ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ
ΜΗΧΑΝΙΚΩΝ
ΑΥΤΟΜΑΤΙΣΜΟΥ

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

ΘΕΜΑ

# ΣΧΕΔΙΑΣΗ ΕΙΚΟΝΙΚΟΥ ΕΡΓΑΣΤΗΡΙΟΥ ΣΥΣΤΗΜΑΤΩΝ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ ΣΕ ΠΕΡΙΒΑΛΛΟΝ MATLAB/SIMULINK(Design of a αvirtual control systems lab in Matlab/Simulink.)

ΟΝΟΜΑ ΦΟΙΤΗΤΗ : ΜΠΑΛΑΤΣΟΣ – ΘΕΟΔΩΡΟΥ ΧΑΡΑΛΑΜΠΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : ΥΦΟΥΛΗΣ ΧΡΙΣΤΟΣ

ΘΕΣΣΑΛΟΝΙΚΗ 2017

**ΕΥΧΑΡΙΣΤΙΕΣ**

Θα ήθελα να ευχαριστήσω τους καθηγητές του τμήματος Μηχανικών Αυτοματισμού για την καθοδήγησή τους και τις γνώσεις τους που μου παρείχαν κατά τη διάρκεια των σπουδών μου.

Συγκεκριμένα θα ήθελα να εκφράσω τις ευχαριστίες μου στον καθηγητή κ. Υφούλη για την συμπαράστασή του καθώς και την κατανόησή του στις δυσκολίες που αντιμετώπισα στην εκπόνηση της πτυχιακής μου εργασίας. Επίσης θέλω να εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου που με στήριξαν με τον καλύτερο τους εαυτό στην εκπαιδευτική μου πορεία και μου στάθηκαν μέχρι το τέλος της.

**ΠΕΡΙΕΧΟΜΕΝΑ**

**ΠΕΡΙΛΗΨΗ**

Στη παρούσα εργασία θα ασχοληθούμε με την δημιουργία ενός περιβάλλοντος εργασίας για την εξέταση πειράμματος σε εικονικό εργαστήριο στο πρόγραμμα Matlab μαζί με τα λογισμικά προγράμματα που περιέχονται σε αυτό (Simulink, GUI). Η εργασία αυτή πραγματοποιήθηκε στα πλαίσια της εκπόνησης της πτυχιακής εργασίας στο τμήμα Μηχανικών Αυτοματισμού του Αλεξάνδρειου Τεχνολογικού Ιδρύματος Θεσσαλονίκης.

Λέξεις κλειδιά : Matlab, Simulink, GUI, εικονικό εργαστήρειο, animation, realtime.

**ABSTRACT**

In this assignment we will deal with the establishment of a working environment for the examination of a test in a virtual lab using Matlab software and the programs contained in it (Simulink, GUI). This assignment was carried out in the context of the preparation of the postgraduate work in the Department of Automation Mechanics of the Alexandrian Technological Institute of Thessaloniki.

Keywords: Matlab, Simulink, GUI, virtual lab, animation, realtime

# 1.ΕΙΣΑΓΩΓΗ

Ο αυτοματισμός είναι μία από τις σημαντικότερες επιστημονικές επιστήμες τη σημερινή εποχή. Αυτό συμβαίνει γιατί οι αυτοματισμοί είναι πλέον αναπόσπαστος κλάδος σε όλες τις τεχνολογικές επιστήμες. Η θεωρία του ψηφιακού αυτόματου ελέγχου αναπτύχθηκε κυρίως τα τελευταία 20 χρόνια λόγω της ταχείας εξέλιξης των ψηφιακών υπολογιστών. Λόγω της μείωσης του κόστους στις διεργασίες έχει εξαπλωθεί η χρήση του και η ανάπτυξή του γνώρισε τρομερή άνοδο τις τελευταίες δεκαετίες.

Τα ψηφιακά συστήματα ελέγχου (digitalcontrolsystems), ή συστήματα ελεγχόμενα με υπολογιστή είναι η επιστημονική περιοχή που σκοπό έχει την ανάπτυξη μεθόδων σχεδιασμού συστημάτων ελέγχου βασισμένα σε ψηφιακό υπολογιστή. Οι μέθοδοι αυτοί έχουν εφαρμοστεί με μεγάλη επιτυχία σε πολλές πρακτικές εφαρμογές.

Η προσομοίωση αποτελείται από οντότητες που έχουν ιδιότητες και μεθόδους που αλληλεπιδρούν μεταξύ τους κατά τη διάρκεια των διεργασιών κάτω από ορισμένες συνθήκες για να παράγουν γεγονότα τα οποία θα αλλάξουν την κατάσταση του συστήματος (ClaudeShannon).

Με τον όρο υπολογιστική προσέγγιση εννοούμε την προσέγγιση που χαρακτηρίζει τα πειράματα σε υπολογιστικό περιβάλλον βασικό και αναπόσπαστο εργαλείο στην διδασκαλία και γενικά της εκπαίδευσης. Τα υπολογιστικά πειράματα πλέον στις επιστήμες είναι ένας από τους τρεις άξονες που διέπουν την θεωρία των επιστημών, το πείραμα, την αριθμητική μοντελοποίηση ή υπολογιστικά πειράματα και τη θεωρία.

Η πτυχιακή εργασία έχει ως θέμα πειραματικά μοντέλα. Εδώ μπαίνει το ερώτημα για ποιο λόγο χρειάζονται και αναπτύχθηκαν τα πειραματικά μοντέλα. Οι λόγοι είναι πολλοί και ποικίλουν. Σαν επιστήμη είναι σχετικά νέα η οποία ξεκίνησε να ανθίζει και να χρησιμοποιείται με την εξέλιξη των υπολογιστών. Λόγω κόστους κάποιον υλικών ή κατασκευών δεν μπορούσαμε να πειραματιστούμε πάνω σε αυτά, οπότε έπρεπε να βρεθεί ένας τρόπος να δούμε πώς θα αντιδράσει σε συνθήκες που ορίζαμε και θέλαμε εμείς χωρίς να παρέμβουμε καθόλου σε αυτό. Εδώ μπαίνει η έννοια του μοντέλου, δηλαδή μπορούμε πλέον να δημιουργούμε το μοντέλο που θα θέλαμε να πειραματιστούμε σε ποικίλα περιβάλλοντα ώστε τα πειράματά μας να μην έχουν τον κίνδυνο καταστροφής της κατασκευή μας, κάτι που θα είχε κόστος επισκευής και χρόνο αποκατάστασής της.

# 2. ΜΟΝΤΕΛΟ ΠΕΙΡΑΜΑΤΟΣ

## 2.1 ΕΙΣΑΓΩΓΗ ΣΤΟ ΠΕΙΡΑΜΑ

Στη συγκεκριμένη πτυχιακή θα ασχοληθούμε με το μοντέλο :
ΨΗΦΙΑΚΟΣ ΕΛΕΓΧΟΣ ΚΛΙΣΗΣ ΕΛΙΚΟΦΟΡΟΥ ΠΛΑΤΦΟΡΜΑΣ

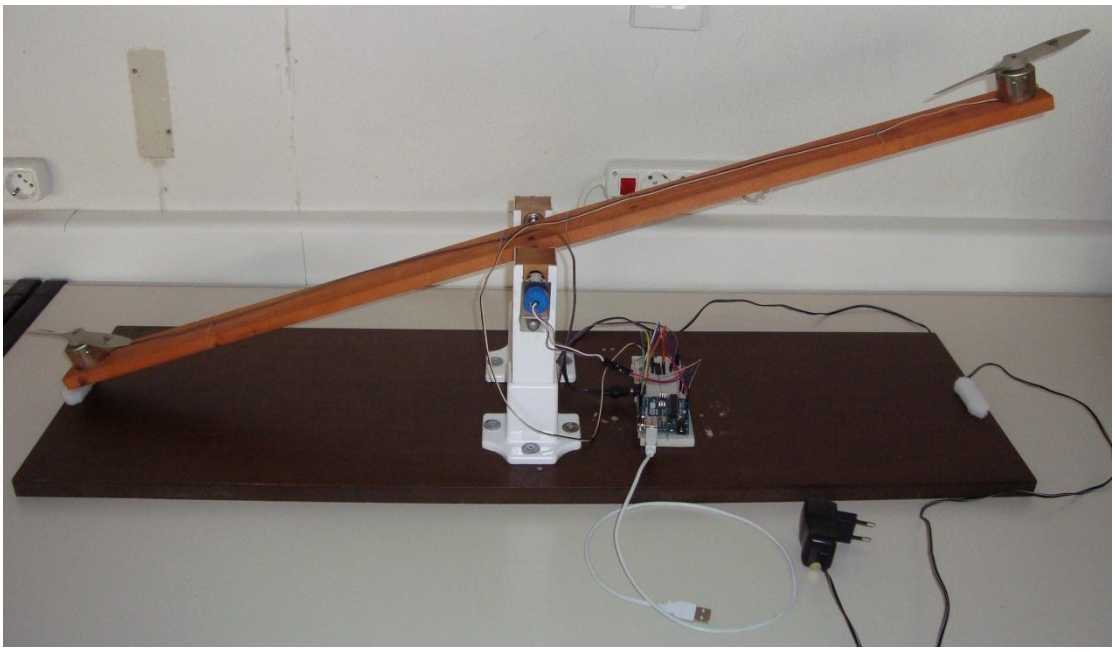
Θα γίνει σχεδίαση και υλοποίηση ενός εικονικού εργαστηρίου κατάλληλου για την υποστήριξη της διδασκαλίας των μαθημάτων ΣΑΕ Ι,ΣΑΕ ΙΙ και ΕΛΕΓΚΤΕΣ. Το εικονικό εργαστήριο θα υλοποιηθεί σε περιβάλλον Matlab/Simulink και θα αποτελείται από κατάλληλο περιβάλλον διεπαφής χρήστη (GUI-graphical user interface), που θα παρέχει την δυνατότητα ρύθμισης διαφόρων παραμέτρων, εκτέλεσης εικονικού πειράματος (με προσομοίωση), και παρακολούθησης του αποτελέσματος σε δύο κύριες μορφές : γραφικές παραστάσεις και σχεδιοκίνηση (animation) σε realtime.

Θα χρησιμοποιήσουμε στοιχεία και θα εισάγουμε το μοντέλο από την πτυχιακή εργασία που πραγματοποίησε με μεγάλη επιτυχία ό συμφοιτητής μας Καρασπύρος Δημήτριος (πιο αναλυτικά θα αναφερθώ στην πτυχιακή για κατανόηση του συστήματος).

Το σύστημα που θα χρησιμοποιήσουμε θα περιέχει έναν PID ελεγκτή ο οποίος με τις κατάλληλες ρυθμίσεις μέσω του GUI που κατασκεύασα θα μπορεί να διαμορφωθεί σε PD, PI καθώς και η λειτουργία του συστήματος χωρίς ελεγκτή. Θα μπορεί ο κάθε φοιτητής και ενδιαφερόμενος να πειραματιστεί με τις εισόδους του πειράματος, διαταραχές, διαφορετικούς ελεγκτές (PID, PD, PI ή και καθόλου ) καθώς και πολλά άλλα στοιχεία που έχουν προσαρμοστεί στο GUI και θα αναφερθούμε εκτενέστερα σε παρακάτω μέρος της πτυχιακής.

Το συγκεκριμένο πείραμα που θα ασχοληθούμε και θα δημιουργήσουμε την διεπαφή είναι το ‘’ΨΗΦΙΑΚΟΣ ΕΛΕΓΧΟΣ ΚΛΙΣΗΣ ΕΛΙΚΟΦΟΡΟΥ ΠΛΑΤΦΟΡΜΑΣ’’ από την πτυχιακή εργασία που πραγματοποίησε με επιτυχία ο συμφοιτητής μας Καρασπύρος Δημήτριος μαζί με τον καθηγητή Υφούλη Χρήστο.

Το σύστημα που θα ασχοληθούμε έχει την δομή που φαίνεται παρακάτω(σχ.1)



σχ1. Σύστημα ελικοφόρου πλατφόρμας

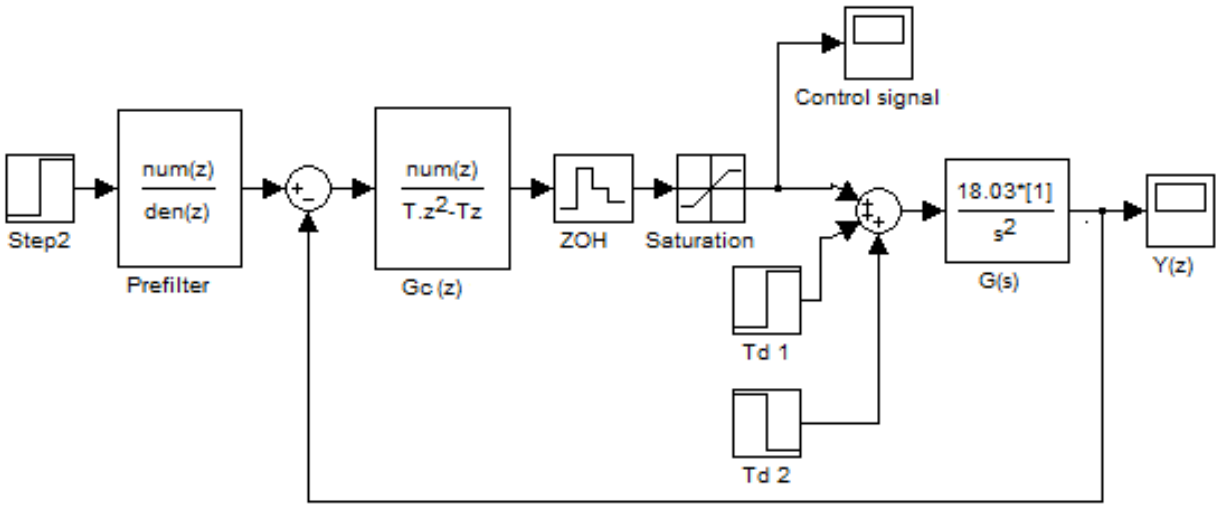
## 2.2 ΑΝΑΛΥΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Θα αναφερθούμε σε στοιχεία του συστήματος μέσα από την πτυχιακή του Δ. Καρασπύρου για μια αρχική κατανόηση του πειράματος.

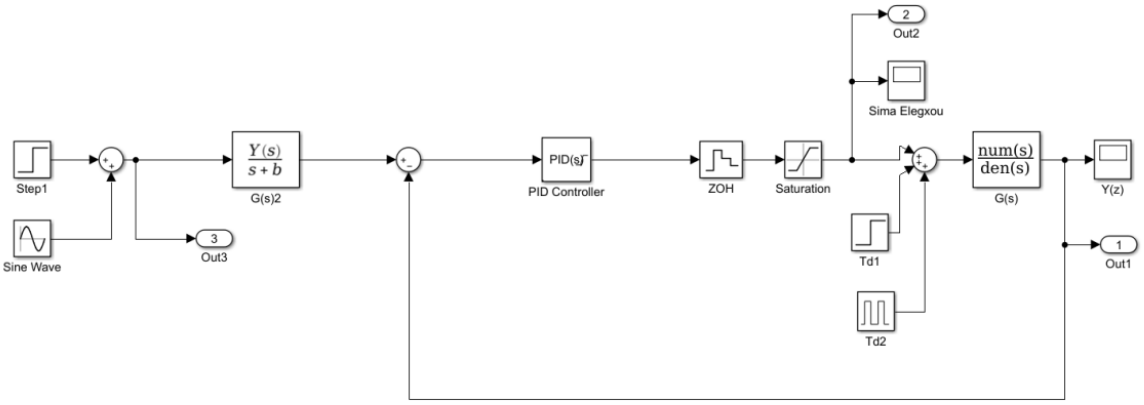
Στο σύστημα θα χρησιμοποιηθεί ελεγκτής τύπου PID. Οι PID ελεγκτές είναι πολύ δημοφιλείς στον κύκλο των σχεδιαστών συστημάτων. Αυτό οφείλεται στη λειτουργική τους απλότητα που επιτρέπει στους μηχανικούς να τους χρησιμοποιήσουν με απλό και άμεσο τρόπο.

Το σύστημα έχει δύο εισόδους, οι οποίες είναι οι δύο κινητήρες. Η έξοδος $Y(s)$ του συστήματος είναι η κλίση της σανίδας την οποία θα μεταβάλουμε εμείς μέσω του GUI δίνοντας μοίρες για την κλίση και με δυο διαφορετικές εισόδους (βηματική, ημιτονική).

Έχουμε το διάγραμμα βαθμίδων για σταθεροποιητικό έλεγχο με στιγμιαία διαταραχή, και επίσης βλέπουμε τις βαθμίδες $T_d$ 1 και $T_d$ 2 που δίνουν τη διαταραχή. Το σύστημα αυτό (σχ.2) με ορισμένες αλλαγές το χρησιμοποιήσαμε για να εισάγουμε το μοντέλο μας (σχ.3) στο Simulink.




σχ.2 Αρχικό σύστημα.




σχ.3 Σύστημα εισαγωγής στο Simulink

# 3. ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

## 3.1 ΕΙΣΑΓΩΓΗ ΣΤΟ MATLAB

Το Matlab (MatrixLaboratory) είναι ένα ενσωματωμένο και σύγχρονο υπολογιστικό περιβάλλον με τεχνικές γνώσεις το οποίο συνδυάζει τους αριθμητικούς υπολογισμούς και χρησιμοποιείται σε ερευνητικές εφαρμογές και πανεπιστημιακά μαθήματα.

Το Matlab περιέχει εκατοντάδες εντολές για μαθηματικούς υπολογισμούς και οπτικοποίηση δεδομένων που το κατατάσσουν σε ένα από τα ισχυρότερα εργαλεία στις φυσικές και μαθηματικές επιστήμες. Μπορεί να χρησιμοποιηθεί για τη σχεδίαση συναρτήσεων, για επίλυση εξισώσεων, για εξάσκηση στη στατιστική και για πολλά άλλα. Είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου και είναι συμβατό με προγράμματα που χρησιμοποιούν συγγενικές του γλώσσες προγραμματισμού, όπως FORTAN και C. Μπορεί να παράγει ήχο και «ζωντανά» γραφικά. Μπορεί να κάνει προσομοιώσεις και μοντελοποίηση (ειδικά αν έχει πρόσβαση όχι μόνο στο βασικό Matlab, αλλά ακόμη και στο «συνεργό» του, το simulinkκαι GUI). Μπορεί να ετοιμάσει υλικά για εξαγωγή στο Διαδίκτυο. Ακόμη, μπορεί να χρησιμοποιηθεί σε συνδυασμό με την επεξεργασία κειμένου και τις δυνατότητες δημοσίευσης του Microsoft Word στην επιφάνεια εργασίας, για να συνδυάσει μαθηματικούς υπολογισμούς με κείμενα και γραφικά, ώστε να παράγει ένα ενσωματωμένο, διαδραστικό και γεμάτο δυνατότητες αρχείο.

Ένα τόσο εξελιγμένο πρόγραμμα εμπεριέχει πολλά χαρακτηριστικά για χρησιμοποίηση από τον χρήστη και πολλές επιλογές. Υπάρχουν πραγματικά εκατοντάδες χρήσιμες οδηγίες στη διάθεση του χρήστη. Η βοήθεια του Matlab (Matlab Help) είναι τεράστια και μπορεί να χρησιμοποιηθεί από τον εκάστοτε χρήστη με ευκολία. Οι βασικές αναφορές, είτε το The MathWorks User's Guide για την παραγωγή, είτε οποιοσδήποτε άλλος ανταγωνιστής, περιέχουν μυριάδες πίνακες που περιγράφουν ένα ατελείωτο ρεύμα εντολών, επιλογών και χαρακτηριστικών για τα οποία ο χρήστης αναμένεται να μελετήσει ή να αποκτήσει πρόσβαση. Το Matlab είναι κάτι παραπάνω από μια αριθμομηχανή. Είναι ένα πολύ χρήσιμο και με πολλές δυνατότητες εργαλείο. Ακόμα κι αν κάποιος γνωρίζει λίγα για το Matlab μπορεί να το χρησιμοποιήσει για να πραγματοποιήσει υπέροχα πράγματα. Το δύσκολο σημείο όμως είναι να βρεθεί ποιες από τις εκατοντάδες εντολές, δεκάδες σελίδες βοήθειας και χιλιάδες αντικείμενα τεκμηρίωσης χρειάζονται να δει για να ξεκινήσει να το χρησιμοποιεί γρήγορα και αποτελεσματικά.

## 3.2 ΕΙΣΑΓΩΓΗ ΣΤΟ SIMULINK

Το SIMULINK είναι ένα λογισμικό πακέτο που επιτρέπει τη μοντελοποίηση, προσομοίωση και ανάλυση δυναμικών συστημάτων. Υποστηρίζει γραμμικά και μη γραμμικά συστήματα, μοντελοποιημένα σε συνεχή ή διακριτό χρόνο, ή ακόμη και υβριδικά συστήματα (εν μέρει μοντελοποιημένα σε συνεχή και εν μέρει σε διακριτό χρόνο). Υποστηρίζονται ακόμη συστήματα με τμηματικά διαφορετικούς χρόνους δειγματοληψίας.

Για τη μοντελοποίηση, το SIMULINK παρέχει στον χρήστη ένα γραφικό περιβάλλον διεπαφής (GUI) που του επιτρέπει την κατασκευή μοντέλων, χρησιμοποιώντας λειτουργίες click-and-drag του ποντικιού. Το SIMULINK περιλαμβάνει ένα τεράστιο πλήθος από βιβλιοθήκες δομικών στοιχείων (blocks), τα βασικότερα από τα οποία είναι οι πηγές (sources), τα στοιχεία «απορρόφησης» (sinks), τα συνεχή γραμμικά στοιχεία, τα μη γραμμικά στοιχεία και τα στοιχεία σημάτων και συστημάτων. Επίσης ο χρήστης έχει την δυνατότητα να τροποποιεί και να δημιουργεί νέα δομικά στοιχεία.

Τα μοντέλα SIMULINK είναι ιεραρχικά (ένα μοντέλο μπορεί να περιέχει μπλοκ τα οποία περιέχουν με τη σειρά τους άλλα μπλοκ), έτσι μπορούν να ιδωθούν σε διάφορα επίπεδα. Ένα σύστημα που έχει ιεραρχική δομή μπορεί να ιδωθεί αρχικά σε υψηλό επίπεδο ως ένα σύνολο διασυνδεμένων υποσυστημάτων, κάθε ένα από τα οποία μοντελοποιείται ως ένα μπλοκ. Στη συνέχεια, κάνοντας διπλό κλικ με το ποντίκι στα επί μέρους μπλοκ, ο χρήστης μπορεί να κατέβει σε χαμηλότερα επίπεδα ώστε να δει αυξανόμενους βαθμούς λεπτομέρειας.

Με τη δημιουργία ενός μοντέλου, μπορεί ο χρήστης να προχωρήσει στη προσομοίωση του, χρησιμοποιώντας μια από τις διάφορες μεθόδους ολοκλήρωσης που παρέχει το SIMULINK. Η παρακολούθηση των αποτελεσμάτων και της προσομοίωσης μπορεί να γίνει με την χρησιμοποίηση παλμογράφου (scopes)και άλλα μπλοκ απεικόνισης καθώς αυτά εξελίσσονται. Επιπλέον, είναι δυνατή η εξαγωγή αποτελεσμάτων της προσομοίωσης στο χώρο εργασίας της MATLAB για περαιτέρω επεξεργασία. Είναι ακόμη δυνατή η χρήση του SIMULINK για προσομοίωση αλλά και έλεγχο συστημάτων σε πραγματικό χρόνο, μέσω της εργαλειοθήκης πραγματικού χρόνου (RealTimeWorkshop).

Περαιτέρω πληροφορίες για τα δομικά στοιχεία και τις βιβλιοθήκες του SIMULINK είναι διαθέσιμες μέσω της βοήθειας της MATLAB.
Αφότου καθοριστεί ένα πρότυπο ο χρήστης έχει την δυνατότητα να δει τα αποτελέσματα της προσομοίωσης καθώς αυτή τρέχει. Επίσης, του είναι επιτρεπτό να αλλάξει τις παραμέτρους και να ξαναδεί τα αποτελέσματα της προσομοίωσης που τον ενδιαφέρουν.

Γενικά το simulink μας ενθαρρύνει να χτίσουμε πρότυπα από την αρχή ή να κάνουμε διαφοροποιήσεις σε υπάρχοντα. Έχουμε στιγμιαία πρόσβαση σε όλα τα υπάρχοντα εργαλεία του Matlab, έτσι ώστε να μπορούμε να πάρουμε διάφορα αποτελέσματα να τα αναλύσουμε και να τα απεικονίσουμε σε γραφική παράσταση.

Με τη βοήθεια του simulink μπορούμε να μετατρέψουμε τον υπολογιστή μας σε ένα εργαστήριο για τη διαμόρφωση και την ανάλυση συστημάτων που αλλιώς ήταν αδύνατο να μελετήσουμε τόσο αναλυτικά και εύκολα τη συμπεριφορά, τους όπως για παράδειγμα το σύστημα φρεναρίσματος ενός αυτοκινήτου. Είναι αρκετά ευχάριστο στο χρήστη καθώς τον εισάγει σε ένα γραφικό περιβάλλον όπου εκεί μπορεί να σχεδιάσει τα συστήματά του χρησιμοποιώντας το ποντίκι και σέρνοντας αντικείμενα στο χώρο εργασίας που του προσφέρεται.

Το simulink είναι ένα λογισμικό ιδιαίτερα πρακτικό και αυτό το αποδεικνύει το ότι το χρησιμοποιούν χιλιάδες μηχανικοί σε όλο τον κόσμο για να προσομοιώσουν και να λύσουν πραγματικά προβλήματα. Στο Simulink το μόνο που απέμεινε είναι τα σύμβολα που λειτουργούν σαν διασύνδεση. Η εσωτερική αναπαράσταση και όλες οι πράξεις γίνονται σε καθαρά ψηφιακή βάση. Διατίθεται μια πληθώρα από έτοιμες συναρτήσεις, απεικονίσεις, και άλλες ευκολίες.

## 3.3 ΕΙΣΑΓΩΓΗ ΣΤΟ GUI

Μια γραφική διεπαφή χρήστη (GUI) είναι ένα γραφικό περιβάλλον εργασίας σε ένα πρόγραμμα που παρέχει στον χρήστη ένα οικείο περιβάλλον που θα διατελέσει την εργασία που επιθυμεί. Ένα καλό GUI μπορεί να κάνει τα προγράμματα ευκολότερα στη χρήση, παρέχοντάς τους μια συνεπή εμφάνιση και με διαισθητικό έλεγχο, όπως κουμπιά (buttons), λίστες (List boxes), ρυθμιστικά (sliders), μενού (Menus) και ούτω καθεξής. Το GUI θα πρέπει να συμπεριφέρεται με ένα κατανοητό και προβλέψιμο τρόπο, ώστε ο χρήστης να γνωρίζει τι να περιμένει, όταν εκτελεί μια ενέργεια. Για παράδειγμα, όταν σε ένα κουμπί ενεργοποιείται από τον χρήστη, το GUI θα πρέπει να κινήσει τη διαδικασία που περιγράφεται στην ετικέτα του κουμπιού.

Οι είσοδοι είναι γνωστές ως ***γεγονότα(events)***, και ένα πρόγραμμα που αποκρίνεται στα γεγονότα ονομάζεται ***γεγονοστρεφές (event driven)***. Τα τρία κυρίαρχα στοιχεία που απαιτούνται για τη δημιουργία ενός GUI στο Matlab είναι τα εξής:

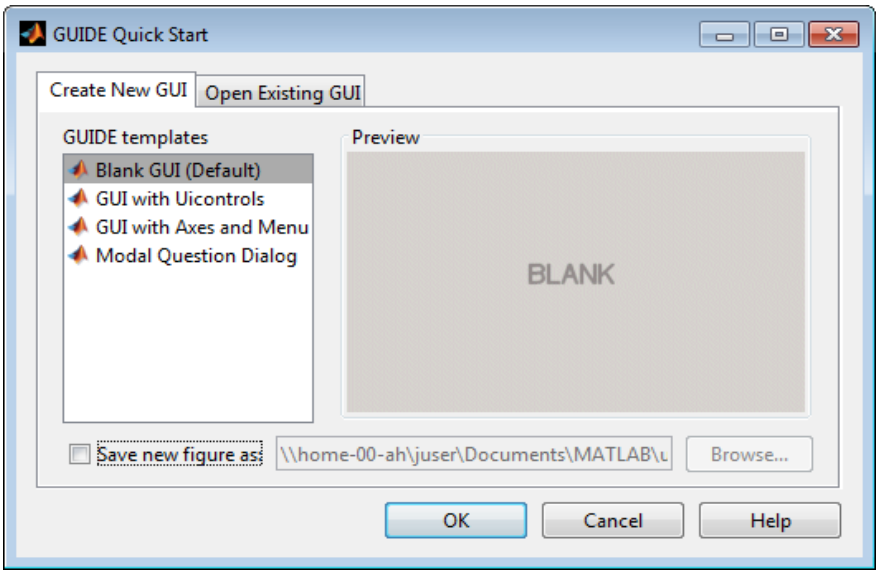
1. Συστατικά (Components). Κάθε στοιχείο στο GUI του Matlab είναι ένα γραφικό στοιχείο (κουμπιά, λίστες, πλαίσια κειμένου κλπ).
   Στους τύπους των στοιχείων περιλαμβάνονται γραφικοί έλεγχοι (Pushbuttons, edit boxes κλπ), στατικά στοιχεία όπως πλαίσια (frames) και συμβολοσειρές κειμένου (text strings), μενού (Menus), και άξονες (axes). Τα δυο πρώτα δημιουργούνται μέσω της συνάρτησης uicontrol, και τα μενού μέσω των συναρτήσεων uimenu και uicontextmenu. Οι άξονες οι οποίοι χρησιμοποιούνται για την παρουσίαση γραφικών δεδομένων, δημιουργούνται μέσω της συνάρτησης axes.

2. Σχήματα (figures). Τα συστατικά ενός GUI πρέπει να διατάσσονται μέσα σε ένα σχήμα, το οποίο είναι ένα παράθυρο στην οθόνη του υπολογιστή. Στο παρελθόν, η δημιουργία των σχημάτων γινόταν αυτόματα, κάθε φορά που υπήρχε απεικόνιση δεδομένων. Ωστόσο, τα κενά σχήματα μπορούν να δημιουργηθούν μέσω της συνάρτησης figure και μπορούν να χρησιμοποιηθούν για να υποστηρίξουν οποιοδήποτε συνδυασμό των components.

3. Επανακλήσεις (callbacks). Τέλος, θα πρέπει να υπάρχει κάποιος τρόπος να εκτελεστεί μια ενέργεια εάν ο χρήστης κάνει κλικ σε ένα κουμπί, ή πληκτρολογήσει πληροφορίες. Ένα κλικ του ποντικιού ή το πάτημα ενός κουμπιού είναι ένα γεγονός (event) και το Matlab θα πρέπει να ανταποκρίνεται σε κάθε περίπτωση, αν το πρόγραμμα πρέπει να εκτελέσει τη λειτουργία του. Για παράδειγμα αν ένας χρήστης κάνει κλικ σε ένα κουμπί, το γεγονός αυτό θα πρέπει να προκαλέσει τον κώδικα του Matlab που υλοποιεί τη λειτουργία του κουμπιού αυτού, να εκτελεστεί. Ο εκτελέσιμος κώδικας που ανταποκρίνεται σε ένα γεγονός ονομάζεται επανάκληση (callback). Πρέπει να υπάρχει ένα callback που θα εφαρμόσει τη λειτουργία του σε κάθε γραφικό στοιχείο του GUI.

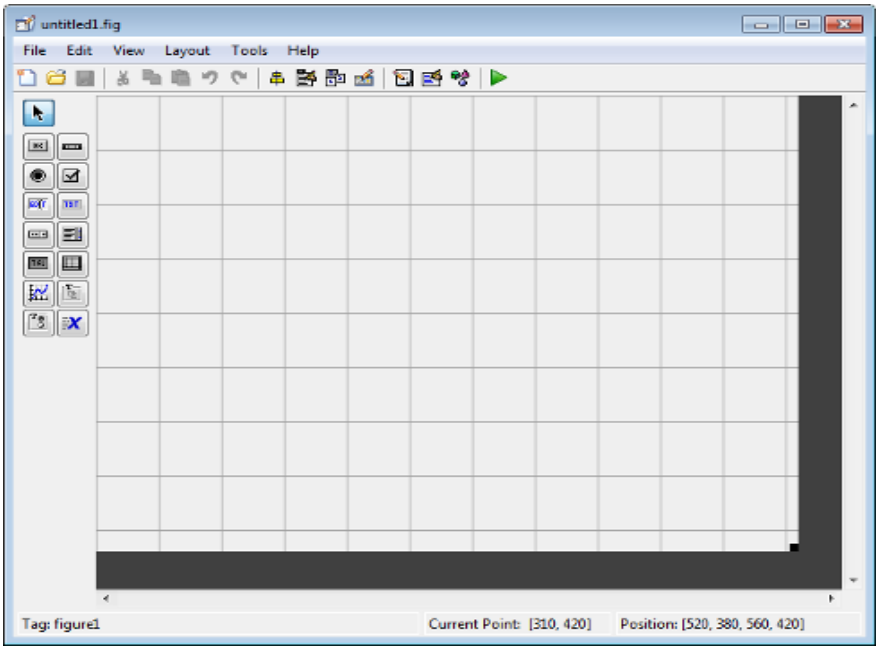
### 3.3.1 ΑΝΑΛΥΣΗ GUI

Για τη δημιουργία ενός project αρκεί να πληκτρολογήσουμε στο workspace του matlab ‘’guide’’ και μας εμφανίζεται το παράθυρο επιλογής project (σχ.4) .




σχ.4 Επιλογή project δημιουργείας GUI

Με την δημιουργία, πατώντας το buttonOK, θα μας εμφανιστεί ο χώρος εργασίας (σχ.5) που θα μπουν τα στοιχεία για την δημιουργία ενός project.

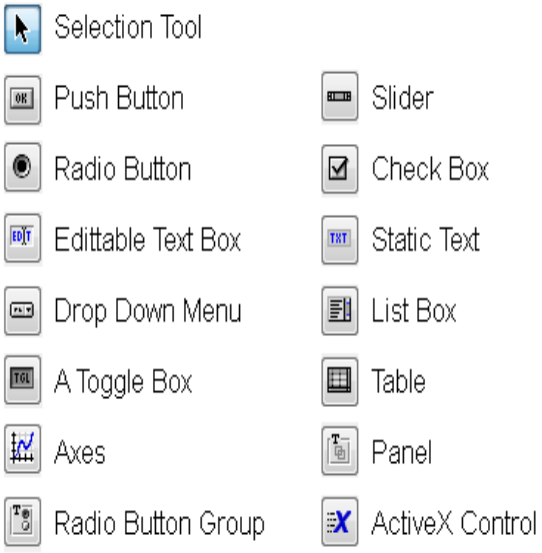



σχ.5 Χώρος εργασίας GUI

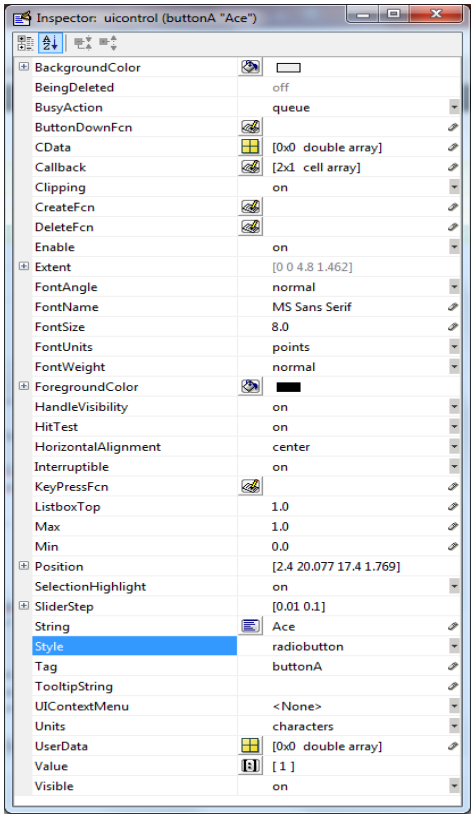
## 3.3.2 ΑΝΑΛΥΣΗ ΣΤΟΙΧΕΙΩΝ GUI

Για να εισάγουμε στοιχεία απλά τα πιάνουμε με το ποντίκι από την αριστερή στήλη που υπάρχουν τα στοιχεία και τα σέρνουμε στο χώρο εργασίας του GUI.

Κάθε στοιχείο (σχ.6) συνοδεύεται από ένα ΄΄propertyinspector΄΄ (σχ.7) το οποίο έχει την παρακάτω μορφή:




σχ.6 στοιχεία GUI

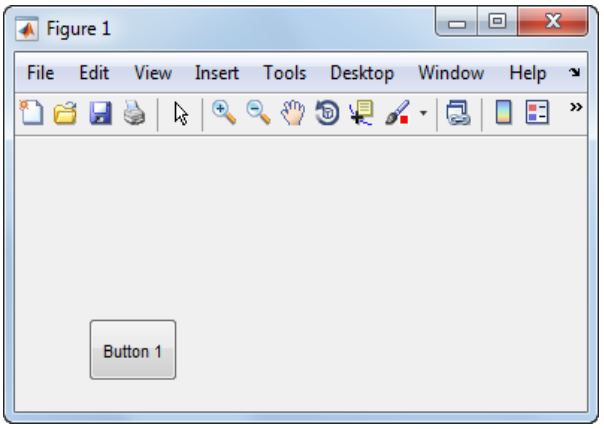



σχ.7 propertyinspector των στοιχείων

## 3.3.3 ΣΥΝΑΡΤΗΣΕΙΣ ΣΤΟΙΧΕΙΩΝ GUI

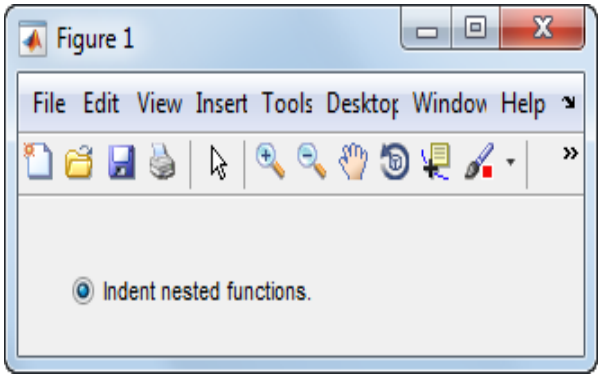
Θα δούμε αναλυτικά το κάθε στοιχείο του GUI και τις συναρτήσεις που δημιουργούνται στο editor του matlab οπού εκεί μπορούμε να επεξεργαστούμε την λειτουργία τους.

- PUSH BUTTON



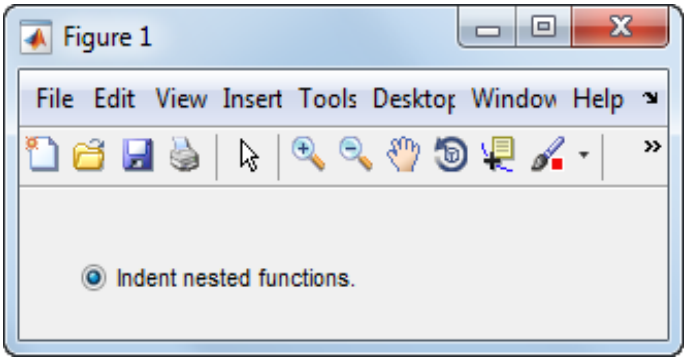

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in radiobutton1.
```

- RADIO BUTTON



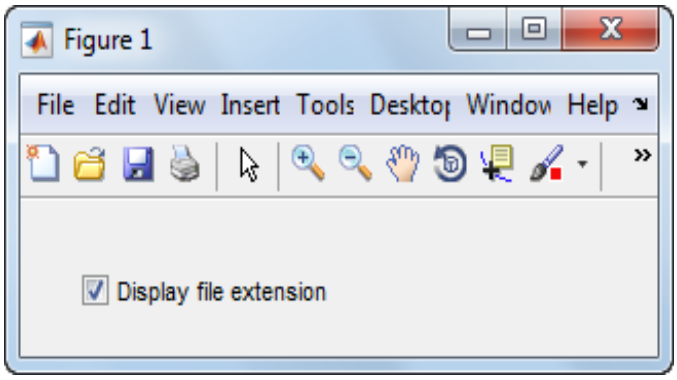

```
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of radiobutton1
```

- TOGGLE BUTTON



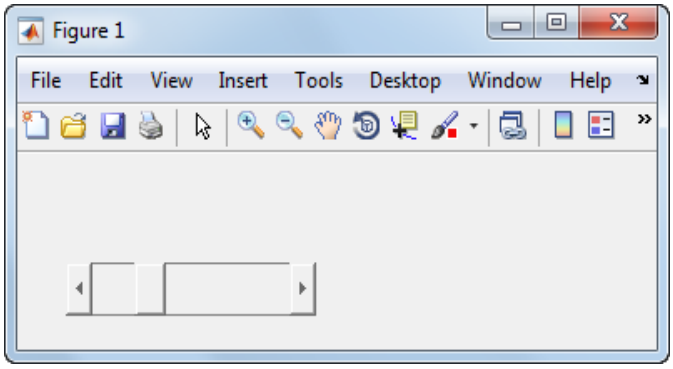

```
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of togglebutton1
```

- CHECK BOX



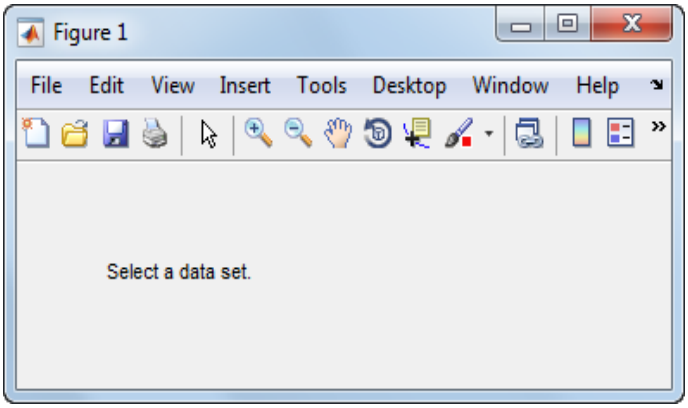

```
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of checkbox1
```

- SLIDER



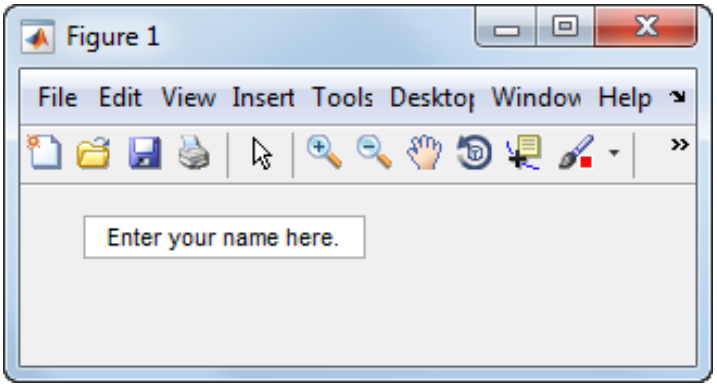

```
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
% get(hObject,'Min') and get(hObject,'Max') to determine range
of slider
```

- STATIC TEXT



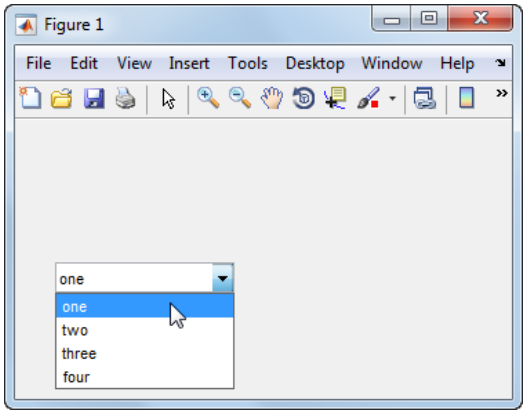

**Στα statictext δεν δημιουργείται καμία συνάρτηση. Το κείμενο μπορεί να γραφτεί απευθείας στο GUI.**

- EDITABLE TEXT FIELD



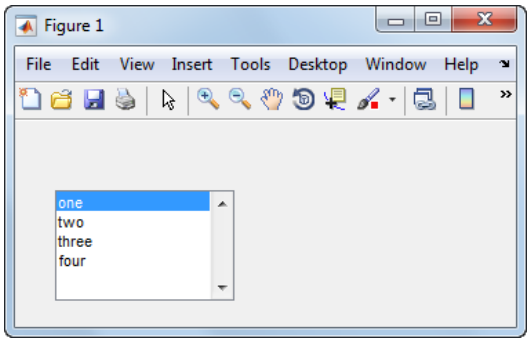

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

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


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

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

- POP – UP MENU



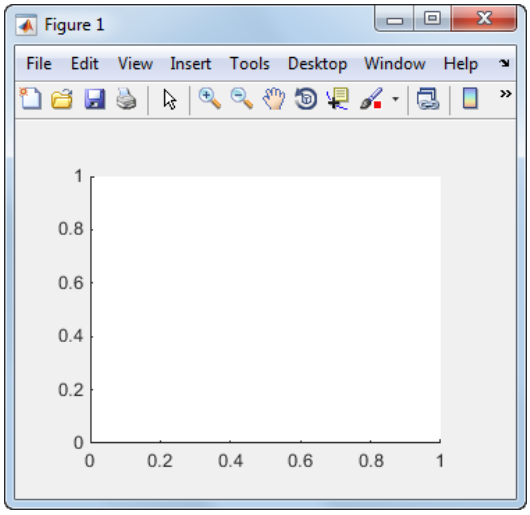

```
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

- LIST BOX



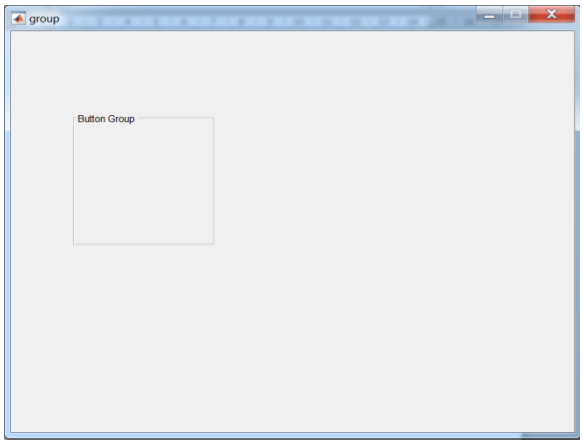

```
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%contents{get(hObject,'Value')} returns selected item from listbox1
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

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

- AXES




Στα axes δεν δημιουργείται κάποια συνάρτηση, μέσω ορισμένου κώδικα μπορούμε να εισάγουμε το τι θα σχεδιάζεται και με ποιο τρόπο.

- BUTTON GROUP
  Κάνουμε ομαδοποίηση κουμπιών για αισθητικούς λόγους και διαχώρισης κατηγορίας εκτελέσεων στο GUI.



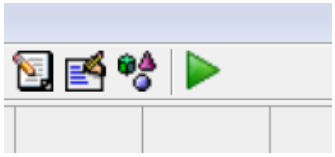

```
% --- Executes just before group is made visible.
function group_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles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

% Choose default command line output for group
handles.output = hObject;

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

## 3.3.4 ΔΗΜΙΟΥΡΓΙΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΑΡΧΕΙΩΝ

### GUI

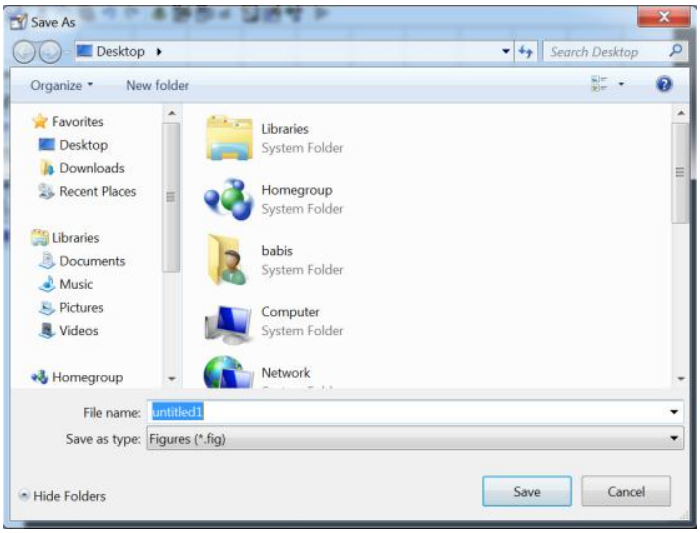
Οπότε με την δημιουργία ενός προγράμματος του GUI και την εισαγωγή στοιχείων δημιουργείται αυτόματα ένας κώδικας στο editor.

Με την συγγραφή του κώδικα που επιθυμούμε για την εργασία και την ολοκλήρωσή του πατώντας το κουμπί RunFigure (Ctrl + T) (σχ.8).



σχ.8 RunFigurebutton

Θα ζητηθεί η αποθήκευση (σχ.9) του προγράμματος, επιλέγουμε το path που θα αποθηκευτεί και το όνομα του αρχείου που επιθυμούμε.



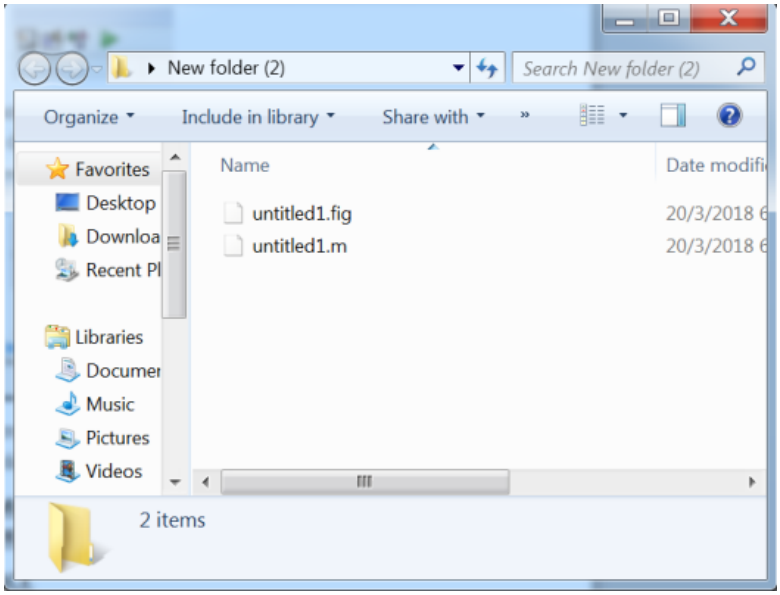

σχ.9 αποθήκευση project

Μετά την αποθήκευση δημιουργούνται δυο αρχεία (σχ.10) με το όνομα που δώσαμε και με την κατάληξη .fig και .m . Το matlab επιτρέπει την συγγραφή δυο τύπων προγραμμάτων:

**Τα .mfiles:**

- **Scripts** –Τα scriptfiles είναι φάκελοι προγράμματος με επέκταση .m. Σε αυτούς τους φακέλους γράφεις μια σειρά από κώδικες οι οποίοι θέλουμε να εκτελούνται μαζί. Τascripts δεν δέχονται εισόδους και δεν επιστρέφουν εξόδους. Λειτουργούν σε συνεργασία από δεδομένα από το workspace.

- **Functions** – Τα functionsfiles είναι επίσης φάκελοι προγράμματος με επέκταση .m. Τα Functions μπορούν να δεχτούν εισόδους και να επιστρέψουν εξόδους. Οι εσωτερικές μεταβλητές χρησιμοποιούνται μόνο τοπικά.




σχ.10 Αρχεία του προγράμματος που δημιουργήσαμε

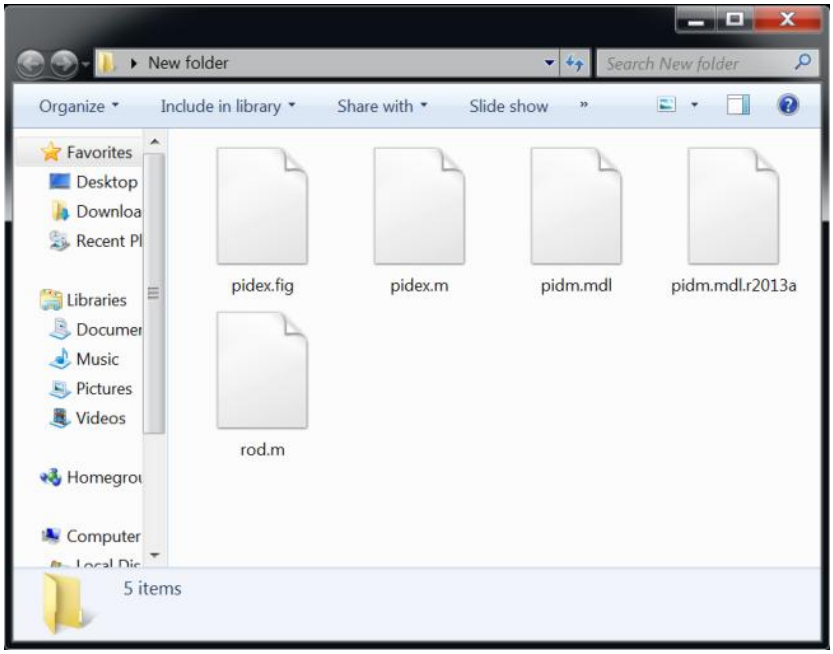
**Τα .fig files:**

Στα .fig files αποθηκεύεται το μοντέλο μας στο GUI.

## 4. ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΑΝΑΛΥΣΗ ΤΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

Το πρόγραμμά μας απαρτίζεται από τους φακέλους (σχ.11) :

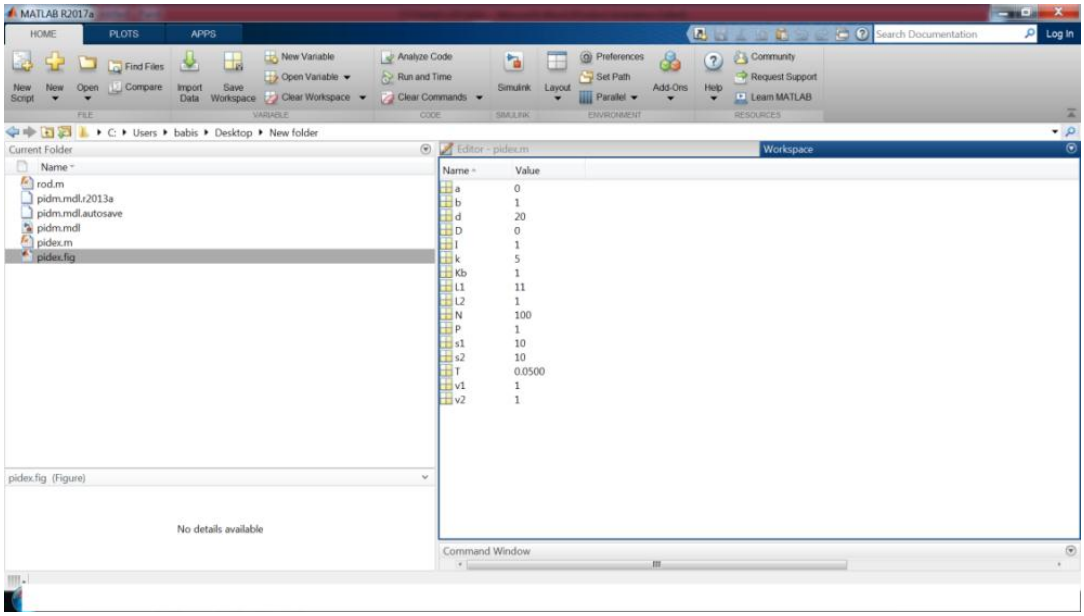
1. **Pidex.fig** (πρόγραμμα GUI)
2. **Pidex.m** (κώδικας προγράμματος pidex)
3. **pidm.mdl** (μοντέλο simulink)
4. **pidm.mdl.r2013a**(κρυπτογραφημένος κώδικας του κώδικά μας)
5. **rod.m** (κώδικας προγράμματος συνάρτησης rod)




σχ.11 Φάκελοι προγράμματος της πτυχιακής εργασίας

## 4.1 ΑΝΑΛΥΣΗ WORKSPACE

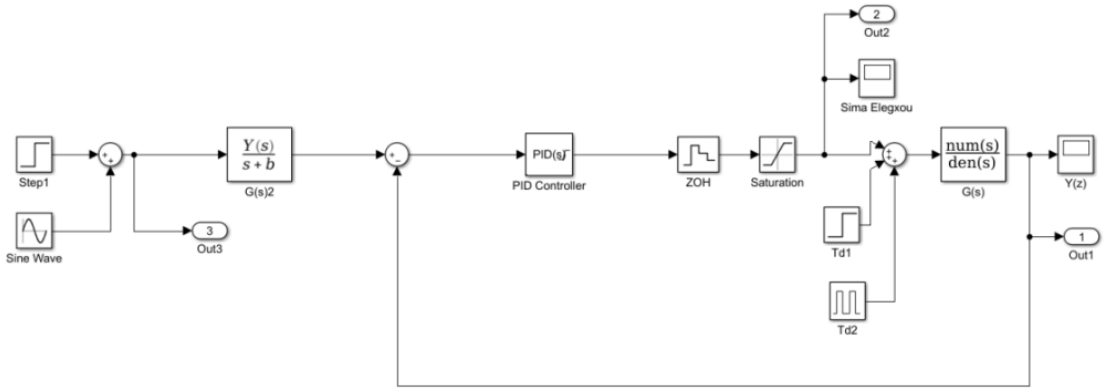
Στο Workspace δημιουργούμε ουσιαστικά τον συνδετικό κρίκο ανάμεσα στο GUI και στο Simulink. Αποθηκεύοντας μεταβλητές που χρησιμοποιούμε και μεταβάλλονται στο GUI στο Workspace δίνουμε την δυνατότητα στο Simulink να χρησιμοποιήσει αυτές τις μεταβλητές ειδάλλως δεν υπάρχει τρόπος επικοινωνίας μεταξύ των δύο σχ.12 .




σχ.12 Workspace και οι μεταβλητές που αποθηκεύονται σε αυτό.

## 4.2 ΑΝΑΛΥΣΗ SIMULINK

Στο Simulink χρησιμοποιούμε το σύστημα του ελικοφόρου από την πτυχιακή του συμφοιτητή Καρασπύρου με ορισμένες αλλαγές και τροποποιήσεις (σχ.13) ώστε να έχουμε περισσότερες μεταβλητές που μπορούμε να μεταβάλουμε. Αυτό μας δίνει την δυνατότητα να έχουμε την μέγιστη κατανόηση του συστήματος μας σε μεγαλύτερο φάσμα και με περισσότερες λεπτομέρειες. Φυσικά μπορούν να προστεθούν και παραπάνω μεταβλητές, αλλά στην συγκεκριμένη εργασία θα ασχοληθούμε με ορισμένες από αυτές.



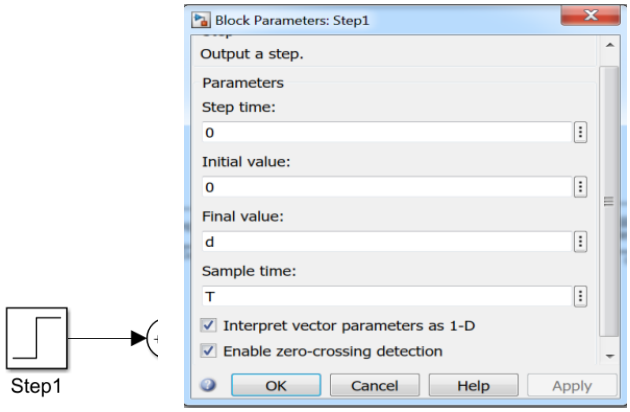

σχ.13 Σύστημα Simulink της εργασίας

## 4.2.1 ΑΝΑΛΥΣΗ ΣΤΟΙΧΕΙΩΝ ΤΟΥ SIMULINK

- **ΕΙΣΟΔΟΙ ΣΥΣΤΗΜΑΤΟΣ**

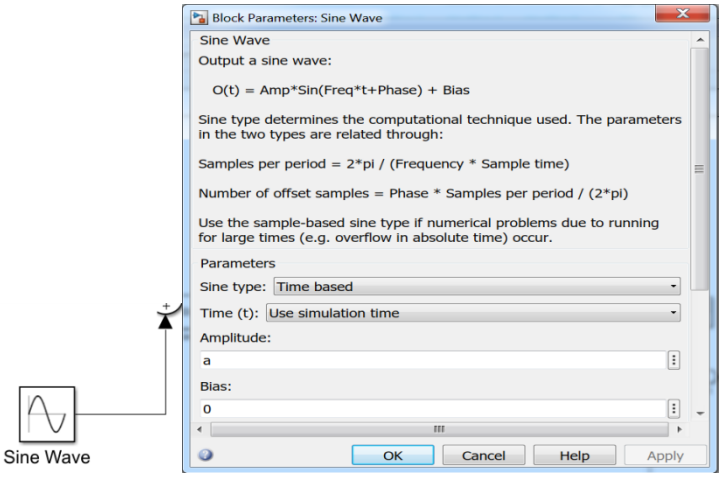
Χρησιμοποιούμε δυο εισόδους στο σύστημα, μία βηματική και μία ημιτονική.

- **ΕΙΣΟΔΟΣ ΒΗΜΑΤΙΚΗ:**




Μας δίνει έναν παλμό για είσοδο και στον οποίο μπορούμε να μεταβάλουμε από το GUI το πλάτος του βήματος από το finalvalue Στον κώδικα το όνομα μεταβλητής είναι dValue.

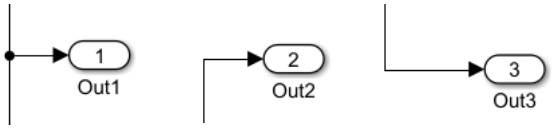
- **ΕΙΣΟΔΟΣ ΗΜΙΤΟΝΙΚΗ:**

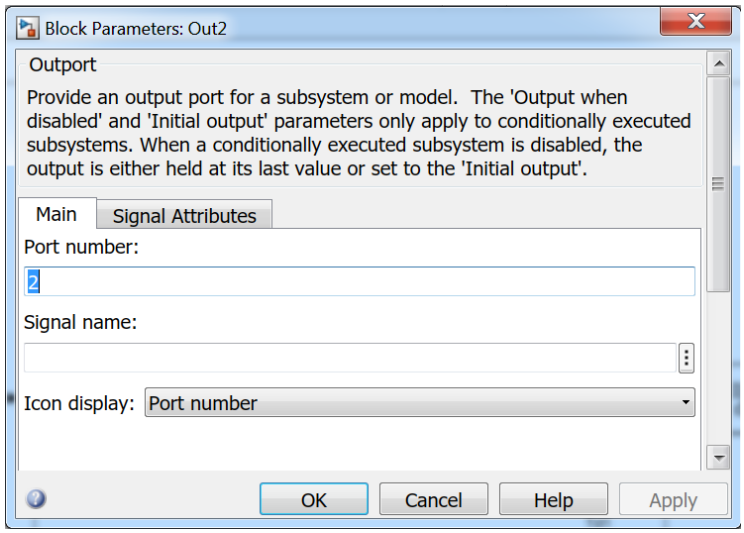



Μας δίνει για είσοδο ένα ημίτονο στο οποίο μπορούμε να από το GUI το πλάτος του παλμού Amplitute. Στον κώδικα το όνομα μεταβλητής είναι dValue.

- **<u>ΕΞΟΔΟΙ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΣΧΕΔΙΑΣΗ ΤΩΝ AXES</u>**

Χρησιμοποιούμε τρία block για να παίρνουμε τις τιμές από την είσοδο, την έξοδο και το σήμα ελέγχου και να τα μεταφέρουμε στο πρόγραμμα ώστε να χρησιμοποιηθούν για την σχεδίαση των γραφημάτων από τις συναρτήσεις.

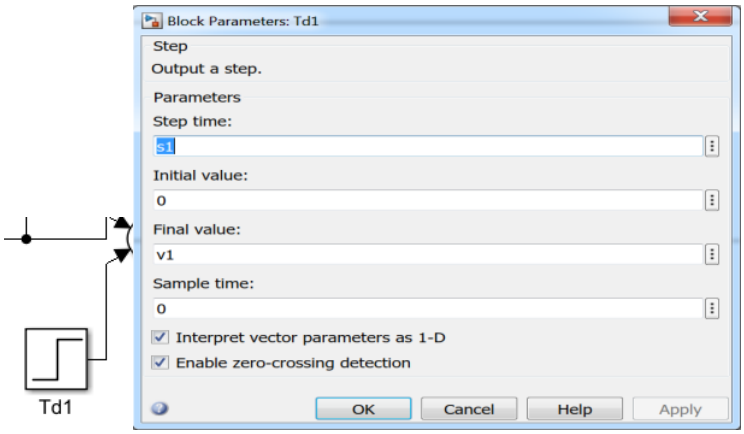


- **ΔΙΑΤΑΡΑΧΕΣ**

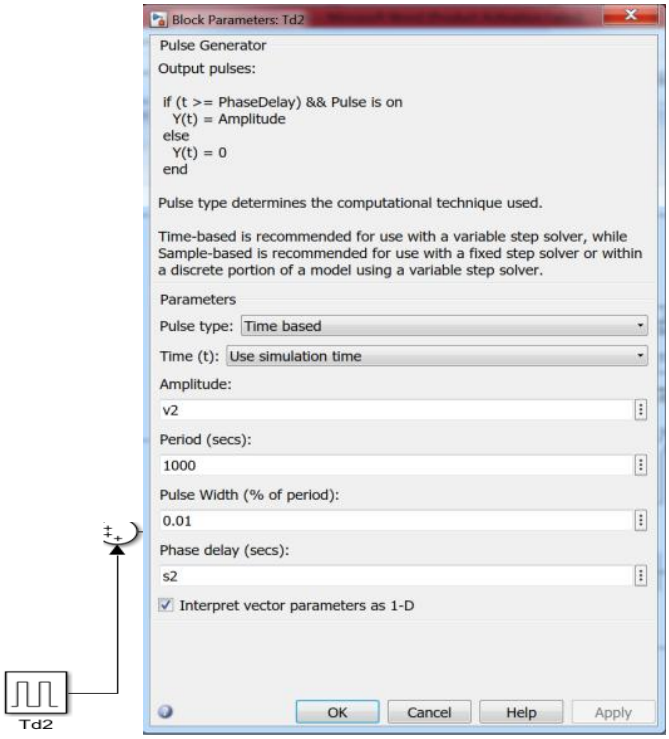
Χρησιμοποιούμε διαταραχές στο σύστημά μας για να προσομοιώσουμε την άσκηση δυνάμεων στο σύστημά μας και για έλεγχο του ελεγκτή μας σε απότομες μεταβολές. Χρησιμοποιούμε 2 εισόδους σήματος, η μία είναι βηματική που η τελική της τιμή μένει σταθερή και η δεύτερη είναι παλμός.

- **ΒΗΜΑΤΙΚΗ ΔΙΑΤΑΡΑΧΗ:**



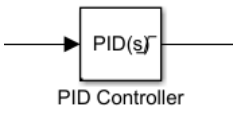

Μεταβάλουμε το πλάτος της βηματικής εισόδου με την μεταβλητή v1 η οποία είναι το Td1: value στο GUI και το χρόνο που θα πραγματοποιηθεί, ουσιαστικά δίνουμε καθυστέρηση, στο s1 η οποία είναι το Td1: time στο GUI.

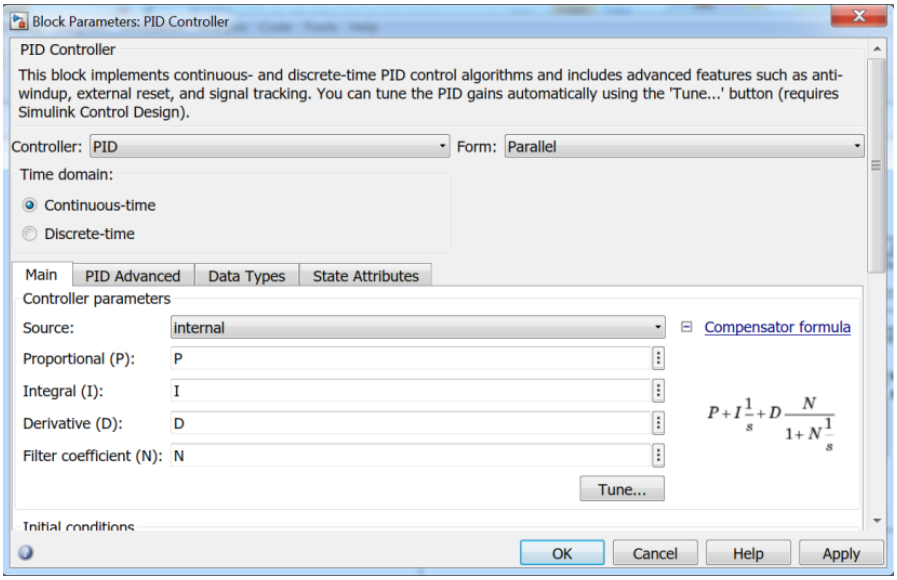
- **ΔΙΑΤΑΡΑΧΗ ΠΑΛΜΟΣ:**

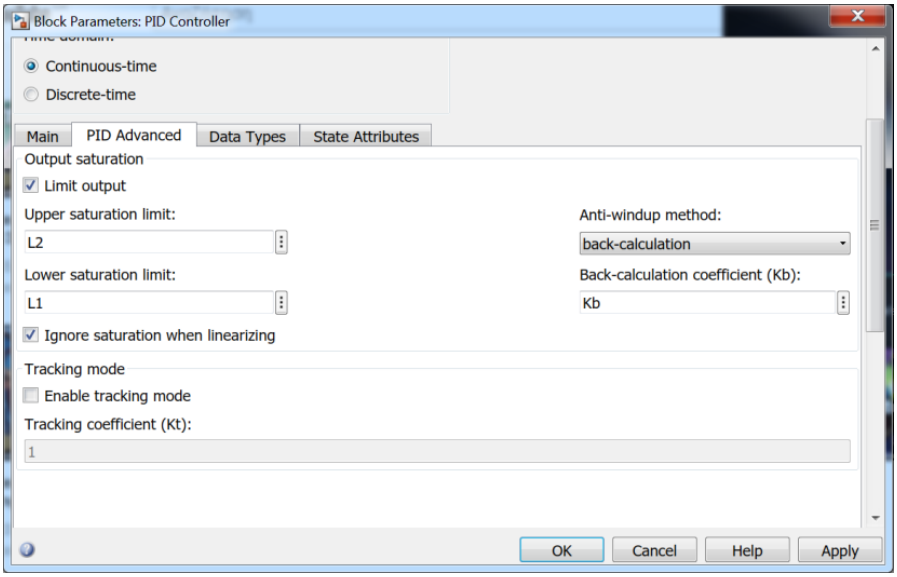



Χρησιμοποίησα είσοδο παλμού με πάρα πολύ μεγάλη περίοδο και πολύ μικρό πλάτος παλμού ώστε στο χρόνο που πειραματιζόμαστε να φαίνεται σαν ένας και μοναδικός παλμός. Μεταβάλουμε το πλάτος του παλμού στην μεταβλητή v2, στο GUI είναι Td2: value: και τον χρόνο που θα πραγματοποιηθεί ο παλμός με την μεταβλητή s2, στο GUI είναι Td2: time:

- **PID CONTROLLER**



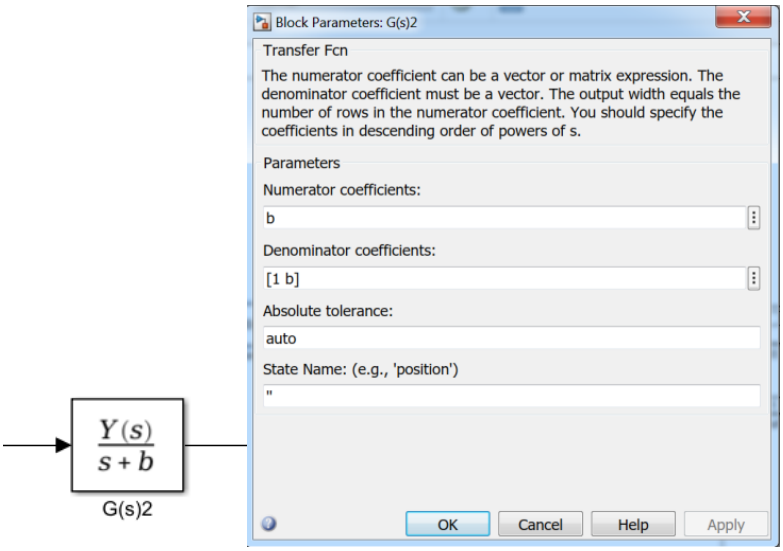




Χρησιμοποιούμε το blocktου PID για να ελέγξουμε το σύστημά μας, Στο συγκεκριμένο block γίνονται οι περισσότερες αλλαγές μας. Μπορούμε να αλλάξουμε τις τιμές P, I και D του ελεγκτή μας ώστε να πειραματιστούμε με όλους τους τύπους ελεγκτών απλά μηδενίζοντας τις μεταβλητές που θέλουμε (P, I, PI, PD, PID). Μπορούμε να αλλάξουμε τον τύπο και το βάρος κάθε μεταβλητή του ελεγκτή μας μεταβάλλοντας τα P, I και D στο GUI.Tofiltercoefficient είναι διαθέσιμο μόνο για ελεγκτές διακριτού χρόνου και μεταβάλλεται με την μεταβλητή N, όλα αυτά βρίσκονται στη καρτέλα main του PIDblock. Στην καρτέλα PID

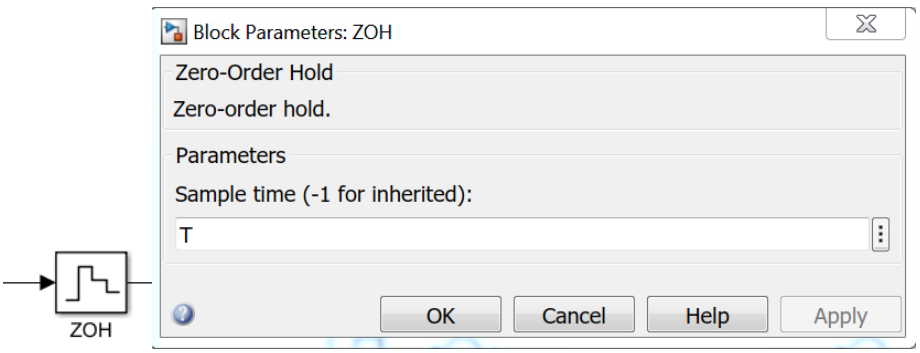
Advanced μπορούμε να μεταβάλλουμε το Upper saturation limit (L2), το Lower saturation limit (L1) και το Back-calculation coefficient (kb).

- **ΦΙΛΤΡΟ**



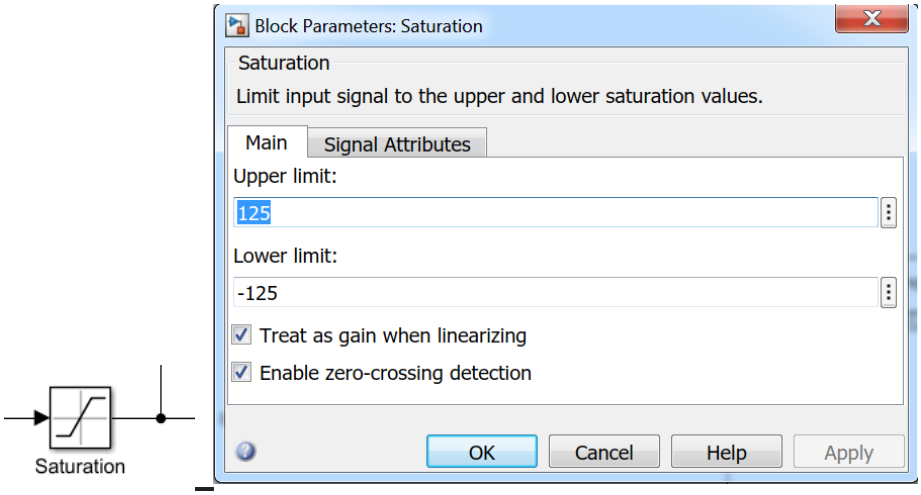

Με την χρήση ενός blockparameter δημιουργούμε ένα φίλτρο προπορίας. Μεταβάλλουμε τον αριθμητή και τον παρονομαστή μέσω της μεταβλητής b η οποία μεταβάλλεται μέσα από το GUI στο filtercoef (b).

- **BLOCK PARAMETER ZOH**



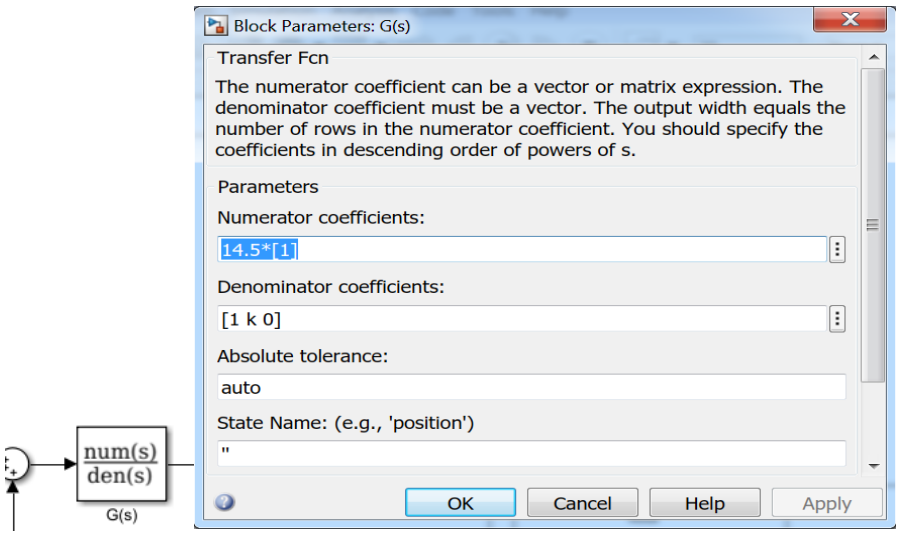

Το block ZOH διατηρεί την είσοδό του για την περίοδο δείγματος που καθορίζεται. Το μπλοκ δέχεται μία είσοδο και παράγει μία έξοδο. Κάθε σήμα μπορεί να είναι κλιμακωτό ή διανυσματικό. Εάν η είσοδος είναι ένα διάνυσμα, το μπλοκ περιέχει όλα τα στοιχεία του διανύσματος για την ίδια περίοδο δειγματοληψίας. Μπορούμε να ορίσουμε το sampletime με την μεταβλητή T, στο GUI είναι samplerate(T).

- **SATURATION DYNAMIC BLOCK**



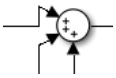

Το The Saturation Dynamic block παράγει ένα σήμα εξόδου που είναι η τιμή του σήματος εισόδου που περιορίζεται στις τιμές κορεσμού από τις θύρες εισόδου

- **ΣΥΝΑΡΤΗΣΗ ΜΕΤΑΦΟΡΑΣ**




Είναι η συνάρτηση μεταφοράς του συστήματός μας την οποία εξάγαμε μέσα από την πτυχιακή του Δ. Καρασπύρου και μεταβάλουμε την μεταβλητή k, στο GUI μεταβάλλεται από το k:

- **ΒΟΗΘΗΤΙΚΑ ΣΤΟΙΧΕΙΑ**



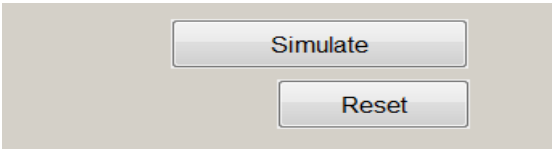
Χρησιμοποιούμε αθροιστές για την εισαγωγή σημάτων, διαταραχών και ανάδραση του συστήματος. Συνολικά χρησιμοποιήθηκαν τρεις αθροιστές.

## 4.3 ΑΝΑΛΥΣΗ GUI

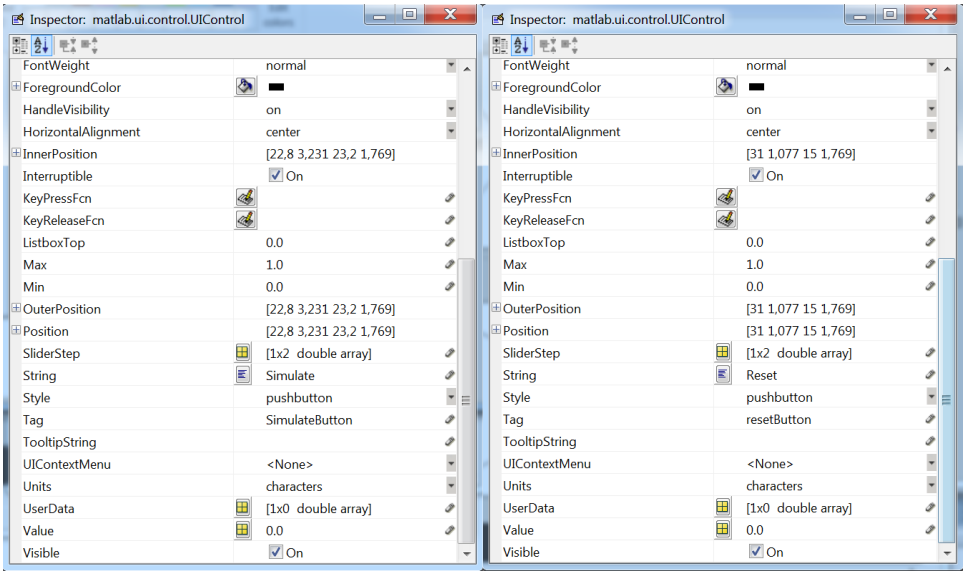
Θα αναφερθούμε στα στοιχεία που χρησιμοποιήθηκαν για την δημιουργία του προγράμματός μας στο GUI.

- **PUSH BUTTONS**

Είναι κουμπιά που με το πάτημά τους εκτελούν εντολές αλλά δεν παραμένουν πατημένα, για εκτέλεση της διεργασίας που εμπεριέχεται στο pushbutton πρέπει να πατηθούν ξανά από τον χρήστη. Χρησιμοποιήθηκαν δύο pushbuttons για τις λειτουργίες των του simulate και του reset.



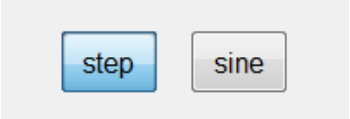

**Simulate Reset Inspector:**



Αλλάξαμε τα string που εμφανίζουν το κείμενο στο GUI επάνω στα buttons και τα tag τα οποία είναι SimulateButton και ResetButton.

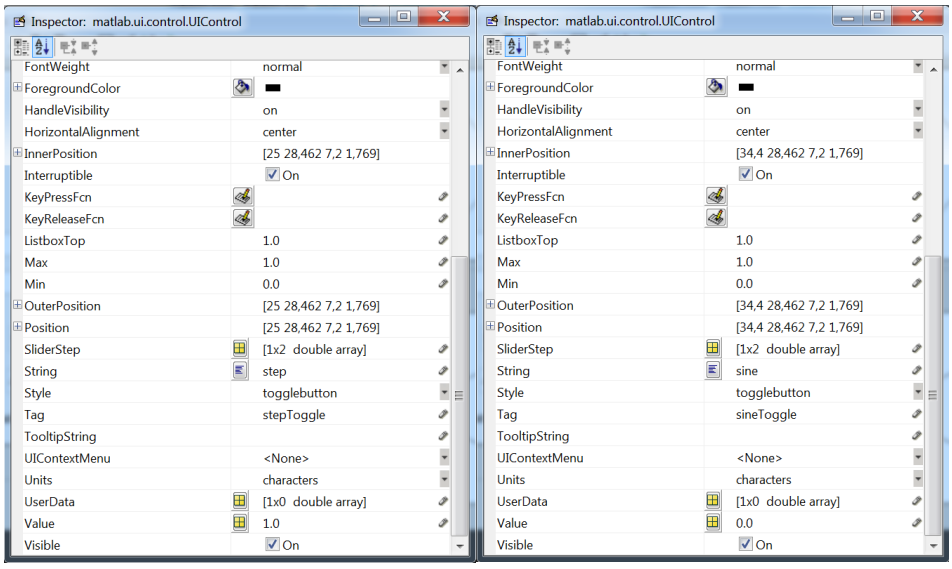
- **TOGGLE BUTTONS**

Είναι κουμπιά που με το πάτημά τους εκτελούνται εντολές, αλλά μετά το πάτημά τους παραμένουν ενεργά και ορίζουν μεταβλητές που θα αλλάζουν η κομμάτια κώδικα που θα εκτελούνται μόνο όταν το συγκεκριμένο button είναι πατημένο, μέχρι να πατηθεί κάποιο άλλο toggle button. Χρησιμοποιήθηκαν δύο toggle buttons για τις λειτουργίες των εισόδων μας,step και sine.




**Inspector:**

**StepSine**

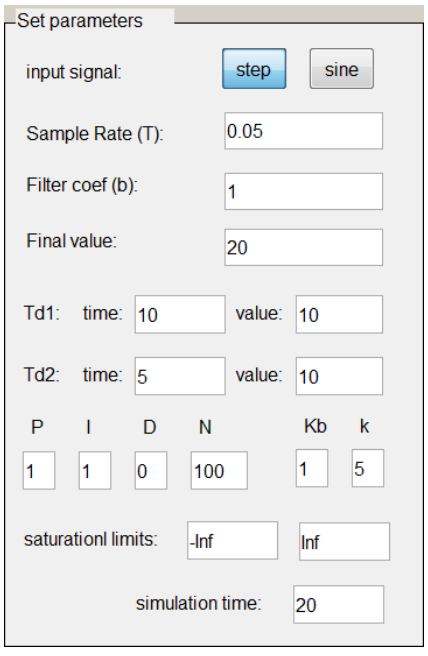



Αλλάξαμε τα string που εμφανίζουν το κείμενο στο GUI επάνω στα buttons και τα tag τα οποία είναι stepToggle και sineToggle.

- **EDIT BOX**

Τα editbox είναι ουσιαστικά πεδία κειμένου που μεταβάλουμε τις τιμές τους είτε από το πρόγραμμα με εντολές είτε με την επέμβαση του χρήστη. Στο συγκεκριμένο πρόγραμμα μόνο ο χρήστης μεταβάλει τις μεταβλητές οι οποίες είναι κείμενο και χρειάζεται η μετατροπή τους σε αριθμητικές τιμές ώστε να μπορούν να χρησιμοποιηθούν από το κώδικά μας και το Simulink. Χρησιμοποιήθηκαν δεκαέξι editbox για εισαγωγή παραμέτρων στο πρόγραμμά μας οι οποίες ελέγχουν τις μεταβλητές που βάλαμε για την διεξαγωγή πειραμάτων που επιθυμούμε να εκτελέσουμε. Τα editbox μεταβάλουν τις εξής μεταβλητές:

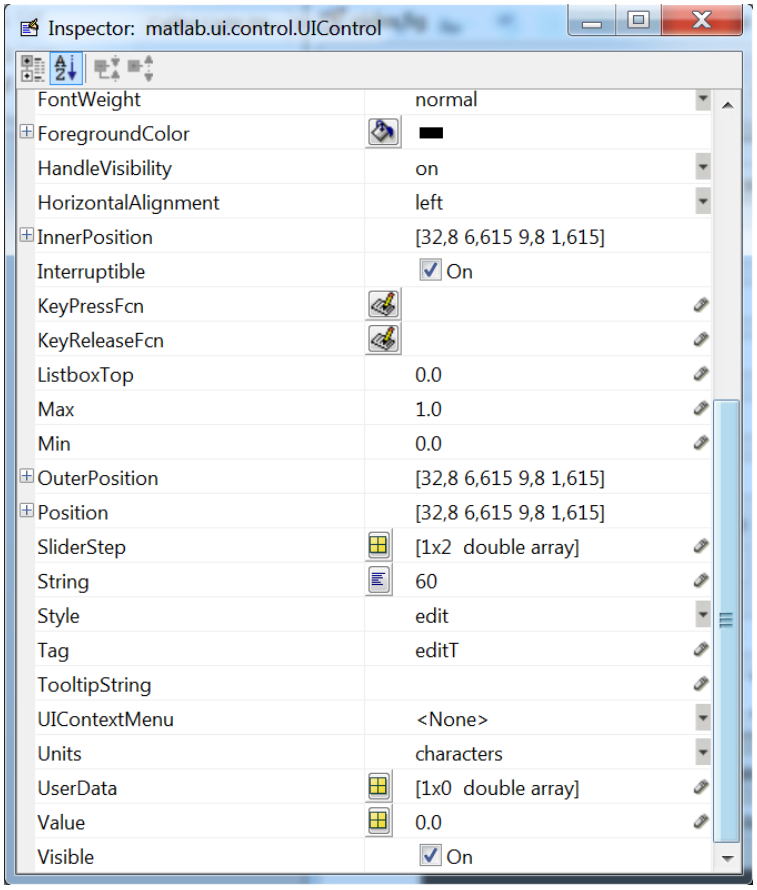
| **Α/Α** | **ΟΝΟΜΑ ΜΕΤΑΒΛΗΤΗΣ ΣΤΟ GUI** | **Α/Α** | **ΟΝΟΜΑ ΜΕΤΑΒΛΗΤΗΣ ΣΤΟ GUI** |
|---|---|---|---|
| **1** | Sample Rate (T) | **9** | I |
| **2** | Filter coef (b) | **10** | D |
| **3** | Final Value | **11** | N |
| **4** | Td1 time | **12** | Kb |
| **5** | Td1 value | **13** | K |
| **6** | Td2 time | **14** | Lower saturation limit (-inf) |
| **7** | Td2 value | **15** | Upper saturation limit (inf) |
| **8** | P | **16** | Simulation time |




σχ.14 Αρχικές τιμές

Έχουμε ορίσει αρχικές τιμές (σχ.14) ώστε να μην είναι κενά τα πεδία και δημιουργούνται προβλήματα στο Simulink από τη στιγμή που έχουμε βάλει μεταβλητές στα block του Simulink.

Στο inspector αυτό που αλλάζουμε είναι τα πεδία του string που είναι η τιμή που δίνουμε με την αρχικοποίηση και μετά με την αλλαγή τους από τον χρήστη καθώς και το tag που είναι το όνομα της συνάρτησης που χρησιμοποιείται στο editor από τον κώδικα.



- **AXES**

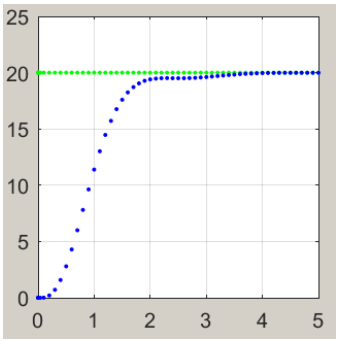
Τα axes είναι τα πεδία όπου δημιουργούνται τα γραφικά. Στην εργασία χρησιμοποιήθηκαν τρία axes:

1. **Εισόδου – Εξόδου**
2. **Σήμα ελέγχου**
3. **Animation**

**Σε όλα τα axes έχουμε realtime δημιουργία των γραφημάτων μας.**

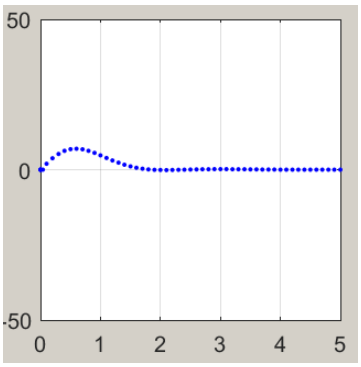
**Στα axes 1 και 2 απεικονίζονται σε δυσδιάστατη μορφή και realtime τα γραφήματά μας :**

1. **Axes1 Είσοδος – Έξοδος**




Με πράσινο χρώμα απεικονίζουμε την είσοδο που δίνουμε στο σύστημα μας και με μπλε είναι η έξοδος του συστήματός μας. Στο συγκεκριμένο διάγραμμα έχουμε ορίσει με κώδικα να μεταβάλλεται το ύψος του κάθετου άξονα ώστε να μπορεί να δείχνει τα σήματα μας για μεγάλα πλάτη.

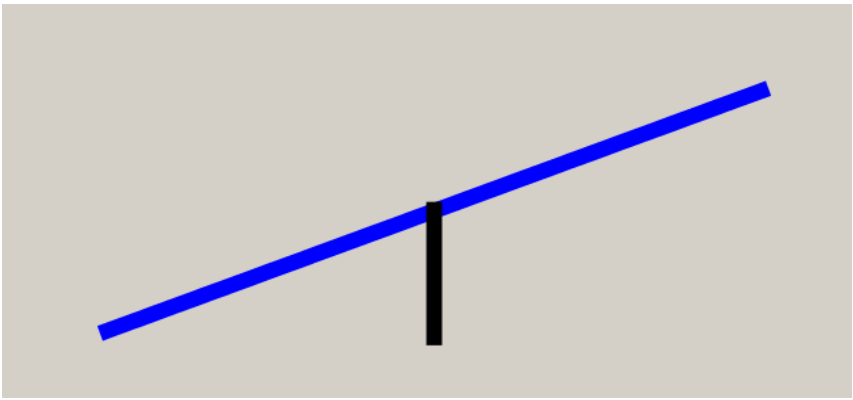
## 2. Axes2 Σήμα ελέγχου




Στο συγκεκριμένο διάγραμμα απεικονίζεται το σήμα ελέγχου του συστήματος.

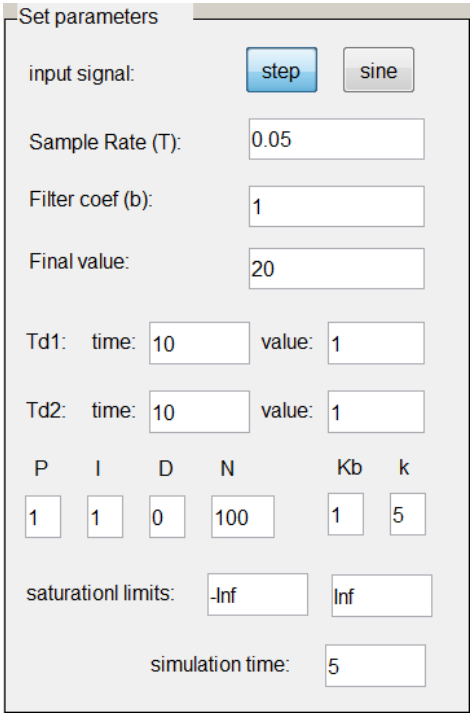
## 3. Axes3 Animation

Στο Axes3 (σχ.15) έχουμε το animation της πειραματικής διαδικασίας. Έχω αποκρύψει τα grid καθώς και τους άξονες για βελτίωση της εικόνας στο πρόγραμμα.



σχ.15 Animationaxes

**Πραγματοποιήσαμε το πείραμά μας με τις παρακάτω τιμές:**




- **<u>ΠΡΟΣΘΕΤΑ ΣΤΟΙΧΕΙΑ GUI</u>**

Έχουμε Statictext για να εισάγουμε ονόματα στα editbox ώστε να γίνεται κατανοητό από τον χρήστη ποιες μεταβλητές του πειράματος αλλάζει και ένα buttongroup που συμπεριλαμβάνει όλες τις μεταβλητές με όνομα Setparameters.

# 5.ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

## 5.1 ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Ο κώδικας γράφτηκε στο περιβάλλον του matlab με χρήση τη γλώσσα scripting που χρησιμοποιείται από το πρόγραμμα.

Μια **γλώσσα σεναρίων** (*scripting language*, *script language*) ή **γλώσσα επέκτασης** (*extension language*) είναι μια γλώσσα προγραμματισμού που επιτρέπει τον έλεγχο μιας ή περισσότερων εφαρμογών. Τα "σενάρια" ("scripts") είναι διακριτά από τον βασικό κώδικα της εφαρμογής, καθώς γράφονται συνήθως σε διαφορετική γλώσσα και συχνά δημιουργούνται ή τροποποιούνται από τον τελικό χρήστη. Τα σενάρια συνήθως διερμηνεύονται από τον πηγαίο κώδικα ή τον κώδικα byte (bytecode), ενώ η εφαρμογή συνήθως έχει ήδη πρώτα μεταγλωττιστεί σε κώδικας μηχανής.

Με την εμφάνιση των γραφικών διασυνδέσεων (graphicaluserinterfaces ή αλλιώς GUI) αναπτύχθηκε ένα εξειδικευμένο είδος γλώσσας σεναρίων για τον έλεγχο του υπολογιστή. Γλώσσες αυτού του είδους αλληλεπιδρούν με τα ίδια γραφικά παράθυρα, μενού, ποντίκια και άλλα γραφικά αντικείμενα και τεχνολογίες που ελέγχει το σύστημα. Αυτό το επιτυγχάνουν προσομοιώνοντας τις ενέργειες ενός ανθρώπου-χρήστη και συνήθως χρησιμοποιούνται για τον αυτοματισμό ενεργειών του χρήστη ή για να ρυθμίσουν μια γνωστή κατάσταση. Όταν ο έλεγχος γίνεται μέσα από προσομοιωμένα πατήματα πλήκτρων ή κλικ του ποντικιού αυτές οι γλώσσες ονομάζονται "μακροεντολές" ("macros").

Αυτές οι γλώσσες θα μπορούσαν θεωρητικά να χρησιμοποιηθούν για να ελέγξουν οποιαδήποτε εφαρμογή τρέχει σε έναν υπολογιστή που βασίζεται σε γραφική διασύνδεση χρήστη, στην πραγματικότητα όμως η υποστήριξη για τέτοιες γλώσσες συνήθως εξαρτάται από την εφαρμογή και το λειτουργικό σύστημα. Υπάρχουν κάποιες εξαιρέσεις σε αυτόν τον περιορισμό. Υπάρχουν γλώσσες αυτού του είδους που βασίζονται στην αναγνώριση γραφικών αντικειμένων από τα πίξελ που φαίνονται στην οθόνη και δεν εξαρτώνται από την υποστήριξη του λειτουργικού συστήματος ή της εφαρμογής.

## 5.2 ΚΩΔΙΚΑΣ ΕΡΓΑΣΙΑΣ

Το πρόγραμμα χωρίστηκε σε δύο κομμάτια, στο κυρίως πρόγραμμα (pidex) και μια συνάρτηση (rod) χρησιμοποιούμε το workspace και το editor για την εγγραφή του κώδικα και την αποθήκευση των μεταβλητών.

Στο κυρίως πρόγραμμα έχουμε τις συναρτήσεις, ονομάζονται callbacksfunctions, που τρέχουν όταν μεταβάλλονται οι τιμές στο GUI καθώς και το κώδικα του animation ο οποίος ζωγραφίζει κάθε φορά καλώντας την συνάρτηση rod η οποία δημιουργεί για κάθε έξοδο που παίρνει από τις τιμές που στέλνουμε από το Simulink το animation το πειράματος.

### 5.2.1 ΚΥΡΙΩΣ ΚΩΔΙΚΑΣ

Στο κυρίως κώδικα αναφέραμε ότι έχουμε την δημιουργία του GUI , τις συναρτήσεις ή callbacksfunctions οι οποίες μεταβάλλουν τις τιμές από τα editbox του GUI και δίνουν τις τιμές στο Simulink για να μεταβάλει τις αντίστοιχες μεταβλητές στα block που χρησιμοποιούμε. Επίσης έχουμε την δημιουργία των axes για να σχεδιάσουμε σε realtime καθώς και το animation καθώς και την αρχικοποίηση των μεταβλητών. Τέλος υπάρχουν στοιχεία κώδικα που δημιουργούνται αυτόματα από το matlab με την έναρξη μιας εργασίας GUI καθώς και στοιχεία κώδικα που βοηθούν στην ομαλή λειτουργία του προγράμματος που δημιουργήθηκε.

```
1.function varargout = pidex(varargin)
2.% PIDEX Application M-file for pidex.fig
3.%PIDEX, by itself, creates a new PIDEX or raises the
existing
4.%singleton*.
5.%
6.%H = PIDEX returns the handle to a new PIDEX or the
handle to
7.%the existing singleton*.
8.%
      9.%PIDEX('CALLBACK',hObject,eventData,handles,...) calls
thelocal

10.%function named CALLBACK in PIDEX.M with the given
input arguments.
11.%
12.%PIDEX('Property','Value',...) creates a new PIDEX or
raises the
13.%existing singleton*.  Starting from the left,
property value pairs are
14.%applied to the GUI before pidex_OpeningFunction gets
called.  An
```

```
15.%unrecognized property name or invalid value makes
property application
16.%stop.  All inputs are passed to pidex_OpeningFcn via
varargin.
17.%
18.%*See GUI Options - GUI allows only one instance to
run (singleton).
19.%
20.%PIDEX('Callback') and PIDEX('Callback', hObject, ...)
call the
21.%named function in PIDEX.M with the given input
      arguments.
22.%
23.% See also: GUIDE, GUIDATA, GUIHANDLES
24.% Copyright 2000-2002 The MathWorks, Inc.
25.% Edit the above text to modify the response to help
pidex
26.% Last Modified by GUIDE v2.5 25-Aug-2017 14:02:53
27.% Begin initialization code - DO NOT EDITN
28.gui_Singleton = 1;
29.gui_State = struct('gui_Name',          mfilename, ...
30.'gui_Singleton',     gui_Singleton, ...
31.'gui_OpeningFcn',    @pidex_OpeningFcn, ...
32.'gui_OutputFcn',     @pidex_OutputFcn, ...
33.'gui_LayoutFcn',     [], ...
34.'gui_Callback',      []);
35.if nargin & isstr(varargin{1})
36.gui_State.gui_Callback = str2func(varargin{1});
37.end
38.if nargout
39.varargout{1:nargout} = gui_mainfcn(gui_State,
   varargin{:});
40.else
41.gui_mainfcn(gui_State, varargin{:});
42.end
43.% End initialization code - DO NOT EDITN
```

**>>Γραμμές 1 – 43** **Οι συγκεκριμένες γραμμές κώδικα δημιουργήθηκαν αυτόματα από το matlab με τη δημιουργεία του GUI.**

```
44.% --- Executes just before pidex is made visible.
45.function pidex_OpeningFcn(hObject, eventdata, handles)
```

**>>Γραμμή 45** **Τρέχει όταν τρέχουμε το αρχείο του GUI.**

```
46.% This function has no output args, see OutputFcn.
47.% hObject    handle to figure
48.% eventdata  reserved - to be defined in a future version
   of MATLAB
49.% handles    structure with handles and user data (see
   GUIDATA)
50.% varargin   command line arguments to pidex (see
   VARARGIN)
51.axes(handles.axes1);plot(0),axis([0 60 0 25]),grid on;
52.axes(handles.axes2);rod(0)%,cla;
53.axes(handles.axes3);plot(0),axis([0 60 -50 50]),grid on;
```

**>>Γραμμές 51 – 53** Αρχικοποίηση των axes ώστε να εμφανίζονται καλύτερα στο πρόγραμμα και πιο συγκεκριμένα στην 63 γραμμή του κώδικα όπου καλείται η συνάρτηση του animation.

```
54.assignin('base','T',str2num(get(handles.tValue,'String')))
55.assignin('base','b',str2num(get(handles.bValue,'String')))
56.assignin('base','d',str2num(get(handles.dValue,'String')))
57.assignin('base','s1',str2num(get(handles.s1Value,'String')))
58.assignin('base','v1',str2num(get(handles.v1Value,'String')))
59.assignin('base','s2',str2num(get(handles.s2Value,'String')))
60.assignin('base','v2',str2num(get(handles.v2Value,'String')))
61.assignin('base','P',str2num(get(handles.editP,'String')))
62.assignin('base','I',str2num(get(handles.editI,'String')))
63.assignin('base','D',str2num(get(handles.editD,'String')))
64.assignin('base','N',str2num(get(handles.editN,'String')))
65.assignin('base','L1',str2num(get(handles.editL1,'String')))
66.assignin('base','L2',str2num(get(handles.editL2,'String')))
67.assignin('base','k',str2num(get(handles.editk,'String')))
68.assignin('base','Kb',str2num(get(handles.editKb,'String')))
69.assignin('base','a',0)
```

**>>Γραμμές 54 – 69** Αποθήκευση των τιμών που βάζουμε στο GUI στο workspace για να μπορέσει το Simulink να τις χρησιμοποιήσει αποθηκεύοντάς τες σαν μεταβλητές. Συγκεκριμένα το base είναι το workspace, ακολουθεί το όνομα της μεταβλητής, η εντολή str2num παίρνει το κείμενο από το inspector και πιο συγκεκριμένα από το string του inspector και το μετατρέπει σε αριθμό (από που θα το παραλάβει το κάνουμε με το get(handles.---), το οποίο handles περιέχει όλα τα στοιχεία του GUI και μετά την τελεία αναφέρουμε από ποιο component του GUI συγκεκριμένα) και τέλος με το string δείχνουμε σε ποιο μέρος του inspector να παραλάβει την τιμή.

```
70.model_open(handles)
```

**>>Γραμμή 70** Είναι η κλήση της συνάρτησης που βρίσκεται στη γραμμή 77.

```
71.handles.cumD=0;
72.handles.i=1;
```

**>>Γραμμές 71 – 72** Αρχικοποίηση μεταβλητών που θα χρησιμοποιηθούν για το simulation

```
% Choose default command line output for pidex
73.handles.output = hObject;
% Update handles structure
74.guidata(hObject, handles);
```

**>>Γραμμές 73 – 74 Δημιουργούνται αυτόματα από το GUI.**

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

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

**>>Γραμμές 75 κ' 76 Δημιουργούνται αυτόματα από το GUI. Οι συγκεκριμένες γραμμές μπορούν και να παραληφθούν.**

```
77.function model_open(handles)
78.if  isempty(find_system('Name','pidm')),
79.   open_system('pidm');
80.   figure(handles.pidFig)
81.end
```

**>>Γραμμές 77 – 81 Είναι η συνάρτηση που καλείται στην γραμμή 70 και ελέγχει αν είναι ανοιχτό το μοντέλο μας και αν δεν είναι το ανοίγει. Στην γραμμή 80 η εντολή φέρνει το GUI πάνω από όλα τα παράθυρα.**

```
82.function varargout = tValue_Callback(h, eventdata, handles)
```

**>>Γραμμή 82 Όνομα της συνάρτησης όπου παίρνει την τιμή που αλλάζουμε στις μεταβλητές του GUI, το συγκεκριμένο παίρνει την τιμή από το SampleRate(T), Το 'h' το έγραψα για ευκολία και είναι η είσοδος της συνάρτησης, όταν αλλάζω την τιμή στο SampleRate(T) στο GUI καλεί την tValue_Callback και δείνει στο h όλο το inspector από το editbox του SampleRate(T).**

```
83.% hObject    handle to tValue (see GCBO)
84.% eventdata  reserved - to be defined in a future
   version ofMATLAB

85.% handles    structure with handles and user data (see
   GUIDATA)
86.NewVal = str2num(get(h,'String'));
```

**>>Γραμμή 86 Από το inspector που βρίσκεται στο h πηγαίνει και παίρνει από το string το κείμενο, το μετατρέπει σε αριθμό και τον αποθηκεύει στο NewVal.**

```
87.if  isempty(NewVal)
88.set(h,'String',0.05)
```

**>>Γραμμές 87 – 88** **Έλεγχος της μεταβλητής NewVal αν είναι κενή και αν είναι βάζουμε μια default τιμή στην συγκεκριμένη περίπτωση η τιμή είναι 0.05, ώστε να μην είναι ποτέ κενή.**

```
89.else
90.assignin('base','T',NewVal)
```

**>>Γραμμή 90** **Βάζουμε στην μεταβλητή που βρίσκεται στο workspace ώστε να μπορεί να χρησιμοποιηθούν από το Simulink ότι τιμή εμπεριέχεται στην μεταβλητή NewVal.**

```
91.end
```

**>>Γραμμές 82 – 91** **Η συγκεκριμένη λειτουργία της συνάρτησης είναι όμοια με τις υπόλοιπες συναρτήσεις που έχουμε στο GUI και μεταβάλουμε τις τιμές στα editbox.**

```
92.functionvarargout = bValue_Callback(h, eventdata, handles)
93.% hObject    handle to bValue (see GCBO)
94.% eventdata  reserved - to be defined in a future version of
   MATLAB
95.% handles    structure with handles and user data (see
   GUIDATA)
96.NewVal = str2num(get(h,'String'));

97.if  isempty(NewVal)
98.set(h,'String',0.75)
99.else,
100.assignin('base','b',NewVal)
101.end
```

**>> Γραμμές 92 – 101** **Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για το Filtercoef(b)**

```
function dValue_Callback(h, eventdata, handles)
% hObject    handle to dValue (see GCBO)
% eventdata  reserved - to be defined in a future version ofMATLAB
% handles    structure with handles and user data (see GUIDATA)
NewVal = str2num(get(h,'String'));

if  isempty(NewVal)
set(h,'String',20)
else
if get(handles.stepToggle,'Value')==1
assignin('base','d',NewVal)
else
assignin('base','a',NewVal)
end
```

**>> Γραμμές 109 – 114** **Έλεγχος για το ποια είσοδο είναι επιλεγμένη**

```
end
```

**>> Γραμμές 102 – 115** **Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 με την διαφορά ότι αλλάζοντας την τιμή για την είσοδο που βάζουμε και μπορεί να είναι επιλεγμένο το step ή το sine, κάνουμε έλεγχο το τι είσοδο έχουμε επιλεγμένη ώστε να αποθηκεύσει στην αντίστοιχη μεταβλητή στο workspace. Γίνεται αλλαγή στο FinalValue για το step και στο Amplitude για το sine.**

```
function s1Value_Callback(h, eventdata, handles)
% hObject    handle to s1Value (see GCBO)
% eventdata  reserved - to be defined in a future version ofMATLAB
% handles    structure with handles and user data (see GUIDATA)
NewVal = str2num(get(h,'String'));

if  isempty(NewVal)
set(h,'String',10)
else
assignin('base','s1',NewVal)
end
```

**>> Γραμμές 116 – 125** **Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για το Td1: το time:, το χρόνο που βάζουμε να πραγματοποιηθεί η πρώτη διαταραχή μας.**

```
126.function v1Value_Callback(h, eventdata, handles)
127.% hObject    handle to v1Value (see GCBO)
128.% eventdata  reserved - to be defined in a future version
    of MATLAB
129.% handles    structure with handles and user data (see
    GUIDATA)

130.NewVal = str2num(get(h,'String'));

131.if  isempty(NewVal)
132.set(h,'String',1)
133.else
134.assignin('base','v1',NewVal)
135.end
```

**>> Γραμμές 126 – 135 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για το Td1: το value:, το μέγεθος που θα έχει η πρώτη διαταραχή μας.**

```
136.function s2Value_Callback(h, eventdata, handles)
137.% hObject    handle to s2Value (see GCBO)
138.% eventdata  reserved - to be defined in a future version
    of MATLAB
139.% handles    structure with handles and user data (see
    GUIDATA)
140.NewVal = str2num(get(h,'String'));

141.if  isempty(NewVal)
142.set(h,'String',10)
143.else
144.assignin('base','s2',NewVal)
145.end
```

**>> Γραμμές 136 – 145 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για το Td2: το time:, το χρόνο που βάζουμε να πραγματοποιηθεί η δεύτερη διαταραχή μας.**

```
146.function v2Value_Callback(h, eventdata, handles)
147.% hObject    handle to v2Value (see GCBO)
148.% eventdata  reserved - to be defined in a future version
    of MATLAB

149.% handles    structure with handles and user data (see
    GUIDATA)
150.NewVal = str2num(get(h,'String'));

151.if  isempty(NewVal)
152.set(h,'String',1)
153.else
154.assignin('base','v2',NewVal)
155.end
```

**>> Γραμμές 146 – 155 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για το Td2: το value:, το μέγεθος που θα έχει η δεύτερη διαταραχή μας.**

```
156.function varargout = SimulateButton_Callback(h, eventdata,
handles)
157.% hObject    handle to SimulateButton (see GCBO)
158.% eventdata  reserved - to be defined in a future version
    ofMATLAB
159.% handles    structure with handles and user data (see
    GUIDATA)

160.if handles.i==1
161.axes(handles.axes1); cla; ylim([0 25]);
162.axes(handles.axes3); cla;
163.end
```

**>> Γραμμές 160 – 163 Στη γραμμή 160 δημιουργούμε μεταβλητή που θα χρησιμοποιηθεί στην εργασία του animation και δείχνει αν έχει τελειώσει το animation, Στις γραμμές 161 και 162 καθαρίζει τα axes εισόδου – εξόδου καθώς και του animation όταν έχει τελειώσει το simulation και πατηθεί ξανά το simulatebutton.**

```
164.init=handles.cumD;
165.handles.cumD=handles.cumD+evalin('base','d');
166.guidata(h, handles);
```

**>> Γραμμές 164 – 166 Στη γραμμή 164 έχουμε τη αύξηση του CumD αλλά δεν το χρησιμοποιεί, το αποθηκεύει για όταν ξαναπατηθεί το κουμπί simulate για να συνεχίσει να ζωγραφίζει από τις μοίρες που ήταν το τελευταίο simulation πριν πατηθεί ξανά το κουμπί simulate στο ίδιο simulation που διεξάγεται. Στις γραμμές 165 και 166 καλούμε το guidata κάθε φορά που χρησιμοποιούμε το handles ώστε να αποθηκεύονται οι τιμές.**

```
167.[timeVector,stateVector,outputVector,controlVector,inVector
    ] = sim('pidm');
```

**>> Γραμμή 167 Παίρνει τις τιμές από το Simulink και τα κάνει plot στους axes, η εντολή sim('pidm') ξεκινάει και τρέχει το Simulink.**

```
168.t=timeVector;
169.a=outputVector;
170.s=2;
171.j=1;
172.dur=str2num(get_param('pidm', 'StopTime'));
173.amp=evalin('base','a');
```

**>> Γραμμές 168 – 173** **Αρχικοποίηση μεταβλητών που θα χρησιμοποιηθούν παρακάτω στον κώδικα καθώς και δημιουργία των πινάκων που παραλαμβάνουμε από το Simulink. Στη γραμμή 170 έχουμε την ταχύτητα του animation και στην γραμμή 171 είναι δείκτης που αυξάνεται στην for που ακολουθεί. Στη γραμμή 172 σώζουμε στην μεταβλητή dur τον χρόνο πειράματος που ορίζουμε στο GUI στο πλαίσιο simulationtime. Στην γραμμή 173 παίρνει τιμή από το workspace με το evalin χωρίς να την μεταβάλει και πιο συγκεκριμένα για την είσοδο του sine.**

```
174.set(handles.axes1,'xlim',[0 dur]);
175.yl=get(handles.axes1,'ylim');
176.if handles.cumD+amp>yl(2)
177.set(handles.axes1,'ylim',[yl(1) handles.cumD+amp+5]);
178.end
179.yl=get(handles.axes1,'ylim');
180.if handles.cumD-amp<0 && handles.cumD-amp<yl(1)
    181.set(handles.axes1,'ylim',[handles.cumD-amp-5 yl(2)]);
182.end
183.if handles.i==1 && amp~=0
184.set(handles.axes1,'ylim',[-amp*3 amp*3]);
185.end
```

**>> Γραμμές 174 – 185** **Φτιάχνει τα όρια στο axes1 ώστε να προσαρμόζεται στα δεδομένα που βάζουμε στην εκτέλεση κάθε πειράματος που διεξάγουμε.**

```
186.setappdata(h, 'CallbackRun', 0);
```

**>> Γραμμές 186,256,202** **Οι συγκεκριμένες γραμμές κώδικα δεν επιτρέπουν να συνεχιστεί να εκτελούνται οι προηγούμενες τιμές και για την εξάλειψη ενός προβλήματος που αντιμετωπίζονταν με τις συγκεκριμένες γραμμές κώδικα .**

```
187.for i=handles.i:s:length(a)
```

**>> Γραμμή 187** **Στην for αυξάνονται δύο μεταβλητές, μία είναι η i που είναι ο χρόνος ενώ το j κάθε φορά που πατάμε το simulate μηδενίζεται για να αρχίσει η δημιουργία των Vector ή πινάκων που χρησιμοποιούμε από το Simulink από την αρχή. Σώζει σε ποιο i βρίσκεται στην global μεταβλητή για να μην μεταβάλλεται.**

```
188.if ~ishandle(h), return; end
```

**>> Γραμμή 188** **Γίνεται έλεγχος του h ή του GUI για να μην έχουμε crush του προγράμματος γιατί άμα το GUI κλείσει h συνάρτηση συνεχίζει να τρέχει, η εντολή return σταματάει την συνάρτηση.**

```
189.if (getappdata(h, 'CallbackRun') == 1), return; end
```
(Γραμμή 186)

```
190.set(handles.pidFig, 'CurrentAxes', handles.axes3);hold
    on;
```
**>> Γραμμή 190 Επιλέγουμε το axes που θα σχεδιαστεί το ControlVector με ένα διαφορετικό τρόπο λόγω προβλήματος με τον κλασσικό τρόπο σχεδιασμού (άλλαζε το focus στο simulatebutton και δεν επιτρεπόταν να πατηθεί το κουμπί)**

```
191.plot(t(i),controlVector(j),'.b');xlim([0 dur]);
```
**>> Γραμμή 191 Σχεδιασμός του ControlVector, κάθε for σχεδιάζει μια κουκίδα με άξονα χρόνου το T.**

```
192.set(handles.pidFig, 'CurrentAxes', handles.axes1);hold
    on;
193.plot(t(i),inVector(j)+init,'.g'),hold on;
```
**>> Γραμμές 192,193 Είναι η ίδια διαδικασία με τις γραμμές 190 και 191 αλλά εδώ είναι για τον axes τις εισόδου.**

```
194.plot(t(i),a(j)+init,'.b'),hold on;
```
**>> Γραμμή 194 Επιλογή axes για την έξοδο και σχεδιασμός εξόδου**

```
195.set(handles.pidFig, 'CurrentAxes', handles.axes2);
```
**>> Γραμμή 195 Επιλογή axes για το animation**

```
196.cla;
```
**>> Γραμμή 196 Καθαρισμός όλων των axes.**

```
197.  rod(a(j)+init);
```
**>> Γραμμή 197 Καλούμε την rod (συνάρτηση animation).**

```
198.if ishandle(h), handles.i=i; guidata(h, handles); end
```
**>> Γραμμή 198 Έλεγχος για το αν εκτελείται το GUI και αποθηκεύει την καινούργια τιμή τοτ I που αυξήθηκε στην for.**

```
            0
199.j=j+s;
```
**>> Γραμμή 199 Αύξηση του δείκτη j για το Vector.**

```
200.drawnow;
```
**>> Γραμμή 200 Σχεδιασμός του animation καθώς και πρόσθεση μιας καθυστέρησης**

```
201.end
202.setappdata(h, 'CallbackRun', 1);
```
(Γραμμή 186)

```
203.%cla;
204.%rod(0)
```
**>> Γραμμές 203,204Κ αθαρίζει τους άξονες και μηδενίζει το animation γυρνώντας την μπάρα στην αρχική της θέσης. Τα έχω απενεργοποιήσει για να μένει η τελική θέση στην μπάρα.**

```
205.if ishandle(h)
206.handles.cumD=0;
207.handles.i=1;
208.guidata(h, handles);
209.end
```
**>> Γραμμές 205,209 Μηδενίζει και αρχικοποιεί τις μεταβλητές όταν τελειώσει το simulation.**

```
210.% --- Executes on button press in stepToggle.
211.function stepToggle_Callback(hObject, eventdata, handles)
212.% hObject    handle to stepToggle (see GCBO)
213.% eventdata  reserved - to be defined in a future version
    ofMATLAB
214.% handles    structure with handles and user data (see
    GUIDATA)

215.uicontrol(handles.frame1);
```
**>> Γραμμή 215 Όταν πατάς το button της εισόδου εμφανίζεται ένα πλέγμα με αποτέλεσμα να μην μπορεί να ξαναπατηθεί αλλάζοντας το focus και με αυτήν την εντολή εξαφανίζουμε το συγκεκριμένο πρόβλημα.**

```
216.set(hObject,'Enable','inactive');
```
**>> Γραμμή 216 Όταν επιλέγεις το button της εισόδου το απενεργοποιεί ώστε να μην μπορεί να επιλεγεί ξανά από τον χρήστη.**

```
217.set(handles.sineToggle,'Enable','on','Value',0);
```
**>> Γραμμή 217 Όταν επιλέγεται η είσοδος step πατώντας το button και είναι ήδη πατημένο το button της εισόδου του sine να ακυρώνεται η επιλογή του sine κάνοντας το vale 0.**

```
218.set(handles.dText,'String','Final value:');
```
**>> Γραμμή 218 Αλλάζει το txt στο editbox της εισόδου σε Value για την step είσοδο.**

```
219.set(handles.dValue,'String',20);
```
**>> Γραμμή 219 Αλλάζει την τιμή τις εισόδου του step.**

```
220.assignin('base','d',20)
221.assignin('base','a',0)
```

**>> Γραμμές 220,221 Αρχικοποιεί τις τιμές των εισόδων step και sine.**

```
222.% --- Executes on button press in sineToggle.
223.function sineToggle_Callback(hObject, eventdata, handles)
224.% hObject    handle to sineToggle (see GCBO)
225.% eventdata  reserved - to be defined in a future version
    of MATLAB
226.% handles    structure with handles and user data (see
    GUIDATA)

227.uicontrol(handles.frame1);
228.set(hObject,'Enable','inactive');
229set(handles.stepToggle,'Enable','on','Value',0);
230.set(handles.dText,'String','Amplitude:');
231.set(handles.dValue,'String',2);
232.assignin('base','d',0)
233.assignin('base','a',2)
```

**>> Γραμμές 227 – 233Είναι η ίδια διαδικασία με την είσοδο step, μαζί με τους ελέγχους και τις αρχικοποιήσεις.**

```
234.function editP_Callback(h, eventdata, handles)
235.% hObject    handle to editP (see GCBO)
236.% eventdata  reserved - to be defined in a future version
    of MATLAB
237.% handles    structure with handles and user data (see
    GUIDATA)

238.NewVal = str2num(get(h,'String'));

239.if  isempty(NewVal)
240.set(h,'String',1)
241.else,
242.assignin('base','P',NewVal)
243.end
```

**>> Γραμμές 234 – 243 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή P του ελεγκτή μας.**

```
244.function editI_Callback(h, eventdata, handles)
245.% hObject    handle to editI (see GCBO)
246.% eventdata  reserved - to be defined in a future version
    of MATLAB
247.% handles    structure with handles and user data (see
    GUIDATA)

248.NewVal = str2num(get(h,'String'));

249.if  isempty(NewVal)
250.set(h,'String',1)
251.else,
252.assignin('base','I',NewVal)
253.end
```

**>> Γραμμές 244 – 253 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή I του ελεγκτή μας.**

```
254.function editD_Callback(h, eventdata, handles)
255.% hObject    handle to editD (see GCBO)
256.% eventdata  reserved - to be defined in a future version
    of MATLAB
257.% handles    structure with handles and user data (see
    GUIDATA)

258.NewVal = str2num(get(h,'String'));

259.if  isempty(NewVal)
260.set(h,'String',0)
261.else,
262.assignin('base','D',NewVal)
263.end
```

**>> Γραμμές 254 – 263 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή D του ελεγκτή μας.**

```
264.function editN_Callback(h, eventdata, handles)
265.% hObject    handle to editN (see GCBO)
266.% eventdata  reserved - to be defined in a future version
    of MATLAB
267.% handles    structure with handles and user data (see
    GUIDATA)

268.NewVal = str2num(get(h,'String'));

269.if  isempty(NewVal)
270.set(h,'String',100)
271.else,
272.assignin('base','N',NewVal)
273.end
```

**>> Γραμμές 264 – 273 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή N του ελεγκτή μας.**

```
274.function editL1_Callback(h, eventdata, handles)
275.% hObject    handle to editL1 (see GCBO)
276.% eventdata  reserved - to be defined in a future version
    of MATLAB
277.% handles    structure with handles and user data (see
    GUIDATA)

278.NewVal = str2num(get(h,'String'));

279.if  isempty(NewVal)
280.set(h,'String',-Inf)
281.else,
282.assignin('base','L1',NewVal)
283.end
```

**>> Γραμμές 274 – 283 Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή saturationlimits: το upperlimit στον PID.**

```
284.function editL2_Callback(h, eventdata, handles)
285.% hObject    handle to editL2 (see GCBO)
286.% eventdata  reserved - to be defined in a future version
    of MATLAB
287.% handles    structure with handles and user data (see
    GUIDATA)

288.NewVal = str2num(get(h,'String'));

289.if  isempty(NewVal)
290.set(h,'String',Inf)
291.else,
292.assignin('base','L2',NewVal)
293.end
```

**>> Γραμμές 284 – 293** **Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή saturationlimits: το lowerlimit στον PID.**

```
294.function editT_Callback(hObject, eventdata, handles)
295.% hObject    handle to editT (see GCBO)
296.% eventdata  reserved - to be defined in a future version
of MATLAB
297.% handles    structure with handles and user data (see
GUIDATA)
298.set_param('pidm','StopTime',get(hObject,'String'));
```

**>> Γραμμές 294 – 298** **Σώζει την τιμή του χρόνου πειράματος T κατευθείαν στο Simulink και αυτό γίνεται με την εντολή se_param.**

```
299.% --- Executes on button press in resetButton.
300.function resetButton_Callback(h, eventdata, handles)
301.% hObject    handle to resetButton (see GCBO)
302.% eventdata  reserved - to be defined in a future version
of MATLAB
303.% handles    structure with handles and user data (see
GUIDATA)

304.setappdata(handles.SimulateButton, 'CallbackRun', 1);
305.guidata(h, handles);
306.axes(handles.axes1),cla;
307.axes(handles.axes2),cla;rod(0);
308.axes(handles.axes3),cla;

309.if ishandle(h)
310.handles.cumD=0;
311.handles.i=1;
312.guidata(h, handles);
313.end
```

**>> Γραμμές 299 – 313** **Είναι η λειτουργία του buttonreset. Στην γραμμή 304 σταματάει το simulation, στις γραμμές 306 με 308 αρχικοποιούμε τα axes και στις γραμμές 309 με 313 αρχικοποιεί τις τιμές για το simulation όπως στις γραμμές 51 – 53.**

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

NewVal = str2num(get(h,'String'));

if  isempty(NewVal)
set(h,'String',5)
else
assignin('base','k',NewVal)
end
```

**<u>>> Γραμμές 314 – 323</u> Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή k στο blockparameters**

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

NewVal = str2num(get(h,'String'));

if  isempty(NewVal)
set(h,'String',1)
else
assignin('base','Kb',NewVal)
end
```

**<u>>> Γραμμές 324 – 333</u> Είναι παρόμοια με τις γραμμές κώδικα 82 – 91 για την μεταβλητή kb στον PID.**

## 5.2.2 ΣΥΝΑΡΤΗΣΗ (ROD)

Όπως είπαμε και παραπάνω η συνάρτηση rod παίρνει τιμές από το outputVector και με την πράξη του rotationmatrix δημιουργεί για κάθε σημείο του πίνακα που παίρνει το σχέδιο της μπάρας.

```
1.functionrod(a)
```

**>>Η τιμή σε μοίρες που παίρνει από το outputVector.**

```
2.t=a*pi/180;
```

**>>Μετατροπή μοιρών σε rad.**

```
3.r=[cos(t) -sin(t); sin(t) cos(t)];
```

**>>Η συνάρτηση ονομάζεται rotationmatrix και με αυτήν ζωγραφίζουμε για τις μοίρες που έχουμε δώσει από το GUI την καινούργια θέση της μπάρας.**

```
4.x=[-4:0.2:4];
```

**>>Δημιουργούμε μια γραμμή.**

```
5.y=zeros(1,length(x));
```

**>>Δημιουργούμε ένα πίνακα γραμμή με μηδενικά με τόσες μεταβλητές όσες περιέχει ο πίνακας x.**

```
6.m=[x;y];
```

**>>Δημιουργώ τον πίνακα x με y για να χρησιμοποιηθεί στο rotationmatrix.**

```
7.n=r*m;
```

**>>Γίνεται η πράξη του rotationmatrix.**

```
8.% plot(n(1,:),n(2,:),'o'),axis([-4.67 4.67 -2 2])%,axis off;
9.% set(gca,'XTickLabel',[],'YTickLabel',[]);
10.%
11.% % show axis
12.% hold on, plot([0 0],get(gca,'ylim'),'k')
13.% hold on, plot(get(gca,'xlim'),[0 0],'k'),grid on;

14.plot(n(1,:),n(2,:),'b','LineWidth',6),axis([-4.67 4.67 -2
   2]),axisoff;
15.holdon;plot([0 0],[-1.5
   0.1],'k','LineWidth',6,'LineSmoothing','on')
16.set(gca,'XTickLabel',[],'YTickLabel',[]);
```

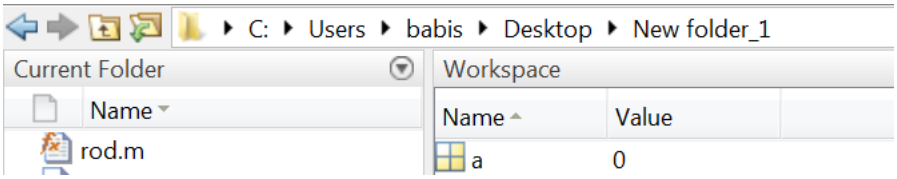
**>>Δημιουργία της μπάρας που θα σχεδιάζεται από το animation.**

# 6. ΠΑΡΑΔΕΙΓΜΑ ΠΕΙΡΑΜΑΤΟΣ

## 6.1 ΕΝΑΡΞΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

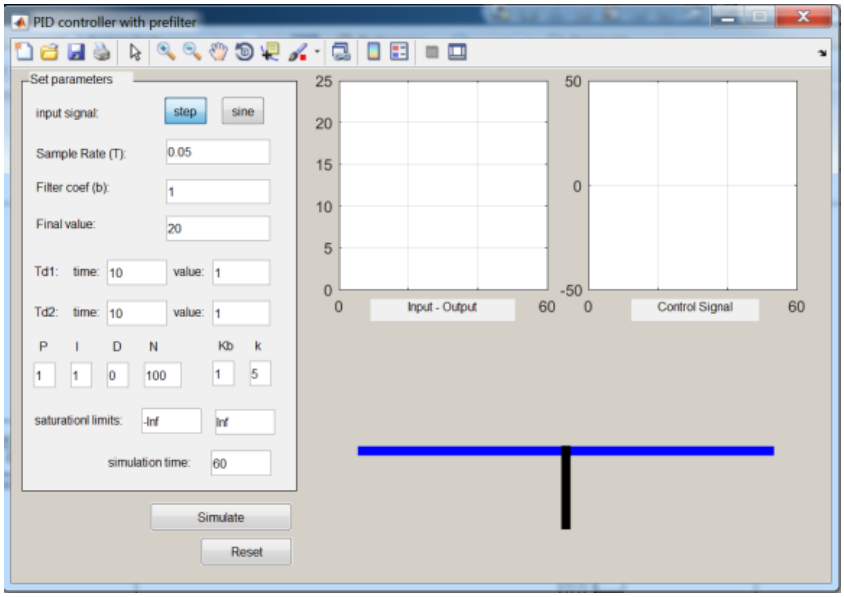
Θα δείξουμε ένα τυπικό παράδειγμα πως από την αρχή ανοίγουμε το πρόγραμμά μας και θα εισάγουμε παραμέτρους στο σύστημά μας για να κατανοήσουμε την όλη δοκιμασία και ώστε να μπορεί κάποιος άλλος χρήστης να χρησιμοποιήσει το πρόγραμμα.

1. Ανοίγουμε το matlab
2. Στο πάνω μέρος του matlab υπάρχει η επιλογή του path ώστε να ανοίξουμε το αρχείο μας και να μπορεί να χρησιμοποιηθεί. Επιλέγουμε το φάκελο που περιέχει τα στοιχεία μας.

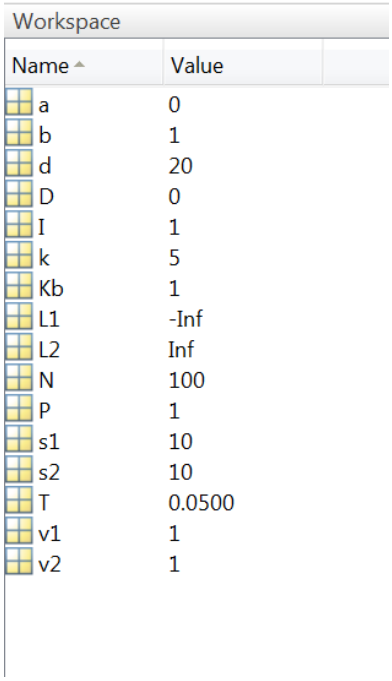



3. Πληκτρολογούμε το όνομα του προγράμματος, στην δικιά μας περίπτωση ‘’pidex’’, στο command windows του matlab.

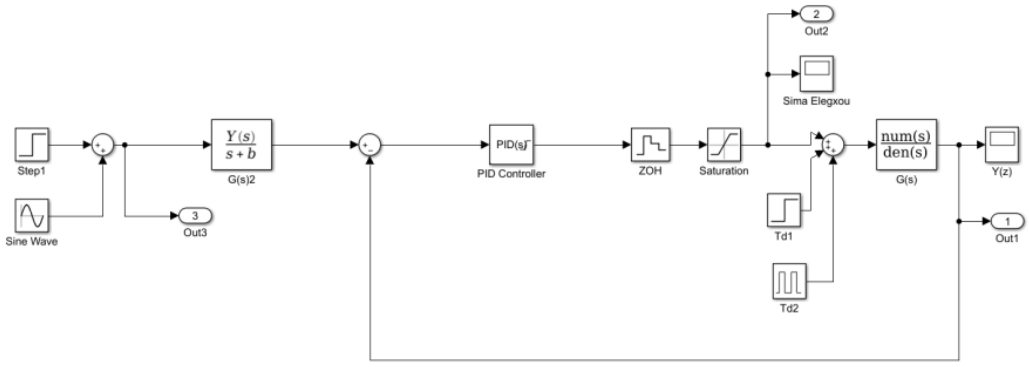
4. Ανοίγει με μια μικρή αλλά λογική καθυστέρηση το GUI μαζί με το Simulink και απευθείας τρέχει τον κώδικα για τις αρχικοποιήσεις μαζί με την εισαγωγή των μεταβλητών μας στο workspace και μετά μπαίνει σε αναμονή για την εισαγωγή των παραμέτρων που επιθυμούμε. Συγκεκριμένα ανοίγουν τα παρακάτω παράθυρα(σχ.16, σχ.17, σχ.18) και μπροστά στον χρήστη έρχεται το παράθυρο του προγράμματός μας, το GUI.




σχ.16 πρόγραμμα GUI για διαχείρηση των παραμέτρων




σχ.17 Μεταβλητές που αποθηκεύονται στο Workspace



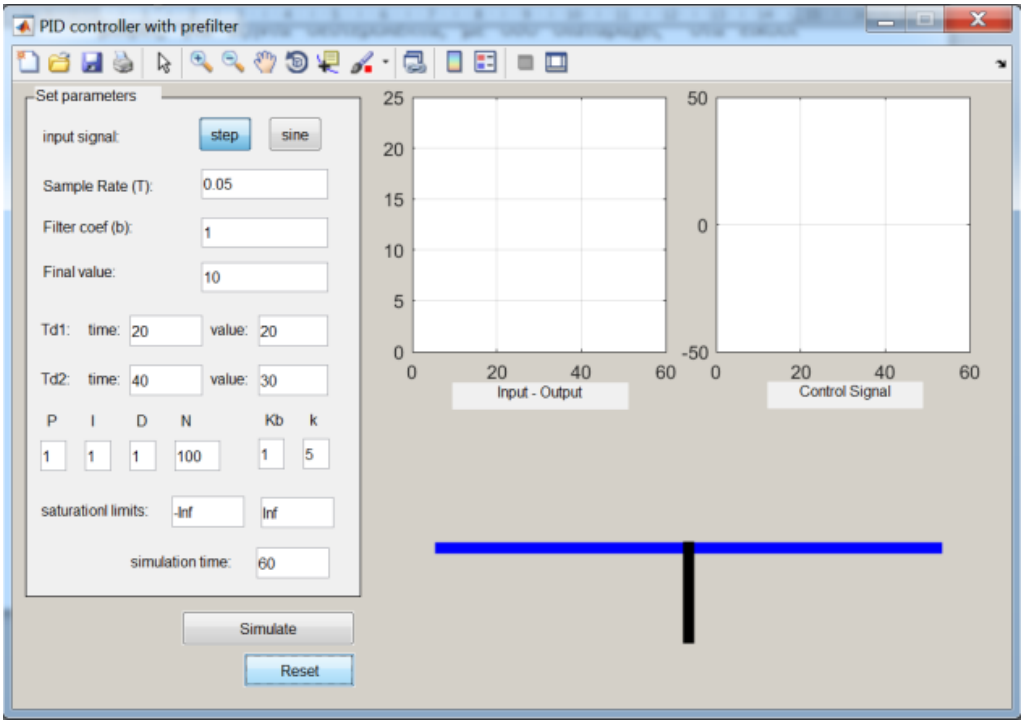

Σχ.18 Σύστημα Simulink της εργασίας

## 6.2 ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ

Από εδώ και πέρα μπορούμε όπως είπαμε και παραπάνω να εισάγουμε τις παραμέτρους μας και να διεξάγουμε τα πειράματά μας.

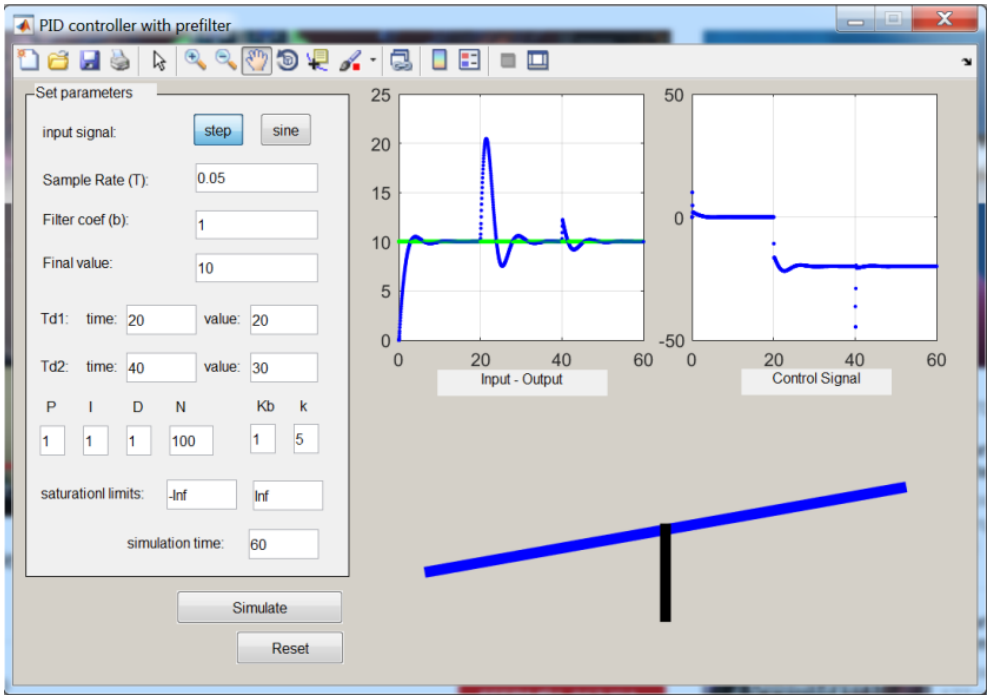
Στο συγκεκριμένο θα έχουμε βηματική είσοδο με πλάτος δέκα μοίρες, για εξήντα δευτερόλεπτα, με δυο διαταραχές στα είκοσι δευτερόλεπτα και στα σαράντα με πλάτος είκοσι μοιρών για την πρώτη και τριάντα μοιρών για την δεύτερη και με PID ελεγκτή.

Εισαγωγή παραμέτρων :



Πατώντας το button simulate το σύστημα μας αρχίζει να τρέχει.

Με το πέρας του χρόνου η τελική μορφή είναι:



## 7. ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ Η ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥΣ

Τα κυριότερα προβλήματα που αντιμετωπίστηκαν στην συγγραφή της πτυχιακής εργασίας παρουσιάστηκαν στην τελειοποίηση του κώδικα όσον αφορά το κομμάτι του realtime καθώς και της μνήμης ώστε να διατηρεί μετά την ξαναχρησιμοποίηση του buttonsimulate κατά τη διάρκεια μιας πειραματικής διαδικασίας να μπορεί συνεχίζει από τις τιμές που είχε και να αποφεύγεται ο μηδενισμός των παραμέτρων, που είχε ως αποτέλεσμα η διαδικασία να διακόπτεται.

Στον κώδικα επίσης λόγω πολλών εντολών δημιουργούνταν προβλήματα στην εκκίνηση του προγράμματος, το πρόβλημα αυτό επιλύθηκε με την προσθήκη τριών γραμμών εντολών στις γραμμές 186, 256, 202, καθώς και με τοποθέτησή τους σε διάφορα μέρη του προγράμματος μέχρι να βρεθεί ή κατάλληλη θέση που λειτουργούσαν και έλυναν το πρόβλημά μας.

Στην αρχή ο κώδικας γράφτηκε στην έκδοση 2007α του matlab και το όνομα του αρχείου ήταν PID2, το συγκεκριμένο όνομα καθώς και πολλά άλλα σε μεταγενέστερες εκδόσεις είχαν προστεθεί σαν εντολές προγραμμάτων με αποτέλεσμα να εμφανίζει σφάλμα κατά την εκκίνηση του προγράμματος. Επιλύθηκε με την αλλαγή του ονόματος του προγράμματος και ορισμένων ονομάτων μεταβλητών που δημιουργούσαν το ίδιο σφάλμα.

## 8. ΒΙΒΛΙΟΓΡΑΦΙΑ

## ΙΣΤΟΣΕΛΙΔΕΣ:


*1. Μοντελοποίηση –Προσομοίωση & Υπολογιστικές Επιστήμες: Μια πρώτη προσέγγιση*
*ΣαράντοςΨυχάρης*
*Παιδαγωγικό Τμήμα Δημοτικής Εκπαίδευσης Πανεπιστημίου Αιγαίου*

*2. https://www.tutorialspoint.com/matlab/matlab_m_files.htm*

*3. https://www.mathworks.com/?s_tid=gn_logo*


## ΠΤΥΧΙΑΚΕΣ ΕΡΓΑΣΙΕΣ :


*4. ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ*

*ΘΕΜΑ: ΨΗΦΙΑΚΟΣ ΕΛΕΓΧΟΣ ΚΛΙΣΗΣ ΕΛΙΚΟΦΟΡΟΥ*

*ΠΛΑΤΦΟΡΜΑΣ*

*Σπουδαστής: Καρασπύρος Δημήτριος*

*5. https://en.wikipedia.org/wiki/Rotation_matrix*

*6.ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ (Α.Τ.Ε.Ι) ΚΑΒΑΛΑΣ*

*ΣΧΟΛΗ ΤΕΧΝΟΛΙΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ*

*ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ*

*ΠΤΥΧΙΑΚΉ ΕΡΓΑΣΙΑ*

*ΘΕΜΑ: Προγραμματισμός Γραφικής Διεπαφής Χρήστη (GUI) στο Matlab για την μοντελοποίηση Συστημάτων από Αριθμιτικές Βάσεις Δεδομένων*

*Κισκορίδου Ιωάννα – Γεραντίδου Σημέλα*

*Καβάλα, Ιουνιος 2013*


## ΒΙΒΛΙΑ :


*7.Γεώργιος Π. Σύρκος Δρ. Πανεπιστημίου Rutgers, ΗΠΑ*

*Καθηγητής ΤΕΙ Πειραιά Τμήμα Αυτοματισμού*

*& Ιωάννης Κ. Κούκος Δρ. Πανεπιστημίου ImperialCollege, Ηνωμένο*

*Βασίλειο, Λέκτορας UMIST, Τμήμα Χημικών Μηχανικών*

*Εισαγωγή στη Σχεδίαση Συστημάτων Ελέγχου με το MATLAB®*

*8.Εισαγωγή στη MATLAB Γ.ΓΕΩΡΓΙΟΥ – Χ. ΞΕΝΟΦΩΝΤΟΣ*

*ΛΕΥΚΩΣΙΑ 2007*

*9. ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ''ΕΙΣΑΓΩΓΗ ΣΤΟ MATLAB-SIMULINK'' ΑΡΝΑΟΥΤΑΚΗΣ ΝΕΚΤΑΡΙΟΣ ΝΟΕΜΒΡΙΟΣ 2005*

*10.FORUMMATLAB*