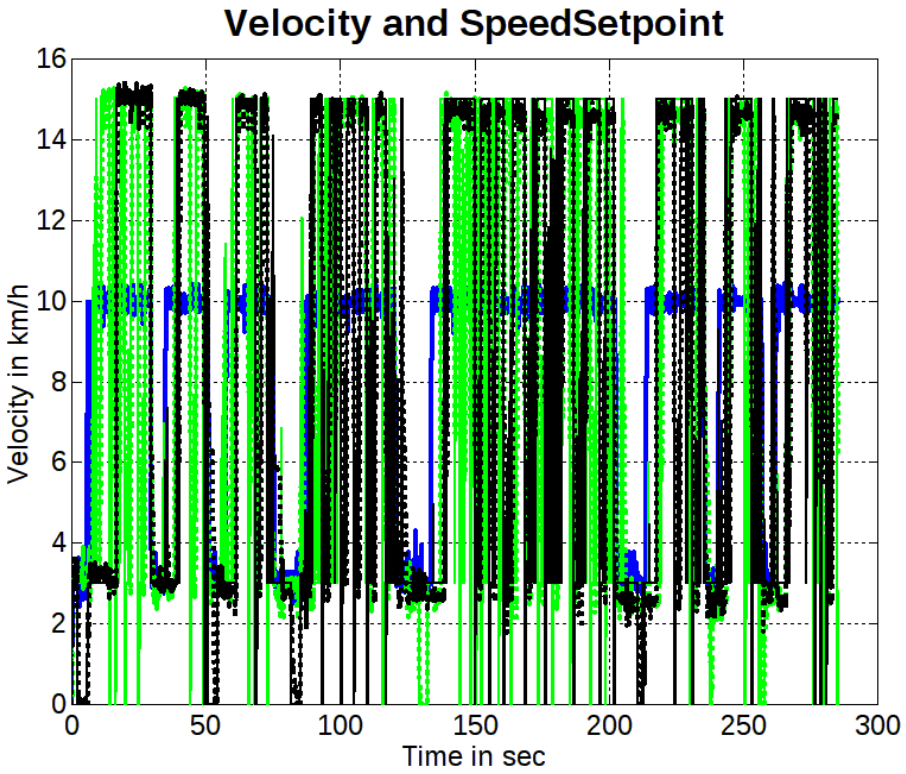
Οι εικόνες που εμφανίζονται σε αυτήν την υποενότητα, ακολουθούν τη λογική και το συμβολισμό του Πίνακας 4.



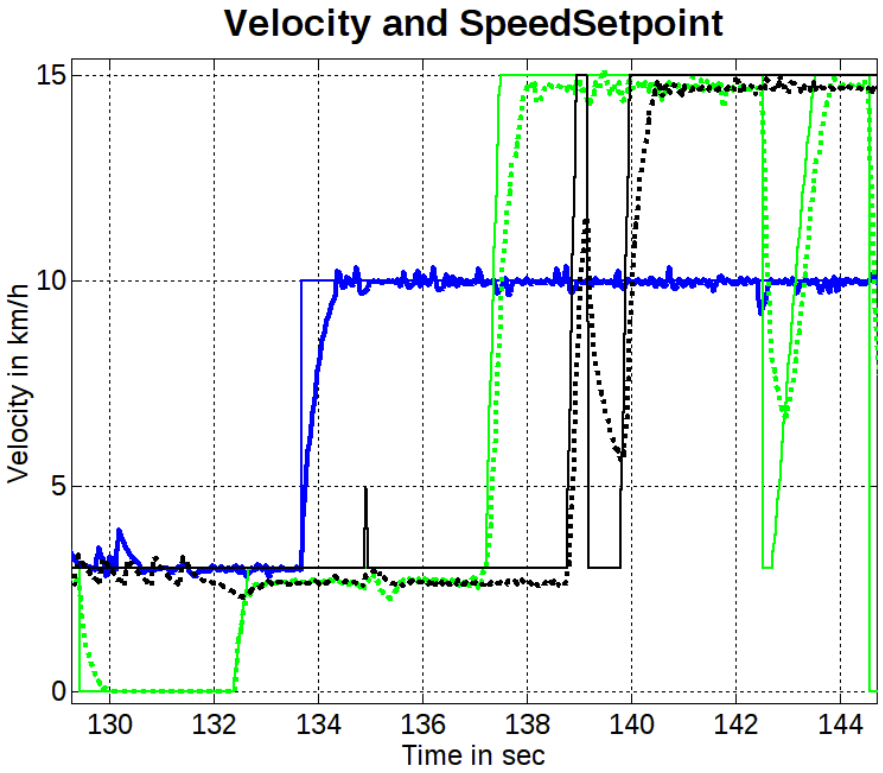
Εικόνα 82: Προσθήκη στην Εικόνα 81 των σημείων - στόχων για τις πορείες των δορυφόρων.



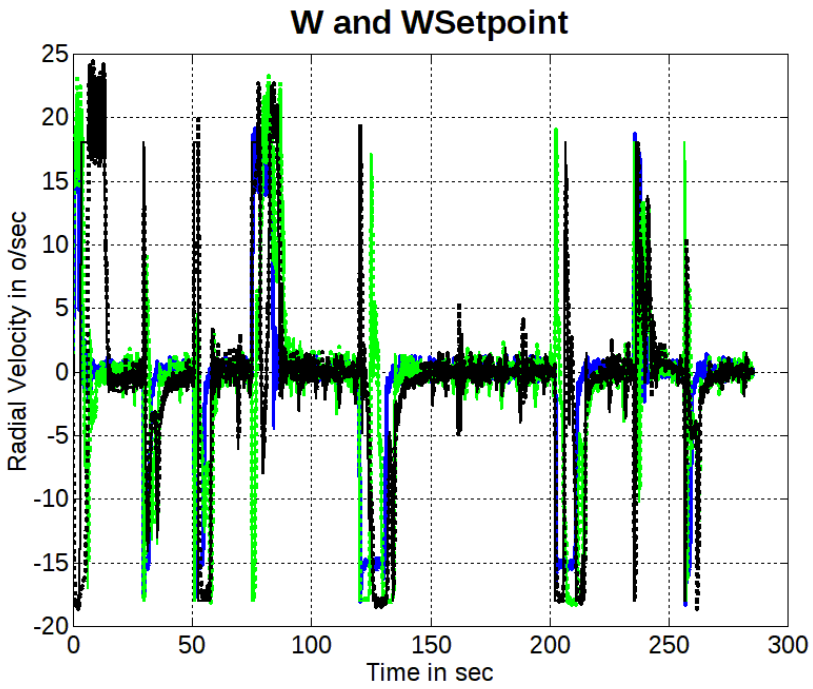
Εικόνα 83: Μεταβολή απόστασης από στόχο ως προς χρόνο, για το σμήνος.



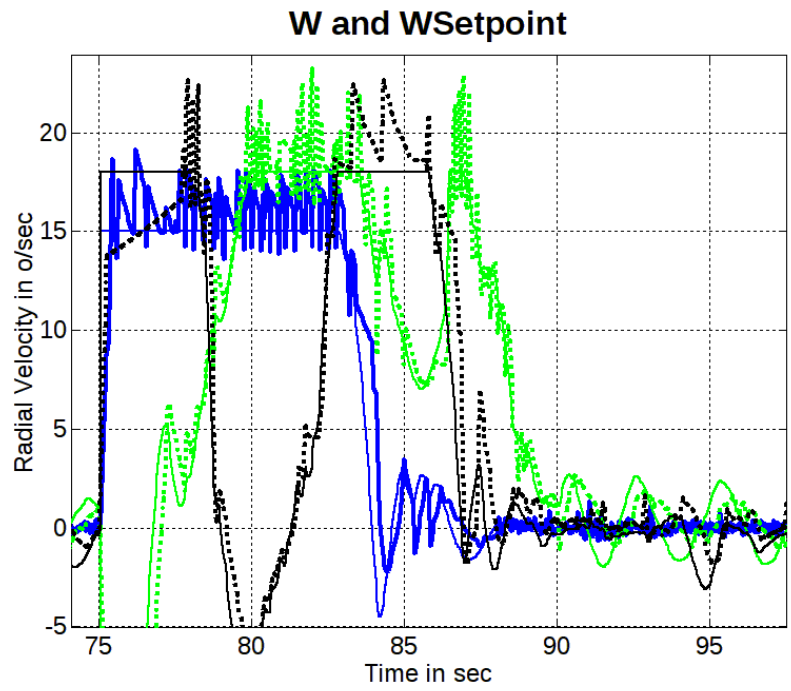
Εικόνα 84: Ταχύτητες σκαφών και εντολές ταχυτήτων από τους εποπτικούς ελεγκτές τους.



Εικόνα 85: Εστίαση σε λεπτομέρεια της Εικόνα 84.



Εικόνα 86: Γωνιακές ταχύτητες σκαφών και εντολές ταχυτήτων από τους εποπτικούς ελεγκτές τους.



Εικόνα 87: Εστίαση σε λεπτομέρεια της Εικόνα 86.

7 Προτάσεις για Μελλοντική Έρευνα

Η παρούσα έρευνα μπορεί να επεκταθεί με πολλούς τρόπους. Ένας τρόπος περιγράφεται στο κεφάλαιο 13 του κειμένου στην Αγγλική, “APPENDIX 1: Foundations of Future Work”, στο οποίο περιγράφουμε έναν επιπλέον αλγόριθμο επεξεργασίας θέσεων του σκάφους. Ο αλγόριθμος αυτός μιμείται τις κινήσεις φορτίων σε ένα τεχνητό πεδίο, που χαρακτηρίζεται από ομοιότητες με το ηλεκτρικό πεδίο. Στο πεδίο αυτό δημιουργούμε κατάλληλους “αυθαίρετους” νόμους, με στόχο την κίνηση των σκαφών σε αλληλεξάρτηση, αλλά και την αποφυγή μεταξύ τους συγκρούσεων.

Ένας άλλος, περισσότερο προφανής, τρόπος είναι να προστεθεί αλγόριθμος αυτόνομου σχηματισμού σμήνους, χωρίς σκάφος αναφοράς. Τα σχηματιζόμενα σμήνη μπορούν να χρησιμοποιηθούν προκειμένου να αξιολογηθούν διαφορετικοί αλγόριθμοι σχηματισμού.

Τέλος, ενδείκνυται η επέκταση σε σκάφη μεγαλύτερης διάστασης, με πολύ μεγαλύτερο εύρος εφαρμογών, ακόμα και για ανοιχτές θάλασσες. Τέτοιες εφαρμογές μπορεί να περιλαμβάνουν:

- Δημιουργία στόλου έρευνας και διάσωσης. Πολλά σκάφη εξοπλισμένα με θερμικές κάμερες και μικρόφωνα μεγάλης ευαισθησίας, κατευθυνόμενα με τεχνητή νοημοσύνη, σε αναζήτηση ανθρώπων μετά από ναυάγιο. Οι αλγόριθμοι τεχνητής νοημοσύνης μπορούν επίσης να διαμορφώσουν τον σχηματισμό που απαιτείται για τον περιορισμό των επιζώντων σε μια περιοχή, προκειμένου να διευκολυνθεί η επιχείρηση διάσωσης.

- Μεταφορές ανοιχτής θαλάσσης. Ένα βασικό εμπορικό πλοίο υπό ανθρώπινο έλεγχο, ακολουθούμενο από πολλά αυτόνομα εμπορικά πλοία διαφόρων μεγεθών σε σχηματισμό. Τα αυτόνομα πλοία μπορούν να μεταφέρουν το επικίνδυνο μέρος του εμπορεύματος, προκειμένου να μην εκτίθεται ανθρώπινο προσωπικό σε κίνδυνο.

- Αυτόνομα ρυμουλκά που προσκολλώνται σε ένα πλοίο, για να το οδηγούν σε στενούς και περίπλοκους διαδρόμους, προστατεύοντάς το από πιθανά επιβλαβείς επαφές-συγκρούσεις.

- Αυτόνομα ρυμουλκά που τοποθετούν με ακρίβεια τα τρυπάνια για εξόρυξη πετρελαίου.

- Αυτόνομος στόλος που εκτελεί χαρτογράφηση βυθού για διάφορες εφαρμογές.

Η λίστα μπορεί να επεκταθεί με πολλές άλλες εφαρμογές. Ωστόσο, η πρόοδος κάθε εφαρμογής θα εξαρτηθεί από την αφοσίωση, τα διαθέσιμα μέσα και την οικονομική στήριξη των ερευνητικών ομάδων που θα εμπλακούν. Τα ηλεκτρονικά που αναπτύχθηκαν στο πλαίσιο της παρούσας διατριβής μπορούν να τροποποιηθούν, ώστε να υποστηρίξουν οποιαδήποτε από αυτές τις υλοποιήσεις, με τις εκτιμήσεις του απαιτούμενου χρόνου ανάπτυξης να κυμαίνονται από αρκετούς μήνες έως ενάμιση χρόνο, ανάλογα με τον αριθμό και τα interfaces των πρόσθετων αισθητήρων που απαιτούνται, καθώς και την έκταση των προσαρμογών του λογισμικού.

8 Δημοσιεύσεις

[1] E. K. Loghis and N. I. Xiros, “Development of Discrete-Time Waterjet Control Systems Used in Surface Vehicle Thrust Vectoring,” *Journal of Marine Science and Engineering*, vol. 10, no. 12, 2022, doi: 10.3390/jmse10121844.

[2] E. C. Loghis, N. I. Xiros, and E. C. Loghis, “Continuous and Discrete-time Models of Surface Watercraft Nonlinear Dynamics,” *Acta Scientific Computer Sciences*, 2022.

[3] G. D. Ntouni *et al.*, “Real-time Experimental Wireless Testbed with Digital Beamforming at 300 GHz,” in *2020 European Conference on Networks and Communications (EuCNC)*, Jun. 2020. doi: 10.1109/eucnc48522.2020.9200948.

[4] L. Wang, N. I. Xiros, and E. K. Loghis, “Design and Comparison of H$ınfty$/H2 Controllers for Frigate Rudder Roll Stabilization,” *Journal of Marine Science and Application*, vol. 18, no. 4, pp. 492–509, Oct. 2019, doi: 10.1007/s11804-019-00116-3.

[5] N. Xiros, V. Tzelepis, and E. Loghis, “Modeling and Simulation of Planing-Hull Watercraft Outfitted with an Electric Motor Drive and a Surface-Piercing Propeller,” *Journal of Marine Science and Engineering*, vol. 7, no. 2, p. 49, Feb. 2019, doi: 10.3390/jmse7020049.

[6] E. C. Loghis, N. I. Xiros, and R. Saxton, *A Dynamic Interaction Simulator for Studying Macroscopic Swarm Self-organization of Autonomous Surface Watercraft*, vol. All Days. 2016.

[7] N. I. Xiros, E. Logis, E. Gasparis, S. Tsolakidis, and K. Kardasis, “Theoretical and experimental investigation of unmanned boat electric propulsion system with PMDC motor and waterjet,” *Journal of Marine Engineering &amp$\mathsemicolon$ Technology*, vol. 8, no. 2, pp. 27–43, Jan. 2009, doi: 10.1080/20464177.2009.11020221.

[8] N. Xiros and E. Loghis, “System Identification Using Neural Nets for Dynamic Modeling of a Surface Marine Vehicle,” Nov. 2014, p. V010T13A015. doi: 10.1115/IMECE2014-38322.

[9] M. G. Farmakopoulos, E. K. Loghis, P. G. Nikolakopoulos, N. I. Xiros, and C. A. Papadopoulos, “Modeling and Control of the Electrical Actuation System of an Active Hydromagnetic Journal Bearing (AHJB),” in *Volume 4B: Dynamics, Vibration, and Control*, Nov. 2014. doi: 10.1115/imece2014-38346.

[10] N. I. Xiros and E. K. Loghis, “Dynamic Modeling and System Identification of a Surface Marine Vehicle for Autonomous Swarm Applications,” in *Volume 4: Dynamics, Control and Uncertainty, Parts A and B*, Nov. 2012. doi: 10.1115/imece2012-89404.

[11] E. C. Loghis, N. I. Xiros, and E. C. Loghis, “An Automatic Steering System for Robust Disturbance Rejection,” in *IASME Transactions2*, 2004.

Οι δημοσιεύσεις είναι διαθέσιμες στο https://orcid.org/0000-0002-3743-131X

9 Αναφορές

[1] T. Fossen, *Guidance and Control of Ocean Vehicles*, 1st ed. Michigan: Wiley, 1994.

[2] T. Perez, *Ship Motion Control: Course Keeping and Roll Stabilisation Using Rudder and Fins*. in Advances in Industrial Control. Springer, 2005. [Online]. Available: https://books.google.gr/books?id=_7apbonEQHgC

[3] "Models for Ships, Offshore Structures and Underwater Vehicles," in *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons, Ltd, 2011, pp. 133–186. doi: https://doi.org/10.1002/9781119994138.ch7.

[4] K. Ogata, *Discrete-Time Control Systems*, 2nd ed. New Jersey: Prentice-Hall Inc., 1995.

[5] E. K. Loghis and N. I. Xiros, "Development of Discrete-Time Waterjet Control Systems Used in Surface Vehicle Thrust Vectoring," *Journal of Marine Science and Engineering*, vol. 10, no. 12, 2022, doi: 10.3390/jmse10121844.

[6] N. W. H. Bulten, "Numerical analysis of a waterjet propulsion system," PhD Thesis, Mechanical Engineering, Eindhoven, 2006. doi: 10.6100/IR614907.

[7] N. I. Xiros, E. Logis, E. Gasparis, S. Tsolakidis, and K. Kardasis, "Theoretical and experimental investigation of unmanned boat electric propulsion system with PMDC motor and waterjet," *Journal of Marine Engineering & Technology*, vol. 8, no. 2, pp. 27–43, Jan. 2009, doi: 10.1080/20464177.2009.11020221.

[8] N. I. Xiros, "Digital Signal Processing," in *Springer Handbook of Ocean Engineering*, M. R. Dhanak and N. I. Xiros, Eds., Cham: Springer International Publishing, 2016, pp. 197–226. doi: 10.1007/978-3-319-16649-0_9.

[9] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 1st ed. US: Prentice-Hall Inc., 1989.

[10] Σ. Γ. Τζαφέστας, *Υπολογιστική Νοημοσύνη Τόμος Α: Μεθοδολογίες*, 1st ed., vol. 1, 2 vols. Πανεπιστημιακές Εκδόσεις Ε.Μ.Π., 2002.

[11] S. Haykin, S. S. Haykin, and S. A. HAYKIN, *Neural Networks: A Comprehensive Foundation*. in International edition. Prentice Hall, 1999. [Online]. Available: https://books.google.gr/books?id=bX4pAQAAMAAJ

[12] N. I. Xiros and E. K. Loghis, "System Identification Using Neural Nets for Dynamic Modeling of a Surface Marine Vehicle," in *Volume 10: Micro- and Nano-Systems Engineering and Packaging*, American Society of Mechanical Engineers, Nov. 2014. doi: 10.1115/imece2014-38322.

[13] Σ. Αλκ. Παϊπέτης, *Τεχνική Μηχανική ΙΙ Δυναμική*, 3rd ed., vol. 2, 3 vols. in Τεχνική Μηχανική, vol. 2. Αθήνα: Amatec S.A., 1992.

[14] C. J. Zarowski, *An Introduction to Numerical Analysis for Electrical and Computer Engineers*. Wiley, 2004. [Online]. Available: https://books.google.gr/books?id=3AihEG52ImkC

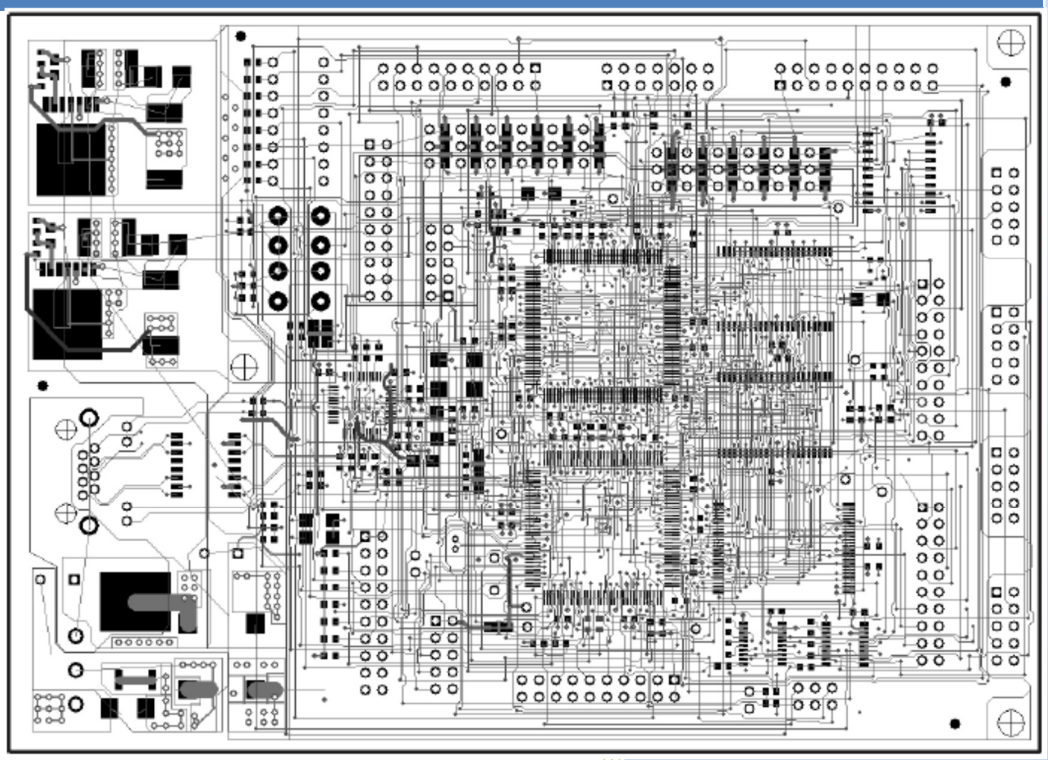
[15] Β. Β. Μάρκελλος, *Αριθμητικές Μέθοδοι*, 1st ed. Αθήνα: Εκδόσεις Συμμετρία, 1995.

[16] B. C. Kuo, *Digital Control Systems*, 1st ed. Hong Kong: CBS Publishing Asia LTD, 1987.

[17] L. A. Pipes and L. R. Harvill, *Applied Mathematics for Engineers and Physicists: Third Edition*. in Dover Books on Mathematics. Dover Publications, 2014. [Online]. Available: https://books.google.gr/books?id=cvmJAwAAQBAJ

[18] I. J. Nagrath and M. Gopal, *Control Systems Engineering*, 2nd ed. Singapore: John Wiley & Sons, 1986.
[19] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems*, 10th ed. US: Wiley, 2010.
[20] N. I. Xiros, *Robust Control of Diesel Ship Propulsion*, 1st ed. London: Springer London, 2002. [Online]. Available: https://doi.org/10.1007/978-1-4471-0191-8

Robust Control
Applied to Surface Vessel Guidance



Eleftherios C. Loghis

Table of Contents

1 Introduction

1.1 Abstract

This PHD thesis deals with the electronics design, modeling, and control of a surface vessel created for reaching specified sets of targets on a two-dimensional map. It is based on a remote-control model of a ferry boat ship, that was assembled by NTUA. All the electronic equipment used, apart from the remote-control unit and its receiver, were designed and manufactured by the author of this thesis.

The details of the development are reported on the thesis, along with the methodology followed for its modelling and control.

After the construction of the vessel, data for its response were collected by performing manual maneuvers using the radio control unit. This data was used to train a neural network, to predict the vessel's response to thrust and steering controls, in a black-box modelling approach. The model that resulted was used to conduct simulations on the control methods developed for the vessel.

For enabling the vessel to reach specified targets on the map, two types of supervisory controllers were synthesized. These algorithms monitor the position of the vessel in relation to its target and generate translational and angular velocity directives that serve as setpoints for the respective control loops.

To effectively cope with the non-linearities of the system, a methodology based on the neural network model was developed, that also decoupled velocity and heading control loops. Robust control techniques were also employed, to reduce the effect of model prediction and control action quantization inaccuracies and attenuate the resulting error.

The response of the vessel was then simulated using several test cases, to examine its maneuvering abilities. A comparison of the two supervisory controllers was also conducted.

Finally, a mini swarm of three vessels was simulated, to investigate the expansion of the methods described in the thesis, in sets of related vessels.

1.2 Novelty of this work and contribution to scientific research

The novelty of the present work is associated with the following points.

The vessel modelling approach.

Modelling of a surface vessel using solely neural networks is not common practice. The reason for this selection was twofold. Firstly, the size of this vessel is small with respect to even the slightest disturbances from the water surface, thus rendering it less accurate for our purpose to adopt other approaches, such as, for example, the Nomoto equations. Secondly, from an academic point of view, it is important to test whether using a "black box" modelling approach can be effective for the class of problems considered here.

Thus, the modelling framework of the present work is substantially different from that of corresponding literature studies, which are based on either physical models or on the Nomoto equations [1]–[3].

PID algorithm implementation using the summation approximation of integral.

In most of the associated literature, implementation of PID control is based on digitizing PID by means of trapezoidal integration – see book by Ogata [4] and the discussion in the present Thesis ("APPENDIX 5: Alternative Implementation of PID Controllers (trapezoidal Integration)"). Since our simulations used summation approximations to integrate accelerations and velocities, we have selected to digitize PID using the summation approximation, to unify our approach regarding integration. The entire procedure is described in the present Thesis and was also published in [5].

Introduction of the "Model Based Controller" (MBC) unit.

The MBC unit, introduced in the present study, creates the relation between the requested accelerations and necessary actuation, by optimizing its selection based on the attainable responses, as predicted by repeatedly applying potential actuations to the vessel neural model. To our knowledge, a similar approach has not been used in open literature. A detailed description is presented in section 5.5.1.

The non-linear supervisory control algorithms.

To our knowledge, the non-linear set of equations that was used to create the laws that guided the vessel to its target point has not been reported in the open literature. It is introduced here for the first time.

1.3 Aknowledgements

As a personal conception, I believe that the effect of our personal efforts is multiplied by a factor that has to do with the people surrounding and supporting us. If these people act supportively, then our energy is multiplied, and we reach our goals a lot easily no matter how hard they are.

Thus, I have to be thankful for having people to my support, that believed in me even when there was no apparent progress.

For the above and many more reasons, I would like to thank Professor Nikolaos Xiros for accepting me as his student and providing me with all his support for the completion of this thesis. I also need to thank professors Tryfon Kousiouris and Nikolaos Maratos for their support in regard to my university obligations.

At this point I feel obliged to devote a separate paragraph to my wife Ioanna Koukopoulou, who has tolerated uncountable hours of my being isolated for studying, implementing or writing, even in our vacations (in fact to rephrase it correctly mainly during our vacations!!). Without her support and understanding completing this thesis would be impossible.

For many reasons I must express my gratitude to my brother Dimitrios Loghis, who is the person that familiarized me with the world of electronic development and turned my interest to passion for designing and programming electronics. He also assisted in construction, during the measurements for the mathematical modeling of the vessel and many other times for which a list is not possible...

Speaking of family members (!), I must be thankful to my aunt Despoina Gourvelou, a mathematician, who put her best efforts into making me develop an as solid background in mathematics as I could, from the early stages of my life to university graduation.

Constantine Xanthos PhD is also a mathematician I must be very thankful too, who assisted me by teaching advanced mathematical subjects during my thesis without accepting any kind of payment!

I also want to thank N. Kouvaras for helping me a lot at the early stages of the development, by assembling the vessel's hull and providing assistance with the mechanical components.

Last but not least, I would like to thank my daytime job managers and colleagues, that supported me every time I had to leave the office for my doctoral thesis obligations.

1.4 Symbols, Variables, Notations and Abbreviations

The aim of this paragraph is to collect all symbols needed in order to understand the meaning of variables and abbreviations used throughout this thesis.

Symbols List	
Symbol	**Definition**
f_c	Continuous Time Neural Network model for surface vessel.
y_T	Engines throttle from user remote control.
δ_R	Turn from user remote control.
y_{TA}	Auto-generated thrust command.
δ_{RA}	Auto-generated steering command.
g	New system that creates remote control outputs (y_T, δ_R), based on $\vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, in such a way as to provide linear system response with respect to $\vec{v}$.
A	The Jacobian of the system responses.
B	The Jacobian of the input matrix.
u_I	Input vector to the linearized system.
u	Translational speed of the vessel in Km/h.
ω	Angular speed of the vessel in °/sec.
x	The system state, i.e. $\begin{bmatrix} u \\ \omega \end{bmatrix}$.
v	Vector of the user set-points for
α	Translational acceleration of vessel ($=\dot{u}$) in m/(sec)^2.
$\dot{\omega}$	Angular acceleration in °/(sec)^2.
$y_{\%}$	Normalized throttle command, for use with neural networks, using formula: $$y_{\%} = \frac{y_T}{256}$$
$\delta_{\%}$	Normalized turn command, for use with neural networks, using formula: $$\delta_{\%} = \frac{\delta_R - 121}{256}$$
$u_{\%}$	Normalized translational velocity, for use with neural networks, using formula: $$u_{\%} = \frac{u}{30\text{Km/h}}$$
$\omega_{\%}$	Normalized angular velocity, for use with neural networks, using formula: $$\omega_{\%} = \frac{\omega_T}{50^o/\text{sec}}$$

$\alpha_{\%}$ or $\dot{u}_{\%}$	Normalized translational acceleration , for use with neural networks, using formula: $$\alpha_{\%} = \frac{\alpha}{3.4\text{m/(sec)}^2}$$
$\dot{\omega}_{\%}$	Normalized angular acceleration, for use with neural networks, using formula: c
y_C	Throttle command from Feedback Controller.
δ_C	Turn command from Feedback Controller.
Ox_0y_0	Absolute reference frame of the two-dimensional space where the vessel and target points exist.
$O'x_Ry_R$	Frame that describes the position of the vessel's center of mass, relative to Ox_0y_0
O'xy	Body-fixed reference frame which follows the vessel's angle of turn
$\vec{R}$	Vector marking the (common) position of frames $O'x_Ry_R$ and O'xy relative to Ox_0y_0.
$\vec{r}$	Vector connecting O' to the next target point.
$\vec{T}$	Vector connecting O to the next target point.
(x_t, y_t)	Target point coordinates with Ox_0y_0 as a reference frame.
θ	Angle between vessel's prow and vector $\vec{r}$. Positive counter-clockwise.
φ	Angle between vessel's prow and x_R axis. Positive counter-clockwise.
φ_r	Angle between vector $\vec{r}$ and x_R or x_0 axis. Positive counter-clockwise.
T_s	Sampling rate for all operations of the system. The sampling rate used is 0.03s.

Table 1: Singals, units and notations.

Abbreviations	
Abbreviation	**Definition**
μC	Microcontroller unit. Refers to the development board, which is the heart of the implementation
SC	Supervisory Controller
FC	Feedback Controller
MBC	Model Based Controller
PID	Proportional Integral Derivative (controller)
IMU	Inertial Measurement Unit. A unit that measures accelerations.
GPS	Global Positioning System. Units receiving the signals of the system of satellites developed for geolocation.
RPM	Rounds Per Minute. Units that indicate the rotational speed.

RC	Radio Controlled. Used to show that a system is radio-controlled.
Rcvr	Receiver.
SPI	Serial Peripheral Interface. A type of serial interface used to exchange data in electronics.
RS232	Recommended Standard 232. One of the most widespread protocols for serial communication on digital systems.
NMEA	National Marine Electronics Association. We use this abbreviation to imply the format of data exchanged with a GPS unit.
PWM	Pulse Width Modulation. An encoding method that uses the time width of a pulse to represent a value.

Notations	
Notation	**Definition**
bold italics	Means software or HDL code command. For example ***plot(X,Y)*** command of Matlab.

1.5 Normalization Factors

Normalization Factor	Value	Units
Thrust Vector	255	Unitless
Steer Vector	Steer vector has middle point (no steering) at 121 units and is normalized by 128. Range: [-0.9453125, 1.046875] but we keep it up to 1, so the input must be up to 249.	Unitless
SpeedX	30	Km/h
Yaw Rate	50	o/sec
Acceleration X	3.4	m/sec^2
Yaw Acceleration	130	o/sec^2

1.6 Limit Values

Variable	Low Limit	High Limit	Units
Thrust Vector	0	255	Unitless
Steer Vector	0	249	Unitless
SpeedX	0	30	Km/h
Yaw Rate	-50	50	o/sec
Acceleration X	-3.4	3.4	m/sec^2
Yaw Acceleration	-130	130	o/sec^2

1.7 Normalized Limits During Neural Networks Training

Variable	Low Limit	High Limit	Units
Thrust Vector	0	1	Unitless
Steer Vector	-1	1	Unitless
SpeedX	0	1	Km/h
Yaw Rate	-1	1	o/sec
Acceleration X	-1	1	m/sec^2
Yaw Acceleration	-1	1	o/sec^2

2 Surface Vessel

The surface vessel used for the thesis is shown in Figure 1:

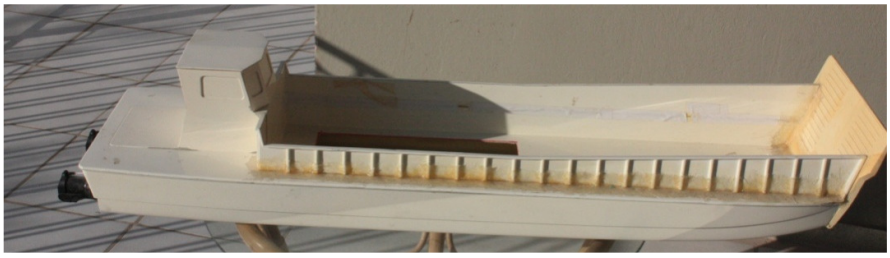


Figure 1: Surface vessel used for the thesis.

It is a ferry boat style, flat hull vessel, equipped with two waterjets for propulsion [6]. This type of propulsion allows turning even from zero surge velocity.

The part of waterjets that extends beyond vessel's hull is shown in Figure 2. More specifically, Figure 2 shows the waterjet nozzles. These nozzles attach to a ball joint which allows them to turn. Thus, by changing the direction of water blast, forces that turn the vessel are created.

Waterjets are powered by two electrical motors [7] (separate for each one). The parameters controlled for the waterjets are:

- RPM of the internal propeller.
- Angle of water blast.
- Forward or backward movement.

All the above parameters are controlled simultaneously for both waterjets. The internal structure supporting these movements is shown in Figure 3.

Figure 2: Vessel waterjets.

Figure 3: Waterjet servo-controls and propulsion motors.

Whilst Figure 3 gives a clear view of the propulsion and turning mechanism of the vessel, Figure 4, adds some marks to aid in the explanation of the parts.

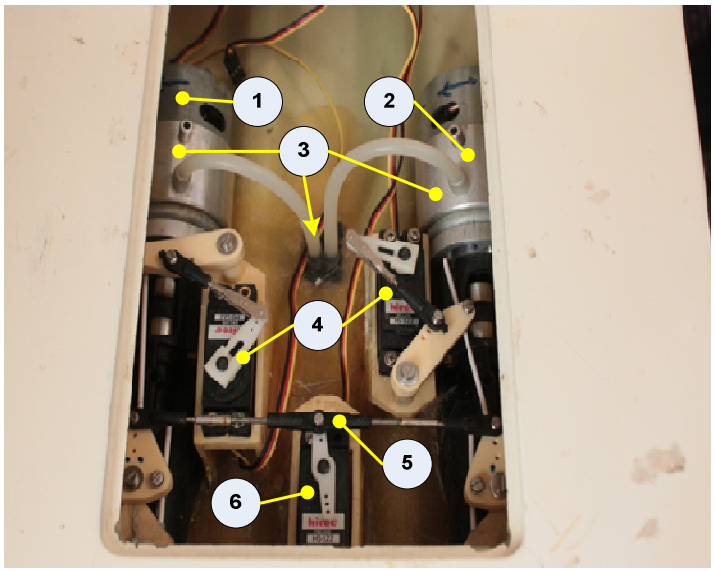


Figure 4: Waterjet controls explained.

Referring to Figure 4, the turning and propulsion system comprises of the following parts:

1. Electric motors. These motors provide the thrust for waterjet propellers.
2. External cooling ring that encapsulates the motors. As the vessel surge velocity increases, an amount of water is forced into the silicone tubes shown by number (3). The water circulates at the outer ring cooling the motors and is rejected from another silicone tube not yet attached at the time this photo was taken.
3. Silicone tubes and input of water for motor cooling.
4. Servos controlling the backward fin that changes the direction of propulsion (forward – backward movement).
5. Shaft that joins the two waterjets for simultaneous turning (left right).
6. The servo motor that is used for turning the waterjets.

2.1 Quick Introduction to Vessel Electronic Equipment

To support all the functionality needed to provide data for modeling and allow for autonomous control, the vessel is equipped with the electronic devices listed in Table 2:

No	Electronic Device	Notes
1	Development board with ARM microcontroller	designed and developed specifically for this vessel. Separate section will be devoted to the hardware capabilities and structure of this board, but for the needs of this and the next sections, it is sufficient to mention that it provides all the interfaces, RAMs and Flash memories that collect and store data

		while ARM processor can on-the-fly process them as required.
2	Remote control receiver	Receiver of the remote radio control unit. Translates the radio signal into PWM outputs that control servo motors.
3	Inertia measurement unit	Denoted as IMU. It belongs to the 6-DoF (degrees of freedom) sensors. Has helped in recording surge acceleration and yaw rate as needed by our application.
4	Electronic magnetometer	3D sensor of Earth's magnetic field.
5	Zigbee remote connection	Radio transmitted serial port. Using this device, we connect development board to PC. A CLI has been developed, in order to send commands and perform certain operations or receive data.
6	GPS unit	Positioning detection unit.
7	Two RPM sensors for the motors	Optical sensors that detected motor rotation.
8	Smart camera	Unit that detected light targets. Initially was aimed to create a positioning command, but abandoned as an idea in process.

Table 2: Vessel electronic equipment.

By using the above equipment, we collected measurements to mathematically model vessel's behavior.

As a reminder, this section is only a brief introduction to vessel's electronics. More details will be given at a section devoted to hardware.

3 Control System Components and Structure

Figure 5 gives a rough sketch of the control system.

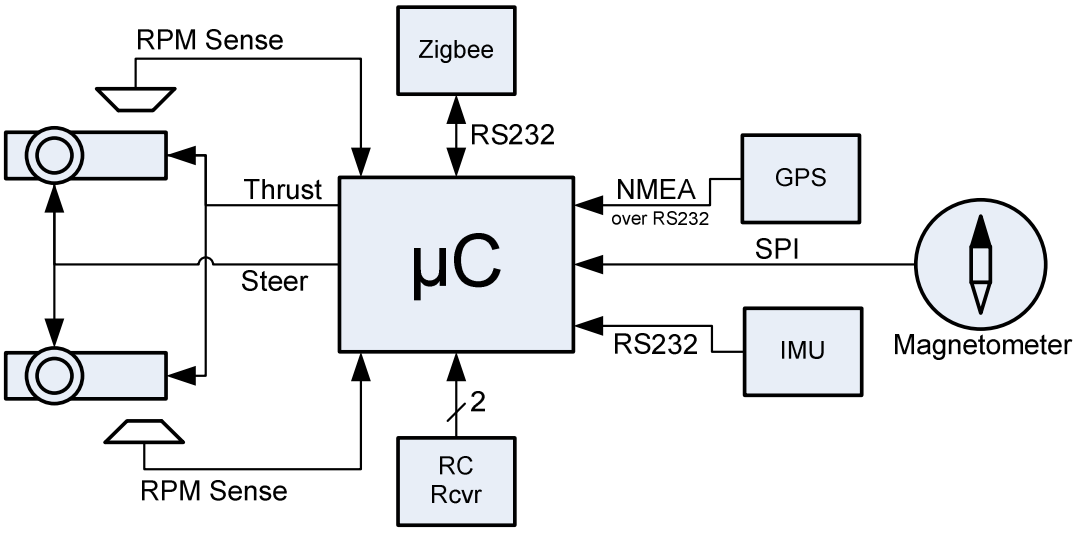


Figure 5: Control system structure.

The development board marked as µC receives data from RPM sensors, GPS, IMU, magnetometer and creates thrust and steer decisions for the waterjets.

For the process of system identification, the remote-control unit was used in order to create the thrust and steer signals and the µC was recording its commands, along with the sensors output.

4 System Identification

To create a model for controlling the vessel, its responses were measured and then methods from the neural network theory were applied [8]. The following sections describe the process.

4.1 Measurements

All field measurements were performed with the hardware recording all parameters to its SRAMS. After the completion of each test, the recorded data were downloaded from serial port using commands from the CLI.

One note, that applies to all measurements, is that the y-axis of thrust and steering measurements is reported as unitless. The reason for this is that it is just a record of the byte representing the radio control receiver output PWM-encoded value. This value has not been related to a physical quantity, and it was also not necessary to do so, since the neural networks that were used to model the vessel translate these inputs (along with translational and angular velocities) to acceleration predictions. Thus, these values remained unitless.

4.1.1 Translational Acceleration Trial No1

The file associated with this measurement is “Verde_2_4_ForMatlab.csv”. The initial planning for this test was to start from a complete stop and go straight initially with half and then with full thrust. Since the available space was rather confined, we had to make a U-turn at the boundaries of the artificial lake we used for the tests. Also, we were not able to use full thrust for sufficiently long time to reach a steady state.

Figures “Figure 6” to “Figure 11” that follow, display the results we recorded.

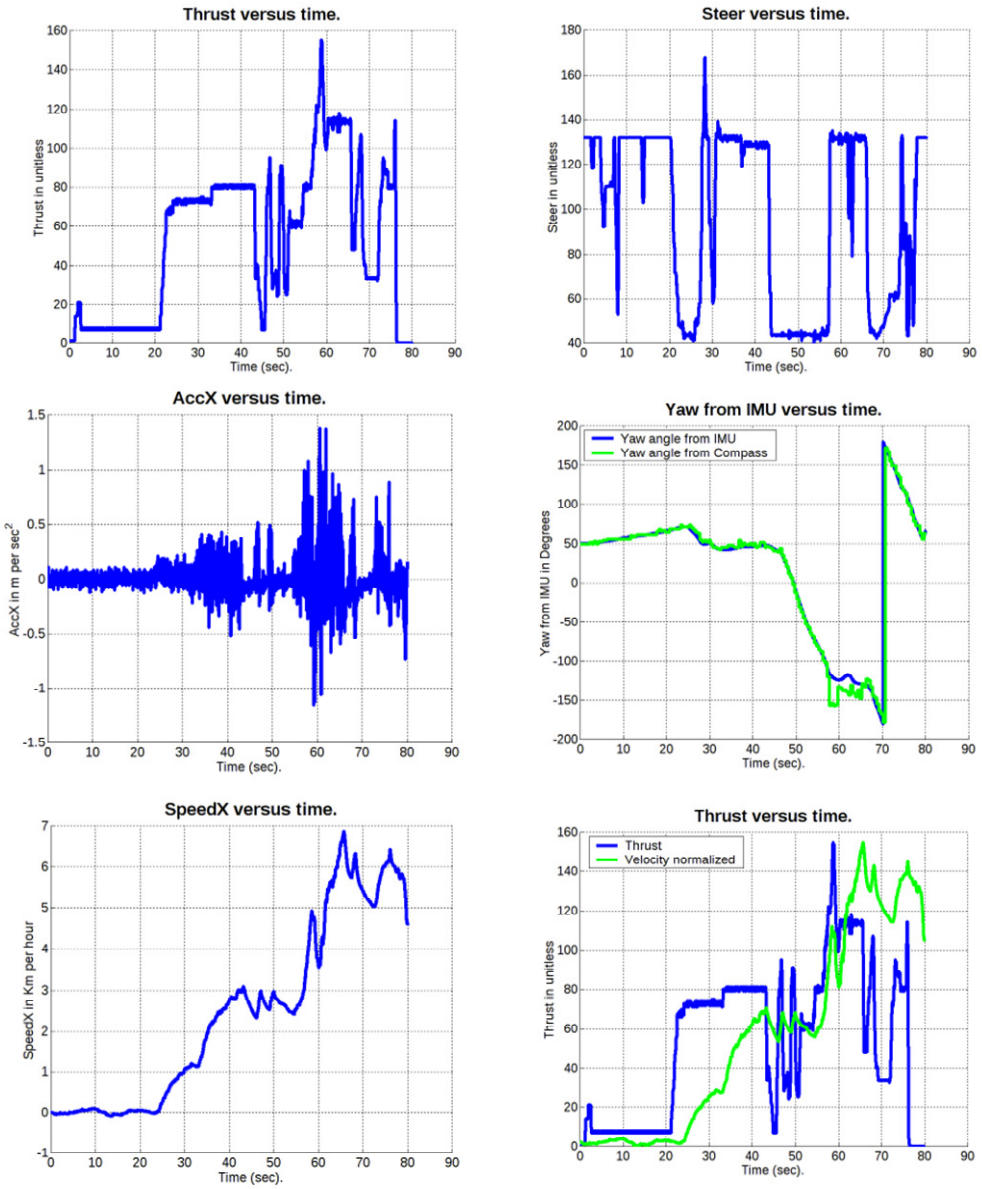


Figure 6: Recorded data for translational acceleration trial No1

One of the things we are.. “proud” of in our measurements, is the almost identical curves that result from digital compass and the integration of IMU yaw rate sensor. This is apparent at the “Yaw from IMU versus time” section of Figure 6.

4.1.2 Translational Acceleration Trial No2

The file associated with this measurement is “Verde_2_6_ForMatlab.csv”. In this measurement set we have recorded vessel response when using half thrust and full trust (after U-turning at the boundaries of the lake).

The results are displayed in Figure 7.

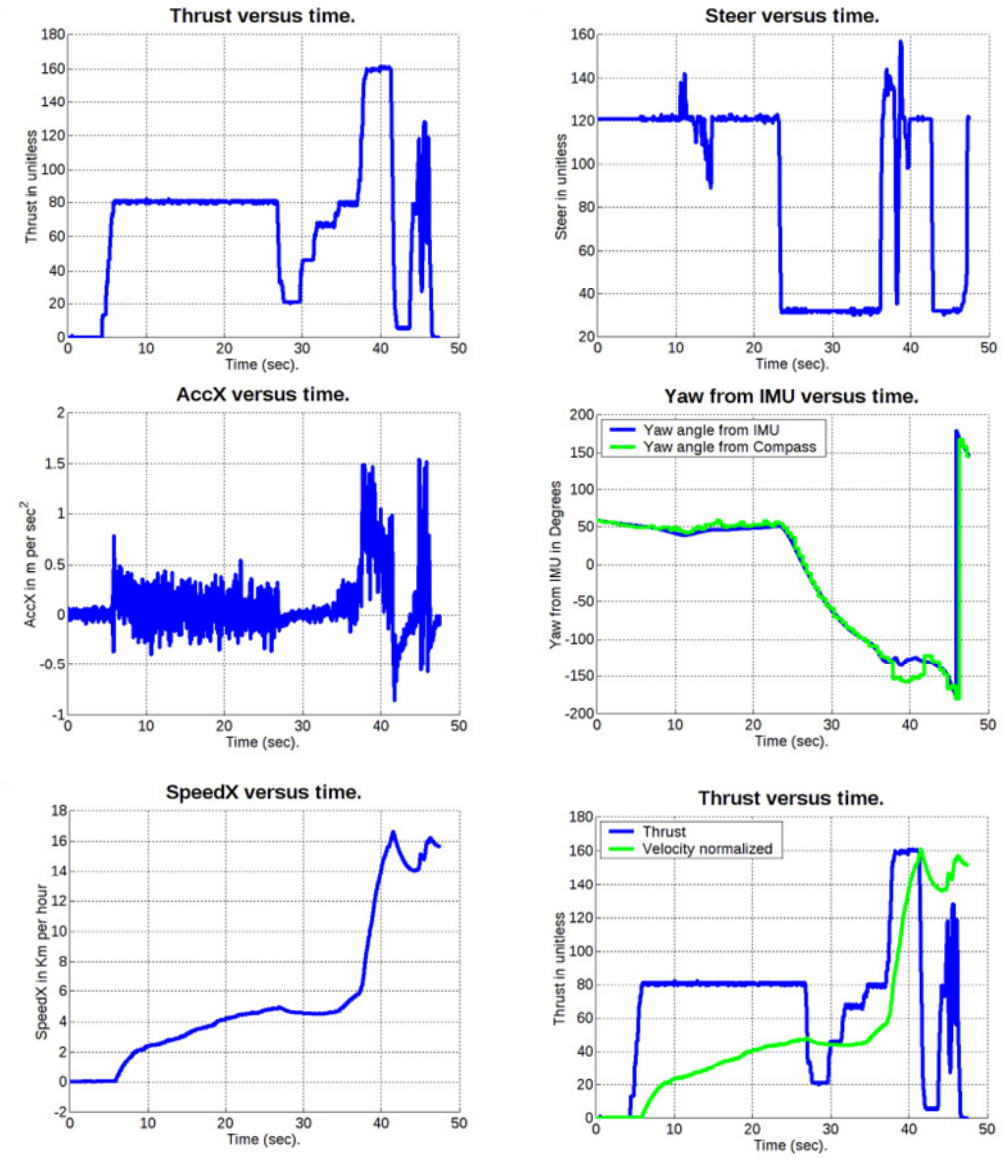


Figure 7: Recorded data for translational acceleration trial No2

4.1.3 Maneuvering Ability Test

The file associated with this measurement is "Verde_2_7_ForMatlab.csv". In this measurement set we recorded Z-type maneuvers.

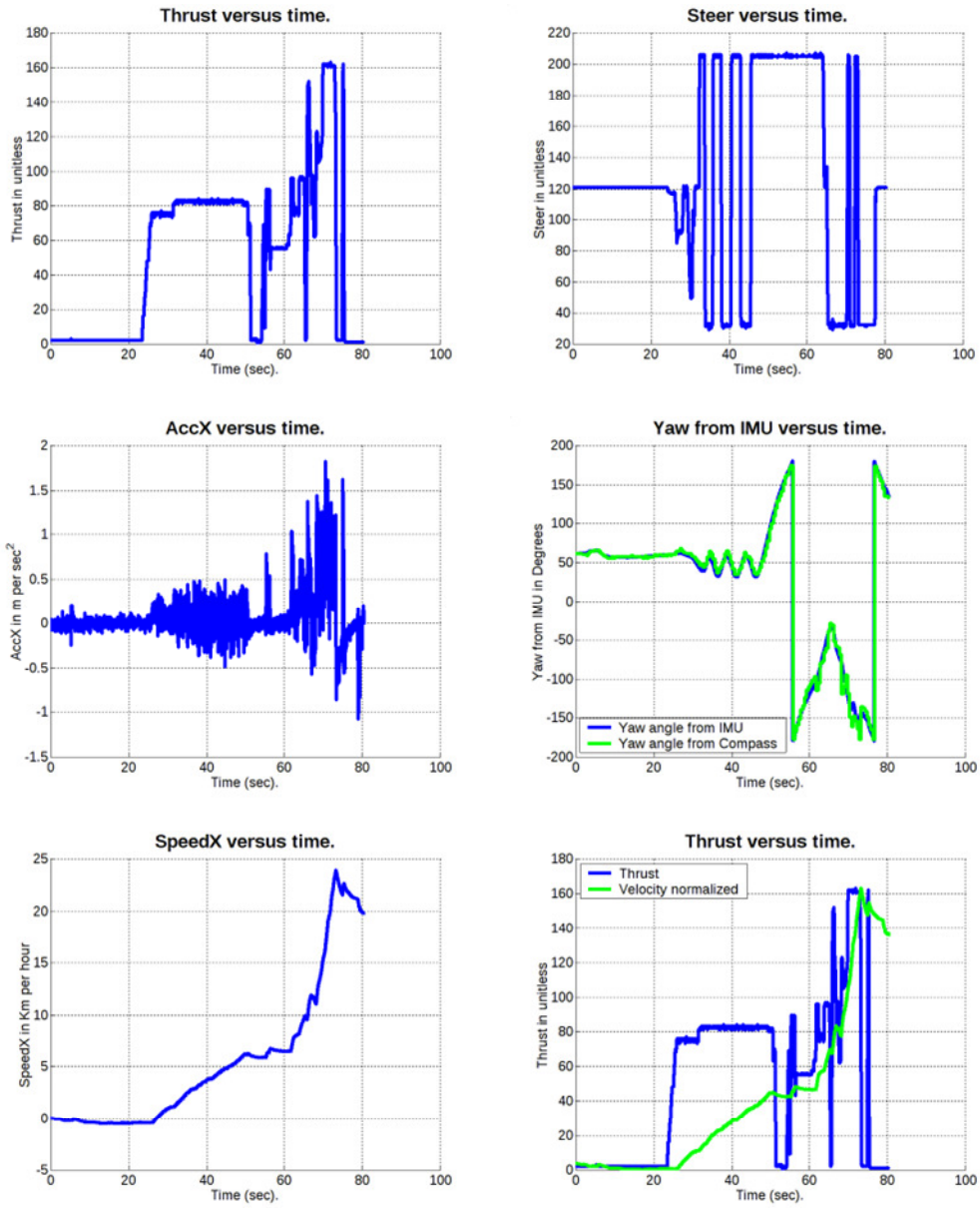


Figure 8: Recorded data for Z-type maneuvers

4.1.4 Clockwise Trials

The file associated with this measurement is "Verde_2_8_ForMatlab.csv". In this measurement set we recorded clockwise turn.

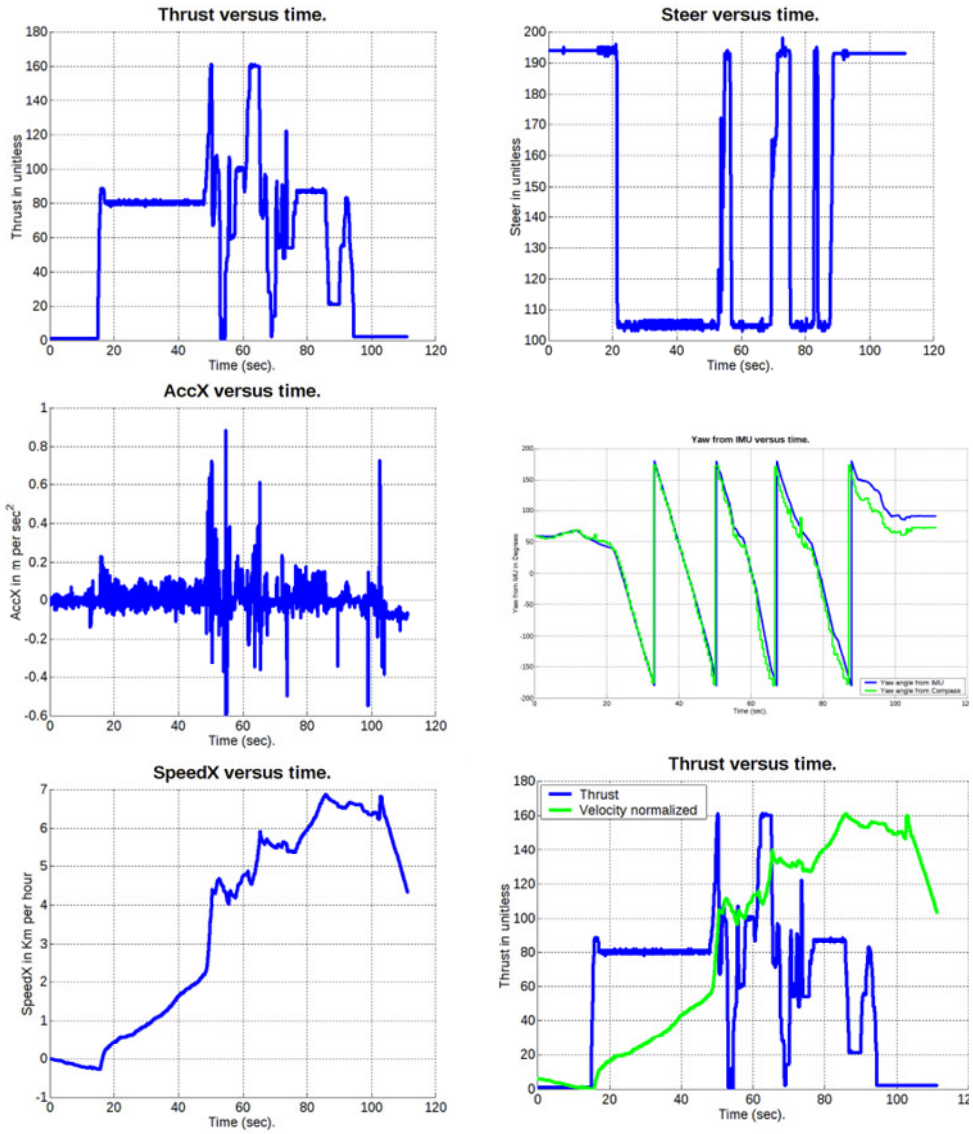


Figure 9: Recorded data for clockwise turn

4.1.5 8-Type Maneuvers

The file associated with this measurement is "Verde_2_10_ForMatlab.csv". In this measurement set we recorded 8-type maneuvers.

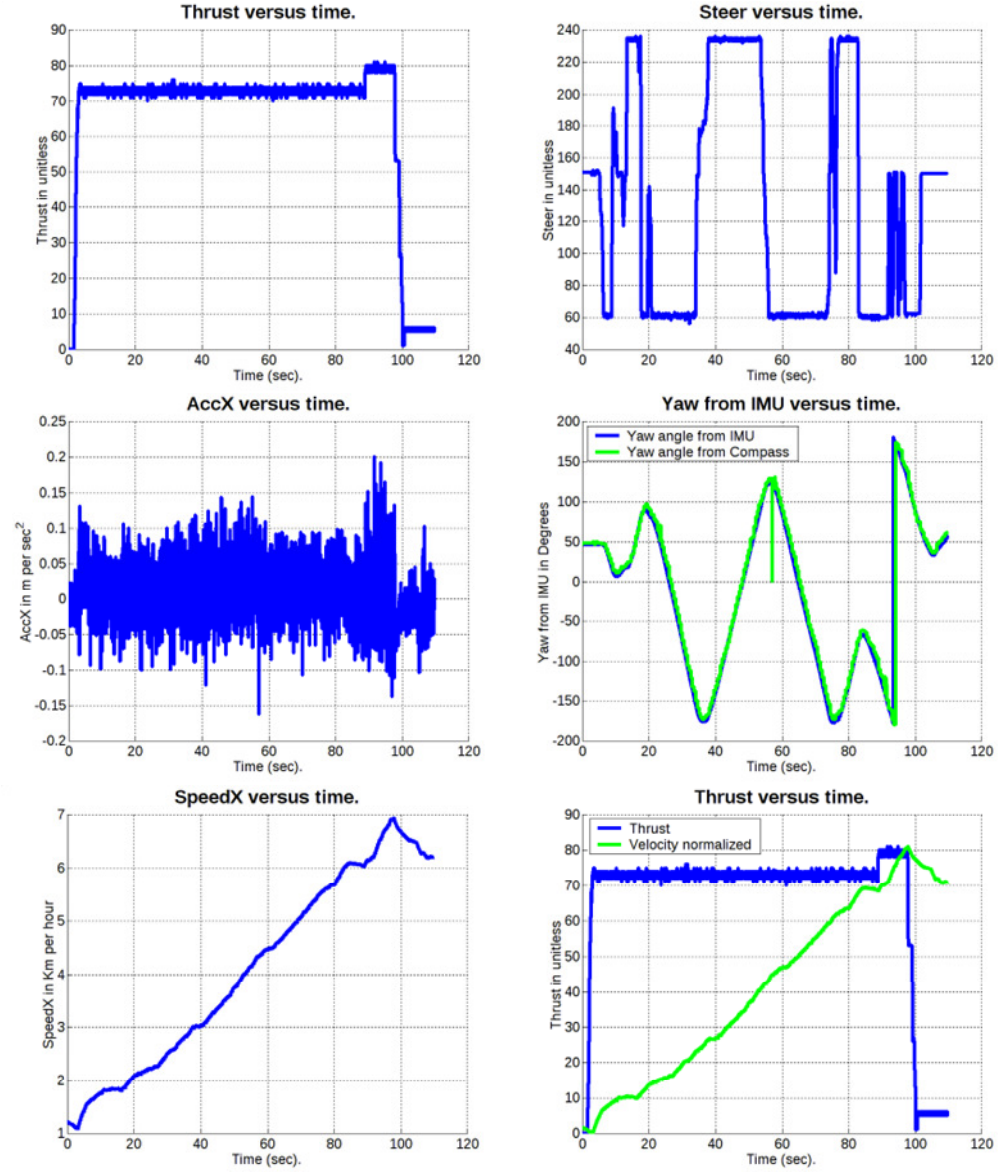


Figure 10: Data recorded during 8-type maneuvering

4.1.6 Second Measurement During 8-Type Maneuvers

The file associated with this measurement is "Verde_2_11_ForMatlab.csv". In this measurement set we recorded 8-type maneuvers once more.

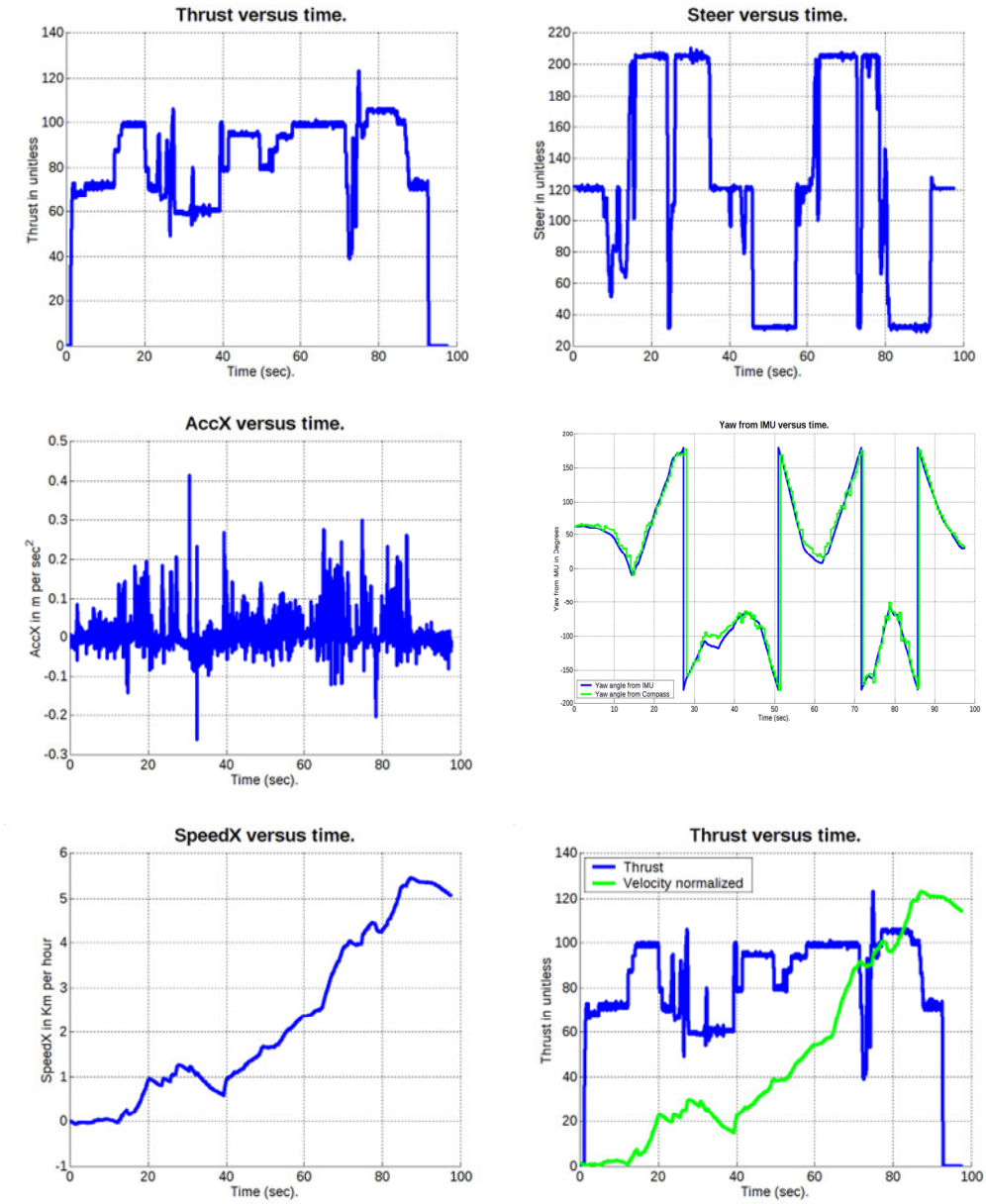


Figure 11: Data recorded during 8-type maneuvering (trial 2)

4.2 State Equations for Craft Dynamics

Based on the measurement data series, we concluded that the craft's maneuvering dynamics can be sufficiently described by a 2nd-order lumped dynamical model with two state variables: forward velocity (speed of advance) and yaw rate (angular velocity) [9]. Using the same data series, two types of state-space models are derived to describe the vehicle: (a) a continuous-time one consisting of two 1st-order ordinary differential equations one for forward acceleration and one for yaw acceleration; (b) a discrete-time NARMA (Nonlinear Autoregressive Moving Average) one consisting of two difference equations, one for each state variable. The implementation of model (a) in Matlab/Simulink© is shown in Figure 7 while that of model (b) in Figure 8. The results in terms of forward velocity and yaw rate obtained by some initial runs of the continuous-time state-space model are shown in Figure 8 while those of the discrete-time one in Figure 9. Note that in the same plot they are validated against the experimental data series obtained by measurement. The equations defining both models are given in the next sections.

4.2.1 State Equations Model in Continuous Time

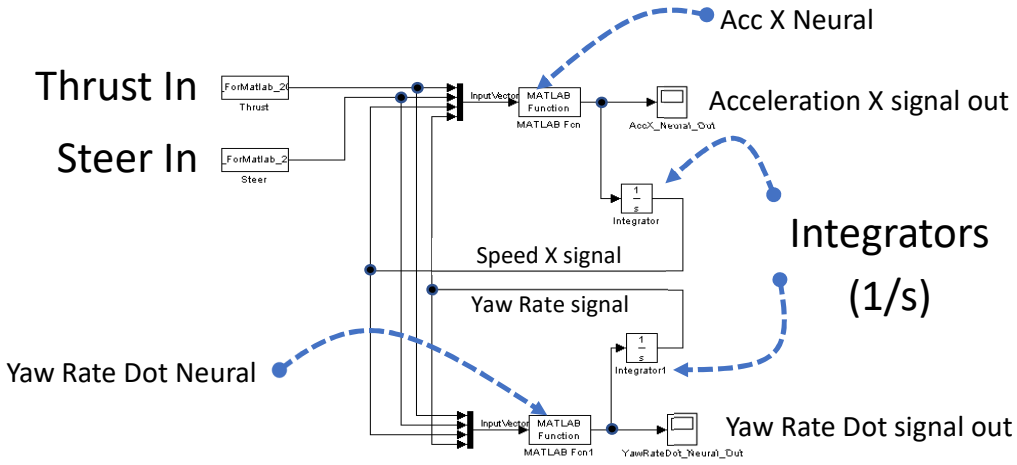


Figure 12: Continuous-time state equations for craft dynamics in Matlab/Simulink©.

In this setup, the model obtains the standard form of first-order nonlinear state equations as shown in (4.1):

$$\dot{\mathbf{x}}(t)=\frac{d\mathbf{x}}{dt}=\mathbf{f}_c\left(\mathbf{x}(t);\boldsymbol{\xi}_{\#}(t)\right),\dot{\mathbf{x}}(t)=\left[a_t(t)\quad a_a(t)\right]^{\mathrm{T}} \tag{4.1}$$

Evidently the state vector **x** includes the translational (rectilinear) velocity and the yaw rate (angular speed) of the vessel, while its time derivative includes translational acceleration a_t and angular acceleration a_a. Vector $\boldsymbol{\xi}_{\#}$ packs the control inputs to the vessel that include the thrust throttle setting y_T and the steering command δ_R or their normalized counterparts as defined in equation (4.2):

$$y_{\%}(t)=\frac{y_T(t)}{256},\delta_{\%}(t)=\frac{\delta_R(t)-121}{128} \tag{4.2}$$

Note that normalized throttle lies between zero and one while normalized steer lies within –1 (full stroke starboard, i.e. clockwise) and +1 (full stroke port, i.e. counterclockwise).

The somewhat abstract function $\mathbf{f}_c$ is defined as a vector function; each one of its components is a neural net yielding a scalar output and accepting four scalar inputs packed in vector $\boldsymbol{\xi}_1$ as follows (in equation (4.3)):

$$\begin{aligned}
&\xi_0=\mathbf{f}_C(\xi_1)=\left[f_{<1>}(\xi_1)\quad f_{<2>}(\xi_1)\right]^T,\\
&\xi_0=\left[\frac{a_t(t)}{3.4\mathrm{ms}^{-2}}\quad\frac{a_a(t)}{130^{\mathrm{O}}\mathrm{s}^{-2}}\right]^T,\\
&\xi_1=\left[y_{\%}(t)\quad\delta_{\%}(t)\quad\frac{u(t)}{30\frac{\mathrm{km}}{\mathrm{h}}}\quad\frac{\omega(t)}{50^{\mathrm{O}}\mathrm{s}^{-1}}\right]^T
\end{aligned} \tag{4.3}$$

Each neural net has three layers (one hidden, one output and one input). The hidden layer of both nets has nine neurons (nodes) since 9 = 2 x 4 +1. Therefore, each net obtains the generic form below. The numeric values of the weight matrices and vectors as well as biases are omitted here for the sake of compactness but will be included in full in a future more extensive work.

$$\begin{aligned}
&f_{<1>}(\xi_1)=\mathbf{w}_{0,<1>}\cdot\boldsymbol{\Phi}_{<1>}\left(\mathbf{W}_{2,<1>}\xi_1+\xi_{2B,<1>}\right)+\xi_{0B,<1>}\\
&f_{<2>}(\xi_1)=\mathbf{w}_{0,<2>}\cdot\boldsymbol{\Phi}_{<2>}\left(\mathbf{W}_{2,<2>}\xi_1+\xi_{2B,<2>}\right)+\xi_{0B,<2>}\\
&\boldsymbol{\Phi}_{<1,2>}(\vec{\chi})=\boldsymbol{\Phi}_{<1,2>}\left(\begin{bmatrix}\chi_1\\\vdots\\\chi_9\end{bmatrix}\right)=\begin{bmatrix}\Phi_{<1,2>}(\chi_1)\\\vdots\\\Phi_{<1,2>}(\chi_9)\end{bmatrix}\\
&\Phi_{<1>}(\chi)=\Phi_{<2>}(\chi)=\tanh(\chi)=\frac{2}{1+e^{-2\chi}}-1
\end{aligned} \tag{4.4}$$

4.2.2 Difference Equations Model (Discrete Time)

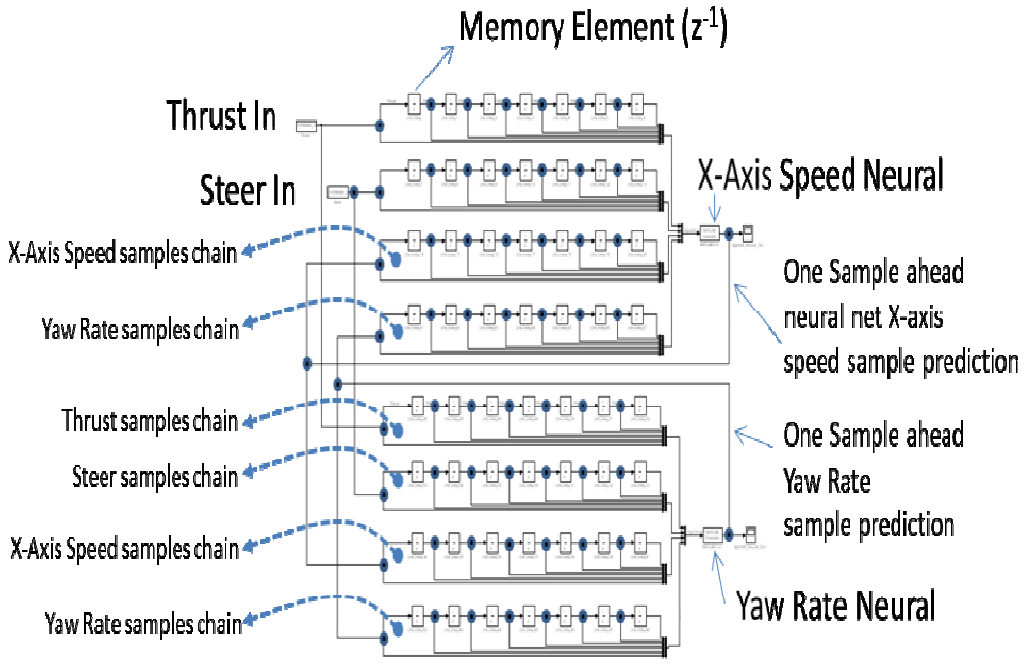


Figure 7: Discrete-time state equations for craft dynamics in Matlab/Simulink©.

In this setup, the model obtains the standard form of nonlinear state equations for the state signal vector sample one time step ahead, as follows (in equation (4.5)):

$$\mathbf{x}(n+1)=\mathbf{f}_d\left(\mathbf{x}\begin{pmatrix}n,\\ n-1,\\ \ldots,\\ n-7\end{pmatrix};\xi_{\#}\begin{pmatrix}n,\\ n-1,\\ \ldots,\\ n-7\end{pmatrix}\right) \tag{4.5}$$

Note that a time window (memory) of seven past samples for the state and control input vectors, which are defined as previously, has been used. The time step Δt is 0.9 s and should not be confused with the sampling period T_S = 30 ms = Δt / 30 used for data series acquisition. All time instants are identified as integer multiples of time step Δt in this model.

The somewhat abstract function $\mathbf{f}_d$ is defined as a vector function; each one of its components is a neural net yielding a scalar output and accepting 32 scalar inputs packed in vector $\boldsymbol{\xi}_1$ as in (4.6):

$$\begin{aligned}&\xi_0=\mathbf{f}_{\mathbf{D}}(\xi_1)=\left[f_{<1>}(\xi_1)\quad f_{<2>}(\xi_1)\right]^{\mathrm{T}},\\ &\xi_0=\left[\frac{u(n+1)}{30\,\frac{\mathrm{km}}{\mathrm{h}}}\quad\frac{\omega(n+1)}{50^{\mathrm{O}}\,\mathrm{s}^{-1}}\right]^{\mathrm{T}},\end{aligned} \tag{4.6}$$

$$\xi_1 = \begin{bmatrix} \xi_y \\ \xi_\delta \\ \xi_u \\ \xi_\omega \end{bmatrix}, \quad \begin{aligned} \xi_y &= \left[y_{\%}(n-7) \quad \cdots \quad y_{\%}(n) \right]^{\mathrm{T}} \\ \xi_\delta &= \left[\delta_{\%}(n-7) \quad \cdots \quad \delta_{\%}(n) \right]^{\mathrm{T}} \\ \xi_u &= \frac{\left[u(n-7) \quad \cdots \quad u(n) \right]^{\mathrm{T}}}{30\,\frac{\mathrm{km}}{\mathrm{h}}} \\ \xi_\omega &= \frac{\left[\omega(n-7) \quad \cdots \quad \omega(n) \right]^{\mathrm{T}}}{50^{\mathrm{O}}\,\mathrm{s}^{-1}} \end{aligned} \tag{4.7}$$

Like previously, each neural net has three layers. However, both nets are now remarkably larger since they accept more inputs. The hidden layer of both nets has now 65 = 2 x 32 +1 neurons. Therefore, each net obtains the generic form of (4.8). The numeric values of the weight matrices and vectors as well as biases will be included in full in a future more extensive work.

$$\begin{aligned} f_{<1>}(\xi_1) &= \mathbf{w}_{0,<1>} \cdot \mathbf{\Phi}_{<1>}\left(\mathbf{W}_{2,<1>} \xi_1 + \xi_{2\mathrm{B},<1>} \right) + \xi_{0\mathrm{B},<1>} \\ f_{<2>}(\xi_1) &= \mathbf{w}_{0,<2>} \cdot \mathbf{\Phi}_{<2>}\left(\mathbf{W}_{2,<2>} \xi_1 + \xi_{2\mathrm{B},<2>} \right) + \xi_{0\mathrm{B},<2>} \\ \mathbf{\Phi}_{<1,2>}(\vec{\chi}) &= \mathbf{\Phi}_{<1,2>}\left(\begin{bmatrix} \chi_1 \\ \vdots \\ \chi_{65} \end{bmatrix} \right) = \begin{bmatrix} \Phi_{<1,2>}(\chi_1) \\ \vdots \\ \Phi_{<1,2>}(\chi_{65}) \end{bmatrix} \\ \Phi_{<1>}(\chi) &= \left(1 + \exp(-\chi)\right)^{-1}, \Phi_{<2>}(\chi) = \tanh(\chi) \end{aligned} \tag{4.8}$$

4.3 SIGNAL ANALYSIS

For both the continuous and the discrete model, signal analysis and processing was required in order to be able to use the experimental data series obtained by measurement for training of the neural nets appearing in the model core [10], [11].
For the continuous-time model it was required to generate a smooth yaw acceleration signal since the instrumentation did not provide such output directly. The signal pre-processing used to obtain that is shown in Figure 8.

Figure 13: Signal processing to generate smooth yaw acceleration signal for the neural nets in the continuous-time state model.

For the discrete-time model we needed to keep the number of inputs to the core neural nets to a minimum. In this end, decimation of the recoded data series to the time scales occurring in the system was required [10], [12]–[15]. Note that, following standard engineering practice, the sampling rate of the measurements was rather high to ensure fidelity. Data series decimation was performed by propagating the recorded data signals through the filtering architecture shown in Figure 9.

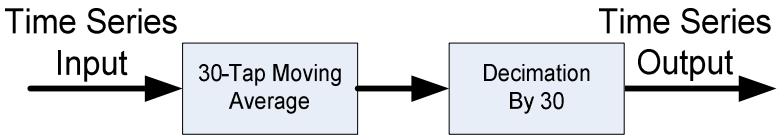


Figure 14: Decimation filtering applied to minimize number of inputs of the neural nets in the discrete-time state model.

Simulation Results and Model Assessment

Based on the field trials for the landing craft scale model in Figure 1 two models were developed with fundamentally different features. The continuous time (CT) model which is the most common in system identification and control engineering literature could not be readily constructed due to a major technical weakness of the low-cost sensor instrumentation employed. Specifically, the lack of data for yaw acceleration directly obtained through measurement. Indeed, the IMU employed was providing with translational acceleration data series for each run but not with yaw angular acceleration data.

To overcome this problem a two-fold approach was employed. An elaborate filtering scheme along with detailed preprocessing of the yaw rate angular speed much more complicated than what applied for the discrete time (DT model. Also, a DT model was developed to cross-check the results of the CT model.

As can be seen in Figure 15, the results obtained from the DT model for both forward speed and yaw rate of the vessel were very close to the measurements. Even though there is no integration, numerical or otherwise, involved the major parameters that needed to be identified for the DT model was the memory (time window size) for each scalar state variable and control input as well as the most suitable decimation factor for the high-rate sampled measurement series. As was demonstrated in practice, a suitable decimation factor choice led to significantly reduced care for the memory size. This combined with the fact that neural nets can perform well even with partial or redundant data series allowed to use the same memory size for all states and control inputs with very good results as observed in Figure 15.

In contrast, even though much more effort was invested in the development of the CT model, the outcome was significantly poorer as seen in Figure 16 for both forward speed and especially yaw rate since no measured data for yaw acceleration were available from the IMU used. To make comparisons easier the same run was tested for both the CT and DT model.

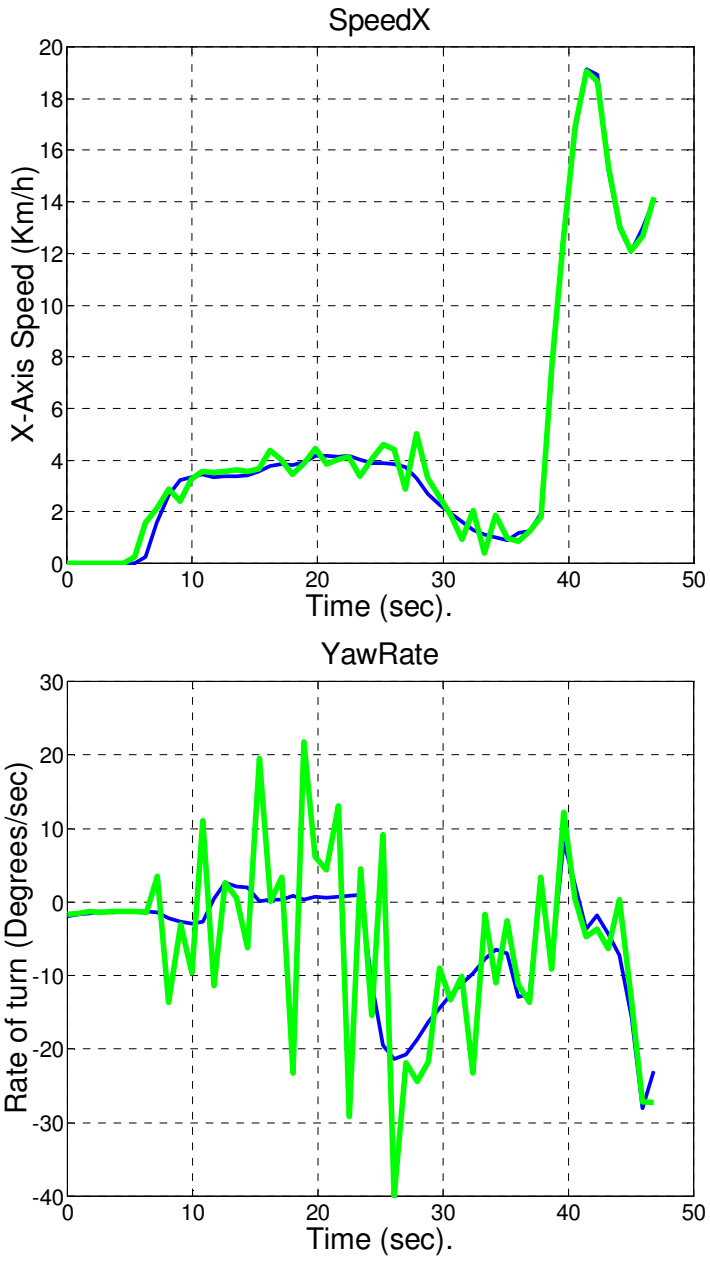


Figure 15: Discrete-time state-space model output (green) vs. measurements (blue): forward velocity (up) & yaw rate (down).

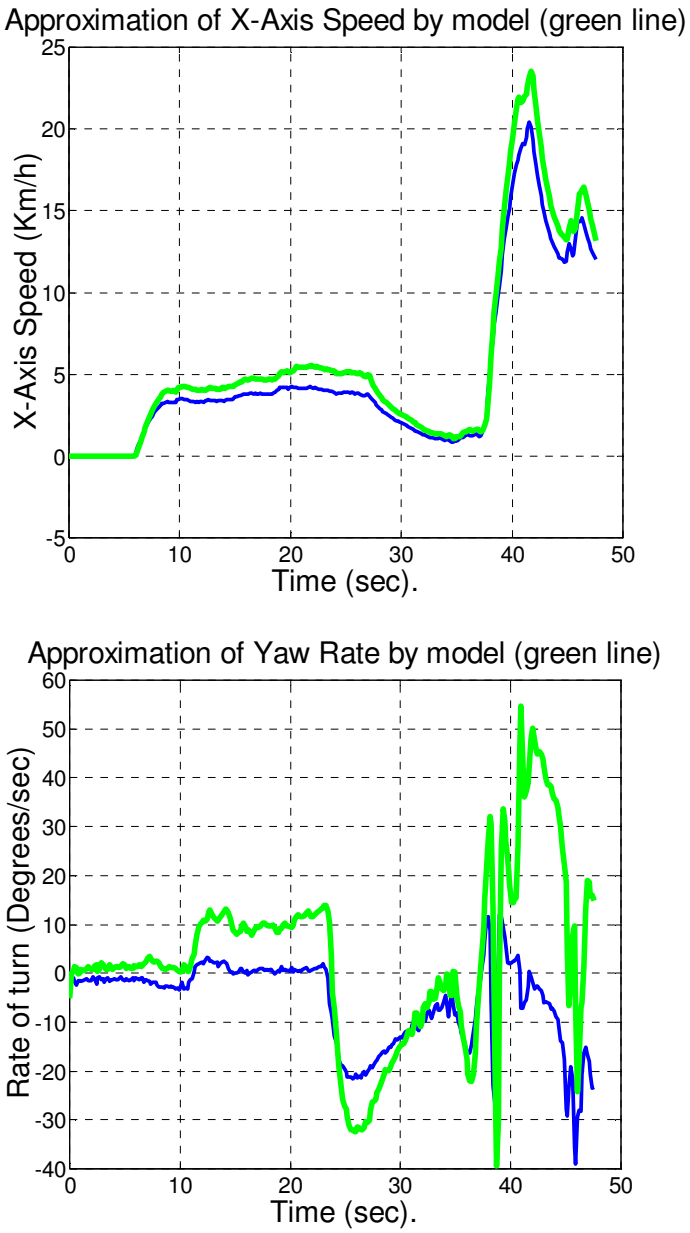


Figure 16: Continuous-time state-space model output (green) vs. measurements (blue): forward velocity (left) & yaw rate (right).

One way to improve the fidelity of the CT model is to employ the DT model to generate data series which can be used in turn to train the neural nets of the CT model especially the one for yaw acceleration. In this scenario, an appropriate data fusion scheme is needed to compromise the smoothening filtering employed for the speed signals and the corresponding outputs of the DT model for which acceleration is not used or needed to develop. Such a data fusion scheme would generate speed signals closer to the measurements but smooth enough to be differentiated with respect to time yielding improved accelerations.

Evidently, another way is to use a different IMU that directly outputs yaw acceleration and not just yaw rate. In this case the DT model can still be of value because it can be employed in a signal processing scheme to remove the noise of from the acceleration measurements; indeed, as can be seen even from the forward speed and acceleration signals there is significant disagreement between the two which is attributed to sensor noise and distortion.

In any case, either the CT or the DT model can be employed in the framework of Robust Probabilistic Control outlined earlier. Since this framework is one of Stochastic Control, therefore intrinsically designed to work with uncertain signals, and works with state distribution functions rather than exact trajectories any modeling errors can be incorporated in a probabilistic model.

4.4 Neural Networks Modelling

For the vessel identification two models were used, each implemented using two neural networks:

Continuous time model using

- AccX network
- Wdot network

Discrete time model using

- SpeedX
- W

The paragraphs that follow intend to give the mathematical formulas that are used for each network output.

4.4.1 Perceptron Neural Network

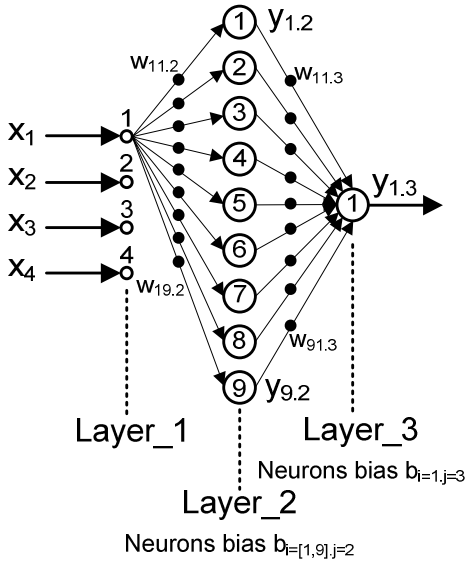


Figure 17: General sketch of a perceptron.

For the perceptron [16], [17] shown in Figure 17, which is exactly the type used in "continuous" time modeling, we have three layers. The first layer is just the inputs, so the nodes are not neurons, just buffers. Inputs are marked as xi where i=[1,4]. Each input times a weight (wij.k), is connected to all neurons of layer 2 (internal layer). For the weights i=[1,4] and means the source input, j=[1,9] and means the neuron accepting the weighed input and finally k=[1,3] and means the destination layer.

Each neuron has an output named yi.j where i is the index of the neuron at layer j. For each neuron there is also a bias bi.j where i is the number of neuron at layer j. Thus the output for the network in Figure 17is:

$$y_{1.3} = \left(\sum_{i=1}^{9} y_{i.2} w_{i1.3} \right) + b_{1.3}$$
$$y_{i.2} = \text{Tansig}\left[\left(\sum_{k=1}^{4} x_k w_{ki.2} \right) + b_{i.2} \right] \quad (4.9)$$
which gives:
$$y_{1.3} = \left[\sum_{i=1}^{9} \left[\text{Tansig}\left[\left(\sum_{k=1}^{4} x_k w_{ki.2} \right) + b_{i.2} \right] w_{i1.3} \right] \right] + b_{1.3}$$

A very important note here is the following: if you examine the units of the parameters at the input of the Neural Network, you will find that (as can be seen from "Table 1: Singals, units and notations.") yT and δR have no units, but u is in Km/h and ω is in rad/sec. How do all these signals combine in the Neural Network equation? One answer is that the actual input to the net are signals y%, δ%, u% and ω%, which are the normalized versions of the signals and as such they are unitless (they have been divided by a constant with the same units). Another answer is that even if the signals were used as is, the network functions as a kind of LUT and as such the units do not have any meaning to it, as long they are the same with the ones presented to the network at the time of training. Though, my personal preference is to use the unitless normalized version.

4.4.2 Mathematical Formulas

Depending on the neuron type, the transfer function differs. The three transfer functions used for the neurons of the implemented networks are:

- Logsig sigmoid function. The formula for logsig is:

$$\text{logsig(n) = 1 / (1 + exp(-n))}$$
$$logsig(n) = \frac{1}{1+e^{-n}} \quad (4.10)$$

- Tansig sigmoid function. The formula for tansig is:

$$\text{tansig(n) = 2/(1+exp(-2*n))-1}$$
$$tansig(n) = \frac{2}{1+e^{-n}} - 1 \quad (4.11)$$

- Purelin function, which is just a linear summation.

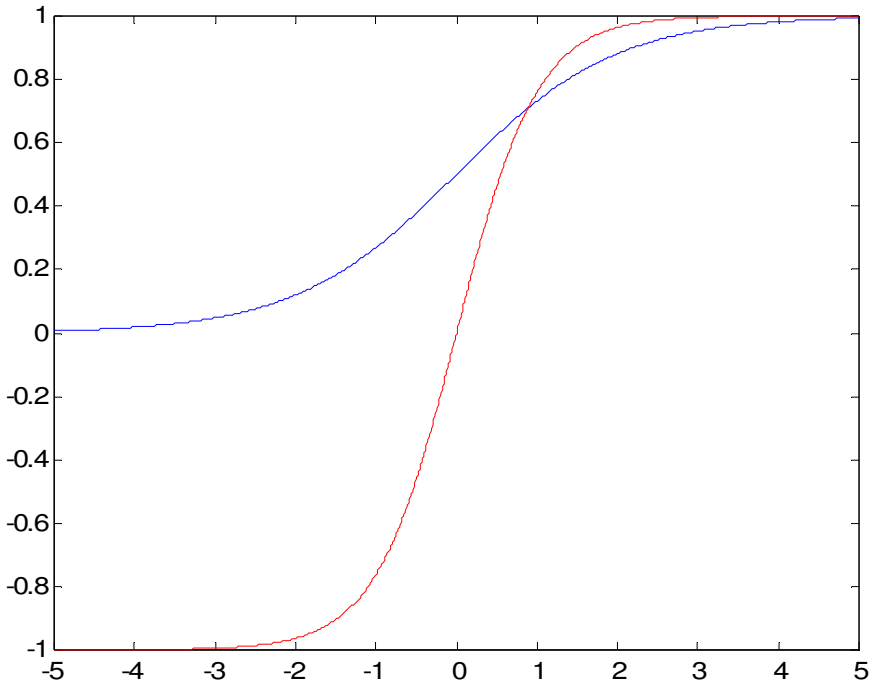


Figure 18: logsig (blue) vs tansig (red) functions.

As can be seen from logsig function has only positive output, while tansig function can output negative values also.

4.4.3 Derivatives of Neuron Functions

4.4.3.1 LOGSIG FUNCTION DERIVATIVE

The derivative of logsig function is as in (4.12):

$$\frac{d}{dn} logsig(n) = \frac{e^{-n}}{(1+e^{-n})^2} \tag{4.12}$$

The function has been verified at Matlab with n = [-5:0.01:5] using differences of subsequent samples divided by the samples step as in (4.13):

$$Deriv(i) = \frac{logsig(n(i+1)) - logsig(n(i))}{n(i+1) - n(i)} \tag{4.13}$$

The results are shown in Figure 19.

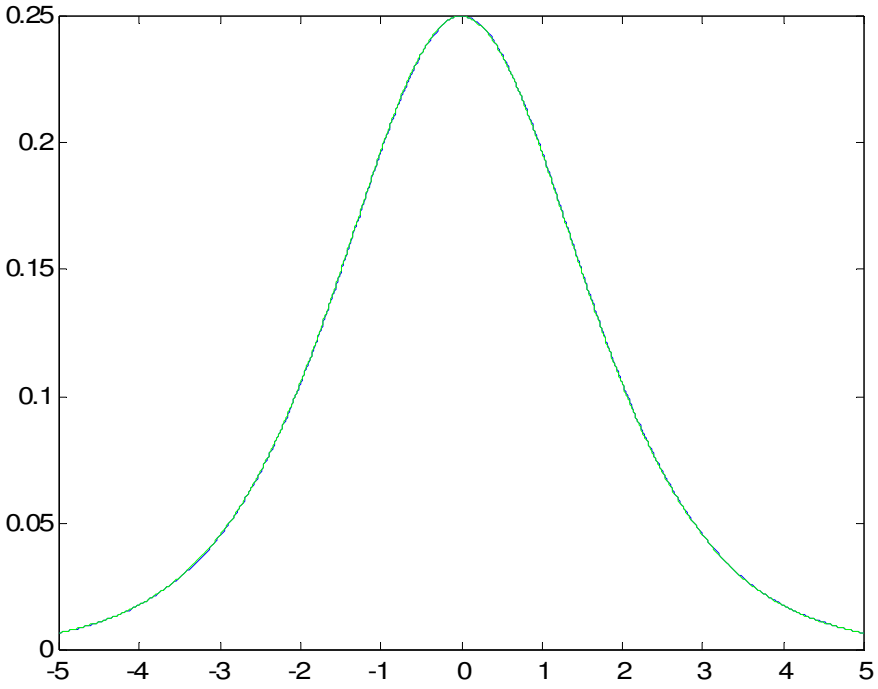


Figure 19: Logsig derivative verification.

The derivative function output is with blue color, while the difference approximation is with green. Maximum deviation between the two is 4.8112e-004.
The code used for testing is in file E:\Work\PHD\MatlabCodes\logsig_der.m.

4.4.3.2 TANSIG FUNCTION DERIVATIVE

The derivative [18]–[20] of tansig function is as in (4.14):

$$\frac{d}{dn}tansig(n) = \frac{4e^{-2n}}{(1+e^{-2n})^2} \tag{4.14}$$

The function has been verified at Matlab with n = [-5:0.01:5] using differences of subsequent samples divided by the samples step as in (4.15):

$$Deriv(i) = \frac{tansig(n(i+1)) - tansig(n(i))}{n(i+1) - n(i)} \tag{4.15}$$

The results are shown in Figure 20.

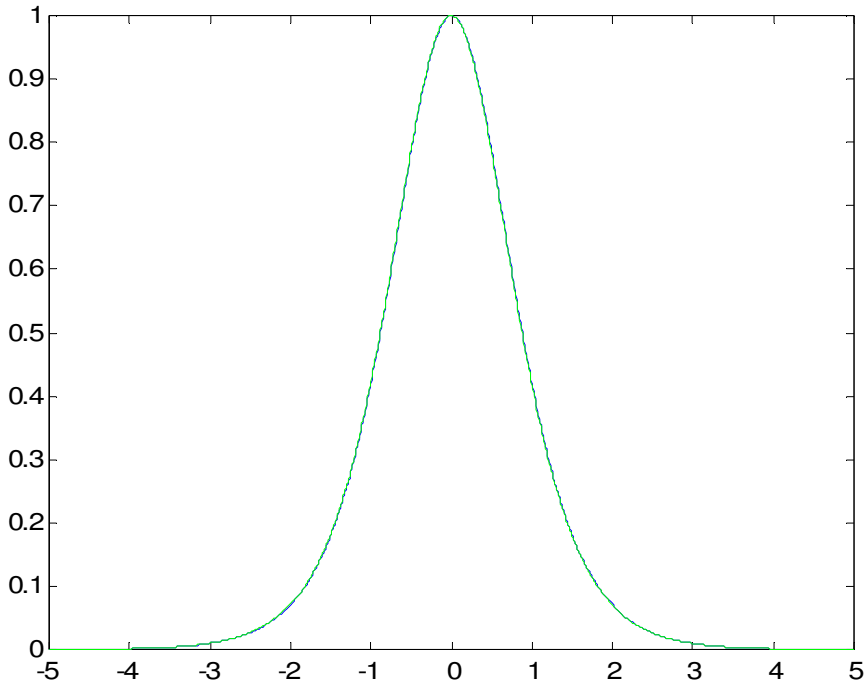


Figure 20: Tansig derivative verification.

The derivative function output is with blue color, while the difference approximation is with green. Maximum deviation between the two is 0.0038.
The code used for testing is in file E:\Work\PHD\MatlabCodes\tansig_der.m.

5 Derivation Of the Full Model and Controllers

5.1 Introduction to Section

After evaluating the method of using equations in the place of the Matlab implementation of Neural Networks, the full model of interconnected Neural Networks must be implemented and evaluated using the data from experiments. This section describes the procedure and internal structure of the model we used.

5.2 Full Vessel Model

As stated in paragraph 4.2.1, to model the vessel two neural networks are used; one that outputs translational acceleration samples, and one that outputs angular acceleration samples. The output of each network in integrated with time with the selected integration algorithm (see section 5.5.2.1), and fed back to both neural networks, to create the next sample as shown in Figure 21 [9].

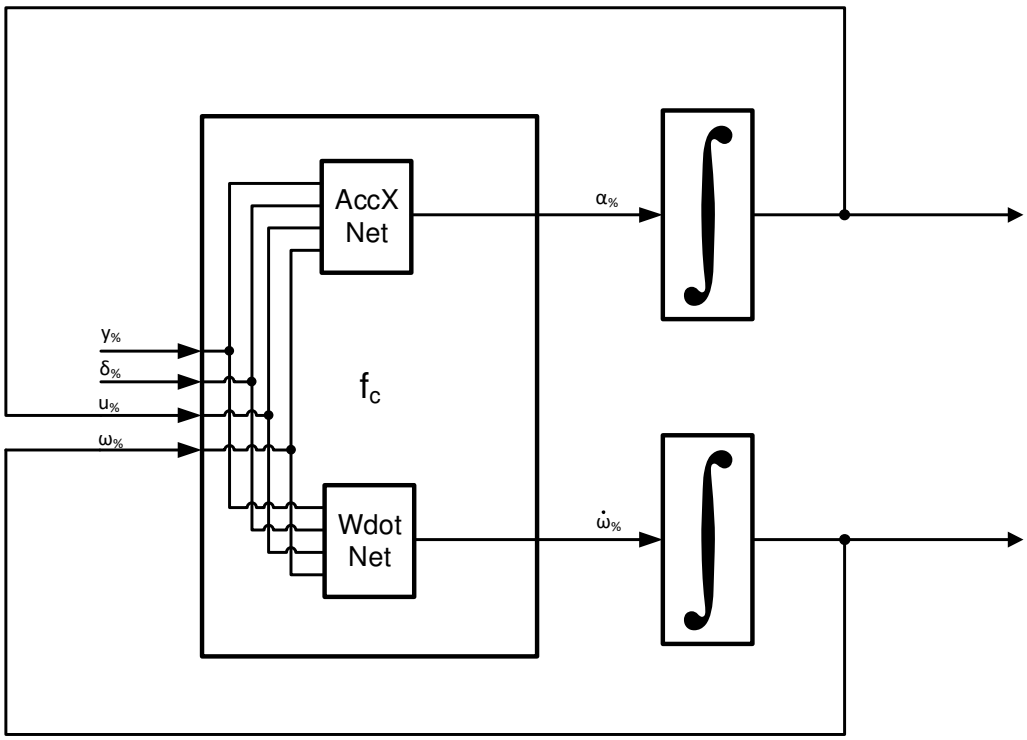


Figure 21: Full Vessel Model Using Neural Networks.

The system of interconnected neural networks along with the integrators is the one used in all test benches for the vessel response. It is also used by the MBC controller as will be shown in paragraph 5.5.1, to create a decision regarding the most suitable control action that should be used.

Although the model is named as “Continuous Time” in section 4.2.1, discrete time is hidden in the integration approximation. This fact enables the use of this setup to our discrete time simulating environments.

It must also be noted that the neural networks have also been trained with discrete time data, that were created every T_s, i.e., every 30 ms.

5.4 Control Scheme

Before starting the description of control scheme it is necessary to present the variables each part of the control scheme receives. The following subsection presents the way positions and velocities are evaluated and the reference axes systems needed for this purpose.

5.4.1 Reference Frames

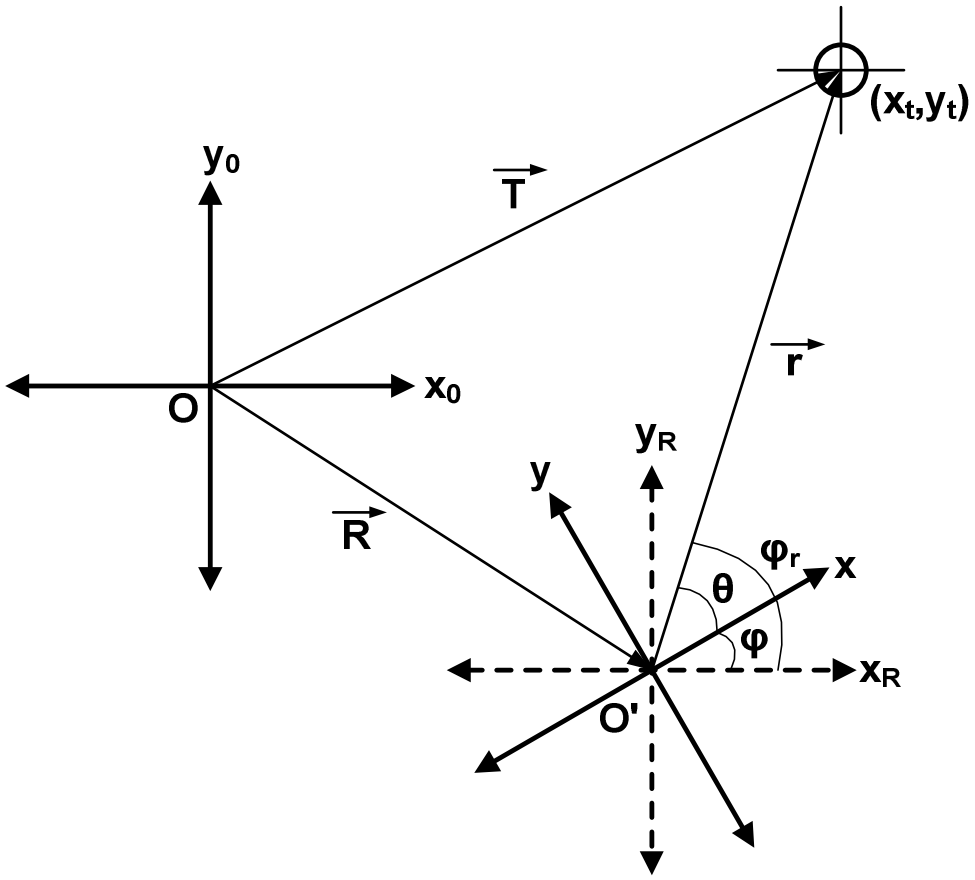


Figure 22: Axes systems.

Referring to Figure 22, three axes systems appear:

- Ox_0y_0
- $O'x_Ry_R$
- $O'xy$

Ox_0y_0 is the absolute reference frame of the two-dimensional space where the vessel and target points exist.

$O'x_Ry_R$ is the frame that describes the position of the vessel's center of mass, relative to Ox_0y_0. Its axes are parallel to Ox_0y_0. Vector $\vec{R}$ connects point O to O' (actually $\vec{R} = \overrightarrow{OO'}$).

O'xy is the body-fixed reference frame [21] which follows the vessel's angle of turn by fixing the x-axis to the middle of vessel's prow.

Table 3 summarizes all terms needed in understanding the reference frames:

Reference Frames Symbols List (repeated from section "1.4 Symbols, Variables, Notations and Abbreviations")	
Symbol	**Definition**
Ox_0y_0	Absolute reference frame of the two-dimensional space where the vessel and target points exist.
$O'x_Ry_R$	Frame that describes the position of the vessel's center of mass, relative to Ox_0y_0
O'xy	Body-fixed reference frame which follows the vessel's angle of turn
$\vec{R}$	Vector marking the (common) position of frames $O'x_Ry_R$ and O'xy relative to Ox_0y_0.
$\vec{r}$	Vector connecting O' to the next target point.
$\vec{T}$	Vector connecting O to the next target point.
(x_t, y_t)	Target point coordinates with Ox_0y_0 as a reference frame.
θ	Angle between vessel's prow and vector $\vec{r}$. Positive counter-clockwise.
φ	Angle between vessel's prow and x_R axis. Positive counter-clockwise.
φ_r	Angle between vector $\vec{r}$ and x_R or x_0 axis. Positive counter-clockwise.

Table 3: Symbols list related to reference frames.

These symbols will be used at the paragraphs that follow, along with the symbols of Table 4:

Section Symbols List (repeated from section "1.4 Symbols, Variables, Notations and Abbreviations")	
Symbol	**Definition**
y_T	Engines throttle from user remote control.
δ_R	Turn from user remote control.
y_{TA}	Auto-generated thrust command.
δ_{RA}	Auto-generated steering command.
u	Translational speed of the vessel in Km/h.
ω	Angular speed of the vessel in °/sec.
α	Translational acceleration of vessel ($=\dot{u}$) in m/(sec)^2.
$\dot{\omega}$	Angular acceleration in °/(sec)^2.
T_s	Sampling rate for all operations of the system. The sampling rate used is 0.03s.

Table 4: Symbols list used for section Control Scheme5.4.

To aid in the understanding of the following sections some abbreviation explanations are added:

Abbreviations List (repeated from section "1.4 Symbols, Variables, Notations and Abbreviations")	
Symbol	**Definition**
SC	Supervisory Controller
FC	Feedback controller
MBC	Model Based Controller
Zoh	Zero-order hold

Table 5: Abbreviations of section Control Scheme.

5.4.2 Full System Overview

To control vessel position in a two-dimensional space, the control scheme shown in Figure 23 was decided:

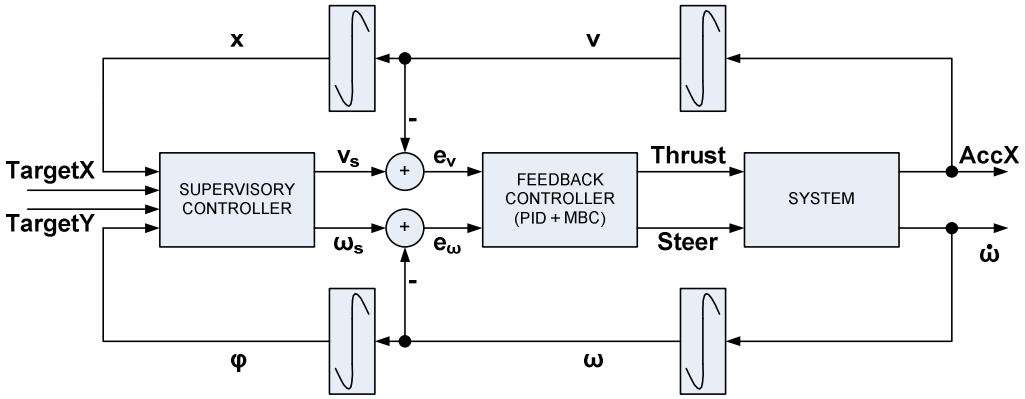


Figure 23: Full system control scheme.

This image is a rough description of the actual control scheme. Critical parts are going to be analysed further in the sections that follow.

As a whole system the Supervisory Controller (SC) takes as input the target position referenced at O'xy axes system, which is a body-fixed reference. Its purpose is to zero the magnitude of vecor $\vec{r}$ which connects the target to the vessel's center of mass. To do so, SC generates request values for vessel's translational velocity and angular velocity.

The SC output is then received by the Feedback Controller (FC), which functions as a thrust and steering generator for the system, in order to force the vessel to achieve the requested velocities.

FC outputs are received by the system, which in turn responds with a translational acceleration output and an angular acceleration output.

5.5 Feedback Controller (FC)

The FC unit comprises of the following parts:

- PIDu: PID controller for regulating the vessel's translational velocity (u).
- PIDω: PID controller for regulating the vessel's translational velocity (u).
- MBC: Model Based Controller which takes as reference the requests from the two PID controllers referenced above and searches for the most appropriate thrust and steer commands for the vessel, in order to produce the requested accelerations from PIDs.

MBC works also as a decoupler for the accelerations request and thus the feedback loops can be separated and be treated as SISO. Any inaccuracies at the system accelerations and at the decoupling the MBC succeeds are added as noises to the output accelerations of the system.

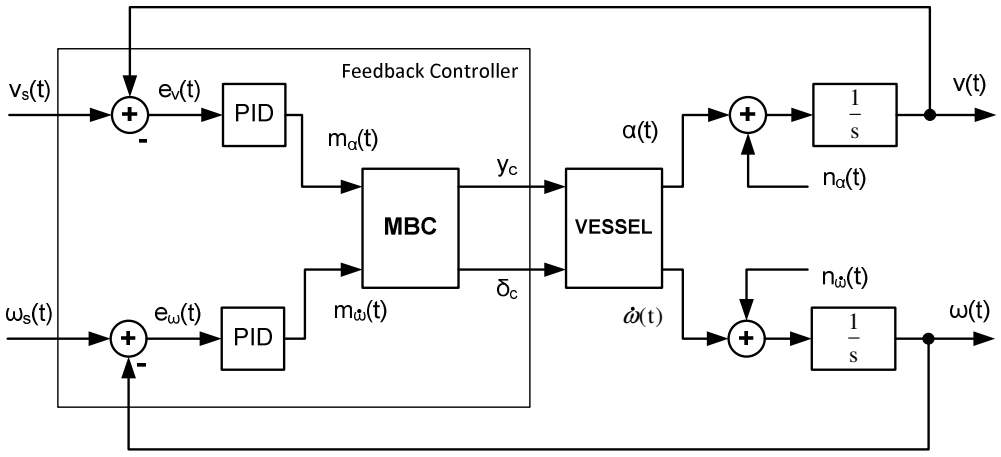


Figure 24: Feedback Controller and its operation in the system.

5.5.1 Model Based Controller (MBC)

The purpose of MBC controller is to act as a device that will have the inverse transfer function of the vessel, so as to pass the setpoints directly to system output. Besides system inversion it decouples the requests for velocity and angular velocity, to enable the use of separate PID controllers for each of the two variables.

The actual implementation of MBC cannot perform its tasks 100%. For this reason, the inaccuracies at the requested outputs along with the decoupling deficiency are modeled as noise in translational and angular accelerations. To suppress this noise, the PIDs that generate control action requests to MBC are designed for suppressed H_∞ of noise input-to-output transfer function.

MBC operates by taking as inputs:

- PID current output for translational acceleration
- PID current output for angular acceleration
- Current translational velocity value.

- Curent angular velocity value.

Employing vessel's neural network models for translational and angular accelerations it uses current translational velocity value and current angular velocity value as constants and generates all possible outputs for translational and angular accelerations by varying thrust and steer inputs of neural networks. Since thrust and steer values are discrete and limited (0 to 255 for thrust and 0 to 249 for steer), the intermediate outcome of MBC is one 256 x 250 element array of possible vector outcomes, where the first element of the vector is translational acceleration and the second angular acceleration. All possible vectors belong to a 2-dimensional "phase" plane where the x-axis is the possible translational acceleration values and the y-axis the possible angular acceleration values. The word "phase" is in quotes because accelerations are not states of the system (respective velocities are..). For the rest of the paragraph this 2-dimensional plane will be called accelerations plane.

MBC has basically two ways of operating, selected by configuration parameters:

- At 'BestAngleOnly' mode, it searches the accelerations plane for the vector that best matches the angle of the requested accelerations vector. This preference creates a noise regarding the magnitude of the accelerations outputs, but this noise is suppressed by the very frequent reevaluation of new target vectors (every 30ms), PID controllers transfer functions and overall the persistence of supervisory controller to keep the vessel at a course for its target point. It is also the first mode that worked successfully.

- At 'BestAngleAndMag ' mode, the decision for the best vector is evaluated among a set of possible output vectors having approvable angle differences to the requested from the PID controllers. The best match is the one that has the magnitude closer to the requested, among the set of approved-angle vectors. This mode is a logical improvement of the previous one, returning better results as it will be shown in the following paragraphs.

As a graphical example, the operation of MBC modes is shown in figures Figure 25 to Figure 29. The blue arrows are the attainable components by the model. The black arrow is the requested by the PIDs and the green the selected. In figures Figure 25 to Figure 29, the right is a zoomed version of the left.

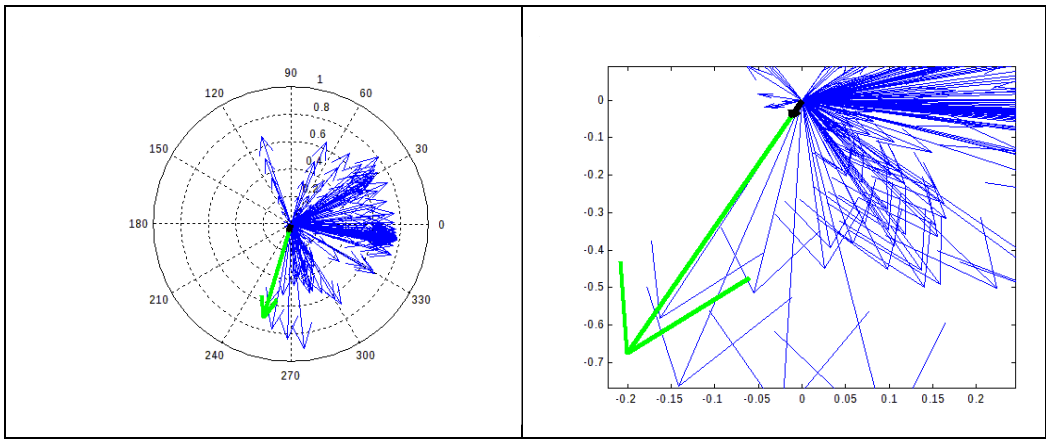


Figure 25: 'BestAngleOnly' mode vector choise (green) and request (black).

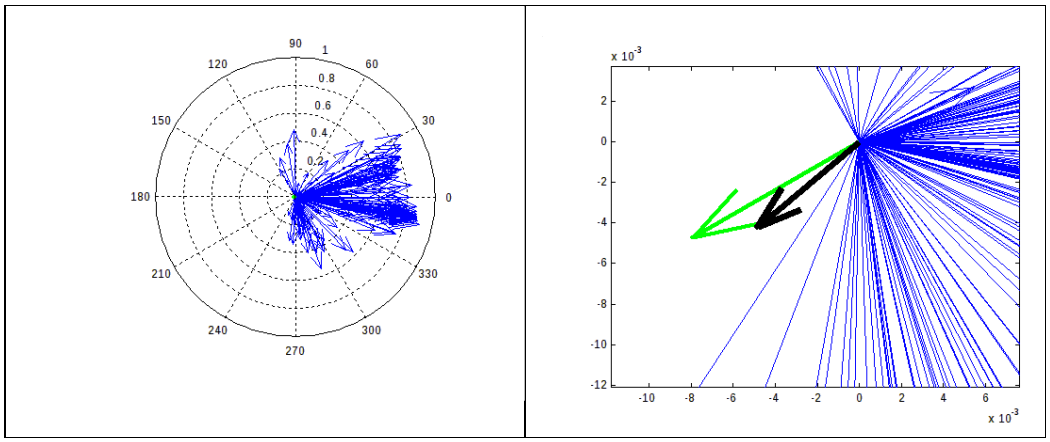


Figure 26: 'BestAngleOnly' mode vector choise (green) and request (black). A second case.

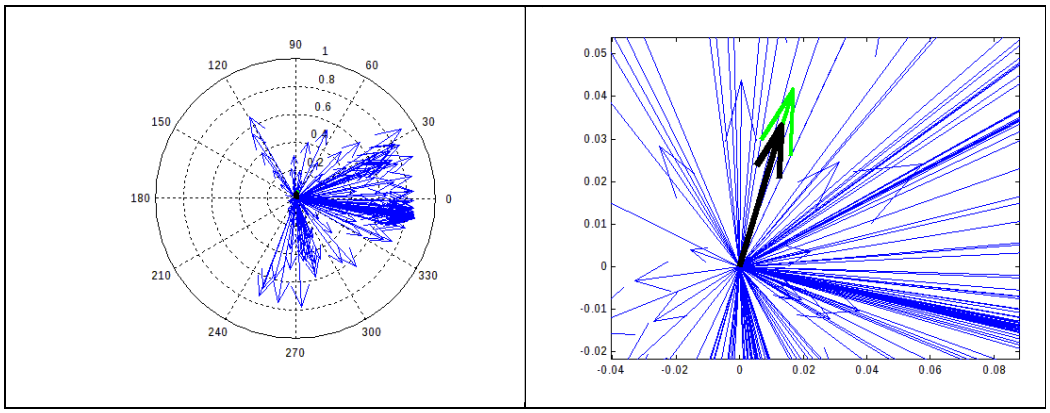


Figure 27: 'BestAngleOnly' mode vector choise (green) and request (black). A third case.

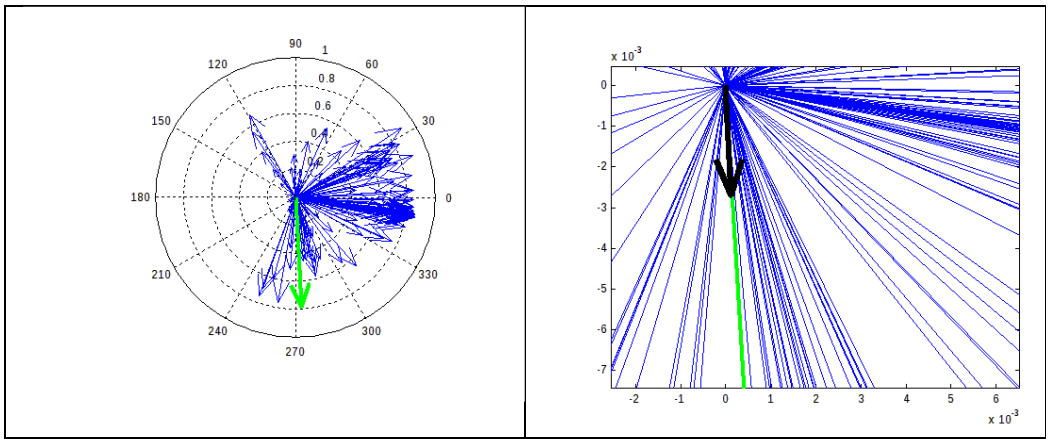


Figure 28: 'BestAngleOnly' mode vector choise (green) and request (black). A fourth case.

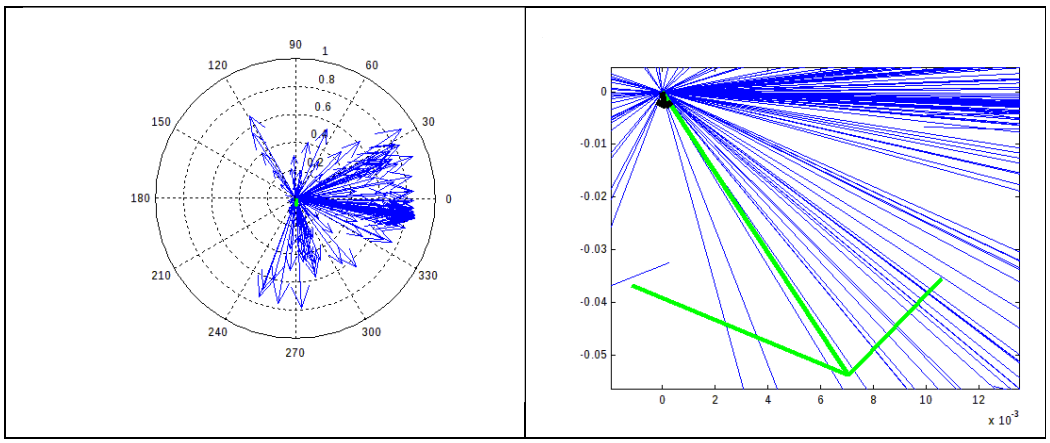


Figure 29: 'BestAngleAndMag' mode vector choise (green) and request (black), for the above fourth case of 'BestAngleOnly' mode.

5.5.1.1 MBC SIMULATION TRIALS

For the evaluation of the MBC controller, a simulation testbed has been designed, which assigns setpoints to MBC and records the output of the controller.

The testbed functions as follows:

- It receives an input file that contains the setpoints of velocity, angular velocity and the simulation repetitions for each setpoint.
- Each simulation step is one T_s (which means 30 ms) apart from the previous. The reason for this is that it operates by connecting MBC to the neural network model of the system. Thus, MBC outputs are applied as inputs to the model. The training of the model has been performed at rate T_s. This dictates T_s as the recommended for the simulations.
- For each setpoint, a loop starts, that operates for the steps assigned at the input file of the testbed. The functionality of the loop is as follows.

- At the beginning a history of three samples is maintained for velocity, angular velocity, and errors between setpoint and current value of velocity and angular velocity.
- A database with the variables described at the previous step is assigned to MBC.
- The MBC generates thrust and steering action for the model as inputs; the model receives as inputs also the current velocities and produces the accelerations (translational and angular) that result from the input variables, approximating the vessel response.
- The accelerations resulting from the previous step are integrated, using the integration approximation of section 5.5.2.1, and the respective velocities are calculated.
- After storing the calculated and resulting values of the current step, the loop begins allover again.

- While each step is performed, the results are stored in files and plotted at the end of all repetitions.

The figures that follow, present the simulation results for the setpoints and simulation repetitions listed at Table 6.

No	Velocity (km/h)	W (o/sec)	Simulation Reps
1	10	0	800
2	15	7	800
3	15	-3	800
4	20	-5	800
5	10	-25	800
6	20	0	800
7	0	0	800
8	10	0	800
9	4	0	800
10	5	7	800
11	2	10	800
12	9	20	800
13	0	0	1000

Table 6: Setpoints used for MBC simulation.

The results from the simulation run for these setpoints follow from Figure 30 up to Figure 35

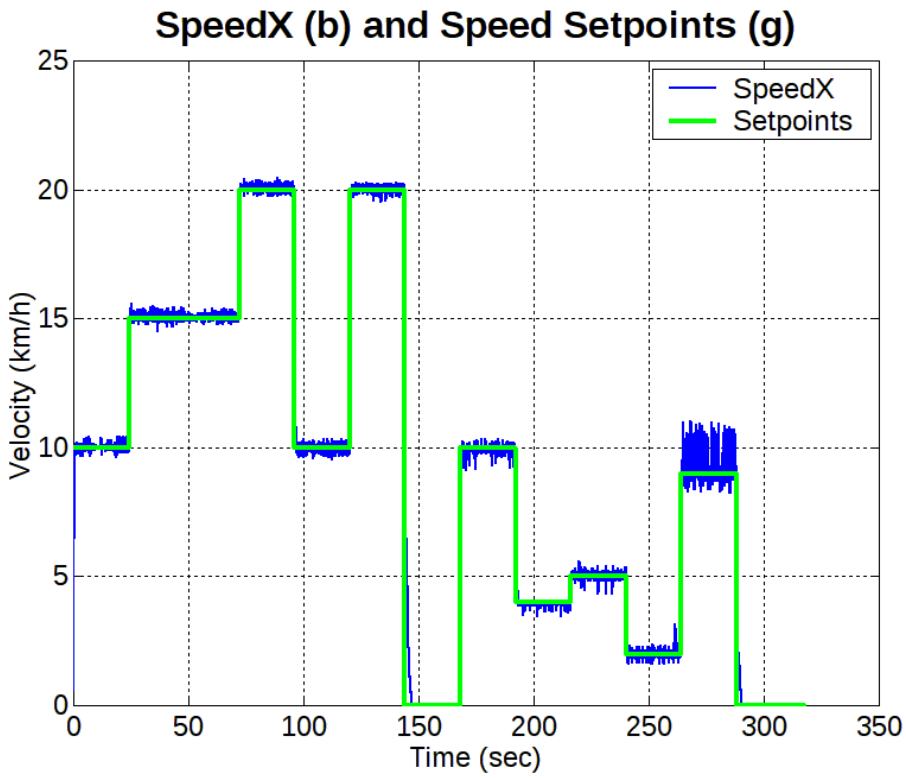


Figure 30: Translational velocity versus velocity setpoints.

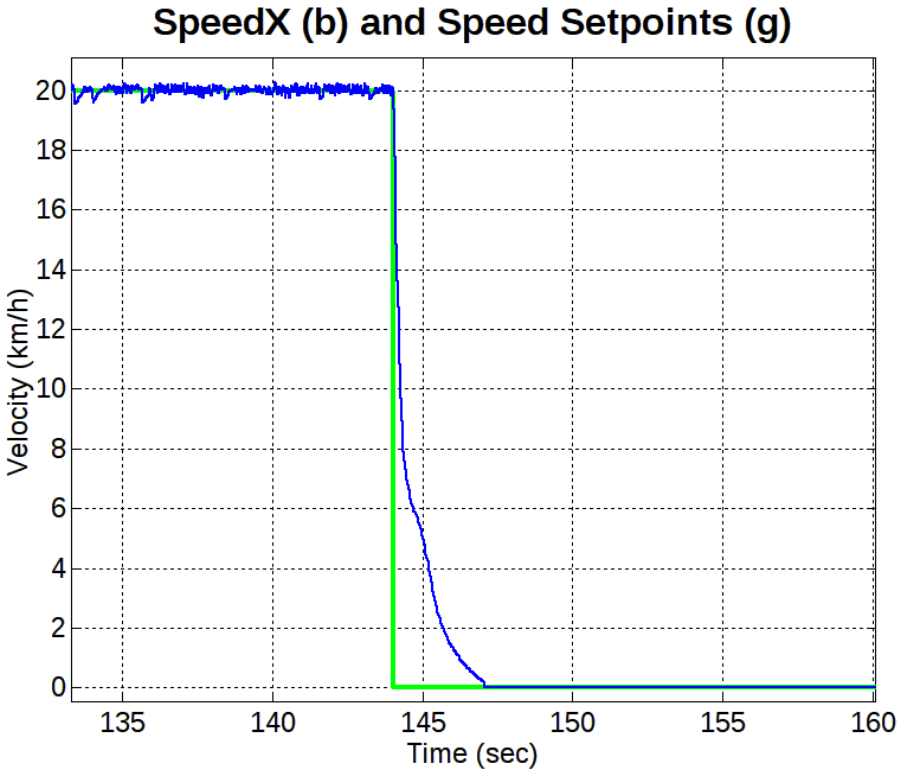


Figure 31: Focus at a certain translational velocity setpoint change.

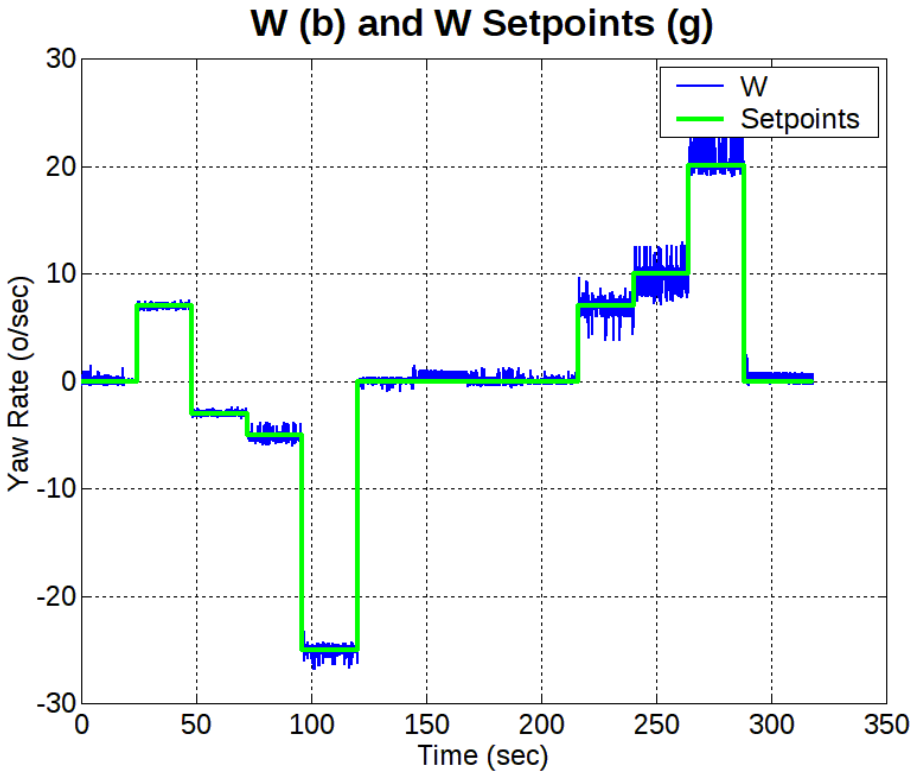


Figure 32: Yaw rate versus yaw rate setpoints.

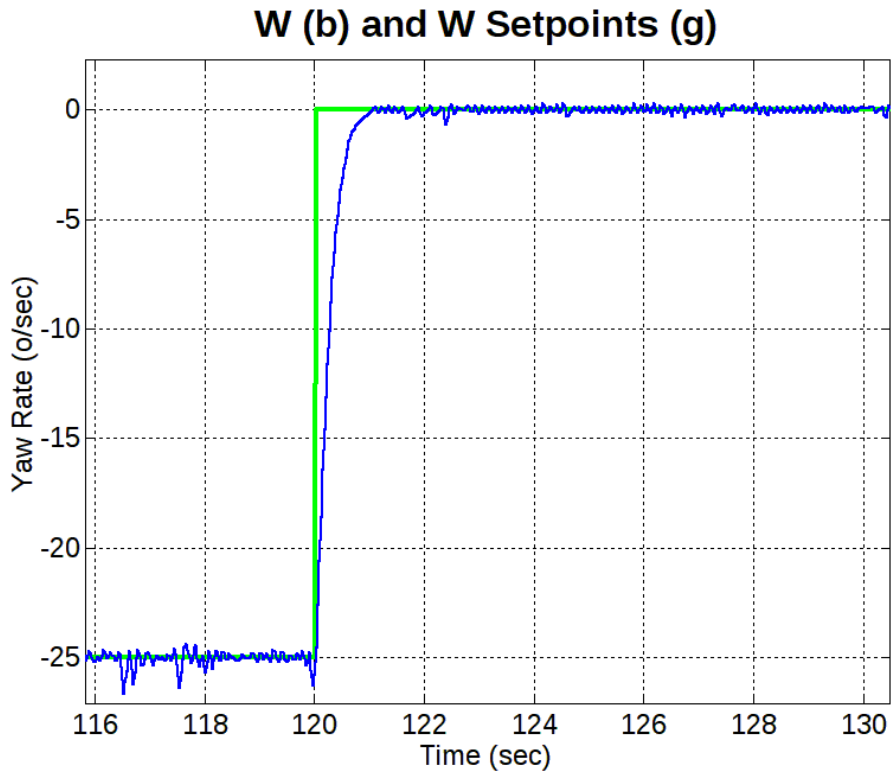


Figure 33: Focus at a certain change of setpoint of yaw rate.

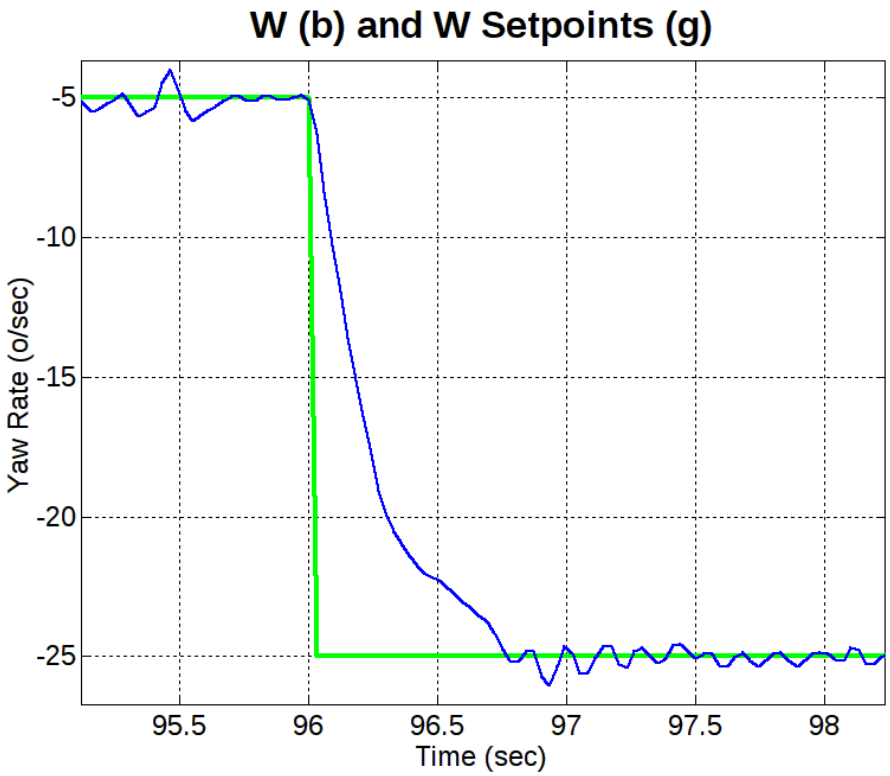


Figure 34: Focus at another change of setpoint of yaw rate.

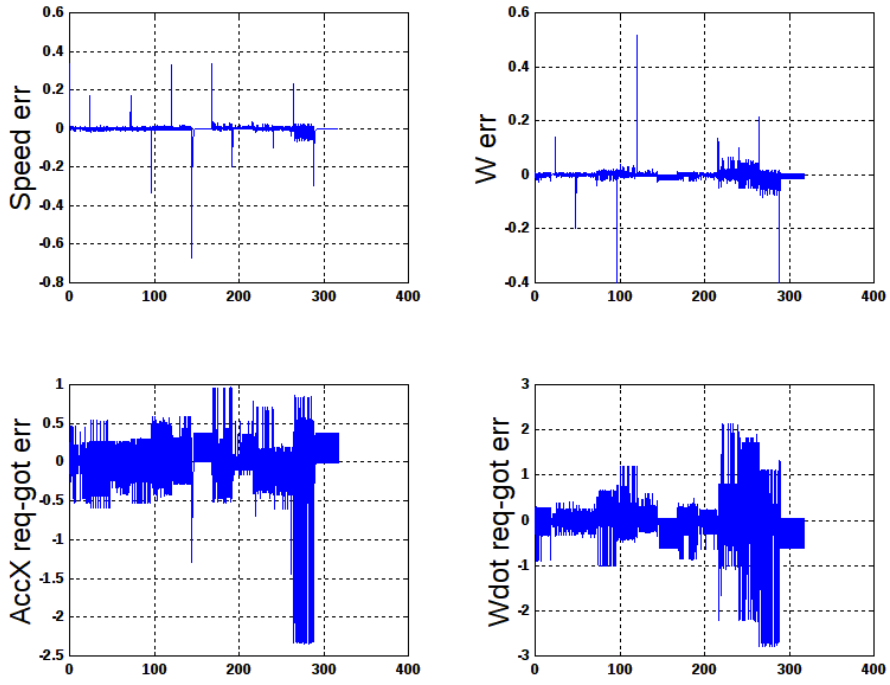


Figure 35: Errors during simulation.

5.5.2 Implementation of PID Controllers (Summation Approximation to Integral)

5.5.2.1 SUMMATION APPROXIMATION TO INTEGRAL

One way of approximating integrals is by using pure summation [13], [15], [22]. This is very much like the approach used by the zero-order hold (ZoH) as we will see in the next sections.

To approximate the integral of a signal by summation, the method depicted in the next figure (Figure 36) is followed [23]:

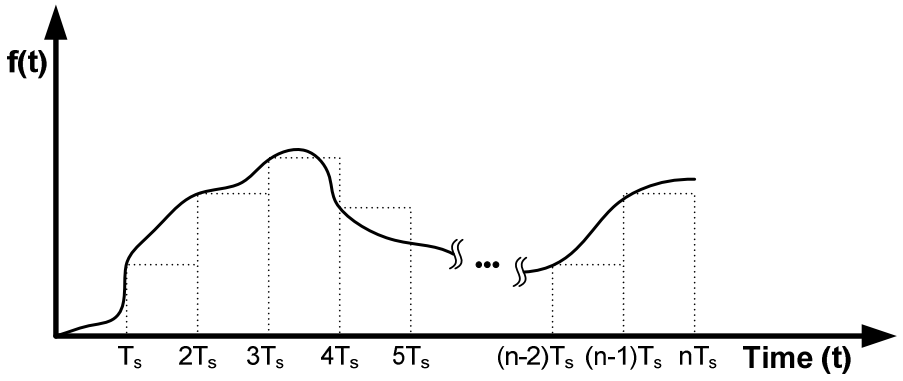


Figure 36: Approximating an integral by summation.

As a reminder, by T_s the sampling period is implied.

As can be seen in Figure 36, the integral of the arbitrary signal is: approximated by the area of parallelograms with height equal to the previous instant sample and width equal to T_s. Such parallelograms give an area of f(k) x T_s for an arbitrary $k \in \mathbb{N}$.

- For time instant T_s: $\int_0^{T_s} f(t)dt \approx f(0)\cdot T_s$
- For time instant $2T_s$: $\int_0^{2T_s} f(t)dt \approx f(0)\cdot T_s + f(1)\cdot T_s$
- For time instant nT_s: $\int_0^{nT_s} f(t)dt \approx f(0)\cdot T_s + f(1)\cdot T_s + \ldots + f(n-1)\cdot T_s$

If we use $y[nT_s]$ as the symbol for the approximation of an integral at nT_s time instant, then from the above equations we have:

$$\int_0^{nT_s} f(t)dt \approx y[nT_s] = f(0)\cdot T_s + f(1)\cdot T_s + \ldots + f(n-1)\cdot T_s \Rightarrow$$

$$y\left[nT_s\right] = \sum_{k=1}^{n} f\left[\left(k-1\right)T_s\right]\cdot T_s \qquad (5.1)$$

Since all the signals we are dealing with are causal, their negative time instant values are zero; for this reason (since $f\left[\left(-1\right)T_s\right] = 0$) we can write (5.1):

$$\boxed{y[nT_s] = \sum_{k=0}^{n} f[(k-1)T_s] \cdot T_s} \tag{5.2}$$

From (5.2) we also notice that:

$$y[T_s] = f[0 \cdot T_s] \cdot T_s = f(0) \cdot T_s$$

$$y[2T_s] = f(0) \cdot T_s + f(T_s) \cdot T_s = y[T_s] + f(T_s) \cdot T_s$$

which means that for the general case

$$y[nT_s] = y[(n-1)T_s] + f[(n-1)T_s] \cdot T_s \tag{5.3}$$

By taking the Z-transform of equation (5.3) we have that

$$Y(z) = z^{-1}Y(z) + z^{-1}T_s F(z) \Leftrightarrow$$

$$\boxed{Y(z) = \frac{z^{-1}T_s}{1 - z^{-1}} F(z)} \tag{5.4}$$

If we want to depict (5.4) in the form of a block diagram, then we get

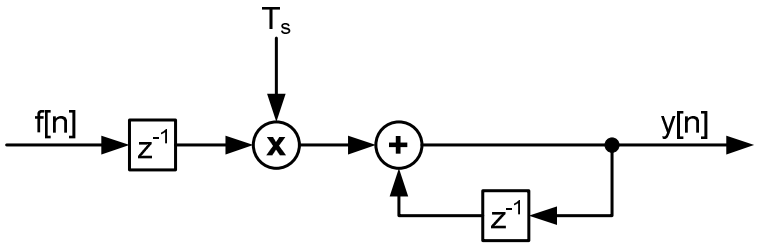


Figure 37: Block diagram form of integration approximation.

A block diagram that is more efficient regarding memory element usage can be derived if we simplify the above diagram by block diagram manipulation or by considering that:

$$Y(z) = z^{-1}Y(z) + z^{-1}T_s F(z) = z^{-1}[Y(z) + T_s F(z)]$$

Which means, that the diagram of Figure 37 can be considered equivalent to

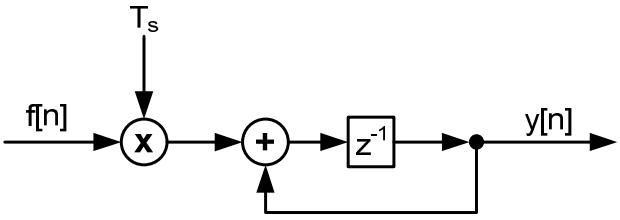


Figure 38: Improved block diagram of integration approximation.

5.5.2.2 DIGITIZING THE ANALOG PID FUNCTIONALITY

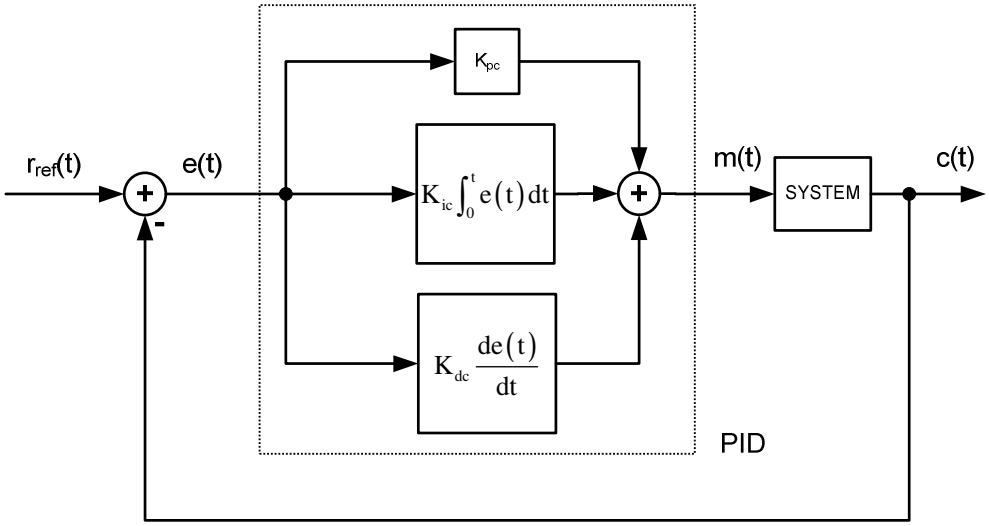


Figure 39: Analog PID.

To convert the analog PID controller to digital [10], we examine each term separately, something that is correct, since linear systems have the property of superposition.

P-term:

The P-term is the simplest of all. It amplifies any incoming signal by a gain of K_{pc}, where the c stands for "continuous", implying the continuous system gain; this was done to avoid mixing these constants with Kp that is derived for digital controllers after manipulations. The c-marking is followed for all PID term gains.
The input to digital PID P-term is the sampled error signal $e[nT_s]$. The output of P-term is:

$$m_p[nT_s] = K_{pc}e[nT_s] \tag{5.5}$$

or in terms of Z-transorm

$$M_p(z) = K_{pc}E(z) \tag{5.6}$$

The block diagram of P-term is the following

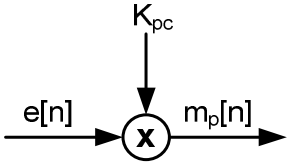


Figure 40: Digital P-term block diagram.

It must be noted here that by e[n] we imply $e[nT_s]$ (also applicable to all discrete signals) and we will use both notations interchangeably throughout this text.

I-term:

For the I-term we use the approximation analyzed in section "Summation Approximation to Integral". This results in the following equations:

$$K_{ic}\int_0^t e(\tau)d\tau = K_{ic}\int_0^{nT_s} e(\tau)d\tau \approx K_{ic}\sum_{k=0}^{n} e\left[(k-1)T_s\right]\cdot T_s$$

$$\boxed{m_i[nT_s] = K_{ic}T_s\sum_{k=0}^{n} e\left[(k-1)T_s\right]} \tag{5.7}$$

or in terms of Z-transform

$$Z\{m_i[nT_s]\} = K_{ic}T_s Z\left\{\sum_{k=0}^{n} e\left[(k-1)T_s\right]\right\} \Rightarrow$$

$$\boxed{M_i(z) = K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}E(z)} \tag{5.8}$$

For the block diagram of i-term we can manipulate equation (5.8) to get

$$M_i(z) = z^{-1}\left[M_i(z) + K_{ic}T_s E(z)\right]$$

for which we have

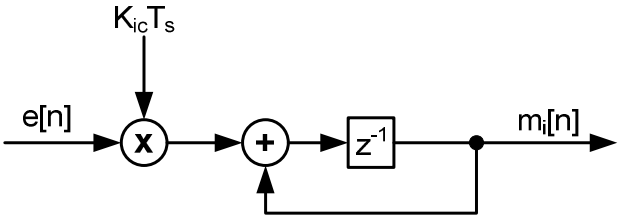


Figure 41: Digitel I-term block diagram.

D-term:

For the D-term we approximate the derivative by differences as follows:

$$m_d(t) = K_{dc}\frac{de(t)}{dt} \approx K_{dc}\frac{\Delta e(t)}{\Delta t} \Rightarrow$$

$$\boxed{m_d[nT_s] = K_{dc}\frac{e[nT_s] - e\left[(n-1)T_s\right]}{T_s}} \tag{5.9}$$

or in terms of Z-transform

$$\boxed{M_d(z) = \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)E(z)} \tag{5.10}$$

The block diagram of (5.10) is

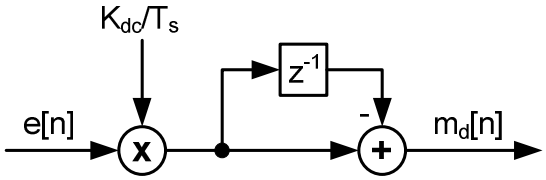


Figure 42: Digital D-term block diagram.

Combining all terms:

After analyzing each term separately, we can now provide the equation for the total output of digital PID controller [5]:

$$m[nT_s] = m_p[nT_s] + m_i[nT_s] + m_d[nT_s] \Rightarrow$$

$$\boxed{m[nT_s] = K_{pc}e[nT_s] + K_{ic}T_s\sum_{k=0}^{n} e[(k-1)T_s] + K_{dc}\frac{e[nT_s] - e[(n-1)T_s]}{T_s}} \quad (5.11)$$

or in terms of Z-transform:

$$M(z) = M_p(z) + M_i(z) + M_d(z) \Rightarrow$$

$$M(z) = K_{pc}E(z) + K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}E(z) + \frac{K_{dc}}{T_s}(1-z^{-1})E(z) \Leftrightarrow$$

$$\boxed{M(z) = \left[K_{pc} + K_{ic}\frac{z^{-1}T_s}{1-z^{-1}} + \frac{K_{dc}}{T_s}(1-z^{-1})\right]E(z)} \quad (5.12)$$

What should be pointed out is that the digitization of the analog PID introduces T_s as a multiplier in I and D terms. This reveals the utmost importance of sampling frequency selection, because improper selection of T_s can lead to unstable digital implementation of an otherwise stable analog PID controlled system.

5.5.2.3 BLOCK DIAGRAMS FOR DIGITAL PID

Equation (5.12) can also be shown as a block diagram after some mathematical manipulations. We eliminate the $(1-z^{-1})$ factor from the denominator, which gives

$$(1-z^{-1})M(z) = \left[(1-z^{-1})K_{pc} + K_{ic}T_sz^{-1} + \frac{K_{dc}}{T_s}(1-z^{-1})^2\right]E(z) \Leftrightarrow$$

$$(1-z^{-1})M(z) = \left[K_{pc} - z^{-1}K_{pc} + K_{ic}T_sz^{-1} + \frac{K_{dc}}{T_s}(1-2z^{-1}+z^{-2})\right]E(z) \Leftrightarrow$$

$$\boxed{M(z) = z^{-1}M(z) + \left[\left(K_{pc} + \frac{K_{dc}}{T_s}\right) + \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)z^{-1} + \frac{K_{dc}}{T_s}z^{-2}\right]E(z)} \quad (5.13)$$

$$\begin{aligned} m[nT_s] &= m[(n-1)T_s] + \left(K_{pc} + \frac{K_{dc}}{T_s}\right) e[nT_s] + \\ &+ \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right) e[(n-1)T_s] + \\ &+ \frac{K_{dc}}{T_s} e[(n-2)T_s] \end{aligned} \tag{5.14}$$

Equations (5.13) or (5.14) give the block diagram of Figure 43:

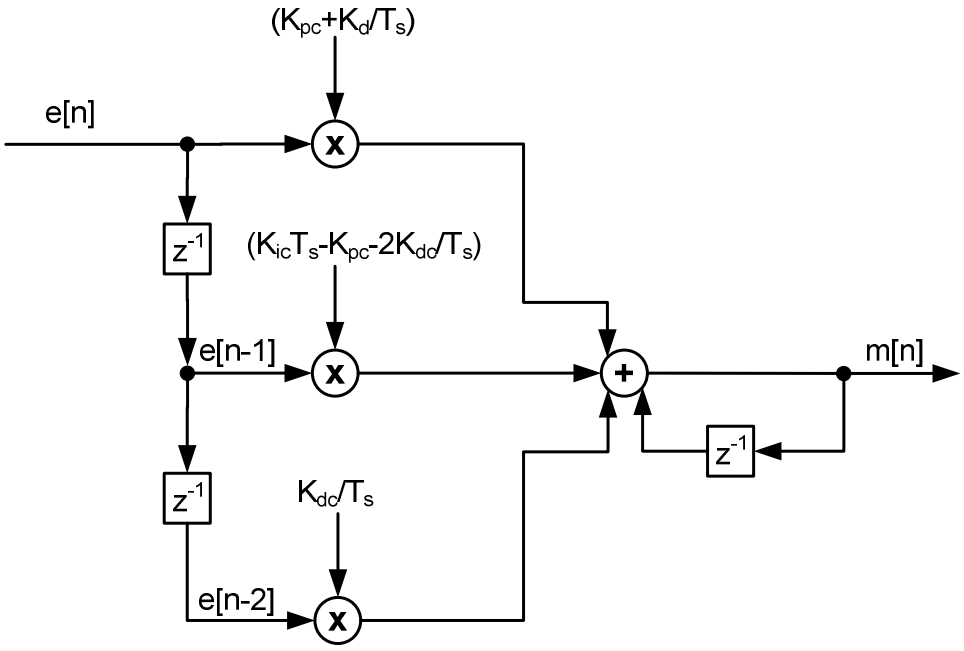


Figure 43: Block diagram for digital PID implementation.

5.5.2.4 ALTERNATIVE PID IMPLEMENTATION (VELOCITY FORM)

There is also an alternative implementation of Digital PID, which focuses on reducing the impact of sudden changes at input. The mathematical trick that implements the above statement will be shown at the derivation of the equations that follows.

This form can also be used in our application, and relevant results will be given.

To derive the equations for velocity form we begin from equation (5.11) of the previous section and elaborate:

$$m[nT_s] = K_{pc} e[nT_s] + K_{ic} T_s \sum_{k=0}^{n} e[(k-1)T_s] + K_{dc} \frac{e[nT_s] - e[(n-1)T_s]}{T_s}$$

We want to find the difference between successive samples of the controller, which will be denoted as $\nabla m\left[nT_s\right]$.

$$\nabla m\left[nT_s\right]=m\left[nT_s\right]-m\left[\left(n-1\right)T_s\right]\Rightarrow$$

$$\begin{aligned}\nabla m\left[nT_s\right]= \quad & K_{pc}\left\{e[nT_s]-e[\left(n-1\right)T_s]\right\}+\\ & K_{ic}T_s\left\{\sum_{k=0}^{n}e\left[\left(k-1\right)T_s\right]-\sum_{k=0}^{n-1}e\left[\left(k-1\right)T_s\right]\right\}+\\ & \frac{K_{dc}}{T_s}\left\{e\left[nT_s\right]-e\left[\left(n-1\right)T_s\right]-e\left[\left(n-1\right)T_s\right]+e\left[\left(n-2\right)T_s\right]\right\}\Leftrightarrow\end{aligned}$$

$$\begin{aligned}\nabla m\left[nT_s\right]= \quad & K_{pc}\left\{e[nT_s]-e[\left(n-1\right)T_s]\right\}+\\ & K_{ic}T_s e\left[\left(n-1\right)T_s\right]+\\ & \frac{K_{dc}}{T_s}\left\{e\left[nT_s\right]-2e\left[\left(n-1\right)T_s\right]+e\left[\left(n-2\right)T_s\right]\right\}\Leftrightarrow\end{aligned}$$

$$\begin{aligned}\nabla m\left[nT_s\right]= \quad & \left(K_{pc}+\frac{K_{dc}}{T_s}\right)e[nT_s]+\\ & \left(K_{ic}T_s-K_{pc}-\frac{2K_{dc}}{T_s}\right)e\left[\left(n-1\right)T_s\right]+\\ & \frac{K_{dc}}{T_s}e\left[\left(n-2\right)T_s\right]\end{aligned}\qquad(5.15)$$

By replacing $e[nT_s]$ with its equivalent i.e. $r_{ref}[nT_s]-c[nT_s]$, (5.15) becomes

$$\begin{aligned}\nabla m\left[nT_s\right]= \quad & \left(K_{pc}+\frac{K_{dc}}{T_s}\right)r_{ref}[nT_s]+\\ & \left(K_{ic}T_s-K_{pc}-\frac{2K_{dc}}{T_s}\right)r_{ref}\left[\left(n-1\right)T_s\right]+\\ & \frac{K_{dc}}{T_s}r_{ref}\left[\left(n-2\right)T_s\right]-\\ & \left(K_{pc}+\frac{K_{dc}}{T_s}\right)c[nT_s]-\\ & \left(K_{ic}T_s-K_{pc}-\frac{2K_{dc}}{T_s}\right)c\left[\left(n-1\right)T_s\right]-\\ & \frac{K_{dc}}{T_s}c\left[\left(n-2\right)T_s\right]\end{aligned}\qquad(5.16)$$

The mathematical trick that is used in derivation of velocity form is to force equality of the last three samples of reference input: $r_{ref}[nT_s] = r_{ref}[(n-1)T_s] = r_{ref}[(n-2)T_s]$ [4].

This was inspired by the factors of r_{ref}, in order to eliminate the proportional and derivative action on the reference input, thus drastically reducing the effect of a change in the reference input. With this assumption (5.16) changes to

$$\begin{aligned}\nabla m[nT_s] = m[nT_s] - m[(n-1)T_s] = \\ K_{ic}T_s r_{ref}[nT_s] - \\ \left(K_{pc} + \frac{K_{dc}}{T_s}\right)c[nT_s] - \\ \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)c[(n-1)T_s] - \\ \frac{K_{dc}}{T_s}c[(n-2)T_s] \Leftrightarrow\end{aligned}$$

$$\boxed{\begin{aligned}m[nT_s] = m[(n-1)T_s] + \\ K_{ic}T_s r_{ref}[nT_s] - \\ \left(K_{pc} + \frac{K_{dc}}{T_s}\right)c[nT_s] - \\ \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)c[(n-1)T_s] - \\ \frac{K_{dc}}{T_s}c[(n-2)T_s]\end{aligned}} \quad (5.17)$$

From (5.17) we can proceed with Z-transform to provide a formula in z.

$$\begin{aligned}M(z) = z^{-1}M(z) + K_{ic}T_s R_{ref}(z) - \left(K_{pc} + \frac{K_{dc}}{T_s}\right)C(z) \\ - \left(K_{ic}T_s - K_{pc} - \frac{2K_{dc}}{T_s}\right)z^{-1}C(z) - z^{-2}\frac{K_{dc}}{T_s}C(z) \Leftrightarrow\end{aligned}$$

$$\begin{aligned}(1-z^{-1})M(z) = K_{ic}T_s\left[R_{ref}(z) - z^{-1}C(z)\right] - K_{pc}(1-z^{-1})C(z) - \\ \frac{K_{dc}}{T_s}(1-2z^{-1}+z^{-2})C(z) \Leftrightarrow\end{aligned}$$

$$\begin{aligned}(1-z^{-1})M(z) = K_{ic}T_s\left[R_{ref}(z) - z^{-1}C(z)\right] - K_{pc}(1-z^{-1})C(z) - \\ \frac{K_{dc}}{T_s}(1-z^{-1})^2 C(z) \Leftrightarrow\end{aligned}$$

$$M(z) = -K_{pc}C(z) + K_{ic}T_s\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)C(z) \qquad (5.18)$$

A block diagram for the implementation of (5.17) or (5.18) is provided in the Figure 44.

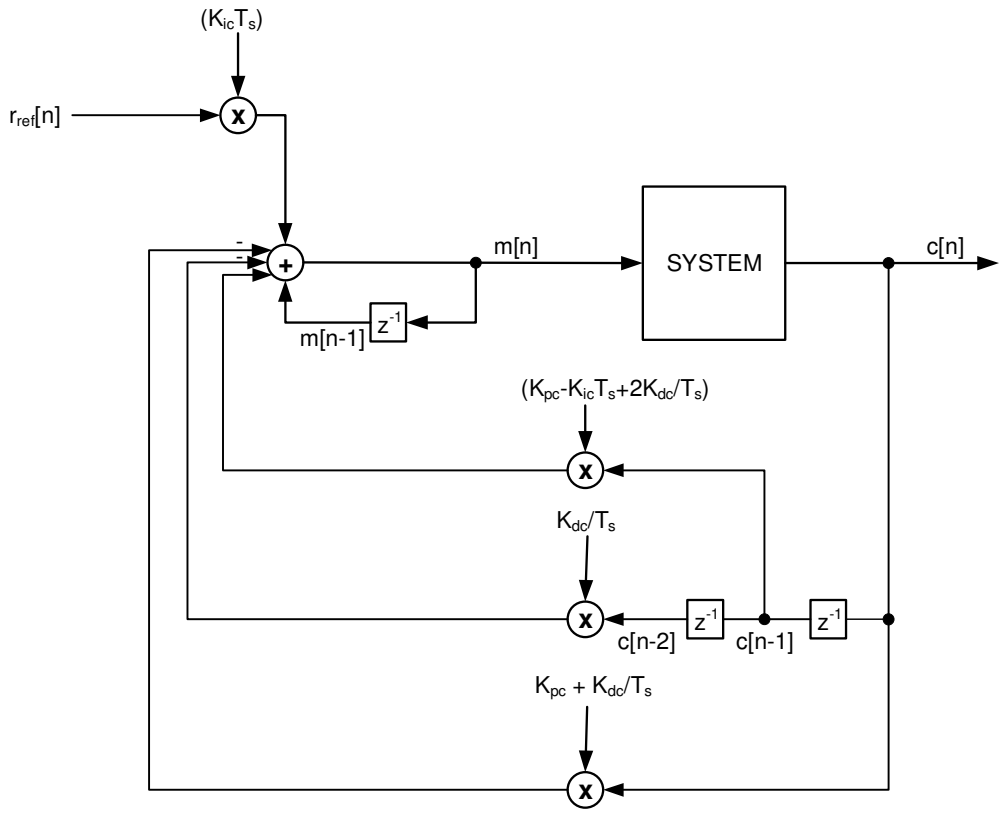


Figure 44: Block diagram of velocity form implementation of PID

5.6 Inserting PIDs into the System

At this point it is important to show how the system is meant to be controlled by the use of PID controllers.

The use of MBC controllers decouples the system, thus two separate PID controlled loops can be formed; one for angular velocity and one for translational velocity. The next important thing is to identify which parts of the loop belong to digital domain and which parts belong to the analog domain. This will determine the correct position and number of analog-to-digital (A/Ds or ADCs) and digital-to-analog (D/As or DACs) that are needed.

Referring to “Figure 23: Full system control scheme.” and “Figure 24: Feedback Controller and its operation in the system.” the parts that belong to the digital domain are all apart from the system itself. This means that the only places where converters interfacing the digital to the analog world are needed are system boundaries.

The block diagram that describes the system interfacing is shown in Figure 45.

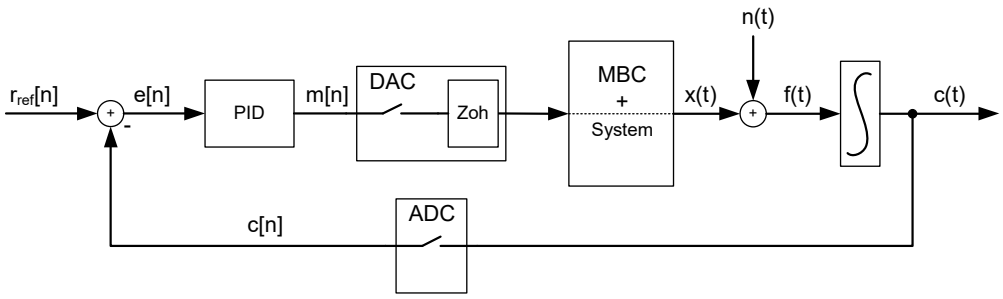


Figure 45: Control loop with PID and data converters.

Due to MBC acting as (almost) the inverse of the system, the output of PID controller passes through Zoh and from there to $\dot{c}(t)$ along with some noise due to

- Modelling uncertainties
- Probable inefficiencies of MBC to fully invert the system
- Probable inefficiencies of MBC to fully decouple the system
- External to the full system causes such as wind or waves

All the above causes are modeled as noise added before f(t).

5.6.1 Mathematical Analysis in Discrete Time

The first thing that must be said, regarding the analysis of the system in discrete time, is that the fictious impulse samplers [4], [11], [15] representing half DAC (the other half of DAC representation being Zoh) and ADC must be synchronized; this means same frequency and phase. This saves a lot of effort in the mathematical domain (such as sample rate converters) and guarantees that the signals available for processing refer to same sampling instant. Also, any internal processing must be aligned with the sampling frequency of both converters and must be carried out at time less that the sampling period.

Having already analyzed the digital PID (refer to equations (5.12) to (5.14)) the next step is to analyze Zoh contribution in discrete time.

5.6.1.1 ZOH MATHEMATICAL TREATMENT

What zero order hold does, is keeping steady the value between sampling instants. Mathematically and referring to Figure 46 [4]:

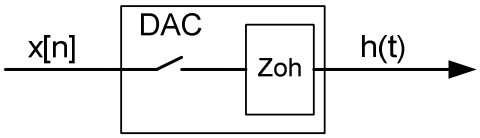


Figure 46: Reference figure for Zoh analysis

$$h(nT_s+t)=x(nT_s),t\in[0,T_s)$$

This means that

$$\begin{aligned}h(t)=&x(0)\left[U(t)-U(t-T_s)\right]+x(T_s)\left[U(t-T_s)-U(t-2T_s)\right]+\dots\\&+x(nT_s)\left[U(t-nT_s)-U\left[t-(n+1)T_s\right]\right]\end{aligned}\tag{5.19}$$

By taking the Laplace transform of (5.19) we have that

$$L\{h(t)\}=x(0)\left(\frac{1}{s}-e^{-sT_s}\frac{1}{s}\right)+x(T_s)\left(e^{-sT_s}\frac{1}{s}-e^{-s2T_s}\frac{1}{s}\right)+\dots$$

$$+x(nT_s)\left[e^{-snT_s}\frac{1}{s}-e^{-s(n+1)T_s}\frac{1}{s}\right]\Rightarrow$$

$$H(s)=\sum_{n=0}^{\infty}\left[x(nT_s)e^{-snT_s}\frac{\left(1-e^{-sT_s}\right)}{s}\right]\Leftrightarrow$$

$$H(s)=\frac{\left(1-e^{-sT_s}\right)}{s}\sum_{n=0}^{\infty}x(nT_s)e^{-snT_s}\tag{5.20}$$

The sum at equation (5.20) is the starred Laplace transform of signal x(t) (equals Z-tranform with $z=e^{sT_s}$), which means that (5.20) translates to

$$H(s)=G_{Zoh}(s)X^*(s)\tag{5.21}$$

which in turn means that

$$\boxed{G_{Zoh}(s)=\frac{\left(1-e^{-sT_s}\right)}{s}}\tag{5.22}$$

Since analysis of the system in discrete time involves Z-transform, then it is important to find the Z-transform of a continuous time transfer function when multiplied by the Zoh transfer function.

To do so we suppose that an arbitrary signal Y(s) is

$$Y(s)=G_{Zoh}(s)G(s)\Rightarrow$$

$$Y(s)=\frac{1-e^{-sT_s}}{s}G(s)=\frac{G(s)}{s}-e^{-sT_s}\frac{G(s)}{s}\tag{5.23}$$

if $\frac{G(s)}{s}=G_1(s)$ then (5.23) becomes

$$Y(s)=G_1(s)-e^{-sT_s}G_1(s)\Rightarrow$$

$$L^{-1}\{Y(s)\} = L^{-1}\{G_1(s)\} - L^{-1}\{e^{-sT_s}G_1(s)\} \Rightarrow$$

$$y(t) = g_1(t) - \int_0^t \delta(t - T_s - \tau) g_1(\tau) d\tau \Rightarrow$$

$$y(t) = g_1(t) - g_1(t - T_s) \Rightarrow$$

$$Y(z) = G_1(z) - z^{-1}G_1(z) \Leftrightarrow$$

$$\boxed{Y(z) = (1 - z^{-1})G_1(z) \text{ where } G_1(z) = Z\left\{\frac{G(s)}{s}\right\}} \quad (5.24)$$

5.6.1.2 APPLYING PREVIOUS RESULTS TO CLOSED LOOP

After completing the analysis of positioning the samplers in the loop, MBC + System result and Zoh transfer function, we are ready to proceed to full loop analysis.

Starting with the continuous time section, we have (in s):

$$F(s) = X(s) + N(s) \quad (5.25)$$

$$X(s) = M^*(s)\frac{1 - e^{-sT_s}}{s} \quad (5.26)$$

Note that with the * we imply the starred Laplace transform i.e. the Laplace transform of an impulse sampled signal.

$$C(s) = F(s)\frac{1}{s} \quad (5.27)$$

Considering N(s) = 0 for the moment (superposition step), we have from (5.25), (5.26) and (5.27) that

$$C(s) = M^*(s)\frac{(1 - e^{-sT_s})}{s} \cdot \frac{1}{s} \Rightarrow$$

$$C(z) = Z\left\{M^*(s)\frac{(1 - e^{-sT_s})}{s} \cdot \frac{1}{s}\right\} = M(z)(1 - z^{-1})Z\left\{\frac{1}{s^2}\right\} = M(z)\cancel{(1 - z^{-1})}\frac{T_s z^{-1}}{\cancel{(1 - z^{-1})^2}} \Leftrightarrow$$

$$C(z) = M(z)\frac{T_s z^{-1}}{(1 - z^{-1})} \quad (5.28)$$

Taking M(z) from (5.12), (5.28) becomes

$$C(z) = \left[K_{pc} + K_{ic}\frac{z^{-1}T_s}{1 - z^{-1}} + \frac{K_{dc}}{T_s}(1 - z^{-1})\right]\frac{T_s z^{-1}}{(1 - z^{-1})}E(z) \quad (5.29)$$

Considering that

$$E(z) = R_{ref}(z) - C(z) \quad (5.30)$$

we have that (5.29) is

$$C(z)=\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}\left[R_{ref}(z)-C(z)\right]\Leftrightarrow$$

$$\boxed{C(z)=\frac{\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}}{1+\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]\frac{T_s z^{-1}}{\left(1-z^{-1}\right)}}R_{ref}(z)} \qquad (5.31)$$

Equation (5.31) is the closed loop pulse transfer function from reference input $R_{ref}(z)$ to output C(z).

The block diagram for the implementation of (5.31) follows in Figure 47:

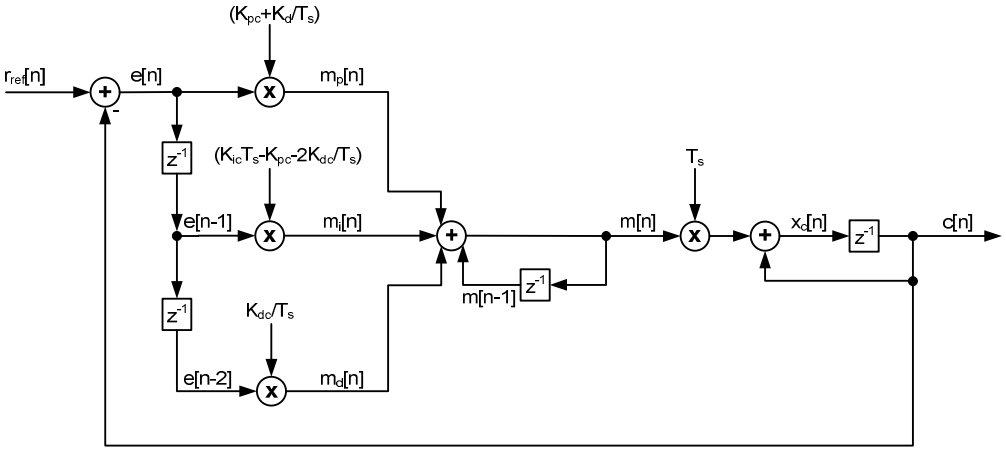


Figure 47: Block diagram for closed loop analysis in discrete time

Elaborating on (5.31) we can also find an equation that gives the same result for cross-checking

$$C(z)=\frac{\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]T_s z^{-1}}{\left(1-z^{-1}\right)+\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{1-z^{-1}}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]T_s z^{-1}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{\left[\frac{K_{pc}\left(1-z^{-1}\right)}{\left(1-z^{-1}\right)}+K_{ic}\frac{z^{-1}T_s}{\left(1-z^{-1}\right)}+\frac{K_{dc}}{T_s}\frac{\left(1-z^{-1}\right)^2}{\left(1-z^{-1}\right)}\right]T_s z^{-1}}{\left(1-z^{-1}\right)+\left[K_{pc}+K_{ic}\frac{z^{-1}T_s}{\left(1-z^{-1}\right)}+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)\right]T_s z^{-1}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{\left[K_{pc}\left(1-z^{-1}\right)+K_{ic}z^{-1}T_s+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)^2\right]T_sz^{-1}}{\left(1-z^{-1}\right)^2+\left[K_{pc}\left(1-z^{-1}\right)+K_{ic}z^{-1}T_s+\frac{K_{dc}}{T_s}\left(1-z^{-1}\right)^2\right]T_sz^{-1}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{K_{pc}T_sz^{-1}-K_{pc}T_sz^{-2}+K_{ic}z^{-2}T_s^2+K_{dc}z^{-1}\left(1-2z^{-1}+z^{-2}\right)}{1-2z^{-1}+z^{-2}+K_{pc}T_sz^{-1}-K_{pc}T_sz^{-2}+K_{ic}z^{-2}T_s^2+K_{dc}z^{-1}\left(1-2z^{-1}+z^{-2}\right)}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{K_{pc}T_sz^{-1}-K_{pc}T_sz^{-2}+K_{ic}z^{-2}T_s^2+K_{dc}z^{-1}-2K_{dc}z^{-2}+K_{dc}z^{-3}}{1-2z^{-1}+z^{-2}+K_{pc}T_sz^{-1}-K_{pc}T_sz^{-2}+K_{ic}z^{-2}T_s^2+K_{dc}z^{-1}-2K_{dc}z^{-2}+K_{dc}z^{-3}}R_{ref}(z)\Leftrightarrow$$

$$C(z)=\frac{\left(K_{pc}T_s+K_{dc}\right)z^{-1}+\left(K_{ic}T_s^2-K_{pc}T_s-2K_{dc}\right)z^{-2}+K_{dc}z^{-3}}{1+\left(K_{pc}T_s-2+K_{dc}\right)z^{-1}+\left(1-K_{pc}T_s+K_{ic}T_s^2-2K_{dc}\right)z^{-2}+K_{dc}z^{-3}}R_{ref}(z)\Rightarrow$$

$$\begin{aligned}c[n]=&\left(K_{pc}T_s+K_{dc}\right)r_{ref}\left[(n-1)T_s\right]+\\&+\left(K_{ic}T_s^2-K_{pc}T_s-2K_{dc}\right)r_{ref}\left[(n-2)T_s\right]+\\&+K_{dc}r_{ref}\left[(n-3)T_s\right]+\\&-\left(K_{pc}T_s-2+K_{dc}\right)c\left[(n-1)T_s\right]+\\&-\left(1-K_{pc}T_s+K_{ic}T_s^2-2K_{dc}\right)c\left[(n-2)T_s\right]+\\&-K_{dc}c\left[(n-3)T_s\right]\end{aligned}\qquad(5.32)$$

5.6.1.3 ANALYZING CLOSED LOOP WITH VELOCITY FORM

The system diagram using velocity form PID becomes as shown in Figure 48. The diagram also presents the actual positions of the converters in the system, and as discussed in the previous section the analysis takes as a fact that the converters are fully synchronized.

To analyze in Z, we must replace the system box with our approximation of the system, which is a Zoh followed by the 1/s function, which gives equation (5.28) (see the respective derivation).

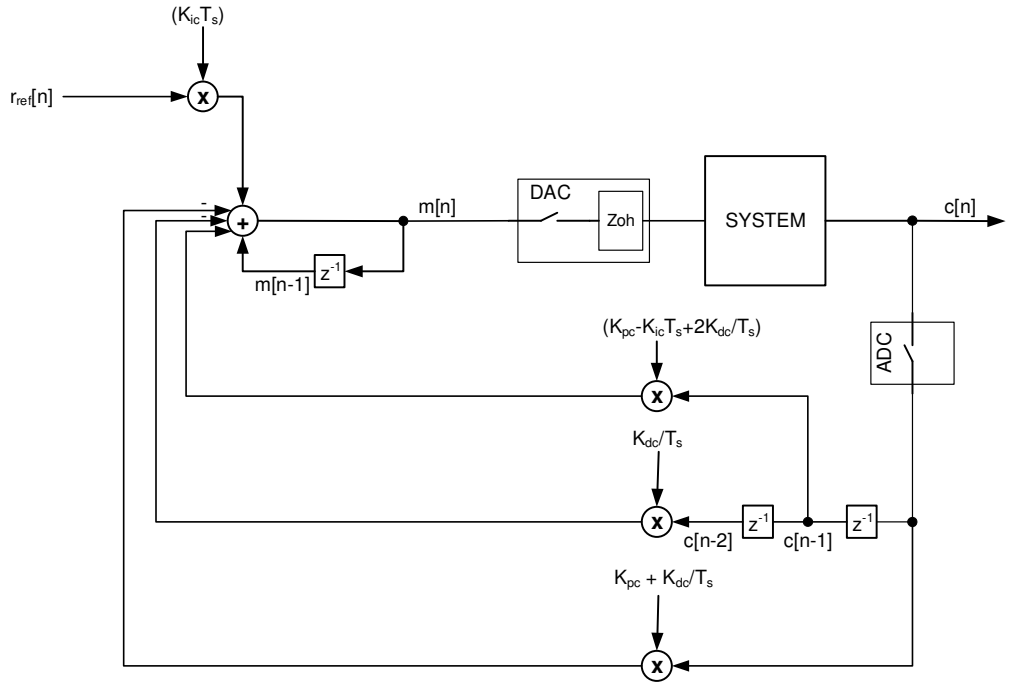


Figure 48: Using Velocity Form to Control the System. Converters are shown.

Repeating (5.28) here we have:

$$C(z) = M(z)\frac{T_s z^{-1}}{(1-z^{-1})} \Leftrightarrow C(z) = z^{-1}\left[C(z) + T_s M(z)\right]$$

which translates to the diagram of Figure 49.

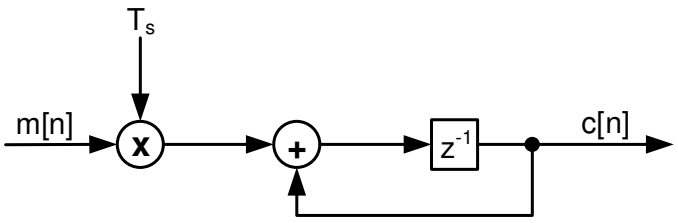


Figure 49: System with Zoh representation.

Replacing the "System" box of Figure 48 with its approximation as shown in Figure 49 we have the result of Figure 50.

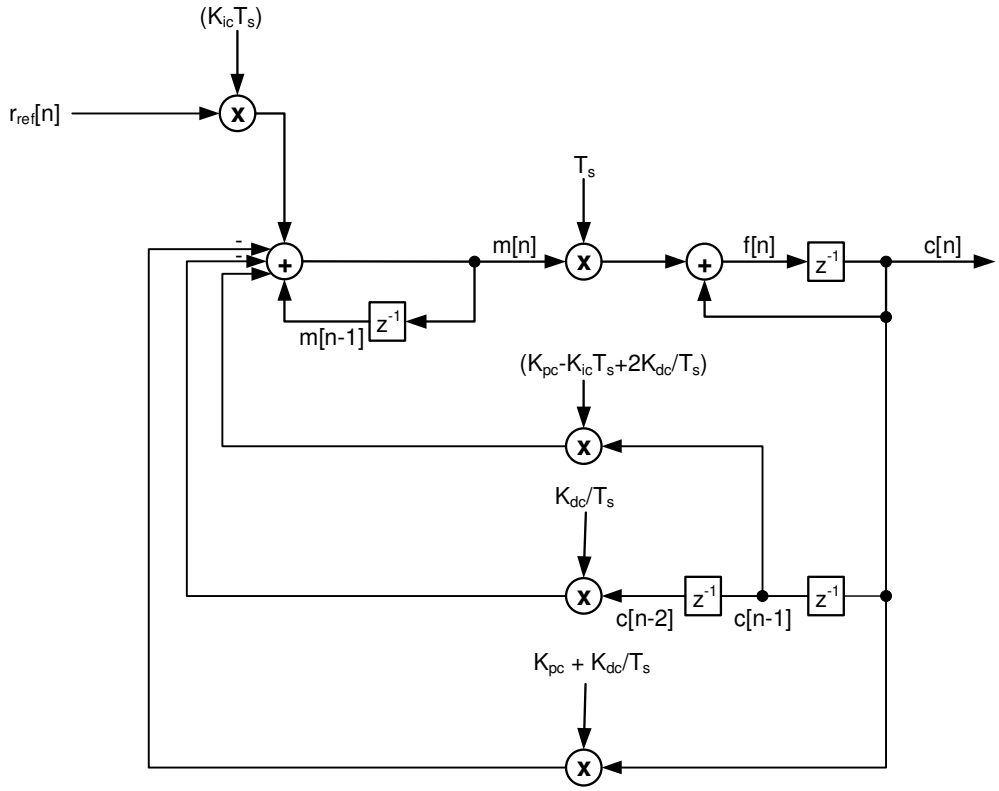


Figure 50: Implementation of system with Velocity Form PID ready for analysis in Z.

This diagram is ready to be analyzed in Z.

Starting with velocity form equation and

$$\left.\begin{aligned} M(z) &= -K_{pc}C(z) + K_{ic}T_s\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)C(z) \\ C(z) &= M(z)\frac{T_s z^{-1}}{\left(1-z^{-1}\right)} \end{aligned}\right\} \Rightarrow$$

$$C(z) = \left\{-K_{pc}C(z) + K_{ic}T_s\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - \frac{K_{dc}}{T_s}\left(1-z^{-1}\right)C(z)\right\}\frac{T_s z^{-1}}{\left(1-z^{-1}\right)} \Leftrightarrow$$

$$C(z)\left(1-z^{-1}\right) = -K_{pc}T_s z^{-1}C(z) + K_{ic}T_s^2 z^{-1}\frac{\left[R_{ref}(z) - z^{-1}C(z)\right]}{\left(1-z^{-1}\right)} - K_{dc}z^{-1}\left(1-z^{-1}\right)C(z) \Leftrightarrow$$

$$C(z)\left(1-z^{-1}\right)^2 = -K_{pc}T_s z^{-1}\left(1-z^{-1}\right)C(z) + K_{ic}T_s^2 z^{-1}R_{ref}(z) - K_{ic}T_s^2 z^{-2}C(z) - K_{dc}z^{-1}\left(1-z^{-1}\right)^2 C(z) \Leftrightarrow$$

$$\left[\left(1-z^{-1}\right)^2 + K_{pc}T_s z^{-1}\left(1-z^{-1}\right) + K_{ic}T_s^2 z^{-2} + K_{dc}z^{-1}\left(1-z^{-1}\right)^2\right]C(z) = K_{ic}T_s^2 z^{-1}R_{ref}(z) \Leftrightarrow$$

$$\left[1 + \left(K_{pc}T_s - 2 + K_{dc}\right)z^{-1} + \left(1 - K_{pc}T_s + K_{ic}T_s^2 - 2K_{dc}\right)z^{-2} + K_{dc}z^{-3}\right]C(z) = K_{ic}T_s^2 z^{-1}R_{ref}(z) \Rightarrow$$

$$\begin{aligned} c[nT_s] = {} & K_{ic}T_s^2 r_{ref}[(n-1)T_s] - \\ & (K_{pc}T_s - 2 + K_{dc})c[(n-1)T_s] - \\ & (1 - K_{pc}T_s + K_{ic}T_s^2 - 2K_{dc})c[(n-2)T_s] - \\ & K_{dc}c[(n-3)T_s] \end{aligned} \tag{5.33}$$

Equation (5.33) gives the response of the velocity-form controlled system at the sampling instants [5].

5.6.1.4 COMPARING TO FULLY ANALOG PID LOOP

An important comparison to test the quality of our implementation and provide the proof of concept is to compare it to a fully analog loop, as if our system was controlled by an analog PID controller.

As shown in section "Selecting PID Gains", the transfer function of an analogous continuous time system is

$$G_{rc}(s) = \frac{\frac{G_p(s)}{s}}{1 + \frac{G_p(s)}{s}} \tag{5.34}$$

where $G_p(s)$ is the transfer function of PID block, i.e.

$$G_p(s) = K_{pc} + K_{ic}\frac{1}{s} + sK_{dc} \tag{5.35}$$

$$G_{rc}(s) = \frac{K_{pc} + K_{ic}\frac{1}{s} + sK_{dc}}{s + K_{pc} + K_{ic}\frac{1}{s} + sK_{dc}} = \frac{K_{pc}s + K_{ic} + K_{dc}s^2}{(1 + K_{dc})s^2 + K_{pc}s + K_{ic}} \Leftrightarrow$$

$$G_{rc}(s) = \frac{\frac{K_{dc}}{(1+K_{dc})}s^2 + \frac{K_{pc}}{(1+K_{dc})}s + \frac{K_{ic}}{(1+K_{dc})}}{s^2 + \frac{K_{pc}}{(1+K_{dc})}s + \frac{K_{ic}}{(1+K_{dc})}} \tag{5.36}$$

This transfer function refers to a system with the structure shown in Figure 51.

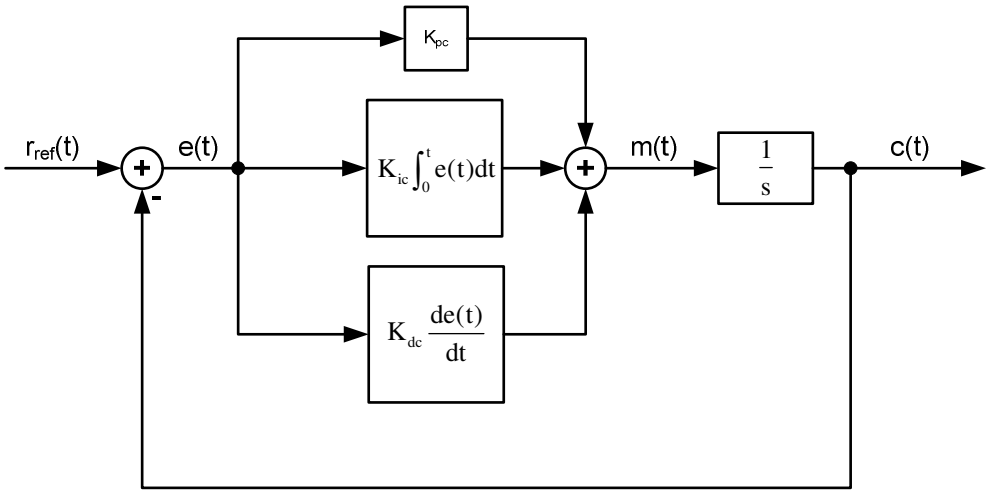


Figure 51: Continuous-time analogous to our control system core.

Comparing denominator of (5.36) to the denominator of standard 2nd order system i.e.

$$s^2 + 2\zeta\omega_n s + \omega_n^2$$

then to get a ζ and ω_n close to 1 we select $K_{pc} = 2$, $K_{ic} = 1$. Since we want to avoid the derivative term if possible (to decrease the noise it can create), for our example we will use $K_{dc} = 0$.

The results of simulating the analog system, the block implementation of the digital system and equation (5.32) are shown in figures “Figure 52: Comparison of continuous-time system to digital implementation” and “Figure 53: Comparison of equation implementation to block implementation”.

In the case of Figure 52 we see that the digital implementation gives results that are very close to those of the continuous-time system, thus we consider it a reliable approximation.

In the case of Figure 53 the result is an exact match as it must be in order to verify the accuracy of both.

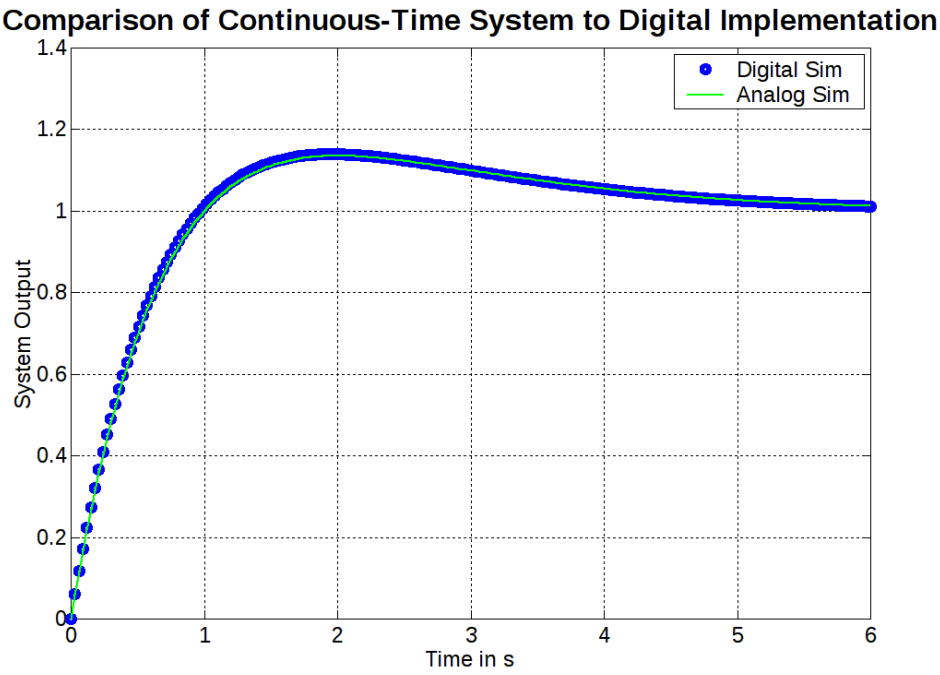


Figure 52: Comparison of continuous-time system to digital implementation

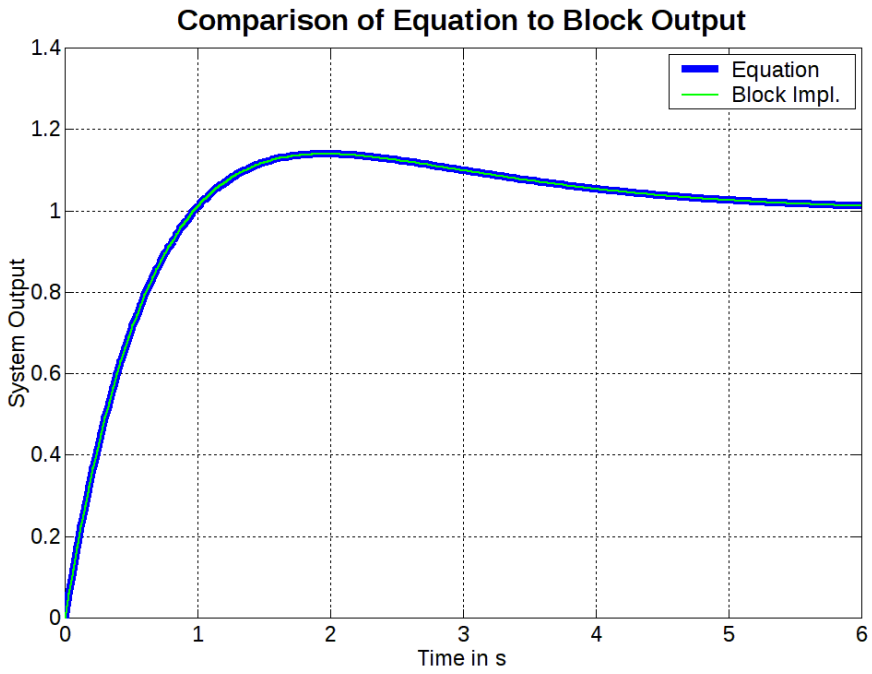


Figure 53: Comparison of equation implementation to block implementation

5.6.1.4.1 Full System Response with Velocity Form PID

Using the same PID gains with the previous section, the analysis can be repeated using the velocity form PID control.

Velocity form implementation is expected to have a more sluggish response, since there has been a mathematical manipulation that reduces the effect of input change.

The results of analyzing the system based on the block diagram of Figure 50 are presented in Figure 54.

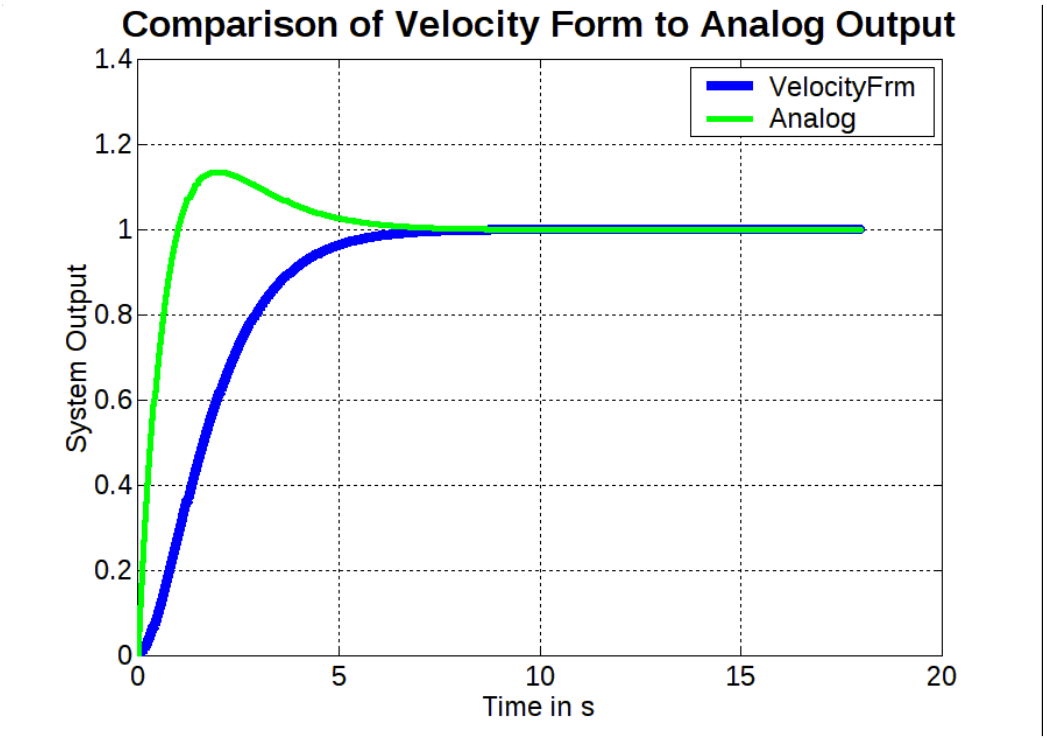


Figure 54: Velocity form PID compared to Analog PID

As expected, the response using velocity form is more sluggish compared to analog controller, though, in this particular case, suits the application better since it smooths out the overshoot.

To verify the implementation we also verified the output by comparing to the output returned using equation (5.33).

The result of the comparison is shown in Figure 55. As apparent from the figure, the results are an absolute match, which proves the equivalence of the two methods of result extraction.

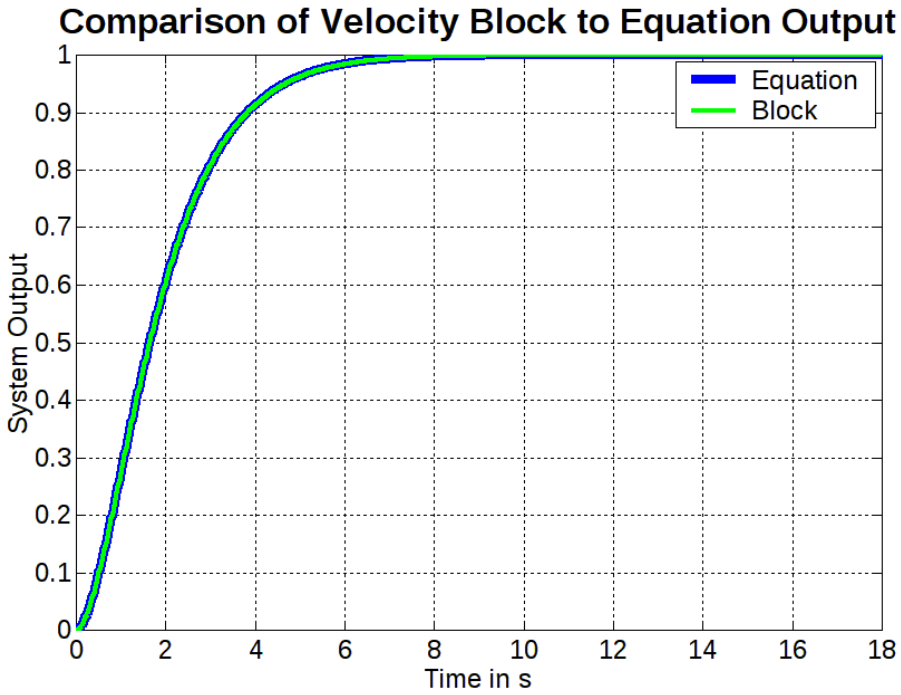


Figure 55: Verifying Velocity Form Control Implementation

5.6.2 Selecting PID Gains

As stated in section "Feedback Controller (FC)", the MBC acts as a decoupler and system inverter, thus the loops for translational velocity and angular velocity can be separated and treated as if MBC and vessel were missing. The inaccuracies that result from this approach are treated as noise added to the accelerations. Other noise sources for the accelerations are waves and wind.

The whole processing of gain selection will be carried out in continuous time and the accuracy of results will be verified by comparing the discrete time results afterwards.

Starting in continuous time, we can depict each of the two loops of our system as follows in Figure 56:

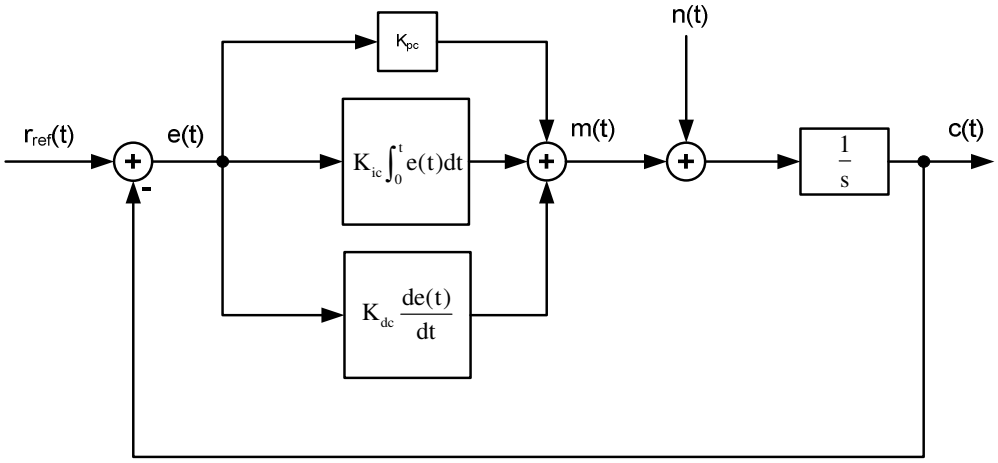


Figure 56: Continuous time representation of system loops.

We refer to two loops since there is one for vessel's translational velocity, and one for angular velocity.

At this point it must be made clear, that this type of system approximation must take into consideration system inertia. Trying, for example, to force the vessel to achieve multiple Hertz in turning will fail due to vessel inertia; therefore, the model will fail to achieve realistic approximation of the vessel. With this in mind, we proceed in the analysis of the loop. Starting with n(t) = 0:

$$E(s) = R_{ref}(s) - C(s)$$

$$C(s) = G_p(s) \cdot \frac{1}{s} E(s)$$

$$C(s) = \frac{G_p(s)}{s}\left[R_{ref}(s) - C(s)\right] \Leftrightarrow$$

$$C(s) = \frac{\frac{G_p(s)}{s}}{1 + \frac{G_p(s)}{s}} R_{ref}(s)$$

$$\boxed{G_{rc}(s) = \frac{\frac{G_p(s)}{s}}{1 + \frac{G_p(s)}{s}}} \tag{5.37}$$

In (5.37) the "rc" index implies from input "r_{ref}" to output "c".

Regarding noise: the transfer function from noise input to output is as follows:

$$C(s) = \left[N(s) + M(s)\right]\frac{1}{s} \tag{5.38}$$

$$M(s) = G_p(s) \cdot \left[-C(s)\right] \Leftrightarrow M(s) = -G_p(s)C(s) \Rightarrow$$

$$C(s) = \left[N(s) - G_p(s)C(s)\right]\frac{1}{s} \Leftrightarrow$$

$$sC(s) = N(s) - G_p(s)C(s) \Leftrightarrow$$

$$C(s) = \frac{N(s)}{s + G_p(s)} \Rightarrow$$

$$G_{nc}(s) = \frac{1}{s + G_p(s)} \tag{5.39}$$

The transfer function $G_p(s)$ of PID is [24]–[27]:

$$G_p(s) = K_{pc} + K_{ic}\frac{1}{s} + sK_{dc} \tag{5.40}$$

Thus from (5.39) and (5.40) we have that

$$G_{nc}(s) = \frac{1}{s + K_{pc} + K_{ic}\frac{1}{s} + sK_{dc}} \Leftrightarrow$$

$$G_{nc}(s) = \frac{s}{s^2 + K_{pc}s + K_{ic} + s^2K_{dc}} \Leftrightarrow$$

$$G_{nc}(s) = \frac{s}{(1 + K_{dc})s^2 + K_{pc}s + K_{ic}} \Leftrightarrow$$

$$G_{nc}(s)=\frac{\frac{s}{(1+K_{dc})}}{s^2+\frac{K_{pc}}{(1+K_{dc})}s+\frac{K_{ic}}{(1+K_{dc})}} \tag{5.41}$$

To make the equations a bit shorter symbol α_d is defined as:

$$\alpha_d=\frac{1}{(1+K_{dc})} \tag{5.42}$$

Using (5.42) equation (5.41) changes to

$$G_{nc}(s)=\frac{\alpha_d s}{s^2+\alpha_d K_{pc}s+\alpha_d K_{ic}} \Rightarrow$$

$$G_{nc}(j\omega)=\frac{\alpha_d j\omega}{-\omega^2+\alpha_d K_{pc}j\omega+\alpha_d K_{ic}} \Leftrightarrow$$

$$G_{nc}(j\omega)=\frac{\alpha_d j\omega}{\alpha_d K_{ic}-\omega^2+\alpha_d K_{pc}j\omega} \Rightarrow$$

$$|G_{nc}(j\omega)|=\frac{|\alpha_d\omega|}{\sqrt{(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2}} \tag{5.43}$$

To ease the calculation of extreme value for (5.43), the derivative of $|G_{nc}(j\omega)|^2$ will be calculated. This way, absolute value and square root will not be present.

$$|G_{nc}(j\omega)|^2=\frac{(\alpha_d\omega)^2}{(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2} \Rightarrow$$

$$\frac{d|G_{nc}(j\omega)|^2}{d\omega}=\frac{2\alpha_d{}^2\omega\left[(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2\right]-\alpha_d{}^2\omega^2\left[2(\alpha_d K_{ic}-\omega^2)(-2\omega)+2(\alpha_d K_{pc})^2\omega\right]}{\left[(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2\right]^2}=$$

$$\frac{2\alpha_d{}^2\omega\left[(\alpha_d K_{ic})^2-2\alpha_d K_{ic}\omega^2+\omega^4+(\alpha_d K_{pc})^2\omega^2\right]-\alpha_d{}^2\omega^2\left[-4\omega(\alpha_d K_{ic}-\omega^2)+2(\alpha_d K_{pc})^2\omega\right]}{\left[(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2\right]^2}=$$

$$\frac{2\alpha_d{}^4 K_{ic}{}^2\omega-4\alpha_d{}^3 K_{ic}\omega^3+2\alpha_d{}^2\omega^5+2\alpha_d{}^3 K_{pc}{}^2\omega^3+4\alpha_d{}^2\omega^3(\alpha_d K_{ic}-\omega^2)-2\alpha_d{}^3 K_{pc}{}^2\omega^3}{\left[(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2\right]^2}=$$

$$\frac{2\alpha_d{}^4 K_{ic}{}^2\omega-4\alpha_d{}^3 K_{ic}\omega^3+2\alpha_d{}^2\omega^5+2\alpha_d{}^3 K_{pc}{}^2\omega^3+4\alpha_d{}^3 K_{ic}\omega^3-4\alpha_d{}^2\omega^5-2\alpha_d{}^3 K_{pc}{}^2\omega^3}{\left[(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2\right]^2}=$$

$$\frac{2\alpha_d{}^4 K_{ic}{}^2\omega+(-4\alpha_d{}^3 K_{ic}\omega^3+2\alpha_d{}^3 K_{pc}{}^2\omega^3+4\alpha_d{}^3 K_{ic}\omega^3-2\alpha_d{}^3 K_{pc}{}^2\omega^3)-2\alpha_d{}^2\omega^5}{\left[(\alpha_d K_{ic}-\omega^2)^2+(\alpha_d K_{pc}\omega)^2\right]^2} \Leftrightarrow$$

$$\frac{d\left|G_{nc}(j\omega)\right|^2}{d\omega}=\frac{2\alpha_d{}^4K_{ic}{}^2\omega-2\alpha_d{}^2\omega^5}{\left[\left(\alpha_dK_{ic}-\omega^2\right)^2+\left(\alpha_dK_{pc}\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{nc}(j\omega)\right|^2}{d\omega}=\frac{2\alpha_d{}^2\omega\left(\alpha_d{}^2K_{ic}{}^2-\omega^4\right)}{\left[\left(\alpha_dK_{ic}-\omega^2\right)^2+\left(\alpha_dK_{pc}\omega\right)^2\right]^2} \quad (5.44)$$

Searching for the extremes, if we equate the derivative to zero, then we have:

$$\frac{d\left|G_{nc}(j\omega)\right|^2}{d\omega}=0\Rightarrow$$

$$\frac{2\alpha_d{}^2\omega\left(\alpha_d{}^2K_{ic}{}^2-\omega^4\right)}{\left[\left(\alpha_dK_{ic}-\omega^2\right)^2+\left(\alpha_dK_{pc}\omega\right)^2\right]^2}=0\Rightarrow$$

$$\omega=0 \text{ or } \omega^4=\alpha_d{}^2K_{ic}{}^2\Leftrightarrow\omega^2=\left|\alpha_dK_{ic}\right|\Leftrightarrow$$

$$\omega=\sqrt{\left|\alpha_dK_{ic}\right|} \quad (5.45)$$

Since $\alpha_d>0, K_{ic}>0$ (5.45) becomes:

$$\boxed{\omega=\sqrt{\alpha_dK_{ic}}} \quad (5.46)$$

The value of $\left|G_{nc}(j\omega)\right|$ at $\omega=\sqrt{\alpha_dK_{ic}}$ is (using (5.43) and (5.46)) :

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{\left|\alpha_d\sqrt{\alpha_dK_{ic}}\right|}{\sqrt{\left(\alpha_dK_{ic}-\left(\sqrt{\alpha_dK_{ic}}\right)^2\right)^2+\left(\alpha_dK_{pc}\sqrt{\alpha_dK_{ic}}\right)^2}}\Leftrightarrow$$

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{\left|\alpha_d\sqrt{\alpha_dK_{ic}}\right|}{\sqrt{\left(\alpha_dK_{ic}-\left|\alpha_dK_{ic}\right|\right)^2+\left(\alpha_dK_{pc}\sqrt{\alpha_dK_{ic}}\right)^2}}\overset{K_{pc}>0,K_{ic}>0,\alpha_d>0}{\Rightarrow}$$

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{\alpha_d\sqrt{\alpha_dK_{ic}}}{\alpha_dK_{pc}\sqrt{\alpha_dK_{ic}}}\Leftrightarrow$$

$$\left|G_{nc}(j\sqrt{\alpha_dK_{ic}})\right|=\frac{1}{K_{pc}}$$

Since the transfer function (5.41) has a zero at 0, and two poles at $\omega\neq0$, the curve starts inclining from zero and turns again down after the second pole (the furthest from zero). This means that the extreme we found is the maximum. Thus, the above equation is $\left\|G_{nc}\right\|_\infty$ but this will be verified also by the bode diagram of G_{nc}.

5.6.2.1 NOISE TO ERROR TRANSFER FUNCTION

Most importantly, the transfer function from noise input to error e(t) must be investigated. If we calculate E(s), we have:

Starting from E(s) calculation, considering $r_{ref}(t) = 0$ ($R_{ref}(s) = 0$).

$$E(s) = -C(s) = -\left[N(s) + M(s)\right]\frac{1}{s} \Rightarrow$$

$$E(s) = -\left[N(s) + G_p(s)E(s)\right]\frac{1}{s} \Leftrightarrow$$

$$\left[1 + \frac{G_p(s)}{s}\right]E(s) = -\frac{N(s)}{s} \Leftrightarrow$$

$$E(s) = -\frac{N(s)}{s + G_p(s)} \Leftrightarrow$$

$$G_{ne}(s) = -\frac{1}{s + G_p(s)} \tag{5.47}$$

which is a phase shifted (by π) version of (5.39). As such, the rest of the above analysis applies exactly, thus

$$G_{ne}(s) = -\frac{\frac{s}{(1+K_{dc})}}{s^2 + \frac{K_{pc}}{(1+K_{dc})}s + \frac{K_{ic}}{(1+K_{dc})}} \tag{5.48}$$

The expected extreme is again at

$$\omega = \sqrt{\alpha_d K_{ic}} \tag{5.49}$$

With the extreme value being:

$$\left|G_{ne}(j\sqrt{\alpha_d K_{ic}})\right| = \frac{1}{K_{pc}} \tag{5.50}$$

The selection of gains begins also with the educated guess that we need a slightly sluggish response from the PID controllers, to avoid large peaks at the issued accelerations to our system. For this reason, it is more probable to select a response with $\zeta \geq 1$ with reference to the standard 2nd order system (see equation (5.51)).

$$G_{2nd}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.51}$$

To implement the above statements for PID gains, we compare (5.48) to the equation for the standard 2nd order system $G_{2nd}(s)$.

Leaving aside the zero at the numerator of (5.48) for the moment, we recognize the following equations by comparing equation (5.48) to (5.51).

$$\omega_n^2 = \frac{K_{ic}}{(1+K_{dc})} \tag{5.52}$$

$$2\zeta\omega_n = \frac{K_{pc}}{(1+K_{dc})} \tag{5.53}$$

From (5.52) we have:

$$\boxed{\omega_n = \sqrt{\frac{K_{ic}}{(1+K_{dc})}}} \tag{5.54}$$

And using (5.54) along with (5.53) we have that

$$2\zeta\sqrt{\frac{K_{ic}}{(1+K_{dc})}} = \frac{K_{pc}}{(1+K_{dc})} \Leftrightarrow$$

$$\boxed{\zeta = \frac{K_{pc}}{2\sqrt{K_{ic}(1+K_{dc})}}} \tag{5.55}$$

Something noticeable and worth mentioning here is that comparing (5.49) (location of extreme) to (5.54) (ω_n equality to system gains) the result is:

$$\omega = \sqrt{\alpha_d K_{ic}} = \sqrt{\frac{K_{ic}}{1+K_{dc}}} = \omega_n$$

The above result means that the extreme is located at the natural undamped frequency of the system.

An initial set of assumptions that can be applied for the loop is:

- $\zeta \geq 1$
- K_{pc} as large as logically possible, to reduce the effect of noise, which also reduces the effect of modeling uncertainties that appear from the hypothesis that MBC acts as a perfect decoupler.
- Increasing K_{pc} comes at the expense of stability margins and this has to be taken into consideration also.
- K_{dc} to be kept small, becaue it increases the effect of noise.

The above statements can be proved by operating initially at equation (5.51) (standard 2nd order system [26], [27]) but this time with the zero at the numerator, to get the exact response:

$$G_{2nd\_s}(s) = \frac{\omega_n^2 s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.56}$$

The presence of a zero at s = 0 at the numerator implies an “AC-coupled” response, in the sense that it starts from zero at low frequencies and rises by 20dB/dec until it finds the pole or poles of the system. The – sign can be inserted along with a gain K, but for the moment we will proceed the analysis without it.

The roots of the denominator can be found as follows:

$$\Delta = (2\zeta\omega_n)^2 - 4\omega_n^2 = 4\zeta^2\omega_n^2 - 4\omega_n^2 \Leftrightarrow$$

$$\boxed{\Delta = 4\omega_n^2(\zeta^2 - 1)} \quad (5.57)$$

The following cases are analyzed:

Case 1: ζ < 0

This case is not feasible since our PID gains are positive and since ζ is given from (5.55) . Even if it was possible, it would lead to instability since the roots would be:

$$s_{1,2} = \frac{2|\zeta|\omega_n \pm \sqrt{4\omega_n^2(\zeta^2-1)}}{2} = \frac{2|\zeta|\omega_n \pm 2|\omega_n|\sqrt{\zeta^2-1}}{2}$$

Since ω_n is positive (it is a square root see (5.54)), this would lead to at least one root with positive real part, which means that this case is discarded.

Case 2: 0 < ζ < 1

In this case we have Δ < 0, which leads to two complex roots with negative real part. The roots are:

$$s_{1,2} = \frac{-2\zeta\omega_n \pm \sqrt{4\omega_n^2(\zeta^2-1)}}{2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \Rightarrow$$

$$s_{1,2} = -\zeta\omega_n \pm j\omega_d$$

where:

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \quad (5.58)$$

To find the step response of the system we work as follows:

$$Y(s) = G_{2nd\_s}(s)\cdot U(s) = \frac{\omega_n^2}{(s+\zeta\omega_n - j\omega_d)(s+\zeta\omega_n + j\omega_d)}\cdot\frac{1}{s} \Leftrightarrow$$

$$Y(s) = \frac{\omega_n^2}{(s+\zeta\omega_n - j\omega_d)(s+\zeta\omega_n + j\omega_d)} \quad (5.59)$$

Using the partial fraction expansion:

$$Y(s) = \omega_n^2\left[\frac{A}{(s+\zeta\omega_n - j\omega_d)} + \frac{B}{(s+\zeta\omega_n + j\omega_d)}\right]$$

where

$$A = \lim_{s\to(-\zeta\omega_n + j\omega_d)}\left[(s+\zeta\omega_n - j\omega_d)\frac{1}{(s+\zeta\omega_n - j\omega_d)(s+\zeta\omega_n + j\omega_d)}\right] \Leftrightarrow$$

$$A = \frac{1}{(-\zeta\omega_n + j\omega_d + \zeta\omega_n + j\omega_d)} = \frac{1}{2j\omega_d} \quad (5.60)$$

and

$$B=\lim_{s\to(-\zeta\omega_n-j\omega_d)}\left[\cancel{(s+\zeta\omega_n+j\omega_d)}\frac{1}{(s+\zeta\omega_n-j\omega_d)\cancel{(s+\zeta\omega_n+j\omega_d)}}\right]\Leftrightarrow$$

$$B=\frac{1}{(-\zeta\omega_n-j\omega_d+\zeta\omega_n-j\omega_d)}=\frac{1}{-2j\omega_d} \tag{5.61}$$

Which means that

$$Y(s)=\frac{\omega_n^2}{2j\omega_d}\left[\frac{1}{(s+\zeta\omega_n-j\omega_d)}-\frac{1}{(s+\zeta\omega_n+j\omega_d)}\right]\Rightarrow$$

$$y(t)=L^{-1}\{Y(s)\}=\frac{\omega_n^2}{2j\omega_d}\left[e^{(-\zeta\omega_n+j\omega_d)t}-e^{(-\zeta\omega_n-j\omega_d)t}\right]\Leftrightarrow$$

$$y(t)=\frac{\omega_n^2e^{-\zeta\omega_nt}}{2j\omega_d}\left[e^{j\omega_dt}-e^{-j\omega_dt}\right]=\frac{\omega_n^2e^{-\zeta\omega_nt}}{2\omega_d}\left(\frac{e^{j\omega_dt}}{j}-\frac{e^{-j\omega_dt}}{j}\right)=\frac{\omega_n^2e^{-\zeta\omega_nt}}{2\omega_d}\left(\frac{e^{j\omega_dt}}{e^{j\frac{\pi}{2}}}+\frac{e^{-j\omega_dt}}{e^{-j\frac{\pi}{2}}}\right)\Leftrightarrow$$

$$y(t)=\frac{\omega_n^2e^{-\zeta\omega_nt}}{2\omega_d}\left[e^{j\left(\omega_dt-\frac{\pi}{2}\right)}+e^{-j\left(\omega_dt-\frac{\pi}{2}\right)}\right]\Rightarrow$$

$$y(t)=\frac{\omega_n^2e^{-\zeta\omega_nt}}{\cancel{2}\omega_d}\cancel{2}\,\mathrm{Re}\left[e^{j\left(\omega_dt-\frac{\pi}{2}\right)}\right]\Rightarrow$$

$$y(t)=\frac{\omega_n^2e^{-\zeta\omega_nt}}{\omega_d}\cos\left(\omega_dt-\frac{\pi}{2}\right)\Rightarrow$$

$$\boxed{y(t)=\frac{\omega_n^2}{\omega_d}e^{-\zeta\omega_nt}\sin(\omega_dt)} \tag{5.62}$$

This is a time decaying oscillatory response and since we do not want oscillations there has to be a very good reason to use $0<\zeta<1$. Some responses using a logical approach regarding ω_d ie an ω_d that creates a sinusoidal response of 4 sec period:

$$\omega_d=\frac{2\pi\,\mathrm{rad}}{4\,\mathrm{sec}}=\frac{\pi}{2}\,\mathrm{rad/sec}$$

Which in turn means that if we select $\zeta = 0.4$, then we have an ω_n of:

$$\omega_d=\omega_n\sqrt{1-\zeta^2}\underset{\zeta=0.4}{\overset{\omega_d=\pi/2}{\Rightarrow}}\omega_n=\frac{\pi}{2\sqrt{1-0.4^2}}\Leftrightarrow$$

$$\omega_n=1.7139\ \text{rad/sec}$$

A few examples of such responses are shown in the figures Figure 57 to Figure 60.

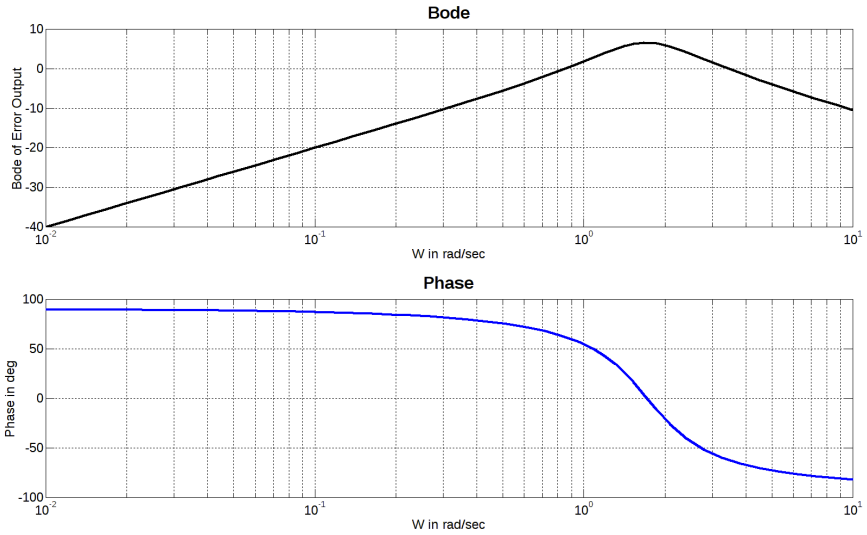


Figure 57: Bode diagram for ζ=0.4, ωn=1.7139, ωd=π/2

The first to be said about this case, is that it has a region where it amplifies the noise. Due to the fact that the real part of the complex roots is the same (complex root pair), the Bode diagram has a maximum and begins to fall after that.

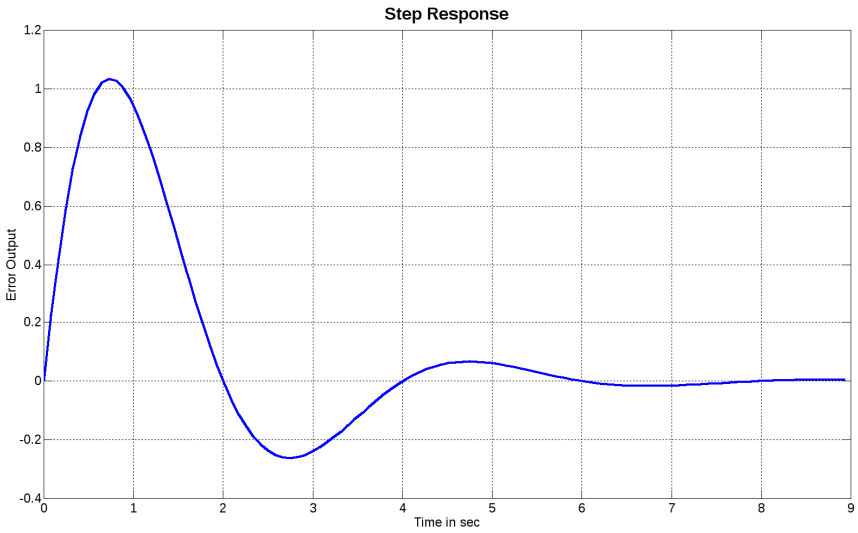


Figure 58: Step response for ζ=0.4, ωn=1.7139, ωd=π/2

The step response goes slightly above 1 as expected from Bode diagram and shows some oscillations.

As will be seen at Figure 59 and Figure 60, lowering ζ even further, will bring the system closer to instability, since the real part of complex roots approaches zero.

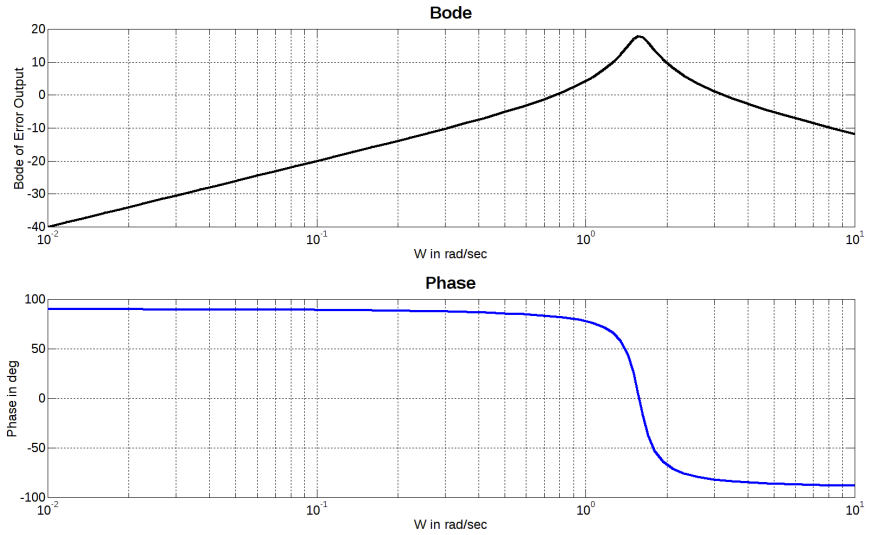


Figure 59: Bode diagram for ζ=0.1, ωn=1.7139, ωd=π/2

The amplification of noise at a certain peak is even greater and the step response suffers a lot of oscillations.

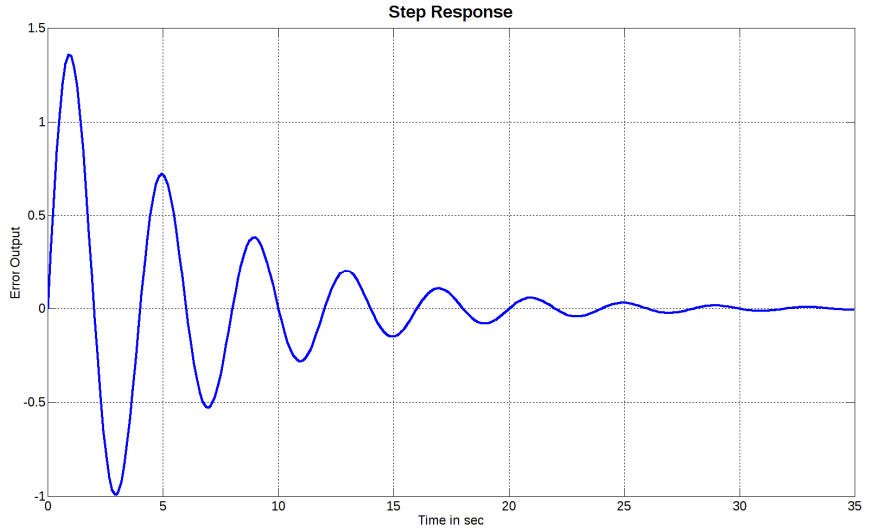


Figure 60: Step response for ζ=0.1, ωn=1.7139, ωd=π/2

Even if we decide to keep ζ below 1, we will have to be close to it.

Case 3: ζ = 1

In this case we have:

$$G_{2nd}(s) = \frac{\omega_n{}^2 s}{s^2 + 2\omega_n s + \omega_n{}^2} \Leftrightarrow G_{2nd}(s) = \frac{\omega_n{}^2 s}{(s+\omega_n)^2}$$

Which means two real roots together at the same point.

The step response of such a system can be found as follows:

$$Y(s) = G_{2nd\_s}(s) \cdot U(s) = \frac{\omega_n^{\,2} \cancel{s}}{(s+\omega_n)^2} \cdot \frac{1}{\cancel{s}} \Leftrightarrow$$

$$Y(s) = \frac{\omega_n^{\,2}}{(s+\omega_n)^2} \Rightarrow$$

$$\boxed{y(t) = \omega_n^{\,2} t e^{-\omega_n t}} \tag{5.63}$$

since from inverse Laplace transform, we have that:

$$L^{-1}\left\{\frac{n!}{(s-a)^{n+1}}\right\} = te^{at}, n = 1, 2, 3, \ldots$$

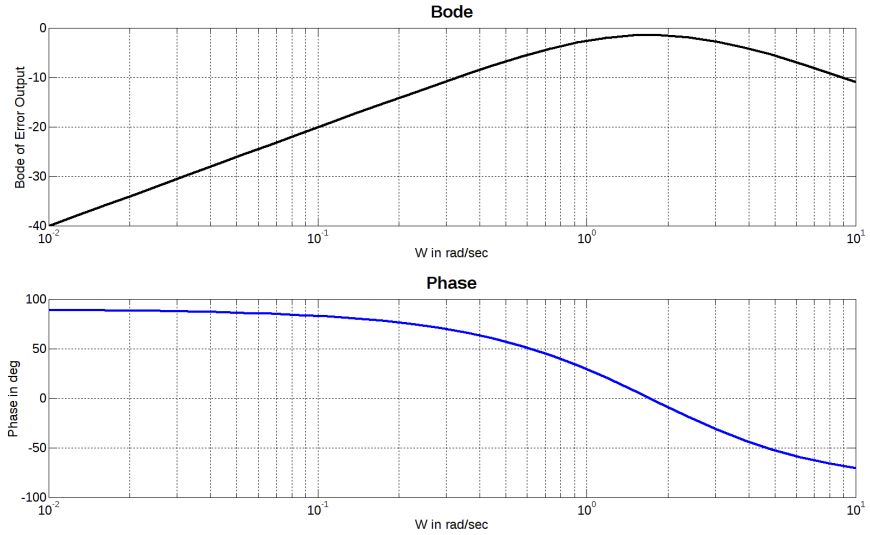


Figure 61: Bode diagram for ζ=1, ωn=1.7139

As can be seen from the Bode diagram, the noise is only attenuated by the system at ζ = 1; there is no gain at any frequency. The step response has no oscillations and zeroes at a time close to the ζ=0.4 (at about 6 sec the response is almost zero).

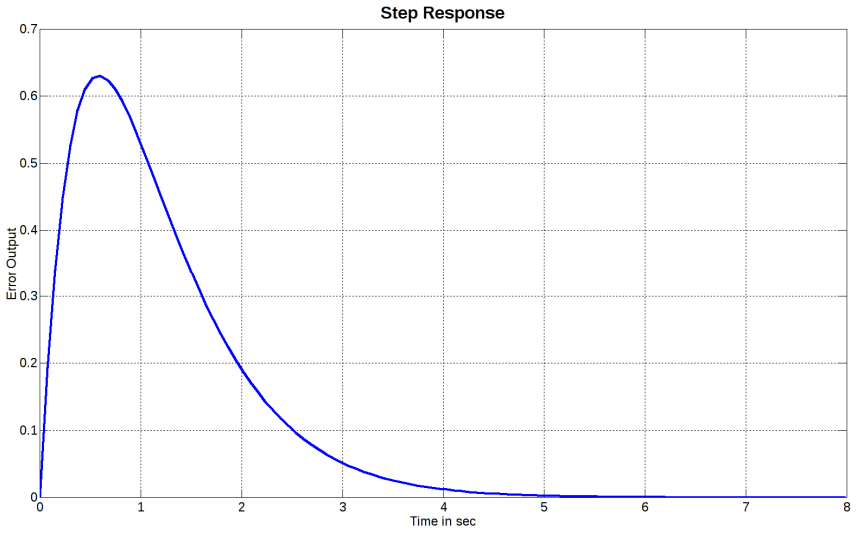


Figure 62: Step response for ζ=1, ωn=1.7139

This is an acceptable behavior which is also enhanced by the fact that the roots of the characteristic polynomial are collocated at the real axis, thus creating attenuation of -40 dB/dec right after them, which means that there is no flattening of the response at the maximum of the noise to error response.

Case 4: ζ > 1

In this case we have two real roots since:

$$\Delta = 4\omega_n^2\left(\zeta^2-1\right) > 0$$

$$s_{1,2} = \frac{-2\zeta\omega_n \pm \sqrt{4\omega_n^2\left(\zeta^2-1\right)}}{2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2-1}$$

It is interesting to investigate whether there can be found a ζ for which one root will be positive.

Suppose $s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2-1} > 0 \Leftrightarrow$

$\omega_n\sqrt{\zeta^2-1} > \zeta\omega_n \Leftrightarrow \sqrt{\zeta^2-1} > \zeta \overset{\zeta>0}{\Rightarrow} \zeta^2-1 > \zeta^2 \Leftrightarrow -1>0$ which is invalid.

Thus, we always have real negative roots. At the theoretical very large ζ, we have that

$$s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2-1} \underset{\sqrt{\zeta^2-1}\approx\zeta}{\overset{\zeta>>1}{\approx}} -\zeta\omega_n + \omega_n\zeta = 0$$

meaning that the root approaches zero.

The step response is:

$$Y(s) = G_{2nd\_s}(s)\cdot U(s) = \frac{\omega_n^2 \not{s}}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\cdot\frac{1}{\not{s}} \Rightarrow$$

$$Y(s)=\omega_n^{\,2}\left[\frac{A}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)}+\frac{B}{\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right] \text{ where}$$

$$A=\lim_{s\to\left(-\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\left[\cancel{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)}\frac{1}{\cancel{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)}\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right]\Leftrightarrow$$

$$A=\frac{1}{\cancel{-\zeta\omega_n}+\omega_n\sqrt{\zeta^2-1}+\cancel{\zeta\omega_n}+\omega_n\sqrt{\zeta^2-1}}\Leftrightarrow$$

$$A=\frac{1}{2\omega_n\sqrt{\zeta^2-1}}$$

$$B=\lim_{s\to\left(-\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)}\left[\cancel{\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\frac{1}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)\cancel{\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}}\right]\Leftrightarrow$$

$$B=\frac{1}{-2\omega_n\sqrt{\zeta^2-1}}$$

Which leads to

$$Y(s)=\frac{\omega_n^{\,\cancel{2}}}{2\cancel{\omega_n}\sqrt{\zeta^2-1}}\left[\frac{1}{\left(s+\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)}-\frac{1}{\left(s+\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)}\right]\Rightarrow$$

$$y(t)=\frac{\omega_n}{2\sqrt{\zeta^2-1}}\left[e^{\left(-\zeta\omega_n+\omega_n\sqrt{\zeta^2-1}\right)t}-e^{\left(-\zeta\omega_n-\omega_n\sqrt{\zeta^2-1}\right)t}\right]\Leftrightarrow$$

$$\boxed{y(t)=\frac{\omega_n}{2\sqrt{\zeta^2-1}}e^{-\zeta\omega_n t}\left[e^{\left(\omega_n\sqrt{\zeta^2-1}\right)t}-e^{-\left(\omega_n\sqrt{\zeta^2-1}\right)t}\right]} \qquad (5.64)$$

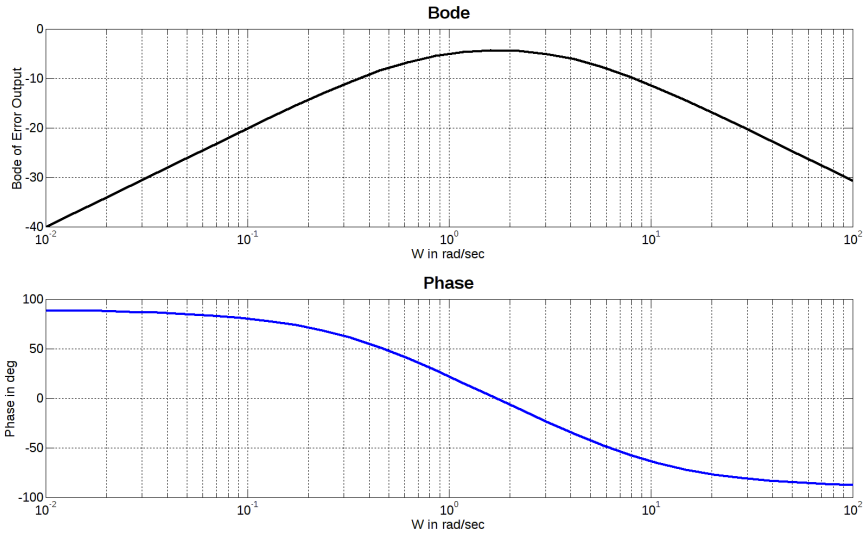


Figure 63: Bode diagram for ζ=1.4, ωn=1.7139

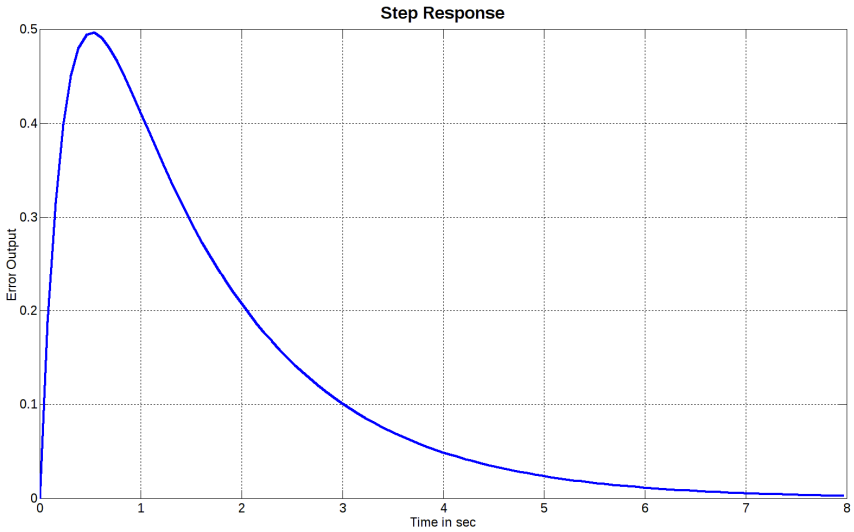


Figure 64: Step response for ζ=1.4, ωn=1.7139

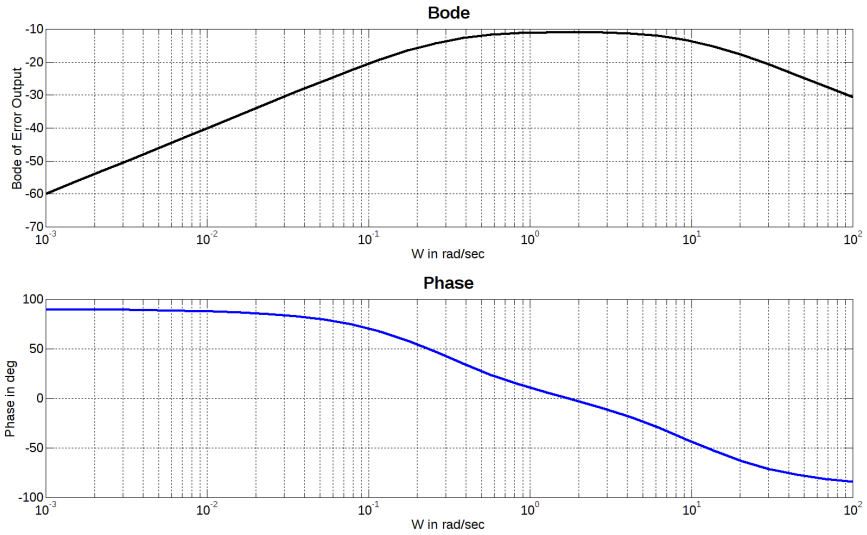


Figure 65: Bode diagram for ζ=3, ωn=1.7139

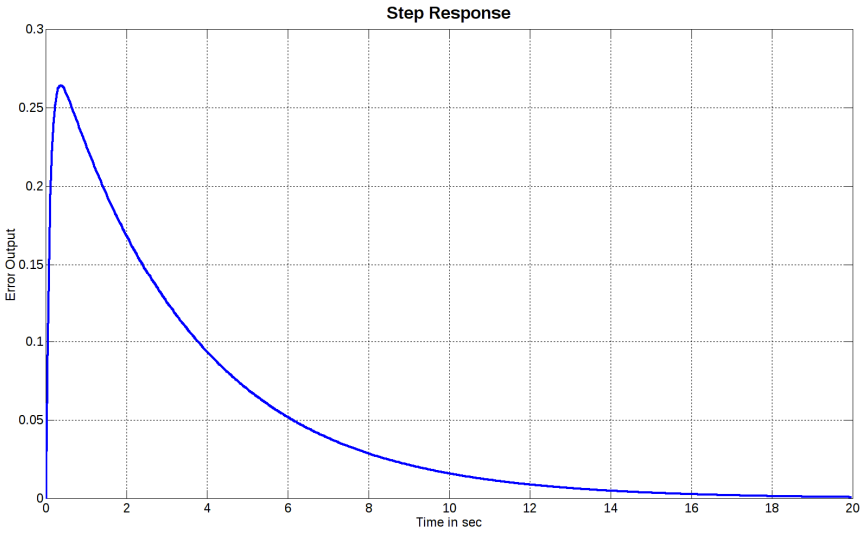


Figure 66: Step response for ζ=3, ωn=1.7139

At this point, a decision must be made regarding the recommended response, from which the PID gains will be deduced.

To make the analysis exact, we repeat here equation (5.48), along with an accurate version of (5.56) for our case:

$$G_{ne}(s) = -\frac{\frac{s}{(1+K_{dc})}}{s^2 + \frac{K_{pc}}{(1+K_{dc})}s + \frac{K_{ic}}{(1+K_{dc})}} \tag{5.65}$$

$$G_{2nd\_s1}(s) = \frac{K\omega_n{}^2 s}{s^2 + 2\zeta\omega_n s + \omega_n{}^2} \tag{5.66}$$

Where K is a negative number (K=-|K|).

$$\omega_n^{\,2} = \frac{K_{ic}}{(1+K_{dc})} \Leftrightarrow$$

$$\boxed{\omega_n = \sqrt{\frac{K_{ic}}{(1+K_{dc})}}} \tag{5.67}$$

$$2\zeta\omega_n = \frac{K_{pc}}{(1+K_{dc})} \Rightarrow 2\zeta\sqrt{\frac{K_{ic}}{(1+K_{dc})}} = \frac{K_{pc}}{(1+K_{dc})} \Leftrightarrow 2\zeta\frac{\sqrt{K_{ic}}}{\sqrt{(1+K_{dc})}} = \frac{K_{pc}}{(1+K_{dc})} \Leftrightarrow$$

$$2\zeta = \frac{K_{pc}}{\left(\sqrt{K_{ic}}\right)\cdot\sqrt{(1+K_{dc})}} \Leftrightarrow$$

$$\boxed{\zeta = \frac{K_{pc}}{2\sqrt{K_{ic}(1+K_{dc})}}} \tag{5.68}$$

$$K\omega_n^{\,2} = -\frac{1}{(1+K_{dc})} \Rightarrow K\frac{K_{ic}}{\cancel{(1+K_{dc})}} = -\frac{1}{\cancel{(1+K_{dc})}} \Leftrightarrow$$

$$\boxed{K = -\frac{1}{K_{ic}}} \tag{5.69}$$

Two cases will be investigated: one with ζ > 1 and another one with ζ = 1.

Making an exact analysis for the steady state of (5.66), we have that:

$$G_{2nd\_s1}(s) = \frac{K\omega_n^{\,2}s}{s^2+2\zeta\omega_n s+\omega_n^{\,2}} \Rightarrow G_{2nd\_s1}(j\omega) = \frac{K\omega_n^{\,2}j\omega}{(j\omega)^2+2\zeta\omega_n j\omega+\omega_n^{\,2}} \Leftrightarrow$$

$$G_{2nd\_s1}(j\omega) = \frac{K\omega_n^{\,2}j\omega}{\omega_n^{\,2}-\omega^2+2j\zeta\omega_n\omega} \tag{5.70}$$

$$\left|G_{2nd\_s1}(j\omega)\right| = \frac{\sqrt{\left(K\omega_n^{\,2}\omega\right)^2}}{\sqrt{\left(\omega_n^{\,2}-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2}} \Leftrightarrow$$

$$\left|G_{2nd\_s1}(j\omega)\right| = \frac{\left|K\omega_n^{\,2}\omega\right|}{\sqrt{\left(\omega_n^{\,2}-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2}} \tag{5.71}$$

Investigating $\left|G_{2nd\_s1}(j\omega)\right|^2$ for an extreme as we did previously, we have that

$$\left|G_{2nd\_s1}(j\omega)\right|^2 = \frac{\left(K\omega_n^{\,2}\omega\right)^2}{\left(\omega_n^{\,2}-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2} \tag{5.72}$$

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=\frac{2\left(K\omega_n{}^2\omega\right)K\omega_n{}^2\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]-\left(K\omega_n{}^2\omega\right)^2\left[2\left(\omega_n{}^2-\omega^2\right)(-2\omega)+2\left(2\zeta\omega_n\omega\right)2\zeta\omega_n\right]}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}=$$

$$=\frac{2K^2\omega_n{}^4\omega\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]-\left(K\omega_n{}^2\omega\right)^2\left[2\left(\omega_n{}^2-\omega^2\right)(-2\omega)+2\left(2\zeta\omega_n\omega\right)2\zeta\omega_n\right]}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}=$$

$$=\frac{2K^2\omega_n{}^4\omega\left(\omega_n{}^4-2\omega_n{}^2\omega^2+\omega^4+4\zeta^2\omega_n{}^2\omega^2\right)-K^2\omega_n{}^4\omega^2\left[-4\omega_n{}^2\omega+4\omega^3+8\zeta^2\omega_n{}^2\omega\right]}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}=$$

$$=\frac{2K^2\omega_n{}^8\omega-\cancel{4K^2\omega_n{}^6\omega^3}+2K^2\omega_n{}^4\omega^5+\cancel{8K^2\zeta^2\omega_n{}^6\omega^3}+\cancel{4K^2\omega_n{}^6\omega^3}-4K^2\omega_n{}^4\omega^5-\cancel{8K^2\zeta^2\omega_n{}^6\omega^3}}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=\frac{2K^2\omega_n{}^8\omega-2K^2\omega_n{}^4\omega^5}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2}\Leftrightarrow$$

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=\frac{2K^2\omega_n{}^4\omega\left(\omega_n{}^4-\omega^4\right)}{\left[\left(\omega_n{}^2-\omega^2\right)^2+\left(2\zeta\omega_n\omega\right)^2\right]^2} \tag{5.73}$$

To find the extreme (which we know beforehand that it will be a maximum), we have

$$\frac{d\left|G_{2nd\_sl}(j\omega)\right|^2}{d\omega}=0\Rightarrow$$

$$2K^2\omega_n{}^4\omega\left(\omega_n{}^4-\omega^4\right)=0\Rightarrow$$

$$\omega=0 \text{ or } \omega_n{}^4=\omega^4\Rightarrow$$

$$\boxed{\omega_{max}=\omega_n} \tag{5.74}$$

since we investigate for positive ω. The maximum value is [28]

$$\left\|G_{2nd\_sl}\right\|_\infty=\frac{\left|K\omega_n{}^3\right|}{\sqrt{\left(2\zeta\omega_n{}^2\right)^2}}=\frac{\left|K\omega_n{}^3\right|}{\left|2\zeta\omega_n{}^2\right|}=\frac{|K|\omega_n{}^3}{2\zeta\omega_n{}^2}\Leftrightarrow$$

$$\boxed{\left\|G_{2nd\_sl}\right\|_\infty=\frac{|K|\omega_n}{2\zeta}} \tag{5.75}$$

Looking at (5.75) infinity norm [29], [30], [31] we deduce that in order to suppress the noise at our system, we should aim for small ω_n and large ζ [28], but everything comes at a cost. Lowering very much ω_n will make the system very sluggish and the same happens with large ζ. At the same time, since the roots of the characteristic polynomial are $s_{1,2}=-\zeta\omega_n\pm\omega_n\sqrt{\zeta^2-1}$ for $\zeta\geq 1$, we see that we compromise relative stability by reducing ω_n AND by increasing ζ!

5.6.3 Case ζ = 1 (for PID Supervisory Controller)

After fixing ζ to 1, and after some experimentation with ω_n is set to $\omega_n = 2$. The transfer function from noise to error that results from this choice is:

$$G_{ne(\zeta=1,\omega_n=2)}(s) = \frac{K\omega_n{}^2 s}{s^2 + 2\zeta\omega_n s + \omega_n{}^2} \Rightarrow$$

$$G_{ne(\zeta=1,\omega_n=2)}(s) = -\frac{s}{s^2 + 4s + 4} \tag{5.76}$$

These selections give the following frequency and step responses:

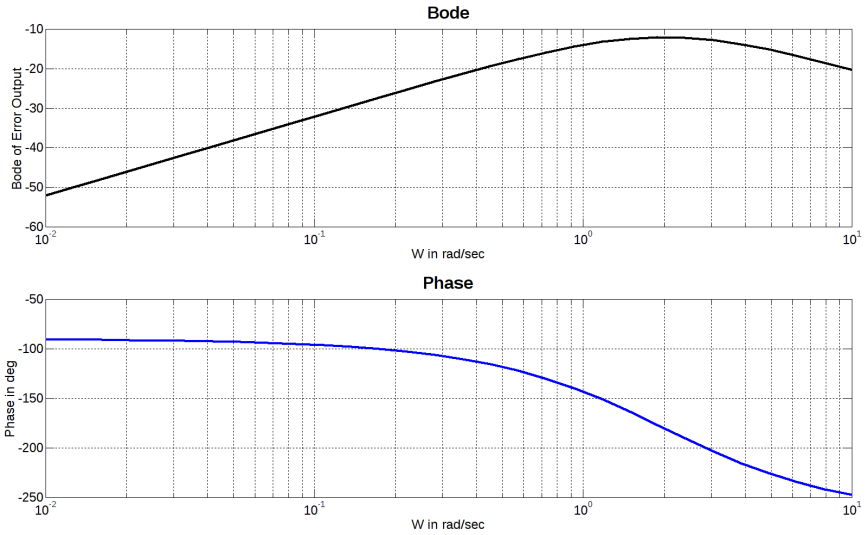


Figure 67: Noise to error transfer function Bode diagram for ζ = 1, ωn = 2 rad/sec.

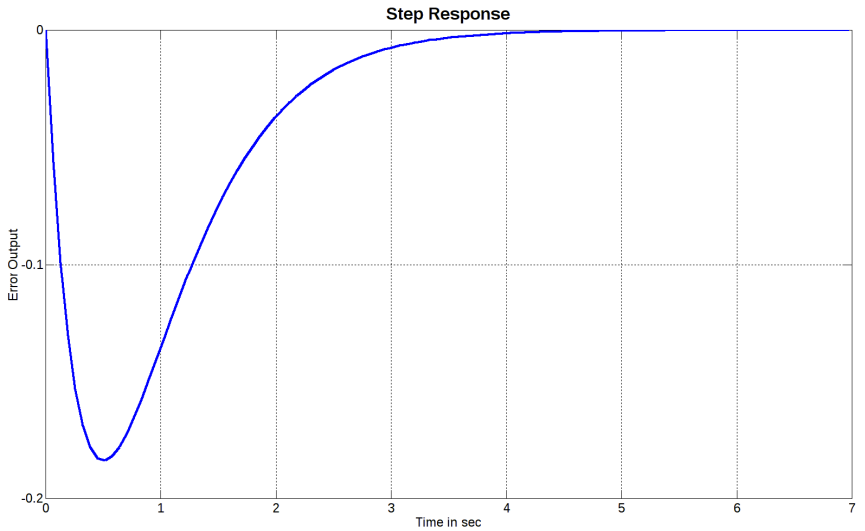


Figure 68: Noise to error step response for ζ = 1, ωn = 2 rad/sec.

As expected from (5.74) the maximum gain of the transfer function from noise to error, is at $\omega = \omega_n = 2\,rad/\sec$. To verify regarding the peak value of the transfer function gain we must first clarify all the constants used.

It is preferable for our application to avoid the derivative term, if possible. This means that we start with:

$K_{dc} = 0$

Using (5.54) with $K_{dc} = 0$ we have that:

$K_{ic} = \omega_n^{\,2} = 4$

Using (5.55) we get:

$K_{pc} = 2\zeta\omega_n = 4$

Finally, using (5.69) we have that

$$K = -\frac{1}{K_{ic}} = -\frac{1}{4}$$

Thus from (5.75) we have that:

$$\left\| G_{ne(\zeta=1,\omega_n=2)} \right\|_\infty = \frac{|K|\omega_n}{2\zeta} = \frac{\frac{1}{4}x2}{2} = \frac{1}{4} \Rightarrow$$

$$\left\| G_{ne(\zeta=1,\omega_n=2)} \right\|_\infty = 20\log\left(\frac{1}{4}\right) \simeq -12\text{dB}$$

Something that can be seen from the figures above.

With these settings for PID gains, the transfer function from reference input to output (G_{rc}) becomes (refer to (5.36)):

$$G_{rc(\zeta=1,\omega_n=2)}(s) = \frac{4s+4}{s^2+4s+4}$$

For this transfer function we have the Bode and step response diagrams of Figure 69 and Figure 70.

Implementing and simulating $G_{rc(\zeta=1,\omega_n=2)}$ in the digital domain, we have for the digital implementations presented in this text (i.e., positional form and velocity form):

$$C_{pform1}(z) = \frac{0.12z^{-1} - 0.1164z^{-2}}{1 - 1.88z^{-1} + 0.8836z^{-2}} R_{ref}(z) \tag{5.77}$$

In Z-transform, which translates to

$$\begin{aligned} c_{pform1}[nT_s] = {} & 0.12r_{ref}[(n-1)T_s] - 0.1164r_{ref}[(n-2)T_s] + \\ & 1.88c[(n-1)T_s] - 0.8836c[(n-2)T_s] \end{aligned} \tag{5.78}$$

in the discrete time domain. For the derivation of these equations, we used (5.32).

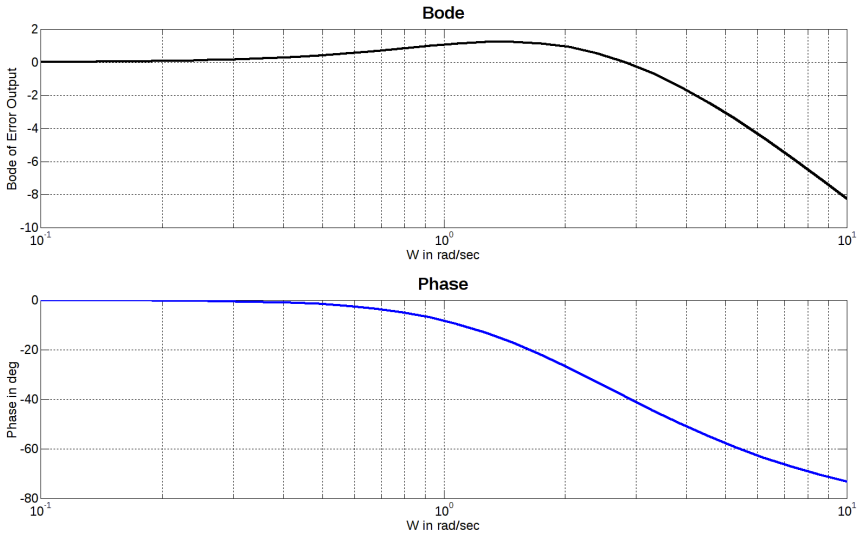


Figure 69: Reference to output transfer function Bode diagram for ζ = 1, ωn = 2 rad/sec.

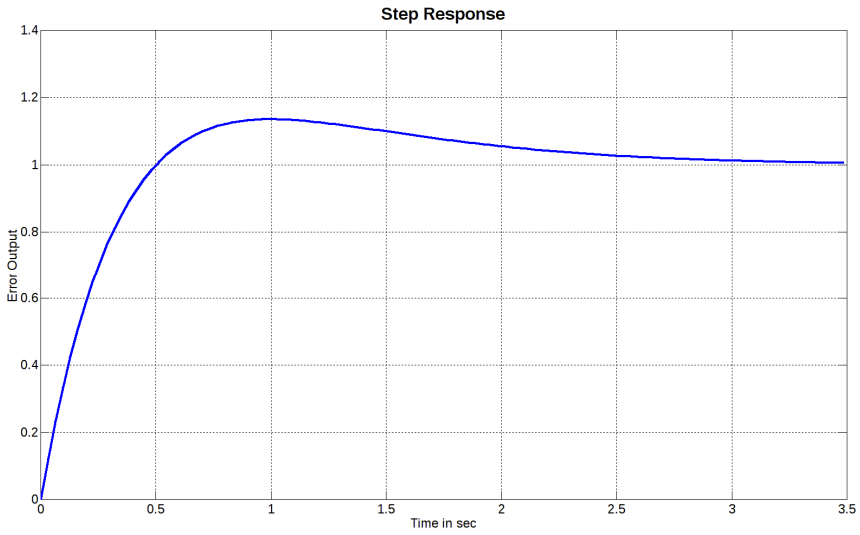


Figure 70: Reference to output transfer function step response for ζ = 1, ωn = 2 rad/sec.

The response of the positional form implementation of the system of equation (5.78) follows in Figure 71. The digital output is presented in comparison to the analog response (green line). The two responses are almost the same; though they both suffer from a slight overshoot, something that will be addressed by the use of the velocity form implementation.

As already mentioned, the velocity form implementations present a more filtered response compared to the respective positional form implementations.

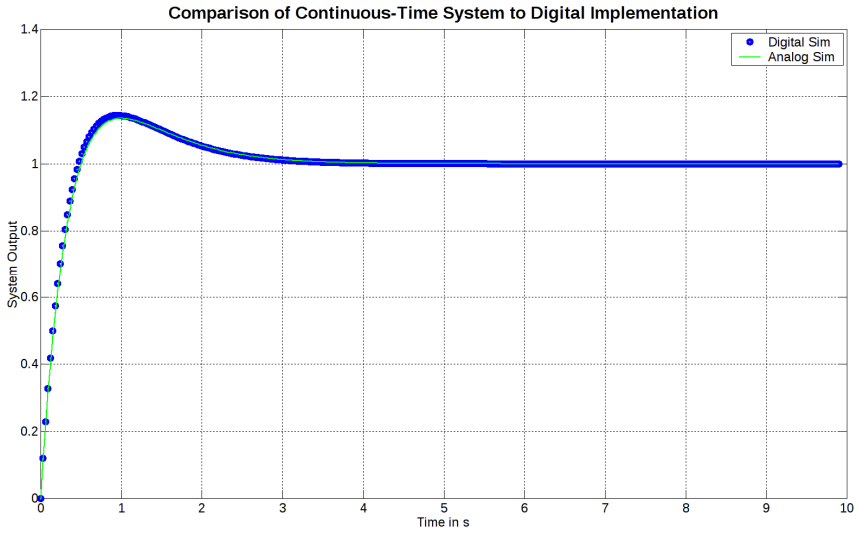


Figure 71: Response of the digital positional form implementation of the system with ζ = 1, ωn = 2 rad/sec.

The velocity form implementation uses equations:

$$C_{vform1}(z) = \frac{0.0036z^{-1}}{1-1.88z^{-1}+0.8836z^{-2}} R_{ref}(z) \tag{5.79}$$

or in discrete time form (based on (5.33):

$$\begin{aligned} c_{vform1}[nT_s] = 0.0036 r_{ref}[(n-1)T_s] + \\ 1.88c[(n-1)T_s] - 0.8836c[(n-2)T_s] \end{aligned} \tag{5.80}$$

The above equations result to the response of Figure 72.

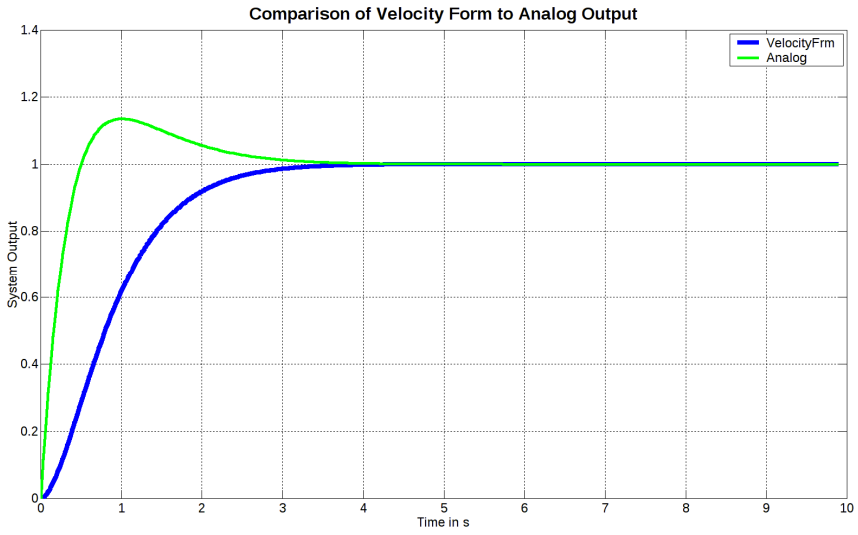


Figure 72: Response of the digital velocity form implementation of the system with ζ = 1, ωn = 2 rad/sec.

As can be clearly deduced from this response, the overshoot has been diminished. Thus, this type of implementation will be used for the PID supervisory controller.

5.6.4 Case ζ = 10 (for MBC)

To quickly adapt to supervisory controller commands, MBC needs fast response, compared to SC, with no overshoot. For this reason, the selection for MBC PID loops has settled to ζ = 10, and $\omega_n = 0.2$. The transfer function from noise to error that results from this choice is:

$$G_{ne(\zeta=10,\omega_n=0.2)}(s) = \frac{K\omega_n^2 s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \Rightarrow$$

$$G_{ne(\zeta=10,\omega_n=0.2)}(s) = -\frac{s}{s^2 + 4s + 0.04} \quad (5.81)$$

These selections give the following frequency and step responses:

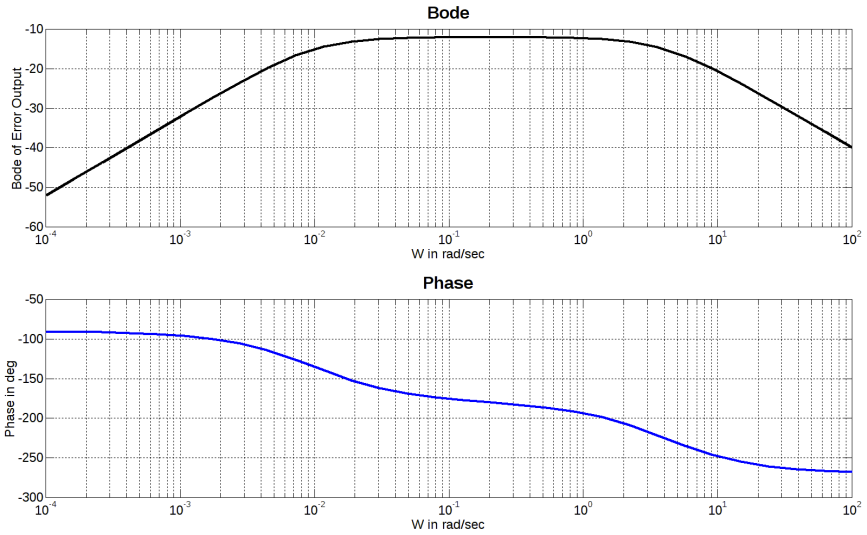


Figure 73: Noise to error transfer function Bode diagram for ζ = 10, ωn = 0.2 rad/sec.

Since ζ > 1, we have two poles on the real axis, which result into a flatness at the peak of the gain graph, until the second pole is found. After the second pole frequency, the gain starts falling, something we exploit to reduce the noise that results from the MBC controller approximation to our setpoint. This noise is expected to be of high frequency compared to the pole frequencies, since the change in the error will be once per sampling period T_s.

Repeating similar analysis to paragraph 5.6.3 we have that we prefer

$K_{dc} = 0$

Using (5.54) with $K_{dc} = 0$ we have that:

$K_{ic} = \omega_n^2 = 0.04$

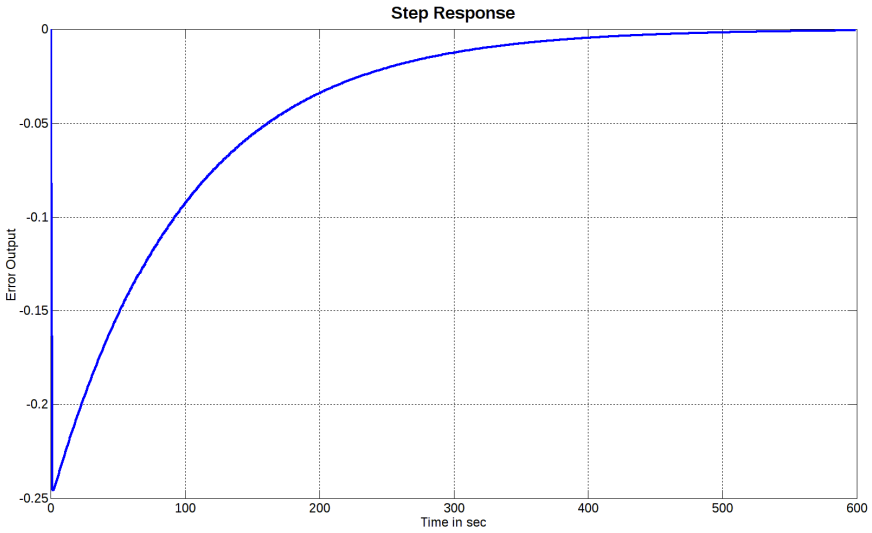


Figure 74: Noise to error transfer function step response for ζ = 10, ωn = 0.2 rad/sec.

Using (5.55) we get:

$$K_{pc} = 2\zeta\omega_n = 4$$

Using (5.69) we have that

$$K = -\frac{1}{K_{ic}} = -\frac{1}{0.04}$$

From (5.75) we have that:

$$\left\| G_{ne(\zeta=10,\omega_n=0.2)} \right\|_\infty = \frac{|K|\omega_n}{2\zeta} = \frac{\frac{1}{0.04}\text{x}0.2}{2\text{x}10} = \frac{1}{4} \Rightarrow$$

$$\left\| G_{ne(\zeta=10,\omega_n=0.2)} \right\|_\infty = 20\log\left(\frac{1}{4}\right) \simeq -12\text{dB}$$

Something which is also verified from the MATLAB resulted figures above.

With these settings for PID gains, the transfer function from reference input to output (G_{rc}) becomes (refer to (5.36)):

$$G_{rc(\zeta=10,\omega_n=0.2)}(s) = \frac{4s+0.04}{s^2+4s+0.04} \qquad (5.82)$$

For this transfer function we have the Bode and step response diagrams of Figure 75 and Figure 76.

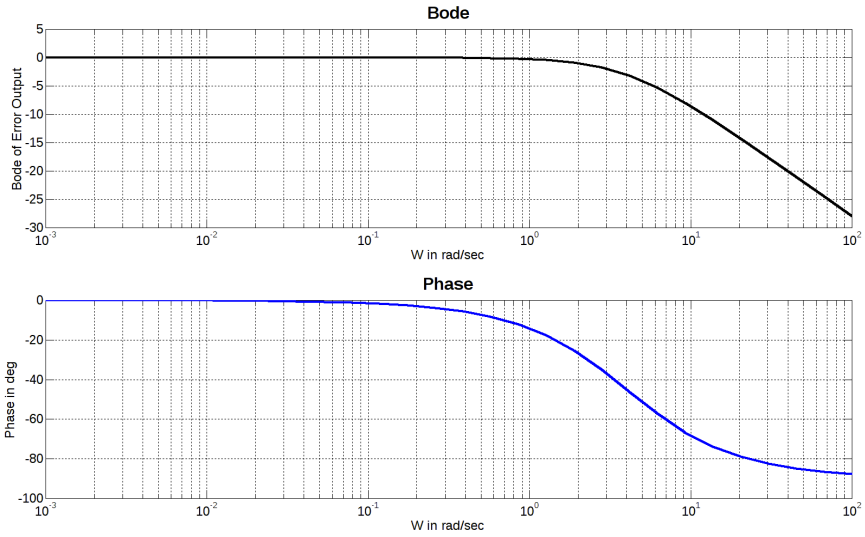


Figure 75: Reference to output transfer function Bode diagram for ζ = 10, ωn = 0.2 rad/sec.

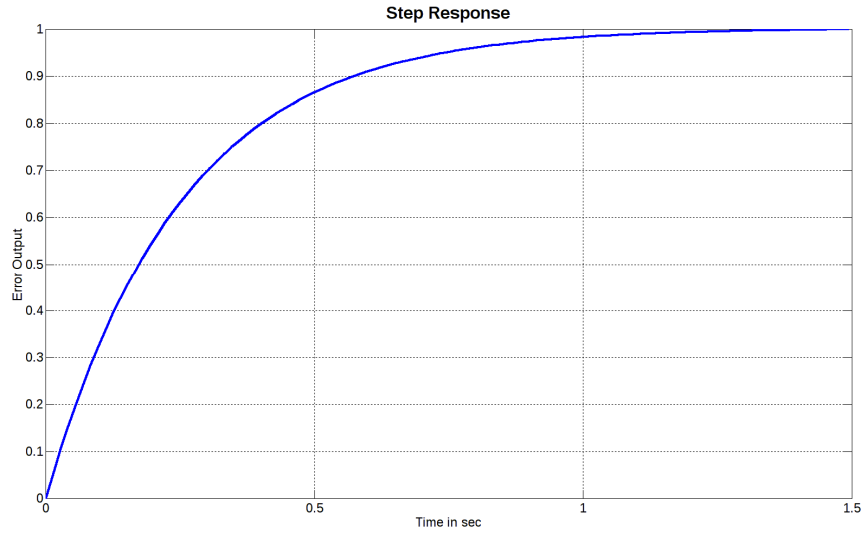


Figure 76: Reference to output transfer function step response for ζ = 10, ωn = 0.2 rad/sec.

Implementing and simulating $G_{rc(\zeta=10,\omega_n=0.2)}$ in the digital domain, we have for the digital implementations presented in this text (i.e., positional form and velocity form):

$$C_{pform2}(z)=\frac{0.12z^{-1}-0.12z^{-2}}{1-1.88z^{-1}+0.88z^{-2}}R_{ref}(z) \tag{5.83}$$

In Z-transform, which translates to

$$\begin{aligned} c_{pform2}[nT_s]=0.12r_{ref}[(n-1)T_s]-0.12r_{ref}[(n-2)T_s]+ \\ 1.88c[(n-1)T_s]-0.88c[(n-2)T_s] \end{aligned} \tag{5.84}$$

in the discrete time domain. For the derivation of these equations, we used (5.32).

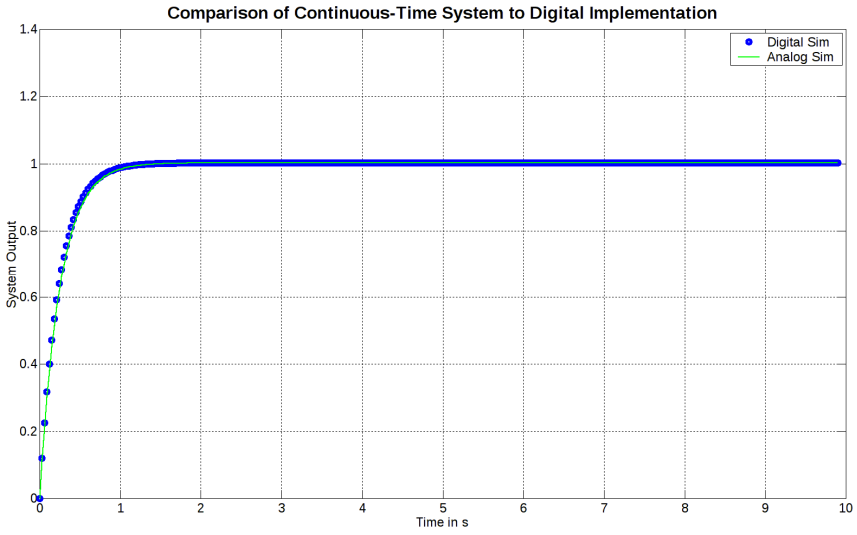


Figure 77: Response of the digital positional form implementation of the system with ζ = 10, ωn = 0.2

Figure 77 shows the response of (5.84). It is a fast response, very close to what can be achieved by the vessel, and about four times faster than (5.80), which is something we want in order to have the internal loop faster than the supervisory; this is needed for the stability of the system.

On the other side, the velocity form implementation using Kpc = 4, Kic = 0.04 is described by the following equations ((5.85) & (5.86)). The response of these equations is shown in Figure 78.

$$C_{vform2}(z) = \frac{3.6 \cdot 10^{-5} z^{-1}}{1 - 1.88 z^{-1} + 0.88 z^{-2}} R_{ref}(z) \tag{5.85}$$

or in discrete time form (based on (5.33):

$$\begin{aligned} c_{vform2}[nT_s] = 3.6 \cdot 10^{-5} r_{ref}[(n-1)T_s] + \\ 1.88c[(n-1)T_s] - 0.88c[(n-2)T_s] \end{aligned} \tag{5.86}$$

This time the multiplication constant for the reference input is extremely small, resulting into an unacceptably slow step response. Thus, for this case velocity form is not applicable.

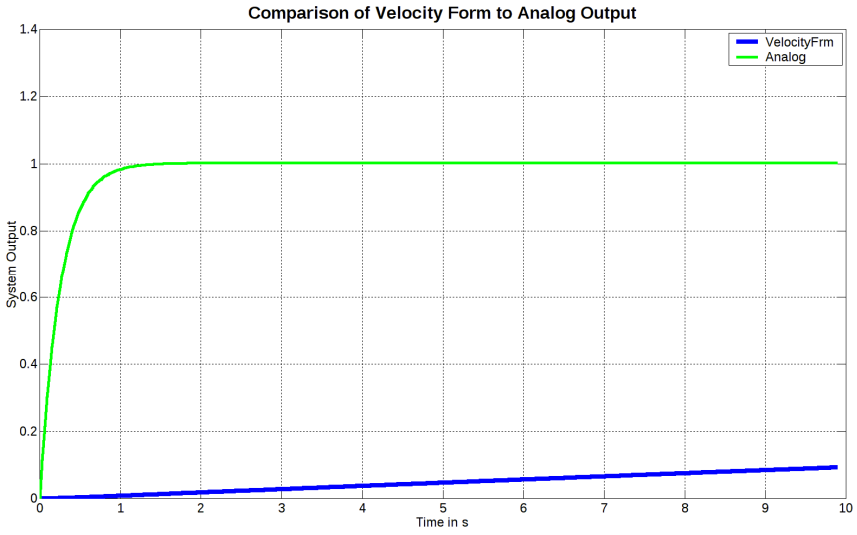


Figure 78: Response of the digital velocity form implementation of the system with ζ = 10, ωn = 0.2

5.7 Supervisory Controller (SC)

Supervisory controller is basically a set of two mathematical equations that force the system to zero the distance and angle from its target. Table 7 lists the symbols used in this paragraph and helps in understanding the ideas used:

Symbol	Description
SC	Supervisory Controller.
V_s	The requested translational velocity from supervisory controller. The control scheme that follows tries to force the system to achieve this request.
ω_s	The requested rate of turn from supervisory controller. The control scheme that follows tries to force the system to achieve this request.
$\vec{r}$	The vector connecting the vessel center of mass to the target.
$\|\vec{r}\|$	The magnitude of $\vec{r}$ which equals the distance from the target.
φ	The angle between the axis parallel to vessel's course and Ox_0y_0 or $O'x_Ry_R$ axes. Positive counter-clockwise.
φ_r	The angle between $\vec{r}$ and Ox_0y_0 or $O'x_Ry_R$ axes. Positive counter-clockwise.
θ	The angle between the axis parallel to vessel's course and $\vec{r}$. Positive counter-clockwise.
R_s	Distance between vessel and target from which the velocity requests of SC change method of calculation.

Table 7: Symbols related to Supervisory Controllers.

The outputs of SC are v_s and ω_s, which are the translational velocity and angular velocity respectively, that are ordered to the rest of the system.

As input it takes the vector $\vec{r}$ (refer to "Figure 22: Axes systems." or Figure 79 which is the same scaled down) which is the vector that connects the vessel's center of mass to the target point at the body-fixed axes O'xy. From this vector, distance from target ($\|\vec{r}\|$) and angle θ is deduced [30].

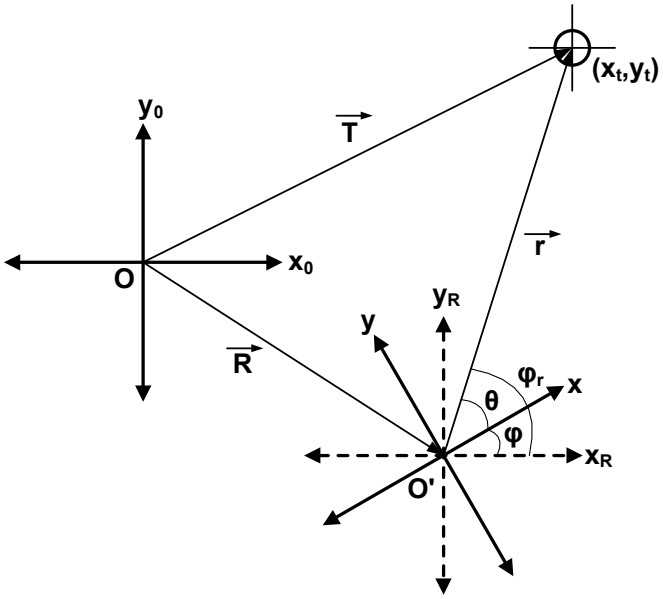


Figure 79: Repetition of axes systems figure.

To elaborate regarding θ angle, we have to refer also to axes $O'x_Ry_R$. These axes remain fixed, regarding their angle, parallel to reference system Ox_0y_0, but they follow the vessel's center of mass. When ω_s is mentioned, which is one of the outputs of supervisory controller, the target rate of change of angle φ is ment.

Defining θ:

$$\boxed{\theta = \varphi_r - \varphi} \tag{5.87}$$

If we consider φ_r as constant (for relatively long distances this is true), then we have

$$\frac{d\theta}{dt} = \dot{\theta} = \frac{d\varphi_r}{dt}^{0} - \frac{d\varphi}{dt} \Leftrightarrow$$

$$\boxed{\dot{\theta} = -\dot{\varphi} = -\omega_s} \tag{5.88}$$

5.7.1 Equations Generating Outputs

Velocity Calculations

For the generation of v_s (translational velocity setpoint) the equations set used is:

$$\begin{aligned} &\text{if } \left(\left\|\vec{r}\right\| > R_s\right) \text{then} \\ &v_s = v_{mset} \\ &\text{else} \\ &v_s = v_{mset} \cdot \left(\frac{\left\|\vec{r}\right\|}{R_s}\right)^2 \end{aligned} \tag{5.89}$$

where R_s is distance between vessel and target which selects the method for velocity calculation.

To prove that equations set (5.89) leads to zero distance we investigate the case where angle θ is zero. This is not an arbitrary decision, since as it will be proved in the "Angular Velocity Calculations" section, angle θ will eventually and stably become zero.

We first have to notice that when θ = 0, then

$$v_s = -\frac{d\left\|\vec{r}\right\|}{dt} \tag{5.90}$$

Ivestigating the case where $\left\|\vec{r}\right\| < R_s$ we have (using (5.89) and (5.90)) that

$$-\frac{d\left\|\vec{r}\right\|}{dt} = v_{mset} \cdot \left(\frac{\left\|\vec{r}\right\|}{R_s}\right)^2 \Leftrightarrow$$

$$\frac{d\left\|\vec{r}\right\|}{dt} = -\frac{v_{mset}}{R_s^2}\left\|\vec{r}\right\|^2 \Leftrightarrow$$

$$\frac{d\left\|\vec{r}\right\|}{\left\|\vec{r}\right\|^2} = -\frac{v_{mset}}{R_s^2}dt \Rightarrow$$

$$\int\frac{d\left\|\vec{r}\right\|}{\left\|\vec{r}\right\|^2} = -\int\frac{v_{mset}}{R_s^2}dt \Rightarrow$$

$$-\frac{1}{\left\|\vec{r}\right\|} + c_1 = -\frac{v_{mset}}{R_s^2}t \tag{5.91}$$

To ease writing we assign symbol r(t) as a substitute of $\left\|\vec{r}(t)\right\|$ for the case we investigate (no change of angle θ and θ = 0). Then equation (5.91) becomes:

$$-\frac{1}{r(t)} + c_1 = -\frac{v_{mset}}{R_s^2}t$$

To find constant c_1 we assign t = 0 and we have that

$$c_1 = \frac{1}{r(0)} \equiv \frac{1}{r_0}$$

Thus (5.91) becomes

$$-\frac{1}{r(t)} = -\frac{1}{r_0} - \frac{v_{mset}}{R_s^2} t \Leftrightarrow$$

$$r(t) = \frac{1}{\frac{1}{r_0} + \frac{v_{mset}}{R_s^2} t} \Leftrightarrow$$

$$\boxed{r(t) = \frac{r_0}{1 + \frac{v_{mset}}{R_s^2} r_0 t}} \quad (5.92)$$

An example respone for the above function with R_s = 20m, r_0 = 20m is shown in Figure 80.

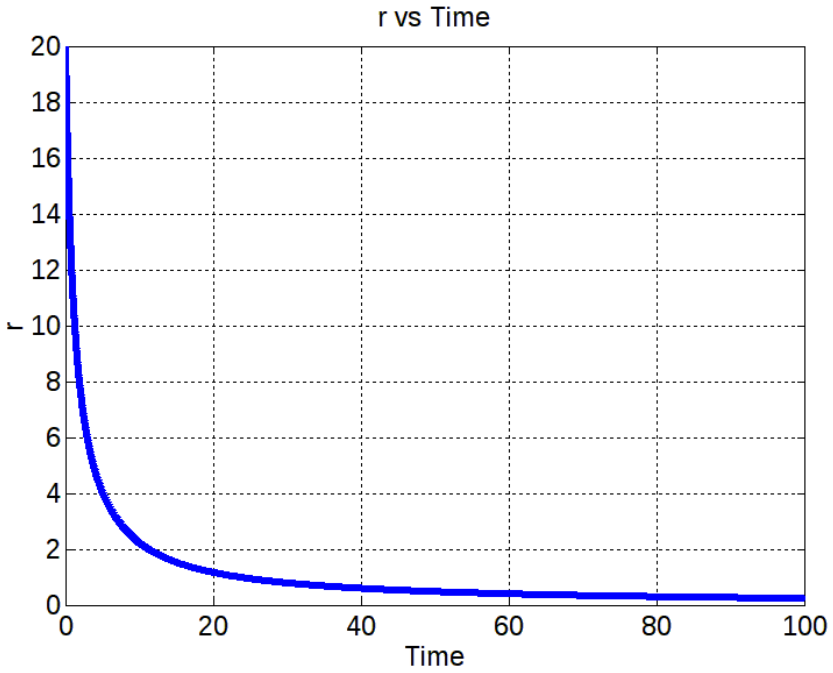


Figure 80: Distance versus time as requested by SC.

Two notes on Figure 80:

- This is not the actual response of the vessel; it is the response that would result if the vessel could follow instantly the commands from SC.
- At first the response might seem sluggish; though it must be noted that "vessel reaching the target" does not necessarily mean that distance is zero. It is an arbitrary selection. Thus, if for example we define that "reaching the target" means vessel within 2m from the target, then the actual time needed for this reduces dramatically.

Just to finalize the section, the limit as t → +∞ should be taken on equation (5.92), to prove that the final distance will be zero.

$$\lim_{t\to\infty}\left[r(t)\right]=\lim_{t\to\infty}\left(\frac{r_0}{1+\frac{v_{mset}}{R_s^{\,2}}r_0 t}\right)=0$$

This proves that the vessel will finally have zero distance from target.

Angular Velocity Calculations

For the generation of the angular velocity setpoint ω_s, first a function $f_s(\theta)$ is defined, where

$$f_s(\theta)\equiv \text{sign}(\theta)\cdot(1-\cos\theta) \tag{5.93}$$

$\theta\in(-\pi,\pi]$ which implies that since $-1\le\cos\theta\le 1$, for $f_s(\theta)$ we have $-2\le f_s(\theta)\le 2$.

Returning to the function for ω_s

$$\omega_s=\frac{\omega_{mset}}{2}\cdot f_s(\theta) \tag{5.94}$$

Noticing that $\omega_s=-\dot{\theta}$ and replacing $f_s(\theta)$ with (5.93),

$$\boxed{\dot{\theta}=-\frac{\omega_{mset}}{2}\cdot\text{sign}(\theta)\cdot(1-\cos\theta)} \tag{5.95}$$

This is a one-dimensional first-order non-linear system [32], [33], for which if we create its flow on the line, we have Figure 81:

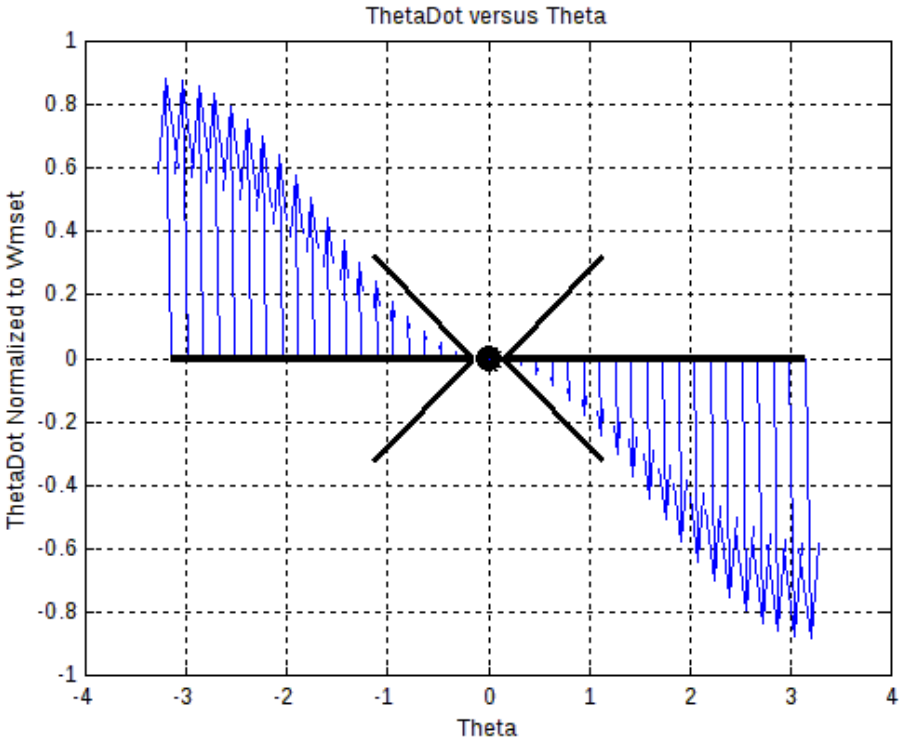


Figure 81: Flow on the line for equation (5.95).

Figure 81 is normalized in Y axis to ω_{mset} which is the maximum angular velocity that the vessel is allowed to achieve (programmable value). The angle is varied from –π to π.

Figure 81 displays the way that SC responds to various values of θ. When θ < 0 it generates an action that produces positive θ-derivative ($\dot{\theta}$), thus forcing θ to become zero (a proof that it reaches and stays at zero will follow). When θ is positive, SC responds with negative $\dot{\theta}$ which again forces the angle to become zero.

It is also interesting to see the same flow on the line for angle φ that is tightly connected to θ. Figure 82 displays the flow.

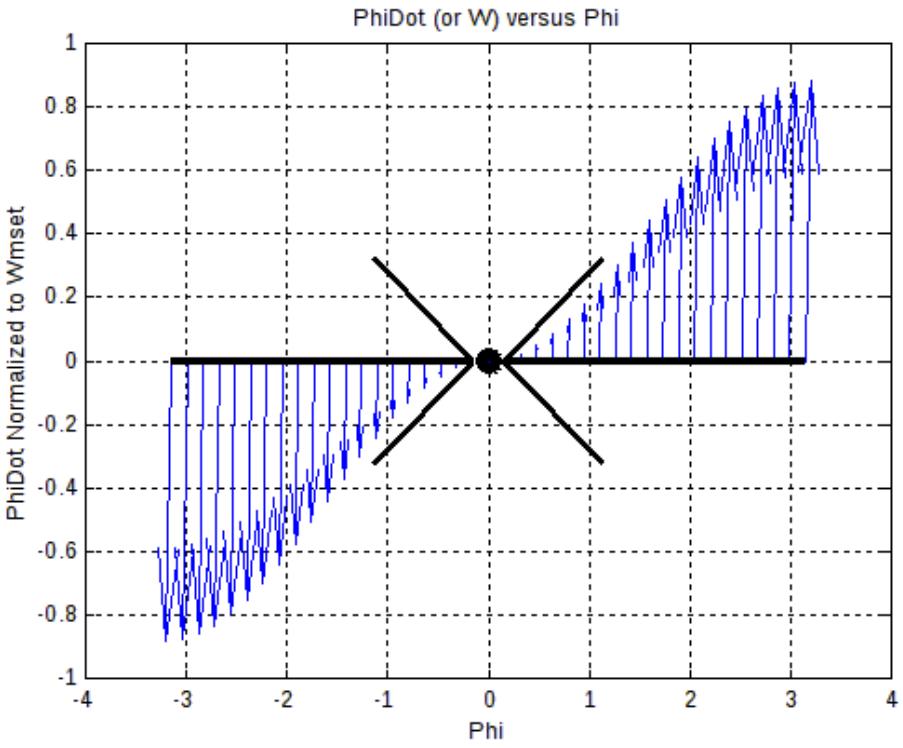


Figure 82: Attractor of Supervisory Controller angle function.

Figure 82 is again normalized in Y axis to ω_{mset} which is the maximum angular velocity that the vessel is allowed to achieve (programmable value). The angle is varied from –π to π.

What Figure 82 implies is that when the angle between vessel translational velocity and the vector connecting the vessel's center of mass to the target is negative (which means that target is on the right), then angular velocity turns negative, thus forcing the vessel to turn right, decreasing φ and as a result decreasing θ. The respective

happens when the angle is positive; the angular velocity truns positive thus decreasing θ once again.

The above processes show that zero angle point at x-axis of Figure 81 and Figure 82 functions as an attractor or stable fixed point if preffered, where the term fixed-point implies that the derivative of the variable under consideration is zero.
To delve into what happens to angles close to 0, Taylor expansion of cosθ around zero angle (or Maclaurin series) is used. The formula for expansion around zero is:

$$f(x)=f(0)+f'(0)x+\frac{f''(0)}{2}x^2+\ldots+\frac{f^{(n)}(0)}{n!}x^2+R_{(n+1)}$$

we have for cosθ:

$$\cos(\theta)=\cos(0)+\frac{d\cos\theta}{d\theta}\Big|_{\theta=0}\cdot\theta+\frac{1}{2}\cdot\frac{d^2\cos\theta}{d\theta^2}\Big|_{\theta=0}\cdot\theta^2+\ldots\Leftrightarrow$$

$$\cos(\theta)=1-\sin(0)\cdot\theta+\frac{1}{2}\cdot(-\cos(0))\cdot\theta^2+\ldots\Rightarrow$$

$$\cos(\theta)\approx 1-\frac{1}{2}\cdot\theta^2 \tag{5.96}$$

With (5.96) resulting from neglecting higher order terms (which also includes remainder $R_{(n+1)}$), since θ is close to zero.

By applying approximation (5.96) to (5.95), we have that for small angles:

$$\dot{\theta}=-\frac{\omega_{mset}}{2}\cdot\text{sign}(\theta)\cdot\left(1-1+\frac{1}{2}\cdot\theta^2\right)\Leftrightarrow$$

$$\boxed{\dot{\theta}=-\frac{\omega_{mset}}{4}\cdot\text{sign}(\theta)\cdot\theta^2} \tag{5.97}$$

At this point the assumption that $\text{sign}(\theta)=1$ is made; this implies approaching from positive angles. Then (5.97) becomes

$$\dot{\theta}=-\frac{\omega_{mset}}{4}\cdot\theta^2 \tag{5.98}$$

Beginning from (5.98) we have for θ(t):

$$\frac{1}{[\theta(t)]^2}\frac{d\theta(t)}{dt}=-\frac{\omega_{mset}}{4}\Leftrightarrow$$

$$\frac{1}{[\theta(t)]^2}d\theta(t)=-\frac{\omega_{mset}}{4}dt\Rightarrow$$

$$\int\frac{1}{[\theta(t)]^2}d\theta(t)=-\int\frac{\omega_{mset}}{4}dt\Rightarrow$$

$$-\frac{1}{\theta(t)}+c_0=-\frac{\omega_{mset}}{4}t$$

(5.99)

To find c_0 we set t = 0 and from (5.99) we get:

$$-\frac{1}{\theta(0)}+c_0=-\frac{\omega_{mset}}{4}0 \Leftrightarrow$$

$$c_0=\frac{1}{\theta(0)}\equiv\frac{1}{\theta_0} \quad (5.100)$$

From (5.99) and (5.100) we have that

$$-\frac{1}{\theta(t)}+\frac{1}{\theta_0}=-\frac{\omega_{mset}}{4}t \Leftrightarrow$$

$$\frac{1}{\theta(t)}=\frac{1}{\theta_0}+\frac{\omega_{mset}}{4}t \Leftrightarrow$$

$$\boxed{\theta(t)=\frac{1}{\frac{1}{\theta_0}+\frac{\omega_{mset}}{4}t}} \quad (5.101)$$

Or (since θ_0 is not 0):

$$\boxed{\theta(t)=\frac{\theta_0}{1+\frac{\omega_{mset}}{4}\theta_0 t}} \quad (5.102)$$

Taking the limit of (5.102) as t → ∞ we have that:

$$\lim_{t\to\infty}\left[\theta(t)\right]=\lim_{t\to\infty}\left(\frac{\theta_0}{1+\frac{\omega_{mset}}{4}\theta_0 t}\right)=0 \quad (5.103)$$

Result (5.103) implies that as time passes, the angle approaches zero. The graphical representation of time evolution of equation (5.103) is shown in Figure 83.

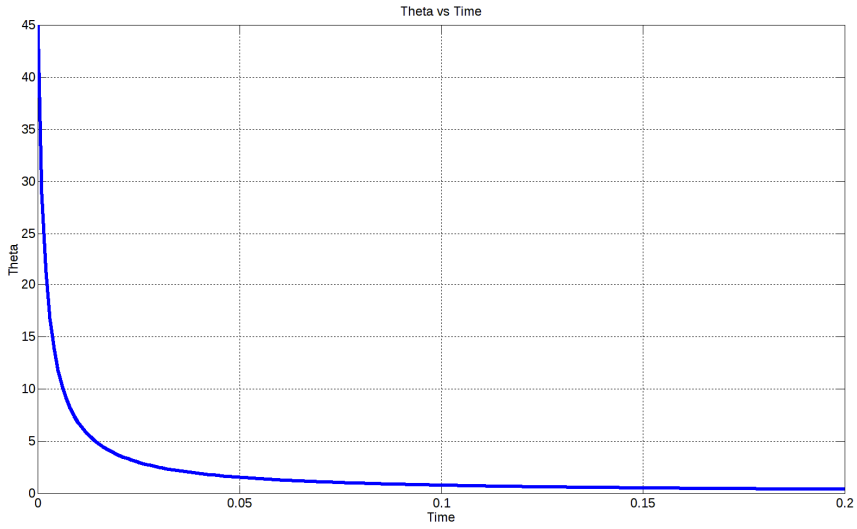


Figure 83: θ(t) versus time.

It must be made clear, that Figure 83 does not display vessel's response, but the theoretical case where the vessel would immediately follow the commands of SC.

5.8 PID as SC

In this section the use of a PID controller in the place of the non-linear SC of section 5.7 is examined.

Operating as SC, the PIDs setup aims to zero the value of the input variable, thus since the reference is zero, the feedback signal is directly fed to the PID input as error.

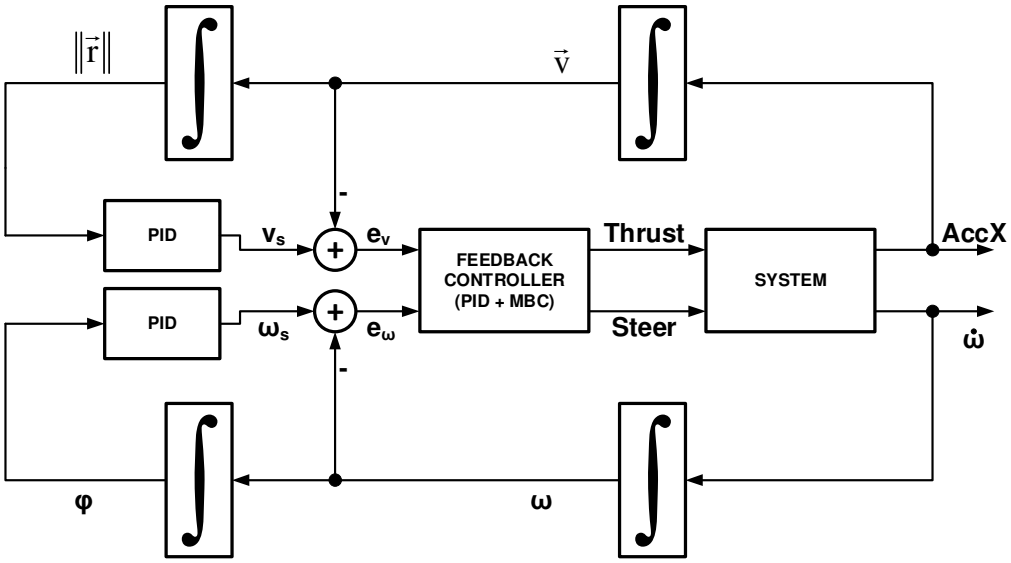


Figure 84: Full system loop with PID as supervisory controllers.

Since the MBC controller is considered to separate the loops, we use two PIDs; one for the distance zeroing and one for the angle zeroing.

Figure 84 displays the control scheme with PID as supervisory controllers. A tricky point that must be clarified, is that $\|\vec{r}\|$ is the magnitude of the vector $\vec{r}$ with reference to Figure 79. It is the vector that connects vessel's center of mass to the target. Although Figure 84 shows vectors at the distance from target loop (upper), the actual PID loop is build on the scalar value $\|\vec{r}\|$. Since translation acceleration is considered only along x-axis of O'xy, this creates a temporarily a logical issue, because we have a mix of vectors and scalars. This is resolved by the angle loop, which aligns O'x axis to $\vec{r}$, forcing operation of distance loop on a single axis.

Section 5.9.2 shows the results of PID SC operation on the vessel model. The gains selected are $K_{pc} = K_{ic} = 4$ and $K_{dc} = 0$ and the PIDs operated in positional form.

5.9 Vessel Response

5.9.1 Using Non-linear Supervisory Controller

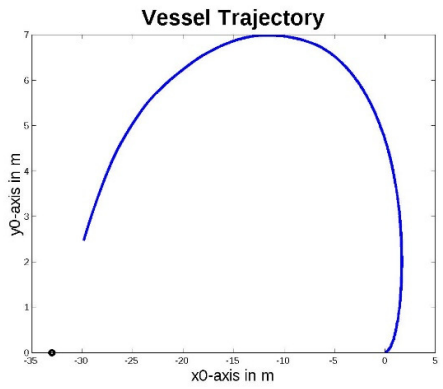
In this section, the results of using supervisory controller of section 5.7 are shown.

Go west test

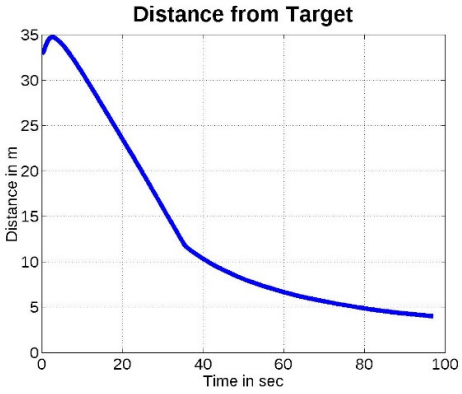
The first test of the section is a "go west" test, where the target lies at point (-33m, 0) and the vessel points towards positive x axis. Due to the construction of SC, the vessel will turn left for angles $0 < \theta \leq 180^{o}$. The test stops when vessel is 4m from the target.

Figure 85(a) shows the trajectory followed by the vessel until it reaches 4m from the target. Figure 85(b) shows how this distance changes versus time, while the vessel follows the path of Figure 85(a).

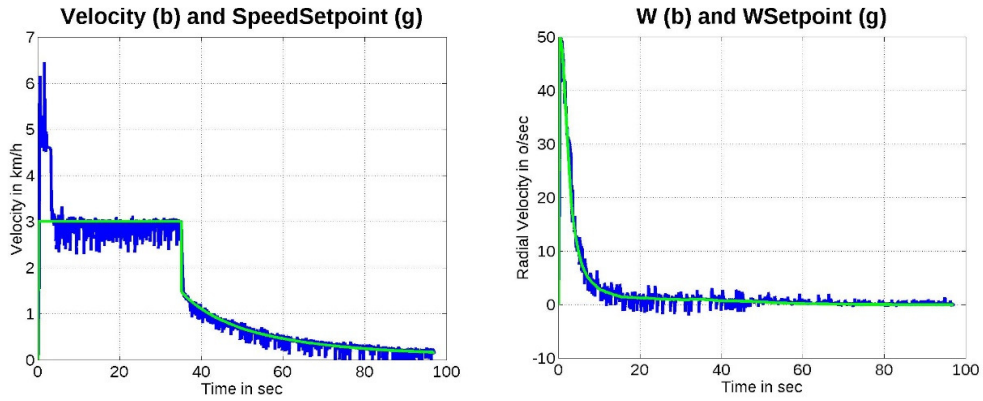
Figure 85(c) & (d) show the setpoints created by the SC and how these were followed by the vessel as a result of MBC actions. Green lines are the setpoints and blue curve follows the actual values.



(a)



(b)



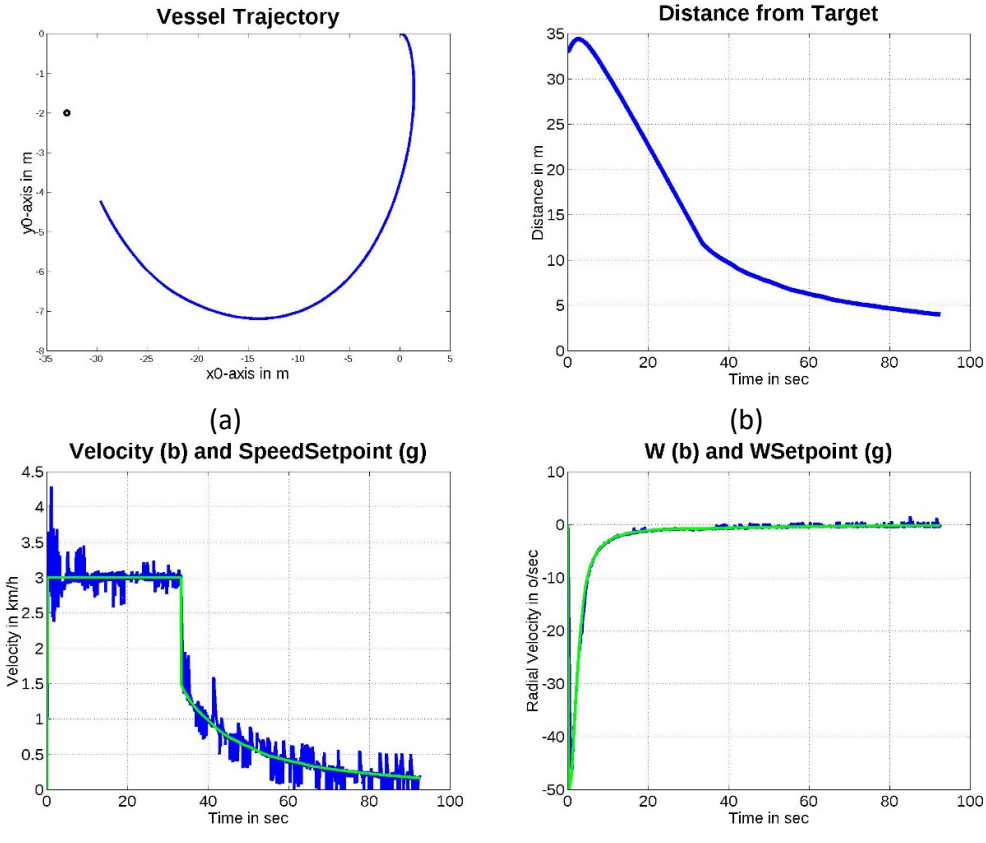
(c) (d)

Figure 85: Go west testing using supervisory controller of §5.7.

Go west right turn test

The second test of the section resembles the first, but this time the target is at (-33, -2), to force SC to make a right turn at the beginning of the course.

The results, regarding trajectory, show that the vessel begins with a right turn indeed. Figure 86(a) proves so. The rest of the plots follow the description of the previous section.



(a) (b)

(c) (d)

Figure 86: Go west right turn testing, using supervisory controller of §5.7.

Multiple setpoints test

The final test of the section involves a scenario that can be directly expanded to following a predefined course. The vessel follows a list of setpoints. The list is as follows in Table 8:

No	Target x	Target y
1	50	50
2	100	50
3	125	0
4	150	100
5	150	-100
6	100	-50
7	50	-50
8	0	0

Table 8: List of target points to be followed.

Figure 87 shows the trajectory followed during the test. The setting of the target selection logic was to pick the next target whenever the vessel had a distance of 4m from the current target. The target points are shown using a black dot.

As can be clearly seen, the vessel passes from every target successfully. The result is also verified from Figure 88(a), which shows that the distance from the target is decreased versus time; suddenly a new peak at the distance curve appears, which is the selection of a new target point from the list.

Figure 88(b) and (c) display the velocity and angular velocity curves, respectively, versus time (blue lines). The green lines display the test points from SC.

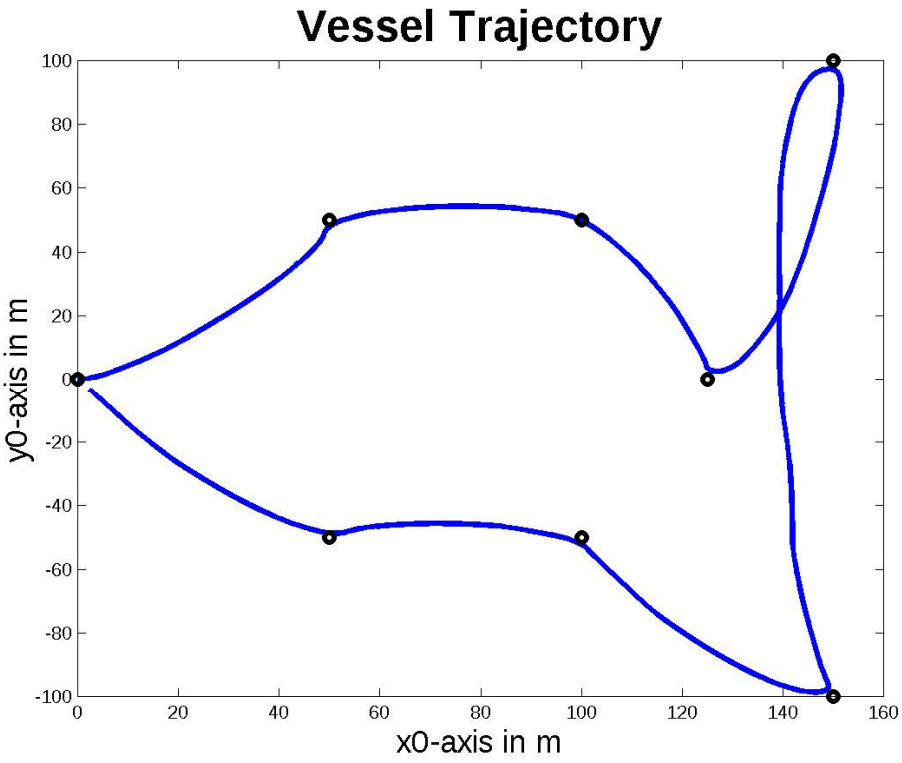
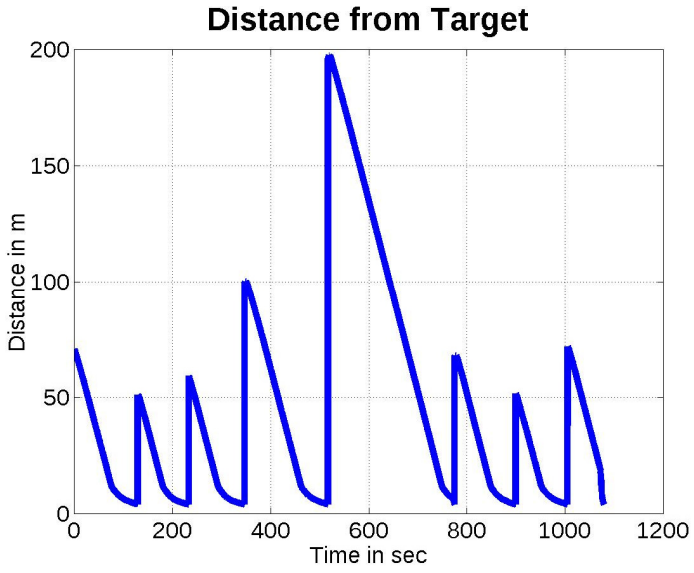
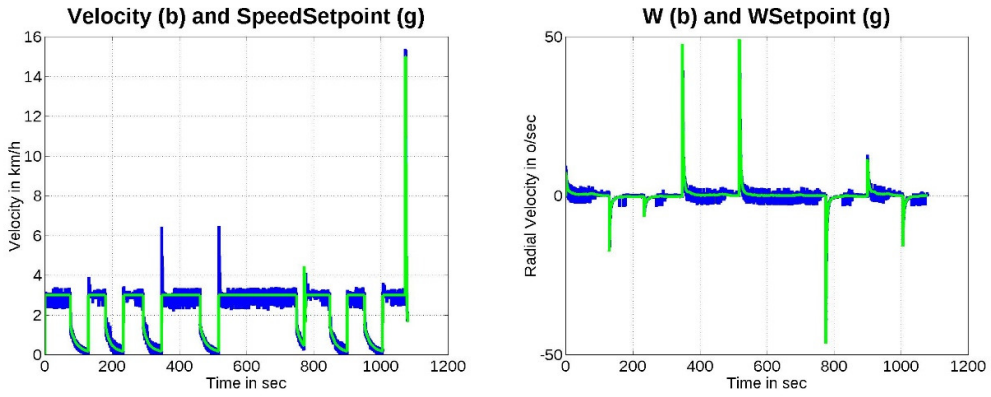


Figure 87: Trajectory followed in multiple targets test.



(a)



(b) (c)

Figure 88: Multiple setpoints following results. (a) distance versus time; (b) Velocity setpoints (green line) and actual velocity samples (blue line); (c) Yaw rate setpoints (green line) and actual yaw rate samples (blue line).

5.9.2 Using PID Supervisory Controller

In this paragraph, the results from using a PID controller as SC are presented. Instead of employing the non-linear SC of section 5.7, we replaced it with a PID having 0 as setpoints (see section 5.8 "PID as SC").

Go west test

In this test we have a target point at (-33, 0) and we monitor the response of the vessel as it tries to find a target at 180^{o}. Due to the design of SC controller the vessel is expected to take a right turn to head for the target.

Figure 89 shows the results of this test. The trajectory of the vessel is shown at Figure 89(a). Comparing to the previous SC design, for the current settings of both SCs, the response of PID SC is more "absolute". It creates large accelerations reaching the target much faster.

Whether or not is a preferred design depends on the application. If it were to be used for human transportation it would be rather unsuitable and appropriate adjustments to PID gains would have to be made. On the other side if the application was UAV, then abrupt movements are not that unsuitable.

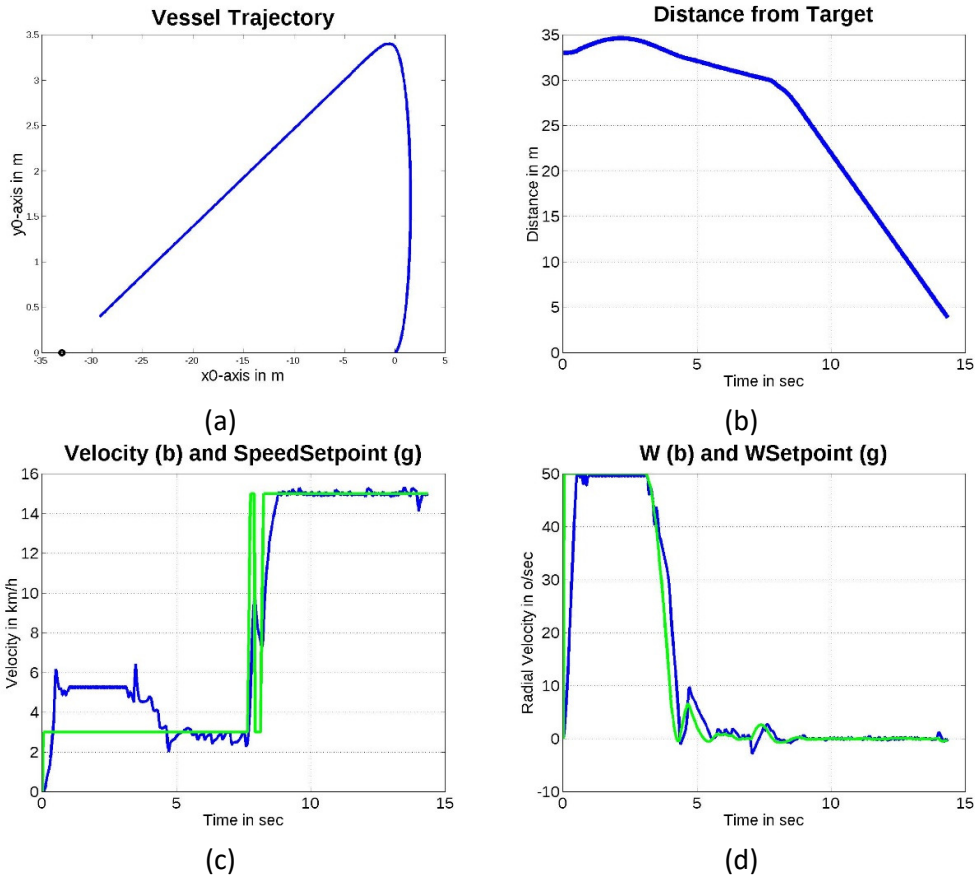


Figure 89: Result from "Go West" test using PID SC.

Go west right turn test trial 1

This test is almost the same to previous paragraph, but this time the target is at (-33, -2), forcing the vessel to take a left turn.

This time an anomaly is created, which is an indication that the RC model is not the same when turning left to turning right. Though the vessel is again successful in reaching its target.

Figure 90 shows the results of the test where an overshoot at the right turn is detected. One easy way to compensate this overshoot (it is also possible to run optimization of PID parameters) is to reduce the maximum yaw rate requested from SC. The results of reducing the yaw rate are shown in the next paragraph.

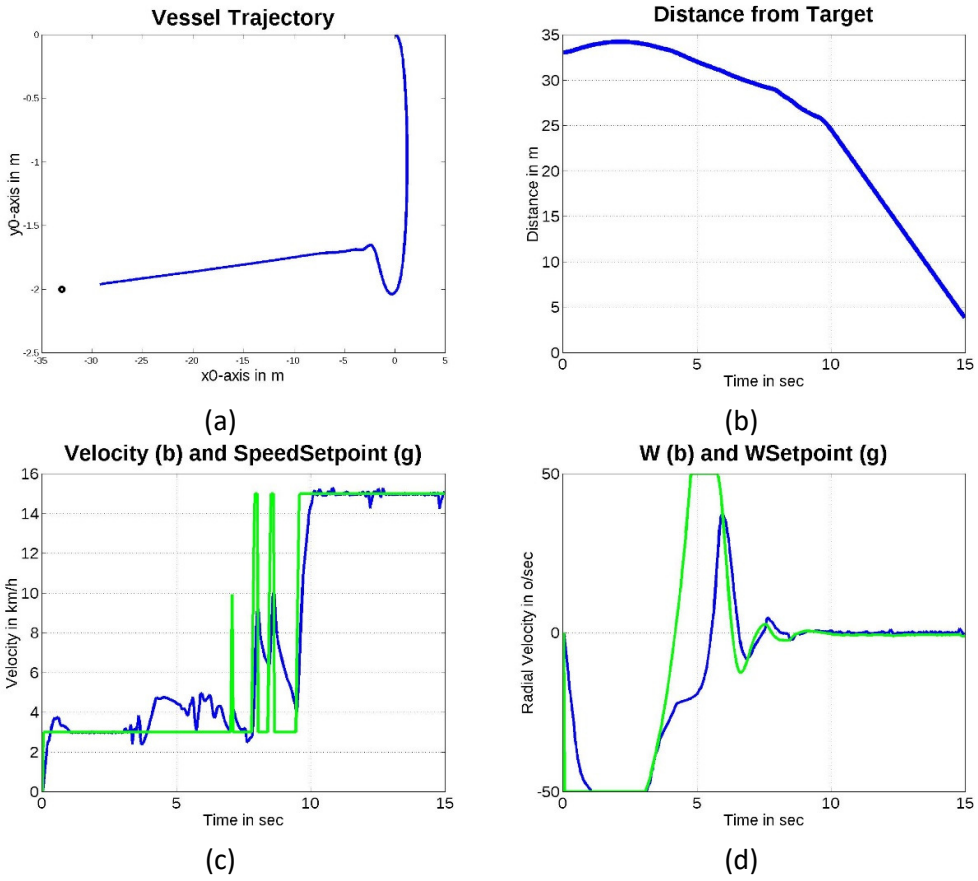


Figure 90: Go west right turn test results.

Go west right turn test trial 2

This test is exactly the same as the previous with the only exception that there is a yaw rate limiter at 25°/sec. This change fixes the overshoot a lot, as Figure 91(a) shows.

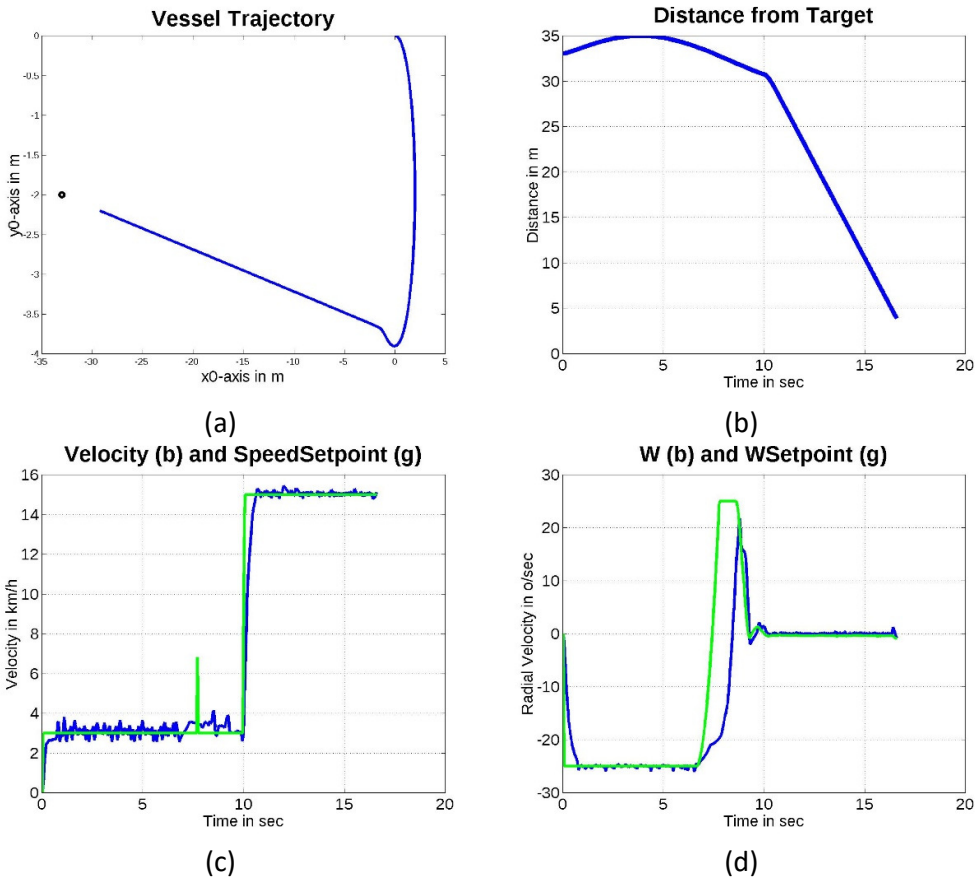


Figure 91: Results of "Go west right turn" test with yaw rate limiting.

Multiple setpoints test

As with the previous SC type, the final test is a multiple-point path that must be followed.

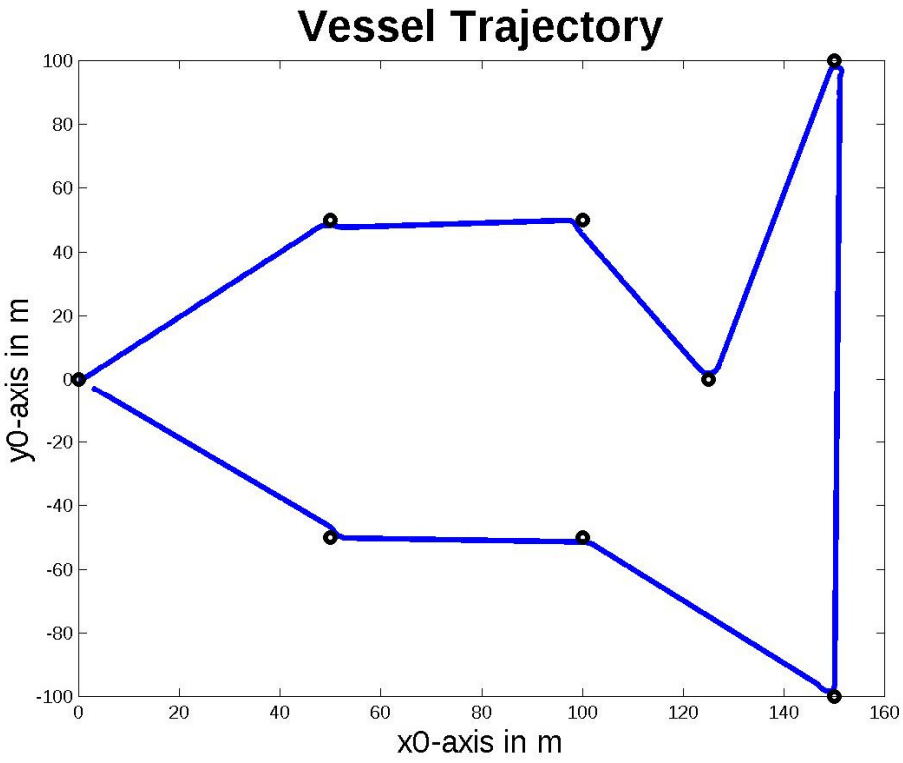
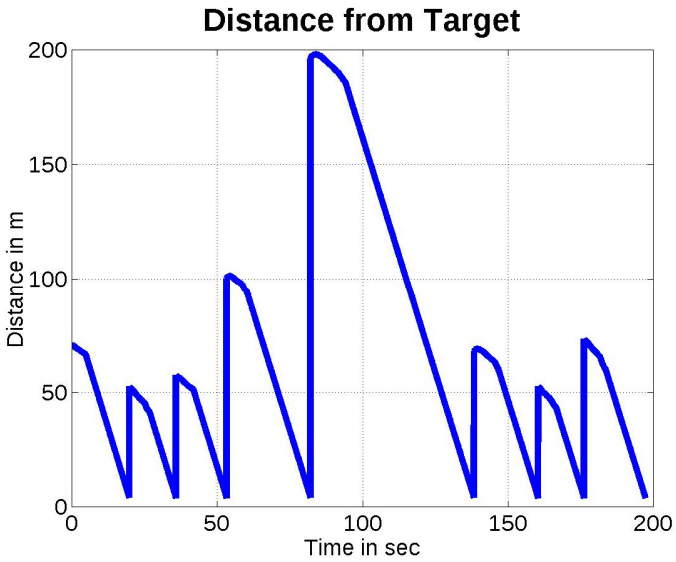


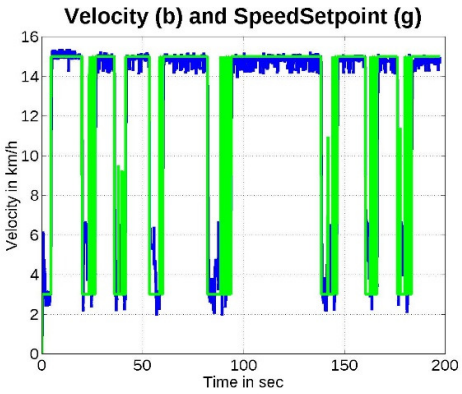
Figure 92: Vessel trajectory using PID SC. Multiple setpoints test.

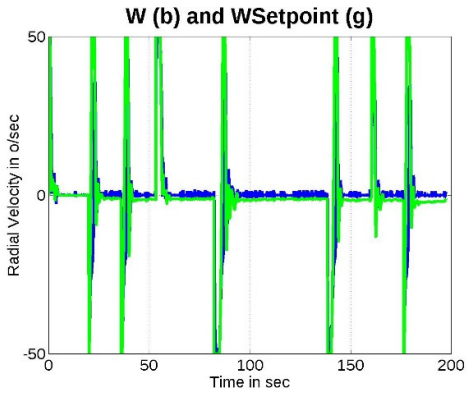
Comparing to the nonlinear SC of the previous section, as we expected from the previous tests the response is a stricter path with the vast majority of paths being straight lines. This behavior was expectable since we saw analogous response at the previous tests.

As discussed earlier, there is no “better” or “worse” between the two responses; it all comes down to what we want as an end-application.



(a)





(c) (d)

Figure 93: Results of multiple setpoints test fro PID SC.

6 Forming Swarms of Vessels

Having tested a single vessel response with satisfactory results, the next step is to proceed in trials for vessel swarms.

Our scenarios will have a master vessel and the slave vessel(s) will follow paths with predefined offsets from master vessel.

We begin these tests with the assumption that the Master vessel should use the non-linear supervisory controller of section 5.7. It creates slower response, and it should be easier to follow. The satellites will have to respond faster to master direction changes, thus the educated guess is that PID supervisory controller suits them more.

6.1 Go West Test with Swarm of 3 Vessels

In this scenario we will use three vessels. Vessel No1 is the master and is the one that will be given setpoints.

Vessels No2 and 3 will be satellites of 1, by having offsets 3m from No1 and angles of $+90^o$ and -90^o respectively. The target points for each of the satellites is dynamically adapted, based on the current Master vessel position and the predefined offsets for it. Table 9 describes the colors, the symbols and what they stand for.

No	Symbol and/or Color	Explanation
1	Blue	Master vessel curve.
2	Green	Satellite vessel curves. This satellite is situated 3m far from master vessel at an angle of $+90^o$.
3	Black	Satellite vessel curves. This satellite is situated 3m far from master vessel at an angle of -90^o.
4	Bold Circles	Master vessel target points.
5	Small Circles	Satellite vessels target points (which one depends on the color).

Table 9: Curves and symbols legend for swarm curves.

The results of this test are shown in Figure 94. Only for the sake of simulation, vessels are considered as points and all of them are starting from (0,0). The green vessel is to stay at $+90^o$ of master vessel course and the black at -90^o both at a distance of 3m.

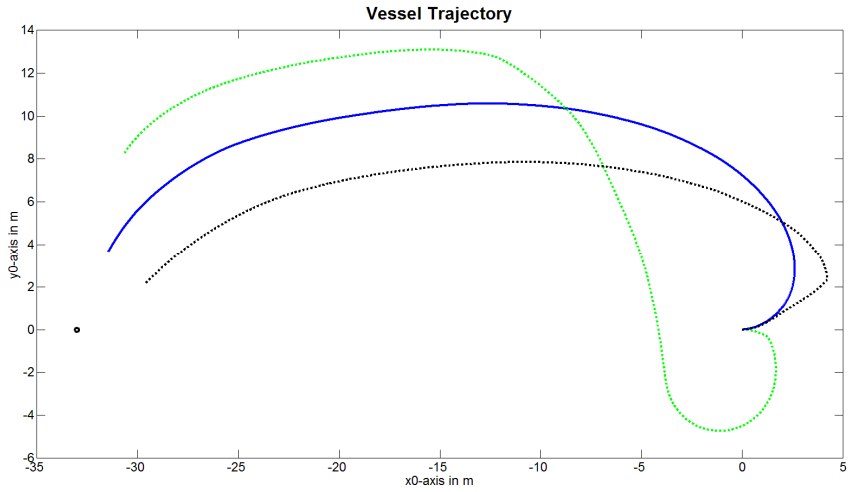


Figure 94: Trajectories of the three vessels at go west test.

As can be seen in Figure 94, the vessels after an initial transient that has to do with the master's turning, the satellite vessels follow the requested path (all the figures used in this subsection are in: "Matlab\Matlab_180910\ModelsRevisionAndControl\Code_synced_with_PHD_doc\FullSystemCode\SwarmTestBedIO\MiniSwarmTestBedResults_20230311_2228\ManualSave").

Figure 95 shows the trajectories of all vessels, along with selected snapshots (one every 50 sampling periods) of setpoints for the satellites (small circles).

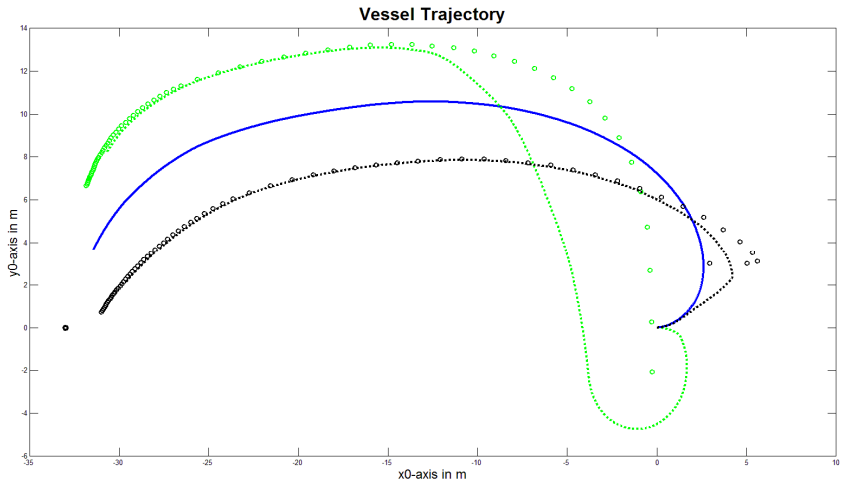


Figure 95: Trajectory of the three vessels, along with the satellite target points.

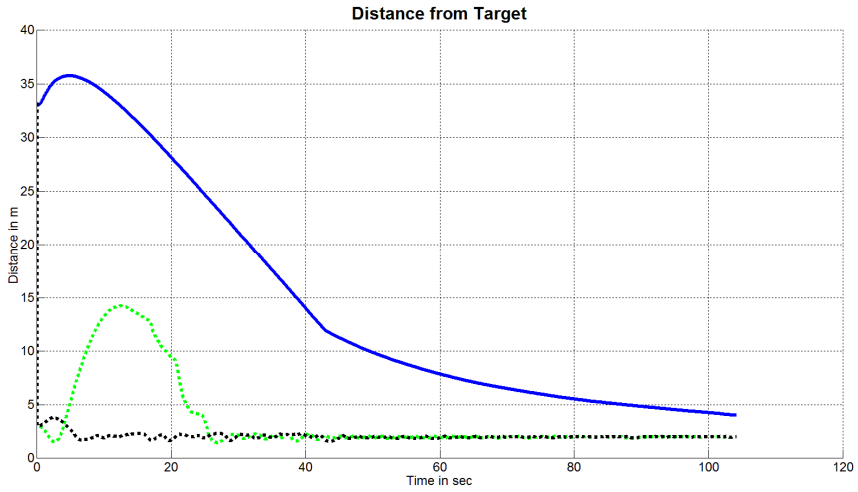


Figure 96: Distance of each of the vessels in the swarm, from their targets.

As Figure 96 shows, the distance of all vessels from their target positions is reduced by the control action.

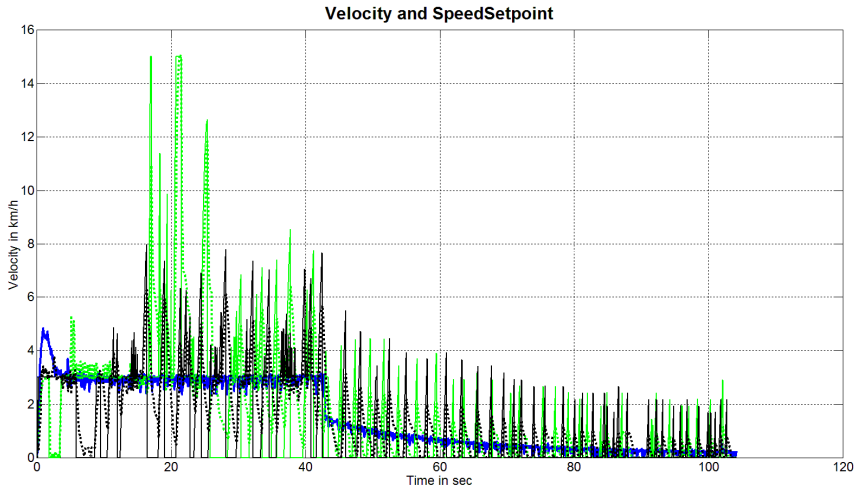


Figure 97: Velocity of vessels along with the command of their supervisory controllers.

Figure 97 contains a lot of information and is not easily readable. It shows for every vessel the target velocity requested by the respective supervisory controller, along with the actual velocity. Figure 98 is a zoom into a part of Figure 97, and reveals that control action follows the requested velocity in the sense that internal control loops can manage. Table 12 aids in understanding of the aforementioned figures.

No	Symbol and/or Color	Explanation
1	Blue noisy curve	Master vessel velocity curve.
2	Blue smooth curve	Master supervisory requested velocity.
3	Green dotted curve	Satellite vessel 1 actual velocity curve.
4	Green smooth curve	Satellite vessel 1 supervisory requested velocity.
5	Black dotted curve	Satellite vessel 2 actual velocity curve.
6	Black smooth curve	Satellite vessel 2 supervisory requested velocity.

Table 10: Symbolization and colors for velocity curves.

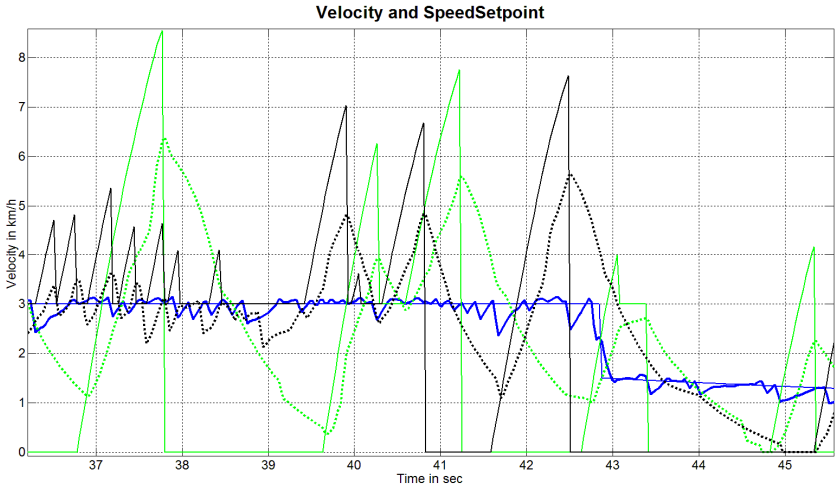


Figure 98: Detail of previous figure.

All the information provided for the previous two curves regarding velocity, apply to angular velocity curves as well. Figure 100 is a zoomed in version of Figure 99.

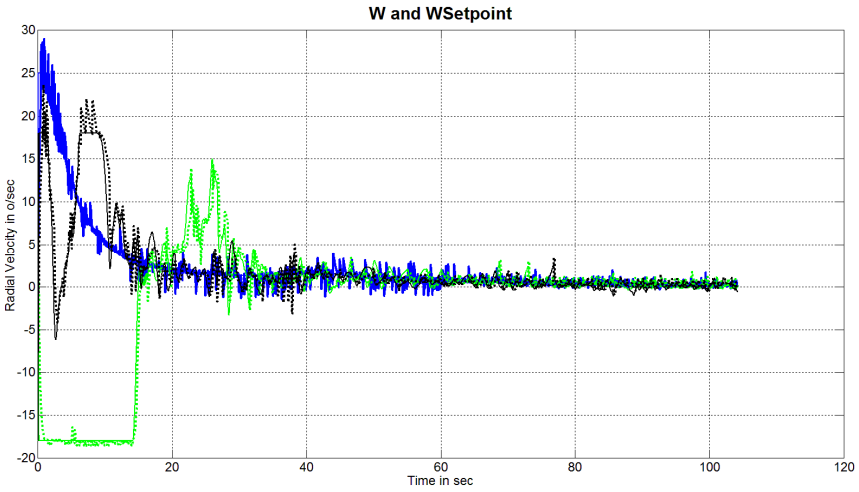


Figure 99: Angular velocity for all vessels along with supervisory setpoints.

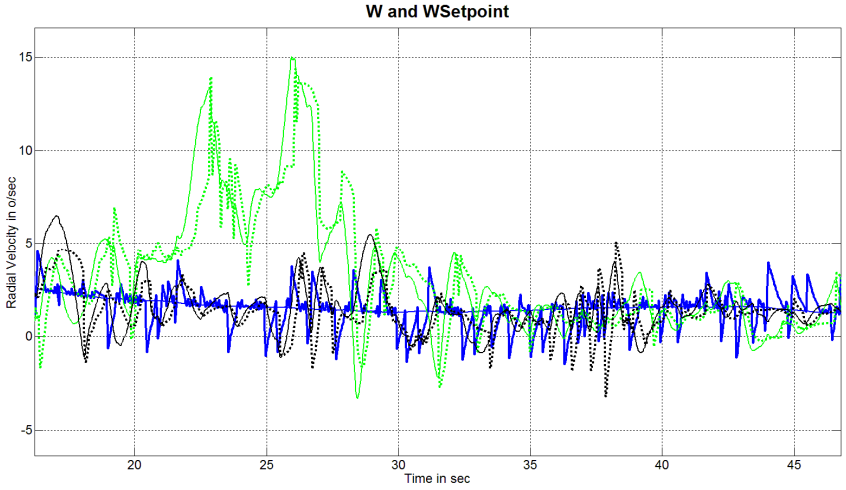


Figure 100: Detail of previous figure.

6.2 Multiple Setpoints Test

This time the tree vessels are requested to follow a multiple target path as the one used in subsection 5.9.1.

No	Target x	Target y
1	50	50
2	100	50
3	125	0
4	150	100
5	150	-100
6	100	-50
7	50	-50
8	0	0

Table 11: List of target points to be followed by master vessel.

The formation and symbols, curve colors etc. are the same for the three vessels, as discussed in the previous subsection.

The result of this simulated experiment was successful, with the satellites following the master vessel, as apparent in Figure 101.

All figures used for this paragraph are taken from simulation results in folder "Matlab\Matlab_180910\ModelsRevisionAndControl\Code_synced_with_PHD_doc\FullSystemCode\SwarmTestBedIO\MiniSwarmTestBedResults_20230311_1855\ManualSave".

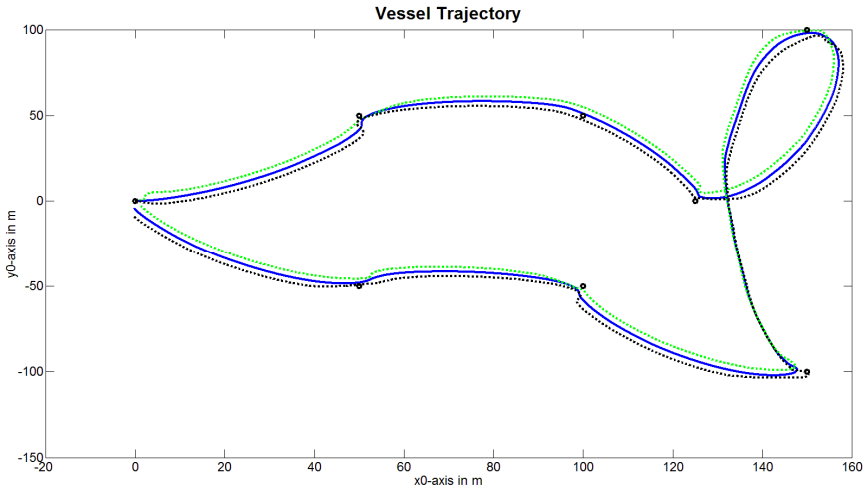


Figure 101: Trajectories of swarm vessels.

Figure 102 displays also snapshots of the targets of satellite vessels along the course of master vessel.

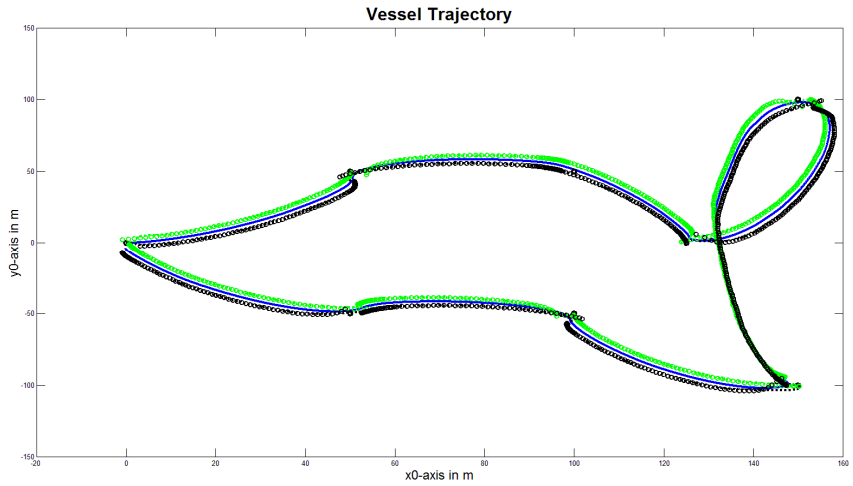


Figure 102: Previous figure along with the setpoints for each satellite.

Figure 103 is a zoomed-in version of Figure 102 where the success of satellites to follow their position targets can be recognized. Satellite paths are shown with dotted lines.

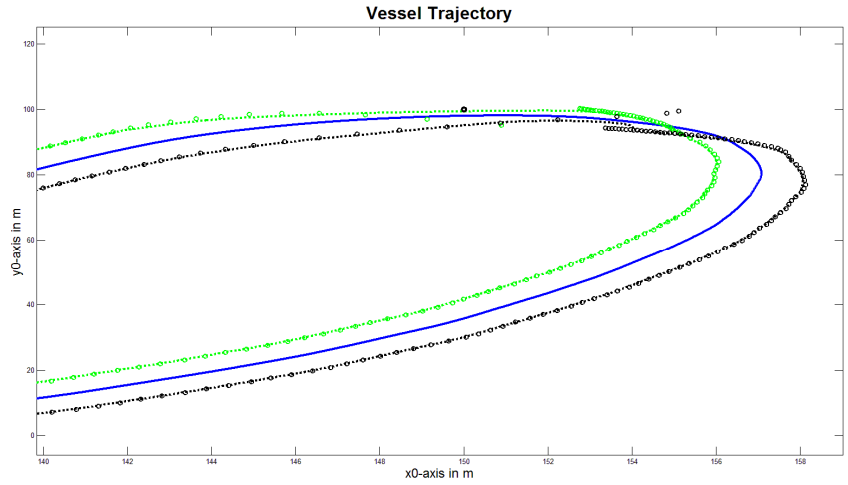


Figure 103: Detail of previous figure.

Figure 104 shows how control actions decrease the distance from each vessel targets. Satellite vessels are forced to stop at 2m distance from their predefined positions relative to master vessel.

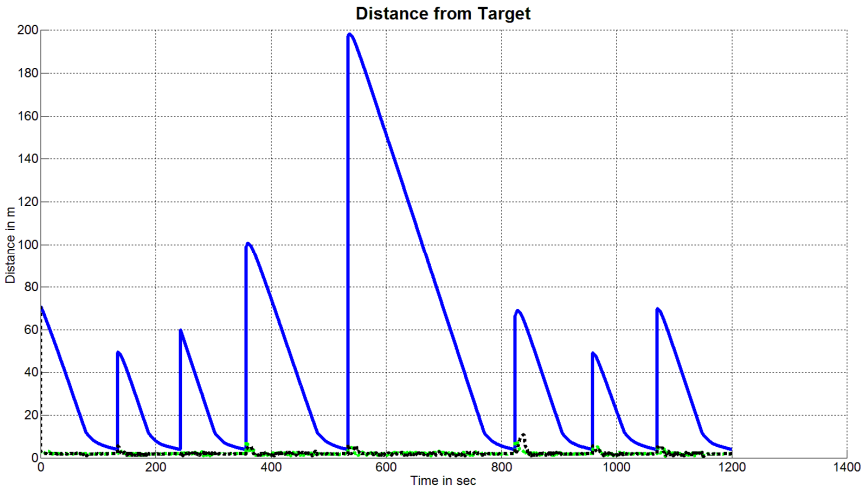


Figure 104: Distance of all vessels from their targets.

The figures that follow, have the same description as used in the previous paragraphs. They describe velocities and for each velocity there is a zoomed version of the figure that follows. Table 10 also applies to this subsection.

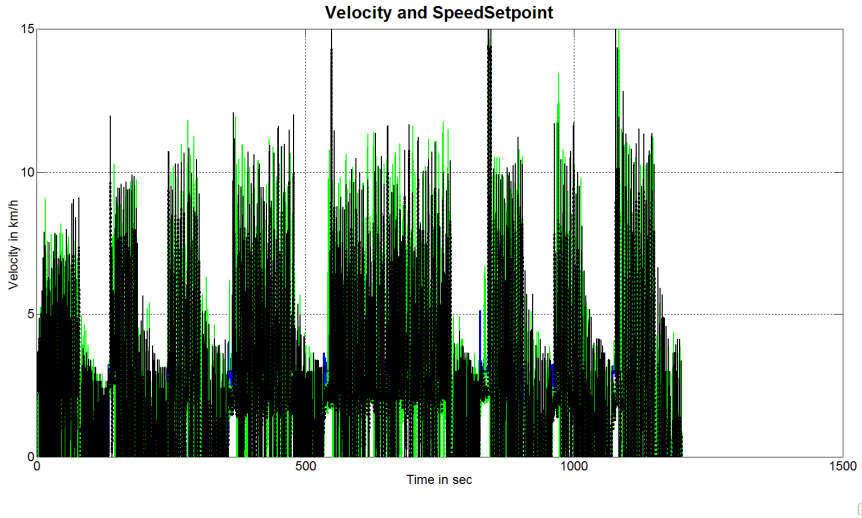


Figure 105: Velocities and velocity setpoints of swarm vessels.

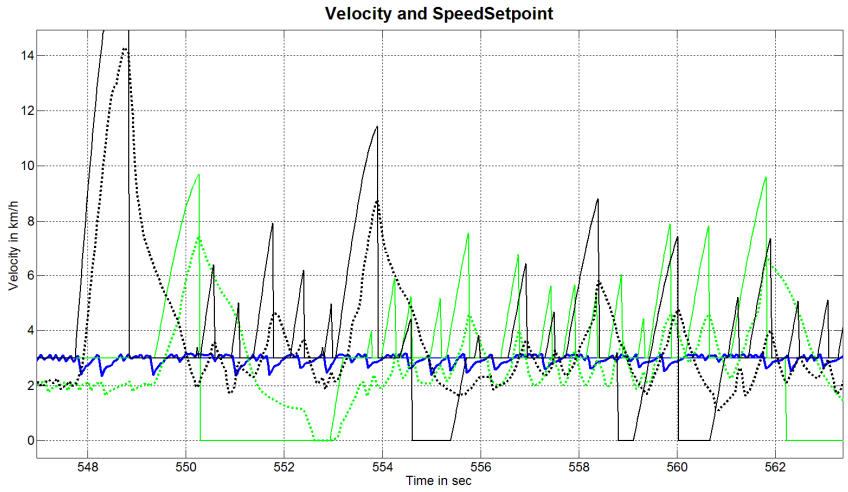


Figure 106: Detail of previous figure.

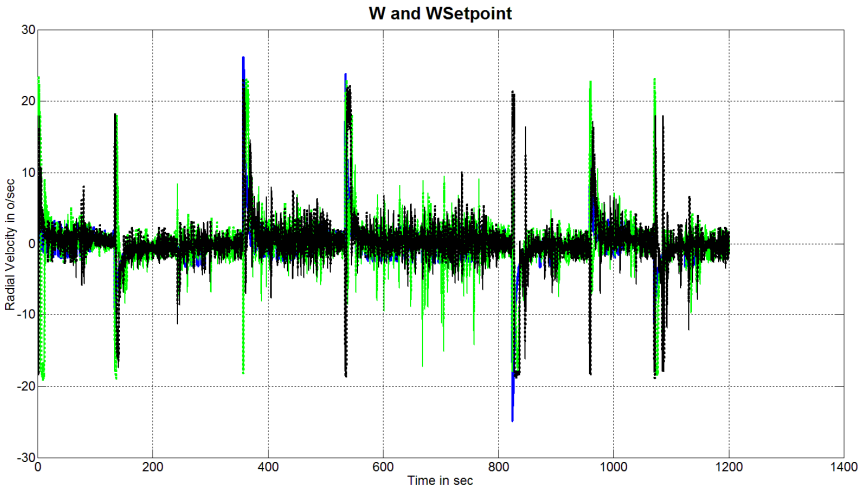


Figure 107: Angular velocities and angular velocity setpoints of swarm vessels.

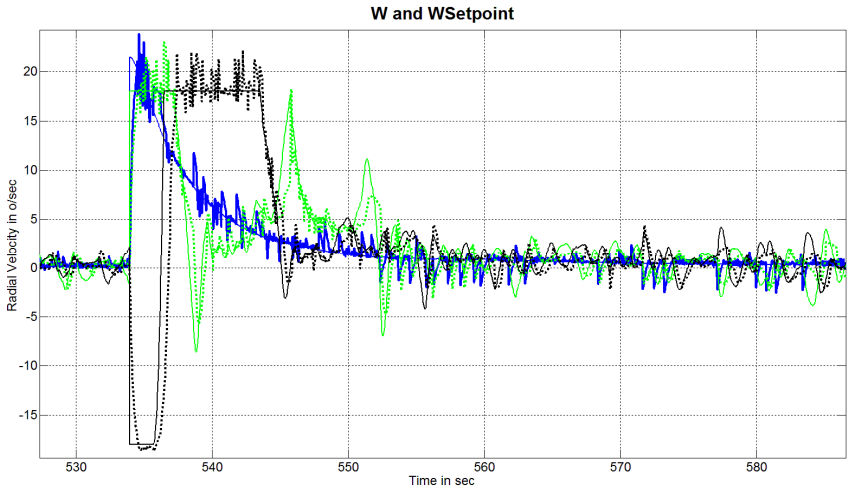
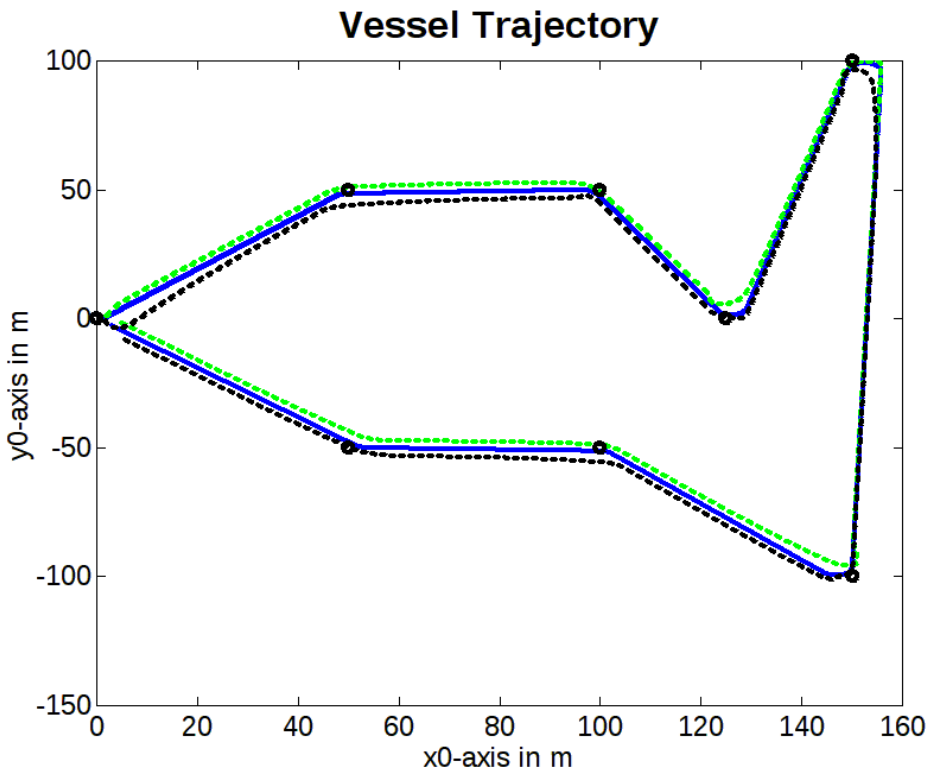


Figure 108: Detail of previous figure.

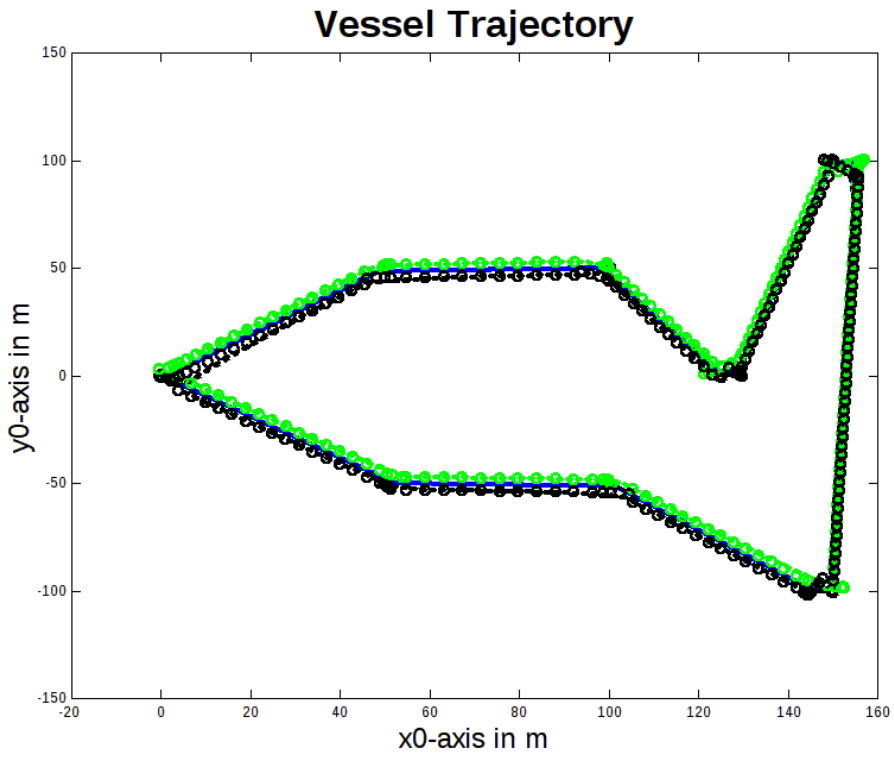
6.3 Multiple Setpoints Test with PID Supervisory for Master

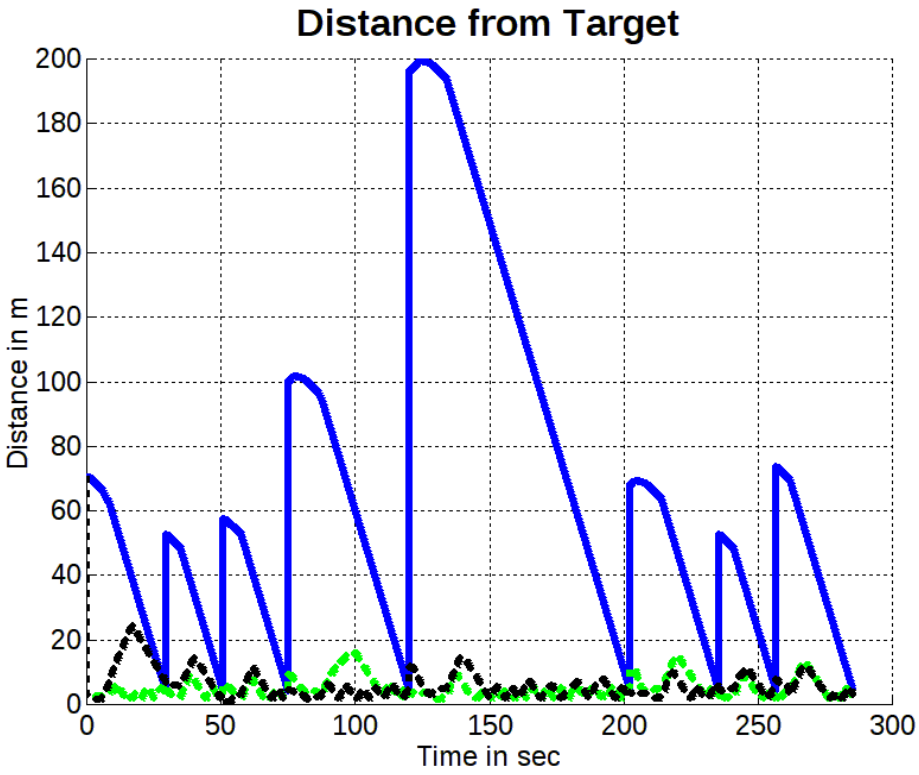
All figures used in this section are stored in "Matlab\Matlab_180910\ModelsRevisionAndControl\Code_synced_with_PHD_doc\FullSystemCode\SwarmTestBedIO\MiniSwarmTestBedResults_20230311_1855\ManualSave".

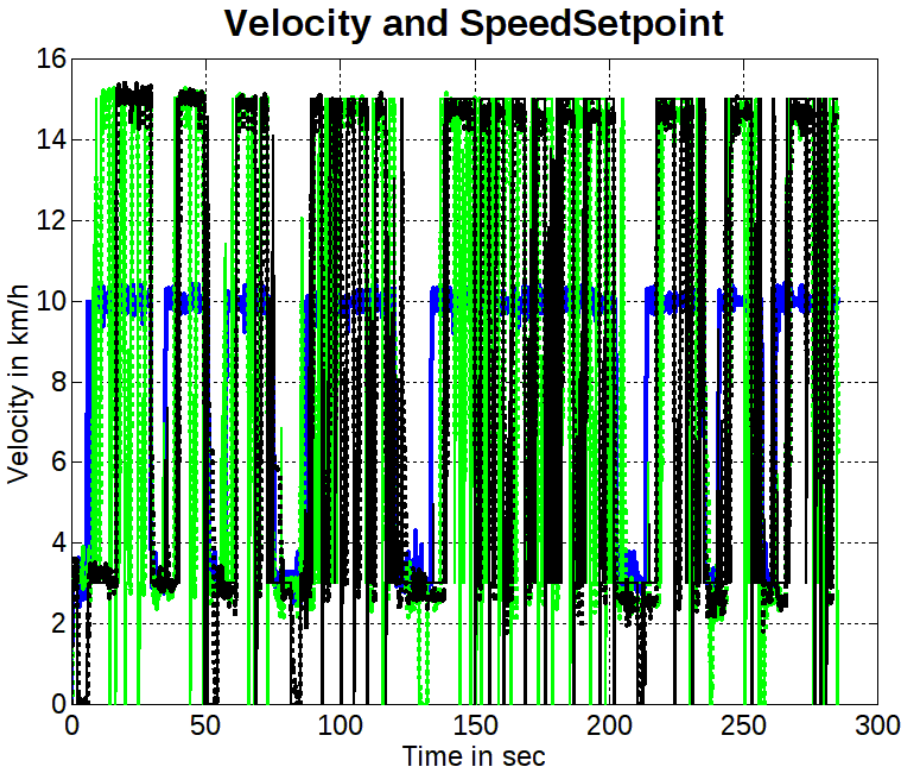


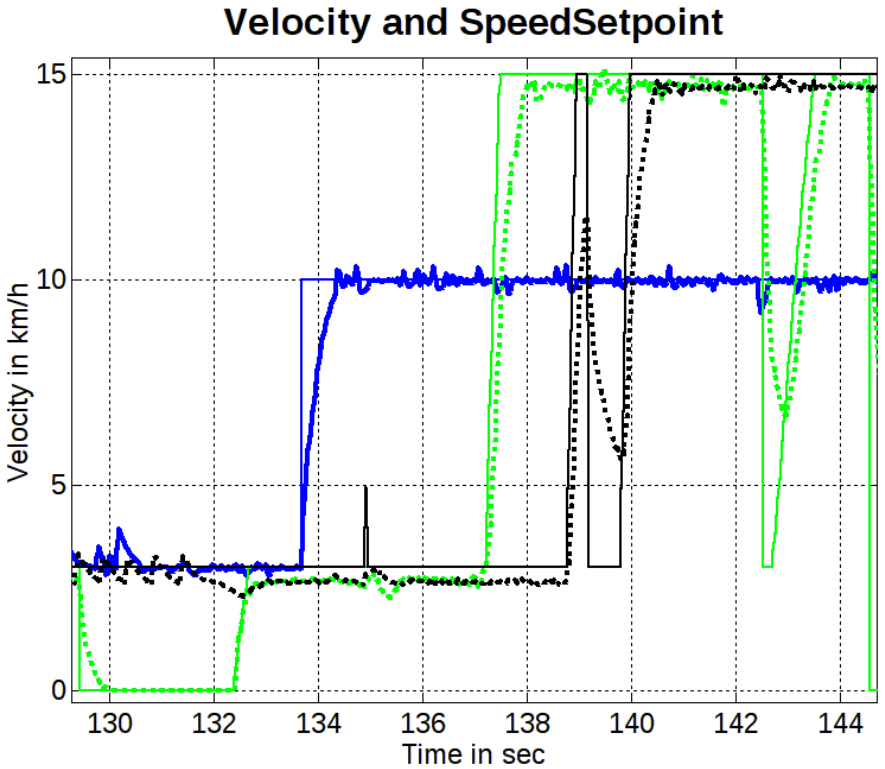
This experiment shows that even if we use PID supervisory controller for the master vessel, satellites can follow. We have tweaked the settings of master vessel though, regarding maximum allowable velocities. Master vessel is a bit slowed-down to allow faster response from the satellites; this makes it easier for satellites to follow master movements.

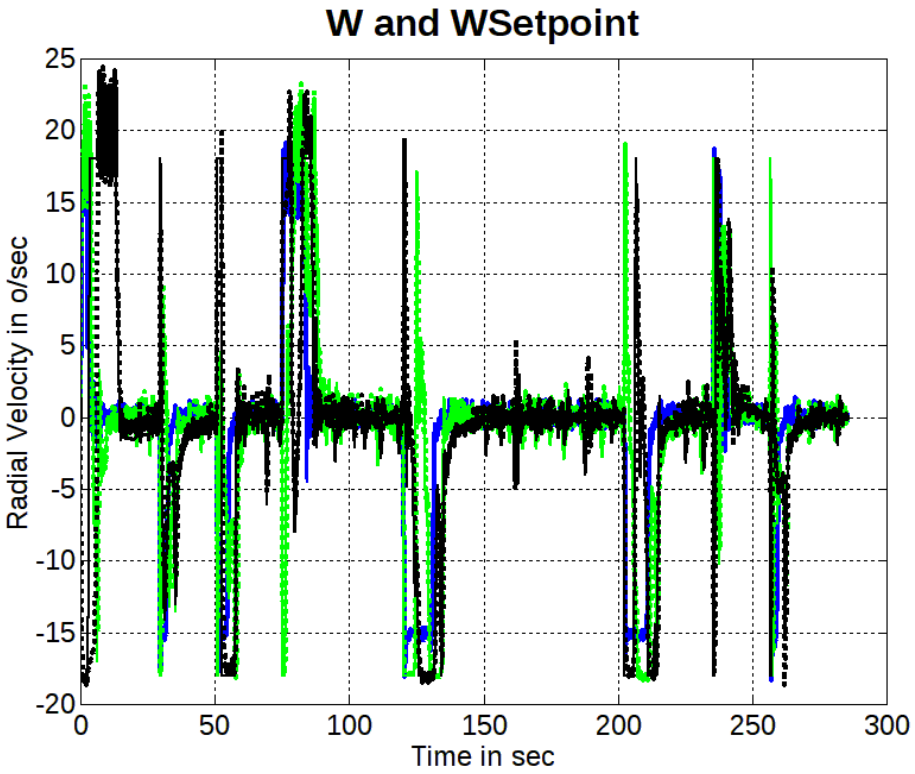
The figures shown in this subsection follow the logic and symbolization of previous subsections.

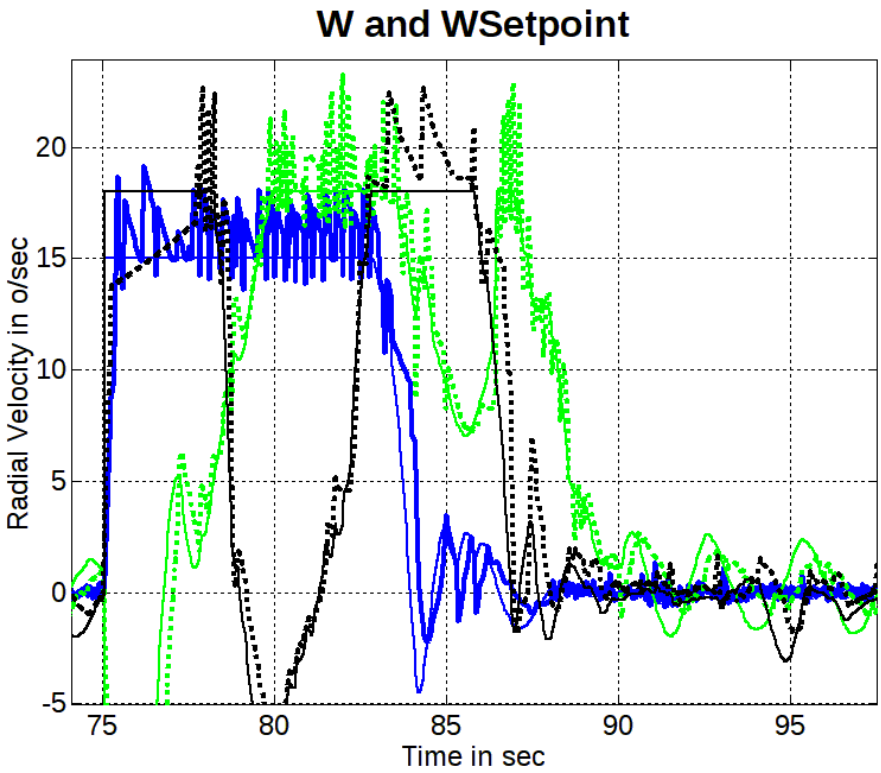

Vessel Trajectory
y0-axis in m
x0-axis in m
150
100
50
0
-50
-100
-150
-20
0
20
40
60
80
100
120
140
160


Distance from Target
Distance in m
Time in sec
200
180
160
140
120
100
80
60
40
20
0
0
50
100
150
200
250
300


Velocity and SpeedSetpoint
Velocity in km/h
Time in sec
0
2
4
6
8
10
12
14
16
0
50
100
150
200
250
300


Velocity and SpeedSetpoint
Velocity in km/h
Time in sec
0
5
10
15
130
132
134
136
138
140
142
144


W and WSetpoint
Radial Velocity in o/sec
Time in sec
25
20
15
10
5
0
-5
-10
-15
-20
0
50
100
150
200
250
300


W and WSetpoint
Radial Velocity in o/sec
Time in sec
20
15
10
5
0
-5
75
80
85
90
95

7 Hardware Description

Figure 109 gives an overview of the surface vessel constructed for this thesis.

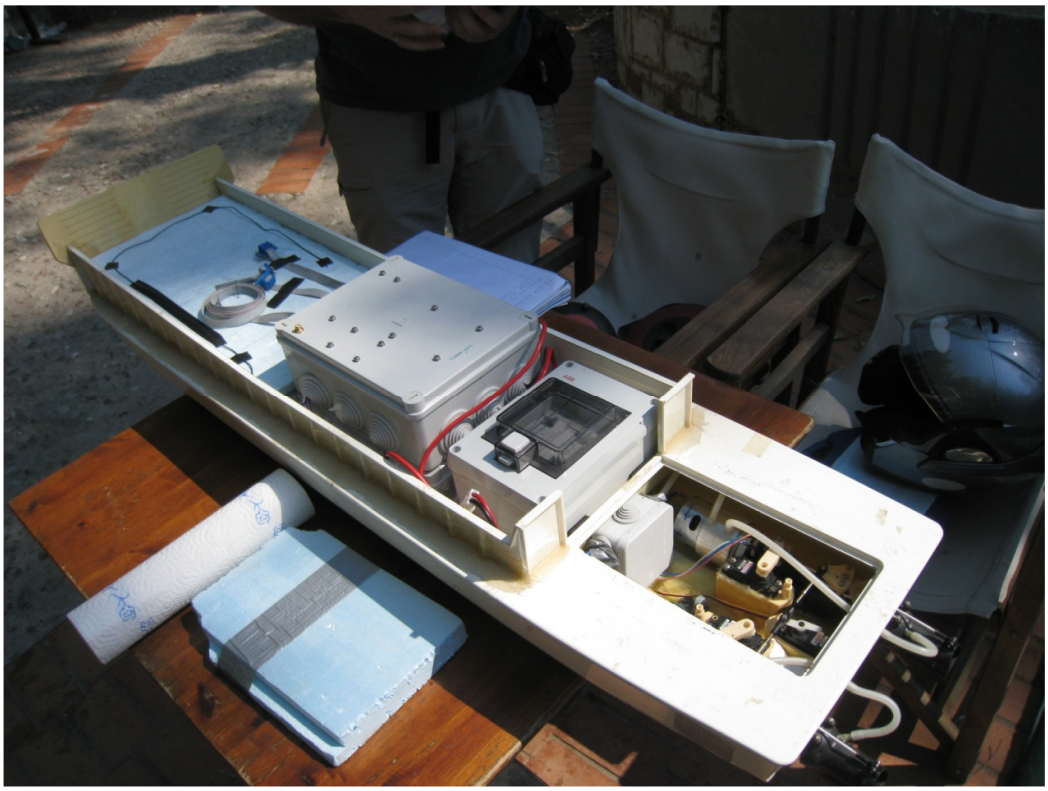
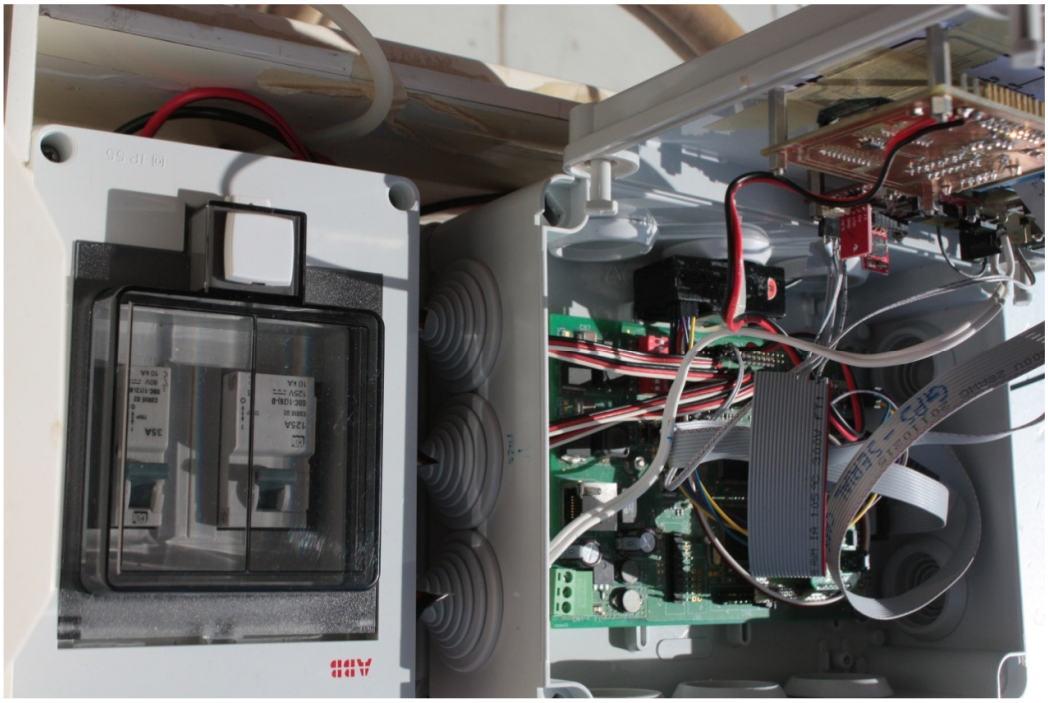
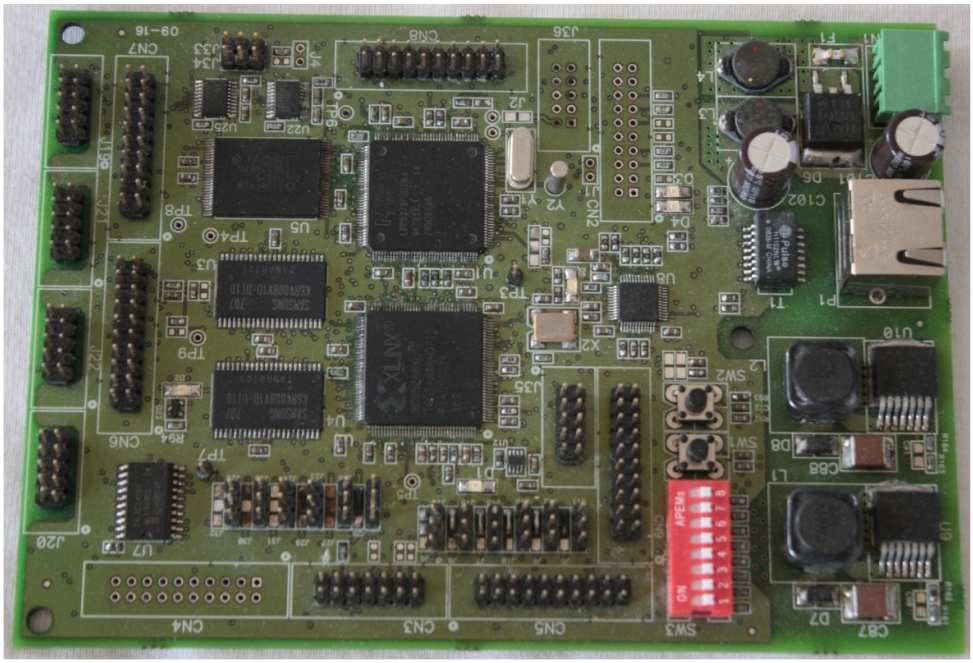


Figure 109: Surface vessel with all systems mounted

7.1 Vessel Electrical Part



7.2 Description of Development Board



AP_IV_v1_v4 is a development board which provides the following resources:

- Microcontroller NXP LPC2378, based on ARM7TDMI a RISC core that can run up to 70MHz.
- Supporting CPLD for hardware functions.
- 1MB SRAM.
- 16MB Flash.
- Four serial interfaces (RS232 levels), that can be used also as CMOS levels RS232.
- Two I2C interfaces.
- 10/100Mb Ethernet Interface.
- Switching power supply providing two voltages +3.3V and +5V at 2.5A each. The system when operates it consumes 100mA to 150mA. The input to the power supplies can have any level between 8.5V to 30V.
- 44 digital I/Os.
- 7 analog inputs.
- One analog output.
- Expansion headers for more I/Os and more system functionality if needed.

7.2.1 Assembly Description

Referring to Figure 110 the main parts of the board are:

No	Description
1	Flash IC (128Mb = 16Mb).
2	Microcontroller digital I/O connector (5xI/O pins, 9xGND, 2x3.3V power pins).
3	Microcontroller.
4	Microcontroller Analog/Digital I/O pin connector (4 Analog inputs or digital IOs, 2x3.3V power pins, 4xGND pins).
5	Debugger connector for the microcontroller (uC's JTAG and a Reset).
6	Power connector.
7	Ethernet connector.
8	+5V power supply.
9	+3.3V power supply.
10	General purpose button. It's function is user-programmable.
11	System Reset button.
12	General purpose DIP switch. SW1 has been programmed to function as a switch that allows re-programming of the microcontroller. SW2 has been programmed to switch +5V supply on and off. All the reset are not programmed.
13	+5V power connector. This connector has 9x5V pins, 9xGND pins and 2x3.3V pins.
14	Analog/Digital I/O connector. This connector has 3xAnalog input pins, or 2xAnalog inputs and one analog output or 3 digital IOs and 4xGND, 2x3.3V and 1x5V pins.

15	Six PWM outputs. Each output is on a three pin connector containing also GND and a selectable +3.3V/+5V supply.
16	Processor supporting CPLD. It also contains watchdog to function as a secondary process monitor.
17	Connector providing 9xGPIO pins from uC (microcontroller). It also has 9xGND pins 2x+3.3V power pins.
18	512KB SRAM IC. One of the two. This is SRAM1.
19	512KB SRAM IC. Second of the two. This is SRAM2. Together with SRAM1 they form a 1MB SRAM memory space.
20	CPLD JTAG connector. This is the connector used for programming the CPLD using the programming cable.
21	Six PWM inputs. These inputs are buffered on-board. They are part of a three pin header also providing GND and +3.3V or +5V, just like the outputs.
22	Connector containing mixed CPLD and uC digital IOs (6 and 2 respectively), along with two +3.3V pins and 9 GND pins.
23	Serial interface 4.
24	Serial interface 3.
25	Connector with 9xDigital IOS, 2x+3.3V pins, 9xGND pins.
26	Serial interface 2.
27	Connector providing 9xuC digital IOs, along with 2x+3.3V and 9xGND pins.
28	Serial interface 1. This serial interface is used also for programming the uC.
29	Two 3x1 headers that provide two I2C interfaces.

Table 12 : Main parts of the board briefly explained.

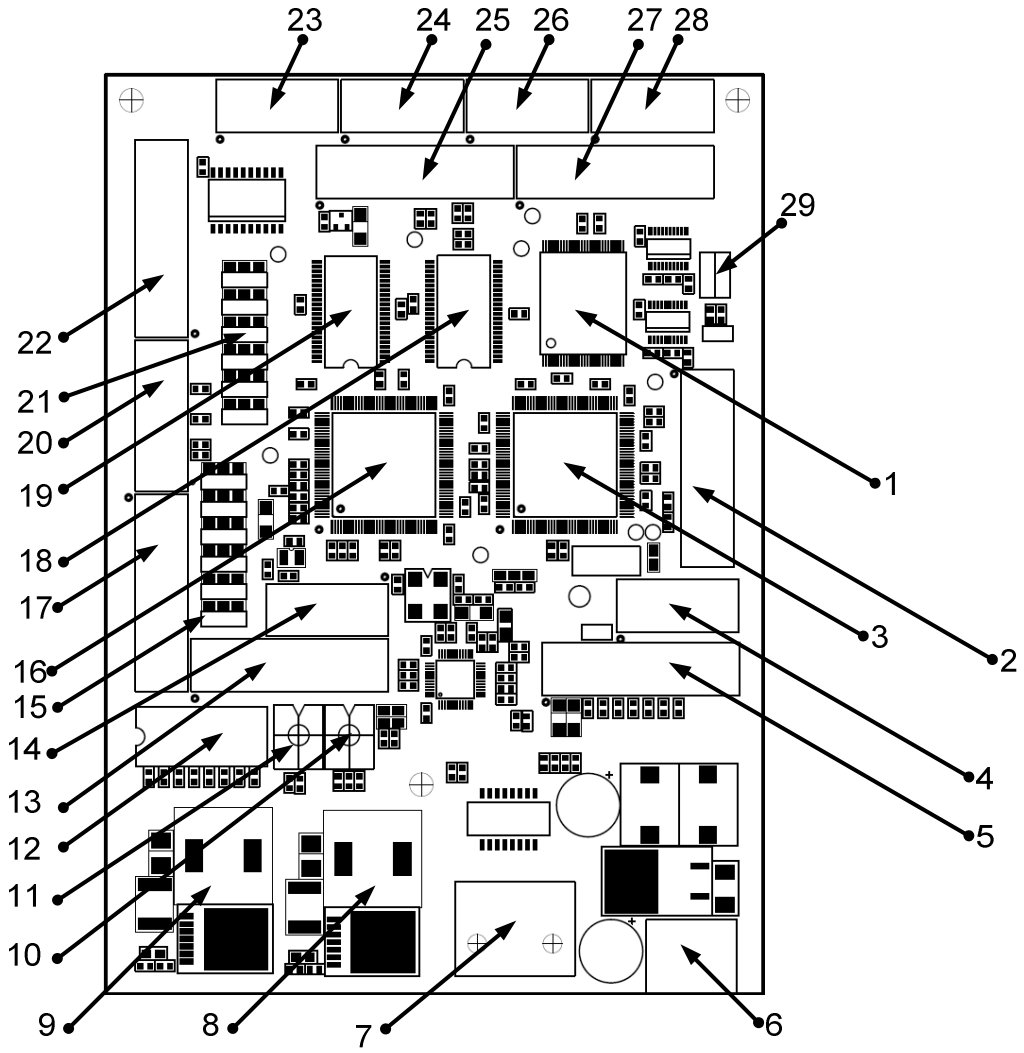


Figure 110 : Main parts of the board.

7.2.2 Clocks

The basic clock is driven with a crystal at 12MHz. The RTC can be driven either by an external crystal or by dividing the internal clock. However, as mentioned in the Users Manual “UM10211_1.pdf” p.569 / 684, in case the external crystal is used, the VBAT pin should be connected to a power source, ie either with + 3.3V, or with an external battery . The maximum voltage that VBAT is allowed to drive is + 4.6V, but we will prefer + 3.3V.

If the external crystal is not used, then the VBAT pin should be disconnected!! For this reason a solder jumper has been added. In addition to the solder jumper, two test points have been included so that in the future, battery power can also be included. Thus TP1 and TP2 must be close together.

7.2.3 Serial Ports

Serial interfaces are passed to the microcontroller pins via the CPLD. This is done to provide for alternative routing of serials ports to devices that speak serially on digital levels (0V - 5V or 0V - 3.3V) and not on the standard RS232.

RS232 ports use 10pin header connectors. The signal correspondence between header and normal DB9 connector is:

HEADER PIN	DB9 FEMALE PIN	SIGNAL	MICROCONTROLLER DIRECTION
1	1		
2	6		
3	2	TxD	Out
4	7	CTS	In
5	3	RxD	In
6	8	DTR	Out
7	4	DSR	In
8	9	RTS	Out
9	5	GND	Power

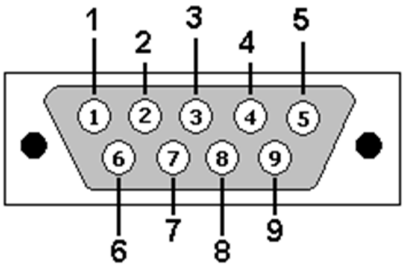
As a reminder, the flat cable is NOT a null-modem cable and the table is referenced to the development board and not the PC.

7.2.4 RS232 on DB9 (EIA/TIA 574)

Signal functions are described in detail in our Signal/pin primer. The column marked **Dir** shows the signal direction with respect to the DTE.

Pin No.	Name	Dir	Notes/Description
1	DCD	IN	Data Carrier Detect. Raised by DCE when modem synchronized.
2	RD	IN	Receive Data (a.k.a RxD, Rx). Arriving data from DCE.
3	TD	OUT	Transmit Data (a.k.a TxD, Tx). Sending data from DTE.
4	DTR	OUT	Data Terminal Ready. Raised by DTE when powered on. In auto-answer mode raised only when RI arrives from DCE.
5	SGND	-	Ground
6	DSR	IN	Data Set Ready. Raised by DCE to indicate ready.

7	RTS	OUT	Request To Send. Raised by DTE when it wishes to send. Expects CTS from DCE.
8	CTS	IN	Clear To Send. Raised by DCE in response to RTS from DTE.
9	RI	IN	Ring Indicator. Set when incoming ring detected - used for auto-answer application. DTE raised DTR to answer.



DB9 (EIA/TIA 574): View - looking into male connector

(male and female connector diagrams)

In serial communications the terminal end (PC) is called the Data Terminal Equipment (DTE) and the modem is called the Data Communications Equipment (DCE) as shown in the diagram of Figure 111.

Figure 111: Serial Communications with a modem.

RS-232 signals have a direction (in or out) depending on whether they are with respect to a DTE or a DCE. In all the pinout diagrams below the signal direction is with respect to the DTE (PC).

7.2.5 Board Programming Using Serial I/F

Flash Magic is used for programming using a serial interface. This program puts the µC in ISP mode by resetting via the DTR signal on the board (DSR on the PC) and keeping pin P2.10 at zero just after reset, via the CTS signal on the board (RTS on the PC). In this way it applies what is mentioned on page 585/684 of UM10211_1.pdf for ISP mode. This support is also available on our board via CPLD.

7.2.6 Ethernet PHY

The Ethernet Phy is the DP83848C of National Instruments. It is the same with the one used at the Keil development board. It is an LQFP48 package without JTAG interface.

The transformer used for the signal output is equipped with a common mode pin and uses an 1:1 primary to secondary ratio. It has the same characteristics with H1102 of Pulse, which is the datasheet recommended transformer.

MAC interface of LPC2378 connects to PHY using RMII interface.

RMII signals are listed in Table 13:

No	Signal Name	MAC	PHY
1	TXD_(1:0)	Output (O)	Input (I)
2	TX_EN	O	I with internal pull-down
3	RXD_(1:0)	I	O Pull down
4	RX_ER	I	O
5	MDC	O	I
6	MDIO	I/O	I/O needs 1.5KΩ pull-up
7	25MHz_OUT	I	O
8	CRS_DV	I	O

Table 13: RMII signals and directions.

LEDs signaling is as follows:

- Green led: link.
- Yellow led: activity.

7.2.7 Differential impedance of Ethernet Signals

Ethernet differential signals are routed to layer IN1. The nets starting from PHY and ending at the transformer are 6mils thick, the distance between them is 7mils, while finally the distance with the GND plane is 7mils. The differential resistance that results for these nets with this geometry and the dielectric thicknesses mentioned in the corresponding chapter of the text is 100.15Ω.

7.2.8 Outputs for servos

Servos have three cables with the colours and functions listed in the following list [34], [35]:

1. Black: GND
2. Red: Power (Vcc) which may vary between 4.8V to 6V.
3. Yellow:Control signal. Control signal is a PWM pulse with peak voltage of 3V – 5V. Pulse duration is
    a. 0.9ms minimum
    b. 2.1ms maximum

c. 1.5ms center
d. 50Hz frequency (20ms period)

Figure 112: Servo motor.

7.2.9 Experimental Data from Servo Measurements

Using an oscilloscope to measure pulses, Hitech remote control has the following output:

Power: 5.28V

Control voltage: 3.3V, with 19.92ms period (about 20ms ➔ 50Hz), minimum pulse width 900μs, maximum pulse width 2.09ms. One step at the knob of the remote control results to a 4.8μs change at the pulse-width.

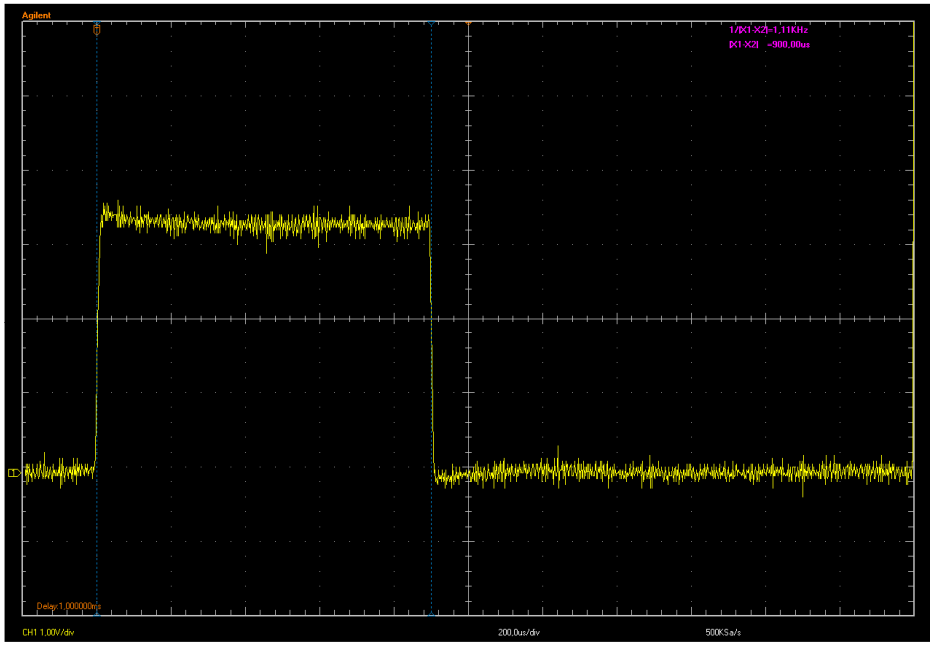


Figure 113: Minimum pulse width.

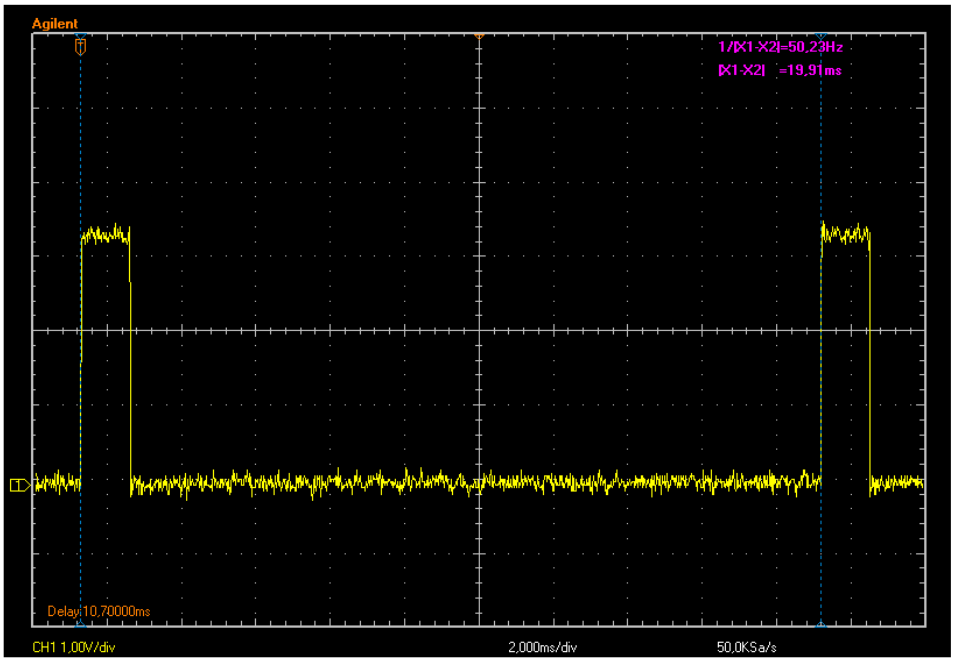


Figure 114: Signal period.

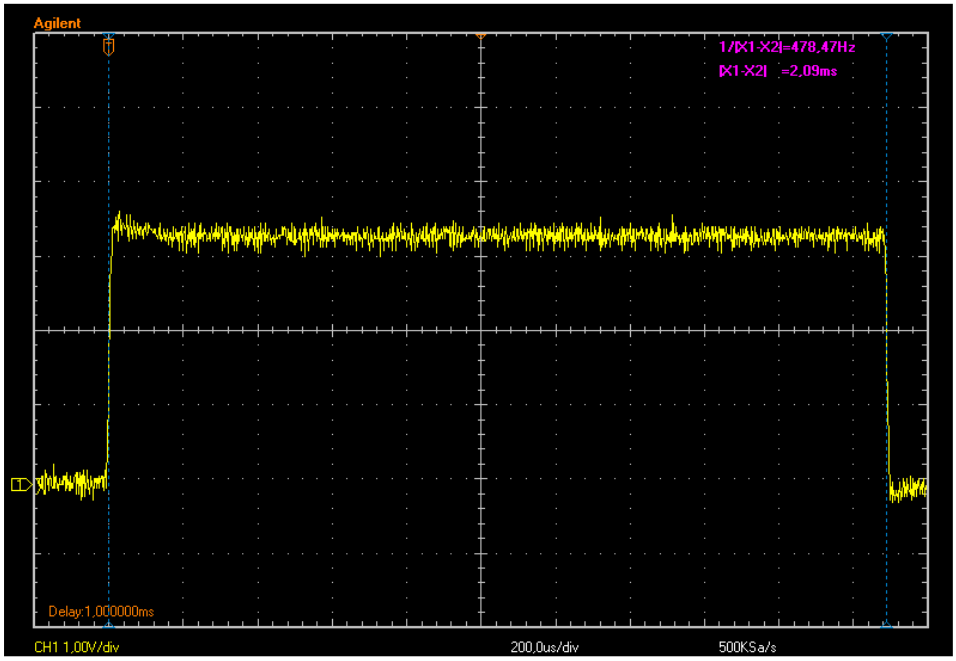


Figure 115: Maximum pulse width.

7.2.10Data from Hitec Servo Manual

"**Pulse Data**

All Hitec servos require 3-5V peak to peak square wave pulse. Pulse duration is from 0.9mS to 2.1mS with 1.5mS as center. The pulse refreshes at 50Hz (20mS).

Voltage Range

All Hitec Servos can be operated within a 4.8V-6V. range.
Only the HS-50 operates exclusively with 4 Nicad cells (4.8 volt).

Wire Color Meanings

On all Hitec servos the Black wire is 'ground', the Red wire (center) is 'power' and the third wire is 'signal'.

Direction of Rotation

All Hitec servos turn Clockwise direction (CW).**"**

7.2.11Info for Servo Interfaces (PWM)

The PWM interfaces offered by the board are 12 in total. 6 of them come directly from the CPLD. The remaining 6 are either inputs or outputs via buffer. More specifically, the initial design for them is as follows:

The 6 that are connected directly to the CPLD will be the ones that will control the servos of the boat, while the other 6 will be the ones to which an external remote control receiver will be connected.

7.2.12DBGEN pin

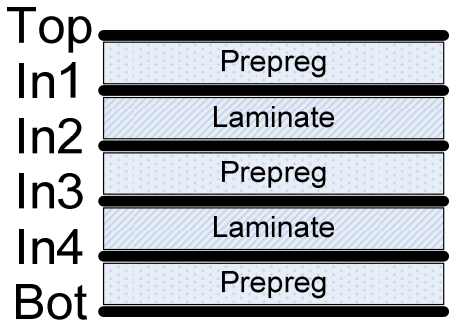
The DBGEN pin is to determine the operation of the JTAG interface. When it is 1, then the JTAG interface acts as a debug port. When it is 0, it works like a normal JTAG. The card is set as the default setting, to be in debug mode (1).

7.2.13Layer Structure

Layer	Utilization
Top	Fanout, GND
In1	Horizontal
In2	GND
In3	Power (Input, +3.3V, +5V)
In4	Vertical
Bottom	Fanout, GND

7.2.14Stackup

The following image shows the layer stackup of the card. Layers come as two copper sheets joined with laminate. The sheets are glued together with prepreg. For the outer layers the process is a little different. They are single copper foils, which are glued to the rest with prepreg.



The total thickness of the card we want is 1.6mm. Therefore since copper is 18µm thick (0.5oz), if we divide the thicknesses of the dielectrics we will have
Dielectric thickness = (1.6mm - 6x18µm) / 5 = approximately 300µm
With 300µm dielectric thickness the total card thickness comes out to be 1,608mm.

7.2.15Critical Signals for Layout

Num	Signal	Note
1	ETH_REF_CLK	
2	ETH_TX_EN	
3	ETH_TXD0	
4	ETH_TXD1	
5	ETH_RXD0	
6	ETH_RXD1	
7	ETH_CRS	

8	ETH_RX_ER	
9	NET00121	(TD+)
10	NET00122	(TD-)
11	NET00049	(TD+_CN)
12	NET00096	(TD-_CN)
13	NET00124	(RD+)
14	NET00125	(RD-)
15	NET00130	(RD+_CN)
16	NET00131	(RD-_CN)
17	CS0	
18	CS1	
19	OE	
20	WE	
21	SRAM1_CS	
22	SRAM_RD	
23	SRAM_WR	
24	SRAM2_CS	
25	FLASH_WR	
26	FLASH_RD	

7.2.16Floating Pins

Microcontroller and CPLD have many floating IOs in the design. For CPLD the features "Bus-hold circuitry on all user pin inputs" are mentioned as well as the following text:

«Each IOB also provides for bus-hold circuitry (also called a“keeper”)that is active during valid user operation. The bus-hold feature eliminates the need to tie unused pins either high or low by holding the last known state of the input until the next input signal is present. The bus-hold circuit drives back the same state via a nominal resistance (R_{BH}) of 50k ohms. See Figure 13. Note the bus-hold output will drive no higher than V_{CCIO} to prevent overdriving signals when interfacing to 2.5V components.»

This text is an excerpt from page 12/17 of "DS054_XC9500XL_family.pdf"

Regarding the microcontroller, the following abstract is indicative of the practice to be followed: «The PINMODE registers control the on-chip pull-up/pull-down resistor feature (the mode)
for all ports. The on-chip pull-up/pull-down resistor can be selected for every pin regardless of the function on this pin with the exception of the I^2C pins and the USB pins (see Section 9–5.13). Two bits are used to control the mode of a port pin. Bits are reserved for unused pins as in the PINSEL registers.»

The Microcontroller has a register through which SW assigns the pull resistors to the pins. Its default state is pull-up. From the above text it appears that in addition to the I2C pins, which we have internal pull-up, there are also P0 [29] -P0 [31] which need external pull-up. «**Remark:** The pin mode cannot be selected for pins P0[27] to P0[31]. Pins P0[27] and P0[28] are dedicated I2C open-drain pins without pull-up/down. Pins P0[29], 0[30], P0[31] are USB specific pins without configurable pull-up or pull-down resistors.»

7.2.17Analog Inputs/Outputs

In the drawing there are two connectors whose signals terminate in analog inputs/outputs of the microcontroller. These connectors are J35 and J36. All pins going from these connectors to µC are analog inputs except pin9 of the J35 which can be either analog In or analog Out.

7.2.18RAMs and ROMs/Flash Memories

There are two types of memory on the card: SRAM and FLASH. The CS0 of µC goes to FLASH, while CS1 goes to CPLD and from there it can be routed to either SRAM1 (U3) or SRAM2 (U4). However, CS1 is also used to access the CPLD.

FLASH accesses are done directly using CS0. Memory or CPLD accesses are selected via Pin CPLD_IO24. When this is 0 then SRAMs are accessed. Which SRAM is selected has to do with which address is given to the CPLD at bit A16 (net CPLD_A16).

Inside the CPLD there is an 8-bit register which stores the Address bits A (23:16).

7.2.19Power Input

The system can be supplied with voltages from 8.5V to 30V. Its switching power supplies operate from 8V and above, but 0.5V is a safety limit for voltage drop from the Schottky diode to the input.

7.2.20Installation of Software Tools

For the creation of code will be a first attempt with the YAGARTO tools which are located at http://www.yagarto.de/. This address was found through http://www.freertos.org/ which FreeRTOS will use as Real Time operating system.

For the first experiments all the files are in the "D: \ Programs \ ARM_Toolchain_Yagarto" location, while the Eclipse projects are in the "D: \ Work \ Software \ Eclipse_Projects" folder.

In the menu project> properties> (list) C / C ++ General> Paths and Symbols, I changed the include paths.

The final development was done using Rowley Crossworks IDE.

7.2.21Notes for PCAD

7.2.21.1 DRILL TABLE GENERATION

The drill table is generated by DrillTab.exe which I must have added first from <menu> →Utils→Customize. Before running the above command, I must have first assigned drill symbols using
<menu>→File→Print→DrillSymbols→Automatic Assign.
To create drill table output use:

1. File – export – NcDrill
2. Set Output Files
    a. Output files for All Holes
    b. All Physical Layers
    c. Extension EXL
    d. All Holes
3. Tools
    a. Auto (for auto assigning of shapes for each drill)
4. N/C Drill Format
    a. Output Units→Inches
    b. Output Code type→ASCII none
    c. Zero suppression→Leading

7.2.21.2 CREATING GERBER FILES

File → Export → Gerber

1. At setup output files+
    a. All Xoffset and Yoffset are 0. Setup for generation of the following files:

Output File	Layers	Tick Boxes
*.tslk	Topsilk + Board + Title	RefDes
*.tpaste	TopPaste + Board + Title	Pads
*.tmask	TopMask + Board + Title	Pads, Vias
*.top	Top + Board + Title	Pads, Vias
…	…	…
*.bot	Bottom + Board + Title	Pads, Vias
*.bmask	Bot Mask + Board + Title	Pads, Vias
*.bpaste	Bot Paste + Board + Title	Pads
*.bslk	Botsilk + Board + Title	RefDes
*.master	Signal layers, board, titles, drill, Notes, dimensions,	Pads, Vias, Pad/via holes, output drill symbols 80mil, all holes

2. Apertures → Auto → close

3. Drill Symbols → Automatic Assign → close
4. Gerber Format → Inches, 4.4, include aperture definitions.

7.2.21.3 DESIGN RULES

Copper pour:

- Line width 5.2mil
- Pad Thermals : Spoke width 9mil
- Via thermals : Direct connect
- Backoff : “Use design rules”. This holds for external fills.

Design Rules:

- Pad-to-Pad : 10mils
- Pad-to-Line : 10mils
- Pad-to-Via : 10mils

7.3 Board Operation

To operate the board, the next steps must be followed: first of all it must be powered by a supply that provides 8.5V to 30V. Next, the supporting CPLD must be programmed. This is a necessary step, since the CPLD handles microcontroller’s reset and also supports microcontrollers programming. Finally a valid “.hex” file must be downloaded through serial interface 1. The sections below analyze the steps mentioned and add more analytical information.

7.3.1 Powering the board

As already mentioned, the board must be powered by a source providing 8.5V to 30V and at least 1W if the board is to be operated stand – alone. If the board will source other circuits, the other circuitry consumption must be considered in addition to 1W.

Figure 116 shows the connections needed for the power connector. Chassis GND is used only for ESD protection of the Ethernet Connector. If the Ethernet connector will be used, this net should be connected to the system’s chassis. If not, short it with GND by soldering J54 solder jumper.

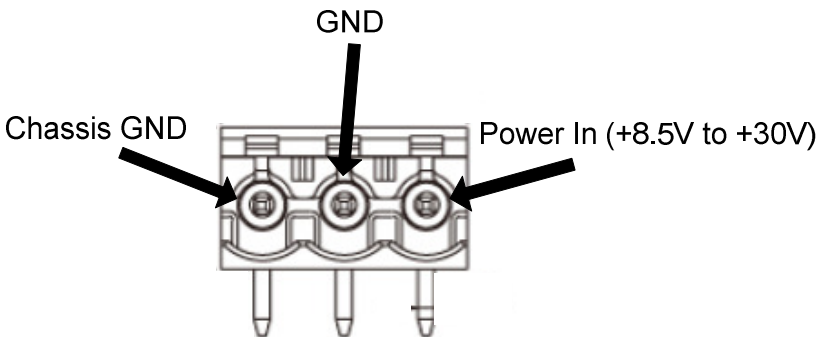


Figure 116 : Power connector.

7.3.2 Programming the CPLD

In order to program the CPLD a Xilinx Platform Cable USB II is preferable (HW-USB-II-G). To acquire one, see links http://www.xilinx.com/onlinestore/program_solutions.htm#pc and http://avnetexpress.avnet.com/store/em/EMController?langId=-1&storeId=500201&catalogId=500201&action=products&N=0&mfr=XLX&hrf=http://www.xilinx.com/onlinestore/program_solutions.htm&term=HW-USB-II-G .

Having acquired the Platform cable, current Xilinx ISE must be installed on a PC. At the time this document was written, the current version was 12.3. The current version can be found in http://www.xilinx.com/support/download/index.htm . The ISE needed is freely available for download.

Having installed the ISE, startup "iMPACT". This is found in <Start> → <All Programs> → <Xilinx ISE …> → <Accessories> → iMPACT. After starting iMPACT the following dialog box appears:

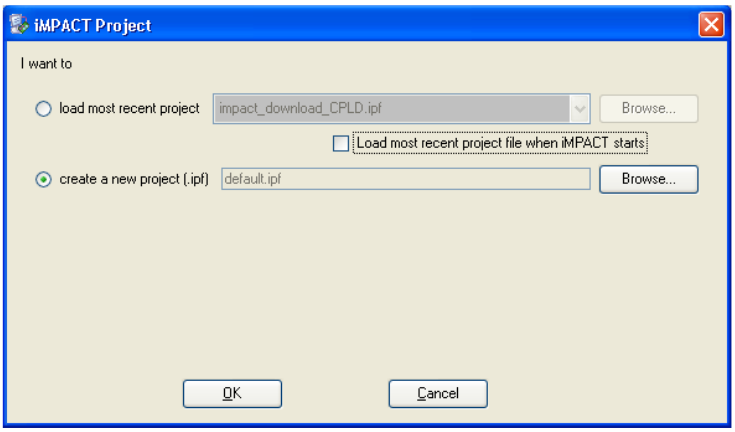


Figure 117 : Initial iMPACT dialog box.

Select create a new project. The following dialog box then appears. Leave setting as is.

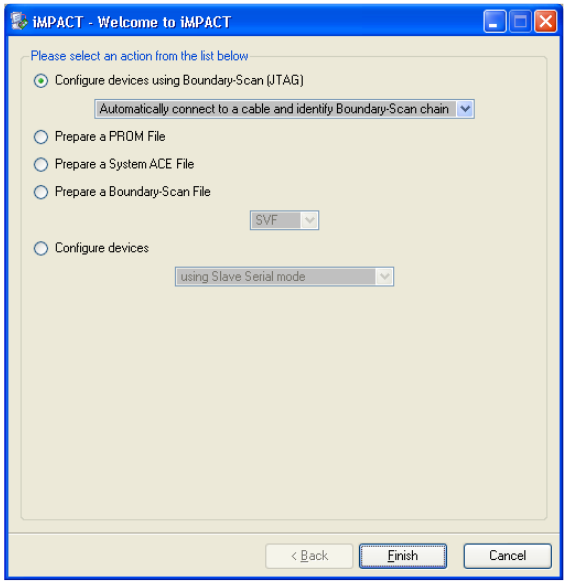


Figure 118 : iMPACT JTAG Configuration dialog box.

Platform cable will perform a scan at the JTAG chain and will detect any connected devices, provided they are working properly and the platform cable is connected correctly. Platform cable connections will be presented in the following subsection.

After the device scan, the chain of devices presented will have the following form presented in boundary scan tab shown in

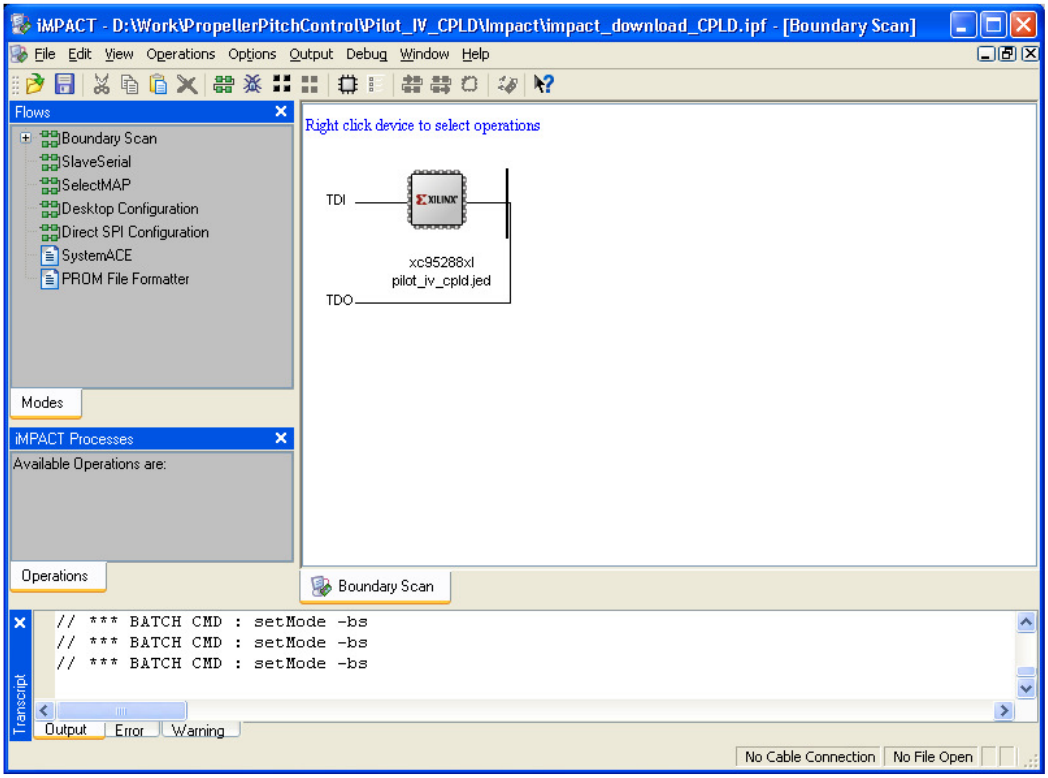


Figure 119 : iMPACT JTAG chain viewer.

At this point right-click on the device presented, choose “Assign new configuration file..” and select the “.jed” file to download to CPLD.

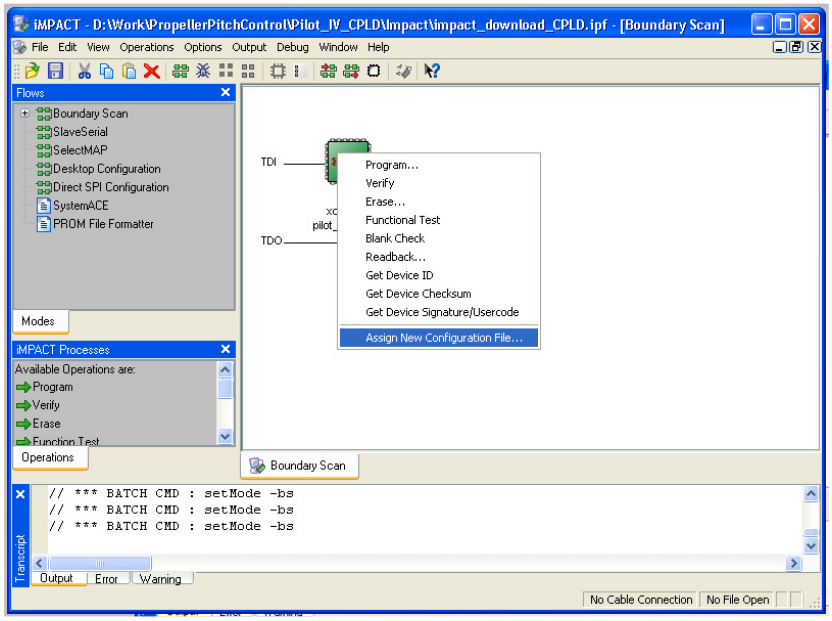


Figure 120 : iMPACT CPLD source file selection.

Right click again the device and select “Program” to download the configuration file. From the next dialog box make the choices you prefer regarding overwrite, verify etc and program the device.

7.3.2.1 PLATFORM CABLE JTAG CONNECTIONS.

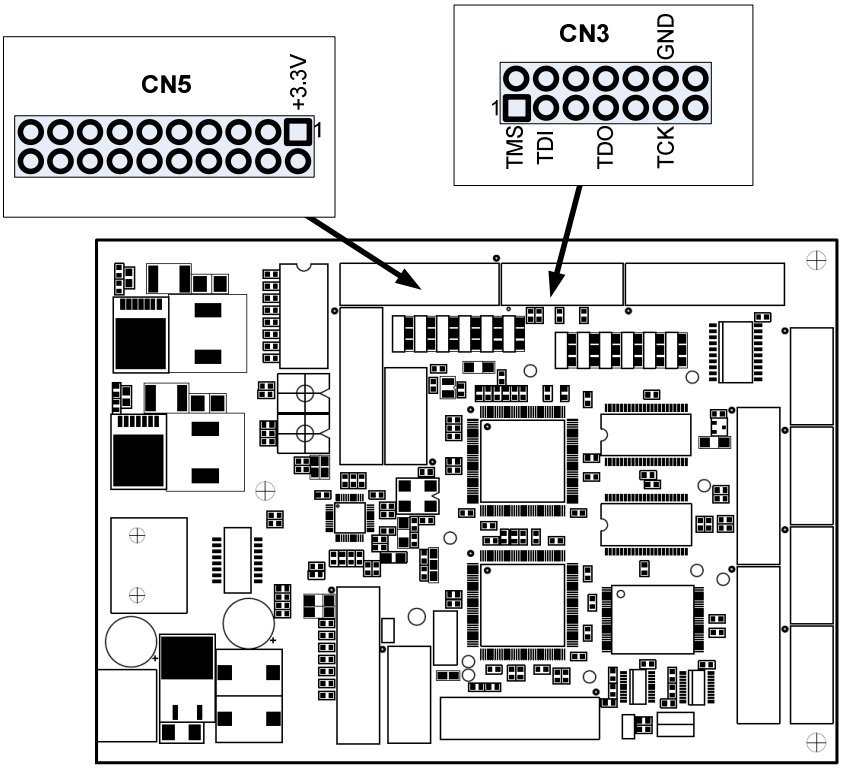


Figure 121 : Where to attach a JTAG cable in order to program the CPLD.

7.3.3 Software Environment.

One software environment that can be used to develop applications for this board is SUN’s Eclipse. Analytical instructions for downloading and installing Eclipse and other necessary tools can be found in the following link: http://www.freertos.org/portlpc2368_Eclipse.html .

7.3.3.1 CODE DOWNLOAD.

When an application has been compiled from the software tools, it can be downloaded to the board through Serial Port 1, using “Flash Magic” application. This application can be downloaded from the following link: http://www.flashmagictool.com/ . Before starting a download, switch No1 of SW3 DIP switch must be at on position. This permits automatic download of the code by just clicking “Start” button at FlashMagic tool.

7.4 Board Resources

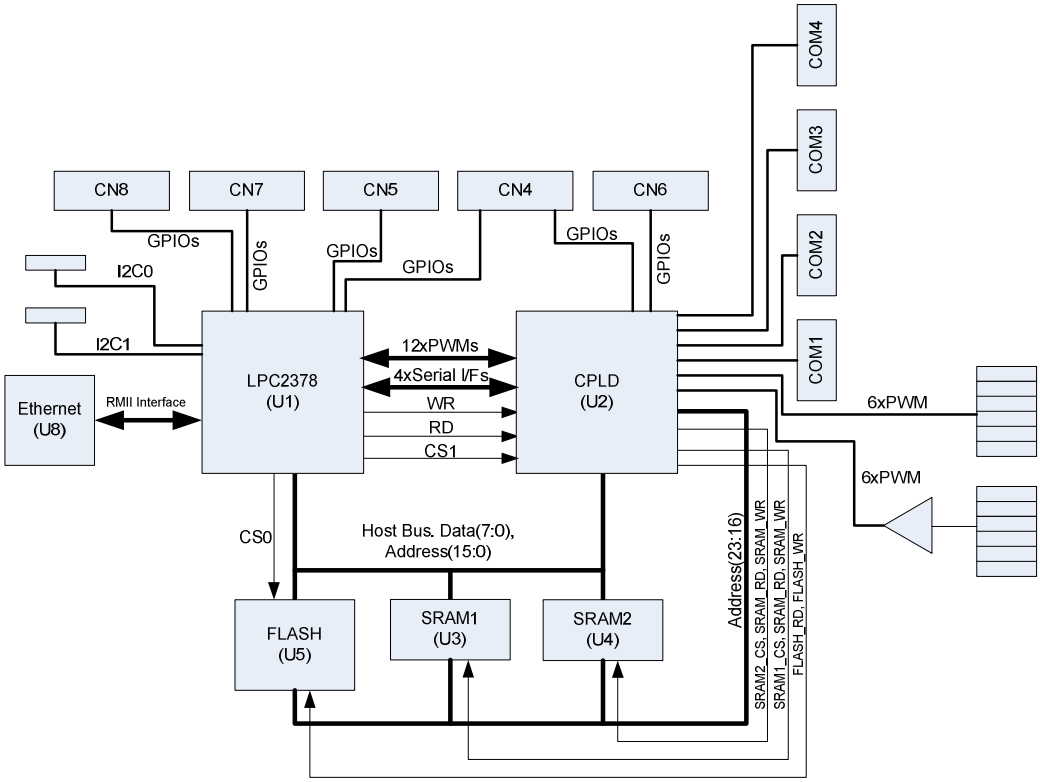


Figure 122 : Board Architecture.

Figure 122 presents the main parts of the system's architecture. The heart of the system is LPC2378 microcontroller (U1). For information about LPC2378 internal structure and operation refer to the NXP site documentation (datasheet, user manual, errata etc). Connectors CN4, CN5, CN7 and CN8 contain 25 GPIOs of the microcontroller. Besides functioning as GPIO, the microcontroller pins can perform other pre-defined interface operations such as I2C, SPI etc. At the early stages of design, carefully plan the interfaces needed in order to allocate the appropriate resources.

Referring again to "Figure 122", memory and CPLD connect to U1 through an 8-bit parallel bus. The bus is controlled by the EMC (External Memory Controller) of LPC2378. EMC supports two chip-select signals and 16-bit addressing (bits 15:0). Chip select 0 (CS0) connects to a 128Mb (= 16MB) flash, used for non-volatile data storage. Since 16-bit addressing is not enough for managing 16MB, CPLD is used as an address expander by providing 8 more address bits (23:16).

Chip select 1 (CS1) is used to access the two SRAM devices and CPLD. Address allocation for CS1 is done by CPLD. Memory mapping for both chip selects will be given in the following section.

7.4.1 Memory map

Board external address space is quite flexible, since the address decoding is CPLD based. Though the present mapping satisfies all possible needs and there is no obvious reason for such alterations. According to the present mapping the addresses are allocated as follows:

Resource	CS	Ext. Address Range	µC Int. Address	Size
FLASH	0	0x0000000 to 0x0FFFFFF	0x8000 0000 - 0x8000 FFFF Access CPLD for bits 23:16 (0x0000000 to 0x0FF0000)	16MB
CPLD	1	0x0000000 to 0x000000F	0x8100 0000 to 0x8100 000F. Note that bit 24 of external address (net cpldn_sram_sel) is 0.	16B
SRAM1	1	0x1000000 to 0x107FFFF	0x8100 0000 to 0x8100 FFFF. Access CPLD for bits 23:16 (0x1000000 to 0x1070000). Note that bit 24 of external address (net cpldn_sram_sel) is 1.	512kB
SRAM2	1	0x1080000 to 0x10FFFFF	0x8100 0000 to 0x8100 FFFF. Access CPLD for bits 23:16 (0x1080000 to 0x10F0000). Note that bit 24 of external address (net cpldn_sram_sel) is 1.	512kB

Table 14 : Address space (external).

At Table 14 net "cpldn_sram_sel" is mentioned a lot of times. This net notifies when 0 that when CS1 is active, the address present at the bus targets access to CPLD internal registers. When 1, the accesses of CS1 are directed to SRAM ICs.

Another important point presented by Table 14 is that the address used in the microcontrollers address space (the one used by software), is different than the one presented in the external address bus. This happens because software accesses the external bus using the LPC2378's EMC hardware, which responds to a specific internal address space. This explains the presence of column "**Ext. Address Range**" and "**µC Int. Address**" of Table 14. When writing software the "**µC Int. Address**" column is

used. On the contrary, when the hardware is probed, column "**Ext. Address Range**" applies.

7.4.2 CPLD Register File

The following table lists the registers available at CPLD and their addresses:

Resource	Ext. Address	µC Int. Address
EXTENDED_ADDRESS_REG	0x0000000	0x8100 0000
UC_PWM_CONF_REG	0x0000001	0x8100 0001
WATCHDOG_CONF_REG	0x0000002	0x8100 0002
CPLD_VERSION_REG	0x0000003	0x8100 0003
DIP_SW_STATUS_REG	0x0000004	0x8100 0004
ACCESS_TEST_REG	0x0000005	0x8100 0005
PERIPHERALS_RST_REG	0x0000006	0x8100 0006
PWM1_VALUE	0x0000007	0x8100 0007
PWM2_VALUE	0x0000008	0x8100 0008
PWM3_VALUE	0x0000009	0x8100 0009
PWM4_VALUE	0x000000A	0x8100 000A
PWM5_VALUE	0x000000B	0x8100 000B
PWM6_VALUE	0x000000C	0x8100 000C
PWM_AN_CTRL	0x000000D	0x8100 000D
PWM1_STATUS	0x000000E	0x8100 000E
PWM2_STATUS	0x000000F	0x8100 000F

Table 15 : CPLD register file.

Table 16 provides a brief description of the functionality of each register:

Resource	Description
EXTENDED_ADDRESS_REG	The most significant register in the CPLD. The value written in this register is appeared as the bits 23:16 in the address bus. Operations available: Read/Write.
WATCHDOG_CONF_REG	Register that enables the external watchdog of the processor (implemented in the CPLD). Operations available: Read/Write.
CPLD_VERSION_REG	Read only register that shows the CPLD version.
DIP_SW_STATUS_REG	Register that contains the values of the four bits of SW3 DIP_switch that connect to CPLD. Operations available: Read.
ACCESS_TEST_REG	Register used to test the access to CPLD

PERIPHERALS_RST_REG	Register that is used to enable reset for a peripheral

Table 16 : CPLD registers description.

Registers that are not described here are either not implemented here or application specific.

7.5 System Connections

Figure 123 shows in detail the signals at each connector. Details for the signals functionality can be found using the block diagram and LPC2378 user manual. Note that all signals starting with “CPLD_” are general purpose programmable IOs of the CPLD.

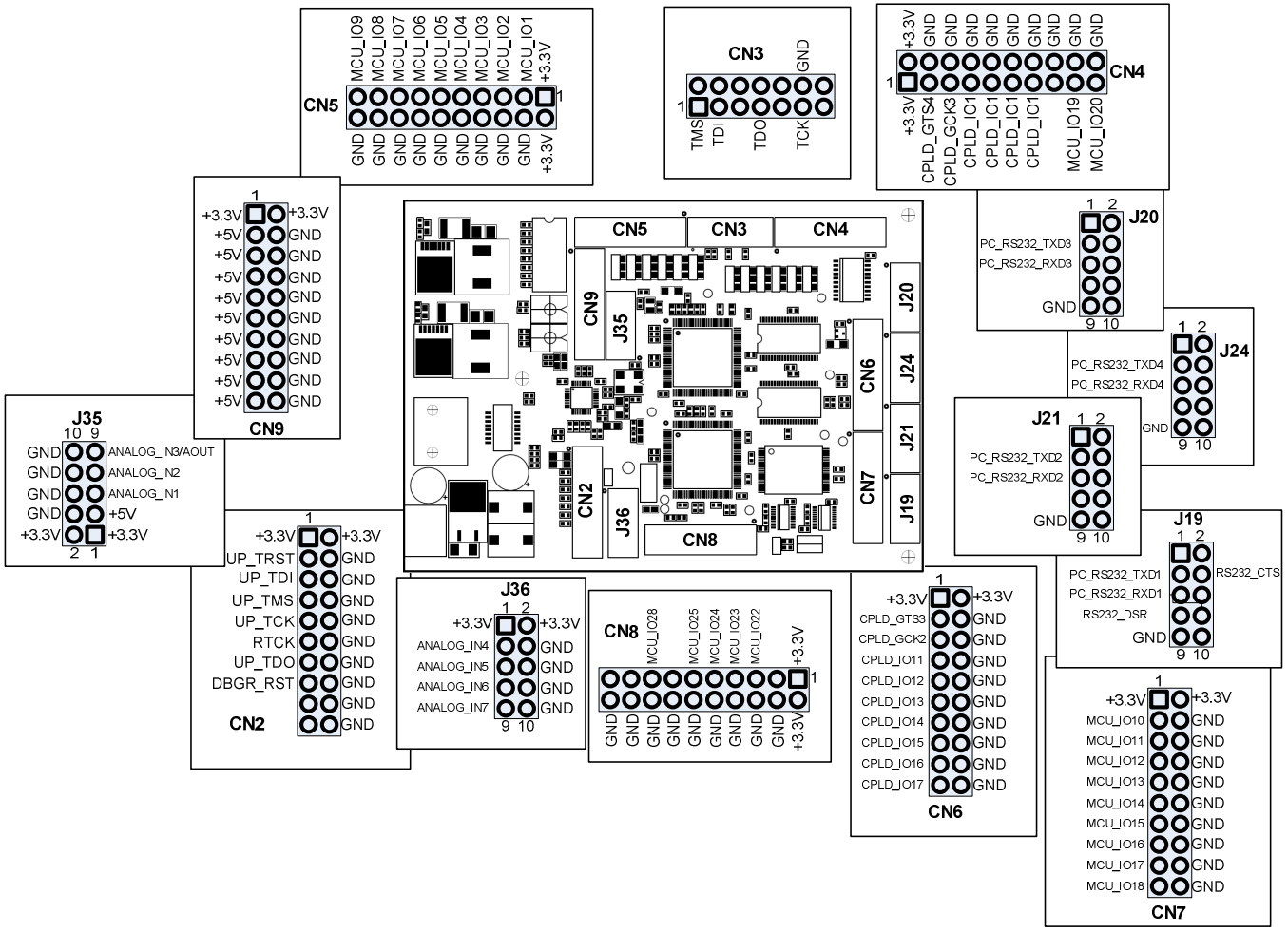


Figure 123 : System Connections.

7.6 CPLD

The CPLD in the autopilot circuit is from the Xilinx 9500XL family. The reason it came in is the flexibility it adds to the circuit. A simple example of this flexibility is that each PWM Out of the pilot can be driven either remotely or by the µC without the burden of the µC either in terms of processing or in terms of inputs / outputs.

VHDL language and Xilinx ISE 9.2i development environment were used to design the CPLD logic. The following sections describe the logic constructed in the CPLD as well as its internal structure.

Instructions for organizing Xilinx Simulation Libraries can be found at
http://www.xilinx.com/itp/xilinx10/isehelp/ise_c_simulation_libraries.htm

7.6.1 Pilot_IV_CPLD

This section describes the CPLD as an entity and its inputs and outputs. The capabilities of the CPLD in the circuit are as follows:

1. Operation of the power-on and reset signals of the system.
2. Buffering the driving signals of the µC-Bus for signal integrity reasons.
3. Increase the width of the address bus of µC.
4. Complexity of PWMs between remote control and µC.
5. Support for µC programming.

The following sections describe the above features separately.

7.6.2 Handling power-on και resets

When the system starts, there is a voltage monitoring circuit which generates a reset signal. The signal lasts 200ms from the recovery of voltage. This reset is given to the CPLD, which distributes it to ICs that have a reset input. These are:

- The Microcontroller.
- The Ethernet physical interface (PHY).
- FLASH memory.

In addition to the simple redistribution of signals, through the CPLD it is possible, using an internal register, to reset through software two of the peripherals of the µC. FLASH and Ethernet PHY. The CPLD with an internal mechanism explained in the chapter Reset_gen_unit, ensures the correct duration of each signal, stripping a task requiring strict timing from the software.

7.6.3 Buffering Control Signals

To avoid the complexity in routing the control signals of the µC-Bus which would lead to degradation of the quality of these signals, the CPLD acts as a digital buffer. The signals supported in this way are the read and write of the µC peripheral memories.

7.6.4 Increase of µC-Bus Address Space

The parallel bus of the microcontroller used provides 16-bits of address space. This is not enough for supporting the SRAMS and FLASH. To fully utilize these devices, 24-bit addresses are required. CPLD has an internal register that stores the missing bits. It also has a chip select generation mechanism for accessing all peripherals.

7.6.5 Multiplexing of PWM Signals

The autopilot has the ability to manage 12 PWM inputs or outputs. More specifically, 6 of the PWMs are completely independent and can be either inputs or outputs, while the other six are either all inputs or all outputs.

In the present embodiment both hexades are considered inputs. Six from the µC, while the remaining six from the remote control connected to the pilot. The CPLD multiplies these inputs and gives one output in pairs. Which one will be selected is determined by a register accessed by the µC-Bus..

7.6.6 Support for µC Programming

The serial port ending in the connector named J19 is used to download the µC code. To perform the programming from PC, it helps to have communication of the serial port with the reset and EINT0 signals of µC. The CPLD undertakes to route these signals when the programming process is activated. The process is activated via switch 1 of the DIP switch located in the circuit (SW3).

7.6.7 Pinout

The following table lists the CPLD pins and provides a brief description of their functionality.

PIN	DIRECTION	DESCRIPTION
resetn	In	System reset.
ext_wdg_restart	In	Not used yet. To be used for restarting of external watchdog.
reset_flashn	Out	FLASH reset.
reset_ucn	Out	µC reset.
reset_eth_phyn	Out	Ethernet PHY reset.
clk_50MHz	In	50MHz clk In.
cpld_gck2	In	Unused signal
cpld_gck3	In	Unused signal

cs1n	In	Chip select in. This input is directed either to the internal chip select or to the SRAMS chip select depending on the cpldn_sram_sel pin value.
oen	In	Read for data bus. Active low.
wen	In	Write for data bus. Active low.
d (7:0)	In/Out	data bus (8-bit).
a (15:0)	In	16 LSbits address pins of µC-bus.
cpld_a (23:16)	Out	The 8 MSb of the µC-Bus address.
flash_vpen	Out	Output that permits the FLASH write access functionality. For the moment always 1, allowing write access.
flash_wrn	Out	Write signal of FLASH IC. It is wen signal buffered for signal integrity reasons.
flash_rdn	Out	Read signal of FLASH IC. It is oen signal buffered for signal integrity reasons.
sram1_csn	Out	Chip select for SRAM 1.
sram2_csn	Out	Chip select for SRAM 2.
sram_wrn	Out	Write signal for both SRAMs. Buffered copy of wen, for signal integrity reasons.
sram_rdn	Out	Read signal for both SRAMs. Buffered copy of oen, for signal integrity reasons.
cpldn_sram_sel	In	Input that separates CPLD form SRAM accesses on µC bus. When 0 the access targets CPLD.
eint0n	Out	Future use output regarding normal µC operation. It is used though during µC programming phase. When uc_prog_en is 0, then uart1_cts is copied at this output. It is used toanable PC software to enable the programming of mode of µC in order to download firmware.
eint1n	Out	Unused. For future use.
eint2n	Out	Unused. For future use.
eint3n	Out	Unused. For future use.
mcu_pwm (6:1)	In	PWM inputs from µC.
cpld_pwm (6:1)	Out	PWM outputs. Using an internal register it is selected if this input will be driven from radio

		control or the µC (from the autopilot control algorithm).
radio_pwm(6:1)	In	PWM inputs from the radio remote control.
pwm_buff_en	Out	Output that activates the buffer of radio_pwm signals.
pwm_buff_dir	Out	Output that changes the direction of radio_pwm signals.
push_button_in	In	Input from SW2 button. When this button is pressed, this input is zero.
disable_5V	Out	Output that disables the 5V regulator when 0.
led_out	Out	Out which when it is 1, illuminates the yellow led of the board. This led is currently used to indicate that the CPLD supports the µC programming mode. It also lights up whenever the SW2 button is pressed, just to indicate that the CPLD is programmed.
uart1_txd	Out	Transmit RS232 signal from CPLD to RS232_1 port. This pin copies the data coming to Pin up_txd1.
uart1_rxd	In	Download (to CPLD) of the RS232_1 port. This pin is copied to output up_rxd1.
up_txd1	In	Input from µC. Copied to pin uart1_txd.
up_rxd1	Out	Out to µC. This pin copies everything that reaches uart1_rxd.
uart2_txd	Out	Transmit RS232 signal from CPLD to RS232_2 port. This pin copies the data coming to Pin up_txd2.
uart2_rxd	In	Download (to CPLD) of the RS232_2 port. This pin is copied to output up_rxd2.
up_txd2	In	Input from µC. It is copied to pin uart2_txd.
up_rxd2	Out	Out to µC. This pin copies everything that reaches uart2_rxd.
uart3_txd	Out	Transmit RS232 signal from CPLD to port RS232_3. This pin copies the data coming to Pin up_txd3.
uart3_rxd	In	Download (to CPLD) of the RS232_3 port. This pin is copied to output up_rxd3.

up_txd3	In	In by µC. Copied to pin uart3_txd.
up_rxd3	Out	Out to µC. This pin copies everything that reaches uart3_rxd.
uart4_txd	Out	Transmit RS232 signal from CPLD to RS232_4 port. This pin copies the data coming to Pin up_txd4.
uart4_rxd	In	Download (to CPLD) of the RS232_4 port. This pin is copied to output up_rxd4.
up_txd4	In	In by µC. Copied to pin uart4_txd.
up_rxd4	Out	Out to µC. This pin copies everything that reaches uart4_rxd.
uart1_dsr	In	In which currently serves only the programming phase of µC. In this phase it checks its reset.
uart1_cts	In	In which currently serves only the programming phase of µC. In this phase it controls the EIN0 input of µC. When after reset In this is low, µC enters programming mode.
uc_prog_en	In	In which is controlled by the first switch of the DIP switch. Allows the passage of programming signals to the µC.
enable_5Vn	In	In which is controlled by the second switch of the DIP switch. When it is 0 (switch in the ON position), it activates the 5V regulator of the circuit.
cpld_gts3	Out	Unused. For future use.
cpld_gts4	Out	Unused. For future use.
cpld_io1	Out	Unused. For future use.
cpld_io2	Out	Unused. For future use.
cpld_io3	Out	Unused. For future use.
cpld_io4	Out	Unused. For future use.
cpld_io5	Out	Unused. For future use.
cpld_io8	Out	Unused. For future use.
cpld_io9	Out	Unused. For future use.
cpld_io11	Out	Unused. For future use.

cpld_io12	Out	Unused. For future use.
cpld_io13	Out	Unused. For future use.
cpld_io14	Out	Unused. For future use.
cpld_io15	Out	Unused. For future use.
cpld_io16	Out	Unused. For future use.
cpld_io17	Out	Unused. For future use.
cpld_io25	Out	Unused. For future use.
cpld_io26	Out	Unused. For future use.
cpld_io27	Out	Unused. For future use.

7.6.8 register

This unit is a register [36], which through a generic command has the ability to reset to give any output value we want. Its function is to store its input at the output on each rising edge of the clock.

PIN	TYPE	DESCRIPTION
Rst	In	Reset input.
Clk	In	Clock input.
Data_in	In	Data input.
Data_out	Out	Data output.

Note that the size of the register is dynamic and depends on the size of the input and output data that will be assigned to it during instantiation in the code.

7.6.9 Signal_debouncer

This unit removes any high frequency noise from the input signal and synchronizes it with the input clock. Used to handle asynchronous signals through modern circuits.

PIN	TYPE	DESCRIPTION
Clk	In	Clock input.
sig_in	In	Input signal (asynchronous).
sig_out	Out	Output signal (synchronous).

To pass the input signal to the output, it must remain constant for three 50MHz clocks.

7.6.10Pwm_mux

The PWM signal multiplexer is used to select between signals from the remote control or μC to drive each PWM output. The CPLD has six independent multiplexers (one for each output), the selection of which is changed by command from the μC-Bus.

As a unit it consists of the signals shown in the Table 17:

PIN	TYPE	DESCRIPTION
in0	In	In 0. Out becomes a copy of this input when the signal sel = 0.
in1	In	In 1. Out becomes a copy of this input when the signal sel = 1.
sel	In	The input selection signal to be copied to the output.
output	Out	The Digital Out of the multiplexer.

Table 17: PWM_MUX unit I/Os.

7.6.11pwm_sampl_clk_gen

This unit generates the clock through which the PWM inputs are detected. Its frequency is 8 times faster than that of PWM.

PIN	TYPE	DESCRIPTION
rst	In	Reset. When it is 0, the production of the clock stops.
Clk	In	25MHz clock input.
Clk_out	Out	Output signal. Frequency 1 pulse per 1.2μs.

Table 18: pwm_sampl_clk_gen I/Os.

The clock for PWM came up with the following reasoning: The pulses received by the system from the remote control are 50Hz with a minimum width of 0.9ms and a maximum of 2.1ms. The step by which this range is analyzed is 4.8μs. Therefore from 0.9ms to 2.1ms we have 1.2ms which can hold 250 steps. So to describe the pulse width we need an 8-bit register. To be more accurate in our measurement, it is better to take 4 samples of the pulse during one step and divide the result by 4 with a right slide of 2 bits.

7.6.12pwm_analyzer

This unit is responsible for analyzing the PWM pulse and the width that this pulse represents in steps.

PIN	TYPE	DESCRIPTION
rst	In	Reset. When it is 0, the unit is initialized.
Clk_25MHz	In	25MHz clock input.
Pwm_clk	In	Pulse detection clock (Out of unit pwm_sampl_clk_gen). Frequency 1 pulse per 1.2μs.
Pwm_in	In	PWM input.
Read_sample_notification	In	A signal that informs about the creation of a new sample.

Sample_data	Out	The new sample.
Err_type	Out	In case of an error, it stores the identity of the error.
Err	Out	Out which goes to a register that stores the error ID.
Read_pwm_interrupt	Out	Interrupt to read the result of PWM pulse width detection.

7.6.13Reset_gen_unit

This unit is responsible for the correct handling of resets by the CPLD. Starting with the study of its inputs and outputs:

PIN	TYPE	DESCRIPTION
resetn	In	System reset input.
Clk	In	50MHz clock input.
pwm_clk	In	Input of a divided 50MHz clock. This input is used to avoid re-implementing long counters for generating lower frequency clocks as the ones used to create delays for various resets.
ext_uc_reset	In	Reset output for µC, from the circuitry that supports µC FLASH programming.
if_reset_flashn	In	FLASH reset that is register-activated. To activate this reset, µC must access CPLD register file through µC bus.
if_reset_eth_phyn	In	Ethernet PHY reset that is register-activated. To activate this reset, µC must access CPLD register file through µC bus.
rst	Out	Reset for CPLD internal circuits.
reset_flashn	Out	Reset output for flash.
reset_eth_phyn	Out	Reset output for Ethernet PHY.
reset_ucn	Out	Reset output for µC.

The following table lists the specifications for the duration of reset signals, of each of the peripherals controlled by µC:

Reset	Min	Max	Note
reset_flashn	25µs	--	28F128J3D.pdf page 31/72.
reset_eth_phyn	1µs	--	DP83848C.pdf page 65/84.

reset_ucn	--	--	Wakeup timer adjusts the time of activation after a reset has been applied to µC.

Since the external reset source is a hardware specific for generating reset signals which produces a reset pulse of at least 200 ms, the above specifications are successfully met. Thus the only functionality added by this unit, is some AND gates to allow the internal register set to handle specific outputs, by using a delayed version of the assigned values to reset generating registers.

8 VHDL Files

File Name	Description
uc_prog_support.vhd	File that implements the unit that enables uC programming from RS232.
pwm_err_reg.vhd	Implementation of a unit that holds the errors detected at input PWM signals.
watchdog.vhd	Unit that monitors uC activity and resets it if it doesn' t operate as specified.
up_if_package.vhd	uC parallel interface unit that is the circuitry that contain all registers accessible from parallel bus.
register.vhd	File implementing a D-FF type register.
reset_gen_unit.vhd	Unit creating internal reset signals.
mux_2_to_1.vhd	Unit that multiplexes two signals into one, based on the value of a select signal.
counter_async_rst.vhd	Counter with asynchronous reset.
pwm_analyzer.vhd	Unit that analyzes the width of PWM pulses.
up_if_unit.vhd	Unit that interfaces with the parallel bus of uC.
Pilot_IV_CPLD.vhd	Top level file for the CPLD created for the autopilot.
Simulations/ register/	Test bench for reg.vhd.

TB_reg.vhd	
Simulations/ CPLD_sim/ TB_Pilot_IV_CPLD_synth_pwm_an_only.vhd	Test bench for Pilot_IV_CPLD.vhd
Simulations/ CPLD_sim/ TB_signal_debouncer.vhd	Test bench for signal_debouncer.vhd.
Simulations/ CPLD_sim/ TB_counter.vhd	Test bench for counter.vhd.
Simulations/ CPLD_sim/ TB_pwm_analyzer.vhd	Test bench for pwm_analyzer.vhd.
Simulations/ CPLD_sim/ SimWithOldModelSim/ TB_pwm_gen_for_test.vhd	Test bench for pwm_gen_for_test.vhd
Simulations/ CPLD_sim/ SimWithOldModelSim/ pwm_test_module.vhd	Module creating PWM test signals.
Simulations/ CPLD_sim/ SimWithOldModelSim/ TB_Pilot_IV_CPLD_pwm_only.vhd	Test bench for Pilot_IV_CPLD.vhd, focusing on PWM inputs.
Simulations/ CPLD_sim/ SimWithOldModelSim/ pwm_gen_for_test.vhd	Module that produces PWM pulse like the radio. The module creates a pulse that has a period of 50ms and the high state varies from 900us width to 2.090ms with 4.8us steps (0 to 248) if steps_to_subtract = 0. If steps_to_subtract receives a value, then the unit will subtract the corresponding steps from the pulse width.
Simulations/ CPLD_sim/ SimWithOldModelSim/ TB_Pilot_IV_CPLD.vhd	Test bench for Pilot_IV_CPLD.

Simulations/ CPLD_sim/ SimWithOldModelSim/ dummy_ibuf_package.vhd	Package for ibuf.vhd constants and types.
Simulations/ CPLD_sim/ SimWithOldModelSim/ ibuf.vhd	Simulated ibuf (Xilinx module for buffering input clocks).
Simulations/ CPLD_sim/ SimWithOldModelSim/ TB_pwm_test_module.vhd	Test bench for pwm_test_module.vhd.
Simulations/ CPLD_sim/ TB_Pilot_IV_CPLD.vhd	Test bench for Pilot_IV_CPLD.vhd.
Simulations/ CPLD_sim/ pwm_generating_device.vhd	Device that helps evaluating the units that notify for PWM error. It creates normal and fake pulses.
Simulations/ CPLD_sim/ TB_rpm_counters_block.vhd	Test bench for rpm_counters_block.vhd.
Simulations/ CPLD_sim/ TB_uc_prog_support.vhd	Test bench for uc_prog_support.vhd.
Simulations/ CPLD_sim/ TB_reset_gen_unit.vhd	Test bench for reset_gen_unit.vhd.
Simulations/ CPLD_sim/ TB_reg.vhd	Test bench for reg.vhd.
Simulations/ CPLD_sim/ TB_pwm_sampl_clk_gen.vhd	Test bench for sampl_clk_gen.
Simulations/ CPLD_sim/ TB_rst_del_element.vhd	Test bench for rst_del_element.vhd (delay element for reset signal).
counter.vhd	An implementation of a counter unit.
rpm_counters_block.vhd	The unit that calculated the rotational frequency of motor a

rst_del_element.vhd	Delay element for reset signals.
pwm_sampl_clk_gen.vhd	The generator of the clock used for measuring PWM signals.
signal_debouncer.vhd	Unit that helps preventing latchup in sampled asynchronous signals.
Pilot_IV_CPLD.ucf	The file describing the pinout of the CPLD to Xilinx compiler.

VHDL Files and Size in Lines

'counter.vhd'	56
'counter_async_rst.vhd'	52
'dummy_ibuf_package.vhd'	6
'ibuf.vhd'	21
'mux_2_to_1.vhd'	37
'Pilot_IV_CPLD.vhd'	584
'pwm_analyzer.vhd'	324
'pwm_err_reg.vhd'	117
'pwm_generating_device.vhd'	73
'pwm_gen_for_test.vhd'	64
'pwm_sampl_clk_gen.vhd'	70
'pwm_test_module.vhd'	82
'register.vhd'	38
'reset_gen_unit.vhd'	175
'rpm_counters_block.vhd'	141
'rst_del_element.vhd'	82
'signal_debouncer.vhd'	52
'TB_counter.vhd'	69
'TB_Pilot_IV_CPLD.vhd'	946
'TB_Pilot_IV_CPLD_pwm_only.vhd'	1062
'TB_Pilot_IV_CPLD_synth.vhd'	958
'TB_Pilot_IV_CPLD_synth_pwm_an_only.vhd'	961
'TB_pwm_analyzer.vhd'	113
'TB_pwm_gen_for_test.vhd'	71
'TB_pwm_sampl_clk_gen.vhd'	58
'TB_pwm_test_module.vhd'	204
'TB_reg.vhd'	114
'TB_reset_gen_unit.vhd'	116
'TB_rpm_counters_block.vhd'	114
'TB_rst_del_element.vhd'	79
'TB_signal_debouncer.vhd'	78
'TB_uc_prog_support.vhd'	102
'uc_prog_support.vhd'	56
'up_if_package.vhd'	63
'up_if_unit.vhd'	629
'watchdog.vhd'	108

TOTAL VHDL LINES **7875**

8.1 CPLD Internal Architecture

The figures that follow provide an overview of the internal architecture of the CPLD developed for our experiments.

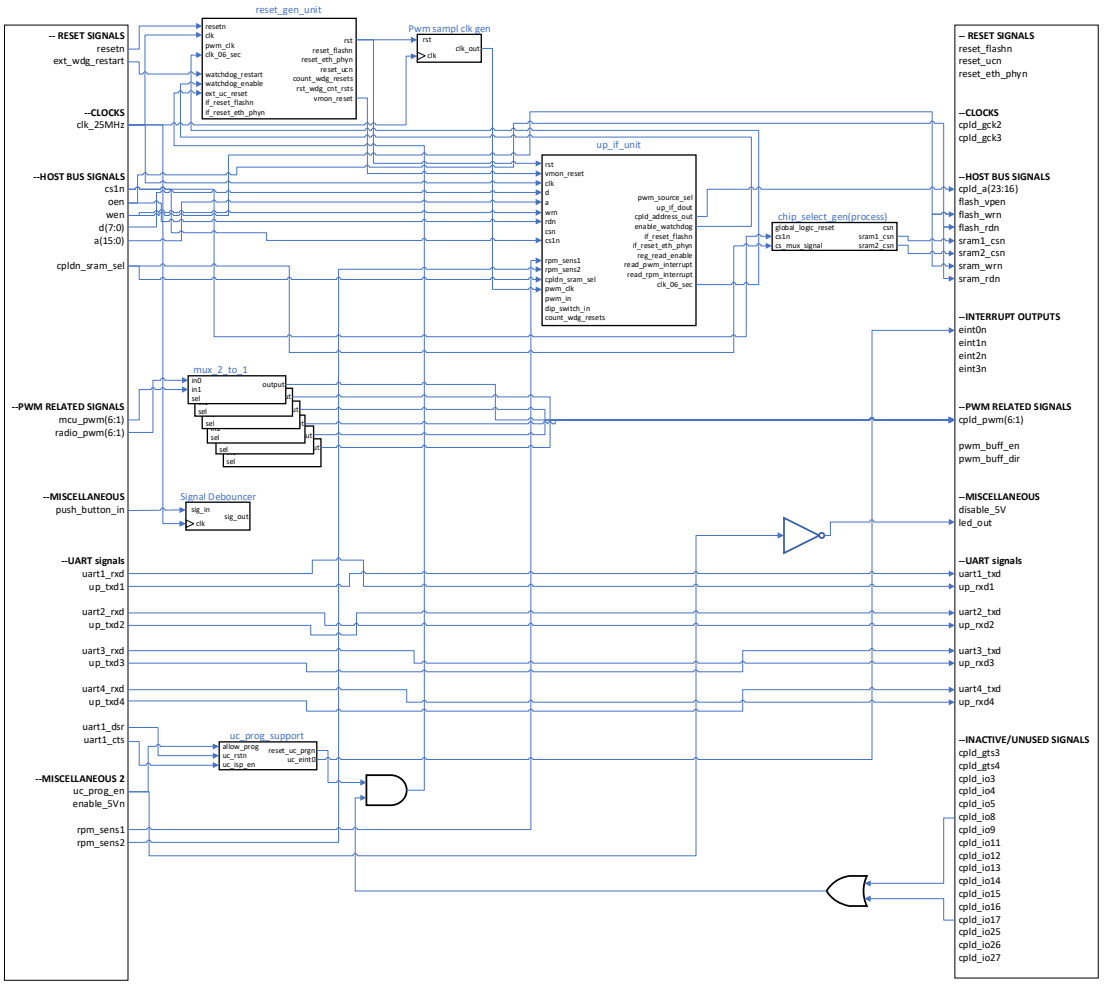


Figure 124: Top level of CPLD (schematic representation of Pilot_IV_CPLD.vhd) .

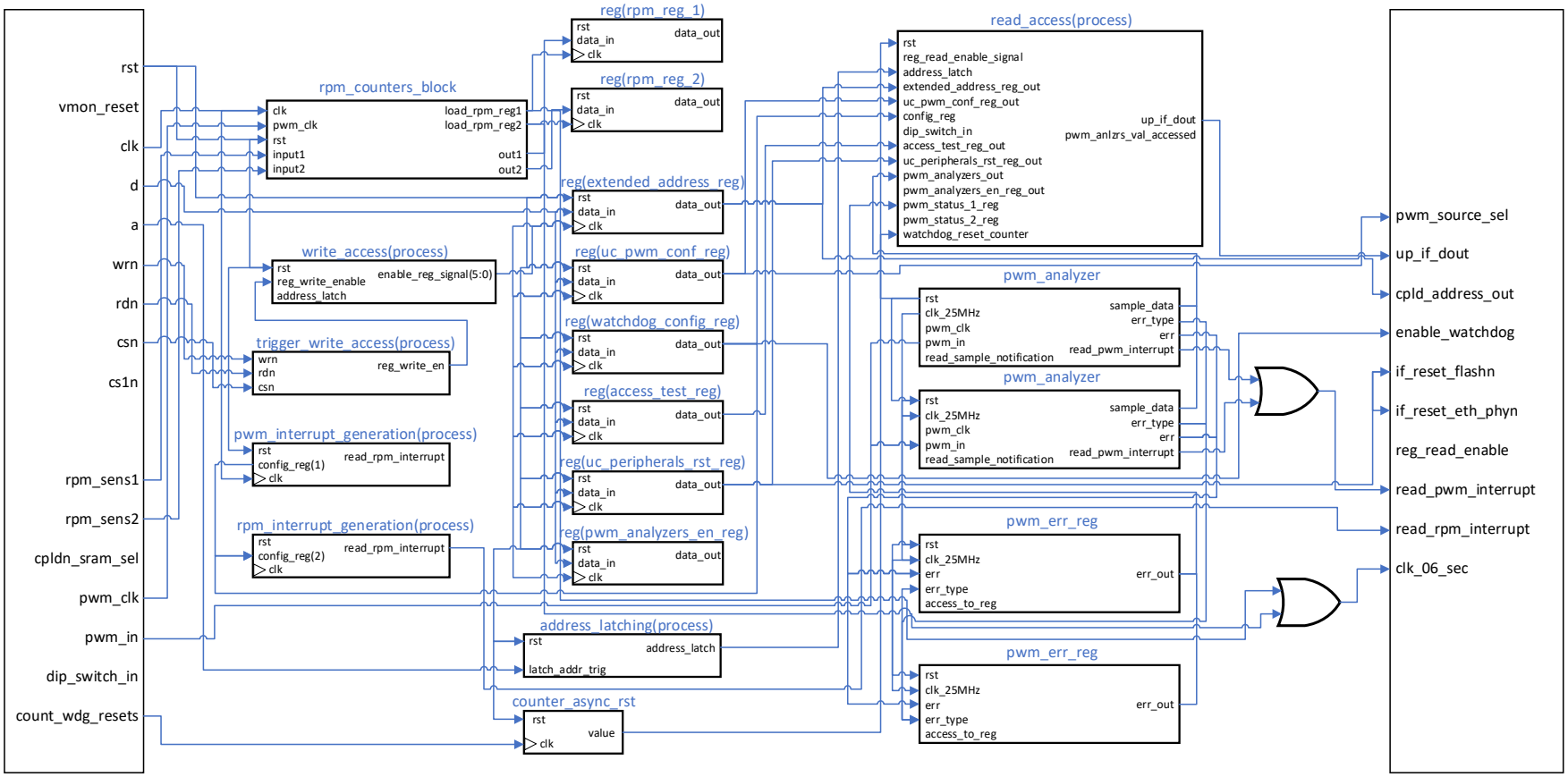


Figure 125: up_if_unit internal structure.

Figure 124 shows a top-level view of the internal structure of the CPLD. Besides simple pass-throughs and gates, some circuits worth mentioning are:

- A set of six multiplexers that are used to redirect the PWM outputs either to manual operation (using radio control) or to auto (autopilot outputs).
- An internal resets generating unit, that ensures that all circuitry begins at the predefined default state.
- A unit creating the clock for PWMs steps. The use of this clock saves internal gates, by avoiding using clock dividers for all PWM counting actions.
- A uC interface unit, that is further analyzed below.
- A chip select generator, redirecting uC accesses to the respective peripheral, based on the address requested.
- A circuit that supports uC programming using UART1.

The boxes at the sides of the interconnected units show the pinout of CPLD.

Figure 125 provides a view of the uC parallel interface unit internal. The boxes at the sides describe the pinout of the unit.

At the middle of the unit, there is a column of register units. This is the register file of the CPLD, that stores all the settings assigned by the processor through the parallel bus. The input of these registers is handled by a unit named “write_access”, which handles the write accesses from the parallel bus.

The content of these registers, along with some read only information such as pwm_analyzer values, or pwm_err_reg status values, are read through the bus with the assistance of read_access unit.

9 Electronic Peripherals

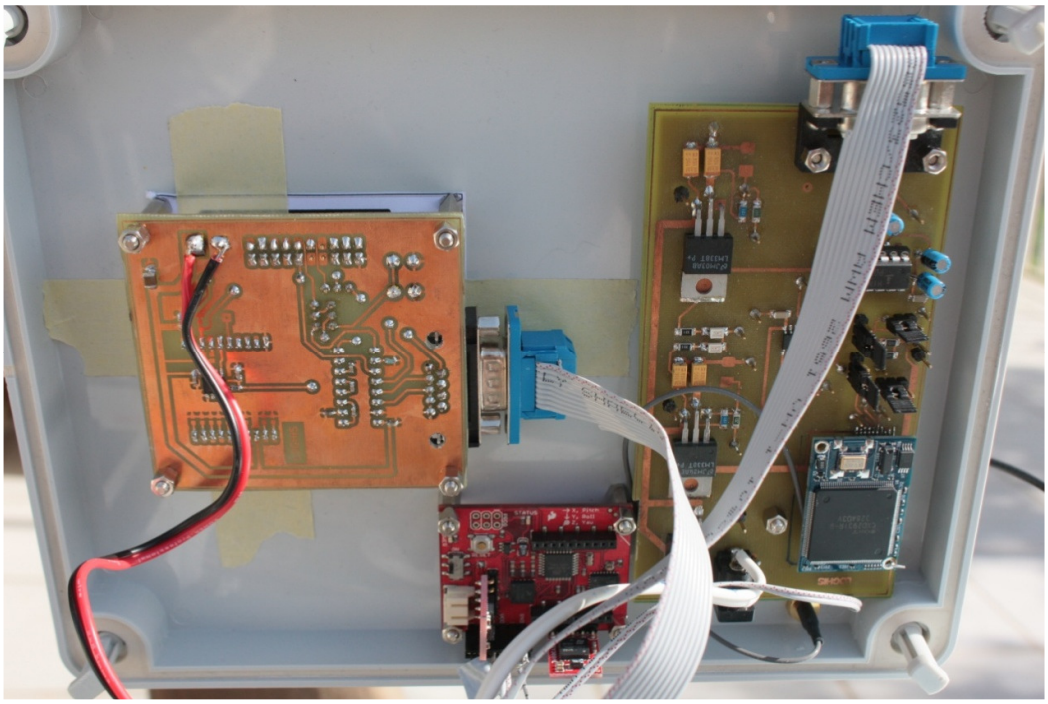


Figure 126 : Overview of some peripherals attached to the hood of main controller box.

In Figure 126 the attached sensors at the main unit's hood are shown. From left to right, the XBee radio link board is the first. On its right and below we have the IMU unit and next to it the GPS board.

Other peripherals not shown are the magnetometer and a camera that follows selected targets.

10 Board Software

The software developed to manage all the system is based on FreeRTOS real time operating system. The core is a scheduler that runs the following tasks:

- vCLITxTask: which is the task that outputs the CLI messages to the serial port that is used for communication with PC. Task that is executed in idle time.
- vIMU_Task: It is the task that collects the output of IMU. It is enabled each time IMU transmits data. Periodic task
- vCompassTask: The task that acquires samples from compass module.
- vGPSrxTask: The task that receives position information from GPS.
- vCLI_Task: The task that receives commands from the user serial port and executes them. It is a non-periodic task based on interrupts.

The basic sensor tasks update a database accessible by SystemMonitor once every 30ms, to make it available for the control algorithm.

Along with the code for the aforementioned tasks, there is also supporting code that has to do with drivers for the hardware peripherals. The following table contains the code files used.

File Name	Description
Camera.h	Header file used for the CMUcam4 camera module.
Camera.c	Code file used for the CMUcam4 camera module.
Camera_ISR.c	Interrupt service routine for camera module.
Camera_ISR.h	Header for interrupt service routine of the camera module.
CLI_ISR.c	Interrupt service routines for the interrupts of serial port used for user CLI.
CLI_ISR.h	Header for the above code file.
Compass.c	Code file for the digital compass module MicroMag3.
Compass.h	Header file for the digital compass code.
Debug_n_CLI.c	The code for the CLI user interface.
Debug_n_CLI.h	Header file for the CLI.
Flash.c	Code file for FLASH memory accesses.
Flash.h	Header file for FLASH.
FreeRTOSConfig.h	Configuration file for modifying FreeRTOS parameters for our specific implementation.
GlobalDefinitions.h	Global variable definitions for the whole code.
GPIO.c	Digital GPIO control code.
GPIO.h	Header for GPIO control.
GPS.c	Code file containing routines for GPS access, message decoding and configuration.
GPS_Globals.h	Header for global variables of GPS peripheral.

GPS.h	Header for GPS functions.
GPS_ISR.c	Code for GPS interrupt service routine.
GPS_ISR.h	Header for GPS interrupt service routine.
IMU.c	Code for inertial measurement unit (IMU).
IMU.h	Header for the IMU code.
IMU_ISR.c	Code for IMU interrupt service routine.
IMU_ISR.h	Header for IMU interrupt service routine.
lpc23xx.h	Header file for microcontroller configuration.
main.c	The main file. All initial configurations are done in this file and the task scheduler is programmed.
ParIF.c	Parallel bus I/F code. This code implements a parallel bus between microprocessor and CPLD, SRAMs, FLASH.
ParIF.h	Parallel bus header file.
PwmIF.c	Code implementing PWM interfaces to control servos.
PwmIF.h	Header for PWM I/Fs file.
PWM_ISR.c	Interrupt service routine for PWM I/Fs.
PWM_ISR.h	Header for the interrupt service routine of PWM I/Fs.
RadioControl.c	Code implementing radio control reading and switching between autonomous and manual handling mode.
RadioControl.h	Header for radio control file.
SeaTests.c	Code implementing sea trial tests in autonomous mode. In this file typical sea trial tests like zig-zag maneuver, or circle tests.
SeaTests.h	Header file for sea trials.
SerialIF.c	File containing the code for serial interfaces of the board.
SerialIF.h	Header file supporting the serial interfaces code.
SRAM.c	File containing the code for accessing the on-board SRAMs.
SRAM.h	Header file supporting the serial SRAMs code.
SystemMonitor.c	Code that handles the database containing all the information available from the vessel's sensors. It also contains a function that displays this information in real time through serial interface.
SystemMonitor.h	Header file supporting the code file described above.
TaskIDs.h	Header file containing the IDs assigned to tasks, in order to recognize them while studying the scheduler.
Test.c	Some test routines for code development.
Test.h	Header file for test routines.

Figure 127 that follows, shows the architecture of the software. The scheduler of FreeRTOS is assigned several tasks that run periodically such as the "Control Algorithm". Along with the periodical tasks there are interrupt-driven tasks that are

activated once an interrupt related to them is asserted. Such task is the “CLI task”, which responds when the user types a command.

The whole system is a hard real-time case of software development since it must integrate the functionality of the components listed in Table 19.

SENSORS AND SAMPLING RATE				
Sensor	**Manufacturer**	**Model**	**Internal Sensor**	**Sampling Rate**
GPS	Laipac	UV40		1 Hz
IMU	Sparkfun	SEN-09184	Gyro	88 Hz
			X-Axis Accelerometer	350 Hz
			Y-Axis Accelerometer	350 Hz
			Z-Axis Accelerometer	150 Hz
Magnetometer (Compass)	PNI	MicroMag3	Magnetometer	2 kHz

Table 19 : Sensors and their maximum data rates.

All listed sensors must create a result that will be exploited by the control algorithm, in order to generate new action on vessel controls every 30 ms.

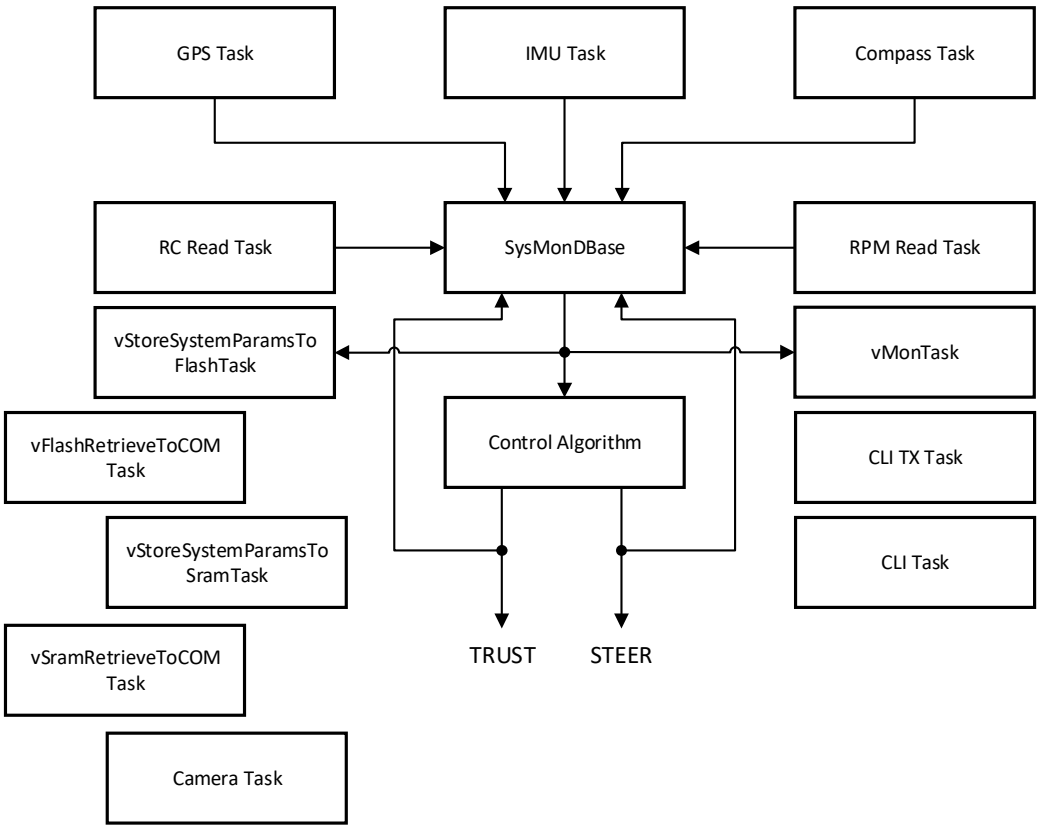


Figure 127 : Overview of software architecture.

C Files and Size in Lines	
'Camera.c'	1017
'Camera_ISR.c'	206
'CLI_ISR.c'	85
'Compass.c'	281
'Debug_n_CLI.c'	3353
'Flash.c'	406
'GPIO.c'	475
'GPS.c'	583
'GPS_ISR.c'	170
'IMU.c'	360
'IMU_ISR.c'	94
'main.c'	517
'ParIF.c'	789
'PwmIF.c'	256
'PWM_ISR.c'	100
'RadioControl.c'	126
'SeaTests.c'	561
'SerialIF.c'	576
'SRAM.c'	405
'SystemMonitor.c'	532
'Test.c'	171
TOTAL C LINES	**11063**

10.1CLI Commands

As an introduction to section, the notation used to describe mandatory fields is <field x>. To describe optional fields, we use {<field x>} notation.

10.1.1 sel Command

Sel command is used to select to which device the microprocessor bus will address the access.

Syntax: sel <field 1>

<field 1> can take the following values:

- CPLD.
- SRAM

CPLD is selected when accesses using wr or rd commands must be directed to CPLD. When SRAM, is selected the rd and wr commands will address the SRAMs instead.

10.1.2 wr Command

wr command writes data to a specified address, either to CPLD or SRAM. The selection between the two is done using sel command described in subsection 10.1.1.

Syntax: wr <field 1> <field 2>

<field 1> is the address to write. The access will be directed to the destination selected by sel command.

<field 2> is the byte to be written.

10.1.3 rd Command

rd command is used to read data from microcontroller peripherals.

Syntax: rd <address to read from> {<bytes to read>}

10.1.4 comread Command

comread command is used to retrieve data from a memory using COM port selected from CLI. This command is used after experiments - trials, to retrieve the data recorded.

Syntax: comread <field 1>

<field 1> can take the following parameters:

- SRAM
- FLASH

When SRAM is selected, the microcontroller will output all recorded samples using COM. It stops at the last recorded sample and does not output all SRAM content.

When FLASH is selected, it reads all recorded samples from FLASH memory.

10.1.5 memtest Command

It is a command that tests the board's SRAM by filling it with a predefined test sequence (0x00, 0x55, 0xAA, 0xFF) and reading from it. If it finds an error in the sequence during read access it reports it, else it reports successful completion of the test.

Syntax: memtest

10.1.6 set Command

Using set command higher address bits or output PWM interfaces can be accessed. More details will be given during command syntax analysis:

Syntax: set <field 1> <field 2> {<field 3>}

<field 1> can take the following parameters:

- addr_hi: to set higher order address bits o the value specified by a byte at <field 2>.
- pwm: In this case <field 2> can be a number indicating which of the output PWMs is accessed. <field 3> in this case is the value that must be assigned.
- thrust: to setup the limits of thrust control PWM. In this case <field 2> can be:
    - idle
    - min
    - max
- steer: to setup the limits of steering control PWM. In this case <field 2> can be:
    - idle
    - min
    - max
- dirport: to setup the limits of steering control PWM when steering left. In this case <field 2> can be:
    - idle
    - min
    - max
- dirstrb: to setup the limits of steering control PWM when steering right. In this case <field 2> can be:
    - idle
    - min
    - max

<field 3> is a value assigned to the parameter described from the previous fields where applicable.

10.1.7 get Command

Get command is used to retrieve specific information as described by the command fields explanation that follows.

Syntax: get <field 1>

<field 1> can take the following parameters:

- stack: this parameter makes the CLI respond the stack watermark for each task. This way the adequateness of the reserved stack for the task is monitored.

- hndlsinfo: using this parameter, CLI responds by displaying the current value of handles for the vessel i.e., thrust, steer, port direction, starboard direction. The last two refer to the movement inversion flaps at the left and right motors respectively.

10.1.8 erase Command

Erase command deletes the contents of the specified memory.

Syntax: erase <field 1>

<field 1> can take the following parameters:

- flash: erases all flash contents.
- sram: erases all SRAM contents.
- flapp: erases the application part of the FLASH memory. Application part is a part reserved in FLASH memory for data that have to do with persistent configurations.

10.1.9 imu Command

This command invokes the IMU debug state of CLI. During this state the commands that are received from keyboard, are redirected to IMU. This way we use the system as a debugger interacting only with IMU.

Syntax: imu

To exit the IMU debug state, type the '~' character and press enter.

10.1.10 gps Command

This command invokes the GPS debug state of CLI. During this state the commands that are received from keyboard are redirected to GPS. This way we use the system as a debugger interacting only with GPS.

Syntax: gps

To exit the GPS debug state, type the '~' character and press enter.

10.1.11 mon Command

This command invokes a task called vMonTask that displays parameters in real time using CLI COM port. It is used to monitor all activities of sensors and cotrols.

Syntax: mon

To exit the monitor task, type 'e' or 'E'.

10.1.12 store Command

Store command is used to invoke memory storage tasks (SRAM or FLASH), that save all current maneuvering parameters to the selected memory once every 30ms. This is the rate that all measurements and states are aligned to, for the control algorithm to operate.

Syntax: store <field 1>

<field 1> can take the following self explained parametes:

- sram
- flash

11 Suggestions for Future Work

This work can be extended in many ways. One way is described in chapter 13 “APPENDIX 1: Foundations of Future Work”, where we create an additional layer of processing vessel position targets, in order to mimic the movements of charged quantities in a custom “electric field” – type of environment, for which we define arbitrary laws, suitable for our application.

Another more evident way is to add a layer for masterless swarm formation and employ swarms of such vessels to test the outcome of various algorithms.

Expansion of the thesis ideas to larger dimension vessels is also appropriate and enables a vastly different scope of applications that can be even at open seas. Such applications can include:

- Search and rescue fleet. Many vessels with thermal cameras and AI driven hearing aids, in search of people after a shipwreck. AI algorithms can also create the formation needed to confine survivors to an area, to ease the life-saving rescue operation.

- Open sea transportation. This application can be a master following scenario, where a human-controlled cargo ship may be followed by several autonomous cargo vessels of various sizes, that, for example, may carry the dangerous part of the merchandise.

- Autonomous tugs that attach to a ship, to guide it through narrow and complicated paths while protecting it from potentially harmful collisions.

- Autonomous tugs that accurately position drills for oil extraction.

- Autonomous fleet performing seabed scanning for various applications.

The list can be expanded with lots of other applications. However, the progress of each application depends on the research team’s commitment and the funding approved for acquiring necessary equipment and for compensating the manhours needed to complete the required tasks. The electronics developed during this thesis can be modified to support each of these applications, with estimates of development time ranging from several months, to a year and a half, depending on the number and interfaces of the additional sensors needed and the extent of software adaptations.

12 Matlab Code Files and Descriptions

12.1StatLin Folder

AccX_net_weights_and_biases.txt
File containing all neural network weights and biases for AccX neural network. To be used to report internal structure.
CompareSim2EqOutAccXNet.m
In this file the equation we have derived for AccX neural network is evaluated in comparison to the output of Matlab sim command. The results show that our mathematical interpretation is exact.
CompareSim2EqOutWdotNet.m
In this file the equation we have derived for Wdot neural network is evaluated in comparison to the output of Matlab sim command. The results show that our mathematical interpretation is exact.
CompJacobian.m

FindContTimeModelOperatingPoint.m
LocateNeighbouringPoints.m
matlab_commands_for_nnet_verification.txt
NumericalJacobian.m
TempFileForCommandsEntry.matlab
TestsWithAccXNet.m
TestsWithWdotNet.m
WdotNN_weigths_and_biases.txt
File containing all neural network weights and biases for Wdot neural network. To be used to report internal structure.

File Name	Lines	Description
FixVector.m	36	Created by: Lefteris Loghis 13/11/13 Version 1. OutVector = FixVector(VectorIn, SamplesToFix, SamplesToUseAsRef) This function fixes problematic samples specified using random generated samples resembling to reference samples. Inputs: ------- VectorIn: The whole vector. SamplesToFix: Array of selected samples to fix using SamplesToUseAsRef as reference. SamplesToUseAsRef: Samples to be used as reference, in order to generate min max values for the new samples. Output: ------- OutVector: The fixed vector.
NormalizeThrust.m	49	Created by: Lefteris Loghis 7/12/19 Version 3. Changes from version 2 (20/11/13): Added Verbose parameter. Changes from version 1: Deletes less than zero samples. OutVector = NormalizeThrust(ThrustVector) This function normalizes thrust vector. This means that it removes any DC offset so as to begin from zero. The maximum theoretical span of thrust is 0 to 255, but in practice I see an 161 decimal divergence from the initial idle value to max.

		Inputs: ------- ThrustVector: The thrust vector. ThrustOffst (Optional): Needed if you want a specific offset. Verbose: Allows reporting of zeroed samples. Output: ------- OutVector: The normalized output vector.
mavg.m	76	Created by: Lefteris Loghis 28/10/13 Version 2. Changes from version 1: Added filter initializer value capability. TimeSeriesOut = mavg(TimeSeriesIn, NoOfTaps, InitializationValue) This function generates a time series vector that is the original after passing from a moving average filter. Inputs: ------- TimeSeriesIn: The input time series vector (one dimension). NoOfTaps: The number of taps of the moving average filter. InitializationValue (Optional): optional initialization value. All filter taps are initialized with this value when specified. If omitted then the default value is 0. Output: ------- TimeSeriesOut: The filtered time series with their delay removed or not depending on RemoveDelay value.
lef_max.m	70	Finds maximum values greater than a threshold.
NormalizeYawAngle.m	51	Created by: Lefteris Loghis 7/11/13 Version 1. OutVector = NormalizeYawRate(YawRateVector) This function normalizes Yaw Rate vector. This means that

		it makes it vary between -180 to 180 degrees. Inputs: ------- YawRateVector: The Yaw Rate vector. Output: ------- OutVector: The normalized output vector. If an error occurs then the result is NaN.
CreateWVectors.m	307	%Created by: Lefteris Loghis %28/10/13 %Version 1. %{decTrust, decSteer, decSpeed, decW} = CreateWVectors(DecimationFactor, Filename, PlotFigures) %This function outputs the decimated series needed to train a %neural network as angular velocity map. % %Inputs: %------- %DecimationFactor: The decimation factor for the vectors (integer). %Filename: The filename of the experiment that will train the neural network. %PlotFigures: If 1 then figures of the original and decimated series will be plot. % Otherwise if 0 or ommitted then no plots will be shown. % %Output: %------- %{decTrust, decSteer, decSpeed, decW}: a composite table consisting of: %decTrust: filtered and decimated thrust handle series. %decSteer: filtered and decimated steer handle series. %decSpeed: filtered and decimated speed X series. %decW: filtered and decimated angular velocity series.
TrainNNetForW.m	559	Created by: Lefteris Loghis 6/3/14 Version 6. (Do not forget to change Version variable). Changes from version 5: Added normalization of signals. Corrected W position at neural input matrix.

```
                Increased info output at "ReadMe.txt".
Changes from version 4:
                It now saves only the Wnet variable.
Changes from version 3:
                Changed input parameters list.
                Changes the way the function acquires the input vectors for the NNet.

This function creates and saves a neural network trained to produce
angular rate samples, by the series contained in the file named Filename.

TrainNNetForW (  Filename, DecimationFactor, k, NetType, Epochs )

Inputs:
-------
Filename: Filename of the experiment selected.
DecimationFactor: Factor by which the training vectors will be decimated.
k: Number of past samples per input.
NetType (optional): Type of neural net internal neurons. Possible values:
                1.'logsig' (default)
                2.'tansig'
Epochs (optional): The max training epochs for the net. Default = 8000.

Output:
-------
Saves workspace created under the name WNeural_yyyymmdd_hhmm.mat in folder
%$currentDirectory\WNeural.
The network will be trained as follows:
| Thrust[n-(k-1)]     |
|...                  |
| Thrust[n-1]         |
| Thrust[n]           |
| ...                 |
| Steer[n]            | = W[n+1]
| ...                 |
| SpeedX[n]           |
| ...                 |
| W[n]                |
```

CreateWTrainVectors.m	385	Created by: Lefteris Loghis 6/3/14 Version 1. (Do not forget to change Version variable). This function creates the vectors used to train the W net. The function was created for debug reasons. CreateWTrainVectors (Filename, SignalsOffsetStruct, NNetTrainingParams, SpecialCommands) Inputs: ------- Filename: Filename of the experiment selected. SignalsOffsetStruct: The offset structure used to calibrate the measurement. NNetTrainingParams: This variable contains the parameters for the training of the net. more accurately: DecimationFactor: Factor by which the training vectors will be decimated. Samples per vector: Number of samples per input. NetType (optional): Type of neural net internal neurons. Possible values: 1.'logsig' (default) 2.'tansig' Epochs (optional): The max training epochs for the net. Default = 8000. Goal: The target goal for the training of the neural net. Output: ------- CreatedVectors: A cell array containing the name of the contents at the first column and the respective vector at the second.
NormalizeYawRate.m	45	Created by: Lefteris Loghis 7/11/13 Version 1. OutVector = NormalizeYawRate(YawRateVector) This function normalizes Yaw Rate vector. This means that it makes it vary between -180 to 180 degrees. Inputs:

		------- YawRateVector: The Yaw Rate vector. Output: ------- OutVector: The normalized output vector. If an error occurs then the result is -1.
DecmtTimeVector.m	67	Created by: Lefteris Loghis 21/1/14 Version 1. ONLY FOR TIME VECTORS, TO MAKE THEM MATCH DECIMATED SERIES. TimeSeriesOut = DecmtTimeVector(TimeVectorIn, DecimationFactor) This function generates a time series vector that is the original after after removing the intermediate samples between decimation sample instants. No filtering has been applied, so its use is only for Time Vectors. The resulting time vector is about DecimationFactor times shorter! Inputs: ------- TimeVectorIn: The input time series vector (one dimension). DecimationFactor: The factor by which input time series are decimated. Output: ------- TimeSeriesOut: The decimated time series.
GetTimeSuffix.m	19	Created by Lefteris Loghis 29/2/20 Version 1 This function returns a string that can be used to mark the time requested. The format it returns is yyyymmdd_hhss. The function takes no arguments and works as follows: TimeSuffix = GetTimeSuffix

Verde_2_2.m	425	Function that converts measurements of Autopilot_IV Version 0.1 Expects a csv file with the following columns separated by spaces: Frame No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ
SpeedXNeural.m	371	Function that trains a neural network to match the vessel's next future sample. Created by: Lefteris Loghis 09/01/14 Version 1.0 SpeedXNeural(Filename, SignalsOffsetStruct, NNetTrainingParams, SpecialCommands) Inputs: ------- Filename: The name of the measurement to train neural network with. SignalsOffsetStruct: The structure used by DisplayMaesurement to input the known offsets needed for the measurement. Signal offset struct by default is: SignalsOffsetStruct.TimeStart = NaN; SignalsOffsetStruct.TimeEnd = NaN; SignalsOffsetStruct.ThrustOffst = NaN; SignalsOffsetStruct.SteerOffst = NaN; SignalsOffsetStruct.DirectionOffst = NaN; SignalsOffsetStruct.AccXOffst = NaN; SignalsOffsetStruct.AccYOffst = NaN; SignalsOffsetStruct.AccZOffst = NaN; SignalsOffsetStruct.RollRateOffst = NaN; SignalsOffsetStruct.PitchRateOffst = NaN; SignalsOffsetStruct.YawRateOffst = NaN; SignalsOffsetStruct.PitchCompassOffst = NaN; SignalsOffsetStruct.SpeedXOffst = NaN; These are the values that it will take if you ommit the parameter. NNetTrainingParams: The parameters used to train the neural network. These are: DecimationFactor = 30; (By which factor will the signals be decimated. 1 = no decimation). k = 8; (How many past vectors will be used for the neural training).

		NetType = {'logsig' 'purelin'}; (The types of the middle and output neurons). Epochs = 8000; (How many training epochs will there be). Goal = 1e-5; (What is the target error of the training). The values presented here are the ones assigned if this parameter is ommitted. SpecialCommands: Special parameters used for DisplayMeasurement Outputs: -------- A neural network trained by the requested measurement stored in {work_folder}/SpeedXNeural/SpeedXNeural_{current_date} Use SpeedXNeural('Verde_2_6_ForMatlab.csv') initially. http://www.mathworks.com/products/neural-network/description5.html http://en.wikipedia.org/wiki/Downsampling NNetTrainingParams = { DecimationFactor, k, NetType, Epochs } These values represent the order in which the file content must be.
FindNetName.m	20	Created by: Lefteris Loghis 27/03/14 Version 1. This function checks the string of the NNetFilename and returns the expected neural network type inside the workspace pointed by the filename.
PlotFigures.m	178	Created by: Lefteris Loghis 27/2/20 Version 2 Differences from Version 1 (15/2/20) 1. The function returns the figure handle. Plots all requested vectors, scaled by the first one. FigureHandle = PlotFigures(Yvectors, Xvector, TitleY, TitleX, TitleGraph)

		Inputs: ------- Yvectors: vector or cell array with vectors and colors - linestyle strings. Xvector: vector for the timeline. TitleY: title for the Y axis. TitleX: title for the X axis. TitleGraph: title for the whole graph. NoNorm: If 1 then the vectors are not nomalized. Outputs: -------- FigureHandle: the plotted figure handle. Example usage: -------------- x_data = [0:99]; Yvectors{1,1} = sin(pi/20*x_data); Yvectors{2,1} = 0.1*sin(pi/20*x_data + pi/4); Yvectors{3,1} = 3 + 5*cos(pi/20*x_data); Yvectors{1,2} = 'b'; Yvectors{2,2} = 'k'; Yvectors{3,2} = 'go'; PlotFigures(Yvectors)
TstDifferentialEqNeuralModel.m	211	Created by: Lefteris Loghis 17/02/14 Version 1. This function tests the accuracy of the vessel model produced by the neural networks Check the input arguments: Use size(Wnet.inputs{1}.range) to find the number of past samples of inputs required, for the case of W neural.
CreateSpdXVectors.m	214	Function that trains a neural network to match the vessel's next future sample. Created by: Lefteris Loghis 09/01/14 Version 1.0

CreateSpdXVectors(Filename, SignalsOffsetStruct, NNetTrainingParams, SpecialCommands)

Inputs:

Filename: The name of the measurement to train neural network with.

SignalsOffsetStruct: The structure used by DisplayMaesurement to input the
known offsets needed for the measurement. Signal offset struct by default is:
SignalsOffsetStruct.TimeStart = NaN;
SignalsOffsetStruct.TimeEnd = NaN;
SignalsOffsetStruct.ThrustOffst = NaN;
SignalsOffsetStruct.SteerOffst = NaN;
SignalsOffsetStruct.DirectionOffst = NaN;
SignalsOffsetStruct.AccXOffst = NaN;
SignalsOffsetStruct.AccYOffst = NaN;
SignalsOffsetStruct.AccZOffst = NaN;
SignalsOffsetStruct.RollRateOffst = NaN;
SignalsOffsetStruct.PitchRateOffst = NaN;
SignalsOffsetStruct.YawRateOffst = NaN;
SignalsOffsetStruct.PitchCompassOffst = NaN;
SignalsOffsetStruct.SpeedXOffst = NaN;
These are the values that it will take if you ommit the parameter.

NNetTrainingParams: The parameters used to train the neural network. These are:
DecimationFactor = 30; (By which factor will the signals be decimated. 1 = no decimation).
k = 8; (How many past vectors will be used for the neural training).
NetType = {'logsig' 'purelin'}; (The types of the middle and output neurons).
Epochs = 8000; (How many training epochs will there be).
Goal = 1e-5; (What is the target error of the training).
The values presented here are the ones assigned if this parameter is ommitted.

SpecialCommands: Special parameters used for DisplayMeasurement

Outputs:

A neural network trained by the requested measurement

		stored in {work_folder}/SpeedXNeural/SpeedXNeural_{current_date}
FindSpeedVect.m	106	Function that converts measurements of Autopilot_IV Version 3.0 Changes from version 2: Used version 6 of DisplayMeasurement command. Thus added special parameter variable. Changes from version 1: Now using DisplayMeasurement to get the needed vector. Expects a csv file with the following columns separated by spaces: Frame_No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ Use FindSpeedVect('Verde_2_6_ForMatlab.csv')
NormalizeDirection.m	93	Created by: Lefteris Loghis 13/11/13 Version 2. Difference from version 1: If an error occurs, the function returns an empty vector instead of -1. OutVector = NormalizeDirection(DirectionVector) This function normalizes Direction vector. This means that it removes any DC offset so as to have zero as minimum. The maximum theoretical span of thrust is 0 to 255, but in practice I see a -91 to +89 decimal deviation from the Idle point. The usual Idle point is 149. This will be used as a target as Idle point for all measurements. Have in mind that if a sample is larger than the idle point, it will be forced to idle, since more forward than no backwards has no practical meaning. Inputs: ------- DirectionVector: The Direction vector. IdlePointValue (optional): If the Idle point value of the input vector does not match its median, then use this value instead. This number must be between 0 and 255. Most possibly it will be something close to 149.

		Output: ------- OutVector: The normalized output vector. If an error occurs then the result is [].
SimulinkDiffceNeuralSpeed.m	44	Created by: Lefteris Loghis 25/03/14 Version 1. This function is used in simulink to apply the input vector to the neural network. Inputs: ------- InputVector: The input vector containing all necessary input instances. Output: ------- OutValue: The output of the neural network.
tst_1.m	425	Function that converts measurements of Autopilot_IV Version 0.1 Expects a csv file with the following columns separated by spaces: Frame No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ
DispMeas.m	425	Function that converts measurements of Autopilot_IV Version 1.0 Expects a csv file with the following columns separated by spaces: Frame No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ
VerifyDifferenceEqNeuralModel.m	403	Created by: Lefteris Loghis 08/03/14 Version 1. This function tests the accuracy of the vessel model produced by the neural networks that model X-axis acceleration and angular acceleration.

		VerifyDifferenceEqNeuralModel (SpeedXNNFilename, WNNFilename, ExperimentFilename, DecimationFactor, PastSamplesPerVector, SignalsOffsetStruct, SpecialCommands) Inputs: ------- SpeedXNNFilename: Filename of the saved environment where the AccX neural net exists. WNNFilename: Filename of the saved environment where the Wdot neural net exists. ExperimentFilename: The filename of the experiment by which the neural net will be tested. DecimationFactor: Define the decimation factor by which all input signals were decimated before training the nets. SignalsOffsetStruct: The same as in Display measurement. SpecialCommands : The same as in Display measurement. Output: ------- Graphical representations presenting the difference between neural net outputs and real measured values. Check the input arguments: Use size(Wnet.inputs{1}.range) to find the number of past samples of inputs required, for the case of W neural.
CheckIfVector.m	13	Result = CheckIfVector (Input) This function checks if the input is a vector. It returns 1 if it is, 0 otherwise. Input: is the argument under test. Result is the output.
PHDobjects/170107/ RotateVector.m	32	Created by: Lefteris Loghis 17/08/2016 Version 1 This function returns the InputVector vector rotated by RotationAngle. RotatedVector = RotateVector(InputVector, RotationAngle) Input arguments: 1. InputVector: The vector that needs to be projected to the rotated axes system.

		2. RotationAngle: The RotationAngle of turn of the new axis system. RotationAngle in rads. Output: RotatedVector: The rotated vector.
PHDobjects/170107/ AddForcesTableRow.m	35	Created by: Lefteris Loghis 29/10/2015 Version 1 This command returns the vector resulting by the addition of all columns of a row ForcesTable. This return vector is the force applied by the system to a charge of a PHDObject. The format of the command is as follows: ResultingForceVector = AddForcesTableRow (ForcesTable, RowNumber) Variables: 1. ForcesTable The cell array of 2x1 vectors, resulting from forces to a charge. 2. RowNumber: The number of row to add.
PHDobjects/170107/ lef_isarray.m	36	Created by: Lefteris Loghis 10/10/2015 Version 1 The format of the command is as follows: Y = lef_isarray (X, number_of_rows, number_of_columns) If only one argument is used then: If X is an array (all dimensions > 1) of numbers then Y = 1. If two arguments are used then: If X is an array OR VECTOR of numbers with rows = number_of_rows then Y = 1. If all arguments are used then : If X is an array OR VECTOR of numbers with rows = number_of_rows and columns = number_of_columns Y = 1.
PHDobjects/170107/ CreatePHDObjects.m	457	Created by: Lefteris Loghis 31/12/2016

		Version 3 Changes from version 2: 1. To support RandomExperimentsBuilder, when you issue the command without arguments, it returns the structure of ObjectConstantsInitialVectors and ObjectMotionVariablesInitialVectors, thus enabling changes to CreatePHDObjects without having to take care for RandomExperimentsBuilder. Changes from version 1: 1. Changed ObjectStructure table. 2. Added automatic calculation of mass center. 3. Added automatic calculation of moment of inertia relative to mass center. The format of the command is as follows: ObjectsStructure = CreatePHDObjects (NumOfObjects, InitialValuesVectors) It returns a structure where the first element is the following cells column table: 'width' --> width of the object (Oy size) 'length_A' --> Length of region A. 'length_AB' --> Length of region AB (the part connecting A to B). 'length_B' --> Length of region B. 'mass_A' --> Mass of region A (considered homogenous) 'mass_AB' --> Mass of region AB (considered homogenous) 'mass_B' --> Mass of region B (considered homogenous) 'quantity_A' --> The A quantity measure that ineracts with B quantity. 'quantity_B' --> The B quantity measure that ineracts with A quantity. 'ShieldSize' --> The size of the region around a charge where the physical law changes to protect the vessel. 'pos_x' --> X position of mass center refering to Oxy axes. 'pos_y' --> Y position of mass center refering to Oxy axes. 'velocity_vector_x' --> Velocity of mass center refering to Oxy. X parameter. 'velocity_vector_y' --> Velocity of mass center refering to Oxy. Y parameter. 'accelleration_vector_x' --> Acceleration of mass center refering to Oxy. X parameter. 'accelleration_vector_y' --> Acceleration of mass center refering to Oxy. Y parameter. 'theta' --> Angle between straight line parallel to the long side of the object passing from mass center and Ox axis of reference. 'omega' --> The rate of change of theta. 'omega_dot' --> The rate of change of omega. 'A_pos_x' --> X position of A point refering to Oxy axes. 'A_pos_y' --> Y position of A point refering to Oxy axes. 'B_pos_x' --> X position of B point refering to Oxy axes.

		'B_pos_y' --> Y position of B point refering to Oxy axes. 'C_pos_x' --> X position of C point refering to Oxy axes. 'C_pos_y' --> Y position of C point refering to Oxy axes. 'D_pos_x' --> X position of D point refering to Oxy axes. 'D_pos_y' --> Y position of D point refering to Oxy axes. 'QA_pos_x' --> X position of A charge refering to Oxy axes. 'QA_pos_y' --> Y position of A charge refering to Oxy axes. 'QB_pos_x' --> X position of B charge refering to Oxy axes. 'QB_pos_y' --> Y position of B charge refering to Oxy axes. 'MomentOfInertia' --> The inertia calculated at the center of mass. 'grav_center_x' --> Center of gravity vector x with reference to bottom left corner of object. 'grav_center_y' --> Center of gravity vector y with reference to bottom left corner of object. 'GA_x' --> x dimension of GA constant vector connecting G (mass center) to A when theta = 0. 'GA_y' --> y dimension of GA constant vector connecting G (mass center) to A when theta = 0. 'GB_x' --> x dimension of GB constant vector connecting G (mass center) to B when theta = 0. 'GB_y' --> y dimension of GB constant vector connecting G (mass center) to B when theta = 0. 'GC_x' --> x dimension of GC constant vector connecting G (mass center) to C when theta = 0. 'GC_y' --> y dimension of GC constant vector connecting G (mass center) to C when theta = 0. 'GD_x' --> x dimension of GD constant vector connecting G (mass center) to D when theta = 0. 'GD_y' --> y dimension of GD constant vector connecting G (mass center) to D when theta = 0. 'th0a' --> Angle of GA constant vector connecting G (mass center) to A when theta = 0. 'th0b' --> Angle of GB constant vector connecting G (mass center) to A when theta = 0. 'th0c' --> Angle of GC constant vector connecting G (mass center) to A when theta = 0. 'th0d' --> Angle of GD constant vector connecting G (mass center) to A when theta = 0. 'length_GA' --> Length of GA vector. 'length_GB' --> Length of GB vector. 'length_GC' --> Length of GC vector. 'length_GD' --> Length of GD vector. 'GQA_x' --> x dimension of GQA constant vector connecting G (mass center) to A charge when theta = 0. 'GQA_y' --> y dimension of GQA constant vector connecting G (mass center) to A charge when theta = 0. 'GQB_x' --> x dimension of GQB constant vector connecting G (mass center) to B charge when theta = 0. 'GQB_y' --> y dimension of GQB constant vector connecting G (mass center) to B charge when theta = 0. The second element of the structure is a two-dimensional table where each column represents the parameters listed in the previous table for each object instantiated. The number of objects present at this stucture is the number of columns of this table.

		Input arguments: 1. NumOfObjects: Number of objects to create. If this number is omitted, one object will be created. 2. InitialValuesVectors: Table providing initial conditions for every object. If this table is omitted, the initial conditions will be set to zero. The same will happen if some columns are omitted; the uninitialized objects will have zero initial values.
PHDobjects/170107/ SimulateMissionSpace.m	739	Created by: Lefteris Loghis 18/4/2017 Version 8 Changes from version 7: 1. Added support for empty entry at charges table (MissionSpace{2}{2}). 2. Updated VersionOfCommand variable. Till now was 4. Changes from version 6: 1. Added one more cell at NewMissionSpace variable and now returns the output simulation folder' at the last cell. 2. Added DivertSimulationOutput which sends the output folder to different location than default. To be used with Random experimentsBuider. Changes from version 5: 1. Added OnlyContractingPower simulation setting. 2. Added ChangeForceConstant simulation setting. 3. Added DisableShieldEffect simulation setting. Changes from version 4: 1. Added collision modelling. 2. Renamed ForceLawBiases to SimulationSettings. 3. Created output for movie. Changes from version 3: 1. Added linear friction forces. 2. Added rotational friction forces. Changes from version 1: 1. Changed OutpFilename variable to DescriptionString. 2. Added ForceLawBiases variable. 3. Added field existance capability. The format of the command is as follows: NewMissionSpace = SimulateMissionSpace (MissionSpace, SimSteps, TimeStepSize, SimMethod, DescriptionString, SimulationSettings) Variables:

		1. MissionSpace: The mission space created by CreatePHDObjectsMission command. 2. SimSteps: The number of steps to be simulated (how many simulation iterations). 3. TimeStepSize: The time interval between each simulation step. 4. SimMethod: Sim method can be one of the following: a. 'timebased' used with fixed time steps of TimeStepSize. This is the only implemented yet. b. 'displacementbased' c. 'rotationbased' d. 'forcebased' e. 'momentbased' f. 'auto' to decide automatically based on what variable is becoming large. 5. DescriptionString: String that is saved in foles and describes the simulation that was carried. 6. SimulationSettings: Parameters used to add restrictions to force application. These are: a. InfinitySim : Simulation of infinity. The distance after which the force must be zero. b. MaxContractingPower : Maximum contraction power allowed. This will force simulation of finite power ability from object (eg vessel). c. MaxRepulsivePower : Maximum repulsive power allowed. This will force simulation of finite power ability from object (eg vessel). d. MaxSafetyPower : Maximum power that can result from shields touching. Again this is in order to adapt simulation to real systems. Variable is either a number or a cell array. If a number then this is assigned to InfinitySim. If a cell array, then the first column must be a string with the variable name and the second must be the value. e. UniformFieldVector: Assign a value if you would like a constant field inside space. Default NaN. f. FrictionForce: Assign a string value e.g.: '-k*u' or '-k*u^2' to create a linear friction force. u is speed variable. Default NaN. g. RotFrictionForce: Assign a string value e.g.: '-k*omega' of '-k*(omega^2)'. Use a string expression involving omega. Default NaN. h. EnableCollisionsSim: Set this parameter to 'Yes' if you want to enable shields collision. Default 'No'. i. CreateMovieFile: Set this to 'Yes' to output MovieOut.mat file in the simulation folder. This file is to be used with ProduceMovieOut function and movie command to visualize vessels movement. Default 'No'. j. OnlyContractingPower: Set this parameter to enable always contracting power except inside shield. Default 'No'. k. ChangeForceConstant: Change K at ApplyPhysicalLaw function. Default NaN. l. DisableShieldEffect: Set this parameter to 'Yes' in order to disable the effects of shield. Default 'No'.
PHDobjects/170107/ MultiVectEuclideanLength.m	46	Created by: Lefteris Loghis This function was last edited at 07/09/15 Version 1. If the vector is

		returns the Euclidean length of a vector, or an array with the lengths of multiple vectors. Syntax: length = MultiVectEuclideanLength (vectors_table, vectors_type) vectors_table: The vector or table of vectors. vectors_type: String 'row' or 'column' defining the type of vectors.
PHDobjects/170107/ RotatePHDobject.m	66	Created by: Lefteris Loghis 7/10/2015 Version 1 The format of the command is as follows: ObjParameters = RotatePHDobject (ObjCellArray, ObjIndex, new_angle) Receives as input the PHD objects structure, the object index in structure and the angle of turn in degrees. It outputs a column vector to replace in the original objects structure. Input variables: 1. ObjCellArray: The objects array. 2. ObjIndex: The index of the object. 3. new_angle: The angle in degrees.
PHDobjects/170107/ CompleteForcesTable.m	19	Created by: Lefteris Loghis 29/10/2015 Version 1 This command returns a full Forces Table, given as input only the upper triangular one. The rest of the elements below the main diagonal result by the addition of the negative transpose. The format of the command is as follows: ResultingForcesTable = CompleteForcesTable (InputForcesTable)
PHDobjects/170107/ CheckIfVector.m	24	Version 2 Lefteris Loghis 20/8/2019

		Changes from Version 1: Added conversion of input to row vector. Result = CheckIfVector (Input) This function checks if the input is a vector. It returns 1 if it is, 0 otherwise. Input: is the argument under test. Result is the output.
PHDobjects/170107/ CreatePHDObjectsMission.m	127	Created by: Lefteris Loghis 18/4/2017 Version 2 Changes from version 1: 1. Added support for empty entry at charges table (MissionSpace{2}{2}). The format of the command is as follows: MissionSpace = CreatePHDObjectsMission (NumOfObjects, ObjConstInitVect, ObjMotionVarInitVect,... ChargesConstInitVect, SpaceDimensions) It takes as input info about the objects, the charges and the space dimensions and produces a cell array to be used for simulation. The simulation takes action in a rectangular-shaped space with the dimensions provided at the SpaceDimensions variable. Input arguments: 1. NumOfObjects: Number of objects to create. If this number is omitted, one object will be created. 2. ObjConstInitVect: Table of column vectors providing initial data for every object. If this table is omitted, the initial data will be set to zero. The same will happen if some columns are omitted; the uninitialized objects will have zero values. Though this variable must have values for all objects since objects with zero dimensions and parameters are useless. ObjConstInitVect fields: 'width' 'length_A' 'length_AB' 'length_B' 'mass_A'

		'mass_AB' 'mass_B' 'quantity_A' 'quantity_B' 'ShieldSize' 3. ObjMotionVarInitVect: Table of column vectors describing initial motion parameters. ObjMotionVarInitVect fields: 'pos_x' 'pos_y' 'velocity_vector_x' 'velocity_vector_y' 'accelleration_vector_x' 'accelleration_vector_y' 'theta' 'omega' 'omega_dot' 4. PosRef: The object point that works as a reference for the [pos_x;pos_y] vector. This variable can be 'a', 'b', 'c', 'd' for the corners of the object or 'g' for its gravity center. All selections will finally end up converting to 'g'. 5. ChargesConstInitVect: Table of column vectors describing point charges inside 'ChargePos_x' 'ChargePos_y' 'ChargeValue' 'ChargePolarity' 6. SpaceDimensions: 'SpaceWidth' 'SpaceHeight' 'SpaceShield' Output: A three cell object, describing the PHD objects at its first cell, the charges at its second

		and the space at its third. For more info regarding PHDObjects see 'CreatePHDObjects.m'.
PHDobjects/170107/ CreateZeroFocesTable.m	15	Created by: Lefteris Loghis 29/10/2015 Version 1 This command returns an empty (all vectors zero) Forces Table. The format of the command is as follows: ZeroForcesTable = CreateZeroFocesTable (NumOfRows, NumOfColumns)
PHDobjects/170107/ LocatePHDobject.m	107	Created by: Lefteris Loghis 7/10/2015 Version 1 The format of the command is as follows: ObjParameters = RotatePHDobject (ObjCellArray, ObjIndex, Displacement) Receives as input the PHD objects structure, the object index in structure and the x and y displacement as a vector. It outputs a column vector to replace in the original objects structure. Input variables: 1. ObjCellArray: The objects array. 2. ObjIndex: The index of the object. 3. Displacement: The angle in degrees.
PHDobjects/170107/ ExtractValueFromParam.m	40	Created by: Lefteris Loghis 5/1/2016 Version 1 Syntax: ValueFound = ExtractValueFromParam (ParametersCellArray, ParameterToFind) Extracts value named as the string given to ParameterToFind variable, from a cell array (ParametersCellArray) whos first column is the string names of the values and the second column is the respective value. example: ParametersCellArray = {

		'InfinitySim', 1 'MaxContractingPower', 2 'MaxRepulsivePower', 3 'MaxSafetyPower', 4 }; ValueFound = ExtractValueFromParam (ParametersCellArray, 'MaxSafetyPower') » ValueFound = ExtractValueFromParam (ParametersCellArray, 'MaxSafetyPower') ValueFound = 4 »
PHDobjects/170107/ LookUpObjRow.m	31	Created by: Lefteris Loghis 16/8/2015 Version 1 The format of the command is as follows: RowNumber = LookUpObjRow (ObjCellArray, RowString) Receives as input the PHD objects structure and the type of information needed (as string) and outputs the row number to retrieve it from. If the string is invalid, it returns an empty vector.
PHDobjects/170107/ ProjectVectorToRotatedAxis.m	33	Created by: Lefteris Loghis 06/08/2016 Version 1 This function returns the InputVector vector as seen by the rotated by RotationAngle coordinate axes system. ProjectedVector = ProjectVectorToRotatedAxis(InputVector, RotationAngle) Input arguments: 1. InputVector: The vector that needs to be projected to the rotated axes system. 2. RotationAngle: The RotationAngle of turn of the new axis system. RotationAngle in rads.

		Output: ProjectedVector: The vector as seen by the rotated axis system.
PHDobjects/170107/ lef_isnumberVector.m	18	Created by: Lefteris Loghis 7/11/13 Version 1. f=lef_isnumberVector(v) This function returns 1 if the number v is a number vector and zero otherwise.
PHDobjects/170107/ lef_isrealarray.m	36	Created by: Lefteris Loghis 10/10/2015 Version 1 The format of the command is as follows: Y = lef_isarray (X, number_of_rows, number_of_columns) If only one argument is used then: If X is an array (all dimensions > 1) of numbers then Y = 1. If two arguments are used then: If X is an array OR VECTOR of numbers with rows = number_of_rows then Y = 1. If all arguments are used then : If X is an array OR VECTOR of numbers with rows = number_of_rows and columns = number_of_columns Y = 1.
PHDobjects/170107/ ApplyPhysicalLaw.m	156	Created by: Lefteris Loghis 8/10/2016 Version 3 Changes from version 2: 1. Added Spring Force capability. 2. Provided for error generation, in case of argument mistreatment. Changes from version 1: 1. Added OnlyContractingPower simulation setting. 2. Added ChangeForceConstant simulation setting. 3. Added DisableShieldEffect simulation setting. ForceSimParameters can be: InfinitySim : Simulation of infinity. The distance after which the force must be zero.

		MaxContractingPower : Maximum contraction power allowed. This will force simulation of finite power ability from object (eg vessel). MaxRepulsivePower : Maximum repulsive power allowed. This will force simulation of finite power ability from object (eg vessel). MaxSafetyPower : Maximum power that can result from shields touching. Again this is in order to adapt simulation to real systems.
PHDobjects/170107/ lef_isrealarrayorvector.m	36	Created by: Lefteris Loghis 10/10/2015 Version 1 The format of the command is as follows: Y = lef_isarray (X, number_of_rows, number_of_columns) If only one argument is used then: If X is an array (all dimensions > 1) of numbers then Y = 1. If two arguments are used then: If X is an array OR VECTOR of numbers with rows = number_of_rows then Y = 1. If all arguments are used then : If X is an array OR VECTOR of numbers with rows = number_of_rows and columns = number_of_columns Y = 1.
PHDobjects/170107/ ProduceMovieOut.m	88	Created by: Lefteris Loghis 20/8/2016 Version 1 This function produces a set of frames that all together form a movie. This movie can be diplayed using "movie" command. The purpose of this function is to provide the movie frames and the command you need to issue in order create the figure where the movie will be shown at. Command syntax: %[MovieFrames, FigureCommand] = ProduceMovieOut (MovieOut, FPS) Input Variables: 1. MovieOut: Is a variable created by SimulateMissionSpace.m, when in SimulationSettings parameter there exists a setting 'CreateMovieFile', 'Yes'. This variable is saved at file "MovieOut.mat" in the the directory of the desired simulation. Load this variable and in your variables a "MovieOut" variable will exist. This variable is the one used as "MovieOut". 2. FPS: Frames Per Second. This variable is the number of frames you need to see per second when the movie will be displayed.

		To play a movie use "movie" MATLAB command. The following lines are from movie command help: MOVIE(M,N,FPS) plays the movie at FPS frames per second. The default if FPS is omitted is 12 frames per second. Machines that can't achieve the specified FPS play as fast as they can.
PHDobjects/170107/ FindCenterOfMassForPHDobj.m	42	Created by: Lefteris Loghis 16/8/2015 Version 1 The format of the command is as follows: CenterOfMassVect = FindCenterOfMassForPHDobj (Objects) Finds the center of mass of a PHD object or objects, with 0,0 starting at their bottom left corner.
PHDobjects/170107/ RandomExperimentsBuilder.m	419	Created by: Lefteris Loghis 19/11/2016 Version 1 PLEASE REMEMBER TO CHANGE "VersionOfRndExpBuilder" VARIABLE WHEN NEW FEATURES HAVE BEEN ADDED!!!!!!!!!!!!!!!!!!!!!!!! The format of the function is: RandomExperimentsBuilder (NumOfExperiments,... NumOfObjects, NumOfObjectsVariance,... ObjConstInitVect, ObjConstInitVectVar,... ObjMotVarInitVect, ObjMotVarInitVectVar,... PosRef,... ChargesConstInitVect, ChrgConstInitVectVar,... SpaceDimensions) DefaultValueCellColumnIndex = 1; Abreviations to shorten names. ObjectConstantsInitialVectors = ObjConstInitVect ObjectConstantsInitialVectorsVariance = ObjConstInitVectVar ObjectMotionVariablesInitialVectors = ObjMotVarInitVect ObjectMotionVariablesInitialVectorsVariance = ObjMotVarInitVectVar ChargesConstInitVectVariance = ChrgConstInitVectVar

PHDobjects/170107/ GetObjectValue.m	52	Created by: Lefteris Loghis 17/8/2015 Version 1 The format of the command is as follows: RowNumber = LookUpObjRow (ObjCellArray, RowString) Receives as input the PHD objects structure and the type of information needed (as string) and outputs the row number to retrieve it from. If the string is invalid, it returns an empty vector.
PHDobjects/170107/ MovePHDobject.m	67	Created by: Lefteris Loghis 7/10/2015 Version 1 The format of the command is as follows: ObjParameters = RotatePHDobject (ObjCellArray, ObjIndex, Displacement) Receives as input the PHD objects structure, the object index in structure and the x and y displacement as a vector. It outputs a column vector to replace in the original objects structure. Input variables: 1. ObjCellArray: The objects array. 2. ObjIndex: The index of the object. 3. Displacement: The angle in degrees.
PHDobjects/170107/ CalculateMomentOfInertia.m	64	Created by: Lefteris Loghis 12/12/2015 Version 2 Difference from version 1 (15/9/2015): 1. Added support for objects with la = lb = 0 and w = 0. Returns the Moment Of Inertia of the object, with respect to an axis perpendicular to the objecct and passing from its center of mass. The center of mass is considered the (0,0) point for the calculations. The format of the command is as follows: MoI = CalculateMomentOfInertia (ma, la, mab, lab, mb, lb, xg, yg, w)

		ma : mass of region A. la : width of region A. mab : mass of region AB. lab : width of region AB. mb : mass of region B. lb : width of region B. xg : X coordinate of center of mass relative to bottom left corner of object. yg : Y coordinate of center of mass relative to bottom left corner of object. w : width of object (dimension Y). %(m*l/w)*(x1^3y1/3 +y1^3x1/3 - x1^3y0/3 - y0^3x1/3 - x0^3y1/3 - y1^3X0/3 + x0^3y0/3 + y0^3x0/3) m(l^2+w^2)/12 syms x y t = double(int(int((x^2+y^2), y, -2, 2), x, -5, -3))
PHDobjects/170107/ lef_isposint.m	12	variation of ISPOSINT True for positive integer values. Mark Beale, 11-31-97 Copyright (c) 1992-1998 by The MathWorks, Inc. $Revision: 1.3 $
lef_min.m	111	Function that returns all minimum values that are within a threshold from minimum.
RemoveAngleRateOffset.m	92	Created by: Lefteris Loghis 7/11/13 Version 1. OutVector = RemoveAngleRateOffset(YawRateVector) This function removes DC offset of Yaw Rate vector so as to have zero as idle value. Inputs: ------- YawRateVector: The Yaw Rate vector. IdlePointValue (optional): If the Idle point value of the input vector cannot be found automatically, then use this value instead. Output: -------

		OutVector: The normalized output vector. If an error occurs then the result is an empty vector ([]). To remove the DC offset we select the initial and final samples of yaw rate (where usually we don't touch the steering) and we create two new vectors where the variance is kept under a limit (e.g. 0.6). Then we take the median value. If the median value of the starting vector is within some limits of the ending vector, then we suppose that we can remove the median of both vectors as one from the original vector. This removes the DC offset.
VerifyDiffEqNeuralModel.m	306	Created by: Lefteris Loghis 14/04/14 Version 2. Changes from version 1: Added SpecialCommands usage. This function tests the accuracy of the vessel model produced by the neural networks that model X-axis acceleration and angular acceleration. VerifyDiffEqNeuralModel (AccXNNFilename, WdotNNFilename, ExperimentFilename, DecimationFactor, SignalsOffsetStruct) Inputs: ------- AccXNNFilename: Filename of the saved environment where the AccX neural net exists. WdotNNFilename: Filename of the saved environment where the Wdot neural net exists. ExperimentFilename: The filename of the experiment by which the neural net will be tested. DecimationFactor: Define the decimation factor by which all input signals were decimated before training the nets. SignalsOffsetStruct: The same as in Display measurement. Output: ------- Graphical representations presenting the difference between neural net outputs and real measured values.

		Check the input arguments: Use size(Wnet.inputs{1}.range) to find the number of past samples of inputs required, for the case of W neural.
FindMaxOfVectors.m	221	Created by: Lefteris Loghis 25/01/14 Version 1. It is a code that opens all experiments measurements and finds the maximum of each measurement and the max of all measurements for AccX, SpeedX, Wdot (YawRateDot), YawRate
NewralNetworksTransferFunction/ MatlabCode/ NNet_TF_Equ_v1.m	66	Function that evaluates a Perceptron neural network function using its equation and not directly the neural network object. Created by: Lefteris Loghis 15/9/18 Version 1.0 y1_3 = NNet_TF_Equ(NNet, x) Inputs: ------- NNet = The neural network object to extract the equation from. x = InputVector Output: ------- y1_3: The output of the one neuron of the third (output) layer. In this case this output is the accelleration at one sampling instant. TODO: ----- 1. Add the capability to change neural cell function using VariousParams cell array.
NewralNetworksTransferFunction/ MatlabCode/ NNet_TF_Equ.m	67	Function that evaluates a Perceptron neural network function using its equation and not directly the neural network object. Created by: Lefteris Loghis 15/9/18 Version 1.0 y1_3 = NNet_TF_Equ(NNet, x)

		Inputs: ------- NNet = The neural network object to extract the equation from. x = InputVector Output: ------- y1_3: The output of the one neuron of the third (output) layer. In this case this output is the accelleration at one sampling instant. TODO: ----- 1. Add the capability to change neural cell function using VariousParams cell array.
NewralNetworksTransferFunction/ MatlabCode/ EquationOfWdotNet.m	8	Neural Network equation that gives the same result with sim(NNet).
NewralNetworksTransferFunction/ MatlabCode/ EquationOfAccXNet.m	8	Neural Network equation that gives the same result with sim(NNet).
NewralNetworksTransferFunction/ MatlabCode/ WdotNet_TF.m	31	Function that outputs the text of the transfer function of Wdot neural network. Created by: Lefteris Loghis 11/9/18 Version 1.0
NewralNetworksTransferFunction/ MatlabCode/ AccX_net_TF.m	41	Function that outputs the text of the transfer function of AccX neural network. Created by: Lefteris Loghis 10/9/18 Version 1.0 y1_3 = AccX_net_TF(x) Inputs: ------- x = InputVectors = [ThrustVector; SteerVector; SpeedXVector; YawRateSamples]; Output: -------

		y1_3: The output of the one neuron of the third (output) layer. In this case this output is the accelleration at one sampling instant.
NewralNetworksTransferFunction/ MatlabCode/ VerifyDiffEqNeuralModelNNEq.m	341	Created by: Lefteris Loghis 18/09/15 Version 2. Changes from Version 1 (18/09/22): 1. Commented line 130, "SteerVectorDecimated = SteerVectorDecimated/128;" because it was dividing steer values twice with 128. 2. Changed division of thrust vector from 255 to 256 as in training. This function tests the accuracy of the vessel model produced by the neural networks that model X-axis acceleration and angular acceleration. VerifyDiffEqNeuralModelNNEq (AccXNNFilename, WdotNNFilename, ExperimentFilename, DecimationFactor, SignalsOffsetStruct) Inputs: ------- AccXNNFilename: Filename of the saved environment where the AccX neural net exists. WdotNNFilename: Filename of the saved environment where the Wdot neural net exists. ExperimentFilename: The filename of the experiment by which the neural net will be tested. DecimationFactor: Define the decimation factor by which all input signals were decimated before training the nets. SignalsOffsetStruct: The same as in Display measurement. Output: ------- Graphical representations presenting the difference between neural net outputs and real measured values.
lef_isnumber.m	13	Created by: Lefteris Loghis 6/11/13 Version 1. f=lef_isnumber(v) This function returns 1 if the number v is a number and zero otherwise.
GetMaxValues.m	48	Created by: Lefteris Loghis

		16/8/19 Version 2 Differences from version 1: 1. Added a variable that requests reply based on the selected version. Notes: Version 1 replied only SpeedX, AccX, YawRate, YawAcc. Version 2 added Thrust amd Steer denomalized. The reply is the denormalized values. Version 1 (10/02/14). It is a code that returns the decided maximum values for the vectors used in measurements. It takes no inputs and returns the cells array: MeasurementMaximumValues = { 'SpeedX', 0, 30; 'AccX', -3.4, 3.4; 'YawRate', -50, 50; 'YawAcc', -130, 130 };
decmt.m	91	Created by: Lefteris Loghis 20/1/14 Version 3. Changes from version 2: Corrected i and j indexes to i_index and j_index. Changes from version 1: Added optional decimation filter initialization feature. TimeSeriesOut = decmt(TimeSeriesIn, DecimationFactor) This function generates a time series vector that is the original after passing from a moving average filter (with DecimationFactor taps) and resampled by DecimationFactor slower sampling rate. The resulting time series is DecimationFactor times shorter! TAKE CARE: NO DELAY COMPENSATION TECHNIQUES HAVE BEEN APPLIED! Inputs: ------- TimeSeriesIn: The input time series vector (one dimension). DecimationFactor: The factor by which input time series are decimated. FilterInitializer (Optional): Initialization value for moving average filter taps.

		Output: ------- TimeSeriesOut: The decimated after filtering time series.
DisplayMeasurement.m	790	Created by: Lefteris Loghis 26/11/13 Version 7. Changes from version 6: Added combined subtraction functionality for SpeedX. Added SignalsOffsetStruct.SubtractionTable for the above purpose. Added SignalsOffsetStruct.ThrustNeededForMotionStart, which is the minimum amount of thrust needed to create effective thrust to move the vessel. Changes from version 5: Added SpecialCommands parameter and removed parameters for time units and close figures. Now all these functions can be implemented using the special Commands parameter. Supports measurements table besides measurement file. Changes from version 4: Added DoNotCloseFigures optional variable. Changes from version 3: Corrected NormalizeMeasurementVector function. Changes from version 2: Supports TimeUnitsInPlot parameter. MeasCellArray = DisplayMeasurement (MeasurementFilename, SelectedSignals, SignalsOffsetStruct, SpecialCommands) MeasCellArray = DisplayMeasurement (MeasurementFilename, SelectedSignals, SignalsOffsetStruct, TimeUnitsInPlot) Function that displays measurements of Autopilot_IV output files after being processed by experiment converter program. It is an evolution of the DispMeas function. Inputs: %------- MeasurementFilename Expects a csv file with the following columns separated by spaces: Frame No, Time(ms), RPM1, RPM2, Thrust, Steer, Direction,

		AccX(g), AccY(g), AccZ(g), Roll(Deg/sec), Pitch(Deg/sec), Yaw(Deg/sec), CompX, CompY, CompZ. Alternatively a measurement table can be provided, with the same structure in its columns. SelectedSignals This is a cell array, each cell of which can have one of the following values 'Time' 'RPM1' 'RPM2' 'Thrust' 'Steer' 'Direction' 'AccX' 'AccY' 'AccZ' 'RollRate' 'PitchRate' 'YawRate' 'YawIMU' 'YawCompass' 'SpeedX' 'Pitch' A cell of this array can also be again cell array with one of these values. This will create a combined plot with all other signals referenced to the first. If referencing is not desired, the last cell should be 'No'. If the last cell is 'Yes' the referencing will be done. SignalsOffsetStruct A struct of the following type: SignalsOffsetStruct.TimeStart = NaN; SignalsOffsetStruct.TimeEnd = NaN; SignalsOffsetStruct.ThrustOffst = NaN; SignalsOffsetStruct.SteerOffst = NaN; SignalsOffsetStruct.DirectionOffst = NaN; SignalsOffsetStruct.AccXOffst = NaN; SignalsOffsetStruct.AccYOffst = NaN;

```
            SignalsOffsetStruct.AccZOffst = NaN;
            SignalsOffsetStruct.RollRateOffst = NaN;
            SignalsOffsetStruct.PitchRateOffst = NaN;
            SignalsOffsetStruct.YawRateOffst = NaN;
            SignalsOffsetStruct.PitchCompassOffst = NaN;
            SignalsOffsetStruct.SpeedXOffst = NaN;
            SignalsOffsetStruct.SubtractionTable = NaN;
            SignalsOffsetStruct.ThrustNeededForMotionStart = NaN;

            This struct presented is the default. Whenever there is
            a NaN at the value, DisplayMeasurement applies the default
            offset removal or whatever procedure. If there is a value
            then the requested value is used for the respective vector
            processing.

SpecialCommands
            A cell array containing one of the following:
                        'Sec' : Time units will be seconds.
                        'SampleNo' : Time units will be sample number.
                        'CloseFigures' : All the open plots will be closed.
                        'DoNotCloseFigures' : The open plots will be retained.
                        'SubgFromAccX' : The g vector will be subtracted from AccX
                                                                using magnetometer value as reference for the angle.
                        'DoNotSubgFromAccX' : The g vector will not be subtracted from AccX.

Outputs:
--------
MeasCellArray
            A cell array containing two cells:
                        1. A structure of all processed signals as follows:
                                    MeasCellArray{1}{selection}.Vector: which contains the vector values.
                                    MeasCellArray{1}{selection}.Name: which contains the vector name.
                                    MeasCellArray{1}{selection}.Units: which contains the vector units.
                        2. A structure cell which explains the way the signals are stored in MeasCellArray{1}.
                                    See example for details.
Example:
Selection = {
```

		'YawIMU'
		'YawCompass'
		}
		Selection{3} = {
		'YawIMU'
		'YawCompass'
		};
		Selection
		
		Selection =
		
		'YawIMU'
		'YawCompass'
		{2x1 cell}
		
		MeasCell = DisplayMeasurement('Verde_2_6_ForMatlab.csv', Selection);
		MeasArray{2}
		
		ans =
		
		'Time'
		'RPM1'
		'RPM2'
		'Thrust'
		'Steer'
		'Direction'
		'AccX'
		'AccY'
		'AccZ'
		'RollRate'
		'PitchRate'
		'YawRate'
		'YawIMU'
		'YawCompass'
		'PitchCompass'
		'SpeedX'

ApplyVectorMax.m	49	Created by: Lefteris Loghis

		09/02/14 Version 1. It is a code that applies (forces) the maximum value of a specified vector to be as defined for it by clipping the absolutely larger positive and negative values. Inputs: ------- InputVector: What it says. The vector you need to clip. VectorType: One of 'SpeedX', 'AccX', 'YawRate', 'YawAcc'. Outputs: -------- ReturnVector: The clipped input vector.
EuclideanLength.m	20	Created by: Lefteris Loghis This function last edited at 24/08/15 Version 3. returns the Euclidean length of a vector. Syntax: length = EuclideanLength (vector)
SimulateNet.m	495	Created by: Lefteris Loghis 10/5/14 Version 4. Changes from version 3: Added ErrorPercentage parameter. Changes from version 2: The software finds net name from NNFilename. This function creates and saves a neural network trained to produce angular rate samples, by the series contained in the file named Filename. SimulateNet (NNFilename, ExperimentFilename, DecimationFactor, SignalsOffsetStruct) Inputs: ------- NNFilename: Filename of the saved environment where the neural net exists. ExperimentFilename: The filename of the experiment by which the neural net will be tested.

		DecimationFactor: Define the decimation factor by which all input signals were decimated before training the net. ErrorPercentage: A float < 1.0 which is the maximum Gaussian error added to input signals. Output: ------- Saves workspace created under the name WNeural_yyyymmdd_hhmm.mat in folder %$currentDirectory\WNeural. The network will be trained as follows: | | | __ __ | | | Thrust[n-(k-1)] | | | | ... | | | | Thrust[n-1] | | | | Thrust[n] | | | | ... | | f | | Steer[n] | | = W[n+1] | | ... | | | | SpeedX[n] | | | | ... | | | | W[n] | | | -- -- | | | Check the input arguments: Use size(Wnet.inputs{1}.range) to find the number of past samples of inputs required, for the case of W neural.
lef_isnumberVector.m	18	Created by: Lefteris Loghis 7/11/13 Version 1. f=lef_isnumberVector(v) This function returns 1 if the number v is a number vector and zero otherwise.
ExpandVector.m	21	Pseudo interpolate a decimated vector in order to compare it with the original signal graphically.

CheckAcc.m	667	Created by: Lefteris Loghis 2/2/14 Version 5. Changes from version 4: Added combined subtraction mode. Changes from version 3: Added capability of changing a plot type. NewOffstTable = CheckAcc (MeasurementFilename, OffsetsTable) Function that plots speed and provides means for creating the offsets and multipliers needed to get a logical speed vector. Inputs: ------- MeasurementFilename Expects a csv file with the following columns separated by spaces: Frame No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ OffsetsTable A table containing initial values and offset steps as follows: __ __ | AccX_offset AccX_step | | Pitch_offset Pitch_step | | AccX _mult AccX_m_st | |__ TimeStart TimeEnd __| If the table extends more than 4 rows, then the rest above 4 contain the a's and b's for subtracting y = a*(x-b), y >= 0 from SpeedX. example: offst = [1.2 0.1 0 1.0 9.81 0.01

		NaN NaN]; CheckAcc ('CorrectedMeasurements/Verde_2_11_ForMatlab.csv', offst)
TrainNNetForWdot.m	614	Created by: Lefteris Loghis 12/02/14 Version 4. Changes From Version 3: Normalized all signals to [0,1] or [-1,1]. Made use of GetMaxValues. Changes From Version 2: Added format command. Normalized error target along with output signal. Corrected the value of cells to 2n+1. Changes From Version 1: Corrected Ts in the differentiation of decimated signals. Ts in this case is the original multiplied by the decimation factor. This function creates and saves a neural network trained to produce angular acceleration samples, by the series contained in the file named Filename. It is the function that treats case 2a of neural network types described for PHD. This type of network is used to produce Wdot for the differential equation model of the vessel. TrainNNetForWdot (Filename, SignalsOffsetStruct, NNetTrainingParams, SpecialCommands) Inputs: ------- Filename: Filename of the experiment selected. DecimationFactor: Factor by which the training vectors will be decimated. k: Number of past samples per input. NetType (optional): Type of neural net internal neurons. Possible values: 1.'logsig' (default) 2.'tansig' Epochs (optional): The max training epochs for the net. Default = 8000. Output: ------- Saves workspace created under the name WNeural_yyyymmdd_hhmm.mat in folder

		%$currentDirectory\WdotNeural. The network will be trained as follows: | | | __ __ | | | Thrust[n-(k-1)] | | | | | ... | | | | | Thrust[n-1] | | | | | Thrust[n] | | | | | ... | | | f | | Steer[n] | | = W[n+1] | | ... | | | | | SpeedX[n] | | | | | ... | | | | | W[n] | | | | -- -- | | |
RetrainNet.m	553	Function that trains a neural network to match the vessel's accelleration response. Created by: Lefteris Loghis 26/11/13 Version 3.0 Changes from version 2.0: Added version variable. Added normalization to all input and training vectors. Modified SimulateNet call. Added more info at ReadMe output. Changes from version 1.0: Uses DisplyaMeasurement to get the appropriate vectors. Asks to proceed to Newral Net training. Supports decimating the input signals. Supports multiple past inputs per signal as neural input. AccXNeural(Filename, SignalsOffsetStruct, NNetTrainingParams, SpecialCommands) Inputs: ------- Filename: The name of the measurement to train newral network with.

		SignalsOffsetStruct: The structure used by DisplayMaesurement to input the known offsets needed for the measurement. Signal offset struct by default is: SignalsOffsetStruct.TimeStart = NaN; SignalsOffsetStruct.TimeEnd = NaN; SignalsOffsetStruct.ThrustOffst = NaN; SignalsOffsetStruct.SteerOffst = NaN; SignalsOffsetStruct.DirectionOffst = NaN; SignalsOffsetStruct.AccXOffst = NaN; SignalsOffsetStruct.AccYOffst = NaN; SignalsOffsetStruct.AccZOffst = NaN; SignalsOffsetStruct.RollRateOffst = NaN; SignalsOffsetStruct.PitchRateOffst = NaN; SignalsOffsetStruct.YawRateOffst = NaN; SignalsOffsetStruct.PitchCompassOffst = NaN; SignalsOffsetStruct.SpeedXOffst = NaN; These are the values that it will take if you ommit the parameter. NNetTrainingParams: The parameters used to train the neural network. These are: DecimationFactor = 1; (By which factor will the signals be decimated. 1 = no decimation). k = 1; (How many past vectors will be used for the neural training). NetType = {'logsig' 'purelin'}; (The types of the middle and output neurons). Epochs = 8000; (How many training epochs will there be). Goal = 1e-5; (What is the target error of the training). The values presented here are the ones assigned if this parameter is ommitted. SpecialCommands: Special parameters used for DisplayMeasurement Outputs: -------- A newral network trained by the requested measurement stored in {work_folder}/AccXNeural/AccXNeural_{current_date}
TstDifferenceEqNeuralModel.m	212	Created by: Lefteris Loghis 17/02/14 Version 1. This function tests the accuracy of the vessel model produced by the neural networks
StoreAngles.m	79	Function that calculates the angles of the boat regarding Earth's

		magnetic field. Theta is the angle regarding to X-axis, referring at XY plane. Positive counter-clockwise. Phi is the angle of the B vector with the XY plane. Thetaacos is ArcCos(x/|B|). Thetaasin is the same but calculating as ArcSin(y/|B|) Phiacos is ArcCos (XY_length/length). Phiasin is ArcSin (z/length). AnglesTable = StoreAngles(VectorsTable) Inputs: ------- VectorsTable: It is the table of [X Y Z] vectors as sent by the magnetometer. Outputs: -------- AnglesTable: A table like this: [Theta Theta Diff1 Phi Phi Diff2] where Theta: the angle regarding to X-axis, referring at XY plane. Positive counter-clockwise. Diff1: the difference between the two theta approximations. Phi: the angle of the B vector with the XY plane. Diff2: the difference between the two theta approximations.
FixMeasurement.m	254	Created by: Lefteris Loghis 13/11/13 Version 1. OutVector = FixVector(VectorIn, SamplesToFix, SamplesToUseAsRef) This function fixes problematic samples specified using random generated samples resembling to reference samples. Inputs: ------- VectorIn: The whole vector. SamplesToFix: Array of selected samples to fix using SamplesToUseAsRef as reference. SamplesToUseAsRef: Samples to be used as reference, in order to generate min max values for the new samples.

		Output: ------- OutVector: The fixed vector.
ThrustSpeed.m	274	Function that converts measurements of Autopilot_IV Version 1.0 Expects a csv file with the following columns separated by spaces: Frame_No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ
SimulinkDiffceNeural.m	33	Created by: Lefteris Loghis 25/03/14 Version 1. This function is used in simulink to apply the input vector to the neural network. Inputs: ------- InputVector: The input vector containing all necessary input instances. Output: ------- OutVector: The output of the neural network.
SimNet.m	264	Created by: Lefteris Loghis 1/11/13 Version 1. This function creates and saves a neural network trained to produce angular rate samples, by the series contained in the file named Filename. SimulateNet (NNFilename, NetName, ExperimentFilename, DecimationFactor) Inputs: ------- NNFilename: Filename of the saved environment where the neural net exists. NetName: The name of the neural net inside the environment.

		ExperimentFilename: The filename of the experiment by which the neural net will be tested. DecimationFactor: Define the decimation factor by which all input signals were decimated before training the net. Output: ------- Saves workspace created under the name WNeural_yyyymmdd_hhmm.mat in folder %$currentDirectory\WNeural. The network will be trained as follows: | | | __ __ | | | Thrust[n-(k-1)] | | | | | ... | | | | Thrust[n-1] | | | | Thrust[n] | | | | ... | | f | | Steer[n] | | = W[n+1] | | ... | | | | SpeedX[n] | | | | ... | | | | W[n] | | | -- -- | | | Check the input arguments: Use size(Wnet.inputs{1}.range) to find the number of past samples of inputs required, for the case of W neural. default call: SimNet ('WNeural/WNeural_2013111_90/WNeural_2013111_90.mat', ... %'Wnet', 'Verde_2_6_ForMatlab.csv', 30);
GetVector.m	56	Created by: Lefteris Loghis 14/11/13 Version 2. Changes from version 1: Supports also MeasurementCellArray from

		GetMeasurement command. OutVector = GetVector(MeasurementCellArray, VectorSelectionString) This function returns the selected, by the second string parameter, vector from a cell array returned from DisplayMeasurement command. Inputs: ------- MeasurementCellArray: The measurement cell array returned from an execution of DisplayMeasurement command. VectorSelectionString: The string which specifies which vector to return. The string value must be one of the following. 'Time' 'RPM1' 'RPM2' 'Thrust' 'Steer' 'Direction' 'AccX' 'AccY' 'AccZ' 'RollRate' 'PitchRate' 'YawRate' 'YawIMU' 'YawCompass' 'SpeedX' Output: ------- OutVector: The selected vector.
GetVectorIndex.m	48	Created by: Lefteris Loghis 13/11/13 Version 1. OutVector = GetVector(MeasurementCellArray, VectorSelectionString) This function returns the selected, by the second string parameter, vector index from a cell array returned from DisplayMeasurement command.

		Inputs: ------- MeasurementCellArray: The measurement cell array returned from an execution of DisplayMeasurement command. VectorSelectionString: The string which specifies which vector to return. The string value must be one of the following. 'Time' 'RPM1' 'RPM2' 'Thrust' 'Steer' 'Direction' 'AccX' 'AccY' 'AccZ' 'RollRate' 'PitchRate' 'YawRate' 'YawIMU' 'YawCompass' 'SpeedX' Output: ------- OutVector: The selected vector index.
NormalizeSteer.m	83	Created by: Lefteris Loghis 12/2/14 Version 1. OutVector = NormalizeSteerAroundZero(SteerVector, MiddlePointValue) This function normalizes steer vector around zero. This means that it removes any DC offset so as to have zero as center value. The maximum theoretical span of steer is 0 to 255, but in practice I see a -91 to +89 decimal deviation from the middle point. The usual middle point is 121. This will be removed from all measurements

		unless otherwise selected by MiddlePointValue variable. After removing DC offset, the vector will be divided by 128, in order to be between -1 and 1, thus making it suitable for neural network training. Inputs: ------- SteerVector: The steer vector. MiddlePointValue (optional): If the middle point value of the input vector does not match its median, then use this value instead. This number must be between 0 and 255. Most possibly it will be something close to 121. Output: ------- OutVector: The normalized output vector. If an error occurs then the result is [].
GetMeasurement.m	58	Created by: Lefteris Loghis 13/11/13 Version 1. MeasCellArray = GetMeasurement (MeasurementFilename) Function that gets measurements of Autopilot_IV output files after being processed by experiment converter program. Inputs: ------- MeasurementFilename Expects a csv file with the following columns separated by spaces: Frame No Time(ms) RPM1 RPM2 Thrust Steer Direction AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Yaw(Deg/sec) CompX CompY CompZ Outputs: -------- MeasCellArray A cell array containing two cells: 1. A structure of all processed signals as follows:

		2. A structure cell which explains the way the signals are stored in MeasCellArray{1}.
StatLin/ TempFileForCommandsEntry.matlab	223	A set of commands to send to Matlab in order to create operating points.
StatLin/ LocateNeighbouringPoints.m	121	Version 1 Lefteris Loghis 11/9/2019 This function takes as input a cell array that is the output of command FindContTimeModelOperatingPoint and returns the neighbouring points of the requested input vector. The selected inout vector must be a member of the vectors used
StatLin/ CommandsForJacobianComputation.matlab	80	Set of commands to produce Jacobians for linearizing vessel NNet models.
StatLin/ FindContTimeModelOperatingPoint.matlab	86	This file is only for debug of FindContTimeModelOperatingPoint.m file. The actual operation of this command is within CreateLinearizedModels.m
StatLin/ FindContTimeModelOperatingPoint.m	701	Lefteris Loghis Version 3 2/1/2022 Changes from version 2: 1. Changed expression length(OperatingPoint) in line 193 to OpPntVariables where [OpPntVariables, OpPointCols] = size(OperatingPoint); 2. Added try-catch blocks to block annoying errors. 3. Added debgging switches. 4. Added CommentText to the return value. Version 2 18/8/2020 Changes from version 1: 1. Added ExecutionTimeReport parameter at PlotSettings variable to report the calculations elapsed time. 2. Added EnableSurfacePlotting parameter at PlotSettings variable to disable plotting surfaces if needed. Previous Version: 1 1/9/2019 Function returning the operating point of the continuous time model based on the

given inputs. The operating point is selected by zeroing (or approximating zero)
the acceleration output of the AccX and Wdot newral networks. The thresholds parameters
can be used to specify how close to zero should the samples be to be considered
as preferable or under consideration.

Function prototype:
OpPoint = FindContTimeModelOperatingPoint (AccXNNFilename, WdotNNFilename, OperatingPoint, Thresholds)

Variables:

AccXNNFilename:
%----------
The path and filename to the neural network for acceleration along X-axis.

WdotNNFilename:
%----------
The path and filename to the neural network for angular acceleration.

OperatingPoint:
%----------
A cell array like the one follows, where NaN is used for the variables left free.
The fixed variables are fixed using de-normalized values (e.g. speed 0kM/h to 30Km/h..
OperatingPoint = {
'Thrust', NaN, 1, [0 255]
'Steer', NaN, 1, [0 249]
'SpeedX', 15, 0.01, [0 30]
'YawRate', 0, 0.01, []
}
The first column of OperatingPoint variable is the names of the parameters involved.
Possible names are:
'Thrust'
'Steer'
'SpeedX'
'YawRate'
The second column is NaN if the variable is not fixed and should be varied.
The third column is the varying step.
The fourth column is a row vector containing the values between which the variable

		will be varied (min, max). If empty the whole range will be used; the variation range is found using GetMaxValues command. Thresholds: %---------- A struct containing the thresholds to search for AccX and YawAcc. Have in mind that these thresholds are for normalized AccX, YawAcc (or Wdot). Thresholds.AccX = 0.001; Thresholds.YawAcc = 0.0001; Command output: OpPoint %---------- A cell array of the following structure: OpPoint = 'TotalInpVect' [4x650 double] 'TotalInpVectIndices' [4x650 double] 'TotalOutpVect' [2x650 double] 'CriticalNorm', [1x? double] 'SecondaryNorm', [1x? double] 'ApproxPoint' [1x1 struct] 'ClosestPoints' [1x1 struct] 'CommentText' [?x1 cell array] The first column is a set of strings describing the content of the second column. Specifically: 'TotalInpVect' is the full vector used as test input to the system. 'TotalInpVectIndices' is the array containing the indexes of samples as column vectors. To fixed values the matrix contains NaN, to show that the variable at this index is not varying. 'TotalOutpVect' is the vector of normalized outputs (responses) to the respective TotalInpVect vector. 'CriticalNorm' is the norm of the vector formed by normalized [AccX, Wdot] to the respective TotalInpVect vector.

		'SecondaryNorm' is the norm of the vector formed by normalized [Thrust, Steer] to the respective TotalInpVect vector. 'ApproxPoint' is a structure containing fields: 'Vect' --> Is the input vector that led to approved points using thresholds as creterion. 'Approx' --> Is the nnet output of the respective input vector found at the same index at Vect field. 'Indices' --> The i,j indexes of the vectors created based on the requested variance. 'Norms' --> The norm (distance) of the Approx column normalized vector from [AccX = 0; Wdot = 0] point. 'ClosestPoints' is a struct containing fields: 'Norm' --> The norm (distance) of the closest point from [AccX = 0; Wdot = 0] point. 'Indices' --> the indexes [i;j] of the i-th, j-th value of varying vectors. 'Vect' --> The input vector that results to the closest point. 'VectIndex' --> The index number of the column of ApproxPoint.Vect containing the closest column vector(s). 'NNetsOut' --> The normalized output of neural networks at the closest points. 'VectDenorm' --> The de-normalized vector(s) of the input vector(s) that result to closest point. 'NNetsOutDenorm' --> The denormalized output of neural networks at the closest point. 'CommentText' is a cell array containing comments for all forcefully accepted points that failed equilibrium criteria.
StatLin/ TestsWithAccXNet.m	414	This file compares AccX neural net from Matlab with the set of equations we say that are used to produce its output.
StatLin/ AnalyzeLinModelsSetCommands.matlab	24	Reference for the use of AnalyzeLinModelsSet command and how it was used in the PHD.
StatLin/ CompareSim2EqOutWdotNet.m	214	This file compares AccX neural net from Matlab with the set of equations we say that are used to produce its output.
StatLin/ CompareSim2EqOutAccXNet.m	226	This file compares AccX neural net from Matlab with the set of equations we say that are used to produce its output.
StatLin/ CreateLinearizedModelsCommands.matlab	77	Reference for the use of CreateLinearizedModels command.
StatLin/ NumericalJacobian.m	226	Lefteris Loghis Version 1 22/9/2019

		Function returning the numerical calculation of the jacobian of the system, described by newral nets AccXNNFilename and WdotNNFilename.
		
		Function prototype:
		NumJacobs = NumericalJacobian(AccXNNFilename, WdotNNFilename, OperatingPoint, OpPoint, InpVector)
		
		Variables:
		
		AccXNNFilename:
		---------- 	
		The path and filename to the neural network for acceleration along X-axis.
		
		WdotNNFilename:
		---------- 	
		The path and filename to the neural network for angular acceleration.
		
		OperatingPoint:
		---------- 	
		A cell array like the one follows, where NaN is used for the variables left free.
		The fixed variables are fixed using de-normalized values (e.g. speed 0kM/h to 30Km/h..
		OperatingPoint = {
		'Thrust', NaN, 1, [0 255]
		'Steer', NaN, 1, [0 249]
		'SpeedX', 15, 0.01, [0 30]
		'YawRate', 0, 0.01, []
		}
		The first column of OperatingPoint variable is the names of the parameters involved.
		Possible names are:
		'Thrust'
		'Steer'
		'SpeedX'
		'YawRate'
		The second column is NaN if the variable is not fixed and should be varied.
		The third column is the varying step.
		The fourth column is a row vector containing the values between which the variable will be varied (min, max). If empty the whole range will be used; the variation range is found using GetMaxValues command.

		OpPoint ---------- A cell array of the following structure: OpPoint = 'TotalInpVect' [4x650 double] 'TotalInpVectIndices' [4x650 double] 'TotalOutpVect' [2x650 double] 'ApproxPoint' [1x1 struct] 'ClosestPoints' [1x1 struct] The first column is a set of strings describing the content of the second column. Specifically: 'TotalInpVect' is the full vector used as test input to the system. 'TotalInpVectIndices' is the array containing the indexes of samples as column vectors. To fixed values the matrix contains NaN, to show that the variable at this index is not varying. 'TotalOutpVect' is the vector of normalized outputs (responses) to the respective TotalInpVect vector. 'ApproxPoint' is a structure containing fields: 'Vect' --> Is the input vector that led to approved points using thresholds as creterion. 'Approx' --> Is the nnet output of the respective input vector found at the same index at Vect field. 'Indices' --> The i,j indexes of the vectors created based on the requested variance. 'Norms' --> The norm (distance) of the Approx column normalized vector from [AccX = 0; Wdot = 0] point. 'ClosestPoints' is a struct containing fields: 'Norm' --> The norm (distance) of the closest point from [AccX = 0; Wdot = 0] point. 'Indices' --> the indexes [i;j] of the i-th, j-th value of varying vectors. 'Vect' --> The input vector that results to the closest point. 'VectIndex' --> The index number of the column of ApproxPoint.Vect containing the closest column vector(s). 'NNetsOut' --> The normalized output of neural networks at the closest points. 'VectDenorm' --> The de-normalized vector(s) of the input vector(s) that result to closest point. 'NNetsOutDenorm' --> The denormalized output of neural networks at the closest point.

		InpVector ---------- A vector containing the values at which the Jacobian will be calculated. The vector elements are: [Thrust_value Steer_value SpeedX_value YawRate_value] Command output: NumJacobs ---------- Two members of this structural type variable: NumJacobs = Normal: [2x2 double] Minus: [2x2 double] 'NumJacobs.Normal' is the numerical approximation of Jacobian by differences, using
StatLin/ CompJacobian.m	196	Lefteris Loghis Version 2 16/8/2020 1. Removed OperatingPoint variable, since the values of velocity (SpeedX) and rate of turn (Yaw Rate) will be provided through the InpVector variable. 2. Provided the capability to normalize the input vector or keep it as is.. Previout version: 1 25/9/2019 Function returning the jacobian of the system computationally not numerically, described by newral nets AccXNNFilename and WdotNNFilename. Function prototype: CompJacobs = CompJacobian(AccXNNFilename, WdotNNFilename, OperatingPoint, InpVector) Variables: AccXNNFilename:

		---------- 	
		The path and filename to the neural network for acceleration along X-axis.
		
		WdotNNFilename:
		---------- 	
		The path and filename to the neural network for angular acceleration.
		
		This function used to have a variable named OperatingPoint where:
		OperatingPoint:
		---------- 	
		A cell array like the one follows, where NaN is used for the variables left free.
		The fixed variables are fixed using de-normalized values (e.g. speed 0kM/h to 30Km/h..
		OperatingPoint = {
		'Thrust', NaN, 1, [0 255]
		'Steer', NaN, 1, [0 249]
		'SpeedX', 15, 0.01, [0 30]
		'YawRate', 0, 0.01, []
		}
		The first column of OperatingPoint variable is the names of the parameters involved.
		Possible names are:
		'Thrust'
		'Steer'
		'SpeedX'
		'YawRate'
		The second column is NaN if the variable is not fixed and should be varied.
		The third column is the varying step.
		The fourth column is a row vector containing the values between which the variable
		will be varied (min, max). If empty the whole range will be used; the variation range
		is found using GetMaxValues command.
		
		InpVector
		---------- 	
		A vector containing the values at which the Jacobian will be calculated. The vector elements are:
		[Thrust_value
		Steer_value
		SpeedX_value
		YawRate_value]

		DoNotNormalize ---------- If 1 the function will not normalize the input vector. If omitted or 0 the function will normalize the vector before using it for the calculations. This function is usually combined with FindContTimeModelOperatingPoint. The output of which contains both the normalized and the de-normalized vectors resulting from finding the operating point needed. If OpPoint{5,2}.Vect is used, the normalized output is going to be used and for this function the normalization must be disabled. If instead OpPoint{5,2}.VectDenorm is used, then the denormalized version of the vector is going to be used, thus the normalization must be enabled. Command output: CompJacobs ---------- A structure containing the computationally calculated Jacobian components. The structure of this matrix is: CompJacobs.A = [Theta_a_over_theta_u Theta_a_over_theta_w Theta_w_dot_over_theta_u Theta_w_dot_over_theta_w]; CompJacobs.B = [Theta_a_over_theta_y Theta_a_over_theta_d Theta_w_dot_over_theta_y Theta_w_dot_over_theta_d];
StatLin/ AnalyzeLinModelsSet.m	213	Created by: Lefteris Loghis 16/11/20 Version 1. This function analyzes the linear model set that has resulted from application of command CreateLinearizedModels, and returns the range and nominal value of every element of matrices A, B
StatLin/ AnalyzeStabilityAndVectLimsCommands.matlab	29	Command used to app;y AnalyzeStabilityAndVectLims command.
StatLin/	356	Created by: Lefteris Loghis

AnalyzeStabilityAndVectLims.m		10/12/22 Version 2 Changes from Version 1 (26/12/21). 1. Changes stability checks to strict > 0 for unstable systems. 2. Changed poles comparison by compariung only the real part. 3. Added TotalPolesVector and ModelPoles return fields. This function analyzes the linear model set that has resulted from application of command CreateLinearizedModels, and returns the for every model the stability proprties. It also returns the limits that were found for every value of input vector used (recovered from the models). LinModelsCellStabilityResults = AnalyzeStabilityAndVectLims(LinModSetFilename, OutputFolderPath) Input Variables: ---------- LinModSetFilename: The filename of the model set to be analyzed. OutputFolderPath: Path to store the output. MarginalStabilityLim: Limit of marginal stability. If not used, MarginalStabilityLim will be set to 0. Output Variable: ---------- LinModelsCellStabilityResults.StbArray: A matrix containing -1 for unstable systems, 0 for marginally stable and 1 for stable systems. LinModelsCellStabilityResults.UnstbList: List of unstable systems indexes. Cell array containing [i, j] indexes. LinModelsCellStabilityResults.InpVectLims: Limits of input vectors scanned amongst all models. LinModelsCellStabilityResults.TotalPolesVector: Vector containing all poles found. LinModelsCellStabilityResults.ModelPoles: Poles for every model at [i j] index of LinearizedModelsCellArray which is found in LinModSetFilename as existing parameter after loading the file. Reminder: The input vector is as follows: [Thrust Steer SpeedX

		YawRate]
StatLin/ TestsWithWdotNet.m	344	This file compares Wdot neural net from Matlab with the set of equations we say that are used to produce its output.

StatLin/
ExtractMBCLUTs.m
	352	Lefteris Loghis
		16/1/2022
		
		Function returning LUTs that predict the [AccX, W_dot]' output of the model for every equillibrium point.
		The selection of LUT is based on the respective [SpeedX, YawRate]' vector. The output of this function
		is to be used by an MBC (Model Based Controller), in order to decide its outputs.
		
		Function prototype:
		OpPoint = MBCStruct (AccXNNFilename, WdotNNFilename, OperatingPoint, Thresholds)
		
		Variables:
		
		AccXNNFilename:
		---------- 	
		The path and filename to the neural network for acceleration along X-axis.
		
		WdotNNFilename:
		---------- 	
		The path and filename to the neural network for angular acceleration.
		
		OperatingPoint:
		---------- 	
		A cell array like the one follows, where NaN is used for the variables left free.
		The fixed variables are fixed using de-normalized values (e.g. speed 0kM/h to 30Km/h..
		OperatingPoint = {
		'Thrust', NaN, 1, [0 255]
		'Steer', NaN, 1, [0 249]
		'SpeedX', 15, 0.01, [0 30]
		'YawRate', 0, 0.01, [-50 50]
		}
		The first column of OperatingPoint variable is the names of the parameters involved.
		Possible names are:
		'Thrust'

		'Steer' 'SpeedX' 'YawRate' The second column is NaN if the variable is not fixed and should be varied. The third column is the varying step. The fourth column is a row vector containing the values between which the variable will be varied (min, max). If empty the whole range will be used; the variation range is found using GetMaxValues command. Thresholds: ---------- A struct containing the thresholds to search for AccX and YawAcc. Have in mind that these thresholds are for normalized AccX, YawAcc (or Wdot). Thresholds.AccX = 0.001; Thresholds.YawAcc = 0.0001; Command output: OpPoint ---------- A cell array of the following structure: OpPoint = 'TotalInpVect' [4x650 double] 'TotalInpVectIndices' [4x650 double] 'TotalOutpVect' [2x650 double] 'CriticalNorm', [1x? double] 'SecondaryNorm', [1x? double] 'ApproxPoint' [1x1 struct] 'ClosestPoints' [1x1 struct] 'CommentText' [?x1 cell array] The first column is a set of strings describing the content of the second column. Specifically: 'TotalInpVect' is the full vector used as test input to the system. 'TotalInpVectIndices' is the array containing the indexes of samples as column vectors. To fixed values the matrix contains NaN, to show that the variable at this index is not varying.

		'TotalOutpVect' is the vector of normalized outputs (responses) to the respective TotalInpVect vector. 'CriticalNorm' is the norm of the vector formed by normalized [AccX, Wdot] to the respective TotalInpVect vector. 'SecondaryNorm' is the norm of the vector formed by normalized [Thrust, Steer] to the respective TotalInpVect vector. 'ApproxPoint' is a structure containing fields: 'Vect' --> Is the input vector that led to approved points using thresholds as creterion. 'Approx' --> Is the nnet output of the respective input vector found at the same index at Vect field. 'Indices' --> The i,j indexes of the vectors created based on the requested variance. 'Norms' --> The norm (distance) of the Approx column normalized vector from [AccX = 0; Wdot = 0] point. 'ClosestPoints' is a struct containing fields: 'Norm' --> The norm (distance) of the closest point from [AccX = 0; Wdot = 0] point. 'Indices' --> the indexes [i;j] of the i-th, j-th value of varying vectors. 'Vect' --> The input vector that results to the closest point. 'VectIndex' --> The index number of the column of ApproxPoint.Vect containing the closest column vector(s). 'NNetsOut' --> The normalized output of neural networks at the closest points. 'VectDenorm' --> The de-normalized vector(s) of the input vector(s) that result to closest point. 'NNetsOutDenorm' --> The denormalized output of neural networks at the closest point. 'CommentText' is a cell array containing comments for all forcefully accepted points that failed equilibrium criteria.
StatLin/ CreateLinearizedModels.m	389	Created by: Lefteris Loghis 2/1/2022 Version 2 Changes from version 1: 1. Added de-normalization factors at 'SpeedX' and 'YawRate' before calling FindContTimeModelOperatingPoint, because FindContTimeModelOperatingPoint was re-normalizing vectors, resulting in very poor variance of speed and yaw rate. 22/8/20 Version 1.

		This command will output for every pair formed by the two input vectors of SpeedValues and WValues the values required for Thrust and Steer to put the system in an equilibrium. LinearizedModelsCellArray = CreateLinearizedModels(AccXNNFilename, WdotNNFilename, SpeedValues, WValues, Thresholds, SetDescription, Resolution, OutputFolderName, OutputFolderPath) ========== Input Variables ---------- AccXNNFilename ---------- The path and filename to the neural network for acceleration along X-axis. WdotNNFilename ---------- The path and filename to the neural network for angular acceleration. SpeedValues ---------- A structure with two fields that contains all the values to be used for speed. The structure contains two fields: Vector: This field contains the vector with the values to be used. Type: This field contains either 'Denormalized' or 'Normalized' to enable this code to understand if if the vector will require normalization or not in order to be used (neural networks have been trained with normalized inputs). WValues ---------- A structure with two fields that contains all the values to be used for yaw rate. The structure contains two fields: Vector: This field contains the vector with the values to be used. Type: This field contains either 'Denormalized' or 'Normalized' to enable this code to understand if if the vector will require normalization or not in order to be used (neural networks have been trained with normalized inputs).

		Thresholds --- A structured variable with two fields that hold the threshold below which the output of the newral network is considered to be close enough to zero, in order to decide on an equilibrium point. Fields: AccX: Threshold for Acceleration Neural Network output. YawAcc: Threshold for Angular Acceleration Neural Network output. SetDescription --- This field should contain a text that will be added to ReadMe output file. The purpose is to describe why the run of this command was issued. Resolution --- Structured variable that contains the step of Thrust and Steer to search the equilibrium. Fields: Thrust Steer OutputFolderName --- String variable containing the name of the folder that will be used to output the model sets. At the end of the name the time and date will be added automatically. OutputFolderPath --- String variable containing the path that the output folder will be saved in. ========== ========== Output Variable --- LinearizedModelsCellArray --- Cell array with each element being a linearized system at the vector contained. This variable is automatically saved at the output folder location. Variable struct fields:

```
   LinearizedModelsCellArray{i,j}.Jacobians.A: Table A of the model.
   LinearizedModelsCellArray{i,j}.Jacobians.B: Table B of the model.
   LinearizedModelsCellArray{i,j}.Vector: The vector for which linearization is achieved.

=============================================================================================
==========================
Application Example
---------------------------------------------------------------------------------------------------------

 cd C:\PHD\Matlab_180910\StatLin
 cd C:\Work\PHD\Autopilot_IV\Software\Matlab_180910\NewralNetworksTransferFunction\MatlabCode

 addpath ../
 addpath ../ModelsRevisionAndControl/
 addpath ../NewralNetworksTransferFunction/MatlabCode/

 AccXNNFilename = '../AccXNeuralFolder/AccXNeural_20140414_1051/AccXNeural_20140414_1051.mat';

 WdotNNFilename = '../WdotNeural/WdotNeural_20140414_1058/WdotNeural_20140414_1058.mat';

 SpeedValues.Vector = [0:1:30];
 SpeedValues.Vector = [0:2:4];
 SpeedValues.Type = 'denormalized';

 WValues.Vector = [-50:5:50];
 WValues.Vector = [-5:5:5];
 WValues.Type = 'denormalized';

 Thresholds.AccX = 0.001;
 Thresholds.YawAcc = 0.0001;

 str_1 = 'Create a linearized models matrix with all possible speed values with 1 kmph step\n';
 str_2 = 'and all possible W values with 5 degpsec step \n';
 SetDescription = strcat(str_1, str_2);

 Resolution.Thrust = 1;
 Resolution.Steer = 1;
```

		OutputFolderName = 'ModelSet'; OutputFolderPath = 'C:/PHD/Matlab_180910/StatLin/LinearizedModelSets'; LinearizedModelsCellArray = CreateLinearizedModels(AccXNNFilename, WdotNNFilename, SpeedValues, WValues, Thresholds, SetDescription, Resolution, OutputFolderName, OutputFolderPath) Notes: ------
tst.m	397	Function that converts measurements of Autopilot_IV Version 0.1 Expects a csv file with the following columns separated by spaces: Frame NoTime(ms) Thrust Steer AccX(g) AccY(g) AccZ(g) Roll(Deg/sec) Pitch(Deg/sec) Roll(Deg/sec) CompX CompY CompZ
DeNormalizeSteerAroundOffset.m	92	Created by: Lefteris Loghis 2/1/22 Version 1. This function performs the inverse action of NormalizeSteerAroundZero, by de-normalizing a vector around a selected offset. This offset must be 121 for the cases we have in the PHD, but changing the center value is also supported. OutVector = DeNormalizeSteerAroundOffset(SteerVector, OffsetOfMidPoint) This function denormalizes steer vector around OffsetOfMidPoint (defeult = 121). This means that it adds any required DC offset so as to have OffsetOfMidPoint as center value. The maximum theoretical span of steer is 0 to 255, but in practice I see a -91 to +89 decimal deviation from the middle point. The usual middle point is 121. This will be added to all measurements unless otherwise selected by OffsetOfMidPoint variable. After adding DC offset, the vector will be divided by 128, in order to be between 0 and 255. Inputs: -------

		SteerVector: The steer vector. OffsetOfMidPoint (optional): If the middle point value of the input vector does not match its median, then use this value instead. This number must be between 0 and 255. Most possibly it will be something close to 121. Output: ------- OutVector: The denormalized output vector. If an error occurs then the result is [].
NormalizeSteerAroundZero.m		Created by: Lefteris Loghis 18/8/19 Version 2. Differences from Version 1 (12/2/14): 1. Enabled the use of a signle number. OutVector = NormalizeSteerAroundZero(SteerVector, MiddlePointValue) This function normalizes steer vector around zero. This means that it removes any DC offset so as to have zero as center value. The maximum theoretical span of steer is 0 to 255, but in practice I see a -91 to +89 decimal deviation from the middle point. The usual middle point is 121. This will be removed from all measurements unless otherwise selected by MiddlePointValue variable. After removing DC offset, the vector will be divided by 128, in order to be between -1 and 1, thus making it suitable for neural network training. Inputs: ------- SteerVector: The steer vector. MiddlePointValue (optional): If the middle point value of the input vector does not match its median, then use this value instead. This number must be between 0 and 255. Most possibly it will be something close to 121. Output:

		------- OutVector: The normalized output vector. If an error occurs then the result is [].
lef_isposint.m		Finds if the number is positive integer.
ExportMeasToCsv.m		Created by: Lefteris Loghis 13/11/13 Version 1. ExportMeasToCsv(Filename, MeasurementCellArray) This function saves a measurement cell array as of GetMeasurement output type to file 'CorrectedMeasurements\Filename'. Inputs: ------- Filename: The filename of the output file (put in CorrectedMeasurements directory. MeasurementCellArray: cell array as of GetMeasurement type.
ModelsRevisionAndControl/ simlink.mdl		
ModelsRevisionAndControl/ TestCTM.m		Created by: Lefteris Loghis 25/1/20 Version 1. This code is a test-bed for the "continuous time model" of vessel modelling. OutVectors = TestCTM ... (StraightForwardMethodActive, ... ThrustVectorDecimated, ... SteerVectorDecimated, ... AccXDCCanceller, ... WdotDCCanceller, ... AccXIntegrLeakage, ... WdotIntegrLeakage, ... DisplayFigures, ... DecimationFactor, ... AccX_net, ...

```
    WdotNet, ...
    DisableSignCheckAtWLeakage, ...
    SpeedXindex, ...
    YawRateIndex, ...
    LimitsArray, ...
    OptVector);

Inputs:
-------
AccXNNFilename: Filename of the saved environment where the AccX neural net exists.
WdotNNFilename: Filename of the saved environment where the Wdot neural net exists.
ExperimentFilename: The filename of the experiment by which the neural net
will be tested.
DecimationFactor: Define the decimation factor by which all input signals
were decimated before training the nets.
SignalsOffsetStruct: The same as in Display measurement.
VerificationSettings: this variable is a structure containing the following fields and subfields:
  Method:
    Name:
      'StraightForward': Evaluation of the model as is with no other changes OR
      'ModifyFeedback': Modification of model feedback by inserting feedback from the measurement file. Two
subfields:
    AdditionalParams: For ModifyFeedback method
      'SamplesPure'
      'SamplesInserted'
  AddDcCancellers: Whatever the method, a DC canceller can be added. Default is addition to both accellerations.
    AccXDCCanceller:
      Enabled: Possible values 'Yes' or 'No'.
      Gain: gain of the output value (default 1).
    WdotDCCanceller: Possible values 'Yes' or 'No'.
      Enabled: Possible values 'Yes' or 'No'.
      Gain: gain of the output value (default 1).
  AccXIntegrLeakage: if enabled, leakage per integrator sample = sgn(sample)*(Leakage + abs(sample)/R).
      Enabled: Possible values 'Yes' or 'No'.
      Leakage: Constant leakage value of integrator of AccX.
      R: Analogous to resistor leakage value of integrator of AccX.
  WdotIntegrLeakage: if enabled, leakage per integrator sample = sgn(sample)*(Leakage + abs(sample)/R).
```

		Enabled: Possible values 'Yes' or 'No'. Leakage: Constant leakage value of integrator of Wdot. R: Analogous to resistor leakage value of integrator of Wdot. OptSettings OptSettings.DiffStep --> The arithmetic step used to produce differentiation. OptSettings.DistStep --> The arithmetic step used to search for the best distance to -Ve direction. OptSettings.MaxRepsAllowed --> Maximum number of reps allowed for the optimization. OptVector.MaxAlwdAnadeltaSteps --> MaximumNumberOfSteps to the current direction of -Ve. OptSettings.MinAllowedProgress --> Minimum allowed error progress for each step. OptSettings.MaxStepDivisions = 10; OptSettings.StepDivider = 10; Output: ------- Graphical representations presenting the difference between neural net outputs and real measured values.
ModelsRevisionAndControl/ CTMcore.m		Simulate nets by interconnecting them. Each net takes as inputs only steer and thrust. The rest of the inputs (V and W) are produced by the networks themselves after their output integration. The integrated outputs are fed back to the networks.
ModelsRevisionAndControl/ ModelOptimizer.m	1106	Created by: Lefteris Loghis 18/2/20 Version 2. Changes from version 1 (3/1/20): 1. Used CTMcore (continuous Time Model core) for simulation of the system. 2. Changed output directory to ../../ModelsRevisionAndControl/ from NewralNetworksTransferFunction. This code is based on VerifyDiffEqNeuralModel_2019ed and tries to optimize the error from ideal response by varying the following parameters: 1. a parameter of AccX DC canceller. 2. a parameter of Wdot DC canceller. 3. Leakage value of AccX integrator. 4. Resistive leakage value of AccX integrator. 5. Leakage value of Wdot integrator. 6. Resistive leakage value of Wdot integrator.

		ModelOptimizer (AccXNNFilename, WdotNNFilename, ExperimentFilename, DecimationFactor, SignalsOffsetStruct, SpecialCommands, VerificationSettings) Inputs: ------- AccXNNFilename: Filename of the saved environment where the AccX neural net exists. WdotNNFilename: Filename of the saved environment where the Wdot neural net exists. ExperimentFilename: The filename of the experiment by which the neural net will be tested. DecimationFactor: Define the decimation factor by which all input signals were decimated before training the nets. SignalsOffsetStruct: The same as in Display measurement. VerificationSettings: this variable is a structure containing the following fields and subfields: Method: Name: 'StraightForward': Evaluation of the model as is with no other changes OR 'ModifyFeedback': Modification of model feedback by inserting feedback from the measurement file. Two subfields: AdditionalParams: For ModifyFeedback method 'SamplesPure' 'SamplesInserted' AddDcCancellers: Whatever the method, a DC canceller can be added. Default is addition to both accellerations. AccXDCCanceller: Enabled: Possible values 'Yes' or 'No'. Gain: gain of the output value (default 1). WdotDCCanceller: Possible values 'Yes' or 'No'. Enabled: Possible values 'Yes' or 'No'. Gain: gain of the output value (default 1). AccXIntegrLeakage: if enabled, leakage per integrator sample = sgn(sample)*(Leakage + abs(sample)/R). Enabled: Possible values 'Yes' or 'No'. Leakage: Constant leakage value of integrator of AccX. R: Analogous to resistor leakage value of integrator of AccX. WdotIntegrLeakage: if enabled, leakage per integrator sample = sgn(sample)*(Leakage + abs(sample)/R). Enabled: Possible values 'Yes' or 'No'. Leakage: Constant leakage value of integrator of Wdot. R: Analogous to resistor leakage value of integrator of Wdot.

		OptSettings OptSettings.DiffStep --> The arithmetic step used to produce differentiation. OptSettings.DistStep --> The arithmetic step used to search for the best distance to -Ve direction. OptSettings.MaxRepsAllowed --> Maximum number of reps allowed for the optimization. OptSettings.MaxAlwdAnadeltaSteps --> MaximumNumberOfSteps to the current direction of -Ve. OptSettings.MinAllowedProgress --> Minimum allowed error progress for each step. OptSettings.MaxStepDivisions = 10; OptSettings.StepDivider = 10; Output: ------- Graphical representations presenting the difference between neural net outputs and real measured values. Notes: ------ This code works as is, but should be changed to have a clear state machine that does all the work for the optimization, chooses how much to the anadelta direction etc..
ModelsRevisionAndControl/ CommandsUsedForModelOptimizer.matlab	383	Matlab file that is used to remind the use of CommandsUsedForModelOptimizer command.
ModelsRevisionAndControl/ RealTimeDCCanceller.m	46	Created by: Lefteris Loghis 11/11/19 Version 1. This function removes the DC signal in real time. It does that using the difference equation y(n) = x(n) - x(n-1) + ay(n-1) see https://www.embedded.com/dsp-tricks-dc-removal/ Yn = RealTimeDCCanceller (Xn, Xn_m_1, Yn_m_1, a) Inputs: ------- Xn: current X sample. Xn_m_1: sample x(n-1) Yn_m_1: sample y(n-1) a: gain parameter. The smallest this parameter is, the fastest the DC canellation is. The closer to 1

		the slower DC cancellation is performed, with a=1 meaning no DC cancellation. Output: ------- Yn: the current output sample with the DC removed.
ModelsRevisionAndControl/ CommandsUsedForSupCtrlSelfTest.matlab	57	Matlab file that is used to remind the use of SupCtrlSelfTest command.
ModelsRevisionAndControl/ CmdsMdlOpt.m	47	Matlab file that is used to remind the use of ModelOptimizer.m command.
ModelsRevisionAndControl/ VerifyDiffEqNeuralModel_2019ed.m	556	Created by: Lefteris Loghis 2/11/19 Version 1. Changes from version 1: Added SpecialCommands usage. This function tests the accuracy of the vessel model produced by the neural networks that model X-axis acceleration and angular acceleration. It is based on VerifyDiffEqNeuralModel command created at 2014 (v2). VerifyDiffEqNeuralModel (AccXNNFilename, WdotNNFilename, ExperimentFilename, DecimationFactor, SignalsOffsetStruct) Inputs: ------- AccXNNFilename: Filename of the saved environment where the AccX neural net exists. WdotNNFilename: Filename of the saved environment where the Wdot neural net exists. ExperimentFilename: The filename of the experiment by which the neural net will be tested. DecimationFactor: Define the decimation factor by which all input signals were decimated before training the nets. SignalsOffsetStruct: The same as in Display measurement. VerificationSettings: this variable is a structure containing the following fields and subfields: Method: Name: 'StraightForward': Evaluation of the model as is with no other changes OR

		'ModifyFeedback': Modification of model feedback by inserting feedback from the measurement file. Two subfields: AdditionalParams: For ModifyFeedback method 'SamplesPure' 'SamplesInserted' AddDcCancellers: Whatever the method, a DC canceller can be added. Default is addition to both accellerations. AccXDCCanceller: Enabled: Possible values 'Yes' or 'No'. Gain: gain of the output value (default 1). WdotDCCanceller: Possible values 'Yes' or 'No'. Enabled: Possible values 'Yes' or 'No'. Gain: gain of the output value (default 1). AccXIntegrLeakage: if enabled, leakage per integrator sample = sgn(sample)*(Leakage + abs(sample)/R). Enabled: Possible values 'Yes' or 'No'. Leakage: Constant leakage value of integrator of AccX. R: Analogous to resistor leakage value of integrator of AccX. WdotIntegrLeakage: if enabled, leakage per integrator sample = sgn(sample)*(Leakage + abs(sample)/R). Enabled: Possible values 'Yes' or 'No'. Leakage: Constant leakage value of integrator of Wdot. R: Analogous to resistor leakage value of integrator of Wdot. Output: ------- Graphical representations presenting the difference between neural net outputs and real measured values.
ModelsRevisionAndControl/ FindBestMach.m	187	Created by: Lefteris Loghis 7/12/19 Version 1. This function finds the best mach, regarding absolute error sum, that results from varying DC blocking parameters and leakage at integrators of the continuous time model. BestErrParams = FindBestMach(DCCancellerConf, LeakageConf, R_LeakageConf) Inputs: -------

		DCCancellerConf: LeakageConf: R_LeakageConf: Output: ------- BestErrParams: Cell array with the following content: BestErrParams{1}: Results for SpeedX with the format of lef_min output. BestErrParams{2}: Results for W with the format of lef_min output. BestErrParams{3}: CasesCellArray: all vectors tested. BestErrParams{4}: Errors of SpeedX with each of the test vectors at CasesCellArray. BestErrParams{5}: Errors of W with each of the test vectors at CasesCellArray.
ModelsRevisionAndControl/ CommandsUsed.matlab	193	Matlab file that is used to remind the use of VerifyDiffEqNeuralModel_2019ed command.
ModelsRevisionAndControl/ FindDirLocation.m	113	
ModelsRevisionAndControl/ SupervisoryControllerSelfTest.m	122	File self testing the first version of Supervisory Controller.
ModelsRevisionAndControl/ SupervisoryController.m	78	First implementation of Supervisory Controller.
ModelsRevisionAndControl/ CommandsUsedForTestCTM.matlab		
ModelsRevisionAndControl/ FindBestMach.m.backup_191214		
ModelsRevisionAndControl/ MBCController.m ModelsRevisionAndControl/ MBCController.matlab		
ModelsRevisionAndControl/		

MBCControllerTestBed.m		
ModelsRevisionAndControl/ MBCControllerTestBed.matlab		
ModelsRevisionAndControl/ SupervisoryController.m		
ModelsRevisionAndControl/ SupervisoryControllerSelfTest.m		
ModelsRevisionAndControl/ SupervisoryController2.m		
ModelsRevisionAndControl/ SupervisoryController2SelfTest.m		
ModelsRevisionAndControl/ FullSystemTestBed.m		
ModelsRevisionAndControl/ FullSystemTestBed.matlab		
AccXNeural.m		

MATLAB Files and Size in Lines	
'AccXNeural.m'	427
'AccX_net_TF.m'	42
'AddForcesTableRow.m'	37
'AnalyzeLinModelsSet.m'	213
'AnalyzeStabilityAndVectLims.m'	356
'ang_tst.m'	10
'ApplyPhysicalLaw.m'	157
'ApplyVectorMax.m'	49
'CalculateMomentOfInertia.m'	64
'CheckAcc.m'	667
'CheckIfVector.m'	24
'CompareSim2EqOutAccXNet.m'	226

'CompareSim2EqOutWdotNet.m'	241
'CompJacobian.m'	196
'CompleteForcesTable.m'	21
'CreateDiffceVectorsForSimlnk.m'	158
'CreateLinearizedModels.m'	390
'CreateMBCVectDiagram.m'	58
'CreatePHDObjects.m'	458
'CreatePHDObjectsMission.m'	127
'CreateSpdXVectors.m'	215
'CreateWTrainVectors.m'	385
'CreateWVectors.m'	307
'CreateZeroFocesTable.m'	15
'CTM.m'	293
'CTMcore.m'	90
'decmt.m'	92
'DecmtTimeVector.m'	68
'decmt_v1.m'	42
'DeNormalizeSteerAroundOffset.m'	92
'DisplayMeasurement.m'	553
'DispMeas.m'	426
'EquationOfAccXNet.m'	9
'EquationOfWdotNet.m'	9
'EuclideanLength.m'	20
'ExpandVector.m'	21
'ExportMeasToCsv.m'	53
'ExtractMBCLUTs.m'	353
'ExtractMBCOpPoint.m'	385
'ExtractValueFromParam.m'	40
'FindBestMach.m'	186
'FindCenterOfMassForPHDobj.m'	42
'FindContTimeModelOperatingPoint.m'	701
'FindDirLocation.m'	112
'FindMaxOfVectors.m'	221

'FindNetName.m'	20
'FindSpeedVect.m'	155
'FixMeasurement.m'	185
'FixVector.m'	36
'FormatOpenFigures.m'	41
'FormatOpenFigures_v2.m'	191
'FullSystemTestBed.m'	953
'FullSystemTestBed2.m'	1174
'GetMaxValues.m'	48
'GetMeasurement.m'	58
'GetObjectValue.m'	52
'GetTimeSuffix.m'	19
'GetVector.m'	56
'GetVectorIndex.m'	48
'iftest.m'	8
'isint.m'	28
'lef_isarray.m'	37
'lef_isnumber.m'	13
'lef_isnumberVector.m'	18
'lef_isposint.m'	12
'lef_isrealarray.m'	37
'lef_isrealarrayorvector.m'	37
'lef_max.m'	71
'lef_min.m'	111
'LocateNeighbouringPoints.m'	124
'LocatePHDobject.m'	107
'LookUpObjRow.m'	31
'mavg.m'	77
'mavg_v1.m'	36
'MBCController.m'	401
'MBCController2.m'	488
'MBCController2TestBed.m'	1114
'MBCControllerTestBed.m'	862

'MBCErrEst.m'	360
'MiniSwarmPosTest.m'	55
'MiniSwarmTestBed.m'	1267
'ModelOptimizer.m'	1105
'MovePHDobject.m'	67
'MultiVectEuclideanLength.m'	46
'NNet_TF_Equ.m'	68
'NNet_TF_Equ_v1.m'	67
'NormalizeDirection.m'	96
'NormalizeSteer.m'	86
'NormalizeSteerAroundZero.m'	79
'NormalizeThrust.m'	50
'NormalizeYawAngle.m'	51
'NormalizeYawRate.m'	45
'NormAngle.m'	37
'NumericalJacobian.m'	227
'PDTestBedCL.m'	79
'PDTestBedCL2.m'	82
'PIDTestBed.m'	260
'PIDTestBedCL3.m'	491
'PIDTestBedCL4.m'	416
'PID_pform.m'	52
'PID_pform2.m'	49
'PID_sform.m'	52
'PID_vform.m'	44
'PID_vform2.m'	49
'PkFreqVsMaxGain.m'	26
'PlotFigures.m'	178
'ProduceMovieOut.m'	105
'ProjectVectorToRotatedAxis.m'	35
'RandomExperimentsBuilder.m'	424
'RealTimeDCCanceller.m'	45
'RemoveAngleRateOffset.m'	93

'RemoveYawRateOffset.m'	93
'RetrainNet.m'	554
'RotatePHDobject.m'	67
'RotateVector.m'	32
'ShowLngth.m'	9
'SimNet.m'	264
'SimulateMissionSpace.m'	739
'SimulateNet.m'	495
'SimulateNet_v3.m'	450
'SimulinkDiffceNeural.m'	34
'SimulinkDiffceNeuralSpeed.m'	44
'SimulinkTest1_Func.m'	3
'SpeedXNeural.m'	372
'StepAndBodeDiagrams.m'	133
'StoreAngles.m'	80
'str.m'	12
'SupCtrlPIDSelfTest.m'	171
'SupervCtrlr3SlfTstv2.m'	162
'SupervCtrlr3SlfTstv3.m'	227
'SupervisoryController.m'	78
'SupervisoryController2.m'	187
'SupervisoryController2SelfTest.m'	148
'SupervisoryController3.m'	166
'SupervisoryController3SelfTest.m'	148
'SupervisoryController3SelfTestv2.m'	159
'SupervisoryControllerPID.m'	223
'SupervisoryControllerPIDSelfTest.m'	165
'SupervisoryControllerSelfTest.m'	122
'test.m'	38
'TestCTM.m'	414
'TestsWithAccXNet.m'	414
'TestsWithWdotNet.m'	344
'ThrustSpeed.m'	275

'TrainNNetForW.m'	559
'TrainNNetForWdot.m'	614
'TrainNNetForW_v3.m'	355
'tst.m'	398
'TstDifferenceEqNeuralModel - Copy.m'	215
'TstDifferenceEqNeuralModel.m'	212
'TstDifferentialEqNeuralModel.m'	211
'tst_1.m'	426
'tst_text.m'	2
'Verde_2_2.m'	426
'VerifyDiffEqNeuralModel.m'	306
'VerifyDiffEqNeuralModelNNEq.m'	341
'VerifyDiffEqNeuralModel_2019ed.m'	555
'VerifyDifferenceEqNeuralModel.m'	370
'VerifyDifferenceEqNeuralModel_v1.m'	371
'WdotNet_TF.m'	32
TOTAL MATLAB LINES	**33700**

13 APPENDIX 1: Foundations of Future Work

13.1Introduction

Despite the advances in autonomous watercraft, both underwater and surface, a robust methodology for distributed control and self-organization in swarms is lacking. In the present work we develop a simulator that allows for investigations of swarm macroscopic dynamics when the interaction forces between the constituent units of the swarm abide with different mathematical laws and governing equations [37]. Every watercraft unit in the swarm is represented as a massless rigid beam of known length connecting two point masses with, in general, different value at the ends. This defines the mass and the moment of inertia of each vehicle in the swarm. The masses of one vehicle need not generally interact with those of other vehicles gravitationally. In contrast the bow and stern masses can be taken to be "charged", with identical or oppositely signed charges, resulting in attractive or repulsive forces obeying some form of easily programmable, mathematical, law. Additionally, the charged masses at the ends of each vehicle will also be subject to externally applied force fields. Each vehicle will as a result move on a path that is not preprogrammed, but is derived dynamically on the basis of the unit's interactions with other units in the swarms together with the effect of any exogenous fields. The macroscopic dynamics of swarms of vehicles complying with programmable artificial physics of the sort introduced above need to be analyzed in order to be able, at a subsequent stage, to reverse engineer the observed dynamics and design governing laws for the interactions so that a desired pattern of swarm behavior or self-organization emerges on the holistic level. A simulator will be used to support analytical findings based on the theory of dissipative structures and self-organizing systems applied to distributed control and automated dynamic planning of surface watercraft systems. A critical detail is the fact that such vehicles are restricted to moving on a plane in the simplest of case assumptions. Simulations using the tool of this work will allow further investigations, enabling the development of a formal synthesis technique of dynamic interaction governing laws meeting arbitrary macroscopic self-organization requirements.

One of the problems of guidance of a swarm of vessels is who creates the guidance commands: who is in charge [1], [2]. For these algorithms we would like the answer to be no one! At least when the vessels operate, the intervention of a human, although possible, is not mandatory. The main idea is that before operation, the "mission space" is described with the form of an area influenced by certain fields. The nature of these fields may either be a form of known natural field (e.g. gravitational or electrostatic), or more a more arbitrary choice that is simply suitable for the application.

This section is the result of experimentation on algorithms for self-organization of autonomous vessels [7], [38], [39]. The driving idea behind these experiments was to find an algorithm that requires no master and creates guidance vectors with just an initial description of the mission space. The paragraphs that follow describe the first steps towards the implementation of this algorithm, present the simulation environment used and examine some applications.

13.2 SIMULATOR ENGINE IMPLEMENTATION

To work on the idea, the first field to be examined is electrostatic. This type of field is very convenient because both repulsive and attractive forces can be created. Although the idea has been studied to support collision avoidance, the enhancement will not be examined in this paper.

Our test bed for the algorithm is a Matlab based simulator we have created. For every case examined, the first step is to describe the vessel(s) more formally. Every vessel is understood as a rectangle with three separate regions, as shown in (Fig. 1). The required parameters to describe a vessel are listed in (Table 20).

Table 20: Parameters for vessel description.

Parameter	Description
w	Width of vessel.
l_A	Length of region A.
l_{AB}	Length of region AB.
l_B	Length of region B.
m_A	Mass of region A.
m_{AB}	Mass of region AB.
m_B	Mass of region B.
Q_A	Region A point charge magnitude.
Q_B	Region B point charge magnitude.

From the parameters listed in (Table 20), the simulator calculates the following:

Table 21: Parameters automatically calculated by simulator.

Parameter	Description
A_x	Point A x coordinate.
A_y	Point A y coordinate.
B_x	Point B x coordinate.
B_y	Point B y coordinate.
C_x	Point C x coordinate.
C_y	Point C y coordinate.
D_x	Point D x coordinate.
D_y	Point D y coordinate.
QAx	Point charge Q_A x location coordinate.
QAy	Point Q_A y coordinate.
QBx	Point Q_B x coordinate.
QBy	Point Q_B y coordinate.
I	Moment of inertia.
G_x	Center of mass x coordinate.
G_y	Center of mass y coordinate.
$\overrightarrow{GA}_x$	Vector from G to A. x-coordinate.
$\overrightarrow{GA}_y$	Vector from G to A. x-coordinate.
$\overrightarrow{GB}_x$	Vector from G to B. x-coordinate.
$\overrightarrow{GB}_y$	Vector from G to B. x-coordinate.
$\overrightarrow{GC}_x$	Vector from G to C. x-coordinate.
$\overrightarrow{GC}_y$	Vector from G to C. x-coordinate.
$\overrightarrow{GD}_x$	Vector from G to D. x-coordinate.
$\overrightarrow{GD}_y$	Vector from G to D. x-coordinate.
θ_{0A}	Angle of GA vector (refer to (Fig. 2)).
θ_{0B}	Angle of GB vector (refer to (Fig. 2)).
θ_{0C}	Angle of GC vector (refer to (Fig. 2)).
θ_{0D}	Angle of GD vector (refer to (Fig. 2)).
$\lVert\overrightarrow{GA}\rVert$	Length of GA vector (refer to (Fig. 2)).
$\lVert\overrightarrow{GB}\rVert$	Length of GB vector (refer to (Fig. 2)).
$\lVert\overrightarrow{GC}\rVert$	Length of GC vector (refer to (Fig. 2)).
$\lVert\overrightarrow{GD}\rVert$	Length of GD vector (refer to (Fig. 2)).
$\overrightarrow{GQ_A}_x$	Vector from G to charge QA. x-coordinate.
$\overrightarrow{GQ_A}_y$	Vector from G to charge QA. y-coordinate.
$\overrightarrow{GQ_B}_x$	Vector from G to charge QB. x-coordinate.
$\overrightarrow{GQ_B}_y$	Vector from G to charge QB. y-coordinate.

To clarify (Fig. 1): The vessel is separated into three regions, each potentially having different masses. This helps in approximating the different loads that can exist on a vessel and gives flexibility in providing arbitrary X-axis offsets to the center of mass from the rectangle's center. We consider the rectangle to be symmetric about the Y-axis, with G (the center of mass) always lying on the X-axis. Point charges are taken to exist at the ends of each rectangle in the middle of the Y-axis.

To start a simulation, for each vessel the following initial conditions must be inserted:

Table 22: Initial motion parameters needed for simulation.

Parameter	Description
G_x or $\vec{r}_x$	Center of mass x coordinate.
G_y or $\vec{r}_y$	Center of mass y coordinate.
$\vec{u}_x$	Velocity vector x element.
$\vec{u}_y$	Velocity vector y element.
$\vec{a}_x$	Acceleration vector x element.
$\vec{a}_y$	Acceleration vector y element.
θ	Angle. Positive counter-clockwise.
ω	Angular velocity. Positive counter-clockwise.
$\dot{\omega}$	Angular acceleration. Positive counter-clockwise.

Table 23: Calculated variables for every simulation step.

Parameter	Description
G_x or $\vec{r}_x$	x-coordinate of center of mass or vector from point O(0,0) of mission space to the center of mass of the object under investigation.
G_y or $\vec{r}_y$	y-coordinate of center of mass or vector from point O(0,0) of mission space to the center of mass of the object under investigation.
$\vec{u}_x$	Velocity vector x element.
$\vec{u}_y$	Velocity vector y element.
$\vec{a}_x$	Acceleration vector x element.
$\vec{a}_y$	Acceleration vector y element.
θ	Angle. Positive counter-clockwise.
ω	Angular velocity. Positive counter-clockwise.
$\dot{\omega}$	Angular acceleration. Positive counter-clockwise.
A_x	Point A x coordinate.
A_y	Point A y coordinate.
B_x	Point B x coordinate.
B_y	Point B y coordinate.
C_x	Point C x coordinate.
C_y	Point C y coordinate.
D_x	Point D x coordinate.
D_y	Point D y coordinate.
QAx	Point charge Q_A x location coordinate.
QAy	Point Q_A y coordinate.
QBx	Point Q_B x coordinate.
QBy	Point Q_B y coordinate.

A critical parameter for the simulation (the most critical, in a sense) is the time-step T_s of the simulation. The time-step is the time interval that will pass before all vectors and forces are recalculated and, as will be shown later, it has a dramatic impact on the algorithm results. For a real world simulation case, T_s can encounter information broadcasting delays, calculation processing time issues and time safety margin effects.

Once started, the simulator engine calculates the parameters listed in the table above (Table 23) for every time-step.

There are two particular ways (at least in a simplified approach) to guide the vessels in mission space:

1. By inserting still point charges at selected points of space.
2. By applying a constant field that exists over the whole space.

Thus if all vessels are aware of their space and fellow vessels' positions and basic parameters, they can create their own heading and velocities based on an analysis of the current mission space instant. The analysis for the case examined requires calculation of all forces applied to every vessel, based on the application of electrostatic laws for the charges and field declared. This leads to issues concerning centralized or decentralized broadcasting of necessary parameters, but for the present paper this is considered a different topic of discussion.

The mathematical laws used by the simulation engine (or the software running at vessels for the real-world applications) to calculate all the applied forces are basic,

$$\vec{F} = \vec{E} \cdot Q$$

for the calculation of the force at a charge induced by a field $\vec{E}$, and

$$\vec{F} = K\frac{Q_i Q_j}{\|\vec{r}\|^2} \cdot \left(-\frac{\vec{r}}{\|\vec{r}\|}\right) = -K\frac{Q_i Q_j}{\|\vec{r}\|^3}\vec{r}, \quad i \neq j,$$

for the calculation of the forces between all distinct pairs of charges in the mission space.

Although K may be any constant considered appropriate, the one currently used is the electrostatic constant, i.e.

$$K = \frac{1}{4\pi\varepsilon_0} = 8.9875 \times 10^9 \frac{Nm^2}{C^2}$$

where

$$\varepsilon_0 = 8.8541 \times 10^{-12}\frac{F}{m}$$

For every vessel, the applied forces to its charges are calculated. As a second step its direction, velocity, acceleration, angle, rate of turn, angular acceleration are found through two basic laws of mechanics [21]:

$$\sum \vec{F} = m\vec{a} \quad (3)$$

$$\sum \vec{M} = I\dot{\vec{\omega}} \quad (4)$$

For the current implementation we have intentionally not included energy dissipation factors such as friction or damping, in order to examine the stability of the simulator outputs. Dissipation may "disguise" instabilities as a side-effect of energy loss.

TEST CASES

The following paragraphs present some case studies of the simulating engine usage.

Test Case 1: One vessel with one static charge

In this case a static charge was placed at point (0,0) of the mission space and the center of mass of the object was placed at distance of 8m from the static charge at the direction of X-axis. A sketch of the test case is shown in (Fig. 3)

The parameters of the test case are listed at (Table 24).

Table 24: Parameters of Test Case 1.

Test case 1 parameters				
Static Parameters		Other Parameters	Motion Parameters	
Parameter	Value		Parameter	Value
w	2 m	$T_s = 5\ ms$ N = 5000 Samples	G_x or $\vec{r}_x$	8 m
l_A	2 m		G_y or $\vec{r}_y$	0 m
l_{AB}	6 m		$\vec{u}_x$	0 m/s
l_B	2 m		$\vec{u}_y$	0 m/s
m_A	2 Kg		$\vec{a}_x$	0 m/s^2
m_{AB}	6 Kg		$\vec{a}_y$	0 m/s^2
m_B	2 Kg		θ	0 °
Q_A	-10^{-5} C		ω	0 °/s
Q_B	10^{-8} C		$\dot{\omega}$	0 $°/s^2$

Theoretically, with infinitesimally small T_s, the result would be either a high frequency oscillation around (0,0) or the object sticking at the static charge [40], [41]. Since it is almost impossible for a simulation instance to coincide with the moment that the point charge passes exactly above the point charge of the origin, the result was the oscillation as shown in (Fig. 4). The diamond shape of the figure results from the "quiver" Matlab command, which creates an arrow from the center of mass to the direction of the vessel. The wide part means "faster".

Test Case 2: One vessel in space with a field

In this case an object was inserted in a space with its center of gravity at point (8,0). The space was under the influence of a uniform electric field: $\vec{E} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$

(Fig. 6) shows the setup. The parameters for this experiment were:

Test case 2 parameters				
Static Parameters		Other Parameters	Motion Parameters	
Parameter	Value		Parameter	Value
w	2 m	$T_s = 50\ ms$ N = 20000 Samples $\vec{E} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$	G_x or $\vec{r}_x$	8 m
l_A	2 m		G_y or $\vec{r}_y$	0 m
l_{AB}	6 m		$\vec{u}_x$	0 m/s
l_B	2 m		$\vec{u}_y$	0 m/s
m_A	2 Kg		$\vec{a}_x$	0 m/s^2
m_{AB}	6 Kg		$\vec{a}_y$	0 m/s^2
m_B	2 Kg		θ	0 °
Q_A	-0.1 C		ω	0 °/s
Q_B	0.1 C		$\dot{\omega}$	0 $°/s^2$

The result was exactly as expected [40], [41]: the field in combination with charges, created a force couple which did not move the object at all, but rotated it counter-clockwise. The turn rate accelerated to 90° where it started decelerating. The reason for the change was that following 90° the charges "changed sides" and the force couple created a clockwise rate of turn. Since the electric field is conservative, deceleration stopped at 180° after which the opposite phenomenon took place, rotating back to 0°.

The resulting angle is shown in (Fig. 5), where it is clear that the angle varies from 0° to 180°. The angle is shown along with the rate of turn (omega), to verify the change in the forces. Note that since the simulator did not include damping factors for these experiments, the oscillation would continue as the one in (Fig. 5) indefinitely.

Test Case 3: Two Vessels of Different Mass

With the previous test cases we verified the simulator's operation for a single object interacting with a static charge or a field. The next step was to gradually start increasing the complexity of the test cases. For this reason we used two simulated vessels. The scenario to be implemented was to set up the parameters in such way so as to make one object perform circles around the other [40], [41].

Sub-case 1: 2 s simulation time Ts = 10e(-4) s.

The approximation used for the test case vessels is presented in (Fig. 7). Two types of vessels were defined: one "heavy" (vessel 1) with mass 100Kg and one "light" (vessel 2) with mass 1Kg. The objective was to make vessel 2 perform circles of 1 m radius, at a frequency of 2 Hz around vessel 1. To achieve this, the attracting forces between the two must sum up to approximately the centripetal force.

The relevant equations are

$$\vec{r}(t) = r(t)\hat{r}(t) \Rightarrow$$

$$\dot{\vec{r}}(t) = \vec{u}(t) = \dot{r}(t)\hat{r}(t) + r(t)\dot{\hat{r}}(t) \xrightarrow{\dot{r}(t)=0}$$

$$\vec{u}(t) = r(t)\dot{\theta}(t)\hat{\theta}(t) \Rightarrow$$

$$\vec{a}(t) = \dot{\vec{u}}(t) = \dot{r}(t)\dot{\theta}(t)\hat{\theta}(t) + r(t)\ddot{\theta}(t)\hat{\theta}(t) + r(t)\dot{\theta}(t)\dot{\hat{\theta}}(t) \xrightarrow{\dot{r}(t)=0,\ddot{\theta}(t)=0,\dot{\hat{\theta}}(t)=-\hat{r}(t)}$$

$$\vec{a}(t) = -r(t)\dot{\theta}^2(t)\hat{r}(t) = -r(t)\omega^2(t)\hat{r}(t) = -r(2\pi f)^2\hat{r} \quad (5)$$

Equation (5) results from the nature of uniform circular motion, for which: $\dot{r}(t) = 0, \dot{\theta}(t) = 0, \ddot{\theta}(t) = 0$. To create a 2 Hz circulation the initial velocity vector was

$$\vec{u} = \begin{bmatrix} 2 \cdot 2\pi\|\vec{r}\| \\ 0 \end{bmatrix} = \begin{bmatrix} 12.5663 \\ 0 \end{bmatrix} \quad (6)$$

To sustain such a motion the centripetal force must be (using Eq. 5):

$$\vec{F}_2 = m_2\vec{a}(t) = -(1\ Kg) \cdot (1\ m) \cdot (2\pi \cdot 2\ Hz)^2 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (7)$$

where $\vec{F}_2$ is the force to be applied to vessel 2, and m_2 is the total mass of vessel 2. Referring to (Fig. 8), the centripetal force is

$$\vec{F}_2 = \vec{F}_{11} + \vec{F}_{12} + \vec{F}_{21} + \vec{F}_{22} \quad (8)$$

Since the angle between $\vec{F}_{11}$ and $\vec{F}_{12}$ and between $\vec{F}_{21}$ and $\vec{F}_{22}$ is very small, it will be considered as zero. Thus from Eq. 2

$$\vec{F}_2 = -4K\frac{Q^2}{\|1\|^3}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (9)$$

which results in Q being: $Q = 6.6276^{-5}$ C (10)

If the duration of simulation (N) is 20,000 Samples, it means that the total time simulated is 2 s, which means 4 encirclements of vessel 2 tovessel 1. The results for this Ts reveal exactly this, as shown in (Fig. 9). In this figure the courses of both centers of masses are printed. As it can be seen,vessel 1moves a bit by making a slightly wavy, averagely almost straight, course as it is pulled be the forces it applies to vessel 2. Vessel 2 on the other hand follows vessel 1 by encircling it almost 4 times. Something from the full last encirclement is lost due to the movement of vessel 1.

Sub-case 2: 30 s simulation time Ts = 10e(-4) s.

The same case was tested for 30 s with the same Ts (N = 300,000 Samples), to see if any instabilities result. The results are shown in (Fig. 10). Results are smooth and expected.

Sub-case 3: 30 s simulation time with varying Ts. Ts = 10e(-3) s and Ts = 10e(-2) s.

In this sub-case the effect of changing Ts became apparent. As Ts is increased, the time elapsing from one instant to the next increases. That means that more time passes before velocities and forces are re-evaluated and thus the shape of the resulting movement changes drastically.

Table 25: Test case 3 parameters.

Test case 3 parameters					
Static Parameters			Motion Parameters		
Parameter	Vessel 1	Vessel 2	Parameter	Vessel 1	Vessel 2
w	0.01 m	0.02 m	G_x or $\vec{r}_x$	0 m	0 m
l_A	0.01 m	0.01 m	G_y or $\vec{r}_y$	0 m	1 m
l_{AB}	0.01 m	0.01 m	$\vec{u}_x$	0 m/s	12.57 m/s
l_B	0.01 m	0.01 m	$\vec{u}_y$	0 m/s	0 m/s
m_A	50 Kg	0.5 Kg	$\vec{a}_x$	0 m/s²	0 m/s²
m_{AB}	0 Kg	0 Kg	$\vec{a}_y$	0 m/s²	0 m/s²
m_B	50 Kg	0.5 Kg	θ	0 °	0 °
Q_A	- Q C	Q C	ω	0 °/s	0 °/s
Q_B	- Q C	Q C	$\dot{\omega}$	0 °/s²	0 °/s²
$T_s = 10^{-4}\ s$			N = 20,000 Samples		
Q = 6.6276^{-5} C					

Generating a test using the values of (Table 25) and selecting Ts = 10e(-3) and N = 30,000 (30 s simulated time is covered once again), the result reveals the effects of Ts enlargement. As (Fig. 11) shows, the encirclements have become larger in radius and their frequency has decreased compared to the previous sub-case of test case 3.

To verify the process we took one further step and tested with Ts = 10e(-2) and N = 3,000 (30 s simulated time). The results are shown in (Fig. 12). The radius of encirclements increased even more and as a result the encirclements are much less.

As can be seen in every sub-case of test case 3, the radius of encirclements always increases as time passes. If this type of motion implementation is to be used, one more layer can be added to the algorithm, to control the charges of vessel 1 dynamically. As a consequence, the radius of vessel 2 encirclements could be maintained inside required limits, though with the tolerance highly dependent on Ts. Ts itself will be the smallest feasible size, depending on the communication setup and processing resources for a given practical implementation.

Test Case 4: Adding Y-axis velocity.

The final test case is a variation of test case 3, which results from adding some initial velocity along Y-axis. Of interest in this case, was whether the Y axis disturbance would create loss of stability at this open-loop algorithm.

Sub-case 1: 30 s simulation time Ts = 10e(-4) s, 10% Y-offset.

In this sub-case a 0.1ux was added as uy to the vessel 2 velocity.

The results are shown in (Fig. 13). The Y-axis offset created Y axis deviation to vessel 1's course, but no unexpected behavior was found whatsoever for the duration simulated.

Table 26: Test case 4, sub-case 1 parameters.

Test case 4 sub-case 1 parameters					
Static Parameters			Motion Parameters		
Parameter	Vessel 1	Vessel 2	Parameter	Vessel 1	Vessel 2
w	0.01 m	0.02 m	G_x or $\vec{r}_x$	0 m	0 m
l_A	0.01 m	0.01 m	G_y or $\vec{r}_y$	0 m	1 m
l_{AB}	0.01 m	0.01 m	$\vec{u}_x$	0 m/s	12.57 m/s
l_B	0.01 m	0.01 m	$\vec{u}_y$	0 m/s	1.257 m/s
m_A	50 Kg	0.5 Kg	$\vec{a}_x$	0 m/s²	0 m/s²
m_{AB}	0 Kg	0 Kg	$\vec{a}_y$	0 m/s²	0 m/s²

m_B	50 Kg	0.5 Kg	θ	0 °	0 °
Q_A	- Q C	Q C	ω	0 °/s	0 °/s
Q_B	- Q C	Q C	$\dot{\omega}$	0 °/s²	0 °/s²
$T_s = 10^{-4}\ s$			N = 300,000 Samples		
Q = 6.6276^{-5} C					

Sub-case 1: 30 s simulation time Ts = 10e(-2) s, 10% -Y-offset.

The same amount of offset, only this time at the opposite direction. What was interesting about this case was the effect of bringing the vessels closer than initially planned. The centripetal force would increase and also create new acceleration components.

The simulation parameters used for the sub-case are shown in Table 27.

Results are shown in (Fig. 14). The figure reveals a rapid increase in encirclements radius, change of shape and center of motion.

Table 27: Test case 4, sub-case 2 parameters.

Test case 4 sub-case 2 parameters					
Static Parameters			Motion Parameters		
Parameter	Vessel 1	Vessel 2	Parameter	Vessel 1	Vessel 2
w	0.01 m	0.02 m	G_x or $\vec{r}_x$	0 m	0 m
l_A	0.01 m	0.01 m	G_y or $\vec{r}_y$	0 m	1 m
l_{AB}	0.01 m	0.01 m	$\vec{u}_x$	0 m/s	12.57 m/s
l_B	0.01 m	0.01 m	$\vec{u}_y$	0 m/s	-1.257 m/s
m_A	50 Kg	0.5 Kg	$\vec{a}_x$	0 m/s²	0 m/s²
m_{AB}	0 Kg	0 Kg	$\vec{a}_y$	0 m/s²	0 m/s²
m_B	50 Kg	0.5 Kg	θ	0 °	0 °
Q_A	- Q C	Q C	ω	0 °/s	0 °/s
Q_B	- Q C	Q C	$\dot{\omega}$	0 °/s²	0 °/s²
$T_s = 10^{-2}\ s$			N = 3,000 Samples		

Q = 6.6276^{-5} C

FIGURES

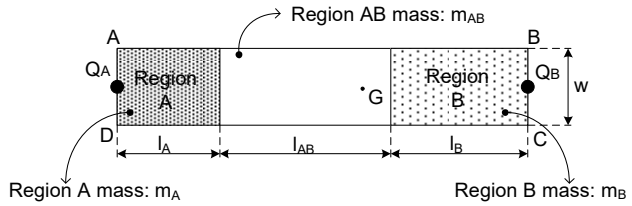


Fig. 1: Vessel approximation.

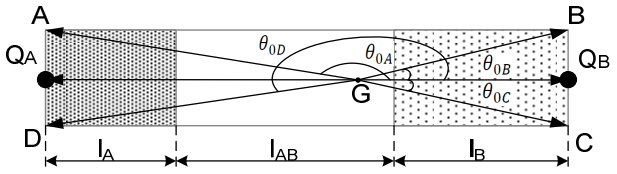


Fig. 2: Vessel description vectors and angles used by simulator.

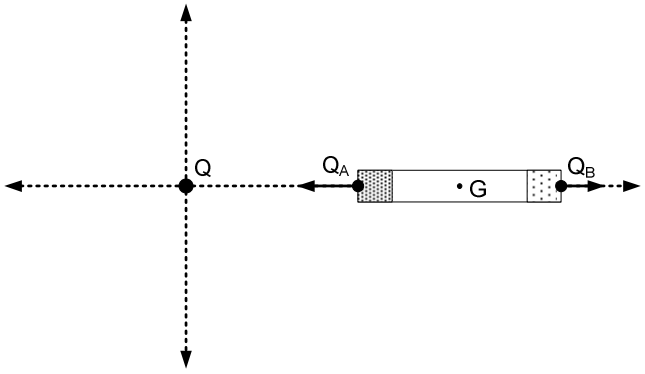


Fig. 3: Test case 1.

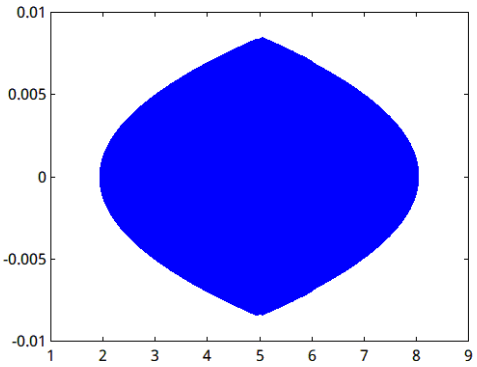


Fig. 4: Test case 1 result.

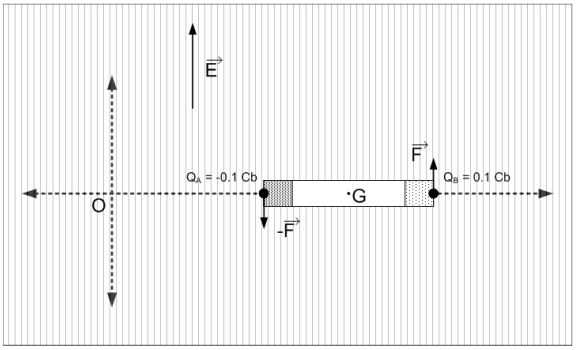


Fig. 6: Test case 2.

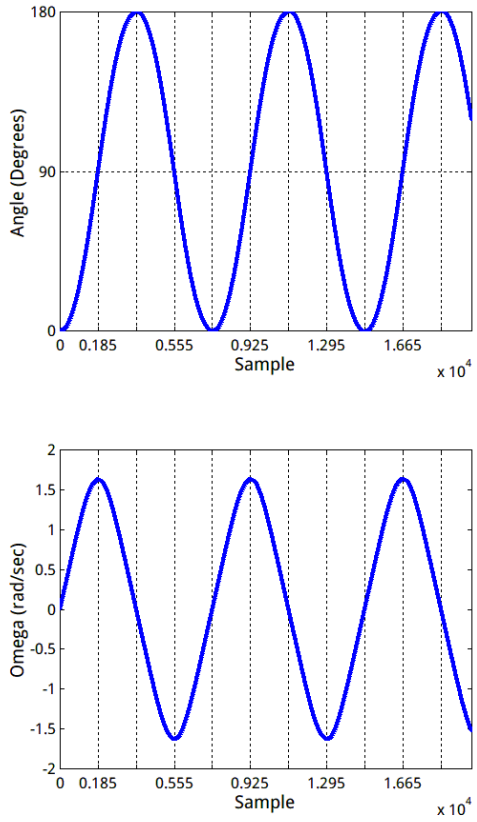


Fig. 5: Test case 2 object angle (θ).

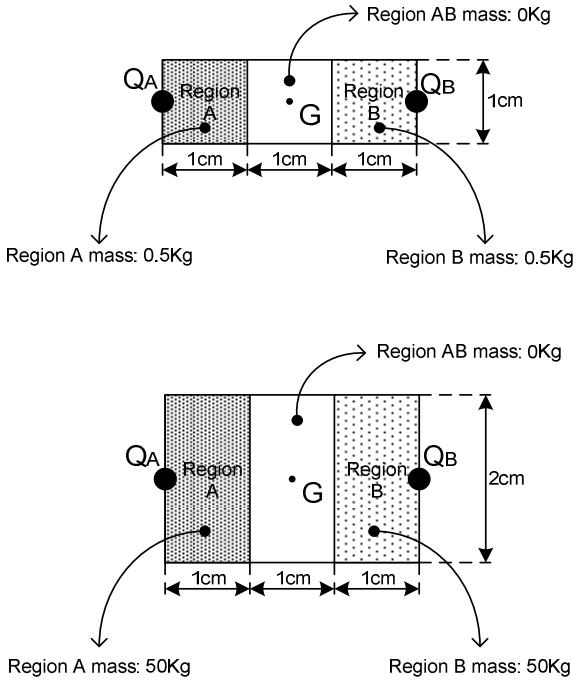


Fig. 7: Test case 3 vessels approximation.

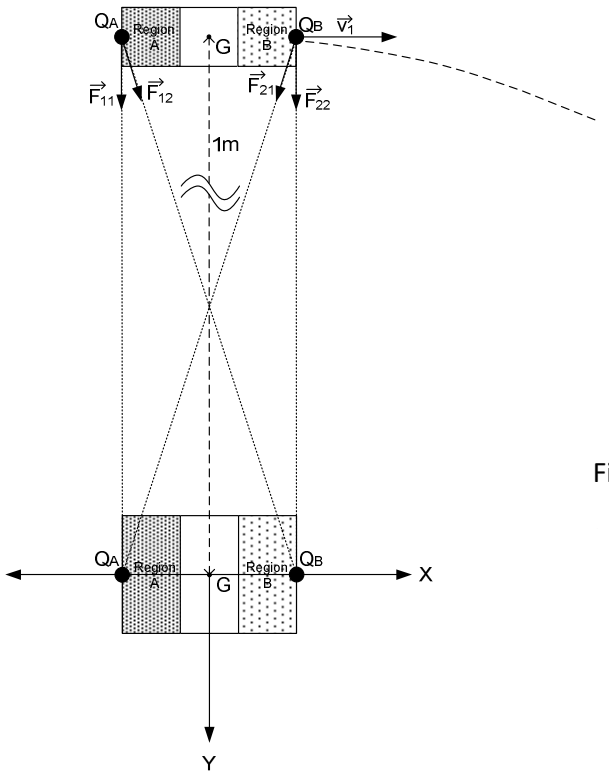


Fig. 8: Test case 3 initial snapshot.

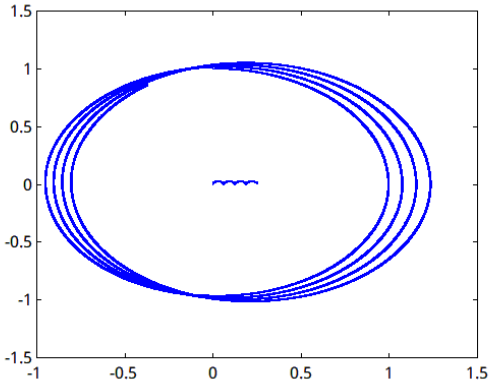


Fig. 9: Result of test case 3, sub-case 1.

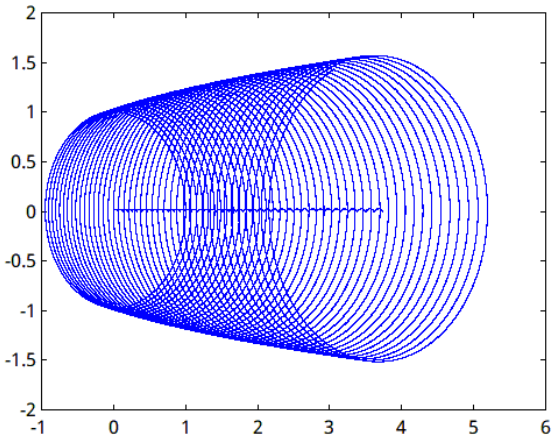


Fig. 10: Results of test case 3, sub-case 2.

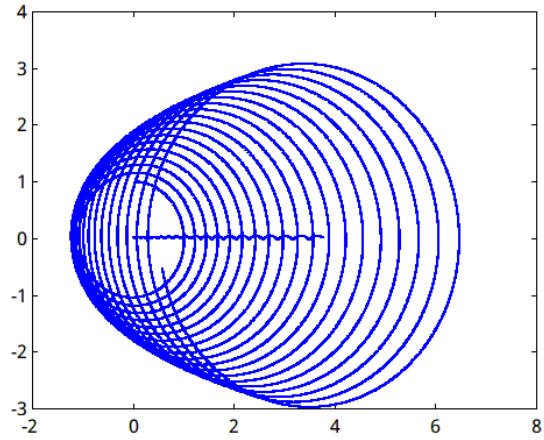


Fig. 11: Results of test case 3, sub-case 3 (Ts = 10e(-3) s).

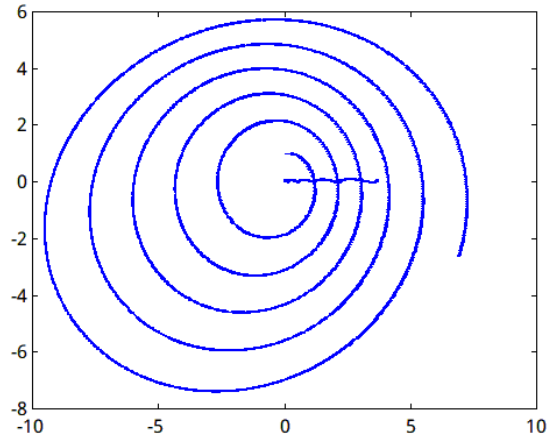


Fig. 12: Results of test case 3, sub-case 3 (Ts = 10e(-2) s).

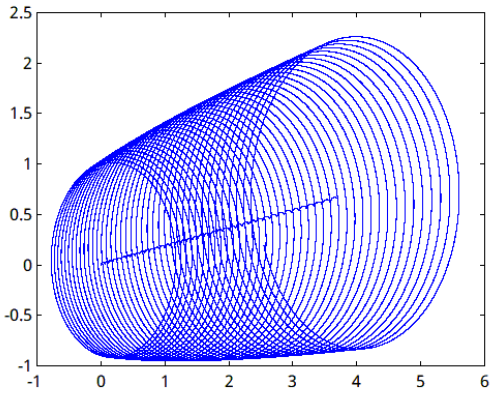


Fig. 13: Results of test case 4, sub-case 1.

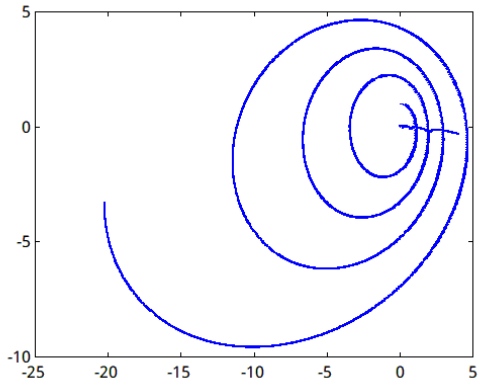
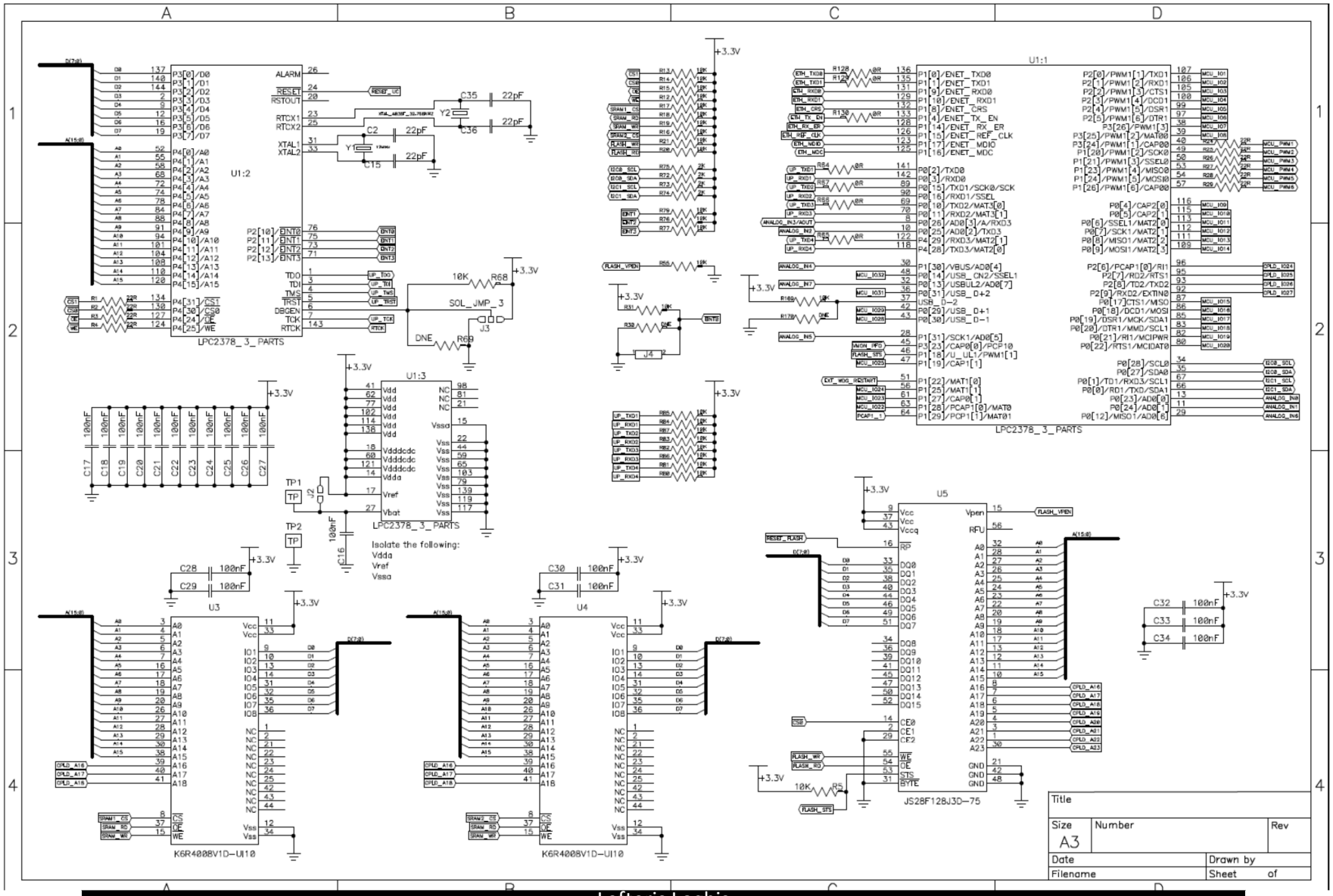


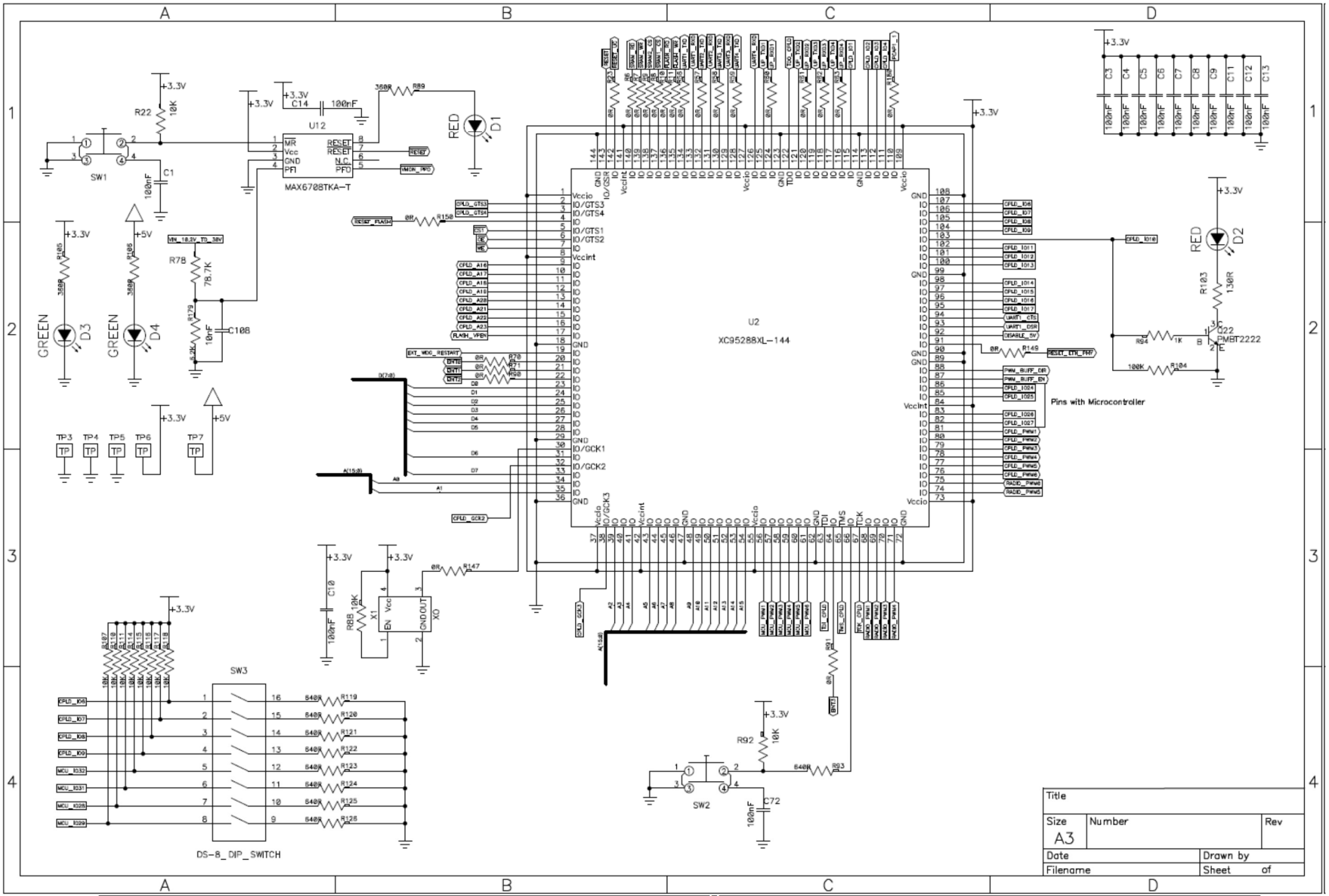
Fig. 14: Results of test case 4, sub-case 2.

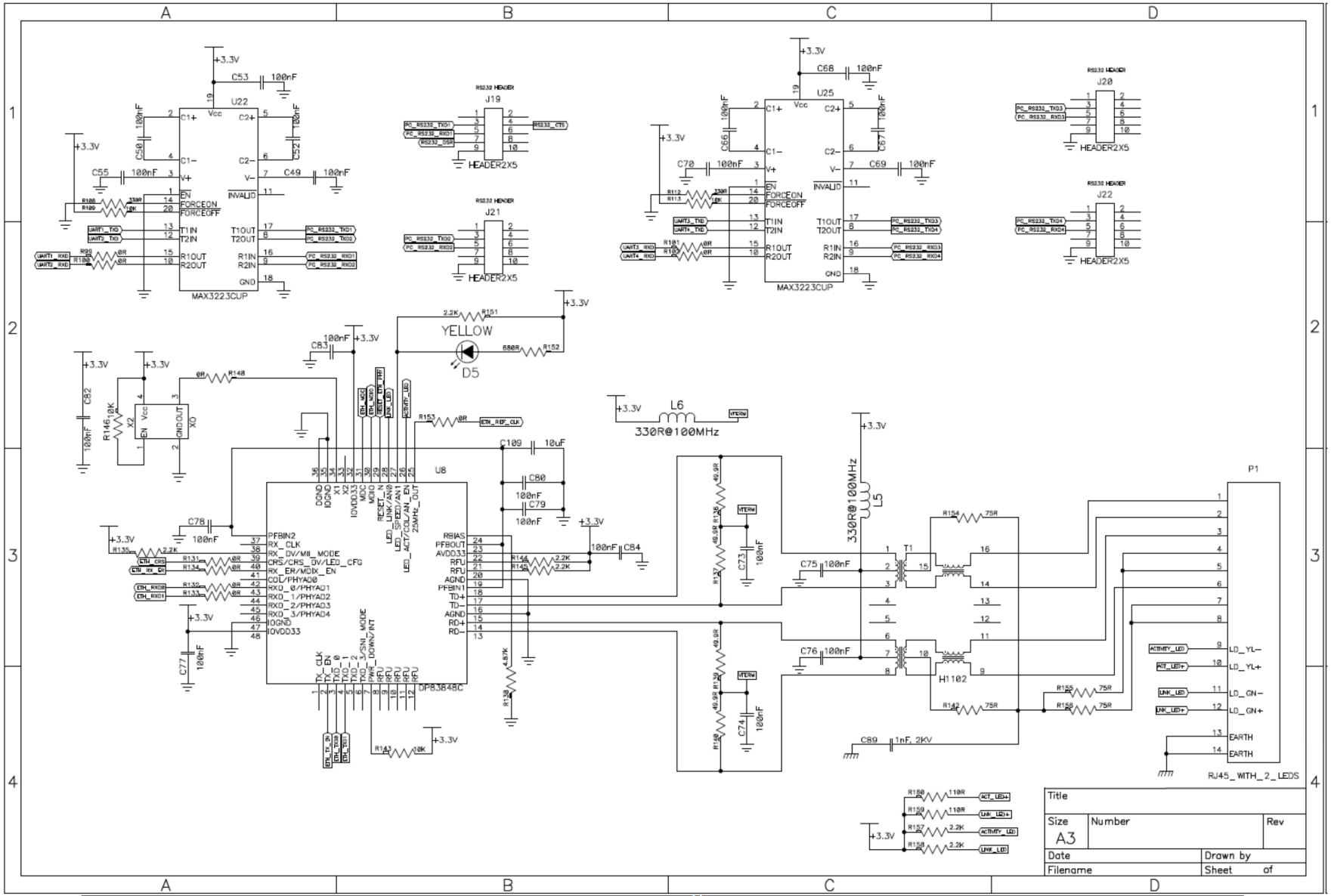
FUTURE WORK

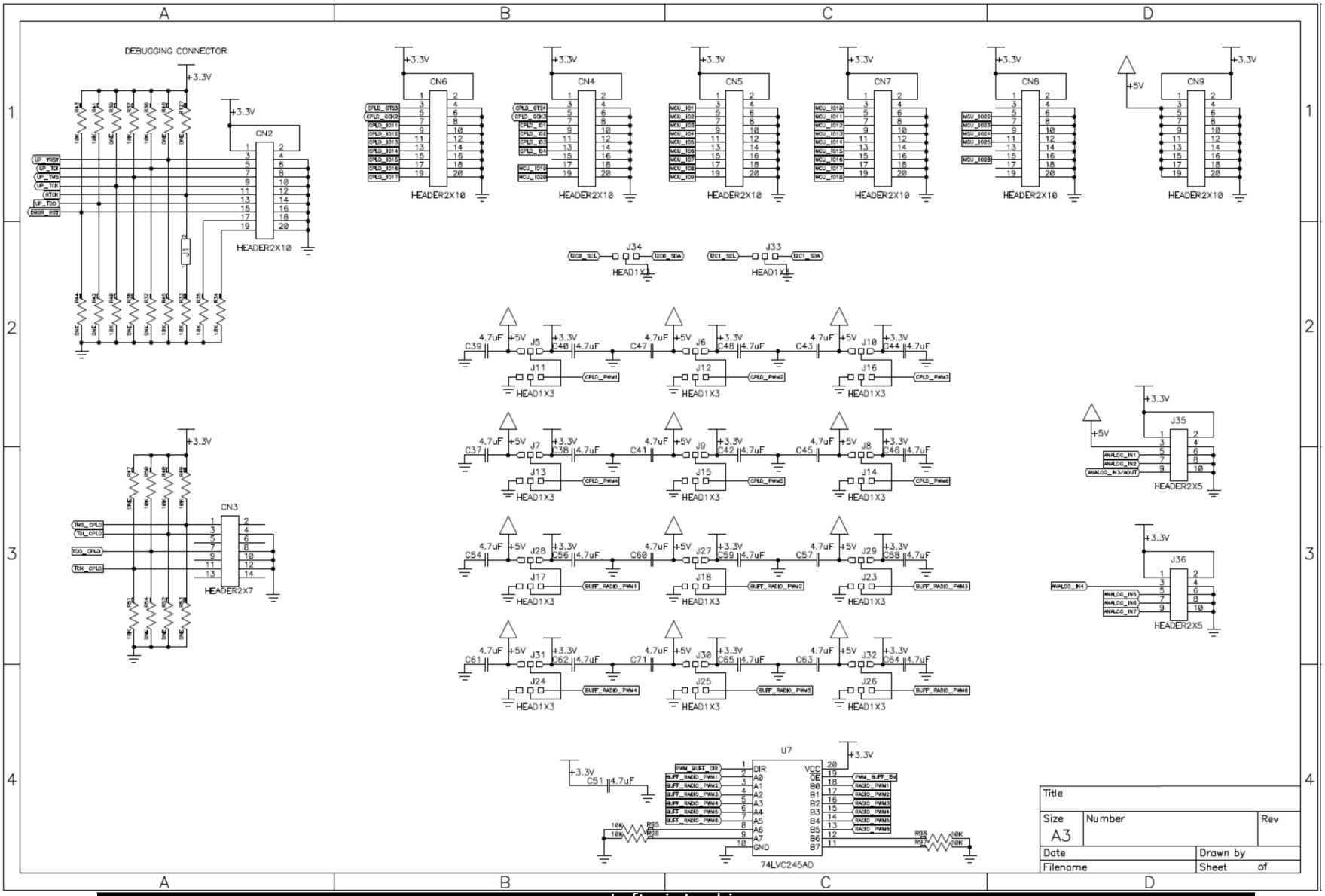
Beginning from the idea of finding a swarm guidance algorithm-methodology, we now have a tool for applying our ideas. The possibilities are essentially limitless. By changing simulation parameters appropriately, we can approximate desired system responses. The next steps in our research will focus on real systems applications and will aim at finding a systematic way to pass from desired responses to the parameters needed in order to achieve them. If motions of a particular accuracy are required, one further layer which employs a control algorithm to regulate key parameters such as vessel charges can be used. In all cases the feasibility of a certain planned course must be evaluated along with the feasible Ts (the rate of algorithm evaluation).

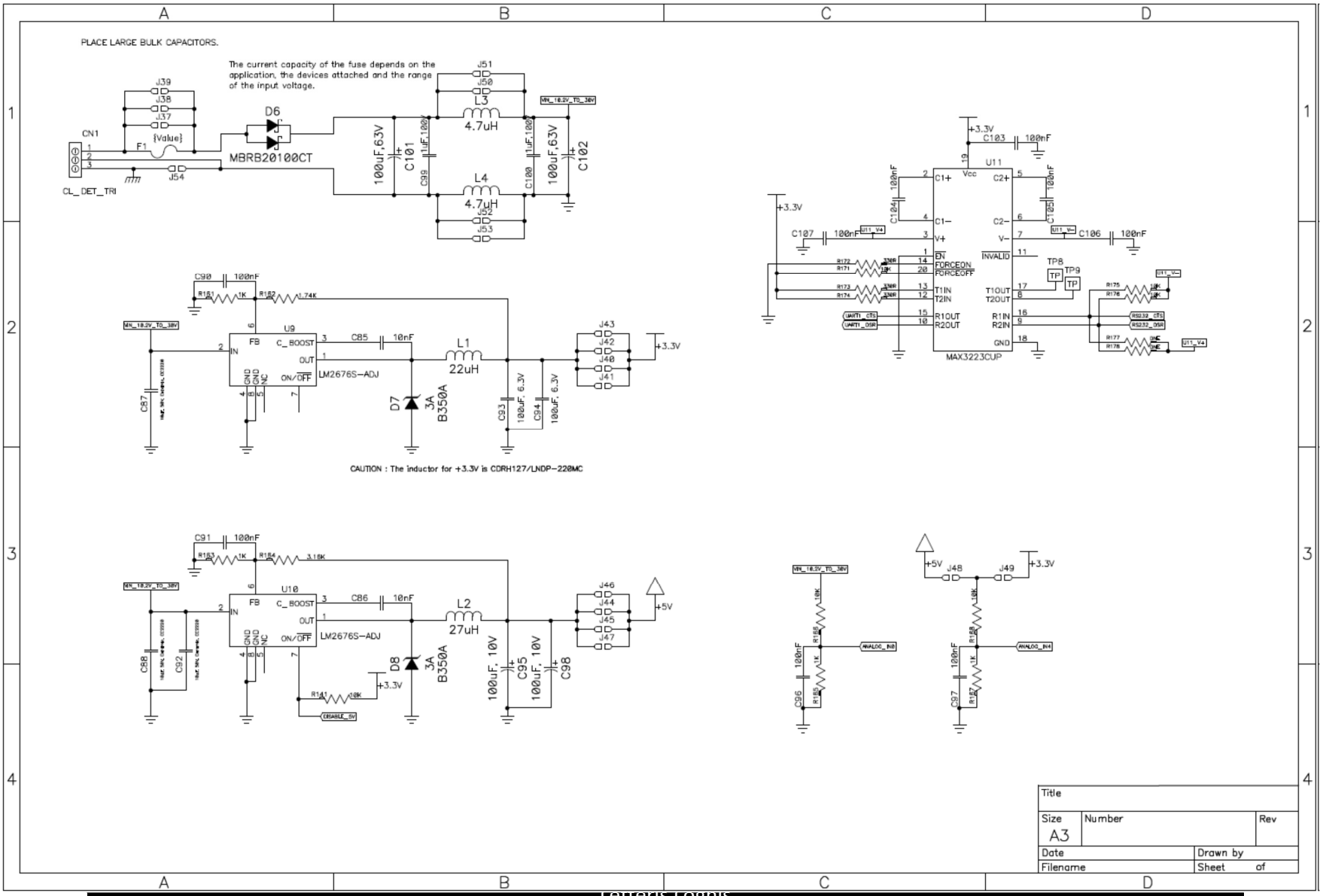
14 APPENDIX 2: AutoPilot_IV Board Schematics


A
B
C
D
1
2
3
4
U1:1
LPC2378_3_PARTS
U1:2
LPC2378_3_PARTS
U1:3
LPC2378_3_PARTS
Isolate the following:
Vdda
Vref
Vssa
U3
K6R4008V1D-UI10
U4
K6R4008V1D-UI10
U5
JS28F128J3D-75
SOL_JMP_3
Title
Size
A3
Number
Rev
Date
Drawn by
Filename
Sheet
of

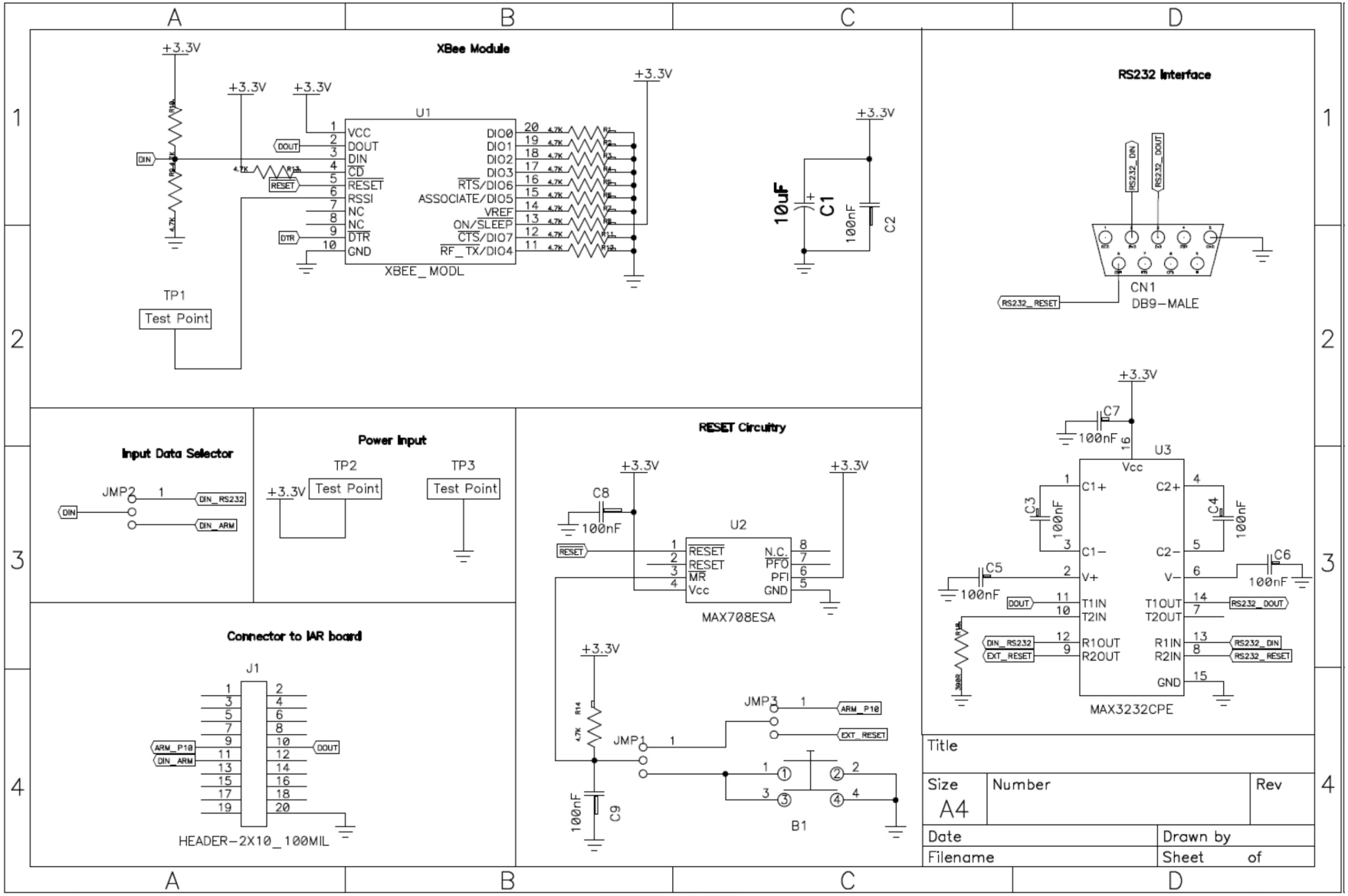

U2
XC95288XL-144
MAX6708TKA-T
DS-8_DIP_SWITCH
Pins with Microcontroller
Title
Size A3
Number
Rev
Date
Drawn by
Filename
Sheet of


A
B
C
D
1
2
3
4
+3.3V
U22
MAX3223CUP
U25
MAX3223CUP
J19
J21
J20
J22
HEADER2X5
YELLOW
D5
L6
330R@100MHz
U8
DP83848C
T1
H1102
P1
RJ45_WITH_2_LEDS
Title
Size
A3
Number
Rev
Date
Drawn by
Filename
Sheet
of

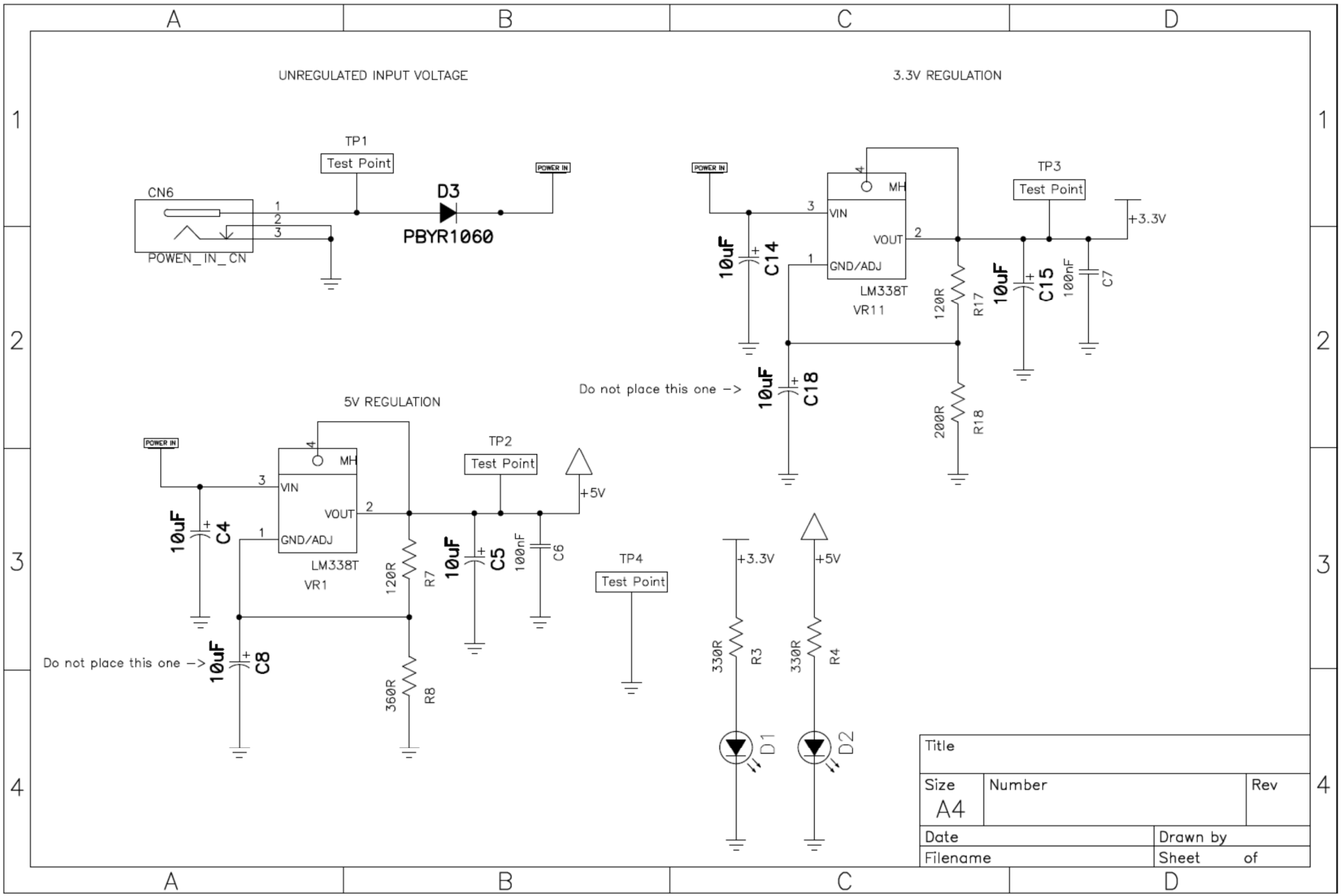

DEBUGGING CONNECTOR
CN2
CN3
CN4
CN5
CN6
CN7
CN8
CN9
HEADER2X10
HEADER2X7
HEADER2X5
HEAD1X3
J33
J34
J35
J36
U7
74LVC245AD
Title
Size
A3
Number
Rev
Date
Drawn by
Filename
Sheet of

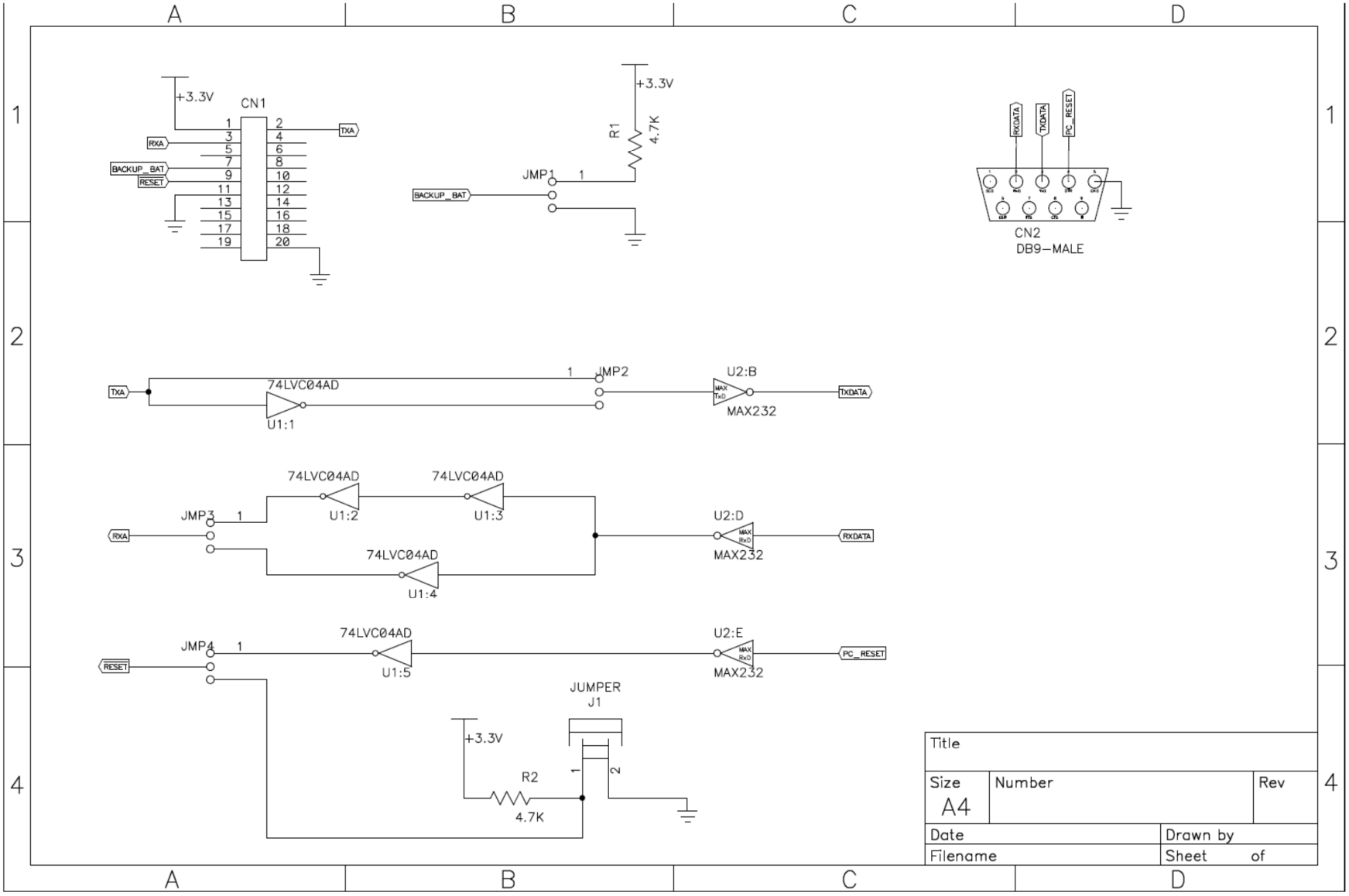

PLACE LARGE BULK CAPACITORS.
The current capacity of the fuse depends on the application, the devices attached and the range of the input voltage.
CN1
CL_DET_TRI
F1
{Value}
J39
J38
J37
J54
D6
MBRB20100CT
J51
J50
L3
4.7uH
L4
4.7uH
J52
J53
100uF,63V
C101
C102
C99
C100
1uF,100V
MIN_18.2V_TO_38V
C90 100nF
R151 1K
R152 1.74K
U9
FB
C_BOOST
IN
OUT
ON/OFF
LM2676S-ADJ
C85 10nF
L1
22uH
D7
3A
B350A
C87
C93 100uF, 6.3V
C94 100uF, 6.3V
J43
J42
J40
J41
+3.3V
CAUTION : The inductor for +3.3V is CDRH127/LNDP-220MC
C91 100nF
R153 1K
R154 3.16K
U10
C86 10nF
L2
27uH
D8
3A
B350A
C88
C92
R141 10K
DISABLE_5V
100uF, 10V
C95
100uF, 10V
C98
J46
J44
J45
J47
+5V
+3.3V
C103 100nF
U11
Vcc
C1+
C1-
C2+
C2-
V+
V-
C104 100nF
C105 100nF
C107 100nF
C106 100nF
EN
FORCEON
FORCEOFF
INVALID
T1IN
T2IN
R1OUT
R2IN
GND
T1OUT
T2OUT
R1IN
R2OUT
MAX3223CUP
TP8
TP9
R172 330K
R171 10K
R173 330K
R174 330K
R175 10K
R176 10K
R177 10K
R178 10K
UART1_CTS
UART1_DSR
RS232_CTS
RS232_DSR
U11_V+
U11_V-
U11_V4
+5V
J48
J49
+3.3V
R165 10K
R166 10K
R167 1K
R168 1K
C96 100nF
C97 100nF
ANALOG_IN3
ANALOG_IN4
Title
Size A3
Number
Rev
Date
Drawn by
Filename
Sheet of

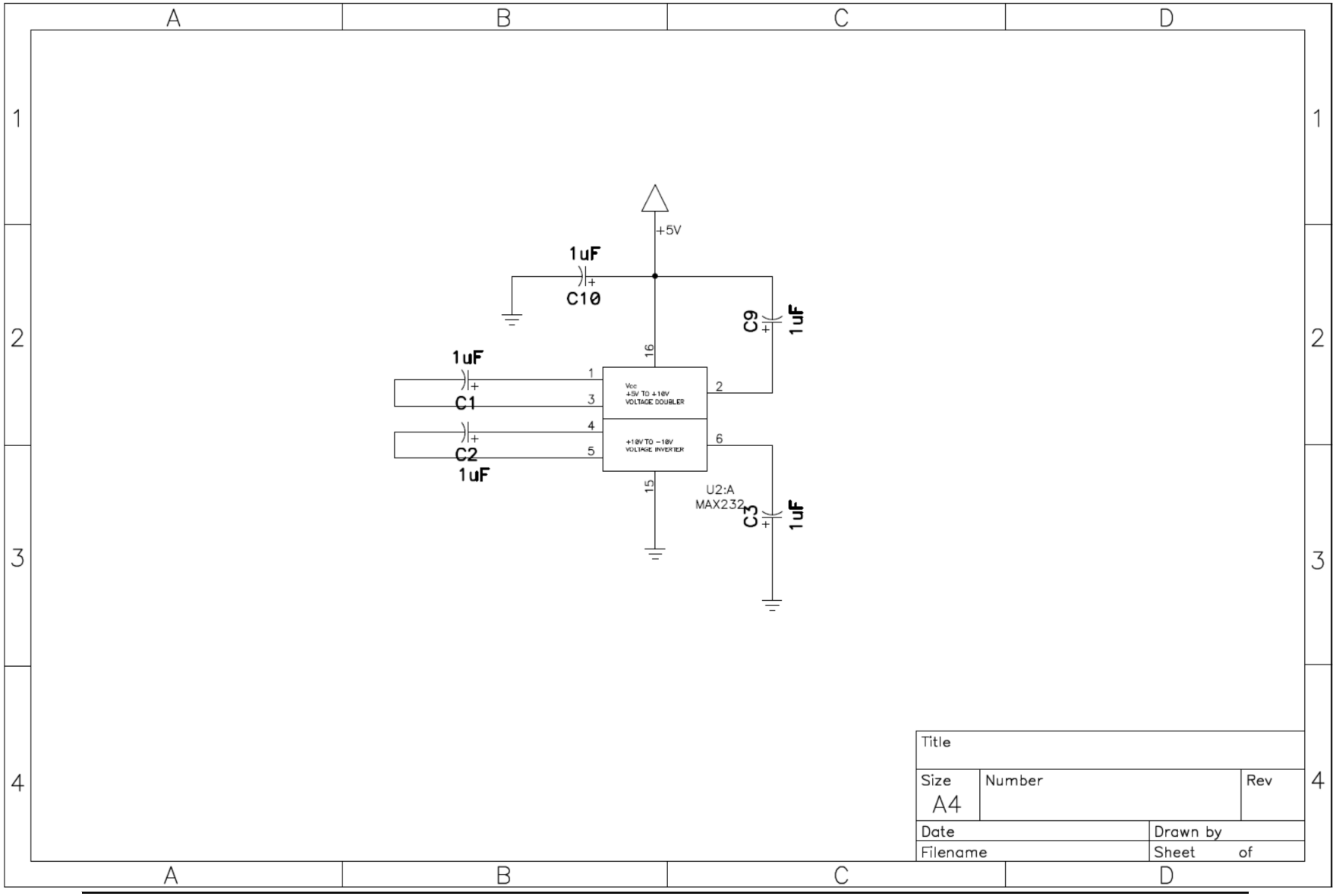
15 APPENDIX 3: ZigBee Module Adaptor Board Schematics


XBee Module
U1
XBEE_MODL
TP1
Test Point
RS232 Interface
CN1
DB9-MALE
Input Data Selector
JMP2
Power Input
TP2
TP3
Test Point
RESET Circuitry
U2
MAX708ESA
U3
MAX3232CPE
Connector to IAR board
J1
HEADER-2X10_100MIL
JMP1
JMP3
B1
Title
Size
A4
Number
Rev
Date
Drawn by
Filename
Sheet of

16 APPENDIX 4: GPS Adaptor Board Schematics


UNREGULATED INPUT VOLTAGE
TP1
Test Point
CN6
POWEN_IN_CN
D3
PBYR1060
POWER IN
3.3V REGULATION
TP3
Test Point
+3.3V
MH
VIN
VOUT
GND/ADJ
LM338T
VR11
10uF
C14
120R
R17
10uF
C15
100nF
C7
Do not place this one –>
10uF
C18
200R
R18
5V REGULATION
TP2
Test Point
+5V
LM338T
VR1
10uF
C4
120R
R7
10uF
C5
100nF
C6
TP4
Test Point
Do not place this one –>
10uF
C8
360R
R8
330R
R3
330R
R4
D1
D2
Title
Size
A4
Number
Rev
Date
Drawn by
Filename
Sheet of


A
B
C
D
1
2
3
4
+3.3V
CN1
RXA
TXA
BACKUP_BAT
RESET
R1
4.7K
JMP1
CN2
DB9–MALE
RXDATA
TXDATA
PC_RESET
74LVC04AD
U1:1
JMP2
U2:B
MAX232
TXDATA
U1:2
U1:3
JMP3
U2:D
RXDATA
U1:4
JMP4
U1:5
U2:E
JUMPER
J1
R2
4.7K
Title
Size
A4
Number
Rev
Date
Drawn by
Filename
Sheet of


A
B
C
D
1
2
3
4
+5V
1uF
C10
C9
1uF
16
1uF
C1
1
3
Vcc
+5V TO +10V
VOLTAGE DOUBLER
2
4
5
+10V TO −10V
VOLTAGE INVERTER
6
C2
1uF
15
U2:A
MAX232
C3
1uF
Title
Size
A4
Number
Rev
Date
Drawn by
Filename
Sheet of

17 APPENDIX 5: Alternative Implementation of PID Controllers (trapezoidal Integration)

For PID controllers the form known as “Velocity Form” is used. The purpose of this section is to show how we reach to the velocity form starting from analog PID controller. Next the way the gains were selected is also presented.
Analog PID has the form [25]–[27], [42]:

$$m(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right] \quad (5.104)$$

where:

- m(t) is the output of controller at the t-th instance of time.
- e(t) is the error of the controlled variable.
- $r_{ref}(t)$ is the reference input for the system to follow.
- c(t) is the system output.

All these variables are schematically imprinted at Figure 128: PID control scheme and variables.

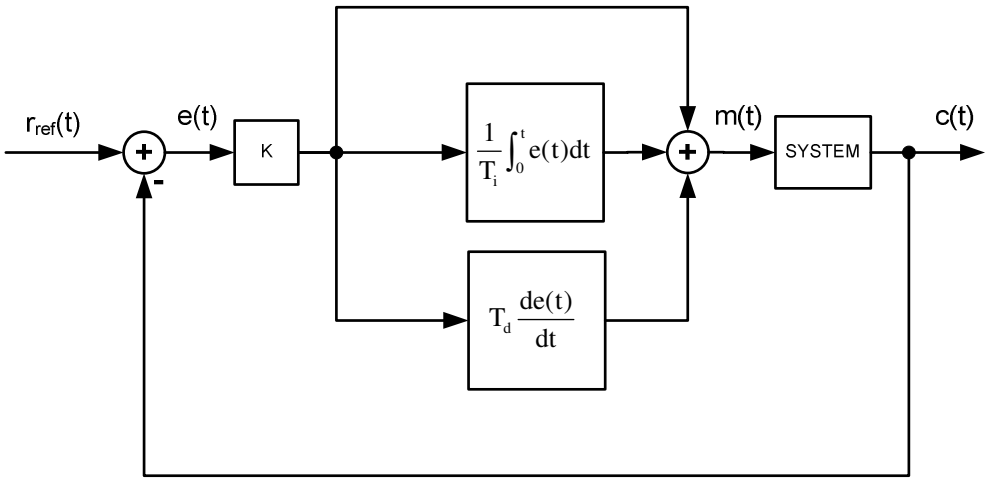


Figure 128: PID control scheme and variables.

Feedback Controller operates in discrete time, thus all variables, functions and operations must be supported by their discrete time equivalent.
To discretize equation (5.105) we first note the conversion of time variable:

$$t \to kT_s \quad (5.106)$$

Thus $m(t) \to m[kT_s]$ and $e(t) \to e[kT_s]$ where k is the sampling instant and T_s the sampling period (30ms).

To discretize the integral part, we use the trapezoidal integration method (see section trapz and cumptrapz for the derivation of the integral discretization). The outcome is:

$$\frac{1}{T_i}\int_0^t e(t)dt \rightarrow \frac{T_s}{T_i}\sum_{n=1}^{k}\frac{e[(n-1)T_s]+e[nT_s]}{2} \tag{5.107}$$

For the dicretiztion of the D-term we use the backward difference method:

$$T_d\frac{de(t)}{dt} \rightarrow T_d\frac{e[kT_s]-e[(k-1)T_s]}{T_s} \tag{5.108}$$

As a consequence of , we have a first discrete form of the PID controller as follows:

$$m[kT_s] = K\left[e[kT_s]+\frac{T_s}{T_i}\sum_{n=1}^{k}\frac{e[(n-1)T_s]+e[nT_s]}{2}+T_d\frac{e[kT_s]-e[(k-1)T_s]}{T_s}\right] \tag{5.109}$$

In order to analyze the response of the discrete PID and design for its parameters, z-transform of equation (2.6) is needed. To obtain such a transform, the following steps are followed:

- Obtain z-transform of $e[kT_s]$:

$$Z\{e[kT_s]\} = \sum_{k=0}^{\infty} e[kT_s]z^{-k} = E(z) \tag{5.110}$$

- Obtain the z-transform of the derivative term:

$$Z\left\{T_d\frac{e[kT_s]-e[(k-1)T_s]}{T_s}\right\} = \frac{T_d}{T_s}\left[E(z)-z^{-1}E(z)\right] = \frac{T_d}{T_s}(1-z^{-1})E(z) \tag{5.111}$$

- Obtain the z-transform of the trapezoidal integration part. To do so the next steps are needed:
  - Find z-transform of hypothetical signal $y_h[kT_s]=\sum_{n=0}^{k}e[nT_s]$:

We have that:

$$y_h[0] = \sum_{n=0}^{0} e[nT_s] = e[0]$$

$$y_h[1T_s] = \sum_{n=0}^{1} e[nT_s] = e[0]+e[T_s] = y_h[0]+e[T_s]$$

$$\vdots$$

$$y_h[mT_s] = \sum_{n=0}^{m} e[nT_s] = y_h[(m-1)T_s]+e[mT_s]$$

which leads to

$$y_h[kT_s]=y_h[(k-1)T_s]+e[kT_s]\Rightarrow$$
$$Z\{y_h[kT_s]\}=Z\{y_h[(k-1)T_s]+e[kT_s]\}\Rightarrow$$
$$Y_h(z)=z^{-1}Y_h(z)+E(z)\Leftrightarrow$$

$$Y_h(z)=\frac{E(z)}{1-z^{-1}} \tag{5.112}$$

- Find z-transform of hypothetical signal $f[kT_s]=\frac{e[(k-1)T_s]+e[kT_s]}{2}$:

$$Z\left\{\sum_{n=0}^{k}f[kT_s]\right\}=Z\left\{\sum_{n=0}^{k}\frac{e[(n-1)T_s]+e[nT_s]}{2}\right\}=$$
$$\frac{1}{1-z^{-1}}Z\left\{\frac{e[(k-1)T_s]+e[kT_s]}{2}\right\}=$$
$$\frac{1}{1-z^{-1}}\bullet\frac{1}{2}\{Z\{e[(k-1)T_s]\}+Z\{e[kT_s]\}\}=$$
$$\frac{1}{2\left(1-z^{-1}\right)}\left[z^{-1}E(z)+E(z)\right]\Leftrightarrow$$

$$Z\left\{\sum_{n=0}^{k}f[kT_s]\right\}=\frac{1+z^{-1}}{2\left(1-z^{-1}\right)}E(z) \tag{5.113}$$

- By combining the above steps, we reach to the required result regarding z-transform of PID:

As a first step we must note that integration approximation starts from n=1. But returning for a moment to signal f[kTs] we have that:

$$Z\left\{\sum_{n=1}^{k}f[kT_s]\right\}=Z\left\{\left[\sum_{n=0}^{k}f[kT_s]\right]-f(0)\right\}$$

but $f(0)=\frac{e[-T_s]-e[0]}{2}=0$ since at k=-1 and k=0 there is no signal or feedback yet.

Thus:

$$Z\{m[kT_s]\}=Z\left\{K\left[e[kT_s]+\frac{T_s}{T_i}\sum_{n=1}^{k}\frac{e[(n-1)T_s]+e[nT_s]}{2}+T_d\frac{e[kT_s]-e[(k-1)T_s]}{T_s}\right]\right\}\Rightarrow$$

$$\boxed{M(z)=K\left[E(z)+\frac{T_s}{T_i}\frac{1+z^{-1}}{2\left(1-z^{-1}\right)}E(z)+\frac{T_d}{T_s}(1-z^{-1})E(z)\right]} \tag{5.114}$$

Moving the idea a bit forward we can andvance (2.11) as follows:

Since $\frac{1+z^{-1}}{2\left(1-z^{-1}\right)}=\frac{1+z^{-1}+1-1}{2\left(1-z^{-1}\right)}=\frac{2-(1-z^{-1})}{2\left(1-z^{-1}\right)}=\frac{2}{2\left(1-z^{-1}\right)}-\frac{1}{2}=\frac{1}{\left(1-z^{-1}\right)}-\frac{1}{2}$

we have from (2.11):

$$M(z)=K\left[E(z)+\frac{T_s}{T_i}\left(\frac{1}{1-z^{-1}}-\frac{1}{2}\right)E(z)+\frac{T_d}{T_s}(1-z^{-1})E(z)\right]\Leftrightarrow$$

$$\boxed{M(z)=\left[\left(K-\frac{T_s}{2T_i}\right)+\frac{KT_s}{T_i}\frac{1}{1-z^{-1}}+\frac{KT_d}{T_s}(1-z^{-1})\right]E(z)} \quad (5.115)$$

which implies that:

Proportional gain K_p is:

$$K_p=K-\frac{KT_s}{2T_i} \quad (5.116)$$

Integral gain K_i is:

$$K_i=\frac{KT_s}{T_i} \quad (5.117)$$

Derivative gain is:

$$K_d=\frac{KT_d}{T_s} \quad (5.118)$$

As a note, it is important to see that sampling period T_s is present as a factor in all gains!

Using the notation of equations (2.13) to (2.15) we have for (2.12):

$$\boxed{G_{PID}(z)=\frac{M(z)}{E(z)}=K_p+\frac{K_i}{1-z^{-1}}+K_d(1-z^{-1})} \quad (5.119)$$

This form of digital PID implementation is called “positional form”.

17.1.1 Velocity Form of Digital PID

To reach the velocity form of digital PID implementation we begin by considering the difference:

$$\nabla m[kT_s] = m[kT_s] - m[(k-1)T_s] \tag{5.120}$$

By employing equation (2.6) for $m[kT_s]$ we have:

$$\begin{aligned}\nabla m[kT_s] = K\{e[kT_s] - e[(k-1)T_s] + \frac{T_s}{2T_i}\{e[(k-1)T_s] + e[kT_s]\} + \\ \frac{T_d}{T_s}\{e[kT_s] - e[(k-1)T_s] - e[(k-1)T_s] + e[(k-2)T_s]\}\}\end{aligned} \tag{5.121}$$

To clarify term $\frac{T_s}{2T_i}\{e[(k-1)T_s] + e[kT_s]\}$:

$$\frac{T_s}{T_i}\sum_{n=1}^{k}\frac{e[(n-1)T_s] + e[nT_s]}{2} - \frac{T_s}{T_i}\sum_{n=1}^{k-1}\frac{e[(n-1)T_s] + e[nT_s]}{2} = \frac{T_s}{T_i}\frac{e[(k-1)T_s] + e[kT_s]}{2} =$$

$$= \frac{T_s}{2T_i}\{e[(k-1)T_s] + e[kT_s]\}$$

Continuing equation (2.18) we have:

$$\nabla m[kT_s] = K\{e[kT_s] - e[(k-1)T_s] + \frac{T_s}{2T_i}\{e[(k-1)T_s] + e[kT_s]\} +$$

$$\frac{T_d}{T_s}\{e[kT_s] - 2e[(k-1)T_s] + e[(k-2)T_s]\}\} \Leftrightarrow$$

$$\nabla m[kT_s] = Ke[kT_s] - Ke[(k-1)T_s] + \frac{KT_s}{2T_i}e[(k-1)T_s] + \frac{KT_s}{2T_i}e[kT_s] +$$

$$+\frac{KT_d}{T_s}\{e[kT_s] - 2e[(k-1)T_s] + e[(k-2)T_s]\} \Leftrightarrow$$

$$\nabla m[kT_s] = Ke[kT_s] + \left\{\frac{K_i}{2_i}e[kT_s] - \frac{K_i}{2_i}e[kT_s]\right\}$$

$$-Ke[(k-1)T_s] + \frac{K_i}{2_i}e[(k-1)T_s] + \frac{K_i}{2}e[kT_s] +$$

$$+K_d\{e[kT_s] - 2e[(k-1)T_s] + e[(k-2)T_s]\} \Leftrightarrow$$

$$\nabla m[kT_s] = Ke[kT_s] - Ke[(k-1)T_s] + \frac{K_i}{2_i}e[(k-1)T_s] + \frac{K_i}{2}e[kT_s] +$$

$$+K_d\{e[kT_s] - 2e[(k-1)T_s] + e[(k-2)T_s]\} \Leftrightarrow$$

$$\nabla m[kT_s]=\left(K-\frac{K_i}{2_i}\right)e[kT_s]+$$
$$-\left(K-\frac{K_i}{2_i}\right)e[(k-1)T_s]+K_i e[kT_s]+$$
$$+K_d\left\{e[kT_s]-2e[(k-1)T_s]+e[(k-2)T_s]\right\}\Leftrightarrow$$

$$\nabla m[kT_s]=\left(K-\frac{K_i}{2_i}\right)\left\{e[kT_s]-e[(k-1)T_s]\right\}+K_i e[kT_s]+ \quad (5.122)$$
$$K_d\left\{e[kT_s]-2e[(k-1)T_s]+e[(k-2)T_s]\right\}$$

Referring to Figure 128, we see that

$$e[kT_s]=r_{ref}[kT_s]-c[kT_s] \quad (5.123)$$

By combining (2.19) and (2.20) we have that

$$\nabla m[kT_s]=\left(K-\frac{K_i}{2_i}\right)\left\{r_{ref}[kT_s]-c[kT_s]-r_{ref}[(k-1)T_s]+c[(k-1)T_s]\right\}+K_i\left\{r_{ref}[kT_s]-c[kT_s]\right\}+$$
$$K_d\left\{r_{ref}[kT_s]-c[kT_s]-2r_{ref}[(k-1)T_s]+2c[(k-1)T_s]+r_{ref}[(k-2)T_s]-c[(k-2)T_s]\right\}\Leftrightarrow$$
$$\nabla m[kT_s]=\left(K-\frac{K_i}{2_i}\right)\left\{r_{ref}[kT_s]-c[kT_s]-r_{ref}[(k-1)T_s]+c[(k-1)T_s]\right\}+K_i\left\{r_{ref}[kT_s]-c[kT_s]\right\}+$$
$$K_d\left\{r_{ref}[kT_s]-2r_{ref}[(k-1)T_s]+r_{ref}[(k-2)T_s]-c[kT_s]+2c[(k-1)T_s]-c[(k-2)T_s]\right\}$$

$$\nabla m[kT_s]=\left(K-\frac{K_i}{2_i}\right)\left\{r_{ref}[kT_s]-c[kT_s]-r_{ref}[(k-1)T_s]+c[(k-1)T_s]\right\}+$$
$$K_i\left\{r_{ref}[kT_s]-c[kT_s]\right\}+$$
$$K_d\left\{r_{ref}[kT_s]-2r_{ref}[(k-1)T_s]+r_{ref}[(k-2)T_s]-c[kT_s]+2c[(k-1)T_s]-c[(k-2)T_s]\right\} \quad (5.124)$$

The target of creating this form is to (kind of) filter sudden changes at the reference input, so as to avoid large control actions for a sudden reference input change.
To do so, we arbitrarily consider that $r_{ref}[kT_s]=r_{ref}[(k-1)T_s]=r_{ref}[(k-2)T_s]$. By applying this hypothesis to (2.21) we have that:

$$\nabla m[kT_s]=\left(K-\frac{K_i}{2_i}\right)\left\{r_{ref}[kT_s]-c[kT_s]-r_{ref}[kT_s]+c[(k-1)T_s]\right\}+$$
$$K_i\left\{r_{ref}[kT_s]-c[kT_s]\right\}+$$
$$K_d\left\{r_{ref}[kT_s]-2r_{ref}[kT_s]+r_{ref}[kT_s]-c[kT_s]+2c[(k-1)T_s]-c[(k-2)T_s]\right\}\Leftrightarrow$$

$$\nabla m[kT_s] =$$
$$-K_p\{c[kT_s]-c[(k-1)T_s]\}+$$
$$K_i\{r_{ref}[kT_s]-c[kT_s]\}+$$
$$-K_d\{c[kT_s]-2c[(k-1)T_s]+c[(k-2)T_s]\}$$

$$\boxed{\begin{aligned} &m[kT_s] = \\ &m[(k-1)T_s]+ \\ &-K_p\{c[kT_s]-c[(k-1)T_s]\}+ \\ &K_i\{r_{ref}[kT_s]-c[kT_s]\}+ \\ &-K_d\{c[kT_s]-2c[(k-1)T_s]+c[(k-2)T_s]\} \end{aligned}} \quad (5.125)$$

Equation (2.22) is used at the practical implementation of the velocity form of digital PID controller.

To use velocity form in a stability analysis of a system, we convert it to its z-transform form:

$$M(z)=z^{-1}M(z)-K_p\left[C(z)-z^{-1}C(z)\right]+K_i\left[R_{ref}(z)-C(z)\right]$$
$$-K_d\left[C(z)-2z^{-1}C(z)+z^{-2}C(z)\right]\Leftrightarrow$$
$$\left(1-z^{-1}\right)M(z)=-K_p\left(1-z^{-1}\right)C(z)+K_i\left[R_{ref}(z)-C(z)\right]-K_d\left(1-2z^{-1}+2z^{-2}\right)C(z)\Leftrightarrow$$
$$\left(1-z^{-1}\right)M(z)=-K_p\left(1-z^{-1}\right)C(z)+K_i\left[R_{ref}(z)-C(z)\right]-K_d\left(1-z^{-1}\right)^2C(z)\Leftrightarrow$$

$$\boxed{M(z)=-K_pC(z)+K_i\frac{R_{ref}(z)-C(z)}{\left(1-z^{-1}\right)}-K_d\left(1-z^{-1}\right)C(z)} \quad (5.126)$$

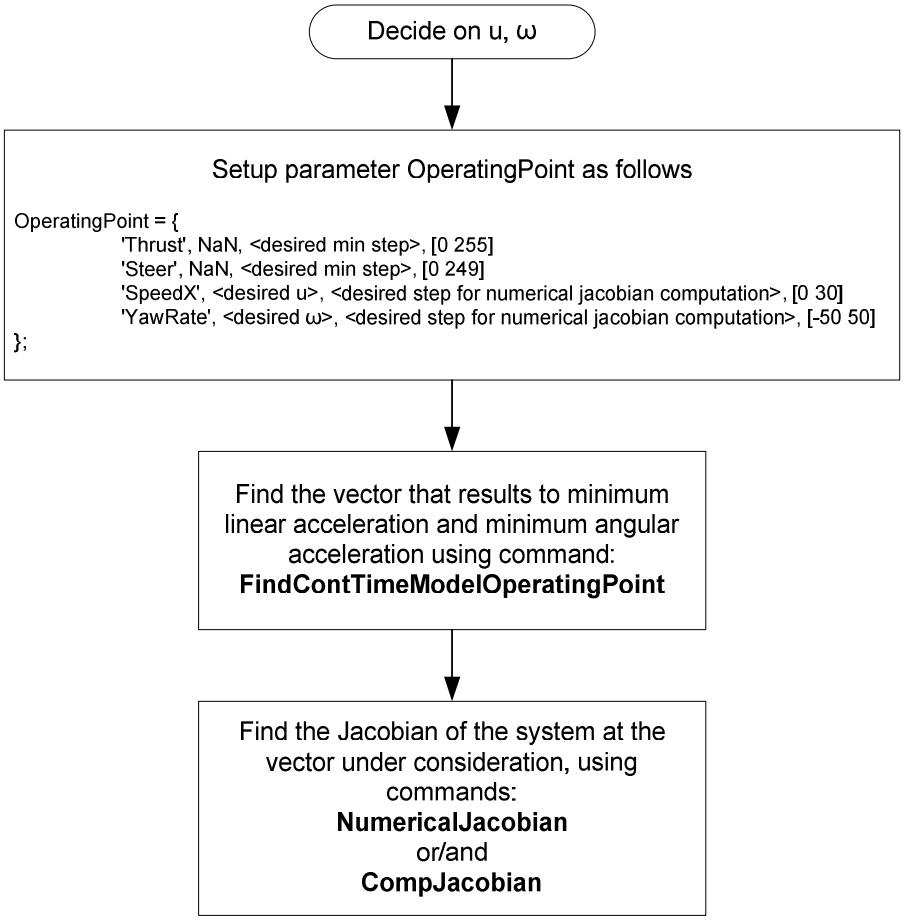
18 APPENDIX 6. Abandoned system Linearization Trials

Before continuing with the idea of MBC controller, we tried to linearize the vessel model [43], [44]. The result was a linear system with poles that covered a lot of space in both positive and negative

After reaching an acceptable training of the neural networks as shown in the previous section, the required procedure in order to reach a model to base the robust controller implementation on, is proceed to linearizing our model at various equilibrium points. The method chosen was to create a grid of equilibrium points for many values of surge velocity and angular velocity. For this reason we divided both velocities into equal spaced fragments and created a grid. For every point at this grid, thrust and steer settings that brought the system to equilibrium were found. The four parameters that comprise the input to the model at the selected equilibrium (ie $[y_{\%}\ \delta_{\%}\ u_{\%}\ \omega_{\%}]'$) were used in conjunction to the model equations to get a linearized model at the specific equilibrium.
The following section provides an example

18.1 Example Selection of Operating Point

The example operating point of linearization will be the most commonly used cruising speed point, i.e. 75% of max speed (about 15Km/h), no turning. For this point we must find the inputs to the system that lead to stabilizing this point at no external disturbances. The algorithm follows the following simple steps:



Based on our implementation, the commands used to get the desired output are:

cd C:\PHD\Matlab_180910\NewralNetworksTransferFunction\MatlabCode

AccXNNFilename = '../../AccXNeuralFolder/AccXNeural_20140414_1051/AccXNeural_20140414_1051.mat';
WdotNNFilename = '../../WdotNeural/WdotNeural_20140414_1058/WdotNeural_20140414_1058.mat';
ExperimentFilename = 'Verde_2_6_ForMatlab.csv';
DecimationFactor = 5;

SignalsOffsetStruct.TimeStart = 281*0.03; %10;
SignalsOffsetStruct.TimeEnd = 1866*0.03; %70;
SignalsOffsetStruct.ThrustOffst = NaN;
SignalsOffsetStruct.SteerOffst = NaN;
SignalsOffsetStruct.DirectionOffst = NaN;
SignalsOffsetStruct.AccXOffst = 1.42;

```
SignalsOffsetStruct.AccYOffst = NaN;
SignalsOffsetStruct.AccZOffst = NaN;
SignalsOffsetStruct.RollRateOffst = NaN;
SignalsOffsetStruct.PitchRateOffst = NaN;
SignalsOffsetStruct.YawRateOffst = 15.625;
SignalsOffsetStruct.PitchCompassOffst = -5;
SignalsOffsetStruct.SubtractionTable = [
        0.009000 191.000000
        0.018000 1141.000000
];
SignalsOffsetStruct.ThrustNeededForMotionStart = 68;

addpath ../../
addpath ../../StatLin/

PlotSettings.LineWidth = 3;
PlotSettings.TitleFontSize = 20;
PlotSettings.TitleFontWeight = 'bold';
PlotSettings.AxesLabelsFontSize = 16;
PlotSettings.AxesFontSize = 14;
PlotSettings.GridOn = 1;
PlotSettings.CloseAll = 1;

OperatingPoint = {
        'Thrust', NaN, 10, [0 255]
        'Steer', NaN, 10, [0 249]
        'SpeedX', 15, 0.01, [0 30]
        'YawRate', 0, 0.01, [-50 50]
};

Thresholds.AccX = 0.001;
Thresholds.YawAcc = 0.0001;

OpPoint = FindContTimeModelOperatingPoint ( AccXNNFilename,
WdotNNFilename, OperatingPoint, Thresholds, PlotSettings );
```

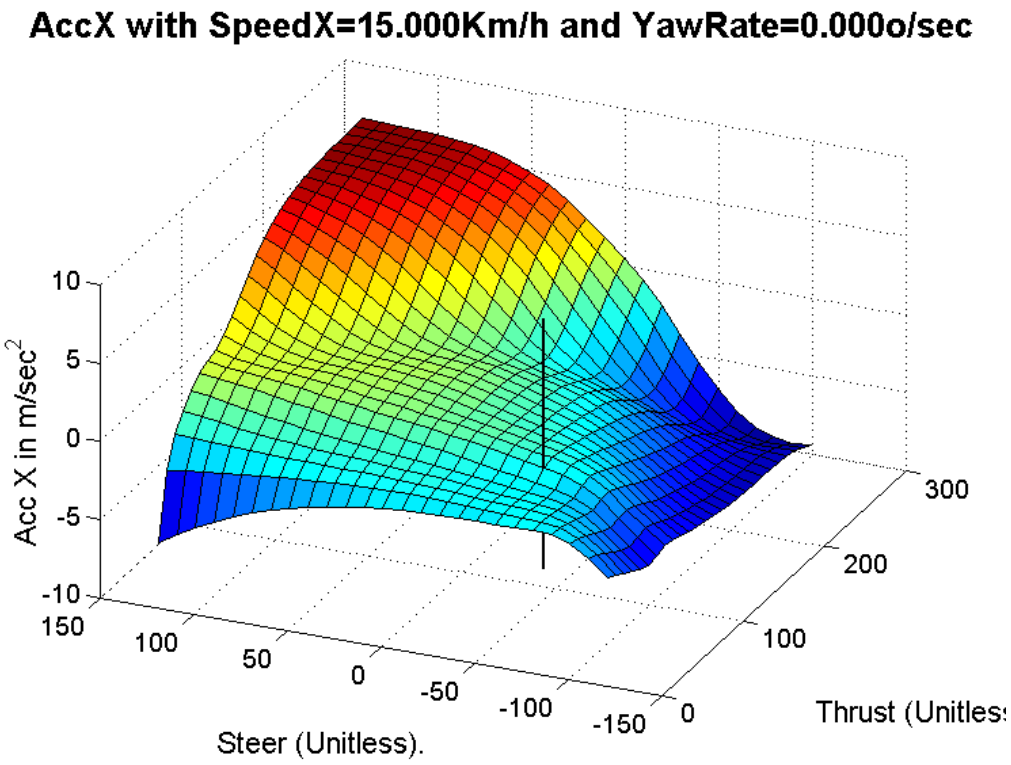


Figure 129: Surface of translational acceleration versus steering and thrust, with equilibrium point marked with a vertical line.

At the figure above the point closest to equilibrium is marked. This point is

» ***OpPoint{3,2}.VectDenorm***

(see above commands), which is vector:

$$\begin{bmatrix} \text{Thrust} \\ \text{Steer} \\ \text{SpeedX} \\ \text{YawRate} \end{bmatrix} = \begin{bmatrix} 60 \\ -61 \\ 15 \\ 0 \end{bmatrix}, \text{ resulting in } \begin{bmatrix} \text{AccX} \\ \text{Wdot} \end{bmatrix} = \begin{bmatrix} -0.1600447594037007 \\ -0.7096720909218274 \end{bmatrix}.$$

To find newral nets output use command:

» ***OpPoint{3,2}.NNetsOutDenorm***

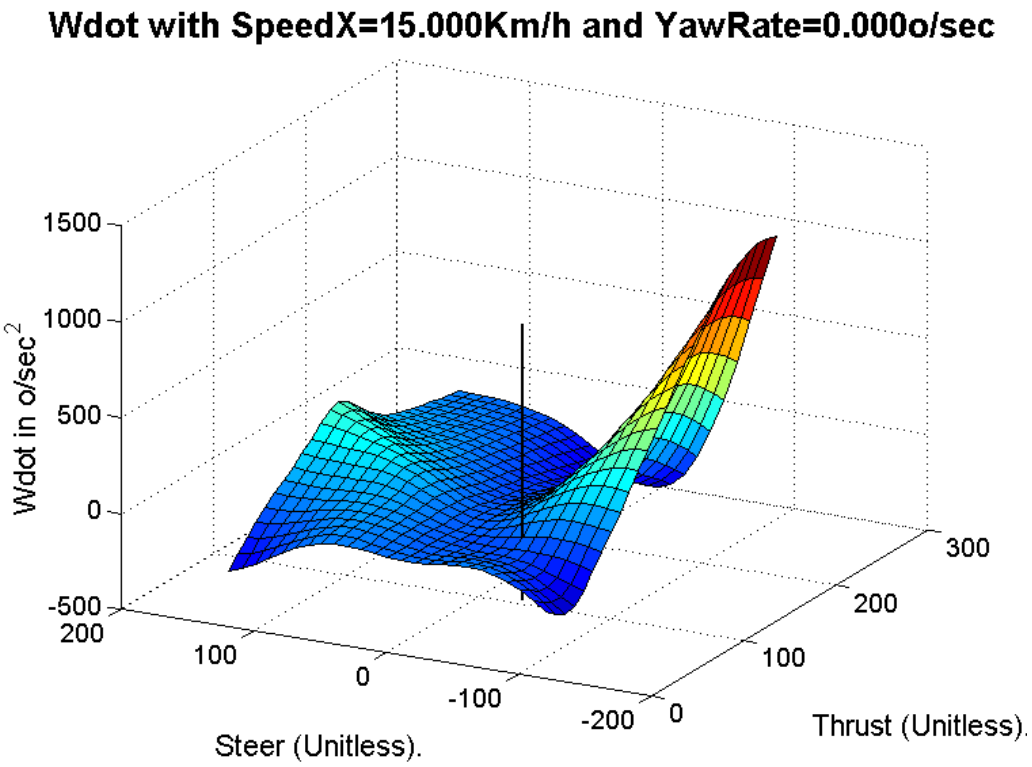


Figure 130: Surface of angular acceleration versus steering and thrust, with equilibrium point marked with a vertical line.

To get the linearized matrices of a standard linear state space equation namely:

$$\dot{\mathrm{x}} = \mathrm{Ax} + \mathrm{Bu}_{\mathrm{I}}$$

we apply the Jacobian

18.2Jacobian of Neural Networks Transfer Function

The Jacobian of the system is [23], [43], [45]:

$$\mathbf{J} = \mathbf{A} = \begin{bmatrix} \frac{\partial \alpha}{\partial \mathrm{u}} & \frac{\partial \alpha}{\partial \omega} \\ \frac{\partial \dot{\omega}}{\partial \mathrm{u}} & \frac{\partial \dot{\omega}}{\partial \omega} \end{bmatrix} \text{ and if we use normalizations: } \mathbf{J}_{\%} = \mathbf{A}_{\%} = \begin{bmatrix} \frac{\partial \alpha_{\%}}{\partial \mathrm{u}_{\%}} & \frac{\partial \alpha_{\%}}{\partial \omega_{\%}} \\ \frac{\partial \dot{\omega}_{\%}}{\partial \mathrm{u}_{\%}} & \frac{\partial \dot{\omega}_{\%}}{\partial \omega_{\%}} \end{bmatrix}. \quad (5.127)$$

For the calculation of the Jacobian two ways are going to be implemented:

- The analytical i.e. using the equations.
- The pure numerical calculation as will be shown below.

18.3Analytical Calculation

Reproducing equations for Neural Networks calculations we have

$$y_{1.3}=\left(\sum_{i=1}^{9}y_{i.2}w_{i1.3}\right)+b_{1.3}$$
$$y_{i.2}=\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right] \qquad (5.128)$$

which gives:

$$y_{1.3}=\left[\sum_{i=1}^{9}\left[\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]w_{i1.3}\right]\right]+b_{1.3}$$

Maintaining the x-notation for inputs, we should show the correspondence of symbols to the symbols used throughout this document:

$\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}y_T\\\delta_R\\u\\\omega\end{bmatrix}$, $y_{1.3}-->\alpha$ or $y_{1.3}-->\dot{\omega}$ depending on which network we are examining.

Thus

$$\mathbf{J}=\mathbf{A}=\begin{bmatrix}\frac{\partial\alpha}{\partial u} & \frac{\partial\alpha}{\partial\omega}\\ \frac{\partial\dot{\omega}}{\partial u} & \frac{\partial\dot{\omega}}{\partial\omega}\end{bmatrix}=\begin{bmatrix}\frac{\partial\alpha}{\partial x_3} & \frac{\partial\alpha}{\partial x_4}\\ \frac{\partial\dot{\omega}}{\partial x_3} & \frac{\partial\dot{\omega}}{\partial x_4}\end{bmatrix} \qquad (5.129)$$

$$\frac{\partial\alpha}{\partial u}=\frac{\partial\alpha}{\partial x_3}=\frac{\partial\left[\left[\sum_{i=1}^{9}\left[\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]w_{i1.3}\right]\right]+b_{1.3}\right]}{\partial x_3}=\sum_{i=1}^{9}\left\{w_{i1.3}\frac{\partial\,\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]}{\partial x_3}\right\}=$$

$$=\sum_{i=1}^{9}\left\{w_{i1.3}\bullet\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\bullet\frac{\partial\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]}{\partial x_3}\right\}=$$

$$=\sum_{i=1}^{9}\left\{w_{i1.3}\bullet\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\bullet w_{3i.2}\right\}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{3i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\}$$

(5.130)

using AccX network weights and biases.

Using the same calculations for the other cases, we have

$$\frac{\partial\alpha}{\partial\omega}=\frac{\partial\alpha}{\partial x_4}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{4i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\} \qquad (5.131)$$

using AccX network weights and biases.

$$\frac{\partial \dot{\omega}}{\partial u}=\frac{\partial \dot{\omega}}{\partial x_3}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{3i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\} \quad (5.132)$$

using Wdot network weights and biases.

$$\frac{\partial \dot{\omega}}{\partial \omega}=\frac{\partial \dot{\omega}}{\partial x_4}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{4i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\} \quad (5.133)$$

using Wdot network weights and biases.

18.4B-Matrix Calculation (Jacobian of Outputs in regard to Inputs)

The matrix that calculates the role of control inputs (thrust and steer) to the system output is matrix B formulated as follows:

$\mathbf{B}=\begin{bmatrix}\frac{\partial \alpha}{\partial y_T} & \frac{\partial \alpha}{\partial \delta_R}\\ \frac{\partial \dot{\omega}}{\partial y_T} & \frac{\partial \dot{\omega}}{\partial \delta_R}\end{bmatrix}$ and if we use normalizations: $\mathbf{B}_{\%}=\begin{bmatrix}\frac{\partial \alpha_{\%}}{\partial y_{\%}} & \frac{\partial \alpha_{\%}}{\partial \delta_{\%}}\\ \frac{\partial \dot{\omega}_{\%}}{\partial y_{\%}} & \frac{\partial \dot{\omega}_{\%}}{\partial \delta_{\%}}\end{bmatrix}$. The computation of matrix **B** is analogous to the computation of matrix **A**

18.5Analytical Calculation

Again here we reproduce equations for Neural Networks calculations as a starting point and progress by defining symbol analogy:

$$y_{1.3}=\left(\sum_{i=1}^{9}y_{i.2}w_{i1.3}\right)+b_{1.3}$$
$$y_{i.2}=\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right] \quad (5.134)$$

which gives:

$$y_{1.3}=\left[\sum_{i=1}^{9}\left[\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]w_{i1.3}\right]\right]+b_{1.3}$$

Maintaining the x-notation for inputs, we should show the correspondence of symbols to the symbols used throughout this document:

$\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}y_T\\\delta_R\\u\\\omega\end{bmatrix}$, $y_{1.3}\text{--->}\alpha$ or $y_{1.3}\text{--->}\dot{\omega}$ depending on which network we are examining.

Thus

$$\mathbf{B}=\begin{bmatrix}\frac{\partial\alpha}{\partial y_T} & \frac{\partial\alpha}{\partial\delta_R}\\ \frac{\partial\dot{\omega}}{\partial y_T} & \frac{\partial\dot{\omega}}{\partial\delta_R}\end{bmatrix}=\begin{bmatrix}\frac{\partial\alpha}{\partial x_1} & \frac{\partial\alpha}{\partial x_2}\\ \frac{\partial\dot{\omega}}{\partial x_1} & \frac{\partial\dot{\omega}}{\partial x_2}\end{bmatrix} \tag{5.135}$$

$$\frac{\partial\alpha}{\partial y_T}=\frac{\partial\alpha}{\partial x_1}=\frac{\partial\left[\left[\sum_{i=1}^{9}\left[\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]w_{i1.3}\right]\right]+b_{1.3}\right]}{\partial x_1}=\sum_{i=1}^{9}\left\{w_{i1.3}\frac{\partial\,\text{Tansig}\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]}{\partial x_1}\right\}=$$

$$=\sum_{i=1}^{9}\left\{w_{i1.3}\bullet\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\bullet\frac{\partial\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]}{\partial x_1}\right\}=$$

$$=\sum_{i=1}^{9}\left\{w_{i1.3}\bullet\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\bullet w_{3i.2}\right\}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{3i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\}$$

(5.136)

using AccX network weights and biases.

Using the same calculations for the other cases, we have

$$\frac{\partial\alpha}{\partial\omega}=\frac{\partial\alpha}{\partial x_4}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{4i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\} \tag{5.137}$$

using AccX network weights and biases.

$$\frac{\partial\dot{\omega}}{\partial u}=\frac{\partial\dot{\omega}}{\partial x_3}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{3i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\} \tag{5.138}$$

using Wdot network weights and biases.

$$\frac{\partial\dot{\omega}}{\partial\omega}=\frac{\partial\dot{\omega}}{\partial x_4}=\sum_{i=1}^{9}\left\{w_{i1.3}w_{4i.2}\frac{4\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]}{\left[1+\exp\left[-2\left[\left(\sum_{k=1}^{4}x_k w_{ki.2}\right)+b_{i.2}\right]\right]\right]^2}\right\} \tag{5.139}$$

using Wdot network weights and biases.

18.6Applying the Principles to Grid

To get the full set of linearized systems for every selected pair of [u ω]', the following procedure was followed:

- Division of u into equally spaced set of points, one every 1 km/h. We applied this division to velocities belonging to [0 30] km/h resulting into 31 points.

- Division of ω into equally spaced set of points, one evey 5°/sec to [-50 50] °/sec, resulting in 21 points.
- Linearization at every one of the 31 x 21 points (651 points total).
- Examine the controllability of each of the resulting systems.
- Examine stability of each of the resulting systems.
- Examine the limits of the resulting matrices for the linearized systems.

The Matlab implementation for this procedure uses functions:

- ***CreateLinearizedModels*** which generates the sets of models (more on these commands at the paragraph explaining Matlab Code).
- ***AnalyzeLinModelsSet*** which analyzes the set of models in regard to its controllability and limits.
- ***AnalyzeStabilityAndVectLims*** which analyzes the set of models mainly in regard to the stability of each resulting model.

By applying the following commands set:

```
cd C:\PHD\Matlab_180910\StatLin
%cd
C:\Work\PHD\Autopilot_IV\Software\Matlab_180910\NewralNetworksTransferFunction\MatlabCode

addpath ../
addpath ../ModelsRevisionAndControl/
addpath ../NewralNetworksTransferFunction/MatlabCode/

AccXNNFilename =
'../AccXNeuralFolder/AccXNeural_20140414_1051/AccXNeural_20140414_1051.mat';

WdotNNFilename =
'../WdotNeural/WdotNeural_20140414_1058/WdotNeural_20140414_1058.mat';

SpeedValues.Vector = [0:1:30];
%SpeedValues.Vector = [0:2:4];
SpeedValues.Type = 'denormalized';

WValues.Vector = [-50:5:50];
%WValues.Vector = [-5:5:5];
WValues.Type = 'denormalized';

Thresholds.AccX = 0.001;
Thresholds.YawAcc = 0.0001;
```

```
str_1 = 'Create a linearized models matrix with all possible speed values with 1
kmph step\n';
str_2 = 'and all possible W values with 5 degpsec step \n';
SetDescription = strcat(str_1, str_2);

Resolution.Thrust = 1;
Resolution.Steer = 1;

OutputFolderName = 'ModelSet';

OutputFolderPath = 'C:/PHD/Matlab_180910/StatLin/LinearizedModelSets';

LinearizedModelsCellArray = CreateLinearizedModels( AccXNNFilename,
WdotNNFilename, SpeedValues, WValues, Thresholds, SetDescription, Resolution,
OutputFolderName, OutputFolderPath );
```

We get the set of linearized models. Also by applying the following commands set:

```
LinModSetFilename =
'C:/PHD/Matlab_180910/StatLin/LinearizedModelSets/ModelSet_20220107_2320/
LinearModelSet.mat';
OutputFolderPath =
'C:/PHD/Matlab_180910/StatLin/LinearizedModelSets/ModelSet_20220107_2320/
';
LinModelsCellResults = AnalyzeLinModelsSet( LinModSetFilename,
OutputFolderPath )
```

and

```
cd C:/PHD/Matlab_180910/StatLin/
addpath ..\
LinModSetFilename =
'C:/PHD/Matlab_180910/StatLin/LinearizedModelSets/ModelSet_20200822_0144/
LinearModelSet.mat';
OutputFolderPath =
'C:/PHD/Matlab_180910/StatLin/LinearizedModelSets/ModelSet_20200822_0144/
';
MarginalStabilityLim = 1e-3;
LinModelsCellStabilityResults = AnalyzeStabilityAndVectLims( LinModSetFilename,
OutputFolderPath, MarginalStabilityLim )
```

we get the following results:

- All resulting systems are controllable.
- We did not search for observability, since the states [u ω] are measured directly.
- All resulting systems were either close to marginal stability or unstable.

The following surfaces are read as follows:

- If a system is stable then it has a value of 1.
- If a system is stable and it has one u = 0 or ω = 0 it has a value of 2.
- If a system is marginally stable then it has a value of 0.
- If a system is unstable then it has a value of -1.
- If a system is unstable and it has one u = 0 or ω = 0 it has a value of -2.

By applying a multiplicative factor of 2 to every point that has one of u or ω zero, we distinguish these from the rest, to use them as an optical reference.

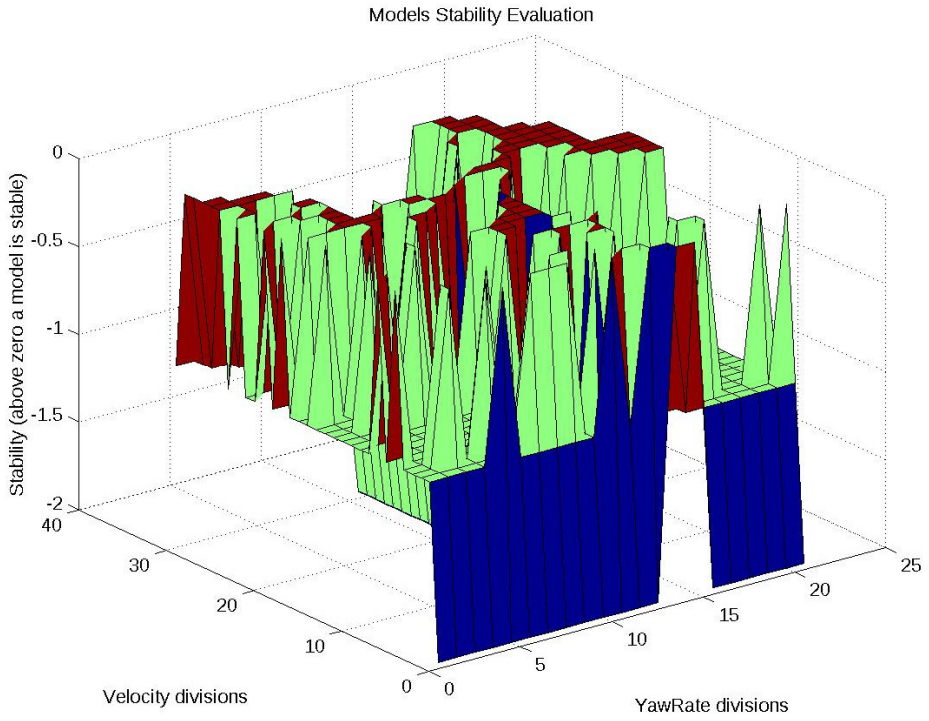


Figure 131: Optical representation of the stability of each system at the grid of linearization.

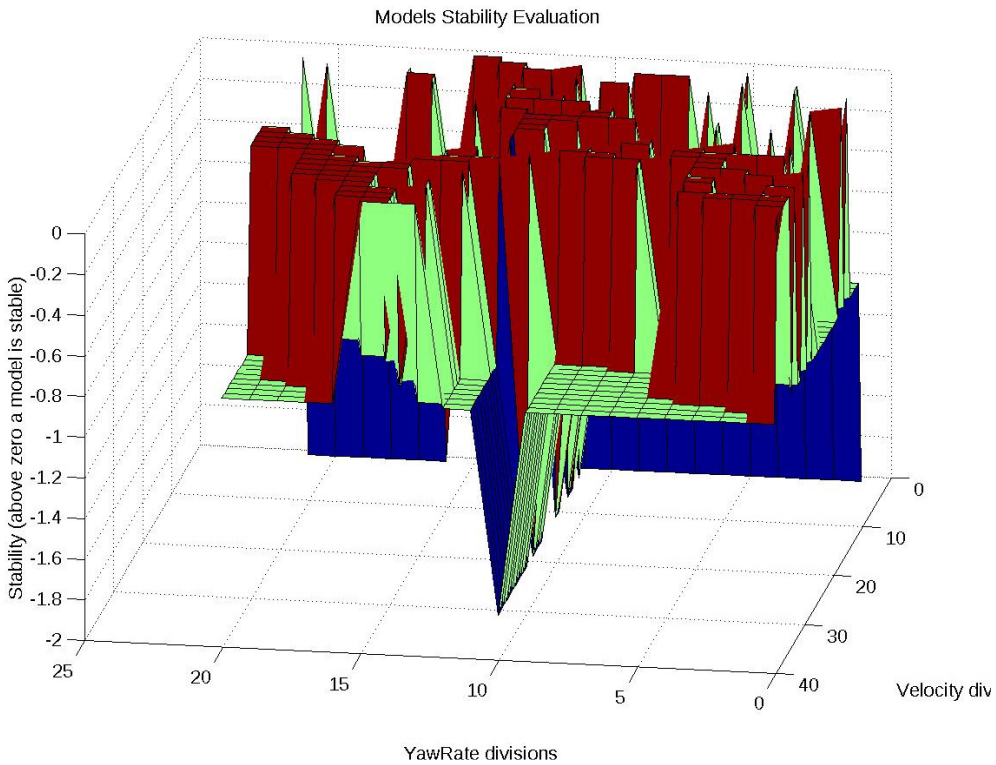


Figure 132: Optical representation of the stability of each system at the grid of linearization, different angle from previous.

19 APPENDIX 7: Alternative Approach for Full Vessel Model

In Figure 133 the block diagram of the full model we used is shown. Besides the neural networks as trained, we have added some more blocks/modules to aid in the accuracy of the total output. These blocks are (as shown from left to right):

- DC Canceller
- Integrator
- Clipper
- Leakage

The description of each module is given in the following subsections:

19.1 DC Caneller

No matter how accurate the modeling from neural network is, there is always an offset at low frequencies, which if passed from an integrator, gives a constantly increasing error between their output and measured results. To compensate for this, we added a digital DC canceller at their output which uses the following equation:

$$y(n) = x(n) - x(n-1) + ay(n-1) \qquad (5.140)$$

Where:

y is the DC Canceller output

x is the DC Canceller input

a is a gain that changes the functionality of DC Canceller. The smallest this parameter is, the fastest the DC canellation is. The closer to 1 the slower DC cancellation is performed, with $a = 1$meaning no DC cancellation.

n is the number of the current sampling instant.

Parameter a was a constant that had to be calculated.

19.2 Integrator

Trapezoidal integrator. For more details refer to paragraph “trapz and cumptrapz” section.

19.3 Clipper

Clipper is a module that does not allow the output of the model to exceed the measured maximum values. This module had nothing variable.

19.4 Leakage

Leakage is a module that creates numeric loss towards zero each sampling instant. This module had variable leage that had to be determined to result in better accuracy of the model.

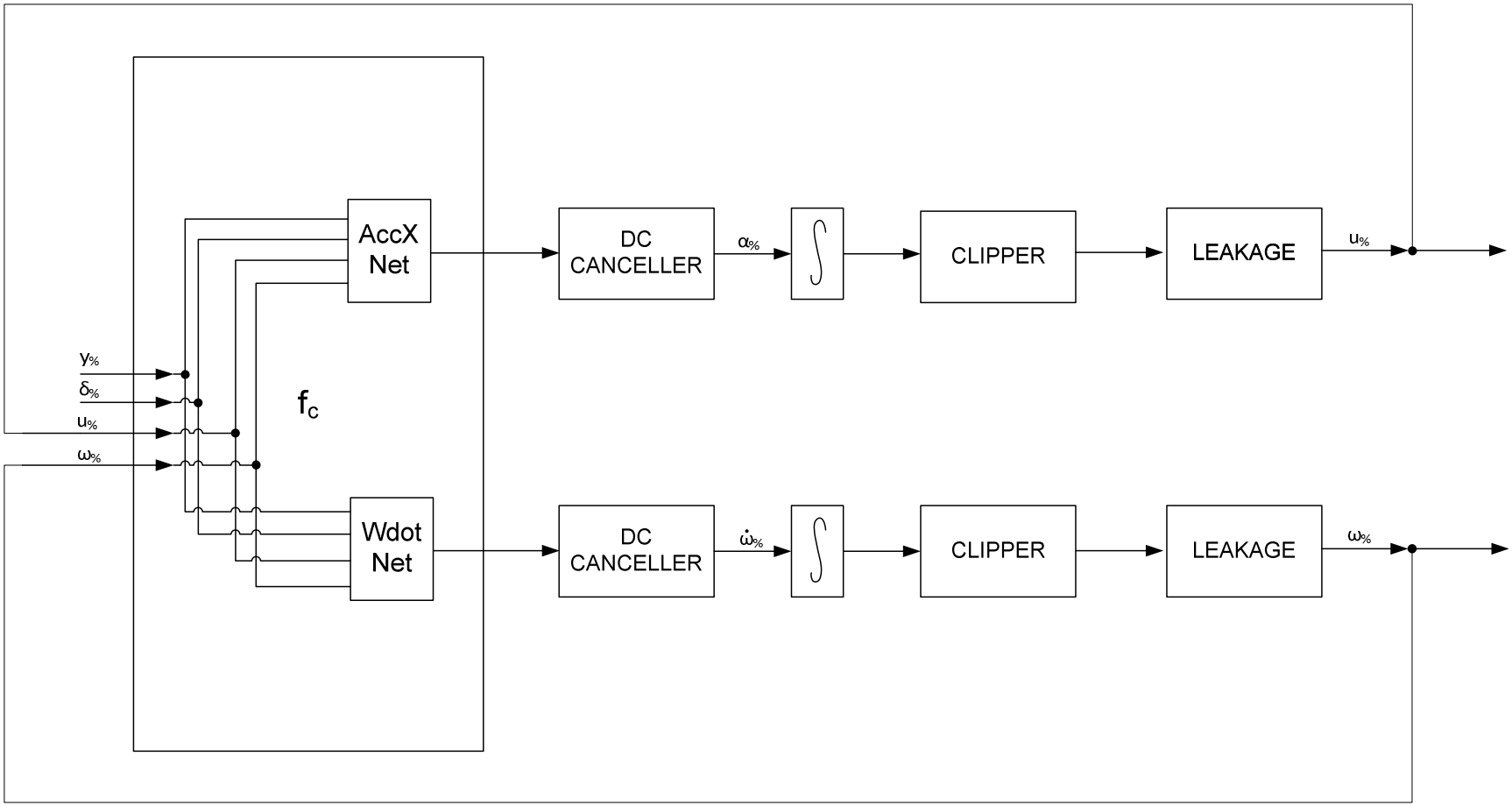


Figure 133: Full vessel model

19.5Optimal Module Parameters Calculation

The next step after deciding on the model of Figure 133 was to determine the values of the unknown parameters i.e. DC caneller gains and leakage values for both paths. The method we used for this purpose was Steepest Descent applied on the vector:

```
OptVector(1) = AccXDCCanceller.Gain;
OptVector(2) = WdotDCCanceller.Gain;
OptVector(3) = AccXIntegrLeakage.Leakage;
OptVector(4) = AccXIntegrLeakage.R;
OptVector(5) = WdotIntegrLeakage.Leakage;
OptVector(6) = WdotIntegrLeakage.R;
OptVector = OptVector(:);
```

20 APPENDIX 8: Useful Matlab Commands

20.1Norm

» help norm

```
 NORM   Matrix or vector norm.
   For matrices...
     NORM(X) is the largest singular value of X, max(svd(X)).
     NORM(X,2) is the same as NORM(X).
     NORM(X,1) is the 1-norm of X, the largest column sum,
                 = max(sum(abs((X)))).
     NORM(X,inf) is the infinity norm of X, the largest row sum,
                 = max(sum(abs((X')))).
     NORM(X,'fro') is the Frobenius norm, sqrt(sum(diag(X'*X))).
     NORM(X,P) is available for matrix X only if P is 1, 2, inf or 'fro'.

   For vectors...
     NORM(V,P) = sum(abs(V).^P)^(1/P).
     NORM(V) = norm(V,2).
     NORM(V,inf) = max(abs(V)).
     NORM(V,-inf) = min(abs(V)).

   See also COND, CONDEST, NORMEST.

 Overloaded methods
   help lti/norm.m
   help ss/norm.m
   help frd/norm.m
```

A rather common use of norm command is in order to find the magnitude of a vector or a complex number. For example :

» norm([3 4])

ans =

5

» sqrt(3^2 + 4^2)

ans =

5

20.2angle

ANGLE Phase angle.
   ANGLE(H) returns the phase angles, in radians, of a matrix with
   complex elements.

   See also ABS, UNWRAP.

Example:
K» angle(2+i*2)

ans =

   7.853981633974483e-001

K» pi/4

ans =

   7.853981633974483e-001

K» rad2deg(ans)

ans =

   45

K» angle(-2-i*2)

ans =

  -2.356194490192345e+000

K» rad2deg(ans)

ans =

 -135

K» 3*pi/4

```
ans =

   2.356194490192345e+000
```

20.3rad2deg

RAD2DEG Converts angles from radians to degrees

deg = RAD2DEG(rad) converts angles from radians to degrees.

See also DEG2RAD, RAD2DMS, ANGLEDIM, ANGL2STR

K» help deg2rad

20.4deg2rad

DEG2RAD Converts angles from degrees to radians

rad = DEG2RAD(deg) converts angles from degrees to radians.

See also RAD2DEG, DEG2DMS, ANGLEDIM, ANGL2STR

20.5Interp1

» help interp1

INTERP1 1-D interpolation (table lookup).
YI = INTERP1(X,Y,XI) interpolates to find YI, the values of
the underlying function Y at the points in the vector XI.
The vector X specifies the points at which the data Y is
given. If Y is a matrix, then the interpolation is performed
for each column of Y and YI will be length(XI)-by-size(Y,2).
Out of range values are returned as NaN.

YI = INTERP1(Y,XI) assumes X = 1:N, where N is the length(Y)
for vector Y or SIZE(Y,1) for matrix Y.

Interpolation is the same operation as "table lookup". Described in
"table lookup" terms, the "table" is [X,Y] and INTERP1 "looks-up"
the elements of XI in X, and, based upon their location, returns
values YI interpolated within the elements of Y.

YI = INTERP1(X,Y,XI,'method') specifies alternate methods.
The default is linear interpolation. Available methods are:

'nearest' - nearest neighbor interpolation
'linear' - linear interpolation
'spline' - cubic spline interpolation
'cubic' - cubic interpolation

All the interpolation methods require that X be monotonic. X can be
non-uniformly spaced. For faster interpolation when X is equally
spaced and monotonic, use the methods '*linear', '*cubic', '*nearest'.
For faster linear interpolation when X is non-uniformly spaced
see INTERP1Q.

**For example, generate a coarse sine curve and interpolate over a
finer abscissa:**

```
x = 0:10; y = sin(x); xi = 0:.25:10;
yi = interp1(x,y,xi); plot(x,y,'o',xi,yi)
```

See also INTERP1Q, INTERPFT, SPLINE, INTERP2, INTERP3, INTERPN.

20.6Network/sim

» help network/sim

SIM Simulate a neural network.

Syntax

[Y,Pf,Af] = sim(net,P,Pi,Ai)
[Y,Pf,Af] = sim(net,{Q TS},Pi,Ai)
[Y,Pf,Af] = sim(net,Q,Pi,Ai)

Description

SIM simulates neural networks.

[Y,Pf,Af] = SIM(net,P,Pi,Ai) takes,
NET - Network.
P - Network inputs.
Pi - Initial input delay conditions, default = zeros.
Ai - Initial layer delay conditions, default = zeros.
and returns:
Y - Network outputs.
Pf - Final input delay conditions.
Af - Final layer delay conditions.

Note that arguments Pi, Ai, Pf, and Af are optional and
need only be used for networks that have input or layer delays.

SIM's signal arguments can have two formats: cell array or matrix.

The cell array format is easiest to describe. It is most
convenient for networks with multiple inputs and outputs,
and allows sequences of inputs to be presented:
P - NixTS cell array, each element P{i,ts} is an RixQ matrix.
Pi - NixID cell array, each element Pi{i,k} is an RixQ matrix.
Ai - NlxLD cell array, each element Ai{i,k} is an SixQ matrix.
Y - NOxTS cell array, each element Y{i,ts} is a UixQ matrix.
Pf - NixID cell array, each element Pf{i,k} is an RixQ matrix.
Af - NlxLD cell array, each element Af{i,k} is an SixQ matrix.
Where:
Ni = net.numInputs
Nl = net.numLayers,
No = net.numOutputs
ID = net.numInputDelays
LD = net.numLayerDelays
TS = number of time steps
Q = batch size
Ri = net.inputs{i}.size
Si = net.layers{i}.size
Ui = net.outputs{i}.size

The columns of Pi, Pf, Ai, and Af are ordered from oldest delay
condition to most recent:
Pi{i,k} = input i at time ts=k-ID.
Pf{i,k} = input i at time ts=TS+k-ID.
Ai{i,k} = layer output i at time ts=k-LD.
Af{i,k} = layer output i at time ts=TS+k-LD.

The matrix format can be used if only one time step is to be simulated (TS = 1). It is convenient for networks with only one input and output, but can also be used with networks that have more.

Each matrix argument is found by storing the elements of the corresponding cell array argument into a single matrix:

P - (sum of Ri)xQ matrix
Pi - (sum of Ri)x(ID*Q) matrix.
Ai - (sum of Si)x(LD*Q) matrix.
Y - (sum of Ui)xQ matrix.
Pf - (sum of Ri)x(ID*Q) matrix.
Af - (sum of Si)x(LD*Q) matrix.

[Y,Pf,Af] = SIM(net,{Q TS},Pi,Ai) is used for networks which do not have an input, such as Hopfield networks when cell array notation is used.

[Y,Pf,Af] = SIM(net,Q,Pi,Ai) is used for networks which do not have an input, such as Hopfield networks when matrix notation is used.

Examples

Here NEWP is used to create a perceptron layer with a 2-element input (with ranges of [0 1]), and a single neuron.

```
net = newp([0 1;0 1],1);
```

Here the perceptron is simulated for an individual vector, a batch of 3 vectors, and a sequence of 3 vectors.

```
p1 = [.2; .9]; a1 = sim(net,p1)
p2 = [.2 .5 .1; .9 .3 .7]; a2 = sim(net,p2)
p3 = {[.2; .9] [.5; .3] [.1; .7]}; a3 = sim(net,p3)
```

Here NEWLIND is used to create a linear layer with a 3-element input, 2 neurons.

```
net = newlin([0 2;0 2;0 2],2,[0 1]);
```

Here the linear layer is simulated with a sequence of 2 input vectors using the default initial input delay conditions (all zeros).

```
p1 = {[2; 0.5; 1] [1; 1.2; 0.1]};
[y1,pf] = sim(net,p1)
```

Here the layer is simulated for 3 more vectors using the previous final input delay conditions as the new initial delay conditions.

```
p2 = {[0.5; 0.6; 1.8] [1.3; 1.6; 1.1] [0.2; 0.1; 0]};
[y2,pf] = sim(net,p2,pf)
```

Here NEWELM is used to create an Elman network with a 1-element

input, and a layer 1 with 3 TANSIG neurons followed by a layer 2
with 2 PURELIN neurons. Because it is an Elman network it has a
tap delay line with a delay of 1 going from layer 1 to layer 1.

net = newelm([0 1],[3 2],{'tansig','purelin'});

Here the Elman network is simulated for a sequence of 3 values
using default initial delay conditions.

p1 = {0.2 0.7 0.1};
[y1,pf,af] = sim(net,p1)

Here the network is simulated for 4 more values, using the previous
final delay conditions as the new initial delay conditions.

p2 = {0.1 0.9 0.8 0.4};
[y2,pf,af] = sim(net,p2,pf,af)

Algorithm

SIM uses these properties to simulate a network NET.

NET.numInputs, NET.numLayers
NET.outputConnect, NET.biasConnect
NET.inputConnect, NET.layerConnect

These properties determine the network's weight and bias values,
and the number of delays associated with each weight:

NET.inputWeights{i,j}.value
NET.layerWeights{i,j}.value
NET.layers{i}.value
NET.inputWeights{i,j}.delays
NET.layerWeights{i,j}.delays

These function properties indicate how SIM applies weight and
bias values to inputs to get each layer's output:

NET.inputWeights{i,j}.weightFcn
NET.layerWeights{i,j}.weightFcn
NET.layers{i}.netInputFcn
NET.layers{i}.transferFcn

See Chapter 2 for more information on network simulation.

See also INIT, ADAPT, TRAIN

20.7 Diff

» help diff

DIFF Difference and approximate derivative.
DIFF(X), for a vector X, is [X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)].
DIFF(X), for a matrix X, is the matrix of column differences,

```
    [X(2:n,:) - X(1:n-1,:)].
  DIFF(X), for an N-D array X, is the difference along the first
    non-singleton dimension of X.
  DIFF(X,N) is the N-th order difference along the first non-singleton
    dimension (denote it by DIM). If N >= size(X,DIM), DIFF takes
    successive differences along the next non-singleton dimension.
  DIFF(X,N,DIM) is the Nth difference function along dimension DIM.
    If N >= size(X,DIM), DIFF returns an empty array.

  Examples:
    h = .001; x = 0:h:pi;
    diff(sin(x.^2))/h is an approximation to 2*cos(x.^2).*x
    diff((1:10).^2) is 3:2:19

    If X = [3 7 5
            0 9 2]
    then diff(X,1,1) is [-3 2 -3], diff(X,1,2) is [4 -2
                                                   9 -7],
    diff(X,2,2) is the 2nd order difference along the dimension 2, and
    diff(X,3,2) is the empty matrix.

  See also GRADIENT, SUM, PROD.

Overloaded methods
  help sym/diff.m
  help char/diff.m
```

20.8Fieldnames

» help fieldnames

FIELDNAMES Get structure field names.
NAMES = FIELDNAMES(S) returns the structure field names associated
with the structure S as a cell array of strings.

See also GETFIELD, SETFIELD.

20.9Creating a Colored Surface in Matlab: surf command

» help surf

SURF 3-D colored surface.
SURF(X,Y,Z,C) plots the colored parametric surface defined by
four matrix arguments. The view point is specified by VIEW.
The axis labels are determined by the range of X, Y and Z,
or by the current setting of AXIS. The color scaling is determined
by the range of C, or by the current setting of CAXIS. The scaled
color values are used as indices into the current COLORMAP.
The shading model is set by SHADING.

SURF(X,Y,Z) uses C = Z, so color is proportional to surface height.

SURF(x,y,Z) and SURF(x,y,Z,C), with two vector arguments replacing
the first two matrix arguments, must have length(x) = n and
length(y) = m where [m,n] = size(Z). In this case, the vertices
of the surface patches are the triples (x(j), y(i), Z(i,j)).
Note that x corresponds to the columns of Z and y corresponds to
the rows.

SURF(Z) and SURF(Z,C) use x = 1:n and y = 1:m. In this case,
the height, Z, is a single-valued function, defined over a
geometrically rectangular grid.

SURF(...,'PropertyName',PropertyValue,...) sets the value of the
specified surface property. Multiple property values can be set
with a single statement.

SURF returns a handle to a SURFACE object.

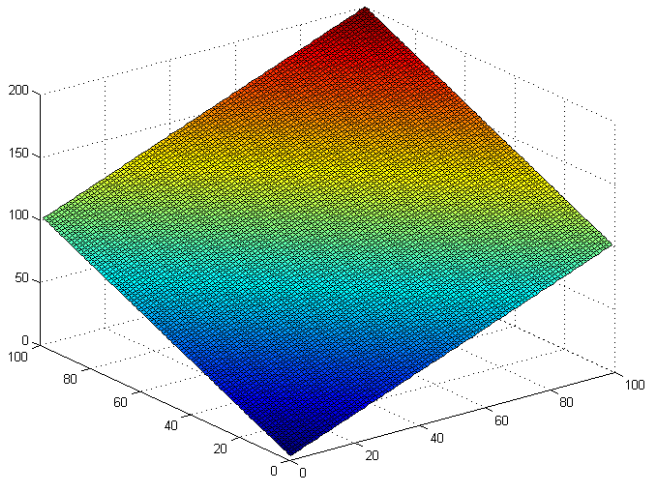
AXIS, CAXIS, COLORMAP, HOLD, SHADING and VIEW set figure, axes, and
surface properties which affect the display of the surface.

See also SURFC, SURFL, MESH, SHADING.

My example:

```
x = 1:100;
y = 1:100;
for x_index = 1:length(x),
        for y_index = 1:length(y),
                z(x_index, y_index) = x(x_index) + y(y_index);
        end
end

surf(x,y,z)
```



20.10 Figures and Properties

Every Matlab figure is an object with properties. To get the property names available for changing we can use command ***get(***<figure_handle>***)***. The current figure handle can be acquired by using command ***gcf*** (get current figure) or by using a variable assigned by the ***figure*** command during figure creation. To change a property of a figure command ***set(***<object_handle>, <property1>, <value1>,…***)*** is used.
For papers and documentation output we usually use the following figure changes:

- Title letter size 24.
- Axes text size 18.
- Grid on.
- Line width 5.

The easiest way to access these properties is to use variables assigned by every command given to get the figure output needed.

```
figH = figure;
plotLinesH = plot(TimeVector,EquNNetAccX);
titleH = title('AccX net Test1');
xlabelH = xlabel('Time in sec');
ylabelH = ylabel('Acc X in m/sec^2');
set(plotLinesH, 'LineWidth', 5);
set(titleH, 'FontSize', 24, 'FontWeight', 'bold');
set(xlabelH, 'FontSize', 18);
set(ylabelH, 'FontSize', 18);
axesH = get(figH, 'CurrentAxes');
```
or
```
axesH = gca
set(axesH, 'FontSize', 18);
grid on;
```

The lineseries will (generally) be a child of the axes. So if you have a have a handle to the axes (or using gca, if it is the current axes), you can query its Children property:

```
>> a = axes;
>> h1 = plot(1:10);
>> h2 = get(a, 'Children');
>> h1 == h2
ans =
     1
```

If you have a very complicated scene (perhaps there are multiple axes involved or maybe hggroups and hgtransforms), it may be easier to use the findobj command:

```
>> h1 = plot(1:10);
>> h2 = findobj('Type', 'line');
>> h1 == h2
ans =
```

1

20.11 lsqnonlin

» help lsqnonlin

LSQNONLIN Solves non-linear least squares problems.
LSQNONLIN solves problems of the form:
min sum {FUN(X).^2} where X and the values returned by FUN can be
x vectors or matrices.

X=LSQNONLIN(FUN,X0) starts at the matrix X0 and finds a minimum to the sum of squares of the functions described in FUN. FUN is usually an M-file which returns a vector of objective functions: F=FUN(X). NOTE: FUN should return FUN(X) and not the sum-of-squares sum(FUN(X).^2)). (FUN(X) is summed and squared implicitly in the algorithm.) See below for more options for FUN.

X=LSQNONLIN(FUN,X0,LB,UB) defines a set of lower and upper bounds on the design variables, X, so that the solution is in the range LB <= X <= UB. Use empty matrices for LB and UB if no bounds exist. Set LB(i) = -inf if X(i) is unbounded below; set UB(i) = inf if X(i) is unbounded above.

X=LSQNONLIN(FUN,X0,LB,UB,OPTIONS) minimizes with the default optimization parameters replaced by values in the structure OPTIONS, an argument created with the OPTIMSET function. See OPTIMSET for details. Used options are Display, TolX, TolFun, DerivativeCheck, Diagnostics, Jacobian, JacobPattern, LineSearchType, LevenbergMarquardt, MaxFunEvals, MaxIter, DiffMinChange and DiffMaxChange, LargeScale, MaxPCGIter, PrecondBandWidth, TolPCG, TypicalX. Use the Jacobian option to specify that FUN may be called with two output arguments where the second, J, is the Jacobian matrix: [F,J] = feval(FUN,X). If FUN returns a vector (matrix) of m components when X has length n, then J is an m-by-n matrix where J(i,j) is the partial derivative of F(i) with respect to x(j). (Note that the Jacobian J is the transpose of the gradient of F.)

X=LSQNONLIN(FUN,X0,LB,UB,OPTIONS,P1,P2,..) passes the problem-dependent parameters P1,P2,... directly to the functions FUN: FUN(X,P1,P2,...). Pass an empty matrix for OPTIONS to use the default values.

[X,RESNORM]=LSQNONLIN(FUN,X0,...) returns
the value of the squared 2-norm of the residual at X: sum(FUN(X).^2).

[X,RESNORM,RESIDUAL]=LSQNONLIN(FUN,X0,...) returns the value of the
residual at the solution X: RESIDUAL = FUN(X).

[X,RESNORM,RESIDUAL,EXITFLAG]=LSQNONLIN(FUN,X0,...) returns a string
EXITFLAG that describes the exit condition of LSQNONLIN.
If EXITFLAG is:
> 0 then LSQNONLIN converged to a solution X.
0 then the maximum number of function evaluations was reached.
< 0 then LSQNONLIN did not converge to a solution.

[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT]=LSQNONLIN(FUN,X0,...) returns a
structure OUTPUT with the number of iterations taken in OUTPUT.iterations,
the number of function evaluations in OUTPUT.funcCount, the algorithm used
in OUTPUT.algorithm, the number of CG iterations (if used) in
OUTPUT.cgiterations,
and the first-order optimality (if used) in OUTPUT.firstorderopt.

[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT,LAMBDA]=LSQNONLIN(FUN,X0,...)
returns
the set of Lagrangian multipliers, LAMBDA, at the solution: LAMBDA.lower
for LB and LAMBDA.upper for UB.

[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT,LAMBDA,JACOBIAN]=LSQNONLIN(FUN,X0
,...)
returns the Jacobian of FUN at X.

20.12 dtansig

» help dtansig

DTANSIG Hyperbolic tangent sigmoid transfer derivative function.

Syntax

dA_dN = dtansig(N,A)

Description

DTANSIG is the derivative function for TANSIG.

DTANSIG(N,A) takes two arguments,
N - SxQ net input.
A - SxQ output.
and returns the SxQ derivative dA/dN.

Examples

Here we define the net input N for a layer of 3 TANSIG neurons.

N = [0.1; 0.8; -0.7];

We calculate the layer's output A with TANSIG and then the derivative of A with respect to N.

A = tansig(N)
dA_dN = dtansig(N,A)

Algorithm

The derivative of TANSIG is calculated as follows:

d = 1-a^2

See also TANSIG, LOGSIG, DLOGSIG.

20.13 dlogsig

» help dlogsig

DLOGSIG Log sigmoid transfer derivative function.

Syntax

dA_dN = dlogsig(N,A)

Description

DLOGSIG is the derivative function for LOGSIG.

DLOGSIG(N,A) takes two arguments,
N - SxQ net input.
A - SxQ output.
and returns the SxQ derivative dA/dN.

Examples

Here we define the net input N for a layer of 3 TANSIG neurons.

N = [0.1; 0.8; -0.7];

We calculate the layer's output A with LOGSIG and then the derivative of A with respect to N.

A = logsig(N)
dA_dN = dlogsig(N,A)

Algorithm

The derivative of LOGSIG is calculated as follows:

d = a * (1 - a)

See also LOGSIG, TANSIG, DTANSIG.

20.14 repmat

» help repmat

REPMAT Replicate and tile an array.
B = REPMAT(A,M,N) replicates and tiles the matrix A to produce the
M-by-N block matrix B.

B = REPMAT(A,[M N]) produces the same thing.

B = REPMAT(A,[M N P ...]) tiles the array A to produce a
M-by-N-by-P-by-... block array. A can be N-D.

REPMAT(A,M,N) when A is a scalar is commonly used to produce
an M-by-N matrix filled with A's value. This can be much faster
than A*ONES(M,N) when M and/or N are large.

Example:
repmat(magic(2),2,3)
repmat(NaN,2,3)

See also MESHGRID.

20.15 intersect

» help intersect

INTERSECT Set intersection.
INTERSECT(A,B) when A and B are vectors returns the values common to both A and B. The result will be sorted. A and B can be cell arrays of strings.

INTERSECT(A,B,'rows') when A are B are matrices with the same number of columns returns the rows common to both A and B.

[C,IA,IB] = INTERSECT(...) also returns index vectors IA and IB such that C = A(IA) and C = B(IB) (or C = A(IA,:) and C = B(IB,:)).

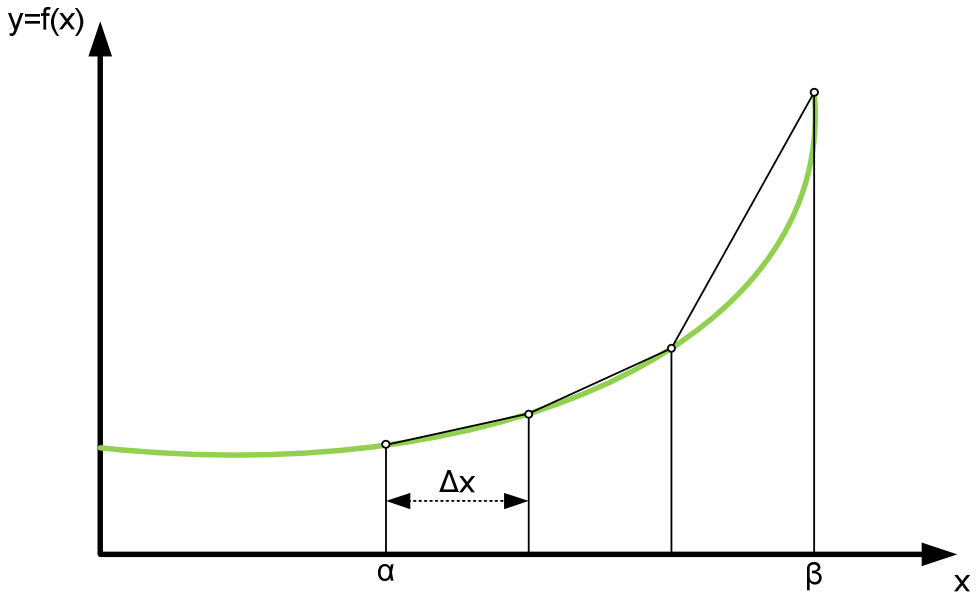
See also UNIQUE, UNION, SETDIFF, SETXOR, ISMEMBER.

Overloaded methods
help cell/intersect.m

Loghis Note: Be careful with interest, because it shows only one of the indexes that these sets are common. If you need all of them you must repetitively apply the command.. It also does not function correctly with the presence of 'NaN' elements in the sets.

20.16 trapz and cumptrapz

These two commands perform trapezoidal integration approximation. Trapezoidal integration is an approximation method to integral. It is as follows:



$$\int_{\alpha}^{\beta} f(x)dx \simeq \sum_{n=0}^{N-1}\left\{\frac{1}{2}\Delta x\left[f(x_n)+f(x_{n+1})\right]\right\} =$$

$$\frac{\Delta x}{2}\left(\left(f(x_0)+f(x_1)\right)+\left(f(x_1)+f(x_2)\right)+\ldots+\left(f(x_{N-1})+f(x_N)\right)\right) =$$

$$\frac{\Delta x}{2}\left(f(x_0)+2f(x_1)+\ldots+2f(x_{N-1})+f(x_N)\right)$$

$$\Delta x = \frac{\beta-\alpha}{N}$$

The formula is such because every trapezoid is a parallelogram plus a triangle, so its area is:

$$f(x_n)\Delta x+\frac{1}{2}(f(x_{n+1})-f(x_n))\Delta x = \frac{\Delta x}{2}(f(x_n)+f(x_{n+1}))$$

The difference between the two commands is that trapz **reduces** the size of the dimension it operates on to 1, and returns only the final integration value. **cumtrapz** also returns the intermediate integration values, preserving the size of the dimension it operates on. Note that command incorporates the ½ factor, thus **the command must not be multiplied by 0.5** to get the correct value.

See https://www.mathworks.com/help/matlab/ref/trapz.html#bua4lsr

Example of usage:

In this example we show how to calculate the integral of f(x) = x^2 from 0 to 3, and how to calculate the same integral by breaking it into two integrals (from 0 to 2 and from 2 to 3).

```
» x = 0:0.001:3;
» fx = x.^2;
```

```
» 0.001*trapz(fx)
ans =
   9.0000

» f1x = fx(1:2000);
» f2x = fx(2000:end);
» 0.001*(trapz(f1x) + trapz(f2x))

ans =

   9.0000

» f2x = fx(2001:end);
» 0.001*(trapz(f1x) + trapz(f2x))

ans =

   8.9960
```

This test shows the following things:

- Usage of command.
- When the spacing of samples is other that unit, we must multiply by the spacing to get the correct result (proved above).
- The correct way to break an integral into two. The last sample of the first space is the first sample of the second.

Rephrasing the result of the above example, we can prove how we can calculate the result of trapz function in case we have a step-by-step calculation of samples:
Suppose $y_n = [f(x_0) \quad f(x_1) \quad \dots \quad f(x_n)]$ then

$$\text{trapz}(y_n) = \frac{1}{2}\Delta x\left[f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)\right]$$

$$\text{trapz}(y_{n-1}) = \frac{1}{2}\Delta x\left[f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-2}) + f(x_{n-1})\right]$$

$$\text{trapz}(y_n) - \text{trapz}(y_{n-1}) = \frac{1}{2}\Delta x\left[2f(x_{n-1}) + f(x_n) - f(x_{n-1})\right] \Leftrightarrow$$

$$\text{trapz}(y_n) - \text{trapz}(y_{n-1}) = \frac{1}{2}\Delta x\left[f(x_{n-1}) + f(x_n)\right] \Leftrightarrow$$

$$\boxed{\text{trapz}(y_n) = \text{trapz}(y_{n-1}) + \frac{1}{2}\Delta x\left[f(x_{n-1}) + f(x_n)\right]}$$

20.17 isfield

ISFIELD True if field is in structure array.
F = ISFIELD(S,'field') returns true if 'field' is the name of a field
in the structure array S.

See also GETFIELD, SETFIELD, FIELDNAMES.

Overloaded methods
help fighandle/isfield.m

20.18 getfield

GETFIELD Get structure field contents.
F = GETFIELD(S,'field') returns the contents of the specified
field. This is equivalent to the syntax F = S.field.
S must be a 1-by-1 structure.

F = GETFIELD(S,{i,j},'field',{k}) is equivalent to the syntax
F = S(i,j).field(k).
In other words, F = GETFIELD(S,sub1,sub2,...) returns the
contents of the structure S using the subscripts or field
references specified in sub1,sub2,etc. Each set of subscripts in
parentheses must be enclosed in a cell array and passed to
GETFIELD as a separate input. Field references are passed as
strings.

See also SETFIELD, ISFIELD, FIELDNAMES.

20.19 setfield

SETFIELD Set structure field contents.
S = SETFIELD(S,'field',V) sets the contents of the specified
field to the value V. This is equivalent to the syntax S.field = V.
S must be a 1-by-1 structure. The changed structure is returned.

S = SETFIELD(S,{i,j},'field',{k},V) is equivalent to the syntax
S(i,j).field(k) = V;
In other words, S = SETFIELD(S,sub1,sub2,...,V) sets the
contents of the structure S to V using the subscripts or field
references specified in sub1,sub2,etc. Each set of subscripts in
parentheses must be enclosed in a cell array and passed to
SETFIELD as a separate input. Field references are passed as
strings.

See also GETFIELD, ISFIELD, FIELDNAMES.

20.20 tic and toc commands (measure elapsed time)

TIC Start a stopwatch timer.

The sequence of commands

TIC, operation, TOC

prints the number of seconds required for the operation.

See also TOC, CLOCK, ETIME, CPUTIME.

TOC Read the stopwatch timer.

TOC, by itself, prints the elapsed time (in seconds) since TIC was used.

t = TOC; saves the elapsed time in t, instead of printing it out.

See also TIC, ETIME, CLOCK, CPUTIME.

Example:

```
» tic
» t1 = toc

t1 =

   4.4460
```

20.21 Return

RETURN Return to invoking function.

RETURN causes a return to the invoking function or to the keyboard.

It also terminates the KEYBOARD mode.

Normally functions return when the end of the function is reached.

A RETURN statement can be used to force an early return.

Example

```
function d = det(A)
if isempty(A)
  d = 1;
  return
else
 ...
end
```

See also FUNCTION, KEYBOARD.

20.22 Ind2sub

IND2SUB Multiple subscripts from linear index.

IND2SUB is used to determine the equivalent subscript values corresponding to a given single index into an array.

[I,J] = IND2SUB(SIZ,IND) returns the arrays I and J containing the equivalent row and column subscripts corresponding to the index matrix IND for a matrix of size SIZ.

For matrices, [I,J] = IND2SUB(SIZE(A),FIND(A>5)) returns the same values as [I,J] = FIND(A>5).

[I1,I2,I3,...,In] = IND2SUB(SIZ,IND) returns N subscript arrays I1,I2,..,In containing the equivalent N-D array subscripts equivalent to IND for an array of size SIZ.

See also SUB2IND, FIND.

Example:

```
» A = [ 1 2 3
4 5 6
7 8 9];

» A(6)

ans =

    8

» A(3,2)

ans =

    8

» [i_index, j_index] = ind2sub(size(A), 6)

i_index =
```

```
    3
j_index =
    2
```

20.23 Exist

```
EXIST  Check if variables or functions are defined.
   EXIST('A') returns:
    0 if A does not exist
    1 if A is a variable in the workspace
    2 if A is an M-file on MATLAB's search path or a file of unknown type
    3 if A is a MEX-file on MATLAB's search path
    4 if A is a MDL-file on MATLAB's search path
    5 if A is a built-in MATLAB function
    6 if A is a P-file on MATLAB's search path
    7 if A is a directory

   EXIST('A') or EXIST('A.EXT') returns 2 if a file named 'A' or 'A.EXT'
   is on MATLAB's search path and the extension isn't a P or MEX
   function extension.

   EXIST('A','var') checks only for variables.
   EXIST('A','builtin') checks only for built-in functions.
   EXIST('A','file') checks for files or directories on MATLAB's search path.
   EXIST('A','dir') checks only for directories.

   EXIST returns 0 if the specified instance isn't found.

   See also DIR, WHAT, ISEMPTY.

 Overloaded methods
   help inline/exist.m

»
```

20.24 Line

```
LINE   Create line.
   LINE(X,Y) adds the line in vectors X and Y to the current axes.
   If X and Y are matrices the same size, one line per column is added.
   LINE(X,Y,Z) creates lines in 3-D coordinates.

   LINE returns a column vector of handles to LINE objects,
   one handle per line. LINEs are children of AXES objects.
```

The X,Y pair (X,Y,Z triple for 3-D) can be followed by
parameter/value pairs to specify additional properties of the lines.
The X,Y pair (X,Y,Z triple for 3-D) can be omitted entirely, and
all properties specified using parameter/value pairs.

Execute GET(H), where H is a line handle, to see a list of line
object properties and their current values. Execute SET(H) to see a
list of line object properties and legal property values.

See also PATCH, TEXT, PLOT, PLOT3.

This function is used to add arbitrary lines to plots, to mark points for example.

20.25 logspace

LOGSPACE Logarithmically spaced vector.
LOGSPACE(d1, d2) generates a row vector of 50 logarithmically
equally spaced points between decades 10^d1 and 10^d2. If d2
is pi, then the points are between 10^d1 and pi.

LOGSPACE(d1, d2, N) generates N points.

See also LINSPACE, :.

20.26 linspace

LINSPACE Linearly spaced vector.
LINSPACE(x1, x2) generates a row vector of 100 linearly
equally spaced points between x1 and x2.

LINSPACE(x1, x2, N) generates N points between x1 and x2.

See also LOGSPACE, :.

20.27 repmat

REPMAT Replicate and tile an array.
B = REPMAT(A,M,N) replicates and tiles the matrix A to produce the
M-by-N block matrix B.

B = REPMAT(A,[M N]) produces the same thing.

B = REPMAT(A,[M N P ...]) tiles the array A to produce a
M-by-N-by-P-by-... block array. A can be N-D.

REPMAT(A,M,N) when A is a scalar is commonly used to produce

an M-by-N matrix filled with A's value. This can be much faster
than A*ONES(M,N) when M and/or N are large.

Example:
repmat(magic(2),2,3)
repmat(NaN,2,3)

See also MESHGRID.

This function can be used when we want to subtract a vector from a matrix, having one common dimension. In the PHD this was used in the MCB controller.
Example:

```
» t2 = [[1:9]; [9:-1:1]]

t2 =

   1   2   3   4   5   6   7   8   9
   9   8   7   6   5   4   3   2   1

» m = [1 2]'

m =

   1
   2

» t3 = t2 - repmat(m, 1, size(t2, 2))

t3 =

   0   1   2   3   4   5   6   7   8
   7   6   5   4   3   2   1   0  -1
```

20.28 subplot

SUBPLOT Create axes in tiled positions.
H = SUBPLOT(m,n,p), or SUBPLOT(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for for the current plot, and returns the axis handle. The axes are counted along the top row of the Figure window, then the second row, etc. For example,

SUBPLOT(2,1,1), PLOT(income)
SUBPLOT(2,1,2), PLOT(outgo)

plots income on the top half of the window and outgo on the bottom half.

SUBPLOT(m,n,p), if the axis already exists, makes it current.
SUBPLOT(H), where H is an axis handle, is another way of making
an axis current for subsequent plotting commands.

SUBPLOT('position',[left bottom width height]) creates an
axis at the specified position in normalized coordinates (in
in the range from 0.0 to 1.0).

If a SUBPLOT specification causes a new axis to overlap an
existing axis, the existing axis is deleted. For example,
the statement SUBPLOT(1,2,1) deletes all existing axes overlapping
the left side of the Figure window and creates a new axis on that
side.

SUBPLOT(111) is an exception to the rules above, and is not
identical in behavior to SUBPLOT(1,1,1). For reasons of backwards
compatibility, it is a special case of subplot which does not
immediately create an axes, but instead sets up the figure so that
the next graphics command executes CLF RESET in the figure
(deleting all children of the figure), and creates a new axes in
the default position. This syntax does not return a handle, so it
is an error to specify a return argument. The delayed CLF RESET
is accomplished by setting the figure's NextPlot to 'replace'.

20.29 ginput

GINPUT Graphical input from mouse.
[X,Y] = GINPUT(N) gets N points from the current axes and returns
the X- and Y-coordinates in length N vectors X and Y. The cursor
can be positioned using a mouse (or by using the Arrow Keys on some
systems). Data points are entered by pressing a mouse button
or any key on the keyboard except carriage return, which terminates
the input before N points are entered.

[X,Y] = GINPUT gathers an unlimited number of points until the
return key is pressed.

[X,Y,BUTTON] = GINPUT(N) returns a third result, BUTTON, that
contains a vector of integers specifying which mouse button was
used (1,2,3 from left) or ASCII numbers if a key on the keyboard
was used.

20.30 Clc

CLC Clear command window.

CLC clears the command window and homes the cursor.

See also HOME.

20.31 MATLAB Operators

MATLAB Operators and Special Characters R2020b

This page contains a comprehensive listing of all MATLAB® operators, symbols, and special characters.

Arithmetic Operators

Symbol	Role	More Information
+	Addition	plus
+	Unary plus	uplus
-	Subtraction	minus
-	Unary minus	uminus
.*	Element-wise multiplication	times
*	Matrix multiplication	mtimes
./	Element-wise right division	rdivide
/	Matrix right division	mrdivide
.\	Element-wise left division	ldivide
\	Matrix left division (also known as *backslash*)	mldivide
.^	Element-wise power	power
^	Matrix power	mpower
.'	Transpose	transpose
'	Complex conjugate transpose	ctranspose

Relational Operators

Symbol	Role	More Information
==	Equal to	eq
~=	Not equal to	ne
>	Greater than	gt
>=	Greater than or equal to	ge
<	Less than	lt
<=	Less than or equal to	le

Logical Operators

Symbol	Role	More Information
&	Find logical AND	and
|	Find logical OR	or
&&	Find logical AND (with short-circuiting)	Logical Operators: Short-Circuit && ||
||	Find logical OR (with short-circuiting)	
~	Find logical NOT	not

20.32 Handling Figure and Axes Properties

In many cases some properties of the figure or its axes must be changed in order to have a required appearance styling. To access these properties, commands get and set are used.

To get a first insight in these commands, Matlab help is presented first.

20.32.1 GET Get object properties.

V = GET(H,'PropertyName') returns the value of the specified property for the graphics object with handle H. If H is a vector of handles, then get will return an M-by-1 cell array of values where M is equal to length(H). If 'PropertyName' is replaced by a 1-by-N or N-by-1 cell array of strings containing property names, then GET will return an M-by-N cell array of values.

GET(H) displays all property names and their current values for the graphics object with handle H.

V = GET(H) where H is a scalar, returns a structure where each field name is the name of a property of H and each field contains the value of that property.

V = GET(0, 'Factory')
V = GET(0, 'Factory<ObjectType>')
V = GET(0, 'Factory<ObjectType><PropertyName>')
returns for all object types the factory values of all properties which have user-settable default values.

V = GET(H, 'Default')
V = GET(H, 'Default<ObjectType>')
V = GET(H, 'Default<ObjectType><PropertyName>')
returns information about default property values (H must be scalar). 'Default' returns a list of all default property values currently set on H. 'Default<ObjectType>' returns only the defaults for properties of <ObjectType> set on H. 'Default<ObjectType><PropertyName>' returns the default value for the specific property, by searching the defaults set on H and its ancestors, until that default is found. If no default value for this property has been set on H or any ancestor of H up through the root, then the factory value for that property is returned.

Defaults can not be queried on a descendant of the object, or on the object itself - for example, a value for 'DefaultAxesColor' can not be queried on an axes or an axes child, but can be queried on a figure or on the root.

When using the 'Factory' or 'Default' GET, if PropertyName is omitted then the return value will take the form of a structure in which each field name is a property name and the corresponding value is the value of that property. If PropertyName is specified then a matrix or string value will be returned.

See also SET, RESET, DELETE, GCF, GCA, FIGURE, AXES.

Overloaded methods
help hgbin/get.m
help axischild/get.m
help arrowline/get.m
help celltext/get.m
help cellline/get.m
help figobj/get.m
help scribehandle/get.m
help framerect/get.m
help axisobj/get.m
help axistext/get.m
help scribehgobj/get.m
help activex/get.m
help daqdevice/get.m
help daqchild/get.m
help rptsp/get.m
help rptcp/get.m
help cursor/get.m
help database/get.m
help fdline/get.m
help fdax/get.m
help fdmeas/get.m
help fdspec/get.m
help lti/get.m
help viewgui/get.m

help response/get.m
help tlchandle/get.m

20.32.2 SET Set object properties.

SET(H,'PropertyName',PropertyValue) sets the value of the
specified property for the graphics object with handle H.
H can be a vector of handles, in which case SET sets the
properties' values for all the objects.

SET(H,a) where a is a structure whose field names are object
property names, sets the properties named in each field name
with the values contained in the structure.

SET(H,pn,pv) sets the named properties specified in the cell array
of strings pn to the corresponding values in the cell array pv for
all objects specified in H. The cell array pn must be 1-by-N, but
the cell array pv can be M-by-N where M is equal to length(H) so
that each object will be updated with a different set of values
for the list of property names contained in pn.

SET(H,'PropertyName1',PropertyValue1,'PropertyName2',PropertyValue2,...)
sets multiple property values with a single statement. Note that it
is permissible to use property/value string pairs, structures, and
property/value cell array pairs in the same call to SET.

A = SET(H, 'PropertyName')
SET(H,'PropertyName')
returns or displays the possible values for the specified
property of the object with handle H. The returned array is
a cell array of possible value strings or an empty cell array
if the property does not have a finite set of possible string
values.

A = SET(H)
SET(H)
returns or displays all property names and their possible values for
the object with handle H. The return value is a structure whose
field names are the property names of H, and whose values are
cell arrays of possible property values or empty cell arrays.

The default value for an object property can be set on any of an

```
   object's ancestors by setting the PropertyName formed by
   concatenating the string 'Default', the object type, and the
   property name.  For example, to set the default color of text objects
   to red in the current figure window:

     set(gcf,'DefaultTextColor','red')

   Defaults can not be set on a descendant of the object, or on the
   object itself - for example, a value for 'DefaultAxesColor' can not
   be set on an axes or an axes child, but can be set on a figure or on
   the root.

   Three strings have special meaning for PropertyValues:
    'default' - use default value (from nearest ancestor)
    'factory' - use factory default value
    'remove'  - remove default value.

   See also GET, RESET, DELETE, GCF, GCA, FIGURE, AXES.

Overloaded methods
   help axischild/set.m
   help editline/set.m
   help arrowline/set.m
   help celltext/set.m
   help cellline/set.m
   help figobj/set.m
   help scribehandle/set.m
   help framerect/set.m
   help axisobj/set.m
   help editrect/set.m
   help axistext/set.m
   help scribehgobj/set.m
   help activex/set.m
   help daqdevice/set.m
   help daqchild/set.m
   help rptsp/set.m
   help rptcp/set.m
   help sgmltag/set.m
   help cursor/set.m
   help database/set.m
   help fdline/set.m
```

```
help fdax/set.m
help fdmeas/set.m
help fdspec/set.m
help tf/set.m
help lti/set.m
help ss/set.m
help frd/set.m
help zpk/set.m
help icplot/set.m
help svplot/set.m
help viewgui/set.m
help impplot/set.m
help pzplot/set.m
help margplot/set.m
help bodplot/set.m
help lsimplot/set.m
help stepplot/set.m
help nicplot/set.m
help response/set.m
help nyplot/set.m
help tlchandle/set.m
```

20.32.3 Where to Find What

To enlarge the letters of an axis:

```
zlabelH = zlabel('Acc X in m/sec^2');
set(zlabelH, 'FontSize', <Font Size>);
```

To enlarge the numbers at an axis:

```
axesH = get(figH1, 'CurrentAxes');
set(axesH, 'FontSize', <FontSize>);
```

To thicken the line

```
» fig_ID = figure

fig_ID =

    1

» ln_ID = plot(x,y)
```

```
ln_ID =

   7.200146484375000e+001

» get(fig_ID)
        BackingStore = on
        CloseRequestFcn = closereq
        Color = [0.8 0.8 0.8]
        Colormap = [ (64 by 3) double array]
        CurrentAxes = [73.0026]
        CurrentCharacter =
        CurrentObject = []
        CurrentPoint = [0 0]
        Dithermap = [ (64 by 3) double array]
        DithermapMode = manual
        DoubleBuffer = off
        FileName =
        FixedColors = [ (10 by 3) double array]
        IntegerHandle = on
        InvertHardcopy = on
        KeyPressFcn =
        MenuBar = figure
        MinColormap = [64]
        Name =
        NextPlot = add
        NumberTitle = on
        PaperUnits = inches
        PaperOrientation = portrait
        PaperPosition = [0.25 2.5 8 6]
        PaperPositionMode = manual
        PaperSize = [8.5 11]
        PaperType = usletter
        Pointer = arrow
        PointerShapeCData = [ (16 by 16) double array]
        PointerShapeHotSpot = [1 1]
        Position = [680 672 560 420]
        Renderer = painters
        RendererMode = auto
        Resize = on
        ResizeFcn =
        SelectionType = normal
```

```
        ShareColors = on
        Units = pixels
        WindowButtonDownFcn =
        WindowButtonMotionFcn =
        WindowButtonUpFcn =
        WindowStyle = normal

        ButtonDownFcn =
        Children = [73.0026]
        Clipping = on
        CreateFcn =
        DeleteFcn =
        BusyAction = queue
        HandleVisibility = on
        HitTest = on
        Interruptible = on
        Parent = [0]
        Selected = off
        SelectionHighlight = on
        Tag =
        Type = figure
        UIContextMenu = []
        UserData = []
        Visible = on

» get(ln_ID)
        Color = [0 0 1]
        EraseMode = normal
        LineStyle = -
        LineWidth = [0.5]
        Marker = none
        MarkerSize = [6]
        MarkerEdgeColor = auto
        MarkerFaceColor = none
        XData = [ (1 by 100) double array]
        YData = [ (1 by 100) double array]
        ZData = []

        ButtonDownFcn =
        Children = []
        Clipping = on
```

```
            CreateFcn =
            DeleteFcn =
            BusyAction = queue
            HandleVisibility = on
            HitTest = on
            Interruptible = on
            Parent = [73.0026]
            Selected = off
            SelectionHighlight = on
            Tag =
            Type = line
            UIContextMenu = []
            UserData = []
            Visible = on

» Axes_ID = get(fig_ID, 'CurrentAxes')

Axes_ID =

   7.300256347656250e+001

» get(Axes_ID)
            AmbientLightColor = [1 1 1]
            Box = on
            CameraPosition = [50 100 17.3205]
            CameraPositionMode = auto
            CameraTarget = [50 100 0]
            CameraTargetMode = auto
            CameraUpVector = [0 1 0]
            CameraUpVectorMode = auto
            CameraViewAngle = [6.60861]
            CameraViewAngleMode = auto
            CLim = [0 1]
            CLimMode = auto
            Color = [1 1 1]
            CurrentPoint = [ (2 by 3) double array]
            ColorOrder = [ (7 by 3) double array]
            DataAspectRatio = [50 100 1]
            DataAspectRatioMode = auto
            DrawMode = normal
            FontAngle = normal
```

FontName = Helvetica
FontSize = [10]
FontUnits = points
FontWeight = normal
GridLineStyle = :
Layer = bottom
LineStyleOrder = -
LineWidth = [0.5]
NextPlot = replace
PlotBoxAspectRatio = [1 1 1]
PlotBoxAspectRatioMode = auto
Projection = orthographic
Position = [0.13 0.11 0.775 0.815]
TickLength = [0.01 0.025]
TickDir = in
TickDirMode = auto
Title = [74.0021]
Units = normalized
View = [0 90]
XColor = [0 0 0]
XDir = normal
XGrid = off
XLabel = [75.0011]
XAxisLocation = bottom
XLim = [0 100]
XLimMode = auto
XScale = linear
XTick = [(1 by 11) double array]
XTickLabel = [(11 by 3) char array]
XTickLabelMode = auto
XTickMode = auto
YColor = [0 0 0]
YDir = normal
YGrid = off
YLabel = [76.0011]
YAxisLocation = left
YLim = [0 200]
YLimMode = auto
YScale = linear
YTick = [(1 by 11) double array]
YTickLabel = [(11 by 3) char array]

```
        YTickLabelMode = auto
        YTickMode = auto
        ZColor = [0 0 0]
        ZDir = normal
        ZGrid = off
        ZLabel = [77.0009]
        ZLim = [-1 1]
        ZLimMode = auto
        ZScale = linear
        ZTick = [-1 0 1]
        ZTickLabel =
        ZTickLabelMode = auto
        ZTickMode = auto

        ButtonDownFcn =
        Children = [72.0015]
        Clipping = on
        CreateFcn =
        DeleteFcn =
        BusyAction = queue
        HandleVisibility = on
        HitTest = on
        Interruptible = on
        Parent = [1]
        Selected = off
        SelectionHighlight = on
        Tag =
        Type = axes
        UIContextMenu = []
        UserData = []
        Visible = on

» get(get(Axes_ID, 'YLabel'))
        Color = [0 0 0]
        EraseMode = normal
        Editing = off
        Extent = [-7.60369 98.8304 0 0]
        FontAngle = normal
        FontName = Helvetica
        FontSize = [10]
        FontUnits = points
```

FontWeight = normal
HorizontalAlignment = center
Position = [-7.3903 99.1202 17.3205]
Rotation = [90]
String =
Units = data
Interpreter = tex
VerticalAlignment = baseline

ButtonDownFcn =
Children = []
Clipping = off
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = off
HitTest = on
Interruptible = on
Parent = [73.0026]
Selected = off
SelectionHighlight = on
Tag =
Type = text
UIContextMenu = []
UserData = []
Visible = on

set(get(Axes_ID, 'YLabel'), 'String', 'Y samples')
is equivalent to say ylabel('Y samples')

20.32.4 Summing-up:

For some papers we needed to change titles to size 24, text of axes size 18, grid on once again. Line width equals to 5.

- Label font:
    - y_ID = ylabel('Y samples')

20.33 cat

CAT Concatenate arrays.
CAT(DIM,A,B) concatenates the arrays A and B along
the dimension DIM.
CAT(2,A,B) is the same as [A,B].

```
   CAT(1,A,B) is the same as [A;B].

   B = CAT(DIM,A1,A2,A3,A4,...) concatenates the input
   arrays A1, A2, etc. along the dimension DIM.

   When used with comma separated list syntax, CAT(DIM,C{:}) or
   CAT(DIM,C.FIELD) is a convenient way to concatenate a cell or
   structure array containing numeric matrices into a single matrix.

   Examples:
     a = magic(3); b = pascal(3);
     c = cat(4,a,b)
   produces a 3-by-3-by-1-by-2 result and
     s = {a b};
     for i=1:length(s),
       siz{i} = size(s{i});
     end
     sizes = cat(1,siz{:})
   produces a 2-by-2 array of size vectors.

   See also NUM2CELL.

 Overloaded methods
   help inline/cat.m
   help lti/cat.m
```

As an example use that applied to PHD code, is the following:

```
t =

   'First comment'
   'Second comment'
» t = cat(1, t, t)

t =

   'First comment'
   'Second comment'
   'First comment'
   'Second comment'
```

So this code concatenated the same cell array by adding a copy at the lines that follow (along the first dimension).

20.34 Sorting a matrix based on one row

Suppose we have the matrix

t2 =

```
   1   2   3   4   5   6   7   8   9
   9   8   7   6   5   4   3   2   1
```

And we want top sort all elements based on row 2 (sort ascending).

```
» [s_row, s_indexes] = sort(t2(2,:))

s_row =

   1   2   3   4   5   6   7   8   9


s_indexes =

   9   8   7   6   5   4   3   2   1
```

By using this version of sort, we also keep the idexes of the sort command usage over row 2. These indexes can be used to call the new sorted version of the whole matrix:

```
» t2(:, s_indexes)

ans =

   9   8   7   6   5   4   3   2   1
   1   2   3   4   5   6   7   8   9
```

20.35 Try – Catch blocks

TRY Begin TRY block.

The general form of a TRY statement is:

TRY, statement, ..., statement, CATCH, statement, ..., statement END

Normally, only the statements between the TRY and CATCH are executed. However, if an error occurs while executing any of the statements, the error is captured into LASTERR and the statements between the CATCH and END are executed. If an error occurs within the CATCH statements, execution will stop unless caught by another TRY...CATCH block. The error string produced by a failed TRY block can be obtained with

LASTERR.

See also EVAL, EVALIN, CATCH, END.

Example:
» try
sfds(35)
catch
disp('Found sth wrong..')
end
Found sth wrong..
» lasterr

ans =

Undefined function or variable 'sfds'.

»

20.36 Textread

TEXTREAD Read formatted data from text file.
[A,B,C,...] = TEXTREAD(FILENAME,FORMAT,N) reads data from the file FILENAME into the variables A,B,C,etc. The number and types of each return argument is given by the FORMAT string. The number of return arguments is the number of conversion specifiers in the FORMAT string. The FORMAT string supports a subset of the specifiers and conventions of the C language FSCANF function. If N is specified, the format string will be reused N times. Values of N smaller than zero cause TEXTREAD to read the entire file (the default).

The FORMAT string can contain whitespace characters (which are ignored), ordinary characters (which are expected to match the next non-whitespace character in the input), or conversion specifications.

Supported conversion specifications:
%d - read a signed integer value (output is a double array)
%u - read a integer value (output is a double array)
%f - read a floating point value (output is a double array)
%s - read a whitespace separated string (output is a cellstr)
%q - read a (possibly double quoted) string (output is a cellstr)

```
   %c  - read characters (including white space) (output is char array)
   %[...]  - reads the longest string containing characters in the set
         between the brackets (output is a cellstr)
   %[^...] - reads the longest non-empty string containing characters
         not in the set between brackets (output is a cellstr)

Using %* instead of % in a conversion causes TEXTREAD to skip the
matching characters in the input (and no output is created for this
conversion).  The % can be followed by an optional field width to
handle fixed width fields. For example %5d reads a 5 digit
integer. In addition the %f format supports the form %<width>.<prec>f.

TEXTREAD works by matching and converting groups of characters from
the input.  These input fields are defined as a string of non-white
space characters that extends to the next white space or delimiter
character or until the maximum field width is exhausted.  Repeated
delimiter characters are significant while repeated whitespace
characters are treated as one.

[...] = TEXTREAD(...,param,value,...) allows param/value pairs to be
used to customize the behavior of TEXTREAD.  Possible param/value
options are:
    'whitespace'   - vector of characters to treat as whitespace
                (default is ' \b\r\n\t')
    'delimiter'    - delimiter characters (default is none)
    'expchars'     - exponent characters (default is 'eEdD')
    'bufsize'      - Maximum string length in bytes (default is 4095)
    'headerlines'  - Number of lines at beginning of file to skip
    'commentstyle' - one of
        'matlab' -- characters after % are ignored
        'shell'  -- characters after # are ignored
        'c'      -- characters between /* and */ are ignored
        'c++'    -- characters after // are ignored

TEXTREAD is useful for reading text files with a known format.  Both
fixed and free format files can be handled.

Examples:
 Suppose the text file mydata.dat contains data in the following form:
   Sally    Type1 12.34 45 Yes
   Joe      Type2 23.54 60 No
```

```
   Bill     Type1 34.90 12 No

  This could be read using the following command
   [names,types,x,y,answer] = textread('mydata.dat','%s %s %f %d %s');

  Read file as a fixed format file while skipping the doubles
   [names,types,y,answer] = textread('mydata.dat','%9c %5s %*f %2d %3s');

  Read file and match Type literal
   [names,typenum,x,y,answer]=textread('mydata.dat','%s Type%d %f %d %s');

  Read m-file into cell array of strings
   file = textread('fft.m','%s','delimiter','\n','whitespace','');

 See also DLMREAD, SSCANF.
```

PHD example:

```
» fname =
'c:/PHD/Matlab_180910/ModelsRevisionAndControl/MBC_Test_Bed_Inputs_and_Results/InputVectors1.txt'

fname =

c:/PHD/Matlab_180910/ModelsRevisionAndControl/MBC_Test_Bed_Inputs_and_Results/InputVectors1.txt

» [Sp W SimReps] = textread(fname,'%f %f %f','headerlines', 1)

Sp =

   10
   15
   15
   20
   10
   20


W =

    0
```

```
     7
    -3
    -5
   -25
     0


SimReps =

    30
    25
    40
    30
    50
    50

»
```

The ‘headerlines’ attribute shows how many lines are omitted at the beginning of the file, before starting to read the required fields.

20.37 Saving

20.37.1 Save Workspace Variables

```
» fname = 'tst.mat';
» a = 1;
» b = 2;
» c = 3;
» d = 4;

» whos
  Name       Size       Bytes  Class

  a          1x1            8  double array
  b          1x1            8  double array
  c          1x1            8  double array
  d          1x1            8  double array
  fname      1x7           14  char array
  tst        1x88         176  char array

Grand total is 99 elements using 222 bytes

» save(fname)

» clear all
» whos
» load('tst.mat')
» whos
  Name       Size       Bytes  Class

  a          1x1            8  double array
  b          1x1            8  double array
  c          1x1            8  double array
  d          1x1            8  double array
  fname      1x7           14  char array
  tst        1x88         176  char array

Grand total is 99 elements using 222 bytes
```

20.37.2 Save Figures (saveas command)

SAVEAS Save Figure or model to desired output format.
SAVEAS(H, FILENAME)
Will save the Figure or model with handle H to file called FILENAME.
The format of the file is determined from the extension of FILENAME.

SAVEAS(H, FILENAME, FORMAT)
Will save the Figure or model with handle H to file called FILENAME.
in the format specified by FORMAT. FORMAT can be the same values as
extensions of FILENAME. The FILENAME extension does not have to be
the same as FORMAT. Given FORMAT overrides FILENAME extension.

Valid options for FORMAT are:

FIG - save figure to a single binary FIG-file. Reload using OPEN.
M - save figure to binary FIG-file, and produce callable M-file for reload.
MFIG - same as M.
MMAT - save figure to callable M-file as series of creation commands
with param-value pair arguments. Large data is saved to MAT-file.
Note: MMAT Does not support some newer graphics features. Use this format
only when code inspection is the primary goal. FIG-files support all
feautures, and load more quickly.

Allowable options also include devices allowed by PRINT.

Examples:

Write current figure to MATLAB fig file

```
saveas(gcf, 'output', 'fig')
```

Write current figure to windows bitmap file

```
saveas(gcf, 'output', 'bmp')
```

See Also OPEN, PRINT

Loghis testing:

```
» tst_fig = figure;
» y = 1:100;
```

```
» tst_fig = figure

tst_fig =

    1

» y = 1:100

» plot(y)
» pwd

ans =

» saveas(tst_fig, 'TestFigure.jpg', 'jpg')
» saveas(tst_fig, 'TestFigure.bmp', 'bmp')
»
```

20.38 Alternative Matlab Plots

Besides plot command there are alternative plotting commands such as:

- quiver: plots vectors between requested points
- compass: plots centered vectors at a compass-like surface.
- quiver3: plots 3D vectors at selected points of surfaces.
- Feather.

many more can be found at the help of these commands.

21 Publications

[1] E. K. Loghis and N. I. Xiros, “Development of Discrete-Time Waterjet Control Systems Used in Surface Vehicle Thrust Vectoring,” *Journal of Marine Science and Engineering*, vol. 10, no. 12, 2022, doi: 10.3390/jmse10121844.

[2] E. C. Loghis, N. I. Xiros, and E. C. Loghis, “Continuous and Discrete-time Models of Surface Watercraft Nonlinear Dynamics,” *Acta Scientific Computer Sciences*, 2022.

[3] G. D. Ntouni *et al.*, “Real-time Experimental Wireless Testbed with Digital Beamforming at 300 GHz,” in *2020 European Conference on Networks and Communications (EuCNC)*, Jun. 2020. doi: 10.1109/eucnc48522.2020.9200948.

[4] L. Wang, N. I. Xiros, and E. K. Loghis, “Design and Comparison of H$ınfty$/H2 Controllers for Frigate Rudder Roll Stabilization,” *Journal of Marine Science and Application*, vol. 18, no. 4, pp. 492–509, Oct. 2019, doi: 10.1007/s11804-019-00116-3.

[5] N. Xiros, V. Tzelepis, and E. Loghis, “Modeling and Simulation of Planing-Hull Watercraft Outfitted with an Electric Motor Drive and a Surface-Piercing Propeller,” *Journal of Marine Science and Engineering*, vol. 7, no. 2, p. 49, Feb. 2019, doi: 10.3390/jmse7020049.

[6] E. C. Loghis, N. I. Xiros, and R. Saxton, *A Dynamic Interaction Simulator for Studying Macroscopic Swarm Self-organization of Autonomous Surface Watercraft*, vol. All Days. 2016.

[7] N. I. Xiros, E. Logis, E. Gasparis, S. Tsolakidis, and K. Kardasis, “Theoretical and experimental investigation of unmanned boat electric propulsion system with PMDC motor and waterjet,” *Journal of Marine Engineering &amp$\mathsemicolon$ Technology*, vol. 8, no. 2, pp. 27–43, Jan. 2009, doi: 10.1080/20464177.2009.11020221.

[8] N. Xiros and E. Loghis, “System Identification Using Neural Nets for Dynamic Modeling of a Surface Marine Vehicle,” Nov. 2014, p. V010T13A015. doi: 10.1115/IMECE2014-38322.

[9] M. G. Farmakopoulos, E. K. Loghis, P. G. Nikolakopoulos, N. I. Xiros, and C. A. Papadopoulos, “Modeling and Control of the Electrical Actuation System of an Active Hydromagnetic Journal Bearing (AHJB),” in *Volume 4B: Dynamics, Vibration, and Control*, Nov. 2014. doi: 10.1115/imece2014-38346.

[10] N. I. Xiros and E. K. Loghis, “Dynamic Modeling and System Identification of a Surface Marine Vehicle for Autonomous Swarm Applications,” in *Volume 4: Dynamics, Control and Uncertainty, Parts A and B*, Nov. 2012. doi: 10.1115/imece2012-89404.

[11] E. C. Loghis, N. I. Xiros, and E. C. Loghis, “An Automatic Steering System for Robust Disturbance Rejection,” in *IASME Transactions2*, 2004.

Publications also available at https://orcid.org/0000-0002-3743-131X

22 References

[1] T. Fossen, *Guidance and Control of Ocean Vehicles*, 1st ed. Michigan: Wiley, 1994.

[2] T. Perez, *Ship Motion Control: Course Keeping and Roll Stabilisation Using Rudder and Fins*. in Advances in Industrial Control. Springer, 2005. [Online]. Available: https://books.google.gr/books?id=_7apbonEQHgC

[3] "Models for Ships, Offshore Structures and Underwater Vehicles," in *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons, Ltd, 2011, pp. 133–186. doi: https://doi.org/10.1002/9781119994138.ch7.

[4] K. Ogata, *Discrete-Time Control Systems*, 2nd ed. New Jersey: Prentice-Hall Inc., 1995.

[5] E. K. Loghis and N. I. Xiros, "Development of Discrete-Time Waterjet Control Systems Used in Surface Vehicle Thrust Vectoring," *Journal of Marine Science and Engineering*, vol. 10, no. 12, 2022, doi: 10.3390/jmse10121844.

[6] N. W. H. Bulten, "Numerical analysis of a waterjet propulsion system," PhD Thesis, Mechanical Engineering, Eindhoven, 2006. doi: 10.6100/IR614907.

[7] N. I. Xiros, E. Logis, E. Gasparis, S. Tsolakidis, and K. Kardasis, "Theoretical and experimental investigation of unmanned boat electric propulsion system with PMDC motor and waterjet," *Journal of Marine Engineering & Technology*, vol. 8, no. 2, pp. 27–43, Jan. 2009, doi: 10.1080/20464177.2009.11020221.

[8] N. I. Xiros, V. Tzelepis, and E. K. Loghis, "Modeling and Simulation of Planing-Hull Watercraft Outfitted with an Electric Motor Drive and a Surface-Piercing Propeller," *Journal of Marine Science and Engineering*, vol. 7, no. 2, 2019, doi: 10.3390/jmse7020049.

[9] N. I. Xiros and E. K. Loghis, "System Identification Using Neural Nets for Dynamic Modeling of a Surface Marine Vehicle," in *Volume 10: Micro- and Nano-Systems Engineering and Packaging*, American Society of Mechanical Engineers, Nov. 2014. doi: 10.1115/imece2014-38322.

[10] N. I. Xiros, "Digital Signal Processing," in *Springer Handbook of Ocean Engineering*, M. R. Dhanak and N. I. Xiros, Eds., Cham: Springer International Publishing, 2016, pp. 197–226. doi: 10.1007/978-3-319-16649-0_9.

[11] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 1st ed. US: Prentice-Hall Inc., 1989.

[12] A. Papoulis, *Signal Analysis*, 1st ed. McGraw-Hill, 1984.

[13] C. J. Zarowski, *An Introduction to Numerical Analysis for Electrical and Computer Engineers*. Wiley, 2004. [Online]. Available: https://books.google.gr/books?id=3AihEG52ImkC

[14] S. S. Soliman and M. D. Srinath, *Continuous and Discrete Signals and Systems*, 2nd ed. New Jersey: Prentice-Hall Inc., 1998.

[15] B. C. Kuo, *Digital Control Systems*, 1st ed. Hong Kong: CBS Publishing Asia LTD, 1987.

[16] Σ. Γ. Τζαφέστας, *Υπολογιστική Νοημοσύνη Τόμος Α: Μεθοδολογίες*, 1st ed., vol. 1, 2 vols. Πανεπιστημιακές Εκδόσεις Ε.Μ.Π., 2002.

[17] S. Haykin, S. S. Haykin, and S. A. HAYKIN, *Neural Networks: A Comprehensive Foundation*. in International edition. Prentice Hall, 1999. [Online]. Available: https://books.google.gr/books?id=bX4pAQAAMAAJ

[18] Β. Β. Μάρκελλος and Ι. Α. Κουτρουβέλης, *Μαθηματικά για Μηχανικούς Ι*, 1st ed., vol. 1 & 2, 5 vols. Εκδόσεις Συμμετρία, 1990.
[19] Β. Β. Μάρκελλος, *Μαθηματικά για Μηχανικούς ΙΙ*, 1st ed., vol. 4 & 5, 5 vols. Εκδόσεις Συμμετρία, 1994.
[20] Π. Μ. Χαντζηκωνσταντίνου, *Μαθηματικά για Μηχανικούς Τεύχος 3*, 1st ed., vol. 3, 5 vols. Εκδόσεις Συμμετρία, 1993.
[21] Σ. Αλκ. Παϊπέτης, *Τεχνική Μηχανική ΙΙ Δυναμική*, 3rd ed., vol. 2, 3 vols. in Τεχνική Μηχανική, vol. 2. Αθήνα: Amatec S.A., 1992.
[22] Β. Β. Μάρκελλος, *Αριθμητικές Μέθοδοι*, 1st ed. Αθήνα: Εκδόσεις Συμμετρία, 1995.
[23] L. A. Pipes and L. R. Harvill, *Applied Mathematics for Engineers and Physicists: Third Edition*. in Dover Books on Mathematics. Dover Publications, 2014. [Online]. Available: https://books.google.gr/books?id=cvmJAwAAQBAJ
[24] W. L. Brogan, *Modern Control Theory*. Prentice Hall, 1991. [Online]. Available: https://books.google.gr/books?id=OPFQAAAAMAAJ
[25] S. P. Bhattacharyya, “The art of control engineering: K. Dutton, S. Thompson and B. Barraclough; Addison-Wesley, Longman, 1997, ISBN: 0-201-17545-2.,” *Autom.*, vol. 37, no. 6, pp. 961–964, 2001.
[26] I. J. Nagrath and M. Gopal, *Control Systems Engineering*, 2nd ed. Singapore: John Wiley & Sons, 1986.
[27] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems*, 10th ed. US: Wiley, 2010.
[28] N. I. Xiros, *Robust Control of Diesel Ship Propulsion*, 1st ed. London: Springer London, 2002. [Online]. Available: https://doi.org/10.1007/978-1-4471-0191-8
[29] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. in Prentice Hall Modular Series for Eng. Prentice Hall, 1998. [Online]. Available: https://books.google.gr/books?id=QviHQgAACAAJ
[30] G. Strang, *Linear Algebra and Its Applications*. Cengage Learning India Private Limited, 2006. [Online]. Available: https://books.google.gr/books?id=ISopuAAACAAJ
[31] F. Lin, *Robust Control Design: An Optimal Control Approach*. in RSP. Wiley, 2007. [Online]. Available: https://books.google.gr/books?id=cpgkjiic9vYC
[32] H. K. Khalil, *Nonlinear Systems*. in Pearson Education. Prentice Hall, 2002. [Online]. Available: https://books.google.gr/books?id=t_d1QgAACAAJ
[33] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. in Studies in Nonlinearity. Avalon Publishing, 2014. [Online]. Available: https://books.google.gr/books?id=aNHPswEACAAJ
[34] D. G. Alciatore and M. B. Histand, *Introduction to Mechatronics and Measurement Systems*, 4th ed. US: McGraw-Hill, 2012.
[35] D. M. Auslander and C. J. Kempf, *Mechatronics: Mechanical System Interfacing*, 1st ed. US: Prentice Hall, 1995.
[36] M. M. Mano, *Digital Design*. in Prentice Hall international editions. Prentice Hall, 2002. [Online]. Available: https://books.google.gr/books?id=8WhBtfnaenkC
[37] E. C. Loghis, N. I. Xiros, and R. Saxton, *A Dynamic Interaction Simulator for Studying Macroscopic Swarm Self-organization of Autonomous Surface*

Watercraft, vol. All Days. in International Ocean and Polar Engineering Conference, vol. All Days. 2016.

[38] N. I. Xiros, E. C. Loghis, and G. Charitos, “An Automatic Steering System for Robust Disturbance Rejection,” in *IASME Transactions2*, 2004.

[39] I. Prigogine, *Non-Equilibrium Statistical Mechanics*. in Dover Books on Physics. Dover Publications, 2017. [Online]. Available: https://books.google.gr/books?id=eCsCDgAAQBAJ

[40] B. Thide, *Electromagnetic Field Theory*. Dover Publications, Incorporated, 2011. [Online]. Available: https://books.google.gr/books?id=LaWtkgEACAAJ

[41] J. D. Jackson, *Classical Electrodynamics*. Wiley, 1962. [Online]. Available: https://books.google.gr/books?id=Td8NAQAAIAAJ

[42] C. T. Kilian, *Modern Control Technology: Components and Systems*, 3rd ed. US: CENGAGE Delmar Learning, 2005.

[43] Τ. Γ. Κουσιουρής, *Θεωρία Γραμμικών Πολυμεταβλητών Συστημάτων*, 1st ed. Αθήνα: Πανεπιστημιακές Εκδόσεις Ε.Μ.Π., 1990.

[44] W. A. Wolovich, *Linear Multivariable Systems*, 1st ed. in Applied Mathematical Sciences, no. 0066–5452. New York: Springer New York, NY, 1974.

[45] A. D. Polyanin and A. V. Manzhirov, *Handbook of Mathematics for Engineers and Scientists*. Taylor & Francis, 2006. [Online]. Available: https://books.google.gr/books?id=G0xUzQEACAAJ