ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΚΕΤΣΕΜΕΝΙΔΗΣ ΕΛΕΥΘΕΡΙΟΣ

Εισηγήτρια: Βελώνη Αναστασία

ΑΘΗΝΑ - 2018

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΚΕΤΣΕΜΕΝΙΔΗΣ ΕΛΕΥΘΕΡΙΟΣ
Α.Μ: 45105

Εισηγητήτρια: Βελώνη Αναστασία

Εξεταστική Επιτροπή:

…………………………………..
…………………………………..

Ημερομηνία εξέτασης ……………………………

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε σε ένα ενδιαφέρον γνωστικό αντικείμενο, την ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB. Την προσπάθειά μου αυτή υποστήριξε θερμά η καθηγήτρια και μέντορας μου κ Βελώνη Αναστασία την οποία θα ήθελα να ευχαριστήσω από τα βαθη της καρδιας μου για την ηθική και ψυχική στήριξη που μου παρείχε αυτά τα χρόνια, καθώς είναι ένας δυναμικός και γεμάτος ζωή άνθρωπος, ένα πρότυπο καθηγήτριας που ερχόταν στη αίθουσα με κέφι να μας διδάξει την δύναμη της γνώσης.

Δε μπορώ να παραλείψω τον εκλεκτό καθηγητή μου αλλά και πρόεδρο του Τμήματος κ. Ιωάννη Έλληνα ο οποίος πέρα από τις γνώσεις που μου πρόσέφερε με έμαθε να αισθάνομαι και συμπεριφέρομα σαν Μηχανικός.

Στην πάντα εξυπηρετική και πρόσχαρη Γραμματεία του Τμήματος μου θέλω να πω ένα μεγάλο ευχαριστώ για όλα.

Ακόμα θα ήθελα να ευχαριστήσω τον πνευματικό μου πατέρα Σεβασμιώτατο μητροπολίτη χαλκίδος κκ Χρυσόστομο Τριανταφύλλου για τις προσευχούλες που έκανε όλα αυτά τα χρόνια και πάντα με ξεκούραζε απο τις δύσκολες στιγμές. Πώς θα μπορούσα να ξεχάσω την υιοθετημένη μου γιαγιάκα κυρία Μαγδαληνή Αργυράκη που πάντα ήταν εκεί με τρομερή υπομονη και πάντα να μου λέει τη σωστή συμβουλή.

Για το τέλος άφησα την οικογένειά μου, τον πατέρα μου Ιορδάνη και την μήτερα μου Μαρία που παρόλες τις δυσκολίες που μας έτυχαν ήταν πάντα δίπλα μου και οτι κι αν γράψω θα ειναι λίγο .

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας Πτυχιακής εργασίας είναι η εισαγωγή στη θεωρία των σημάτων και συστημάτων μέσα από αναλυτικά παραδείγματα γραμμένα στο εργαλείο προγραμματισμού Matlab. Η ύλη που καλύπτεται εκτείνεται πέραν των αναλογικών σημάτων και συστημάτων και σε αυτά του διακριτού χρόνου. Η Πτυχιακή εργασία συνοδεύεται από ένα ΠΑΡΑΡΤΗΜΑ όπου επιλύονται θέματα που αφορούν τα Σήματα και Συστήματα με χρήση των λογισμικών SIMULINK και OCTAVE.

ABSTRACT

The purpose of this thesis is to introduce into the theory of signals and systems through analytical examples written in the Matlab programming tool. Matter covered extends beyond analog signals and systems and to discrete time. The Diploma Thesis is accompanied by an APPENDIX where issues related to Signals and Systems are solved using the SIMULINK and OCTAVE software.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Σήματα και Συστήματα, Μετασχηματισμοί στο πεδίο του χρόνου, Μετασχηματισμοί στο πεδίο της συχνότητας.

SCIENTIFIC AREA: Signals and Systems, Transformations in the field of time, Transformations in the frequency domain.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Σήματα, Συστήματα, Matlab, Μετασχηματισμός Laplace, Μετασχηματισμός Z, Μετασχηματισμός Fourier, SIMULINK, OCTAVE.

KEYWORDS: Signals, Systems, Matlab, Laplace Tranform, Z Transform, Fourier Transform SIMULINK, OCTAVE

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΠΕΡΙΕΧΟΜΕΝΑ

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΚΕΦΑΛΑΙΟ 1 - MATLAB

1.1 Τί είναι το Matlab

Το MATLAB είναι ένα πρόγραμμα υπολογιστών για όσους χρησιμοποιούν αριθμητικούς υπολογισμούς, ειδικά στη γραμμική άλγεβρα (πίνακες). Ξεκίνησε ως ένα πρόγραμμα "Εργαστηρίου Πινάκων" ("MATrixLABoratory") που είχε σκοπό να παρέχει αλληλεπιδρώσα προσπέλαση στις βιβλιοθήκες Linpack και Eispack. Από τότε έχει αναπτυχθεί αρκετά, για να γίνει ένα ισχυρότατο εργαλείο στην οπτικοποίηση, στον προγραμματισμό, στην έρευνα, στην επιστήμη των μηχανικών, και στις επικοινωνίες.

Στο δυναμικό του Matlab συμπεριλαμβάνονται μοντέρνοι αλγόριθμοι, δυνατότητες χειρισμού τεράστιων ποσοτήτων δεδομένων, και ισχυρά προγραμματιστικά εργαλεία. Το Matlab δεν είναι σχεδιασμένο για συμβολικούς υπολογισμούς, αλλά αντισταθμίζει αυτή την αδυναμία του επιτρέποντας στο χρήστη να συνδέεται άμεσα με το Maple. Η επιφάνεια αλληλεπίδρασης βασίζεται κυρίως σε κείμενο, γεγονός που μπορεί να συγχύσει μερικούς χρήστες.

Οι βασικοί λόγοι που έχουν καταστήσει το MATLAB ένα από τα πιο **δημοφιλή επιστημονικά πακέτα λογισμικού** είναι οι εξής:

α) το περιβάλλον του είναι φιλικό προς τον χρήστη

β) παρέχει άμεσες δυνατότητες γραφικής απεικόνιση

γ) έχει πληθώρα ενσωματωμένων συναρτήσεων

δ) παρέχει τη δυνατότητα προσθήκης συναρτήσεων γραμμένων από τον χρήστη

ε) ο προγραμματισμός στο MATLAB είναι απλός στ) περιλαμβάνει πληθώρα εργαλείων από διάφορες επιστημονικές περιοχές

Το MATLAB είναι ένα εύχρηστο και αρκετά ευέλικτο υπολογιστικό περιβάλλον για υλοποίηση επιστημονικών εφαρμογών σε ένα ευρύ φάσμα πεδίων, όπως η Γραμμική Άλγεβρα, η Στατιστική και η Επεξεργασία Σήματος και Εικόνας. Το περιβάλλον της MATLAB υποστηρίζει την εκτέλεση απλών μαθηματικών υπολογισμών αλλά και πιο σύνθετων λειτουργιών πάνω σε εξειδικευμένες περιοχές εφαρμογών καθώς περιέχει ένα σύνολο συναρτήσεων και εξωτερικών βιβλιοθηκών (Toolboxes) για εφαρμογές όπως η στατιστική ανάλυση δεδομένων κ.α. Επίσης η MATLAB έχει εξαιρετικές δυνατότητες για γραφικά και είναι εφοδιασμένη με αρκετές συναρτήσεις για εύκολο και ευέλικτο σχεδιασμό επίπεδων καμπυλών, τρισδιάστατων επιφανειών, ισοϋψών, παραμετρικών δισδιάστατων αλλά και τρισδιάστατων καμπυλών κα.

1.2 Πλεονεκτήματα και Μειονεκτήματα του Matlab

Το MatLab αποτελεί ένα πολύ δημοφιλή και με μεγάλη γενικά αποδοχή προγραμματιστικό περιβάλλον καθώς μπορεί να χρησιμοποιηθεί σε πάρα πολλούς διαφορετικούς τομείς – εκπαιδευτικούς και βιομηχανικούς. Κυριότερο πλεονέκτημά του είναι η εκτέλεση πράξεων σε πίνακες, το οποίο χρησιμεύει στην επεξεργασία εικόνων καθώς και στην ανάλυση συστημάτων ελέγχου.

Επιπλέον, είναι πολύ εύκολη η ανάπτυξη κώδικα γραμμένο σε MatLab, καθώς διαθέτει μια ισχυρή, τεχνική γλώσσα, απ' ότι για παράδειγμα σε γλώσσα προγραμματισμού C++. Ο κώδικας αυτός είναι συμπαγής και λιγότερος σε έκταση, επομένως ευκολότερος στην υλοποίηση. Για παράδειγμα, κώδικας 10 – 20 γραμμών σε C++, μπορεί να γραφτεί σε 1 με 2 γραμμές MatLab. Λαμβάνοντας αυτό υπόψιν, αντιλαμβανόμαστε την ευκολία που δίνει στην υλοποίηση προγραμμάτων μεγαλύτερου μεγέθους. Επίσης, η μεγάλη ποικιλία σε βιβλιοθήκες γλυτώνει τον χρήστη από την κατανάλωση άσκοπου χρόνου και ενέργειας στην εγκατάσταση και σύνθεση καινούργιων εργαλείων, ώστε να ικανοποιήσουν τις ανάγκες του για υλοποίηση του project. Αντίθετα, μ' αυτόν τον τρόπο, έχει την δυνατότητα να δοκιμάζει και να εκτελεί καινούργιες ιδέες στη στιγμή.

Μέρος της δημοφιλίας του προγράμματος προέρχεται λόγω της μεγάλης λειτουργικότητάς του, η οποία βασίζεται στην ευρεία βάση χρηστών του και την εκτενή καταγραφή και ανταλλαγή λύσεων και ιδεών μεταξύ τους μέσω του MathWorks. Όλο αυτό βοήθησε στην σχεδίαση και δημιουργία πολλών και διαφορετικών βιβλιοθηκών και εργαλείων ώστε να ικανοποιούν τις πολλές και ξεχωριστές ανάγκες των χρηστών. Παραδείγματος χάρη, πολλά projects έχουν άμεση και εύκολη λύση καθώς προηγούμενοι χρήστες δούλεψαν πάνω σ' αυτά οδηγώντας στην ανάπτυξη των κατάλληλων εργαλείων για τους επόμενους.Τέλος, το MatLab υπερέχει στην καταγραφή των συναρτήσεων καθώς προσφέρει ποικίλους τρόπους απεικόνισης των δεδομένων τα οποία είναι φιλικά προς το χρήστη.

Πρώτο και κυριότερο μειονέκτημα του MatLab είναι το κόστος. Μια πλήρης έκδοση του προγράμματος κοστίζει πέντε με δέκα φορές περισσότερο από έναν μεταγλωττιστή C ή Fortran. Ωστόσο, η χρήση του μειώνει σε σημαντικά επίπεδα το χρόνο που χρειάζεται ένα συγκεκριμένο project για να ολοκληρωθεί. Ακόμα κι έτσι όμως η αγορά του από κάποιον κρίνεται πολύ ακριβή. Γι' αυτό, υπάρχει η έκδοση student η οποία είναι σχεδόν ίδια με την πλήρη έκδοση και σε πολύ χαμηλότερη τιμή.

Επίσης το MatLab χρησιμοποιεί μια επεξηγηματική γλώσσα και έτσι εκτελεί τις εντολές πιο αργά απ' ότι θα έκαναν γλώσσες που χρησιμοποιούν μεταγλωττιστές. Το πρόβλημα αυτό , παρ' όλα αυτά, λύνεται με το σωστό «χτίσιμο» του προγράμματος στο MatLab, αυξάνοντας μ' αυτόν τον τρόπο την ταχύτητα εκτέλεσης του κώδικα και επομένως και των εργασιών.

1.3 Πώς δουλεύει το Matlab

Το Matlab δουλεύει εκτελώντας τις μαθηματικές εντολές που εισάγετε στο παράθυρο εντολών. Η προκαθορισμένη επιλογή είναι, κάθε έξοδος να τυπώνεται απευθείας στο παράθυρο.

Ακόμα, σας επιτρέπεται να εκχωρείτε ένα όνομα σε μία έκφραση για δική σας ευκολία. Να θυμάστε ότι το όνομα που εκχωρείτε είναι μόνο ένα όνομα, και δεν αναπαριστά καμία μαθηματική μεταβλητή (όπως θα έκανε στο Maple, για παράδειγμα). Κάθε όνομα πρέπει να έχει μία τιμή κάθε στιγμή. Αν προσπαθήσετε να διαβάσετε την τιμή ενός μη προσδιορισμένου ονόματος, θα πάρετε μήνυμα λάθους.

ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

Σχεδόν τα πάντα στο Matlab είναι πίνακες, είτε μοιάζουνε με τέτοιους είτε όχι. Αυτό μπορεί να σας πάρει λίγο μέχρι να το συνηθίσετε. Θα εισάγουμε τις λειτουργίες με το στυλ πίνακα μαζί με τα αριθμητικά τους ισοδύναμα, ώστε να μπορείτε να καταλάβετε τα καλούπια που παρουσιάζονται στο συντακτικό.

1.3.1 Σύμβολα και στίξη

Το Matlab σχεδιάστηκε για να χρησιμοποιεί σαφώς προκαθορισμένη σημειογραφία. Δοκιμάστε τα παρακάτω παραδείγματα στον υπολογιστή σας.

Είσοδος	**Έξοδος**	**Σχόλια**

`2 + 3`
`7-5`
`34*212`
`1234/5786`
`2^5`
	`ans = 5`
	`ans = 2`
	`ans = 7208`
	`ans = 0.2173`
	`ans = 32`
		Η αριθμητική δουλεύει όπως αναμενόταν. Σημειώστε ότι στο αποτέλεσμα δίνεται το όνομα "ans" κάθε φορά.

`a = sqrt(2)`	`a = 1.4142`	Μπορείτε να επιλέξετε τα ονόματα που θέλετε για τα διάφορα αντικείμενα.

`b = a, pi, 2 + 3i`	`b = 1.4142`
	`ans = 3.1416`
	`ans = 2.0000 + 3.0000i`
		Μπορείτε να χρησιμοποιήσετε κόμματα για να βάλετε περισσότερες από μία εντολές σε μία γραμμή. Τα Pi, i, και j είναι σημεία επαφής.

`c = sin(pi)` `eps`	`c = 1.2246e-016` `ans = 2.2204e-016`	Το "eps" είναι το τρέχον όριο ακρίβειας. Οτιδήποτε μικρότερο από "eps" είναι πιθανότατα μηδέν. Σημειώστε ότι το Matlab καταλαβαίνει (και περιμένει από εσάς να καταλαβαίνετε!) την επιστημονική σημειογραφία.
`d = [1 2 3 4 5 6 7 8 9]`	`d = 1 2 3 4 5 6 7 8 9`	Τα "d", "e", και "f" είναι όλα

`e = [1:9]` `f = 1:9`	`e = 1 2 3 4 5 6 7 8 9` `f = 1 2 3 4 5 6 7 8 9`	διανύσματα. Είναι ισοδύναμα. Παρατηρείστε τη χρήση του τελεστή ":" - μετράει (με μονάδες) από ένα νούμερο στο επόμενο.

`g = 0:2:10`
`f(3)`
`f(2:7)`
`f(:)`
	`g = 0 2 4 6 8 10`
	`ans = 3`
	`ans = 2 3 4 5 6 7`
	`1 2 3 4 5 6 7 8 9`
		Περισσότερες χρήσεις του συμβόλου ":". Παρατηρείστε ότι μπορείτε να το χρησιμοποιείτε για να πάρετε ένα κομμάτι ενός διανύσματος (ή πίνακα, ή κύβου, κλπ), ή για να το πάρετε ολόκληρο.

`h = [1 2 3];`
`h'`
	`ans = 1`
	`      2`
	`      3`
		Ένα ερωτηματικό ";" θα αποτρέψει την εμφάνιση της εξόδου. Ένα απλό σύμβολο " ' " υπολογίζει την αλληλομετάθεση ενός πίνακα, ή σε αυτή την περίπτωση, αλλάζει θέση μεταξύ διανυσμάτων-γραμμών και διανυσμάτων-στηλών.

`h * h'`
`h .* h`
`h + h`
	`ans = 14`
	`ans = 1 4 9`
	`ans = 2 4 6`
		Μαθηματικές πράξεις σε διανύσματα. Το * είναι πολλαπλασιασμός πινάκων, επομένως οι διαστάσεις πρέπει να ταιριάζουν.
		Το " .* " είναι πολλαπλασιασμός εισόδου με είσοδο.

`g = [ 1 2 3; 4 5 6; 7 8 9]`	`g = 1 2 3` `    4 5 6`	Εισάγοντας έναν πίνακα.

	`7 8 9`	

`g(2,3)`
`g(3,:)`
`g(2,3) = 4`
	`ans = 6`
	`ans = 7 8 9`
	`g = 1 2 3`
	`    4 5 4`
	`    7 8 9`
		Προσπελαύνοντας τα στοιχεία ενός πίνακα.
		Παρατηρείστε τη χρήση του συμβόλου ":" για να προσπελάσετε μία ολόκληρη γραμμή.

`g^2`
 	
`g .^ 2`
	`ans =  30  36  38`
	`        52  65  68`
	`       102 126 134`
	`ans =  1   4   9`
	`      16  25  16`
	`      49  64  81`
		Το πρώτο πολλαπλασιάζει τον πίνακα με τον εαυτό του. Το δεύτερο υψώνει στο τετράγωνο κάθε είσοδο του πίνακα.

1.3.2 Έλεγχος της εξόδου

Πριν προχωρήσουμε σε πιο εξειδικευμένα πράγματα στους πίνακες, θα ήταν φρόνιμο να αναφέρουμε θέματα μορφοποίησης.

Η εντολή που πρέπει να ξέρετε είναι `format` .

- Για να ελέγχετε τα κενά μεταξύ των γραμμών, χρησιμοποιείστε το `format compact` .
- Για να δείτε και τα 15 ψηφία που χρησιμοποιήθηκαν στον υπολογισμό, χρησιμοποιείστε το `format long` .
- Για να δείτε μόνο 5 ψηφία, χρησιμοποιείστε το `format short` .
- Για να αποτρέψετε εντελώς την έξοδο, χρησιμοποιείστε ένα ερωτηματικό στο τέλος της εντολής.

Για να δείτε κι άλλες επιλογές, πληκτρολογείστε `help format` .

Σημειώστε ότι το Matlab *πάντα* χρησιμοποιεί "διπλή" ακρίβεια (περίπου 15 ψηφία) υπολογισμούς του. Αυτές οι εντολές ρυθμίζουν μερικώς τον τρόπο που εμφανίζεται η έξοδος.

1.3.3 Πίνακες

1.3.3.1 Ενσωματωμένοι τρόποι κατασκευής

Υπάρχουν πολλοί ενσωματωμένοι τρόποι κατασκευής πινάκων. Ακολουθούν μερικοί:

Είσοδος	**Έξοδος**	**Σχόλια**

`rand(2)`
 	
`rand(2,3)`
	`ans = 0.3243 0.8301`
	`0.2640 0.6964`
	`ans = 0.3335 0.2878 0.2599`
	`0.5802 0.2640 0.6771`
		Δημιουργεί έναν πίνακα με στοιχεία εισόδου τυχαία κατανεμημένα ανάμεσα στα 0 και 1

`zeros(2)`
 	
`ones(2)`
	`ans = 0 0`
	`0 0`
	`ans = 1 1`
	`1 1`
		Δημιουργεί έναν 2x2 πίνακα με όλα τα στοιχεία εισόδου μηδενικά (ή ίσα με τη μονάδα).

`eye(2)`	`ans = 1 0` `0 1`	Ταυτοτικός πίνακας I.

`hilb(3)`	`ans = 1.0000 0.5000 0.3333`
	`0.5000 0.3333 0.2500`
	`0.3333 0.2500 0.2000`
		Πινακας Hilbert

1.3.3.2 Αλληλουχία

Μπορούν να σχηματιστούν νέοι πίνακες από τους παλιούς. Υποθέστε ότι έχουμε

```
-> a = [1 2; 3 4]


a = 1 2
    3 4
```

Τότε μπορούμε να φτιάξουμε έναν καινούργιο πίνακα με οποιονδήποτε από τους παρακάτω τρόπους.

Είσοδος	**Έξοδος**
`[a, a, a]`	`ans = 1 2 1 2 1 2` `3 4 3 4 3 4`

`[a; a; a]`	`ans = 1 2`
	`      3 4`
	`      1 2`
	`      3 4`
	`      1 2`
	`      3 4`

`[a, zeros(2); zeros(2), a']`	`ans = 1 2 0 0`
	`      3 4 0 0`
	`      0 0 1 3`
	`      0 0 2 4`

Το Matlab έχει όλες τις συνηθισμένες λειτουργίες ενσωματωμένες, καθώς και τις περισσότερες από τις πιο ασήμαντες.

Είσοδος	**Έξοδος**	**Σχόλια**

`k=[16 2 3;5 11 10;9 7 6]`	`k = 16  2  3`
	`     5 11 10`
	`     9  7  6`
		Ορίζει έναν πίνακα.

`rref(k)`	`ans = 1  0  0`
	`      0  1  0`
	`      0  0  1`
		Τύπος ελαχιστοποίησης γραμμών στο επίπεδο.

`rank(k)`	`ans = 3`	Η τάξη.
`det(k)`	`ans = -136`	Η ορίζουσα.

`inv(k)`	`ans =  0.0294 -0.0662  0.0956`
	`      -0.4412 -0.5074  1.0662`
	`       0.4706  0.6912 -1.2206`
		Ο αντίστροφος του πίνακα.

`[vec,val] = eig(k)`	`vec = -0.4712 -0.4975 -0.0621`
	`      -0.6884  0.8282 -0.6379`
	`      -0.5514  0.2581  0.7676`
	`val = 22.4319       0       0`
	`            0 11.1136       0`
	`            0       0 -0.5455`
		Τα ιδιοδιανύσματα και οι ιδιοτιμές του πίνακα.
		Οι στήλες του "vec" είναι τα ιδιοδιανύσματα,
		και τα διαγώνια στοιχεία του "val" είναι οι ιδιοτιμές

Υπάρχουν πολλές ακόμα συναρτήσεις όπως αυτές. Πληκτρολογείστε `help matfun` για να τις δείτε όλες.

1.3.3.3 Προγραμματισμός

Οι πίνακες μπορούν επίσης να κατασκευαστούν μέσω προγραμματισμού.
Ακολουθεί ένα παράδειγμα που χρησιμοποιεί δύο βρόγχους "for".

```
-> for i=1:10,
       for j=1:10,
           t(i,j) = i/j;
       end
   end
```

Παρατηρείστε ότι δεν υπάρχει έξοδος, αφού η μόνη εντολή που θα μπορούσε να παράγει έξοδο (`t(i,j) = i/j;`) τελειώνει με ερωτηματικό. Χωρίς το ερωτηματικό, το Matlab θα τύπωνε τον πίνακα t 100 φορές!

1.3.3.4 Περισσότερες λειτουργίες πινάκων

Όπως είδαμε και ενωρίτερα, τα +, -, *, και / ορίζονται διαισθητικά για τους πίνακες. Όταν υπάρχει αμφιβολία για το αν μία λειτουργία θα εκτελεί αριθμητικές πράξεις με τους πίνακες (σε αντίθεση με τις αριθμητικές πράξεις μεταξύ των στοιχείων εισόδου ένα-προς-ενα), να θυμάστε ότι το ".*" (τελεία-αστεράκι) θα πολλαπλασιάζει τα στοιχεία εισόδου ένα προς ένα, και το "*" (αστεράκι) θα εκτελεί πολλαπλασιασμούς πινάκων.

1.3.3.5 Αποθήκευση πινάκων

Σε αντίθεση με το Maple και το Mathematica, το Matlab δεν έχει την έννοια του φύλλου εργασίας ("worksheet"). Όταν βγείτε από τοMatlab, δεν θα προτραπείτε να σώσετε την εργασία σας.

Αν θέλετε ένα αντίγραφο της δουλειάς σας, μπορείτε να γυρίσετε σε "logging" πληκτρολογώντας

`diary 'c:\scratch\session.txt'` σε μηχανήματα NT

`diary '~/session.txt'` σε UNIX.

Παρατηρείστε ότι η έξοδος θα σωθεί παράλληλα με την είσοδο, οπότε το αρχείο δε μπορεί να χρησιμοποιηθεί άμεσα ως έγγραφο.

Μπορεί απλά να θέλετε να σώσετε έναν ή περισσότερους πίνακες. Η εντολή diary είναι ένας πολύ άχαρος τρόπος για να το κάνετε αυτό.

Για να σώσετε την τιμή της μεταβλητής "x" σε ένα αρχείο απλού κειμένου που ονομάζεται "x.value" χρησιμοποιείστε την

```
save x.value x -ascii
```

Για να σώσετε όλες τις μεταβλητές σε ένα αρχείο που λέγεται "mysession.mat" σε μορφή επαναφόρτωσης, χρησιμοποιείστε την

```
save mysession
```

Για να επανακτήσετε αυτό την προσπάθειά σας, χρησιμοποιείστε την

```
load mysession
```

Στη συνέχεια, για να δείτε τα σωσμένα αρχεία, σε συστήματα UNIX πληκτρολογείστε τις εντολές:

```
% more session.txt
% more x.value
```

Σε σύστημα Windows NT, ανοίξτε το κατάλληλο αρχείο με το Notepad.

1.3.4 Βαθμωτά μεγέθη

Ένα βαθμωτό μέγεθος είναι απλώς ένας αριθμός. Το Matlab τους αποθηκεύει εσωτερικά ως πίνακες, αλλά τους μεταχειρίζεται σαν να ήταν αριθμοί.

Όλες οι λειτουργίες που εμπλέκουν ένα βαθμωτό μέγεθος και έναν πίνακα επηρεάζουν ένα-προς-ενα τα στοιχεία εισόδου του πίνακα, με μία εξαίρεση: τον τελεστή της δύναμης ("^"). Με το a όπως ορίστηκε παραπάνω, δοκιμάστε τα παρακάτω:

Είσοδος	**Έξοδος**	**Σχόλια**
`b=2`	`b=2`	Ορίζει το b να είναι ένα βαθμωτό μέγεθος.
`a + b`	`ans = 3 4` `      5 6`	Η πρόσθεση δουλεύει στοιχείο-προς-στοιχείο.
`a * b`	`ans = 2 4` `      6 8`	Το ίδιο και ο πολλαπλασιασμός.
`a ^ b`	`ans = 7 10`	Αυτή είναι ύψωση *πίνακα* σε δύναμη - a*a

	`15 22`	
`a .^ b`	`ans = 1 4` `9 16`	Ύψωση σε δύναμη στοιχείου-προς-στοιχείο.

1.3.5 Διανύσματα

Ένα διάνυσμα είναι απλά ένας πίνακας με μία μόνο γραμμή (ή μία στήλη). Το Matlab κάνε μία διάκριση ανάμεσα στα διανύσματα-γραμμή και στα διανύσματα-στήλη, και θα βγάλει ανάλογο μήνυμα αν δεν πάρει αυτό που περίμενε.

Σε όλες τις αριθμητικές λειτουργίες, το Matlab μεταχειρίζεται ένα διάνυσμα ως πίνακα, οπότε γνωρίζουμε ήδη πώς να πολλαπλασιάζουμε ένα διάνυσμα με ένα βαθμωτό μέγεθος. Το Matlab επίσης σας δίνει τις παρακάτω λειτουργίες:

Είσοδος	**Έξοδος**	**Σχόλια**
`v = [1 2 3]` `u = [3 2 1]`	`v = [1 2 3]` `u = [3 2 1]`	Ορίζει ένα ζευγάρι διανυσμάτων
`v * u`	`Error`	Οι διαστάσεις δε συμφωνούν.
`v * u'`	`ans = 10`	Δουλεύει αν πάρουμε την αλληλομετάθεση.
`dot(v,u)`	`ans = 10`	Το εσωτερικό γινόμενο είναι το ίδιο πράγμα.
`cross(v,u)`	`ans = -4 8 -4`	Το εξωτερικό γινόμενο δουλεύει μόνο για 3-διάστατα διανύσματα.

1.4 Επίλυση εξισώσεων

Μία από τις κύριες χρήσεις των πινάκων είναι η αναπαράσταση συστημάτων γραμμικών εξισώσεων. Αν ο `a` είναι ένας πίνακας που περιέχει τους συντελεστές ενός συστήματος γραμμικών εξισώσεων, το `x` είναι ένα διάνυσμα-στήλη που περιέχει τους "αγνώστους", και το `b` είναι το διάνυσμα στήλη της "δεξιάς πλευράς", οι σταθεροί όροι, τότε η εξίσωση πινάκων *a x =b* αναπαριστά το σύστημα των εξισώσεων.

Αν ξέρετε λίγη γραμμική άλγεβρα, θα μπορούσατε να χρησιμοποιήσετε τις εντολές του Matlab που παρουσιάστηκαν πιο πάνω για να επαυξήσετε τον πίνακα και στη συνέχεια να βρείτε τον τύπο ελαχιστοποίησης γραμμών στο επίπεδο. Παρόλα αυτά, το Matlab παρέχει έναν πιο απλό μηχανισμό για την επίλυση γραμμικών εξισώσεων:

```
->  x = a \ b
```

Αυτό μπορεί να διαβαστεί ως "το x ισούται με το a-επί τον αντίστροφο του b."

Δοκιμάστε τον με τους πίνακες

```
-> a = [1 2 3; 4 5 6; 7 8 10]; b = [1 1 1]';
```

Θα πρέπει να πάρετε

```
x = -1 1 0
```

Για να επαληθεύσετε αυτό τον ισχυρισμό, δοκιμάστε το παρακάτω:

```
-> a*x, a*x - b, eps
```

Τα αποτελέσματα είναι:

```
ans = 1 1 1
ans = 1.0e-015 *
    -0.1110
    -0.6661
    -0.2220
ans = 2.2204e-016
```

Παρατηρείστε ότι το `a*x - b` βρίσκεται πολύ κοντά στο `eps` - που σημαίνει ότι είναι όσο πιο κοντά στο μηδέν γίνεται.

Αν δεν υπάρχει λύση, παρέχεται μία λύση "ελαχίστων τετραγώνων" (το a*x - b είναι το μικρότερο δυνατό). Εισάγετε

```
a(3,3) = 9; b = [1 1 0]';
```

(που κάνει τον πίνακα ιδιάζον και αλλάζει το b) και εισάγετε ξανά τις παραπάνω εντολές, χρησιμοποιώντας το επάνω βελάκι για να τις επαναφέρετε. Παρατηρείστε ότι η λύση είναι λίγο ανακριβής.

1.5 Γραφήματα

Το Matlab έχει εξαιρετικές δυνατότητες γραφικών (πρέπει να χρησιμοποιείτε κάποιο τερματικό που υποστηρίζει γραφικά για να χρησιμοποιείτε τις δυνατότητες). Παρόλα αυτά, χρησιμοποιώντας γραφικά στο Matlab είναι θεμελιωδώς διαφορετικό από το να χρησιμοποιείτε γραφικά στο Maple ή στο Mathematica.

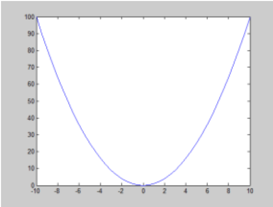
1.5.1 Έννοιες Γραφημάτων

Πριν να δούμε τις δυνατότητες καταγραφής σχεδίου του Matlab, σκεφτείτε τί *είναι* στην πραγματικότητα ένα γράφημα. Ένα γράφημα είναι ένα σύνολο σημείων, στις 2, 3 ή ακόμα και 4 διαστάσεις, τα οποία μπορεί να είναι ή και να μην είναι συνδεδεμένα με γραμμές ή πολύγωνα. Τα περισσότερα μαθηματικά πακέτα λογισμικού το κρύβουν αυτό από το χρήστη δειγματοληπτώντας από μία συνεχή συνάρτηση για να παράγει τα σημεία.

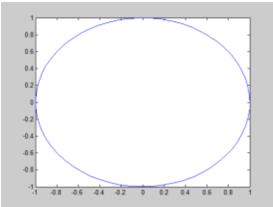
Το Matlab είναι σχεδιασμένο για να δουλεύει με πίνακες, και όχι με συναρτήσεις. Οι πίνακες είναι ένας αξιόπιστος τρόπος να αποθηκεύετε ένα σύνολο από νούμερα - που είναι ακριβώς αυτό που χρειάζεται όταν χρησιμοποιείτε γραφήματα. Άρα όλες οι εντολές για γραφήματα στο Matlab δέχονται *πίνακες* ως ορίσματά τους, σε αντίθεση με μία συνάρτηση. Αν έχετε συνηθίσει τα διαγράμματα σε μορφή συναρτήσεων, το Matlab μπορεί να σας πάρει κάποιο διάστημα να το συνηθίσετε. Από την άλλη μεριά, ο τρόπος που η προσέγγιση του Matlab κάνει πολύ εύκολο το να δει κανείς τα δεδομένα και να δημιουργήσει γραφήματα που βασίζονται σε λίστες σημείων.

Ένα άλλο εξαιρετικό χαρακτηριστικό του συστήματος γραφημάτων του Matlab είναι ο τρόπος με τον οποίο εμφανίζει τη γραφική έξοδο. Στο Matlab, υπάρχει (συνήθως) μόνο ένα παράθυρο για το διάγραμμα. Οι διαδοχικές εντολές για τη δημιουργία του διαγράμματος θα προστεθούν στο παλιό διάγραμμα, εκτός και αν ζητήσετε να δημιουργηθεί ένα καινούργιο. Αυτό επιτρέπει να φτιαχτεί ένα διάγραμμα, και στη συνέχεια να το ρυθμίσετε ώστε να ταιριάζει στις ανάγκες σας.

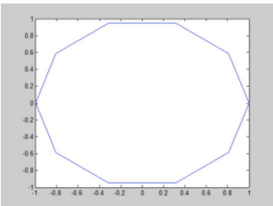
1.5.2 Βασικά 2-διάστατα Γραφήματα

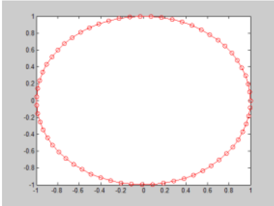
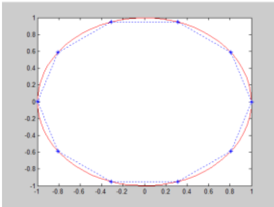
Είσοδος	**Έξοδος**	**Σχόλια**
`x = -10:0.5:10;` `y = x .^ 2;`	(καμία)	Δημιουργεί έναν 1x41 πίνακα που μετράει από το -10 στο ως 10, και έναν 1x41 πίνακα που προκύπτει υψώνοντας στο τετράγωνο τα στοιχεία του πρώτου πίνακα.
`plot(x,y)`		Σχεδιάζει τα σημεία. Κάθε ζευγάρι είναι σχεδιασμένο, ώστε το <x(1), y(1)> να είναι ένα σημείο, το <x(2), y(2)> να είναι ένα σημείο, κλπ.

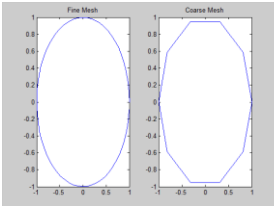
`t = 0:0.1:2*pi;`
`x = cos(t);`
`y = sin(t);`
	(καμία)	Παράγει κάποιους καινούργιους 1x63 πίνακες.

`plot(x,y)`		Σχεδιάζει τα σημεία. Αυτό λέγεται παραμετρική σχεδίαση. Παρατηρείστε ότι αντικαθιστά το προηγούμενο διάγραμμα.

`t = 0:pi/5:2*pi;`
`u = cos(t);`
`v = sin(t);`
	(καμία)	Παράγει κάποιους καινούργιους 1x11 πίνακες. Δείχνει πώς να ελέγχετε την "ανάλυση" του διαγράμματος.

`figure,plot(u,v)`		Δημιουργεί ένα νέο παράθυρο για το διάγραμμα, και σχεδιάζει τα σημεία. Παρατηρείστε πόσο ανομοιόμορφος είναι ο πίνακας, αφού χρησιμοποιήσαμε μόνο 11 δειγματικά σημεία.

`plot(x,y, 'ro-')`		Σχεδιάζει και πάλι την επίμαχη έκδοση με κόκκινο, με κύκλους στα στοιχεία δεδομένων, που συνδέονται με ευθείες.
`plot(x,y, 'r-',` `u,v,'b*:')`		Στο ίδιο διάγραμμα, σχεδιάζει την επίμαχη έκδοση με κόκκινο, και σχεδιάζει τη χαμηλής προτεραιότητας έκδοση με μπλε αστεράκια στα στοιχεία δεδομένων και με διακεκομμένες γραμμές.

`figure`
`subplot(1,2,1)`
`plot(x,y)`
`title('Fine Mesh')`
`subplot(1,2,2)`
`plot(u,v)`
`title('Coarse Mesh')`
		Δημιουργεί ένα καινούργιο σχήμα.
		Το χωρίζει σε μία γραμμή, δύο στήλες, και δίνει βάση στο πρώτο τετράγωνο.
		Σχεδιάζει.
		Δίνει στο τρέχον σχέδιο έναν τίτλο.
		Δίνει βάση στο δεύτερο τετράγωνο.
		Σχεδιάζει.
		Του δίνει έναν τίτλο.

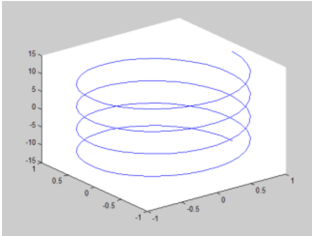
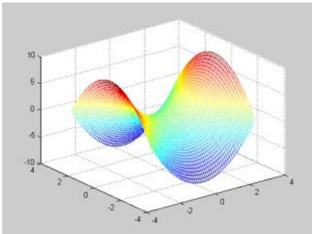
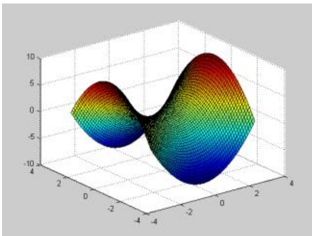
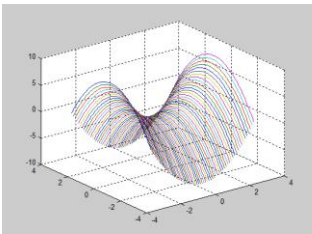
Δείτε τα παρακάτω αρχεία βοήθειας για περισσότερες επιλογές και ιδέες: `help plot`, `help comet`, `help semilogy` και `help fill` .

Το Matlab παρέχει πολύ ισχυρά χαρακτηριστικά στο παράθυρο σχεδίασης. Χρησιμοποιείστε τη μπάρα εργαλείων στην κορυφή για να προσθέσετε βέλη, γραμμές, και κείμενο με σχόλια στο διάγραμμά σας.

1.5.3 Βασικά 3-διάστατα Γραφήματα

Είσοδος	**Έξοδος**	**Σχόλια**

`t = -4*pi:pi/16:4*pi;`
`x = cos(t);`
`y = sin(t);`
	(καμία)	Παράγει τα δεδομένα...

`z = t;`		
`plot3(x,y,z)`		... και ζωγραφίζει ένα ελικοειδές..
`[x, y] = meshgrid(-3:0.1:3,-3:0.1:3);` `z = x.^2 - y.^2;`	(καμία)	Παράγει δεδομένα για μία επιφάνεια διαγράμματος.
`mesh(x,y,z)`		Ζωγραφίζει την επιφάνεια χρησιμοποιώντας ένα πλέγμα.
`surf(x,y,z)`		Ζωγραφίζει μία επιφάνεια ως μία "μπαλωμένη" επιφάνεια.
`plot3(x,y,z)`		Παρατηρείστε ότι εξακολουθεί να τη σχεδιάζει, αλλά ως ένα σύνολο από τοξοειδείς καμάρες.

1.6 Προγραμματισμός

1.6.1 Πώς να προγραμματίζετε

Μπορεί κανείς να δημιουργήσει ένα πρόγραμμα Matlab χρησιμοποιώντας τον editor της αρεσκείας του, όπως και μπορεί να το αποθηκεύσει σε ένα αρχείο για να το χρησιμοποιήσει αργότερα. Τέτοια αρχεία συνήθως ονομάζονται "m-αρχεία" (Matlab αρχεία) και πρέπει το όνομα τους να τελειώνει με "`.m`" (`π.χ. foo.m`). Η χρήση m-αρχείων αποτελεί καλή πρακτική προγραμματισμού στην Matlab, ειδικά εάν πρόκειται να χρησιμοποιήσετε ένα σύνολο εντολών πολλές φορές, και σίγουρα θα αυξήσει την παραγωγικότητα σας.

Διαλέξτε File -> New ->M-file για να ενεργοποιήσετε τον ενσωματωμένο editor/debugger της Matlab, ή χρησιμοποιήστε τον editor της αρεσκείας σας. Ακολούθως δημιουργείστε το παρακάτω αρχείο και αποθηκεύστε το με το όνομα sketch.m :

```
[x y] = meshgrid(-3:.1:3, -3:.1:3);
z = x.^2 - y.^2;
mesh(x,y,z);
```

Στη συνέχεια, στο Matlab, βεβαιωθείτε ότι το directory όπου ανήκε το m-αρχείο βρίσκεται στο path. Ελέγξετε το πληκτρολογώντας `pathtool` και επαληθεύοντας ότι το directory σας υπάρχει. (Όσοι δεν έχετε γραφικά μπορείτε να χρησιμοποιήσετε την εντολή `addpath` *`directory`* .)

Τώρα δώστε την εντολή

```
sketch
```

Το αποτέλεσμα είναι αυτό που θα παίρνατε εάν είχατε εισαγάγει τις τρεις γραμμές του αρχείου, μετά το σύμβολο προτροπής.

Επίσης μπορείτε να εισάγετε δεδομένα με τον εξής τρόπο: αν ένα αρχείο με το όνομα mymatrix.m και βρίσκεται στο τρέχον directory όπου δουλεύετε περιέχει τις γραμμές

```
A = [2 3 4; 5 6 7; 8 9 0];
```

τότε η εντολή `mymatrix` διαβάζει αυτό το αρχείο και παράγει τον A. Παρόλα αυτά, για μεγάλους πίνακες, είναι πιο ασφαλές να χρησιμοποιείτε τις εντολές `save` και `load` του Matlab.

1.6.2 Συναρτήσεις

Οι συναρτήσεις είναι όπως και κάθε άλλο m-αρχείο, αλλά δέχονται ορίσματα, και μεταγλωττίζονται την πρώτη φορά που χρησιμοποιούνται σε ένα δοσμένο %session% (για λόγους ταχύτητας).

Χρησιμοποιείστε τον αγαπημένο σας editor για να δημιουργήσετε ένα αρχείο με το όνομα sqroot.m, που να περιέχει τις παρακάτω γραμμές.

```
function sqroot(x)
% Compute square root by Newton's method

% Initial guess
xstart = 1;

for i = 1:100
        xnew = ( xstart + x/xstart)/2;
        disp(xnew);
        if abs(xnew - xstart)/xnew < eps, break, end;
        xstart = xnew;
end
```

Στο Matlab δώστε τις εντολές

```
format long
sqroot(19)
```

Θα πρέπει να δείτε την έξοδο της συνάρτησής σας.

Δύο προειδοποιήσεις:

- Μία συνάρτηση έχει πρόσβαση στις μεταβλητές που βρίσκονται στο "workspace" από το οποίο κλήθηκε, αλλά οι μεταβλητές που δημιουργούνται μέσα στη συνάρτηση (οι xstart και xnew, στο προηγούμενο παράδειγμα)είναι τοπικές, που σημαίνει ότι δεν είναι προσβάσιμες από το workspace από όπου καλείται η συνάρτηση. Για περισσότερες πληροφορίες, δείτε το κεφάλαιο "Προγραμματισμός Μ-Αρχείων" στο εγχειρίδιο, *Χρησιμοποιώντας το Matlab*.
- Παρατήρηση: αν διορθώνετε μία συνάρτηση κατά τη διάρκεια μίας προσπάθειας, χρησιμοποιείστε την `clear` *`function_name`* για να σβήσετε τη μεταγλωττισμένη έκδοση, ώστε να διαβαστεί η καινούργια.

1.6.3 Μαζικός Προγραμματισμός

Το MATLAB μπορεί να τρέξει σε "batch mode", με έναν παρόμοιο τρόπο. Αν ένα αρχείο με το όνομα "test.m" περιέχει τις εντολές που θέλετε να τρέξετε (που δεν υποστηρίζουν γραφικά), σε σύστημα UNIX πληκτρολογείστε:

```
% matlab < test.m > homework.out
```

Αυτό διαβάζεται ως "Τρέξε το MATLAB, με είσοδο από το test.in, και έξοδο στο test.out." Το αρχείο που δίνουμε ως είσοδο δε χρειάζεται να ονομάζεται κάτι-τελεία-m", αλλά **πρέπει** να τελειώνει με το `quit` .

1.6.4 Ιδέες προγραμματισμού

Τα "m-αρχεία" που υπάρχουν ήδη στοMATLAB παρέχουν πολλά παραδείγματα! Για να βρείτε τα σημεία όπου βρίσκονται χρησιμοποιείστε το `path`. Αυτό θα σας οδηγήσει και σε κάποια demos.

Μπορείτε ακόμα να δοκιμάσετε να πληκτρολογήσετε `demo` για να πάρετε μία ιδέα για το εύρος των εργασιών που μπορούν να επιτευχθούν με το Matlab.

Δοκιμάστε το `help function` για μία άσκηση προγραμματιστών.

1.7 ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ MATLAB

1.7.1 Πράξεις και βασικά σύμβολα

- **Πρόσθεση (Αφαίρεση) :** σύμβολο + (σύμβολο -).
- **Ίσον :** πληκτρολογούμε **Enter** .
- **Πολλαπλασιασμός : σύμβολο ***
- **Διαίρεση : σύμβολο** /
- **Στη δύναμη: σύμβολο** ^ π.χ. το x^y δίνει το x στη δύναμη y.
- **sqrt(z)** δίνει την τετραγωνική ρίζα του z. Αλλος τρόπος γραφουμε z^1/2
- **pi** είναι το π ≈3.14159.
- **i** είναι η φανταστική μονάδα $\sqrt{-1}$.
- **inf** είναι το +∞.
- **abs(z)]** δίνει την απόλυτη τιμή ενος πραγματικού ή ενός μιγαδικού αριθμού z .

1.7.2 Βασικές Συναρτήσεις

- **cos(z)** δίνει το cosine του z.
- **sin(z)** δίνει το sine του z
- **log(z)** δίνει τον φυσικό λογάριθμο του z (λογάριθμος με βάση το e).
- l**ogb(z)** δίνει το λογάριθμο στη βάση b.
- a**cos(z) δίνει το** arc του cosine δηλαδή το $\cos^{-1}(z)$ του z.
- Όμοια ορίζεται το **asin(z)**
- **sign(x)**δίνει το -1, 0 or 1 εξαρτάται πότε ο πραγματικός αριθμός x είναι αρνητικός, μηδέν ή θετικός.
- **exp(x)** δίνει την εκθετική συνάρτηση e^x.
- **diff(g)** παριστάνει την παράγωγο μιας συνάρτησης g
- **diff(g,x)** παριστάνει την παράγωγο μιας συνάρτησης g, ως προς τη μεταβλητή

1.7.3 Πως ορίζονται Συναρτήσεις, Ακολουθίες και Πίνακες

Για ορίσουμε μια συνάρτηση θα πρέπει όπως είπαμε να προκαθορίζουμε τις μεταβλητές.

Παράδειγμα για να ορίσουμε τη συνάρτηση $f(x,y,z) = x^2+y^2 + \cos(z)$ γράφουμε :

```
» syms x y z
» f=x^2+y^2+cos(z)
 f =
x^2+y^2+cos(z)
```

Προσοχή σε μια επόμενη εντολή θα αναφερθούμε στη συνάρτηση που ορίσαμε με το σύμβολο f . Παράδειγμα η μερική παράγωγος της $f(x,y,z) = x^2+y^2 + \cos(z)$ ώς προς z:

```
» diff(f,z)
  ans = -sin(z)
```

Όμοια **για να ορίσουμε μια ακολουθία** πρέπει να σημειώνουμε τον δείκτη ή τους δείκτες αυτής.

Παράδειγμα το άθροισμα της σειράς 1/n n=1 n=2

```
» n=sym('n')
n =     n
» symsum(1/n,1,2)
  ans = 3/2
```

Παράδειγμα ορισμού ενός πίνακα :

Εντολή

```
» a=[2,5;2,1]
a =

    2     5
    2     1
```

Τώρα για να πάρουμε τον αντίστροφο του πίνακα a γράφουμε

```
» a^-1
ans =

   -0.1250    0.6250
    0.2500   -0.2500
```

Το γινόμενο πινάκων ορίζεται με το συμβολο * έτσι

```
» (a^-1)*a

ans =

    1     0
    0     1
```

1.7.4 Διάφορες άλλες βασικές εντολές

- < μικρότερο, <= μικρότερο ή ίσο, όμοια > και **>= . ίσον == .**

! , ~ Ψευδές

- **min([x_1,x_2, ...])** δίνει τον ελάχιστο του διανύσματος **[x_1,x_2, ...]**

π.χ.

» min([1,6,7])

ans =1

- Όμοια ορίζεται **max([x_1,x_2, ...]) .**

- **Limit (παράσταση, 0) δίδει το όριο της παράστασης (πρέπει να έχει καθαρισθεί μια μεταβλητή) καθώς το** x τείνει στο 0.

Π.χ. Να ευρεθεί το όριο $\lim_{x \to 0}\sin(x)/x$ έχουμε:

Πρώτα ορίζουμε μεταβλητές

» syms x y

» limit(sin(x)/x,0)

ans =

1

1.7.5 *Αλγεβρικές εντολές*

- **expand(παράσταση) :** αναπτύσσει σε απλά πολυώνυμα μια παράσταση που είναι είτε γινόμενο πολυωνύμων ή (θετικές και ακέραιες) δύναμη πολυωνύμου.

Π.χ.

» expand((x+2)*(x-2))

ans = x^2-4

» expand((x-1)^3)

ans = x^3-3*x^2+3*x-1

- **factor(**πολυώνυμο) παραγοντοποιεί απλά πολυώνυμα

Π.χ.

» factor(x^2 -4)

ans = (x+2)*(x-2)

- **factor(ακέραιος)**

Δίνει τη λίστα των πρώτων παραγόντων του ακεραίου αριθμού

factor(325)

ans = 5 5 13

- **mod(m, n)** δίνει το υπόλοιπο της διαίρεσης του m δια του n.

- **floor(x)** δίνει τον ακέραιο, που είναι μικρότερος ή ίσος του x.
- **ceil(x)** δίνει τον ακέραιο, που είναι μεγαλύτερος ή ίσος του x.

π.χ.

floor(2.4) ans= 2

ceiling(2.6) ans = 3.

Για το γινόμενο πινάκων χρησιμοποιούμε το σύμβολο * και δια τον αντίστροφο πίνακα το σύμβολο ^-1 ή το σύμβολο inv( ).

Π.χ.

```
» A=[1,2;3,4]
A =  1    2
     3    4
» B=[1,1;3,7]
B =    1    1
       3    7
» A*B
ans = 7    15
```

    15   31

A^-1

ans =

    -2.0000   1.0000
     1.5000   -0.5000

1.7.6 Ολοκλήρωση

Για την ολοκλήρωση κατ αρχήν ορίζουμε τις μεταβλητές και τις συναρτήσεις.
Μετά γράφουμε για το αόριστο ολοκλήρωμα **int(συνάρτηση, μεταβλητή)**
Για να υπολογίσουμε ένα ορισμένο ολοκλήρωμα γράφουμε
int(συνάρτηση, μεταβλητή, ελάχιστο,μέγιστο).

Π.χ.

» syms x

» f=1/x

f =1/x

» int(f,x,1,2)   ans =log(2)

» int(f,x)   ans =log(x)

1.7.7 Λύση εξισώσεων και συστημάτων

Για τη λύση μιας εξίσωσης μιας μεταβλητής x γράφουμε:

[x]=solve(‘εξίσωση’)

- **ΠΡΟΣΟΧΗ** το ίσον που συνδέει τα δύο μέλη μιας εξίσωσης συμβολίζεται με ΕΝΑ ίσον, δηλαδή = .
- **π.χ.**

» solve('2*x^2 + 3*x + 5=0')

ans =

[-3/4+1/4*i*31^(1/2)]

[-3/4-1/4*i*31^(1/2)]

Όμοια για τη λύση συστήματος των μεταβλητών x,y γράφουμε

[x,y]=solve('εξίσωση 1', 'εξίσωση 2')

π.χ.

- **» [x,y]=solve('x+ 2*y=4','x+y=3')**
-
- **x = 2**
- **y =1**

1.7.8 Γραφικές Παραστάσεις

Το λογισμικό Matlab μας δίνει τη δυνατότητα να κατασκευάσουμε δι-διάστατα γραφήματα. Η εντολή είναι (αφού ορίζουμε πρώτα τη μεταβλητή ή και τη συνάρτηση)

ezplot(συνάρτηση, ελάχιστο, μέγιστο)

π.χ. η συνάρτηση του Gauss στο διάστημα (-3,3)

```
syms x y
» f=exp(-x^2)
f =
exp(-x^2)
» ezplot(f,-3,3)
```

Παρόλο που, όπως είδαμε στο προηγούμενο παράδειγμα, υπάρχουν μερικές συμβολικού τύπου εντολές η λογική των εντολών του λογισμικού Matlab για την κατασκευή γραφημάτων είναι διαφορετική από αυτή του Mathematica.

Πρέπει να τονίσουμε ότι το Matlab κάνει κυρίως γραφήματα μιάς λίστας.

Για τη γραφική παράσταση μιας λίστας g γράφεται με την εντολή plot(g)

Παράδειγμα θέλουμε να κάνουμε το γράφημα της sin(1/t) στο [0,1] Γράφουμε πρώτα τη λίστα

t=0:1/1000:1

» g=sin(t.^-1)

(προσοχή στην εντολή t.^-1 σημαίνει ότι θα πάρω το 1/α κάθε στοιχείου α της λίστας t).

Μετά κάνουμε το γράφημα της λίστας με την εντολή

» plot(g)

Τριδιάσταση παράσταση καμπύλης (που ορίζεται με παραμετρική εξίσωση {x(t),y(t)})

Ορίζουμε πρώτα μια λίστα τιμών t (στό παράδειγμα από 0 έως 10 π με βήμα π/50), μετά γράφουμε plot3(x(t),y(t),t)

Π.χ.

» t=0:pi/50:10*pi

» plot3(cos(t),sin(t),t)

1.7.9 Αθροίσματα

Πρέπει να τονίσουμε ότι το Matlab κάνει κυρίως αθροίσματα μιας λίστας.

Για το άθροισμα λίστας g γράφουμε την εντολή sum(g)

Παράδειγμα το άθροισμα της sin(n)/n από n=1 έως n=100.

» t=0:1/100:1

» g=sin(t)./t (προσοχή στην εντολή ./t)

» sum(g)

ans = 1.0604

1.8 Χρήσιμες εντολές του MATLAB

`who` Η εντολή εμφανίζει τις μεταβλητές στο περιβάλλον εργασίας

`whos` Η εντολή είναι όπως η who με επιπλέον πληροφορίες ως προς τον τύπο των μεταβλητών και τον χώρο που καταλαμβάνουν σε bytes.

`x = input(`string`)` ή `x = input ([`string1,string2,…,stringN`])`

Η εντολή `input` εμφανίζει ένα μήνυμα τύπου string (ή συνένωση από strings) στην οθόνη και αναμένει είσοδο από το πληκτρολόγιο. Η είσοδος αποθηκεύεται στη μεταβλητή αριστερά της ισότητας (π.χ. στο x). Το string (συμβολοσειρά) πρέπει να περικλείεται σε απλά απόστροφα.

`disp(`string`)` ή `disp([`string1,string2,…,stringN`])`

Η εντολή `disp` εμφανίζει ένα ή περισσότερα strings στην οθόνη.

Παράδειγμα:

```
>> x = input('Dose enan arithmo:  ')
Dose enan arithmo:  2
x =
     2
>> disp(['O arithmos einai o ',int2str(x)]);
O arithmos einai o 2
>>
```

`int2str(x)` μετατρέπει τον ακέραιο x σε string

`num2str(x)` μετατρέπει τον πραγματικό αριθμό x σε string `pause` σταματάει προσωρινά την εκτέλεση μέχρι να πατηθεί κάποιο πλήκτρο

`pause(n)` παύση n δευτερολέπτων

`pause on` ή `pause off`

ενεργοποίηση/απενεργοποίηση των pause που ακολουθούν στο πρόγραμμά μας (για να τρέχουν και σε non-interactive mode)

`echo on` και `echo off`

ενεργοποιεί/απενεργοποιεί το echoing ενός script αρχείου. Εμφανίζει τόσο τις εντολές όσο και τα σχόλια του προγράμματός μας. Μέσω του `echo` δεν είναι πάντα απαραίτητα τα `disp` για την εμφάνιση μηνυμάτων.

1.8.1 SET PATH -- ΚΑΘΟΡΙΣΜΟΣ ΤΟΥ PATH ΣΤΟ MATLAB

Το MATLAB αναζητεί ονόματα από script αρχεία, functions ή και αρχεία δεδομένων είτε των βιβλιοθηκών του ή του χρήστη, ελέγχοντας τους φακέλους με τη σειρά που έχουν καταχωρηθεί από την `SET PATH`.

Η εισαγωγή ενός νέου path (π.χ. A:\ ή C:\TEMP\) γίνεται με επιλογή του SET PATH από το menu FILE ή ενός απλού κλίκ στο εικονίδιο του PATH BROWSER. Τα βήματα που ακολουθούμε είναι τα εξής:

1. κάνουμε κλικ στο Browse… και επιλέγουμε το PATH που επιθυμούμε να προσθέσουμε
2. ενεργοποιούμε (κάνοντας απλό κλικ) το Current Directory ώστε να εμφανισθούν τα αρχεία που αναγνωρίζει το MATLAB στο δεξί παράθυρο

3. Επιλέγουμε το Add to Path… από το menu Path του Path Browser

4. Στο dialog-box που εμφανίζεται επιλέγουμε προσθήκη στο τέλος Add to back και κάνουμε κλικ στο OK

5. Επιλέγουμε το Save Path από το menu FILE.

6. Επιλέγουμε το Exit Path Browser από το menu FILE.

Η διαγραφή ενός PATH γίνεται με επιλογή του Remove from Path από το menu Path του Path Browser και στη συνέχεια ακολουθούμε τα βήματα 5 και 6.

Τέλος, αν θέλουμε να δώσουμε προτεραιότητα σε κάποιους φακέλους, μπορούμε να τα μεταφέρουμε με hold-and-drag πιό πάνω ή πιό κάτω στη σειρά των paths.

1.8.2 ΔΗΜΙΟΥΡΓΙΑ SCRIPT ΑΡΧΕΙΩΝ ΜΕ ΤΟΝ EDITOR ΤΟΥ MATLAB

Μπορούμε να δημιουργήσουμε script αρχεία στο MATLAB μέσω του editor, επιλέγοντας:

File → New → M-file

Τα script αρχεία στο MATLAB έχουν την επέκταση ".m" και ονομάζονται M-αρχεία. Μια άλλη συνηθισμένη κατηγορία αρχείων στο MATLAB είναι τα αρχεία δεδομένων (ASCII ή Binary) με επέκταση ".mat".

Ενα script αρχείο περιλαμβάνει το σύνολο των εντολών ενός προγράμματος που θα εκτελεστούν σειριακά από τον διερμηνευτή του MATLAB. Τα σχόλια (αρχίζουν από %) δεν εκτελούνται, όμως όταν εμφανίζονται στις πρώτες γραμμές του script μπορούν να χρησιμοποιηθούν ως *online help* μέσω της εντολής `help`. Για παράδειγμα, έστω ότι ένα script περιλαμβάνει τα εξής σχόλια και εντολές:

```
%Paradeigma eisagwgns onomatos kai emfanisns tou.
%To programma katharizei tis metablntes kai tnv othovn
%kai zntaei to ovoma tou xrnstn. Stn suvexeia to
%emfavizei stnv othovn.
```

```
clear; clc; format compact; echo on;

str=input('Please type your name enclosed in quotes: ');

%Paradeigma Xrnsns tou echo avti tou disp('Hit a key...')

echo on

%Hit any key to continue

echo off

pause   %Pausn

%Deikse to ovoma

disp(['To ovoma sou eivai ',str]);

disp('End of program...');
```

LOOPS - CONTROL STATEMENTS (σχεδόν όπως στη C)

<u>for loop</u>

```
j = 0;
for i=1:10          %με βήμα 1
   j = j + 1;
end
j                   %για να δούμε την τιμή στο j
```

```
k = 0;
for i=0:5:20        %με βήμα 5
   for j=1:3        %διπλό loop, το j με βήμα 1
      k = i*5+j;
   end
end
k

help for
```

while loop

```
j = 5;
while j>0          % while (συνθήκη = true) do
   disp(['j = ',num2str(j)]);
   j = j - 1;
end
j

help while
```

if … then … else

```
j = 5;
k = -4;
if ((j~=k+1) & (k>=0)) | j==abs(k)
   k = k+j;
elseif j==k+1
   k = k+2*j;
else
   k = 0;
end
k
```

```
help if
help relop
help elseif
```

1.9 Η ΣΧΕΣΗ ΤΟΥ MATLAB ΜΕ ΤΟ SIMULINK

Το **Simulink** είναι ένα περιβάλλον block διαγράμματος που χρησιμοποιείται για προσομοίωση πολλαπλών επιπέδων (multidomain) καθώς και για μοντελοποίηση. Επιπλέον υποστηρίζει τη δημιουργία αυτόματου κώδικα και συνεχείς δοκιμές κι επαλήθευση συστημάτων.

Το Simulink παρέχει έναν γραφικό συντάκτη και προσαρμόσιμες block βιβλιοθήκες. Επίσης έχει την δυνατότητα να δίνει λύσεις σε μοντελοποίηση και προσομοίωση δυναμικών συστημάτων. Το περιβάλλον του Simulink είναι ενοποιημένο με το MatLab, δίνοντας του τη δυνατότητα να ενσωματώνει αλγόριθμους του τελευταίου σε μοντέλα και να εξάγει αποτελέσματα προσομοίωσης σ’ αυτό για περαιτέρω ανάλυση.

Το Simulink είναι το κομμάτι αυτό του Matlab που μας δίνει τη δυνατότητα να δημιουργήσουμε ένα καινούριο σύστημα. Το σύστημα αυτό μπορεί να έχει αθροιστές, φίλτρα, γεννήτριες σήματος κ.λ.π. Το κάθε ένα κομμάτι από του ολικού συστήματος (όπως αυτά που αναφέραμε παραπάνω) ονομάζεται μπλόκ. Συνδέοντας όπως επιθυμούμε όλα τα επιλεγμένα μπλοκ δημιουργούμε ένα ολοκληρωμένο σύστημα.

Υπάρχουν δύο τρόποι για να δημιουργήσουμε ένα σύστημα.

Ο πρώτος τρόπος περιγράφεται πολύ περιληπτικά παρακάτω. Αν πληκτρολογήσουμε στο περιβάλλον του Matlab την εντολή ‘SIMULINK’ και μετά πατήσουμε το enter στο πληκτρολόγιο του υπολογιστή, τότε εμφανίζεται στην οθόνη μας η βιβλιοθήκη με τους καταλόγους των ήδη υπαρχόντων μπλοκ. Από εδώ και πέρα χρησιμοποιώντας μόνο το mouse μπορούμε να τοποθετήσουμε σε ένα νέο παράθυρο, που θα ανοίξουμε από την επιλογή FILE της βιβλιοθήκης, ότι μπλοκ επιθυμούμε. Οι συνδέσεις γίνονται πάλι χρησιμοποιώντας το mouse.

Πατώντας το αριστερό πλήκτρο του mouse και, κρατώντας το πατημένο, 'σέρνουμε' την έξοδο του ενός μπλοκ στην είσοδο του επόμενου.

Ένας δεύτερος τρόπος για να δημιουργήσουμε ένα σύστημα είναι να εκμεταλλευτούμε τις εντολές του Simulink.

1.10 Η ΣΧΕΣΗ ΤΟΥ MATLAB ΜΕ ΤΟ OCTAVE

Το **Octave** είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού , που προορίζετε κυρίως για αριθμητικούς υπολογισμούς. Παρέχει μια διασύνδεση γραμμής εντολών (CLI) για την επίλυση γραμμικών και μη γραμμικών αριθμητικών προβλημάτων, καθώς και για την εκτέλεση άλλων αριθμητικών πειραμάτων χρησιμοποιώντας γλώσσα που είναι ως επί το πλείστον συμβατή με το MATLAB.

Το Octave αρχικά δημιουργήθηκε σαν ένα συνοδευτικό λογισμικό σε ένα βιβλίο προπτυχιακού μαθήματος για σχεδιασμό χημικού αντιδραστήρα. Σήμερα αναπτύσσεται υπό την επίβλεψη του Dr. John W. Eaton. Διατίθεται με την άδεια GNU General Public License. Η χρησιμότητα του Octave αυξάνεται καθώς είναι συντακτικά συμβατό με το MATLAB που συχνά χρησιμοποιείται στη βιομηχανία και στα πανεπιστήμια.

Το Octave έχει κατασκευαστεί κυρίως ώστε να είναι συμβατό με το MATLAB. Μοιράζεται ουσιαστικά πολλά κοινά χαρακτηριστικά μαζί του όπως:

- Πίνακες ως βασικό τύπο δεδομένων.
- Η ενσωματωμένη υποστήριξη των μιγαδικών αριθμών.
- Ισχυρές ενσωματωμένες λειτουργίες μαθηματικών και εκτενείς βιβλιοθήκες
- Επεκτασιμότητα υπό μορφή συναρτήσεων ορισμένων από το χρήστη.

Αλλά υπάρχουν και πολλές διαφορές. Ας δούμε μερικές από αυτές.

- ✓ Το Octave υποστηρίζει τελεστές αυτόματης προσαύξησης και των φορέων ανάθεσης όπως η C ενώ το Matlab όχι. Πχ i++; ++i; i +=1; Κλπ
- ✓ Το Matlab μπορεί να φορτώσει άδεια αρχεία ενώ το Octave όχι.
- ✓ Το Octave εκτυπώνει στην οθόνη με τις εντολές fprintf και printf ενώ το Matlab

όχι.

- ✓ Για να συνεχίσουμε το κώδικα στην επόμενη γραμμή στο Matlab πρέπει να γράψουμε '…' ενώ στο Octave μπορούμε απλά να συνεχίσουμε από κάτω.
- ✓ Λογικές Πράξεις:
  Όχι-ίσο: Octave '~=' ή '!=' Matlab '~='
  Και: Octave '&' or '&&' Matlab '&'
  Ή: Octave '|' or '||' Matlab '|'
- ✓ Το Matlab ξεκινάει τα σχόλια στον κώδικα με % ενώ το Octave με % ή #
- ✓ Για την ύψωση σε δύναμη, Octave μπορεί να χρησιμοποιήσει '^' ή '**', Matlab απαιτεί '^'.

Επίσης μια σημαντική διαφορά είναι ότι δεν υπάρχουν όλες οι συναρτήσεις που έχει το Matlab στο Octave.

ΚΕΦΑΛΑΙΟ 2 - ΣΗΜΑΤΑ ΚΑΙ ΣΥΣΤΗΜΑΤΑ

2.1 Σήματα – Ταξινόμηση των Σημάτων

Ορίζουμε το ***σήμα*** (*signal*) σαν μία μονοσήμαντη συνάρτηση χρόνου , που μεταφέρει πληροφορίες. Συνεπώς , για κάθε χρονική στιγμή (ανεξάρτητη μεταβλητή) υπάρχει μία μοναδική τιμή της συνάρτησης (εξαρτημένη μεταβλητή). Η τιμή αυτή μπορεί να είναι ένας πραγματικός αριθμός , όπου στην περίπτωση αυτή έχουμε ένα πραγματικό σήμα (*real-valued signal*) , ή μπορεί να είναι ένας μιγαδικός αριθμός , όπου στην περίπτωση αυτή έχουμε ένα μιγαδικό σήμα (*complex-valued signal*). Σε όλες τις περιπτώσεις , η ανεξάρτητη μεταβλητή (δηλαδή ο χρόνος) είναι πραγματικός αριθμός.

Σε μια συγκεκριμένη περίπτωση , βρίσκουμε ότι η πιο κατάλληλη μέθοδος για την απεικόνιση των σημάτων εξαρτάται από το συγκεκριμένο τύπο του σήματος που θεωρούμε. Ανάλογα με το χαρακτηριστικό που ενδιαφέρει , μπορούμε να ξεχωρίσουμε τέσσερις διαφορετικές κατηγορίες σημάτων.

(1) Περιοδικά Σήματα , Απεριοδικά Σήματα.

Περιοδικό σήμα (*periodic sugnal*) g(t) είναι μια συνάρτηση , η οποία ικανοποιεί τη συνθήκη

$$g(t) = g(t + T_0) \quad (2.1)$$

για κάθε t , όπου t δηλώνει χρόνο και T_0 είναι μια σταθερά. Η μικρότερη τιμή της T_0, η οποία ικανοποιεί αυτή τη συνθήκη , ονομάζεται περίοδος του g(t). Συνεπώς , η περίοδος T_0 καθορίζει τη διάρκεια ενός πλήρους κύκλου του g(t).

Κάθε σήμα , για το οποίο δεν υπάρχει τιμή του T_0 , που να ικανοποιεί τη συνθήκη της Εξ. (2.1) , ονομάζεται *μη περιοδικό ή απεριοδικό σήμα* (*nonperiodic or aperiodic signal*) .

(2) Ντετερμινιστικά Σήματα , Τυχαία σήματα.

Ντετερμινιστικό σήμα *(deterministic signal)* είναι αυτό , για το οποίο δεν υπάρχει καμία αβεβαιότητα , όσον αφορά την τιμή του σε κάθε χρονική στιγμή. Συνεπώς , βρίσκουμε ότι τα ντετερμινιστικά σήματα μπορούν να αναπαρασταθούν σαν πλήρως καθορισμένες συναρτήσεις του χρόνου.

Από την άλλη μεριά , ***τυχαίο σήμα*** *(random signal)* είναι εκείνο , για το οποίο υπάρχει κάποιος βαθμός αβεβαιότητας , προτού καν εμφανισθεί. Ένα τέτοιο σήμα μπορεί να θεωρηθεί ότι ανήκει σε μία συλλογή ή σύνολο σημάτων , όπου κάθε σήμα του συνόλου είναι διαφορετικό.

(3) Ενεργειακά Σήματα , Σήματα Ισχύος.

Ορίζουμε την ***ολική ενέργεια*** *(total energy)* ενός σήματος g(t) σαν

$$E = \lim_{T\to\infty} \int_{-T}^{T} |g(t)|^2 dt = \int_{-\infty}^{\infty} |g(t)|^2 dt \qquad (2.2)$$

και τη ***μέση ισχύ*** *(average power)* του σαν

$$P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |g(t)|^2 dt \qquad (2.3)$$

Λέμε ότι το σήμα g(t) είναι ***ενεργειακό σήμα*** *(energy signal)* , αν , και μόνο αν , η ολική ενέργεια του σήματος ικανοποιεί τη συνθήκη

$$0 < E < \infty \qquad (2.4)$$

Λέμε ότι το σήμα g(t) είναι ***σήμα ισχύος*** *(power signal)* , αν και μόνο αν , η μέση ισχύς του σήματος ικανοποιεί τη συνθήκη :

$$0 < P < \infty \qquad (2.5)$$

(4) Αναλογικά Σήματα , Ψηφιακά σήματα.

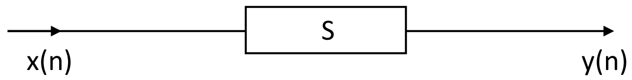
Ένα ***αναλογικό*** *σήμα (analog signal)* είναι συνεχής συνάρτηση του χρόνου , με συνεχές πλάτος επίσης. Αναλογικά σήματα δημιουργούνται όταν μία φυσική κυματομορφή , όπως ένα ακουστικό ή οπτικό κύμα μετατρέπεται σε ηλεκτρικό σήμα. Η μετατροπή επιτυγχάνεται μέσω ενός *μετατροπέα (transducer).* Στα παραδείγματα περιλαμβάνονται: το μικρόφωνο , που μετατρέπει ηχητικές μεταβολές πίεσης σε αντίστοιχες μεταβολές τάσης ή ρεύματος και το φωτοηλεκτρικό κύτταρο , που κάνει το ίδιο για μεταβολές έντασης του φωτός .
Από την άλλη πλευρά , ένα σήμα *διακριτού χρόνου (discrete-time)* ορίζεται μόνο σε διακριτές χρονικές τιμές. Έτσι , σ' αυτή τη περίπτωση η ανεξάρτητη μεταβλητή λαμβάνει μόνο διακριτές τιμές , που είναι συνήθως ομοιόμορφα κατανεμημένες. Συνεπώς , σήματα διακριτού χρόνου περιγράφονται σαν σειρές δειγμάτων , των οποίων τα πλάτη μπορούν να λάβουν συνεχείς τιμές. Όταν κάθε δείγμα ενός σήματος διακριτού χρόνου είναι *κβαντισμένο (quantized)* (δηλαδή το πλάτος επιτρέπεται να λάβει μόνο ένα πεπερασμένο σύνολο διακριτών τιμών) και στη συνέχεια *κωδικοποιημένο (coded)* , το τελικό σήμα αναφέρεται σαν *ψηφιακό σήμα (digital signal).* Η έξοδος από έναν ψηφιακό υπολογιστή είναι ένα παράδειγμα ψηφιακού σήματος . Φυσικά , ένα αναλογικό σήμα μπορεί να μετατραπεί σε ψηφιακή μορφή με *δειγματοληψία (sampling)* στο χρόνο , κβαντισμό και κωδικοποίησή του.

2.2 Συστήματα

Ένα Σύστημα διακριτού χρόνου , επενεργώντας σε ένα σήμα διακριτού χρόνου x(n) , έχει σαν αποτέλεσμα ένα άλλο διακριτό σήμα y(n). Όταν y(n)=x(n) , $\forall n$, τότε λέμε ότι έχουμε να κάνουμε με ένα **ταυτοτικό σύστημα.** Το αρχικό σήμα x(n) λέγεται **είσοδος** του συστήματος , ενώ το αποτέλεσμα , δηλαδή το σήμα y(n) , λέγεται **έξοδος** του συστήματος. Ο μετασχηματισμός που το σύστημα επιφέρει στο σήμα εισόδου μπορεί , είτε να εκφράζεται συναρτησιακά , είτε όχι. Εν γένει , μπορούμε να παρουσιάσουμε ένα σύστημα σαν ένα **τελεστή** που εφαρμόζεται στο σήμα x(n):

$$y(n) = S\left[x(n)\right] \tag{2.6}$$

Το σχήμα 2.1 δίνει γραφικά την προηγούμενη σχέση



Σχήμα 2.1: Γραφική αναπαράσταση συστήματος , εισόδου και εξόδου

Το σύστημα του προηγούμενου σχήματος είναι γνωστό σαν ένα σύστημα μίας εισόδου – μίας εξόδου ή **σύστημα SISO** (Single-input , single – out – put). Πολλά συστήματα του φυσικού κόσμου έχουν περισσότερες εισόδους ή και εξόδους. Ένα σύστημα με πολλές εισόδους και μία έξοδο , όπως ένας αθροιστής , είναι γνωστό σαν σύστημα MISO (Multi-input , single output). Συστήματα με πολλές εισόδους και πολλές εξόδους είναι γνωστά σαν **συστήματα MIMO** (Multi-input , Multi output). Η έννοια του συστήματος είναι πολύ γενική και πολλές σχέσεις αιτίας – αποτελέσματος , είτε ηλεκτρικών , είτε μηχανικών , είτε οικονομικών μεγεθών μπορούν να την υιοθετήσουν. Τα περισσότερα φυσικά συστήματα είναι συνεχούς χρόνου. Τα διακριτά συστήματα ή συστήματα διακριτού χρόνου συνδέονται γενικά με την δραστηριότητα του ανθρώπου.

2.2.1 Βασικές ιδιότητες των Συστημάτων

Γραμμικότητα: Ένα γραμμικό σύστημα είναι εκείνο , στο οποίο εφαρμόζεται η αρχή της **επαλληλίας** (superposition).

Αυτό σημαίνει ότι , αν η είσοδος του συστήματος μπορεί να αναλυθεί σε ένα άθροισμα – με τον κατάλληλο συντελεστή βάρους – πολλών σημάτων , τότε η έξοδος είναι η επαλληλία ή υπέρθεση των αποκρίσεων του συστήματος σε κάθε ένα από τα σήματα της εισόδου .

Ένα σύστημα λέγεται γραμμικό αν :

α) Η απόκριση στο σήμα $x_1(n)+x_2(n)$ είναι $y_1(n)+y_2(n)$.

β) Η απόκριση στο $\alpha x_1(n)$ είναι $\alpha y_1(n)$.

Οι δύο προηγούμενες προτάσεις μπορούν να συνδυασθούν ως εξής:

$$\alpha x_1(n)+bx_2(n) \Rightarrow \alpha y_1(n)+by_2(n) \quad (2.7)$$

Χρονικά αναλλοίωτο: Λέγεται ένα σύστημα , όταν μία χρονική μετάθεση του σήματος εισόδου έχει σαν αποτέλεσμα μόνο την ίδια χρονική μετάθεση του σήματος εξόδου.

$$x(n) \to y(n) \Rightarrow x(n-m) \to y(n-m) \quad (2.8)$$

Συστήματα με μνήμη και χωρίς μνήμη: Ένα σύστημα λέγεται χωρίς μνήμη , αν η είσοδός του κάθε χρονική στιγμή εξαρτάται από την είσοδο , την ίδια μόνο στιγμή. Το σύστημα της επόμενης σχέσης είναι χωρίς μνήμη:

$$y(n)=3x^2(n)-1 \quad (2.9)$$

Αντίθετα , τα συστήματα:

$$y(n)=x(n-1) \quad (2.10)$$

και

$$y(n) = \sum_{k=-\infty}^{n} x(k) \qquad (2.11)$$

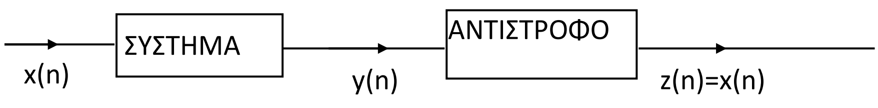
είναι συστήματα με μνήμη.

Αιτιατά και μη αιτιατά Συστήματα: Ένα σύστημα λέγεται αιτιατό , αν η έξοδός του σε κάθε χρονική στιγμή εξαρτάται από τις τιμές της εισόδου στο παρόν και στο παρελθόν. Τα συστήματα των σχέσεων (2.9) , (2.10) και (2.11) είναι αιτιατά. Αντίθετα , το σύστημα:

$$y(n) = \frac{1}{2}\left[x(n-1) + x(n+1)\right] \qquad (2.12)$$

δεν είναι αιτιατό , σύμφωνα με τον προηγούμενο ορισμό.
Στην πράξη χρησιμοποιούμε πολλές φορές μη αιτιατά συστήματα , κυρίως όταν δεν πρόκειται για εφαρμογές εντός γραμμής. Είναι προφανές ότι όλα τα συστήματα χωρίς μνήμη είναι και αιτιατά.

Αντιστρεψιμότητα: Ένα σύστημα είναι αντιστρέψιμο , αν , από την παρατήρηση της εξόδου , μπορούμε να οδηγηθούμε στην ανακατασκευή της εισόδου. Άμεσο συμπέρασμα είναι ότι , αν ξέρουμε το αντίστροφο και το συνδέσουμε «σε σειρά» με το αρχικό σύστημα , όπως στο σχήμα (2.2) , καταλήγουμε σε ένα **ταυτοτικό σύστημα**.



Σχήμα 2.2: Αντιστρεψιμότητα συστήματος

Ευστάθεια: Πρόκειται για μια πολύ σπουδαία και χρήσιμη ιδιότητα των συστημάτων. Ένα σύστημα είναι ευσταθές , όταν , από εισόδους φραγμένου πλάτους, προκύπτουν έξοδοι φραγμένου πλάτους. Πρόκειται για ένα είδος ευστάθειας που είναι γνωστή σαν ευστάθεια BIBO (Bounded input-bounded output).

2.3 Απόκριση συχνότητας ΓΧΑ - συστημάτων

Θα θεωρήσουμε μια ειδική κατηγορία εισόδων , τα σήματα της μορφής:

$$x(n) = e^{j\Omega n} \qquad -\infty < n < \infty \qquad (2.13)$$

και θα εξετάσουμε την απόκριση ενός ΓΧΑ συστήματος σε τέτοια σήματα. Θέτοντας $x(n) = e^{j\Omega n}$

$$y(n) = \sum_{i=-\infty}^{\infty} h(i)e^{j\Omega(n-i)} = e^{j\Omega n} \sum_{i=-\infty}^{\infty} h(i)e^{-j\Omega i} \qquad (2.14)$$

Ονομάζουμε:

$$H(\Omega) = \sum_{i=-\infty}^{\infty} h(i)e^{-j\Omega i} \qquad (2.15)$$

Ο μετασχηματισμός $h(i) \to H(\Omega)$ είναι γνωστός σαν **μετασχηματισμός Fourier** της ακολουθίας h(n).

Ο προηγούμενος ορισμός γενικεύεται για οποιοδήποτε σήμα x(n) , για το οποίο ορίζεται ένα νέο σήμα X(Ω) σαν μετασχηματισμός Fourier του x(n):

$$X(\Omega) = \sum_{i=-\infty}^{\infty} x(i)e^{-j\Omega i} \qquad (2.16)$$

Προκύπτει ότι:

$$y(n) = e^{j\Omega n} \bullet H(\Omega) = x(n) \bullet H(\Omega) \quad (2.17)$$

Η ποσότητα Η(Ω) που ορίζεται από την σχέση (2.74) ονομάζεται **απόκριση συχνότητας** του ΓΧΑ – συστήματος. Η ονομασία αυτή είναι δικαιολογημένη , αφού , ουσιαστικά , η συνάρτηση $e^{j\Omega n}$ αντιπροσωπεύει την οικογένεια των ημιτονοειδών συναρτήσεων . Η απόκριση συχνότητας αφορά συστήματα στη μόνιμη κατάσταση (Γενικά , όταν αναφερόμαστε σε ημιτονικές εισόδους , υποθέτουμε ότι αυτές έχουν εφαρμοσθεί στο μακρινό παρελθόν , ώστε τα πιθανά μεταβατικά φαινόμενα να έχουν αποσβεσθεί.).

Επειδή , γενικά , η συνάρτηση Η(Ω) είναι μιγαδική μπορεί να εκφρασθεί με τις εξής δύο μορφές:

$$H(\Omega) = \mathrm{Re}\left[H(\Omega)\right] + j\,\mathrm{Im}\left[H(\Omega)\right] \quad (2.18)$$

$$H(\Omega) = \left|H(\Omega)\right| e^{j\varphi(\Omega)} \quad (2.19)$$

Η δεύτερη έκφραση εισάγει το πλάτος και τη φάση , κατά την οποία μεταβάλλεται το σήμα εισόδου. Στην πράξη , χρησιμοποιείται κυρίως η (2.78).

Η καμπύλη $\left|H(\Omega)\right|$, συναρτήσει της συχνότητας , ονομάζεται **χαρακτηριστική πλάτους** , ενώ η καμπύλη φ=φ(Ω) ονομάζεται **χαρακτηριστική φάσης** του συστήματος. Αν η είσοδος είναι το σήμα

$$x(n) = A\cos(\Omega n + \theta) \quad (2.20)$$

τότε , με χρησιμοποίηση του τύπου του Euler , μπορεί κανείς να εκφράσει την έξοδο ως εξής:

$$y(n) = A\left|H(\Omega)\right|\cos(\Omega n + \theta + \varphi(\Omega)) \quad (2.21)$$

Η απόκριση συχνότητας ενός ΓΧΑ-συστήματος έχει τις εξής ιδιότητες:

α) Είναι συνεχής συνάρτηση του Ω.

β) Είναι περιοδική συνάρτηση του Ω με περίοδο 2π.

γ) Για h(n) ευρισκόμενα στο πεδίο των πραγματικών αριθμών , το $|H(\Omega)|$ είναι άρτια συνάρτηση , ενώ το φ(Ω) είναι περιττή συνάρτηση , στο διάστημα (-π , π). Το αυτό ισχύει για τις ποσότητες Re[H(Ω)] και Im[H(Ω)] , αντιστοίχως.

2.4 Συνέλιξη

Είναι ίσως η πιο σπουδαία έννοια της θεωρίας σήματος και είναι αυτή που συνδέει τη θεωρία σήματος με την θεωρία συστημάτων , αφού , βασικά , εκφράζει τη **γραμμικότητα** και το **χρονικό αναλλοίωτο** των συστημάτων. Η σχέση εισόδου – εξόδου ενός **γραμμικού χρονικά αναλλοίωτου** (ΓΧΑ) συστήματος εκφράζεται σαν ένα συνελικτικό άθροισμα. Επίσης , άλλες σπουδαίες έννοιες της επεξεργασίας σημάτων , όπως η **αυτοσυσχέτιση** και η **ετεροσυσχέτιση** είναι συναφείς με την συνέλιξη.
Από την πρακτική πλευρά , η συνέλιξη είναι στην βάση πλήθους υπολογισμών που σχετίζονται με πραγματοποίηση συστημάτων. Η ύπαρξη ταχέων αλγορίθμων για τον υπολογισμό της συνέλιξης της δίνει άλλη μία διάσταση , με σημασία.

Ορισμός: Η συνέλιξη δύο σημάτων x(n) και h(n) είναι ένα τρίτο σήμα y(n) που ορίζεται ως:

$$y(n) \overset{\Delta}{=} x(n) * h(n) \overset{\Delta}{=} \sum_{k=-\infty}^{\infty} x(k)h(n-k) \qquad (2.22)$$

Για να συγκλίνει το άπειρο άθροισμα του δευτέρου μέλους της (3.1) , πρέπει ένα από τα δύο σήματα x(n) & h(n) να είναι σήμα ενέργειας. Αν και τα δύο σήματα είναι σήματα ισχύος , τότε μπορεί να χρησιμοποιηθεί ένας λίγο διαφορετικός ορισμός της συνέλιξης:

$$y(n)^{\Delta} = x(n) * h(n)^{\Delta} = \lim_{M\to\infty} \frac{1}{(2M+1)} \sum_{k=-M}^{M} x(k)h(n-k) \qquad (2.23)$$

Ας σημειωθεί αμέσως ότι στις προηγούμενες σχέσεις το κάθε δείγμα σήματος y(n) εξαρτάται **από όλα** τα δείγματα των σημάτων x(n) και h(n).

Ιδιότητες:

α) Αντιμεταθετική ιδιότητα.

$$x(n)*h(n)= h(n)*x(n) \qquad (2.24)$$

β) Προσεταιριστική ιδιότητα.

$$x(n)*[h(n)*g(n)]=[x(n)*h(h)]*g(n) \qquad (2.25)$$

γ) Επιμεριστική ιδιότητα.

$$x(n)*[h(n)+g(n)]=x(n)*h(n)+x(n)*g(n) \qquad (2.26)$$

Απόδειξη της αντιμεταθετικής ιδιότητας

$$x(n) * h(n) = \sum_{k} x(k)h(n-k) = \sum_{l} x(n\text{-}l)h(l) = \sum_{l} h(l)x(n-l) = h(n) * x(n),$$

όπου l= n-k (αλλαγή μεταβλητής).

Απόδειξη της προσεταιριστικής ιδιότητας

Θέτουμε z(n)=h(n)*g(n)

Τότε:

$$z(n) = \sum_{k} h(n)g(n-k) = \sum_{l} h(n-l)g(l)$$
$$x(n) * z(n) = \sum_{k} x(k)z(n-k)$$

Αλλά $z(n-k) = \sum_{l} h(n-k-l)g(l)$, άρα:

$$x(n) * z(n) = \sum_{k} x(k) \sum_{l} h(n-k-l) \bullet g(l) = \sum_{l}\sum_{k} x(k)h\left[(n-l)-k\right]g(l)$$

Θέτουμε:

$$y(n) = x(n) * h(n) = \sum_{k} x(k)h(n-k)$$

Επομένως:

$$\sum_{k} x(k)h\left[(n-l)-k\right] = y(\hat{n}-l)$$
$$x(n) * z(n) = \sum y(n-l)g(l) = y(n) * g(n)$$

Ας σημειωθεί ότι , για $h(n) \stackrel{\Delta}{=} \delta(n)$, από την σχέση (3.1) και την σχέση που εκφράζει το τυχόν σήμα ως άθροισμα μοναδιαίων παλμών προκύπτει ότι

$$x(n) * \delta(n) = x(n) \qquad (2.27)$$

Επομένως , το σήμα δ(n) είναι το ταυτοτικό στοιχείο της συνέλιξης.

Υπολογισμός της συνέλιξης: Ο υπολογισμός της συνέλιξης ανάγεται στις εξής πράξεις:

ΑΝΑΚΛΑΣΗ ( ) , ΜΕΤΑΘΕΣΗ ΣΤΟΝ ΧΡΟΝΟ ($\leftrightarrow$) , ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ (x) , ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΕΠΙ ΔΙΑΦΟΡΙΚΟ (ΜΙΚΡΟ) ΧΡΟΝΟ (dt), ΟΛΟΚΛΗΡΩΣΗ ($\int$) , ΕΠΑΝΑΛΗΨΗ ( ).

2.5 ΑΝΑΛΥΣΗ FOURIER

<u>Περιοδικές συναρτήσεις</u>

Μία συνάρτηση $f(x)$ λέμε ότι έχει περίοδο ή είναι περιοδική αν για κάθε x έχουμε:

$$f(x+P) = f(x) \quad (2.28)$$

όπου P είναι μία θετική σταθερά. Η μικρότερη θετική τιμή του P καλείται ελάχιστη περίο-δος ή απλά περίοδος της $f(x)$.

π.χ. η συνάρτηση $\sin x$ έχει περιόδους $2\pi, 4\pi, \ldots, 2n\pi$, γιατί:

$$\sin x = \sin(x+2n\pi)$$

Η περίοδός της, ή ελάχιστη περίοδος, είναι 2π.

π.χ. η συνάρτηση $\tan x$ έχει περίοδο π.

<u>Τμηματικά συνεχείς συναρτήσεις</u>

Μία συνάρτηση $f(x)$ λέγεται <u>τμηματικά συνεχής</u> ή <u>συνεχής κατά τμήματα</u> σ` ένα διάστημα αν:

(i) Το διάστημα αυτό μπορεί να διαιρεθεί σ` ένα πεπερασμένο πλήθος υποδια-στημάτων σε κάθε ένα από τα οποία η $f(x)$ είναι συνεχής.

(ii) Υπάρχουν τα όρια της $f(x)$ όταν το x τείνει μονόπλευρα στα άκρα κάθε διαστήματος και είναι πεπερασμένα.

Το όριο της $f(x)$ από δεξιά συμβολίζεται συχνά με:

$$\lim_{\varepsilon \to 0} f(x+\varepsilon) = f(x+0) \quad \text{όπου } \varepsilon > 0 \quad (2.29)$$

Το όριο της $f(x)$ από αριστερά συμβολίζεται συχνά με:

$$\lim_{\varepsilon \to 0} f(x-\varepsilon) = f(x-0) \quad (2.30)$$

2.5.1 Ορισμός της σειράς Fourier

Έστω ότι η $f(x)$ είναι ορισμένη στο διάστημα $(-L, L)$ και ότι έξω από αυτό το διάστημα ορίζεται με τη σχέση $f(x+2L) = f(x)$, δεχόμαστε δηλαδή ότι η $f(x)$ έχει περίοδο $2L$. Η σειρά Fourier ή το ανάπτυγμα Fourier που αντιστοιχεί στην $f(x)$ είναι εξ ορισμού το:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} \left(\alpha_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right) \quad (2.31)$$

όπου οι συντελεστές Fourier a_n και b_n είναι:

$$a_n = \frac{1}{L} \int_{-L}^{L} f(x) \cos\frac{n\pi x}{L} dx \quad (2.32)$$

$$n = 0, 1, 2, \ldots$$

$$b_n = \frac{1}{L} \int_{-L}^{L} f(x) \sin\frac{n\pi x}{L} dx \quad (2.33)$$

Εάν η $f(x)$ έχει περίοδο $2L$ οι συντελεστές a_n και b_n μπορούν να προσδιοριστούν και από τις σχέσεις:

$$a_n = \frac{1}{L} \int_{c}^{c+2L} f(x) \cos\frac{n\pi x}{L} dx \quad (2.34)$$

$$b_n = \frac{1}{L} \int_{c}^{c+2L} f(x) \sin\frac{n\pi x}{L} dx \quad (2.35)$$

όπου c είναι ένας πραγματικός αριθμός. Επίσης είναι εύκολο να δούμε ότι:

$$\frac{a_0}{2} = \frac{1}{2L} \int_{-L}^{L} f(x) dx \quad (2.36)$$

ΘΕΩΡΗΜΑ Dirichlet

Εάν:

(i) Η $f(x)$ είναι ορισμένη και μονότιμη στο διάστημα $(-L, L)$ εκτός ίσως από ένα πεπερασμένο πλήθος σημείων.

(ii) Η $f(x)$ είναι περιοδική με περίοδο $2L$.

(iii) Οι $f(x)$ και $f'(x)$ είναι τμηματικά συνεχείς στο $(-L, L)$,

τότε η σειρά (1) συγκλίνει με συντελεστές (2) (3) ή (4) (5) στην:

(a) $f(x)$, εάν το x είναι σημείο συνεχείας

(b) $\dfrac{f(x+0)+f(x-0)}{2}$ εάν το x είναι σημείο ασυνέχειας.

(iv) Το ολοκλήρωμα $\int_0^{2L} |f(x)|\,dx$ είναι πεπερασμένο.

Σύμφωνα με το Θεώρημα μπορούμε να γράψουμε:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(\alpha_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right) \qquad (2.37)$$

ή

$$\frac{1}{2}\left[f(x+0)+f(x-0)\right] = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(\alpha_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right) \qquad (2.38)$$

όταν το x είναι σημείο ασυνέχειας.

Οι συνθήκες (i) (ii) και (iii) είναι ικανές αλλά όχι αναγκαίες!!

Περιττές και άρτιες συναρτήσεις

Μία συνάρτηση $f(x)$ καλείται περιττή, εάν:

$$f(-x) = -f(x)$$

π.χ. $x^3,\ x^5 - 3x^3 + 2x,\ \sin x,\ \tan x$

Μία συνάρτηση $f(x)$ καλείται άρτια, εάν:

$$f(-x) = f(x)$$

π.χ. $x^4,\ 2\mathrm{x}^6 - 4x^2 + 5,\ \cos x,\ e^x + e^{-x}$

Η σειρά Fourier που αντιστοιχεί σε μία περιττή συνάρτηση περιέχει μόνο όρους με ημίτονα, ενώ η σειρά Fourier που αντιστοιχεί σε μία άρτια συνάρτηση περιέχει μόνο συνη-μίτονα και πιθανώς μία σταθερά.

Όταν θέλουμε την ημιτονική ή συνημιτονική σειρά Fourier ορίζουμε την συνάρτηση στο διάστημα $(0, L)$ που είναι το μισό του διαστήματος $(-L, L)$ και μετά ορίζουμε την συνάρτηση έξω από αυτό έτσι ώστε να είναι περιττή ή άρτια αντίστοιχα, οπότε ορίζεται προφανώς και στο άλλο μισό διάστημα $(-L, 0)$. Οπότε έχουμε:

$a_n = 0 \qquad b_n = \frac{2}{L}\int_0^L f(x)\sin\frac{n\pi x}{L}dx$ για σειρά ημιτόνων

(2.39)

$b_n = 0 \qquad a_n = \frac{2}{L}\int_0^L f(x)\cos\frac{n\pi x}{L}dx$ για σειρά συνημίτονων (2.40)

Η ταυτότητα του PARSEVAL

$$\frac{1}{L}\int_{-L}^{L}[f(x)]^2 dx = \frac{a_0^2}{2} + \sum_{n=1}^{\infty}(a_n^2 + b_n^2) \qquad (2.41)$$

όπου a_n και b_n είναι οι συντελεστές Fourier για την $f(x)$ και υποθέτουμε ότι η $f(x)$ ικα-νοποιεί τις συνθήκες του Dirichlet.

Ομοιόμορφη σύγκλιση

Θεωρούμε μία σειρά με απείρους όρους $\sum_{n=1}^{\infty} U_n(x)$. Ορίζουμε το μερικό άθροισμα των R πρώτων όρων της σειράς με την σχέση:

$$S_R(x) = \sum_{n=1}^{R} U_n(x) \qquad (2.42)$$

Η σειρά των απείρων όρων λέγεται συγκλίνουσα ή λέμε ότι συγκλίνει στην $f(x)$ σε κάποιο διάστημα εάν:

$$\forall \varepsilon > 0 \text{ και } \forall x \text{ εντός του διαστήματος}$$

$$\exists N > 0 : |S_R(x) - f(x)| < \varepsilon \text{ , για } \quad R > N$$

όπου $N = N(\varepsilon, x)$. Η $f(x)$ καλείται άθροισμα της σειράς των απείρων όρων.

Όταν το N εξαρτάται από το ε αλλά όχι από το x στο διάστημα αυτό τότε λέμε ότι η σειρά συγκλίνει ομοιόμορφα στην $f(x)$.

Δύο αξιοσημείωτες ιδιότητες των σειρών που συγκλίνουν ομοιόμορφα δίνονται από τα παρακάτω Θεωρήματα.

ΘΕΩΡΗΜΑ: Εάν κάθε όρος μιας σειράς με απείρους όρους είναι συνεχής στο διάστημα (a,b) και η σειρά συγκλίνει ομοιόμορφα στην $f(x)$ στο διάστημα αυτό, τότε:

1. η $f(x)$ είναι επίσης συνεχής στο διάστημα αυτό
2. η σειρά μπορεί να ολοκληρωθεί κατά όρους δηλ.:

$$\int_a^b \left\{\sum_{n=1}^{\infty} U_n(x)\right\} dx = \sum_{n=1}^{\infty} \int_a^b U_n(x)dx \qquad (2.44)$$

ΘΕΩΡΗΜΑ: Εάν κάθε όρος μίας σειράς με απείρους όρους έχει παράγωγο και η σειρά των παραγώγων συγκλίνει ομοιόμορφα, τότε η αρχική σειρά μπορεί να παραγωγισθεί κατά όρους δηλ.:

$$\frac{d}{dx}\sum_{n=1}^{\infty} U_n(x) = \sum_{n=1}^{\infty} \frac{d}{dx} U_n(x) \qquad (2.45)$$

ΘΕΩΡΗΜΑ: (Κριτήριο του Weirstrass) Εάν υπάρχει ένα σύνολο σταθερών M_n , $n = 1, 2,$ 3,…. τέτοιο ώστε για κάθε x σ' ένα διάστημα να είναι $|U_n(x)| \le M_n$ και εάν ακόμα η $\sum_{n=1}^{\infty} M_n$ συγκλίνει, τότε η $\sum_{n=1}^{\infty} U_n(x)$ συγκλίνει ομοιόμορφα στο διάστημα αυτό.

ΘΕΩΡΗΜΑ: Εάν η $f(x)$ είναι τμηματικά συνεχής στο $-L \le x \le L$ και τα a και x είναι σημεία του διαστήματος αυτού, τότε η σειρά Fourier που αντιστοιχεί στην $f(x)$ μπορεί να ολοκληρωθεί κατά όρους από a έως x και να δώσει μια σειρά που συγκλίνει στην $\int_a^x f(u)du.$

Μιγαδική μορφή της σειράς Fourier

$$e^{i\theta} = \cos\theta + i\sin\theta \quad , \quad e^{-i\theta} = \cos\theta - i\sin\theta \qquad i^2 = -1$$

Μπορούμε να γράψουμε την σειρά Fourier για την $f(x)$

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{n\pi x}{L}} \qquad (2.46)$$

ΚΕΦΑΛΑΙΟ 3 - ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ

3.1 ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 1 – ΕΚΜΑΘΗΣΗ ΤΟΥ MATLAB

Το εργαστήριο **Σημάτων και Συστημάτων** διεξάγεται στο περιβάλλον του προγράμματος MATLAB με χρήση τόσο του βασικού κορμού του πακέτου που παρέχει πληθώρα έτοιμων ενσωματωμένων συναρτήσεων και πλούσια γραφικά όσο και μερικών εργαλειοθηκών (**toolboxes**) του. Σε σύντομο χρονικό διάστημα, ο χρήστης μπορεί να χρησιμοποιεί τις έτοιμες υπορουτίνες για την επίλυση προβλημάτων από διάφορες εφαρμογές.

3.1.1 ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΟΥ MATLAB

Για να μπούμε στο MATLAB κάνουμε **διπλό κλίκ** στο εικονίδιο του MATLAB και για να βγούμε πληκτρολογούμε **quit**. Το προτρεπτικό σήμα (prompt) του MATLAB είναι το >>.

3.1.2 Απλές αριθμητικές πράξεις

Το MATLAB χρησιμοποιεί τους τελεστές **+**, **-**, ***** και / για τις τέσσερις αριθμητικές πράξεις. Παραδείγματος χάριν:

```
3/5

ans =
0.6
```

3 + 5 + 2

ans =

10

3^3

ans =

27

Για πιό πολύπλοκες εκφράσεις χρησιμοποιούνται παρενθέσεις κατά τον συνήθη τρόπο:

2^5 + 4*(33 - 2*(6+2/7))

ans =

113.7143

3.1.3. Ενσωματωμένες συναρτήσεις

Το MATLAB μας παρέχει ένα πλήθος ενσωματωμένων συναρτήσεων όπως τετραγωνική ρίζα, εκθετικές και λογαριθμικές συναρτήσεις, τριγωνομετρικές και αντίστροφες τριγωνομετρικές συναρτήσεις κ.ά.:

```
sqrt(2)
              % τετραγωνική ρίζα
```

ans = **1.4142**

```
exp(1)
              % εκθετική συνάρτηση
```

ans = **2.7183**

```
log(exp(1))  % φυσικός λογάριθμος
```

ans = **1**

```
log10(10^2)         % δεκαδικός λογάριθμος
```

ans = **2**

% Τριγωνομετρικές συναρτήσεις

pi

% η σταθερά π

ans **3.1416** **=**

sin(pi/4) **% ημίτονο**

ans = **0.7071**

cos(pi/2) **% συνημίτονο**

ans **6.1232e-017** **=** **← πρακτικά το αποτέλεσμα είναι 0**

tan(pi/4) **% εφαπτομένη**

ans = **1.0000**

asin(0.5) **% τόξο ημιτόνου**

ans =

0.5236

```
atan(1)  % τόξο εφαπτομένης
```

ans =

0.7854

όπου το σύμβολο % χρησιμοποιείται για την εισαγωγή σχολίων. Αν η γωνία δίνεται σε μοίρες, τότε την μετατρέπουμε σε ακτίνια πολλαπλασιάζοντας με το π/180.

\

3.1.4. Σταθερές και μεταβλητές

Το MATLAB μας επιτρέπει να δίνουμε στις σταθερές και μεταβλητές ονόματα της επιλογής μας. Για παράδειγμα, έστω ότι θέλουμε να υπολογίσουμε το ακόλουθο:

sin(60*pi/180)^2 + cos(60*pi/180)^2

Με χρήση σταθερών και μεταβλητών, ο υπολογισμός μπορεί να γίνει ως εξής:

```
theta = 60*pi/180;
a = sin(theta);
b = cos(theta);
a^2 + b^2
```

ans =

1

Τα σύμβολα theta, a και b αντιπροσωπεύουν σταθερές ή μεταβλητές ανάλογα με το άν επιτρέπεται να αλλάζουν στη συνέχεια ή όχι. Το σύμβολο ans είναι μεταβλητή και μπορεί επίσης να χρησιμοποιηθεί σε περαιτέρω υπολογισμούς όπως στο παράδειγμα:

```
60*pi/180
```

ans =

1.0472

```
sin(ans)
```

ans =

0.8660

3.1.5 ΣΕΙΡΕΣ – ΔΙΑΝΥΣΜΑΤΑ

Στο MATLAB η σειρά ορίζεται ως μία διατεταγμένη συλλογή αριθμών που περικλείεται από αγκύλες [...] με τα στοιχεία να διαχωρίζονται είτε από κενά είτε από κόμματα.

```
odd = [1 3 5 7 9 11 13 15 17 19]
```

odd =

1	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**

```
even = [2,4,6,8,10,12,14,16,18,20]
```

even =

2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**

```
dekadikoi = [1.0 1.2 1.4 1.6 1.8 2.0]
```

dekadikoi =

1.0000	**1.2000**	**1.4000**	**1.6000**	**1.8000**	**2.0000**

Το πλήθος των στοιχείων μιας σειράς υπολογίζεται από την συνάρτηση length του MATLAB:

```
length(even)
```

ans =

10

3.1.6 Πράξεις με σειρές

Εστω οι σειρές $A = [a_1\ a_2\ \dots\ a_n]$ και $B = [b_1\ b_2\ \dots\ b_n]$.

Η πρόσθεση και η αφαίρεση των δύο σειρών ορίζονται ως εξής:

$$A + B = [a_1 + b_1, a_2 + b_2, \dots, a_n + b_n]$$

$$A - B = [a_1 - b_1, a_2 - b_2, \dots, a_n - b_n]$$

Για παράδειγμα:

```
odd + even
```

ans =

3	7	11	7	9	11	13	15	17	39

```
even – odd
```

ans =

1	1	1	-7	-9	-11	-13	-15	-17	1

Στην περίπτωση που τα στοιχεία της σειράς βρίσκονται σε ίσες αποστάσεις, τότε δεν χρειάζεται η αναλυτική εισαγωγή της σειράς αλλά μόνο το πρώτο στοιχείο, το βήμα και το τελευταίο στοιχείο με διαχωριστικό σύμβολο το ’:’. Για παράδειγμα, οι παραπάνω σειρές odd και even μπορούν να ορισθούν και ως εξής:

```
odd = 1:2:19
```

odd =

1	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**

```
even = 2:2:20
```

even =

2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**

3.1.7 Ορισμός πράξης πολλαπλασιασμού σειρών:

$A \;.\!*\; B = [a_1b_1,\ a_2b_2,\ \ldots,\ a_nb_n]$

όπου το σύμβολο .* σημαίνει πολλαπλασιασμός στοιχείου προς στοιχείο. Για παράδειγμα:

```
odd.*even
```

ans =

2	**12**	**30**	**56**	**90**	**132**	**182**	**240**	**306**	**380**

3.1.8 Ορισμός διαίρεσης (από αριστερά και από δεξιά) σειρών:

$\boldsymbol{A \;./\; B = [a_1/b_1,\ a_2/b_2,\ \ldots,\ a_n/b_n]}$

A .\ B = [a_1\\$b_1$, a_2\\$b_2$, …, a_n\\$b_n$] ⇔ B

./

odd./even

ans =

Columns 1 through 7

0.5000	**0.7500**	**0.8333**	**0.8750**	**0.9000**	**0.9167**	**0.9286**

Columns 8 through 10

0.9375	**0.9444**	**0.9500**

3.1.9 Ορισμός ύψωσης σε δύναμη:

$A .\hat{}\ m = [a_1^m, a_2^m, \ldots, a_n^m]$

natural.^2

ans =

1	**4**	**9**	**16**	**25**	**36**	**49**	**64**	**81**	**100**

3.1.10 Διανύσματα γραμμής – στήλης

Οι σειρές της προηγούμενης ενότητας μπορούν να θεωρηθούν και ως διανύσματα γραμμής (οριζόντια) με στοιχεία τα αντίστοιχα στοιχεία της σειράς. Αν και η δήλωση διανυσμάτων γραμμής μπορεί να είναι η ίδια με τη δήλωση των σειρών, είναι καλό να περιλαμβάνουμε τα στοιχεία του διανύσματος μέσα σε αγκύλες **[ ]** όπως:

odd = [1:2:19]

odd =

1	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**

even = [2,4,6,8,10,12,14,16,18,20]

even =

2	**7**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**

Η δήλωση ενός διανύσματος στήλης είναι ίδια ως προς τη μορφή με αυτήν ενός διανύσματος γραμμής εκτός από το διαχωριστικό σύμβολο που τώρα είναι είτε το ; είτε η αλλαγή γραμμής:

A=[1;2;3;4;5]

A =

1
2
3
4
5

Η μετατροπή ενός διανύσματος στήλης σε γραμμής και το αντίστροφο μπορεί να γίνει με την χρήση του αναστρόφου διανύσματος που συμβολίζεται με την απόστροφο:

```
At = A'
```

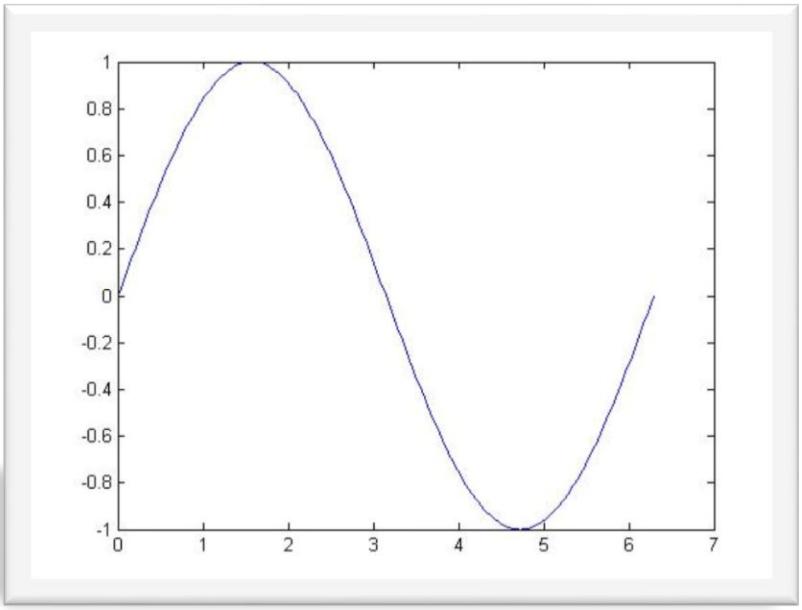
At =

1	2	3	4	5

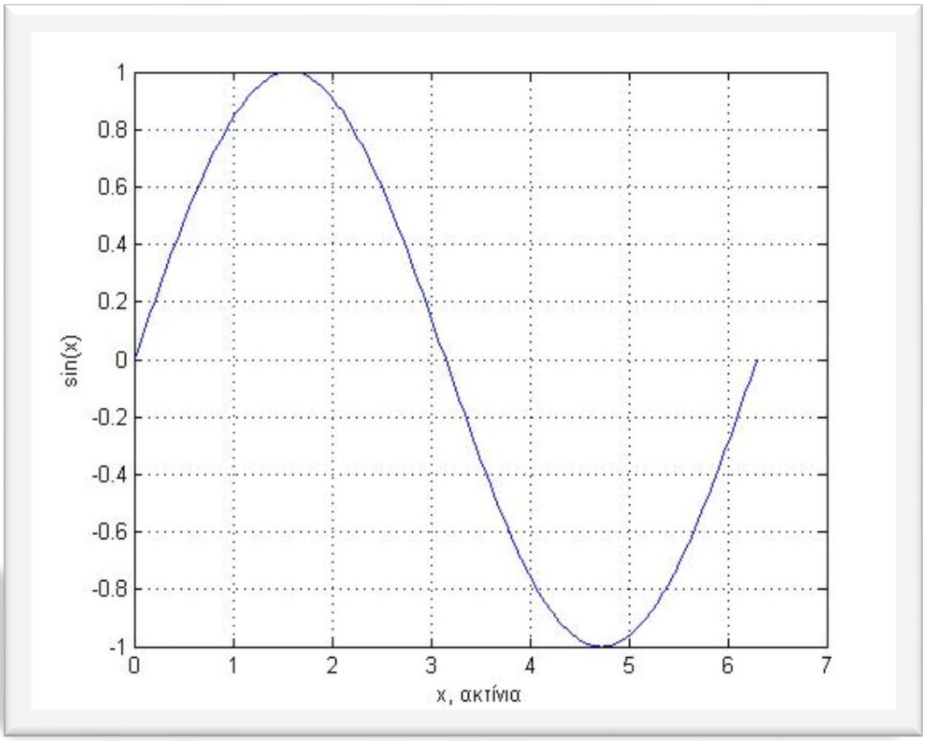
3.1.11 Γραφική παράσταση

Εστω ότι θέλουμε να κάνουμε τη γραφική παράσταση της ημιτονοειδούς συνάρτησης στο διάστημα [0, 2π]. Η βασική συνάρτηση του MATLAB για δισδιάστατες απεικονίσεις είναι η **plot** (για λεπτομέρειες πληκτρολογήστε **help plot**). Αλλες χρήσιμες συναρτήσεις είναι η **grid** που σχεδιάζει τον κάνναβο και οι **xlabel**, **ylabel** για την εισαγωγή κειμένου στις γραφικές παραστάσεις.

```
x = 0: pi/90: 2*pi;
y = sin(x);
plot(x,y)
```




```
Grid
xlabel('x, ακτίνια')
ylabel('sin(x)')
```



3.1.12 ΠΙΝΑΚΕΣ

Οι πίνακες στο MATLAB περικλείονται σε αγκύλες [ ] και εισάγονται με απλό τρόπο. Με χρήση των διαχωριστικών ***κενό ή κόμμα*** για τα στοιχεία γραμμής και του ’;’ για την αλλαγή γραμμής μπορούμε να ορίσουμε έναν πίνακα ως εξής:

```
A = [1 2 3; 4 5 6; 7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

Μπορούμε επίσης να γράψουμε κάθε γραμμή ξεχωριστά όπως:

```
A= [1 2 3
     4 5 6
     7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

Η πρόσβαση στα στοιχεία του πίνακα γίνεται με χρήση δύο δεικτών μέσα σε παρένθεση με τον πρώτο να προσδιορίζει τη γραμμή και τον δεύτερο τη στήλη:

A(1,3)

ans =

3

A(3,2)

ans =

8

Οι διαστάσεις ενός πίνακα δίνονται με τη συνάρτηση **size**:

size(A)

ans =

3	**3**

Η δημιουργία ενός νέου πίνακα από δύο πίνακες A και B που έχουν ίδιο πλήθος στηλών είναι επίσης δυνατή με χρήση της λειτουργίας [A; B] όπως στο παράδειγμα:

```
A = [1 2 3; 4 5 6; 7 8 9];
B = [10 11 12; 13 14 15];
C = [A; B]
```

C =

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

Για να εξαγάγουμε έναν υποπίνακα από την γραμμή x1 έως τη γραμμή x2 και από τη στήλη y1 έως τη στήλη y2 μέσα από κάποιον πίνακα A, χρησιμοποιούμε την μορφή **A(x1:x2; y1:y2)**. Για παράδειγμα, αν θέλουμε να εξαγάγουμε τον υποπίνακα που αποτελείται από τις δύο πρώτες γραμμές και στήλες του C θα έχουμε:

```
C(1:2,1:2)
```

ans =

1	2
4	5

Αν θέλουμε να εξαγάγουμε όλες τις γραμμές ή όλες τις στήλες, τότε δεν χρειάζεται να το δηλώσουμε αναλυτικά αλλά χρησιμοποιούμε μόνο το σύμβολο ’:’

```
C(:,1:2)
```

ans =

1	**2**
4	**5**
7	**8**
10	**11**
13	**14**

```
C(1,:)
```

ans =

1	**2**	**3**

3.1.13 Εκτύπωση πίνακα

Εστω ότι θέλουμε να δημιουργήσουμε έναν πίνακα με τα ημίτονα των γωνιών 0, 10, 20, ..., 90 μοιρών. Αυτό μπορεί να γίνει με χρήση των λειτουργιών της αναστροφής και της παράθεσης:

```
angle = 0:10:180;
sine = sin(pi*angle/180);
[angle' sine']
```

ans =

0	**0**
10.0000	**0.1736**
20.0000	**0.3420**
30.0000	**0.5000**
40.0000	**0.6428**
50.0000	**0.7660**
60.0000	**0.8660**
70.0000	**0.9397**
80.0000	**0.9848**
90.0000	**1.0000**
100.0000	**0.9848**
110.0000	**0.9397**
120.0000	**0.8660**
130.0000	**0.7660**
140.0000	**0.6428**
150.0000	**0.5000**
160.0000	**0.3420**
170.0000	**0.1736**
180.0000	**0.0000**

3.1.14 Δημιουργία αρχείων Μ

Οι ακολουθίες εντολών του MATLAB μπορούν να γραφούν σε αρχεία των οποίων οι ονομασίες θα έχουν κατάληξη m, και θα ονομάζονται κατ’ αναλογία **αρχεία-Μ**.

Η εκτέλεση του αρχείου-Μ μπορεί να γίνει είτε με τη χρήση του εικονιδίου “save and run” που βρίσκεται πάνω στο toolbar του editor του matlab, είτε πληκτρολογώντας το όνομα ενός τέτοιου αρχείου, χωρίς το m, προκαλούμε την εκτέλεση όλων των εντολών.

Για **<u>παράδειγμα</u>** ένα αρχείο-Μ δημιουργείται από το μενού **File → New → M file**

Πληκτρολογώντας τα παρακάτω θα δημιουργήσουμε ένα πρόγραμμα που θα κάνει την γραφική παράσταση μιας εξίσωσης πρώτου βαθμού:

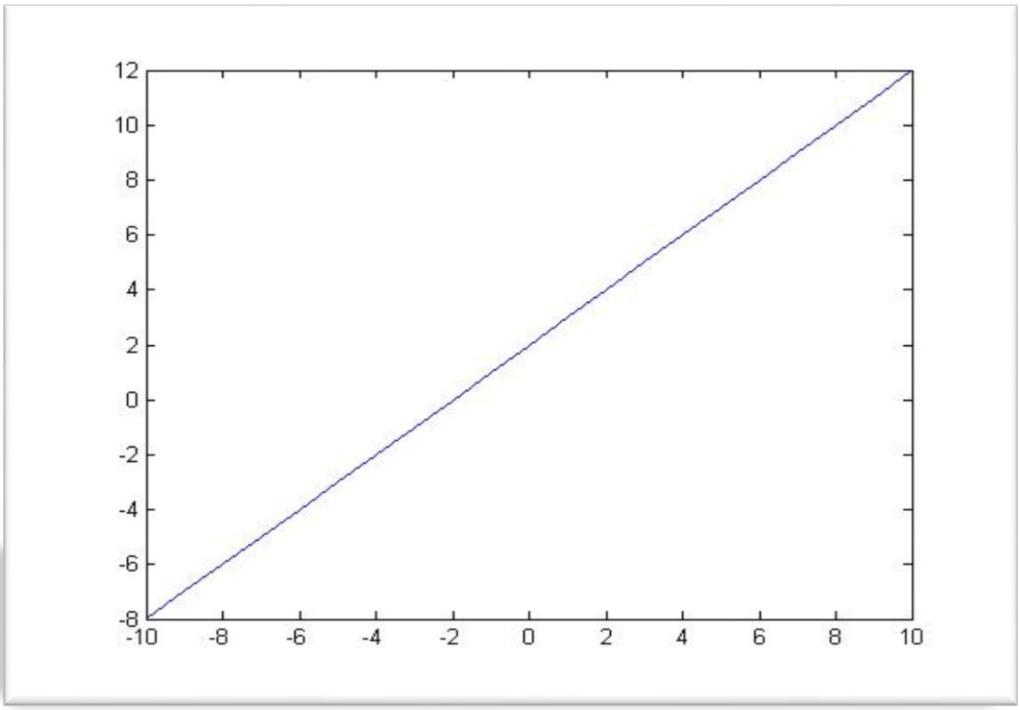
```
%example1
%ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΣΥΝΑΡΤΗΣΗΣ ΠΡΩΤΟΥ ΒΑΘΜΟΥ
%ΔΙΝΟΥΜΕ ΤΙΜΕΣ ΣΤΟΥΣ ΠΑΡΑΓΟΝΤΕΣ ΤΗΣ ΕΞΙΣΩΣΗΣ
a=1;
b=2;
%ΔΙΝΟΥΜΕ ΤΙΣ ΤΙΜΕΣ ΤΟΥ Χ
x=-10:1:10;
%ΓΡΑΦΟΥΜΕ ΤΗΝ ΠΡΩΤΟΒΑΘΜΙΑ ΕΞΙΣΩΣΗ
y=a*x+b;
%ΣΧΕΔΙΑΖΟΥΜΕ ΤΗ ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ
grid;
plot(x,y);
```

Αποθηκεύεστε το αρχείο με την εντολή **File→ Save as** στον κατάλογο **work** με το όνομα **example1**.

Για να τρέξετε το αρχείο που μόλις δημιουργήσατε αρκεί να πληκτρολογήσετε

```
example1
```

Στο command window του Matlab.



Στο MATLAB μπορούμε να προγραμματίσουμε τις συναρτήσεις που εμείς θέλουμε βάζοντας σαν πρώτη λέξη του προγράμματος το function. Αυτά τα αρχεία ονομάζονται **αρχεία συναρτήσεων** και λαμβάνουν εξωτερικά ορίσματα τα οποία περιέχονται σε παρενθέσεις αμέσως μετά το όνομα της συνάρτησης.

Για **παράδειγμα** το προηγούμενο πρόγραμμα που έκανε τη γραφική παράσταση μιας εξίσωσης πρώτου βαθμού μπορεί να τροποποιηθεί έτσι ώστε να δίνει το αποτέλεσμα της εξίσωσης αν του δώσουμε τιμές για τα a, b και x

```
function
y=firstorder(a,b,x)
%firstorder(A,B,X)
%ΣΥΝΑΡΤΗΣΗ ΠΡΩΤΟΥ ΒΑΘΜΟΥ Y=AX+B
%ΔΩΣΤΕ ΤΟ Α ΤΟ Β ΚΑΙ ΤΟ Χ ΚΑΙ ΘΑ ΠΑΡΕΤΕ ΤΟ ΑΠΟΤΕΛΕΣΜΑ
%ΓΡΑΦΟΥΜΕ ΤΗΝ ΠΡΩΤΟΒΑΘΜΙΑ ΕΞΙΣΩΣΗ
```

```
y=a*x+b;
End
```

3.1.15 ΜΙΓΑΔΙΚΟΙ ΑΡΙΘΜΟΙ

Στα **παραδείγματα** που ακολουθούν βλέπουμε την ευκολία με την οποία χειρίζεται το MATLAB τους μιγαδικούς. Παρατηρήστε ότι εκτός από τις απλές πράξεις, χρησιμοποιούμε στις εκφράσεις και κάποιες συναρτήσεις όπως οι **sqrt** (τετραγωνική ρίζα) και **sin** (ημίτονο). Η συνάρτηση **abs** υπολογίζει το μέτρο ενός μιγαδικού αριθμού ή απλά την απόλυτη τιμή ενός πραγματικού. Η **angle** υπολογίζει το όρισμα (γωνία) του μιγαδικού σε ακτίνια, ενώ τέλος οι **real** και **imag** επιστρέφουν το πραγματικό και φανταστικό μέρος αντίστοιχα ενός μιγαδικού.

Παραδείγματα:

```
z=1-2i % i is the imaginary unit
```

```
z =
1.0000 - 2.0000i
```

```
w=1-2j % j is also the imaginary unit
```

```
w =
1.0000- 2.0000i
```

```
z1=3*(2-sqrt(-1)*3)
```

z1 **6.0000 - 9.0000i** **=**

```
z2=sqrt(-2)
```

0 + 1.4142i **z2 =**

3.1.16 ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ

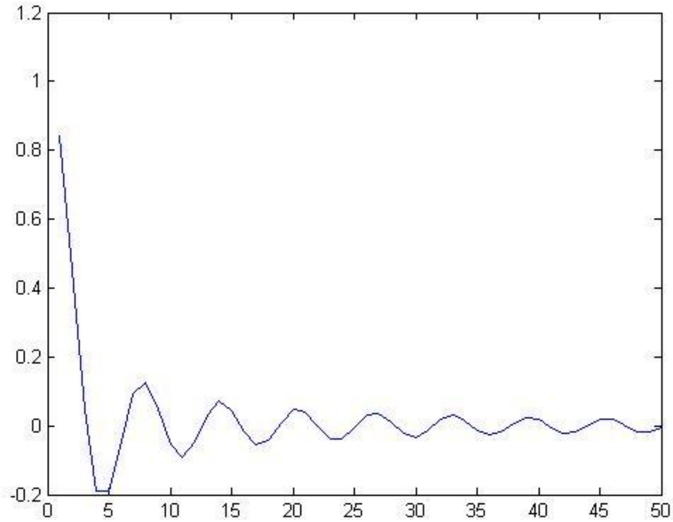
<u>(α) Στον χώρο δύο διαστάσεων</u>

Η πλέον συνηθισμένη συνάρτηση που παράγει γραφικές παραστάσεις δύο διαστάσεων και ήδη έχουμε χρησιμοποιήσει είναι η **plot**. Θα γνωρίσουμε τις διάφορες μορφές της μέσα από παραδείγματα.

Στην πιο απλή περίπτωση η **plot(x)** κάνει γραφική παράσταση των στοιχείων του <u>διανύσματος</u> x ως προς τον αύξοντα αριθμό τους.

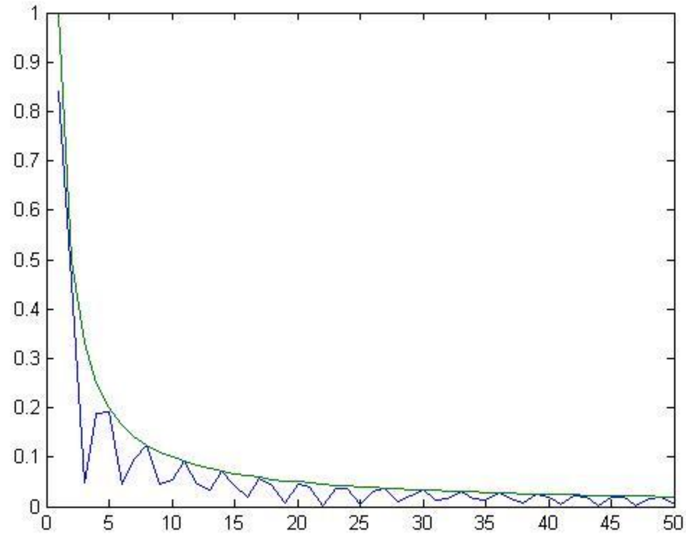
```
n=1:50;
an=sin(n)./n;
plot(an)
```

Παρατηρούμε ότι η **plot** ανοίγει ένα παράθυρο γραφικών (***Figure window***), σχεδιάζει άξονες και τοποθετεί τα σημεία ενώνοντας τα με ευθείες γραμμές.



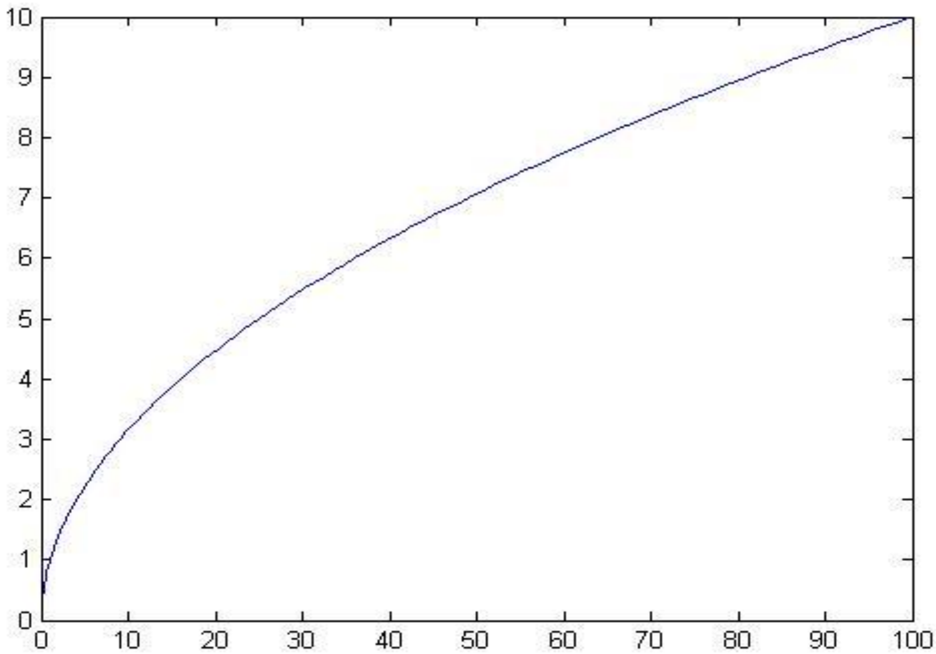
Αν το x είναι πίνακας, τότε στο ίδιο σχήμα γίνεται μία γραφική παράσταση για κάθε στήλη του πίνακα :

```
for n=1:50
a(n,1)=abs(sin(n))/n;
a(n,2)=1/n;
end
plot(a)
```



Στην περίπτωση που το x είναι διάνυσμα μιγαδικών αριθμών, τότε γίνεται γραφική παράσταση των σημείων που παριστούν τους μιγαδικούς με άξονες που αντιστοιχούν στο πραγματικό και στο φανταστικό μέρος

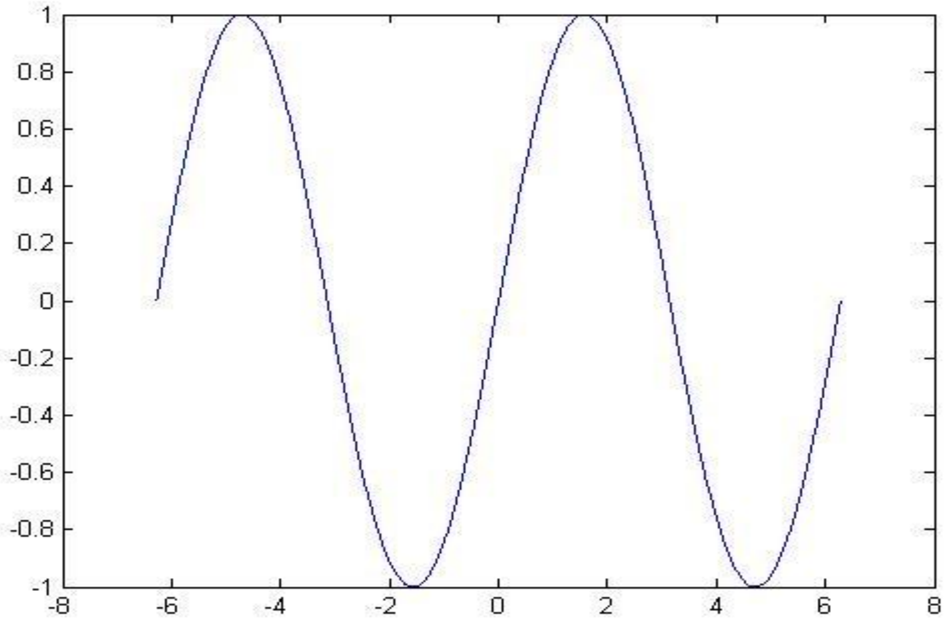
```
x=0:0.1:10;
z=x.^2+x*I;
plot(z)
```



Παρατηρούμε επίσης ότι αν το παράθυρο Figure είναι ήδη ανοιχτό, η **plot** το καθαρίζει και σχεδιάζει νέα γραφική παράσταση.

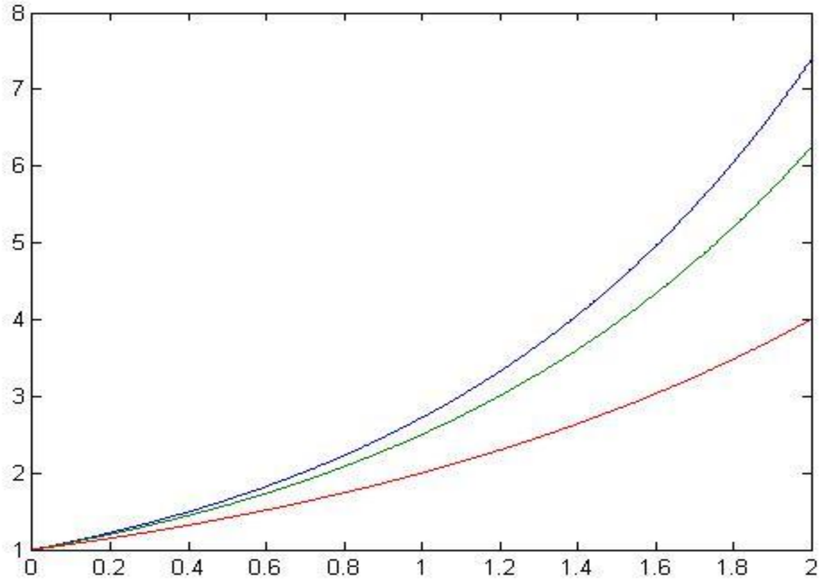
Αν χρησιμοποιήσουμε την μορφή **plot(x,y)**, θα πάρουμε γραφική παράσταση των στοιχείων του y ως προς τα αντίστοιχα του x :

```
x=linspace(-2*pi,2*pi,1000);
y=sin(x);
plot(x,y)
```



Υπάρχει η δυνατότητα να σχεδιαστούν μαζί πολλές γραφικές παραστάσεις με διαφορετικά χρώματα, αν ορίσουμε στην plot πολλά ζεύγη διανυσμάτων

```
x=linspace(0,2,1000);
y1=exp(x);
y2=2.5.^x;
y3=2.^x;
plot(x,y1,x,y2,x,y3)
```

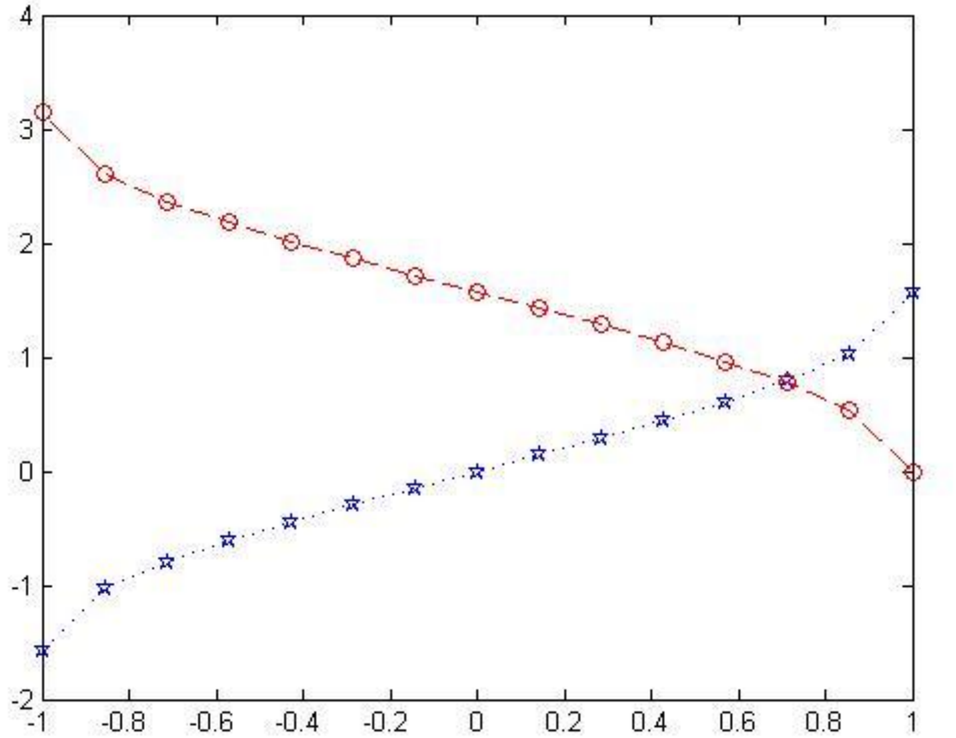


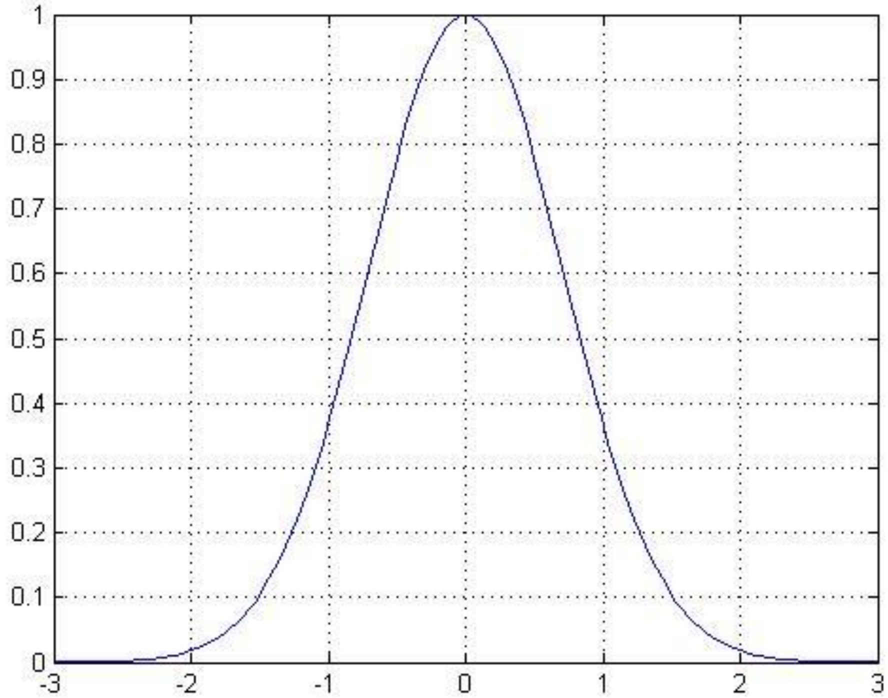
Οι παρακάτω εντολές θα δημιουργήσουν δύο καμπύλες σε μία γραφική παράσταση. Η πρώτη θα έχει μπλε χρώμα, τα σημεία θα είναι πεντάγωνα και η γραμμή διάστικτη, ενώ η δεύτερη θα σχεδιαστεί με κόκκινο χρώμα, τα σημεία θα είναι κύκλοι και η γραμμή διακεκομμένη .

```
x=linspace(-1,1,15);
y1=asin(x);
y2=acos(x);
plot(x,y1,'b:p',x,y2,'r--o')
```

Η εντολή **grid on** προσθέτει ένα πλέγμα γραμμών για καλύτερη ανάγνωση των συντεταγμένων, ενώ η **grid off** αφαιρεί το πλέγμα. Η **grid** απλά λειτουργεί σαν διακόπτης ανάμεσα στις δύο καταστάσεις .

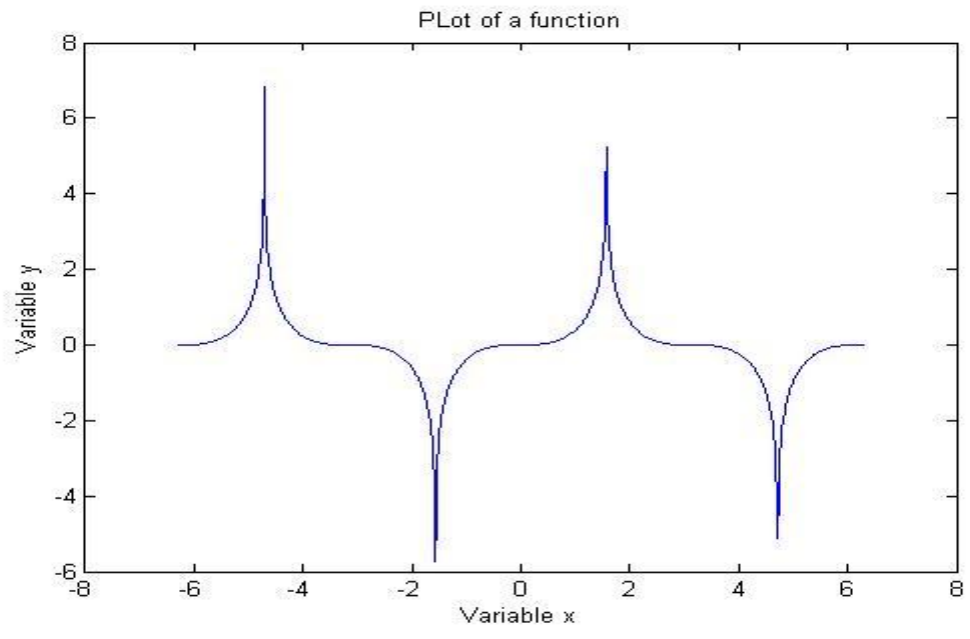
```
x=linspace(-3,3,150);
y=exp(-x.^2);
plot(x,y)
grid on
```


4
3
2
1
0
-1
-2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-3
-2
-1
0
1
2
3

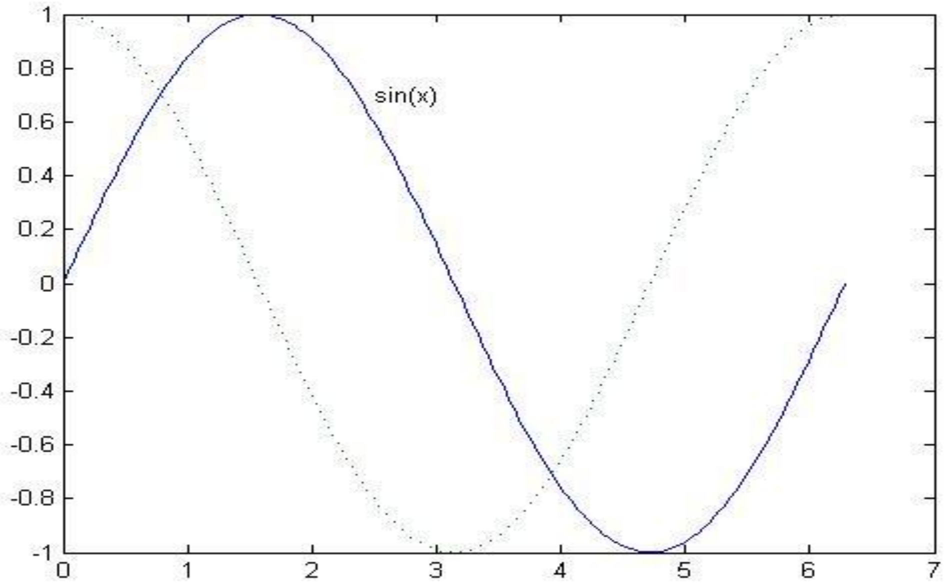
Μπορούμε να προσθέσουμε ετικέτες (*labels*) στον οριζόντιο και κατακόρυφο άξονα μίας γραφικής παράστασης με τις εντολές **xlabel** και **ylabel**. Μπορούμε ακόμη να βάλουμε τίτλο σε όλη την γραφική παράσταση με την εντολή **title**

```
x=-2*pi:.01:2*pi;
y=atanh(sin(x))-sin(x);
plot(x,y)
title('PLot of a function')
xlabel('Variable x')
ylabel('Variable y')
```



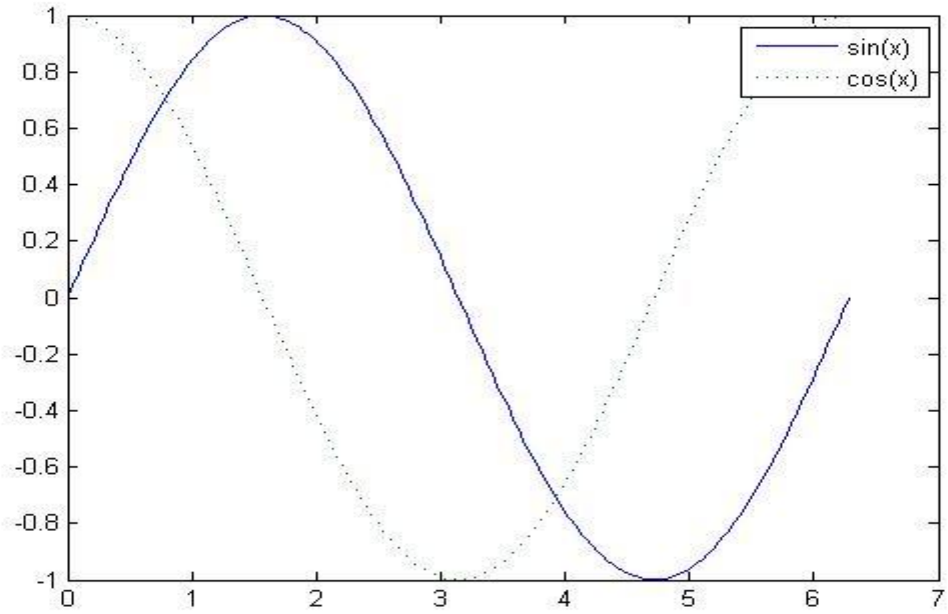
Υπάρχει τρόπος να προσθέσουμε κείμενο μέσα στο γράφημα με την εντολή **text**. Η γενική μορφή είναι **text(x,y,'κείμενο')** όπου (x, y) είναι οι συντεταγμένες του σημείου όπου θα γραφεί το κείμενο. Με τις παρακάτω εντολές θα δημιουργήσουμε μία γραφική παράσταση με τις καμπύλες sin(x) και cos(x) και θα προσθέσουμε ένα κείμενο στη θέση (2.5,0.7).

```
x=linspace(0,2*pi,150);
y=sin(x);
z=cos(x);
plot(x,y,x,z,':')
text(2.5,.7,'sin(x)')
```



Αντί να παρεμβάλουμε κείμενο μέσα στο γράφημα για να ξεχωρίζουμε τις γραμμές, μπορούμε να εισάγουμε μία συνολική επιγραφή με την εντολή **legend**. Αυτόματα εισάγεται ένα ορθογώνιο το οποίο περιέχει τις επιγραφές που θα ορίσουμε για κάθε καμπύλη του γραφήματος. Το ορθογώνιο αυτό μπορούμε να το μετακινήσουμε με το mouse (click and drag) σε όποιο σημείο του γραφήματος επιθυμούμε.

```
legend('sin(x)','cos(x)')
```



Η επιγραφή διαγράφεται με την εντολή **legend off**

Αν θέλουμε να έχουμε γραφήματα σε πολλά παράθυρα, μπορούμε να ανοίξουμε νέα παράθυρα γραφικών από το **File** του κυρίως menu επιλέγοντας **New Figure**. Ιδιαίτερα χρήσιμη είναι και η δυνατότητα να σχεδιάσουμε πολλά διαφορετικά γραφήματα με διαφορετικούς άξονες στο ίδιο παράθυρο. Η εντολή **subplot(m,n,p)** διαιρεί το ενεργό παράθυρο γραφικών σε $m \times n$ θέσεις ενώ ταυτόχρονα καθιστά ενεργή την p-θέση.

Η διαίρεση του παράθυρου αναιρείται με την εντολή **subplot(1,1,1)**.

```
x=linspace(-2*pi,2*pi,1000);
y1=sin(x);
y2=x;
z=y1+y2;
w=y1.*y2;
subplot(2,2,1)
plot(x,y1)
axis([-2*pi,2*pi,-1,1])
title('sin(x)')
```

```
subplot(2,2,2)
plot(x,y2)
axis([-2*pi,2*pi,-2*pi,2*pi])
title('x')
```

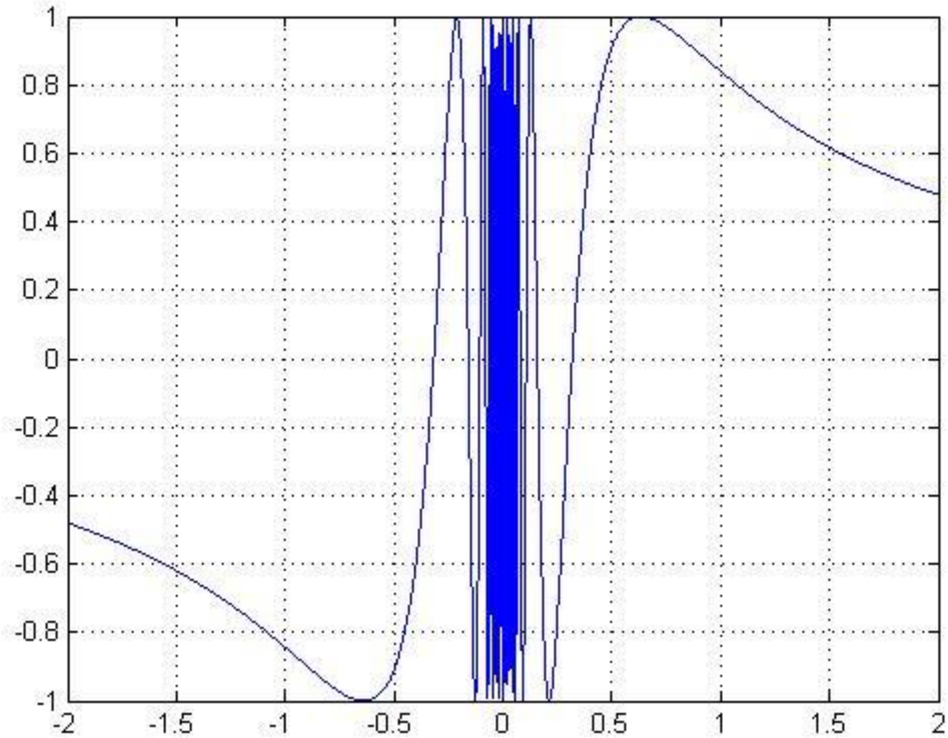
```
subplot(2,2,3)
plot(x,z)
axis([-2*pi,2*pi,-2*pi,2*pi])
title('sin(x)+x')
```

```
subplot(2,2,4)
 plot(x,w)
axis([-2*pi,2*pi,-2*pi,2*pi])
title('x*sin(x)')
```

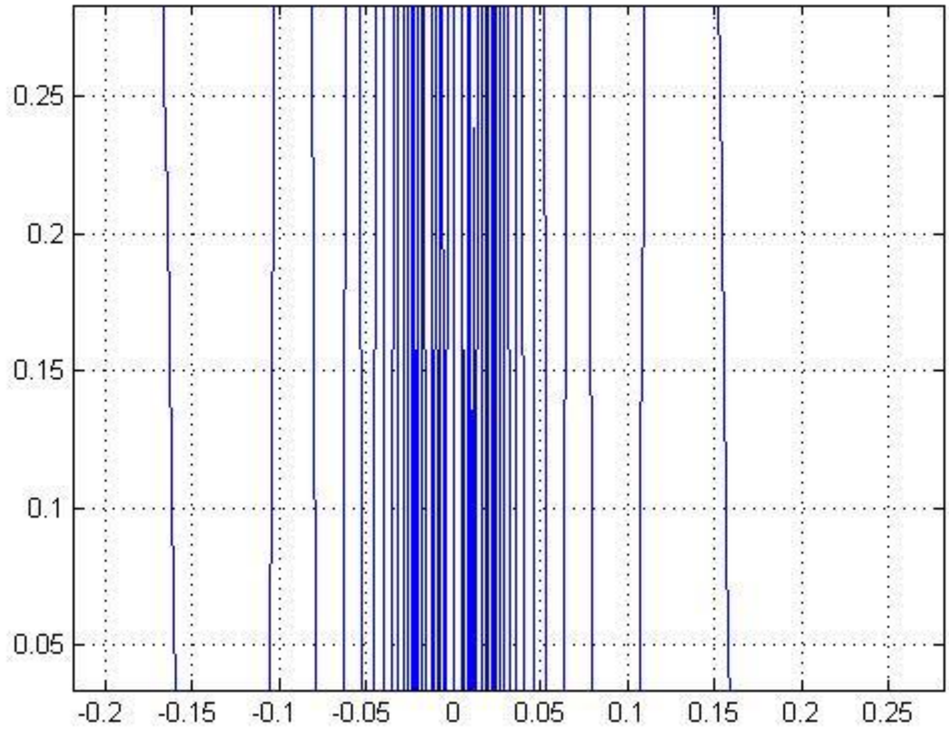
Αν θελήσουμε να εξετάσουμε μία περιοχή του γραφήματος καλύτερα, μπορούμε να την επεκτείνουμε με την εντολή **zoom on**. Δίνοντας την εντολή, ενεργοποιείται η εστίαση. Στο παράθυρο γραφικών, με click στο αριστερό πλήκτρο του mouse επεκτείνουμε την περιοχή ενώ με click στο δεξιό πλήκτρο απομακρυνόμαστε. Μπορούμε ακόμη να οριοθετήσουμε μία περιοχή που θα επεκτείνουμε με click and drag.

Η εντολή **zoom off** αναστέλλει την εστίαση. Στο παρακάτω σχήμα βλέπουμε (α) τη γραφική παράσταση της συνάρτησης που δημιουργείται από τις παρακάτω εντολές και (β) επέκταση μίας περιοχής του γραφήματος.

```
x=linspace(-2,2,3000);
y=sin(1./x);
plot(x,y)
grid on
```




```
zoom on
```



Μερικές ακόμη εντολές γραφικών παραστάσεων που αντικαθιστούν την **plot** σε ειδικές περιπτώσεις είναι:

loglog: Οι κλίμακες των αξόνων είναι λογαριθμικές.

semilogx: Η κλίμακα του x-άξονα είναι λογαριθμική.

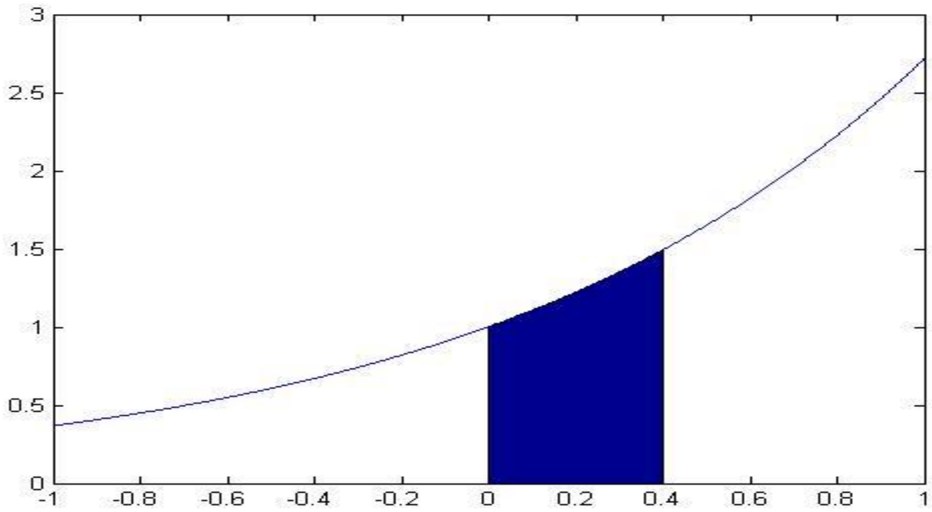
semilogy: Η κλίμακα του y-άξονα είναι λογαριθμική.

area(x,y): Η περιοχή ανάμεσα στην καμπύλη και στον x-άξονα σκιαγραφείται.

Παράδειγμα: Με τις παρακάτω εντολές σκιαγραφούμε μία περιοχή κάτω από την καμπύλη $y = \exp(x)$ Προσέξτε την εντολή **hold on** η οποία διατηρεί το προηγούμενο γράφημα στο παράθυρο έτσι ώστε το νέο να σχεδιαστεί επάνω του. Για να διακοπεί η διατήρηση, δίνουμε **hold off**.

```
x=-1:0.001:1;
y=exp(x);
plot(x,y)
hold on
```

```
x1=0:0.01:0.4;
y1=exp(x1);
area(x1,y1)
```



Γραφικές παραστάσεις σε πολικές συντεταγμένες δημιουργούνται με την εντολή **polar(t,r)** όπου t είναι διάνυσμα με γωνίες σε rad και r διάνυσμα με αποστάσεις. Η μορφή της γραφικής παράστασης (χρώματα, σημεία και γραμμές) μπορεί να ρυθμιστεί με ένα σύνολο από ειδικούς χαρακτήρες με τον ίδιο ακριβώς τρόπο που είδαμε στην εντολή plot.

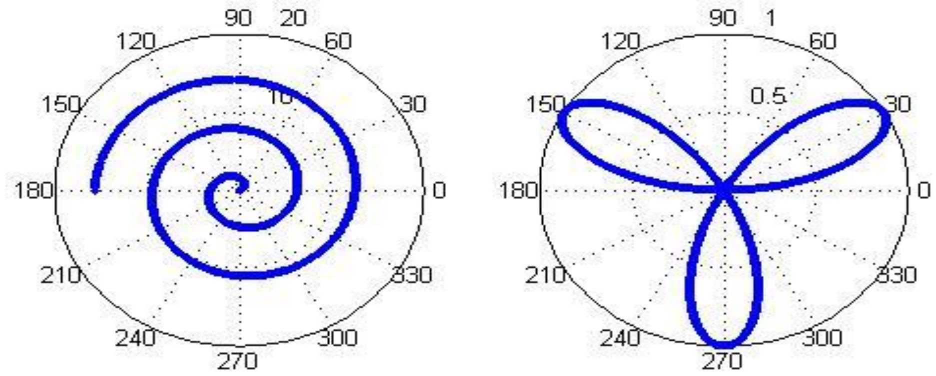
Παράδειγμα: Θα σχηματίσουμε τις γραφικές παραστάσεις των συναρτήσεων που δίνονται σε πολικές συντεταγμένες

$r=\theta$ **(σπείρα του Αρχιμήδη)** **και** $r=\sin(3\theta)$ **(τρίφυλλο)**

Οι εντολές που θα δώσουμε για να πάρουμε τις δύο γραφικές παραστάσεις στο ίδιο παράθυρο είναι:

```
heta=linspace(0,5*pi,1000);
r1=theta;
r2=sin(3*theta);
```

```
subplot(1,2,1)
polar(theta,r1,'.')
subplot(1,2,2)
polar(theta,r2,'.')
```



(β) Τρισδιάστατες γραφικές παραστάσεις

Για τη δημιουργία καμπυλών στον τρισδιάστατο χώρο (*line plots*), χρησιμοποιούμε την εντολή **plot3**. Η σύνταξη και τα χαρακτηριστικά της είναι ίδια με αυτά της **plot** με τη διαφορά ότι εδώ ορίζουμε τριάδες διανυσμάτων αντί για ζεύγη. Η γενική μορφή είναι δηλαδή:

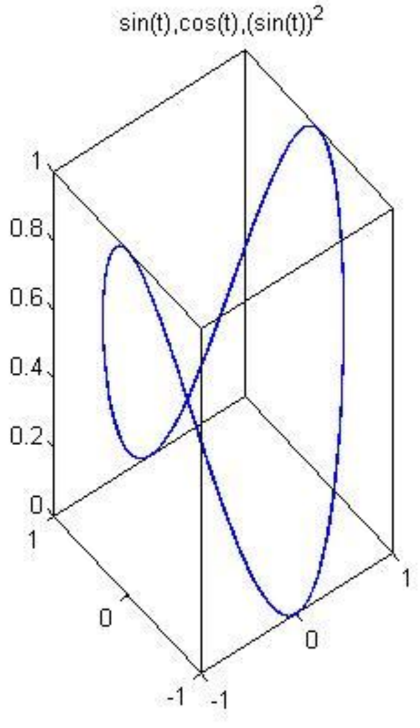
plot3($x_1, y_1, z_1, '\mathit{string}\_1', \ldots, x_n, y_n, z_n, '\mathit{string}\_n')$

όπου string είναι η σειρά ειδικών χαρακτήρων για τη ρύθμιση του χρώματος, της γραμμής και των σημείων. Η αρχή αυτή επεκτείνεται και σε όλες τις εντολές που τη συνοδεύουν. Για παράδειγμα, η εντολή **text** δέχεται 3 συντεταγμένες, ενώ για ετικέτα στον z-άξονα χρησιμοποιούμε την εντολή **zlabel**. Οι εντολές **grid**, **box**, **subplot** κ.α. έχουν την ίδια ακριβώς λειτουργία.

Παράδειγμα: Θα σχεδιάσουμε τις καμπύλες που ορίζονται από τις διανυσματικές συναρτήσεις

$\vec{r}_1(t) = (\sin t, \cos t, \sin^2 t)$ **και** $\vec{r}_2(t) = (\sin t, \cos t, t)$.

```
t=linspace(0,10*pi,1000);
subplot(1,2,1)
 plot3(sin(t),cos(t),(sin(t)).^2)
title('sin(t)
cos(t)
(sin(t))^2')
 box on
```



Μία άλλη μορφή γραφικών παραστάσεων στον τρισδιάστατο χώρο είναι και οι επιφάνειες που παριστούν συναρτήσεις δύο μεταβλητών $f(x, y)$. Η διαδικασία παραγωγής τους είναι η ακόλουθη:

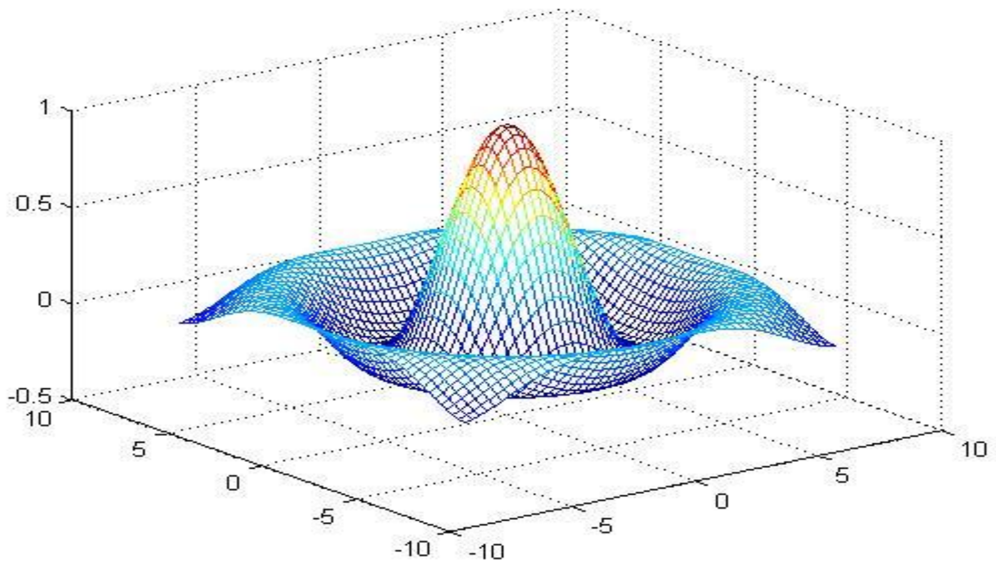
1. Ορίζουμε δύο διανύσματα *x* και *y* τα οποία έχουν σημεία στο εύρος των τιμών

των αντίστοιχων μεταβλητών για τις οποίες θα γίνει η γραφική παράσταση,

2. Με την εντολή **meshgrid** μετατρέπουμε τα δύο προηγούμενα διανύσματα σε πίνακες X και Y όπου οι γραμμές του X είναι επαναλήψεις του διανύσματος x και οι στήλες του Y επαναλήψεις του διανύσματος y.
3. Υπολογίζουμε τις τιμές της συνάρτησης f για τα αντίστοιχα στοιχεία και των δύο πινάκων X και Y.
4. Η γραφική παράσταση γίνεται με τη βοήθεια της εντολής **mesh**.

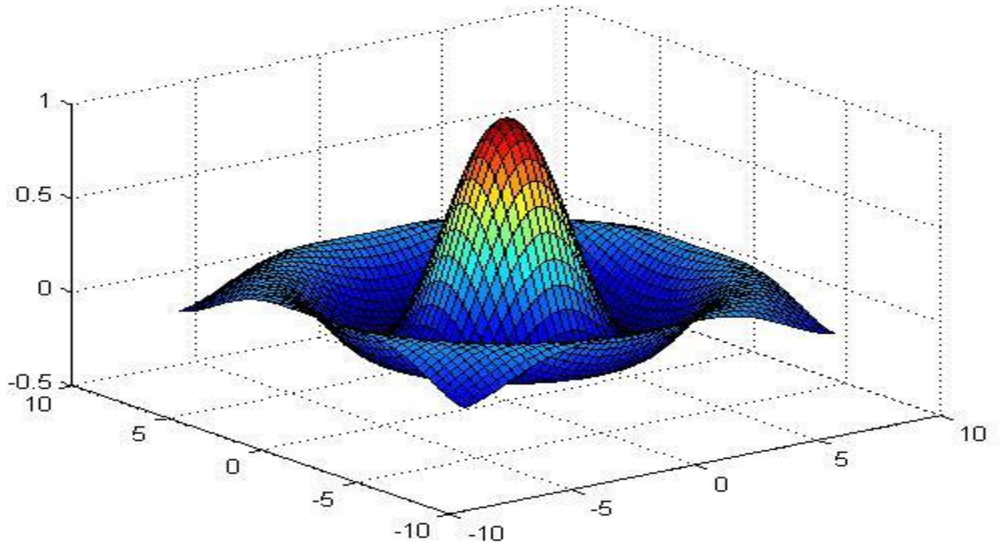
Οι παρακάτω εντολές είναι παράδειγμα σχεδιασμού μίας επιφάνειας με τη μορφή πλέγματος :

```
x=linspace(-7.5,7.5,50);
y=x;
[X,Y]=meshgrid(x,y);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(X,Y,Z)
```



Η εντολή **surf** είναι ακριβώς ίδια με την mesh με την μόνη διαφορά ότι η surf γεμίζει με χρώματα την επιφάνεια. Δοκιμάστε την εντολή

```
surf(X,Y,Z)
```



3.1.17 Συμβολικές εκφράσεις

Το MATLAB διαθέτει μία συλλογή από συναρτήσεις κάτω από το όνομα ***Symbolic Math Toolbox*** που χρησιμεύουν σαν εργαλεία για την εκτέλεση συμβολικών πράξεων όπως η επίλυση αλγεβρικών και διαφορικών εξισώσεων, η παραγώγιση και η ολοκλήρωση συναρτήσεων, η εύρεση οριζουσών και χαρακτηριστικών ριζών κ.α..

Σε κάθε συμβολική έκφραση είναι σημαντικό να δηλώνουμε αρχικά τις μεταβλητές που παίρνουν μέρος στις εκφράσεις σαν **συμβολικές μεταβλητές** (*symbolic variables*). Αυτό γίνεται με τις συναρτήσεις **sym** (για μία μόνο μεταβλητή) και **syms** (για πολλές μαζί μεταβλητές). Για παράδειγμα, με τις εντολές που ακολουθούν, δηλώνουμε μία μεταβλητή x σαν συμβολική και στη συνέχεια την χρησιμοποιούμε για να υπολογίσουμε την παράγωγο της συνάρτησης cosx

.

x=sym('x')

x =

x

diff(cos(x))

ans =

-sin(x)

Με τις παρακάτω εντολές, δηλώνουμε 4 συμβολικές μεταβλητές μαζί, κατασκευάζουμε ένα συμβολικό πίνακα με αυτές και κατόπιν υπολογίζουμε την ορίζουσά του.

➔ **syms('a','b','c','d')**

➔ **M=[a,b;c,d]**

M =

[a, b]
[c, d]

det(M)

ans =

a*d-b*c

(α) Αριθμητές και παρονομαστές ρητών παραστάσεων

Όταν έχουμε μια σύνθετη έκφραση με κλάσματα που θέλουμε να την μετατρέψουμε σε ρητή παράσταση, χρησιμοποιούμε την εντολή **numden**. Αυτή μας επιστρέφει ξεχωριστά τον αριθμητή και τον παρονομαστή της ρητής παράστασης. Για παράδειγμα, αν θέλουμε να μετατρέψουμε την παράσταση

$$\frac{1}{x+1}-\frac{2}{x-2}+\frac{2}{x+5}$$

σε ρητή, δίνουμε τις εντολές:

```
syms x
g=1/(x+1)-2/(x-2)+2/(x+5)
```

```
g =
1/(x+1)-2/(x-2)+2/(x+5)
```

```
[n d]=numden(g)
```

```
n =
x^2-11*x-24

d =
(x+1)*(x-2)*(x+5)
```

Προκύπτει λοιπόν ότι η ζητούμενη παράσταση είναι

$$\frac{x^2 - 11x - 24}{(x+1) * (x-2) * (x+5)}$$

(β) Συμβολικά αθροίσματα – σειρές

Η εντολή **symsum** χρησιμοποιείται για να υπολογίζει συμβολικά αθροίσματα εκφράσεων. Η πιο βολική μορφή της είναι η **symsum**(*έκφραση, μεταβλητή-δείκτης, κάτω όριο, πάνω όριο*).

Για παράδειγμα, θα υπολογίσουμε τα παρακάτω αθροίσματα:

$$\sum_{k=0}^{n} w^k = \frac{1 - w^{n+1}}{1 - w}$$

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=0}^{\infty} w^k = \frac{1}{1 - w}$$

$$\sum_{k=0}^{\infty} \frac{x^k}{k!} = e^x$$

```
syms x w k n
f1=w^k;f2=k;f3=x^k/sym('k!');
symsum(f1,k,0,n)
```

ans =

```
w^(n+1)/(w-1)-1/(w-1)
```

```
symsum(f2,k,1,n)
```

ans =

1/2*(n+1)^2-1/2*n-1/2

symsum(f1,k,0,inf)

ans = **-1/(w-1)**

symsum(f3,k,0,inf)

ans = **exp(x)**

<u>(γ) Παραγώγιση</u>

Η παραγώγιση συναρτήσεων γίνεται με την εντολή **diff**. Οι διαφορετικές της μορφές είναι:

- **diff**(*έκφραση*): Παραγωγίζει την έκφραση ως προς μία από τις συμβολικές μεταβλητές της. Είναι προτιμότερο να χρησιμοποιείται όταν η έκφραση περιλαμβάνει <u>μία μόνο μεταβλητή</u>.
- **diff**(*έκφραση, n*): Παραγωγίζει την έκφραση n φορές.
- **diff**(*έκφραση, μεταβλητή, n*): Παραγωγίζει την έκφραση n φορές ως προς την μεταβλητή που θα καθορίσουμε.

Για παράδειγμα, θα υπολογίσουμε τις μερικές παραγώγους πρώτης και δεύτερης τάξης της συνάρτησης 2 μεταβλητών

$$\boldsymbol{f(x,y) = e^{-2x} * siny}$$

```
syms x y
f=exp(-2*x)*sin(y)
```

f =

```
exp(-2*x)*sin(y)
```

```
df_dx=diff(f,x)
```

df_dx =

```
-2*exp(-2*x)*sin(y)
```

```
df_dy=diff(f,y)
```

df_dy =

```
exp(-2*x)*cos(y)
```

(δ) Ολοκλήρωση

Η εντολή με την οποία μπορούμε να υπολογίζουμε ολοκληρώματα, είναι η **int**. Έχει και αυτή διάφορες μορφές αλλά οι πιο πλήρεις είναι οι εξής:

int(*έκφραση, μεταβλητή*): Υπολογίζει το αόριστο ολοκλήρωμα ως προς τη συμβολική μεταβλητή που θα ορίσουμε. Υπάρχει περίπτωση είτε το ολοκλήρωμα να είναι αδύνατο να δοθεί σε κλειστή μορφή, είτε η συνάρτηση να είναι τόσο πολύπλοκη που το MATLAB να μη μπορεί να υπολογίσει την αντιπαράγωγο.

int(*έκφραση, μεταβλητή, κάτω όριο, πάνω όριο*): Υπολογίζει το ορισμένο ολοκλήρωμα σε κάποιο διάστημα ολοκλήρωσης.

Για **παράδειγμα**, θα υπολογίσουμε τα ολοκληρώματα:

$$\int \frac{3x+7}{(x^2+2x+5)^2}dx = \frac{x-2}{2(x^2+2x+5)} + \frac{1}{4} * tan^{-1} * (\frac{x+1}{2})$$

```
syms x
g1=(3*x+7)/(x^2+2*x+5)^2
```

g1 =

(3*x+7)/(x^2+2*x+5)^2

```
int(g1,x)
```

ans =

1/16*(8*x-16)/(x^2+2*x+5)+1/4*atan(1/2*x+1/2)

(ε) Επίλυση εξισώσεων

Η συνάρτηση **solve** χρησιμοποιείται για την εύρεση των ριζών μιας συμβολικής έκφρασης. Μπορούμε να καθορίσουμε μεταβλητή ως προς την οποία θα γίνει η επίλυση (συνιστάται).

```
syms a b c x
```

```
solve(a*x^2+b*x+c,x)
```

ans =

```
-1/2*(b-(b^2-4*a*c)^(1/2))/a
-1/2*(b+(b^24*a*c)^(1/2))/a
```

3.2 ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2–ΣΗΜΑΤΑ ΣΥΝΕΧΟΥΣ ΚΑΙ ΔΙΑΚΡΙΤΟΥ ΧΡΟΝΟΥ

3.2.1 ΣΗΜΑΤΑ ΣΥΝΕΧΟΥΣ ΧΡΟΝΟΥ

Το MATLAB δεν υποστηρίζει συναρτήσεις (και κατ' επέκταση σήματα) συνεχούς χρόνου, π.χ. **s(t) = cos(t)**. Παρόλα αυτά μπορεί κανείς εύκολα να προσεγγίσει τέτοιες συναρτήσεις χρησιμοποιώντας αντίστοιχες συναρτήσεις διακριτού χρόνου με πολύ μικρό βήμα (step size). Για παράδειγμα, οι παρακάτω εντολές

```
t = 0:0.01:10;
y = cos(t);
```

παράγουν μια προσέγγιση της συνάρτησης συνεχούς χρόνου **s(t)=cos(t)**. Η χρήση του τελεστή (operator) : με τρία ορίσματα παράγει το παρακάτω διάνυσμα για την ανεξάρτητη μμεταβλητή του χρόνου:

t = {0.00, 0.01, 0.02, …. , 9.98, 9.99, 10.00}

Από εδώ και στο εξής θα αναφερόμαστε σε συναρτήσεις αυτού του τύπου ως «συνεχείς», αν και στη πραγματικότητα αυτές είναι διακριτού χρόνου με την προϋπόθεση ότι το βήμα της ανεξάρτητης μμεταβλητής επιλέγεται επαρκώς μικρό.

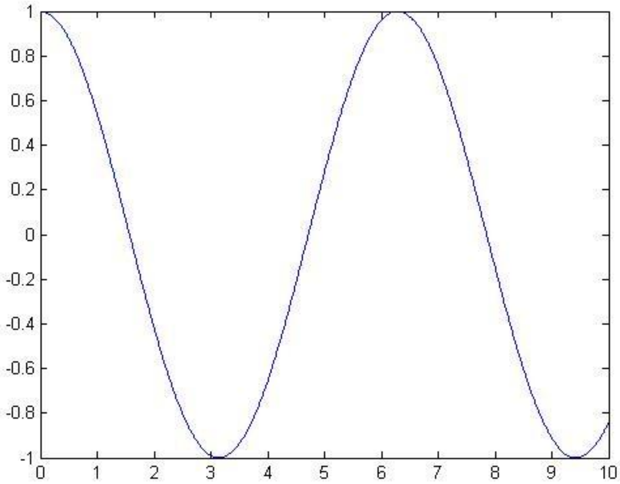
Για να σχεδιάσετε μια συνάρτηση, χρησιμοποιείστε την εντολή (command) **plot**.
Για παράδειγμα η παρακάτω ακολουθία εντολών έχει σαν αποτέλεσμα τη σχεδίαση της συνάρτησης cos(t),

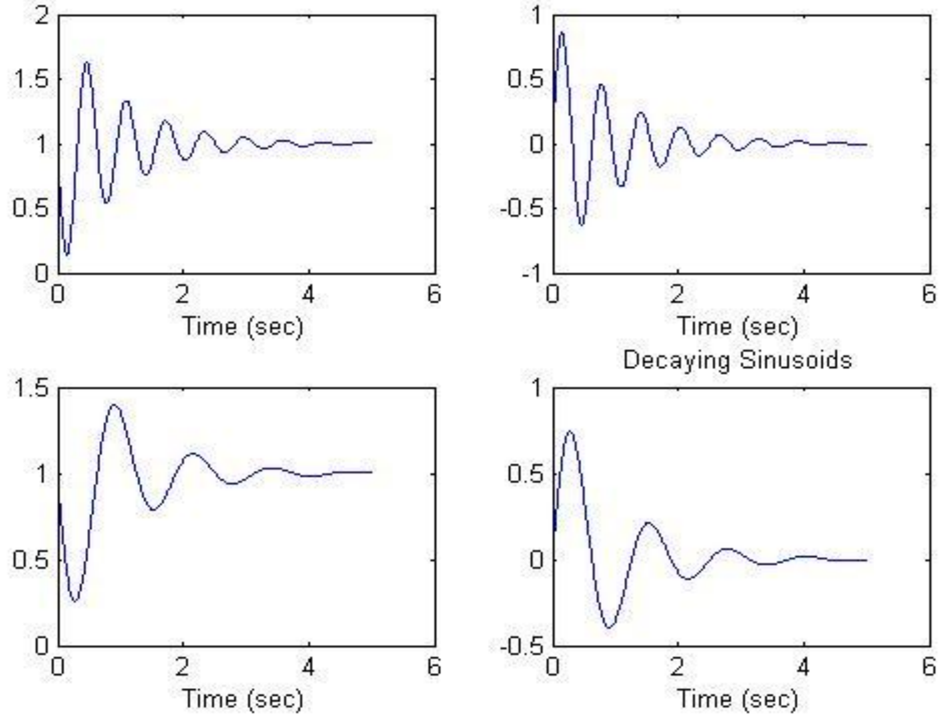
```
t = 0:0.01:10;
y = cos(t);
plot(t,y)
```

Για να σχεδιάσετε στο ίδιο παράθυρο πολλαπλά διαγράμματα διαφόρων συναρτήσεων, χρησιμοποιείστε την εντολή (command) **subplot.**
Δοκιμάστε την παρακάτω ακολουθία εντολών

```
t = 0: .01:5;
y1 = 1 - exp(-t).*sin(10*t);
y2 = exp(-t).*sin(10*t);
y3 = 1 - exp(-t).*sin(5*t);
y4 = exp(-t).*sin(5*t);
subplot(221)
plot(t,y1), xlabel('Time (sec)')
subplot(222)
plot(t,y2), xlabel('Time (sec)')
subplot(223)
plot(t,y3), xlabel('Time (sec)')
subplot(224)
plot(t,y4), xlabel('Time (sec)')
subplot(111)
```

```
title('Decaying Sinusoids')
```





3.2.2. ΣΗΜΑΤΑ ΔΙΑΚΡΙΤΟΥ ΧΡΟΝΟΥ

Το MATLAB είναι δομημένο κατά τέτοιο τρόπο ώστε να αντιμετωπίζει τις μεταβλητές ως διανύσματα και πίνακες.

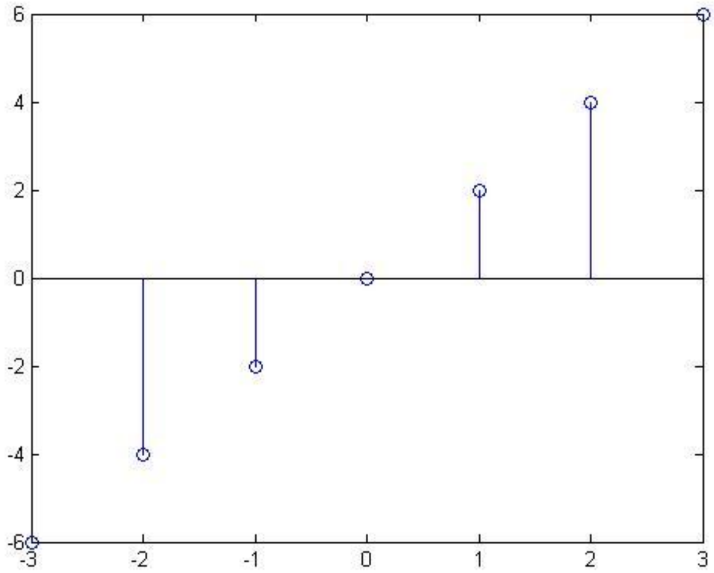
Για την αναπαράσταση ενός σήματος διακριτού χρόνου θα πρέπει να ορισθεί ένα διάνυσμα ακεραίων το οποίο θα αντιπροσωπεύει τους δείκτες n του σήματος και ένα διάνυσμα που θα περιέχει τις τιμές του σήματος για κάθε τιμή του δείκτη x[n].

Για **<u>παράδειγμα</u>**, δοθέντος του σήματος διακριτού χρόνου

$$X[n] = \begin{cases} 2n, & -3 \le n \le 3 \\ 0, & \alpha\lambda\lambda o\acute{\upsilon} \end{cases}$$

πρέπει να χρησιμοποιηθούν οι παρακάτω εντολές για να παραχθούν τα διανύσματα που θα αναπαριστούν τους δείκτες και το σήμα

```
n = [-3:3];
x = 2*n;
stem(n,x)
```



3.2.3 ΕΝΕΡΓΕΙΑ ΣΗΜΑΤΟΣ

Είναι γνωστό ότι η ενέργεια ενός σήματος διακριτού χρόνου δίνεται από τη σχέση

$$E_{\infty=} \sum_{n=-\infty}^{\infty} |X[n]|^2$$

Μπορούμε εύκολα να υπολογίσουμε την ενέργεια τέτοιων σημάτων στο MATLAB χρησιμοποιώντας τις εντολές **sum**, **abs**, και ^, όπως παρουσιάζεται παρακάτω

```
n = [-3:3];
x = 2*n;
energy = sum(abs(x).^2);
```

energy =

$\boxed{\textbf{112}}$

3.2.4 ΠΕΡΙΟΔΙΚΑ ΣΗΜΑΤΑ

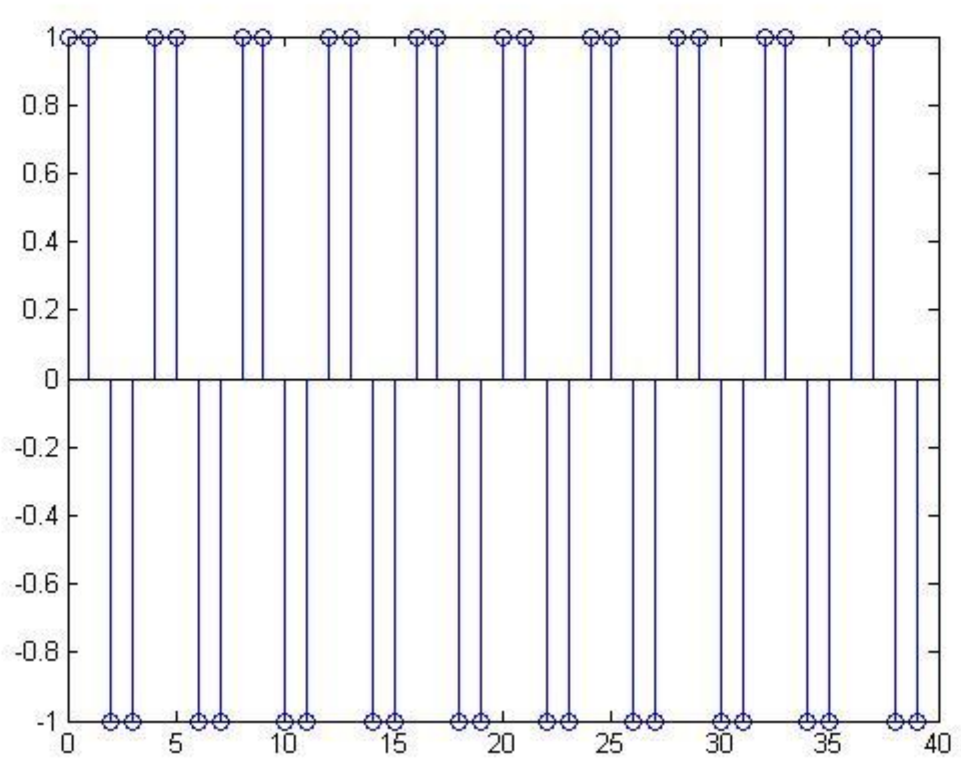
Είναι πολύ εύκολο στο MATLAB να φτιάξουμε περιοδικά σήματα διακριτού χρόνου.
Για παράδειγμα, το περιοδικό σήμα διακριτού χρόνου που ορίζεται για μια περίοδο από τη παρακάτω σχέση (περίοδος No=4)

$$X[n] = \begin{cases} 1, n = 0,1 \\ -1, n = 2,3 \end{cases}$$

μπορεί να παραχθεί και να σχεδιασθεί στο MATLAB εισάγοντας την παρακάτω ακολουθία εντολών

```
N = 10;
 x = [1 1 -1 -1];
 xr = repmat(x,1,N);
 n = 0:length(xr)-1;
stem(n,xr);
```

Σημειώστε ότι η εντολή **repmat** παράγει αντίγραφα ενός διανύσματος ή πίνακα, ενώ η εντολή **Length** επιστρέφει τον αριθμό των στοιχείων ενός διανύσματος.



3.3 ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3 - ΣΥΝΕΛΙΞΗ

Για κάθε γραμμικό και χρονικά αναλλοίωτο σύστημα συνεχούς χρόνου ισχύει ότι η απόκριση y(t) του όταν αυτό διεγείρεται από είσοδο x(t) δίνεται από τη σχέση:

Y(t)=$\int_{-\infty}^{+\infty} x(\tau) * h(t-\tau)d\tau = x(t) * h(t)$

Γραφικός προσδιορισμός της συνέλιξης

Για να υπολογίσουμε την έξοδο ενός ΓΧΑ συστήματος με τη βοήθεια του ολοκληρώματος της συνέλιξης για

κάθε χρονική στιγμή t **ακολουθούμε τα βήματα**:

1^0 Βήμα: **Ανάκλαση:** Αναστρέφουμε ένα από τα δύο σήματα π.χ την κρουστική απόκριση, δηλαδή προσδιορίζουμε την h(-τ). Θα μπορούσαμε να είχαμε αναστρέψει το x(t)

2^0 Βήμα: **Χρονική Μετατόπιση**: Μετατοπίζουμε την h(-τ) κατά t και έτσι προσδιορίζουμε την h(t-τ).

3^0 Βήμα: **Πολλαπλασιασμός:** Προσδιορίζουμε το γινόμενο x(τ) h(t-τ).

4^0 Βήμα: **Ολοκλήρωση ή Εμβαδομέτρηση**: Ολοκληρώνουμε το γινόμενο αυτό .

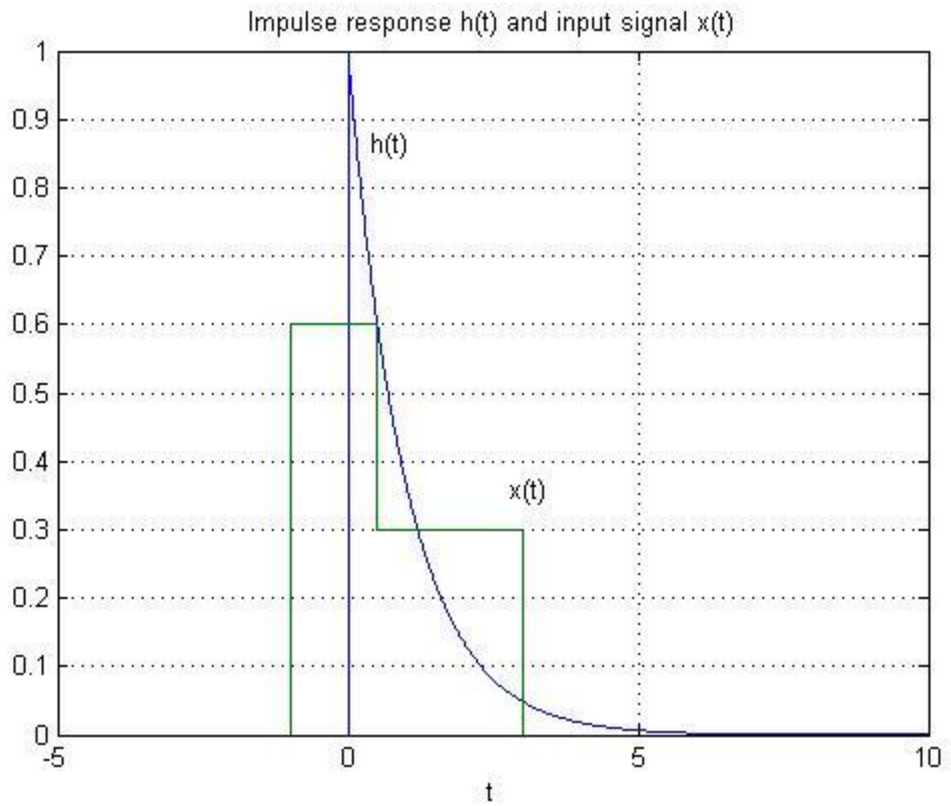
5^0 Βήμα:**Επανάληψη**: Τα βήματα αυτά επαναλαμβάνονται για τις διάφορες τιμές του χρόνου.

Ας δούμε ένα παράδειγμα.

Έστω το ΓΧΑ (Γραμμικό χρονικά αμετάβλητο) σύστημα με κρουστική απόκριση **h(t) = exp(-t)u(t)**.

Έστω η είσοδος **x(t)** που έχει την τιμή **0.6 για -1 < t < 0.5**, την τιμή **0.3 για 0.5 < t < 3**, και είναι **0 αλλού**.

Τα σήματα x και h φαίνονται στο παρακάτω σχήμα:



Τα παραπάνω διαγράμματα προέκυψαν στο Matlab πληκτρολογώντας την εξής σειρά εντολών:

```
th1=linspace(0,10,1001);
h1=exp(-th1);
h=[0 h1];
th=[0 th1];
tx=[-1 -1 0.5 0.5 3 3];
x=[0 0.6 0.6 0.3 0.3 0];
plot(th,h,tx,x)
grid
xlabel('t')
axis([-5 10 0 1]) % Αλλαγή των ορίων των αξόνων
title('Impulse response h(t) and input signal x(t)')
gtext('x(t)')
gtext('h(t)')
```

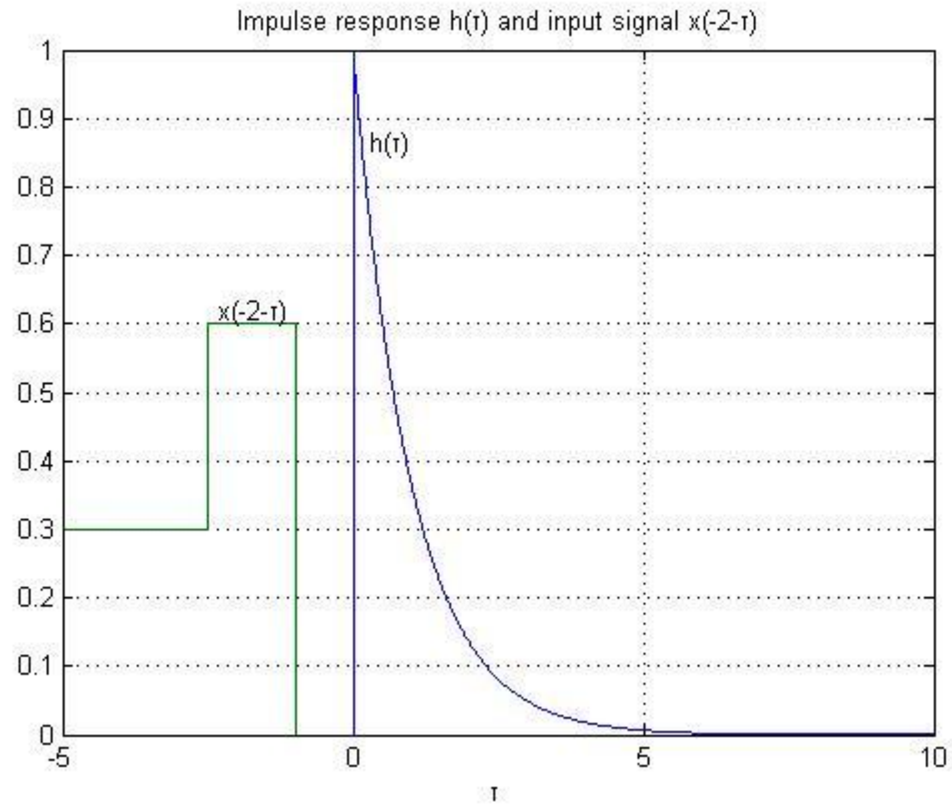
Οι εντολές **gtext** χρησιμοποιήθηκαν για να βάλουν τις ετικέττες x(t), h(t) δίπλα στις αντίστοιχες καμπύλες.

Ας περάσουμε τώρα στον υπολογισμό της συνέλιξης.

- Προφανώς, επειδή η τιμή του x δεν είναι σταθερή παρά μόνο κατά τμήματα, το ολοκλήρωμα του γινομένου **h(τ)x(t-τ)** θα πρέπει να υπολογιστεί ξεχωριστά για τα διάφορα τμήματα.

Στο παραπάνω σχήμα το t=0. **Για t<-1, δεν υπάρχει επικάλυψη** ανάμεσα στα γραφήματα των h(τ) και x(t-τ), συνεπώς το y(t)=0.

Ας το δούμε αυτό σ' ένα σχήμα για t=-2:



Πώς προέκυψε το παραπάνω; Απλά προσθέτοντας το –2 στο διάνυσμα –tx: **plot(th,h,-2-tx,x)**

```
th1=linspace(0,10,1001);
h1=exp(-th1);
h=[0 h1];
th=[0 th1];
tx=[-1 -1 0.5 0.5 3 3];
x=[0 0.6 0.6 0.3 0.3 0];
plot(th,h,-2-tx,x)
grid
xlabel('τ')
axis([-5 10 0 1]) % Αλλαγή των ορίων των αξόνων
title('Impulse response h(τ) and input signal x(2-τ)')
gtext('x(-2-τ)')
gtext('h(τ)')
```

- Αν t>-1, τότε υπάρχει μη-μηδενική επικάλυψη των δύο σημάτων. Για –1<t<0.5 υπάρχει επικάλυψη με το h(τ) μόνο του τμήματος του x με τιμή 0.6.

- Στο διάστημα αυτό, δηλαδή –1<t<0.5, το ολοκλήρωμα υπολογίζεται ως εξής

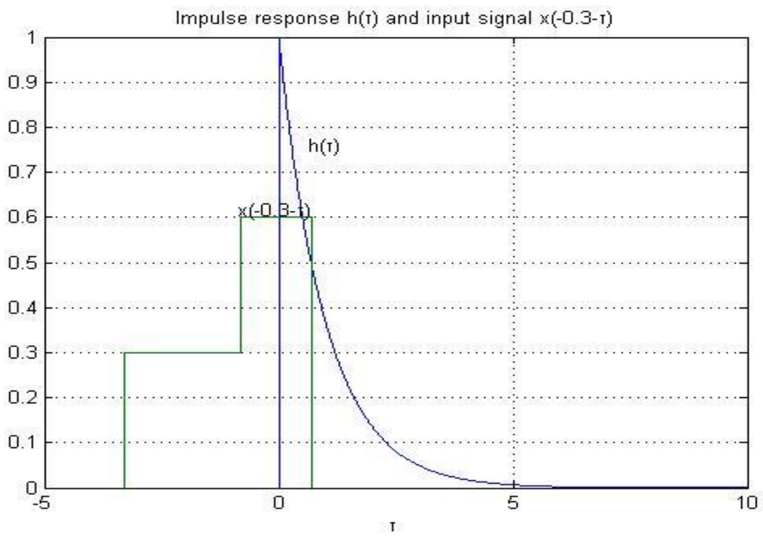
$$\boldsymbol{y(t)=\int_{-1}^{t} x(\tau)*h(t-\tau)d\tau}$$

$$\boldsymbol{=\int_{-1}^{t} 0.6*e^{-(t-\tau)}\,d\tau = 0.6*e^{-t}\int_{-1}^{t} e^{\tau}\,d\tau}$$

$$\boldsymbol{=0.6*e^{-t}*(e^{t}-e^{-1})=}$$

$$\boldsymbol{=0.6(1-e^{-1-t})\,,\quad -1<t<0.5}$$

Στο ακόλουθο σχήμα φαίνεται η κατάσταση για t=-0.3.



Ξανά, η εντολή γραφικής παράστασης ήταν : plot(th,h,-0.3-tx,x)

```
th1=linspace(0,10,1001);
h1=exp(-th1);
h=[0 h1];
th=[0 th1];
tx=[-1 -1 0.5 0.5 3 3];
x=[0 0.6 0.6 0.3 0.3 0];
plot(th,h,-0.3-tx,x)
grid
xlabel('τ')
axis([-5 10 0 1]) % Αλλαγή των ορίων των αξόνων
title('Impulse response h(τ) and input signal x(-0.3-τ)')
gtext('x(-0.3-τ)')
gtext('h(τ)')
```

- Η επόμενη περίπτωση είναι να επικαλύπτεται με το h(τ) ολόκληρο το κομμάτι του x με τιμή 0.6 ενώ το κομμάτι 0.3 να επικαλύπτεται μόνο μερικά. **Αυτό συμβαίνει για 0.5<t<3**, και φαίνεται στο παρακάτω σχήμα για την περίπτωση t=2.

- Η τιμή του συνελικτικού ολοκληρώματος υπολογίζεται τότε ως εξής :

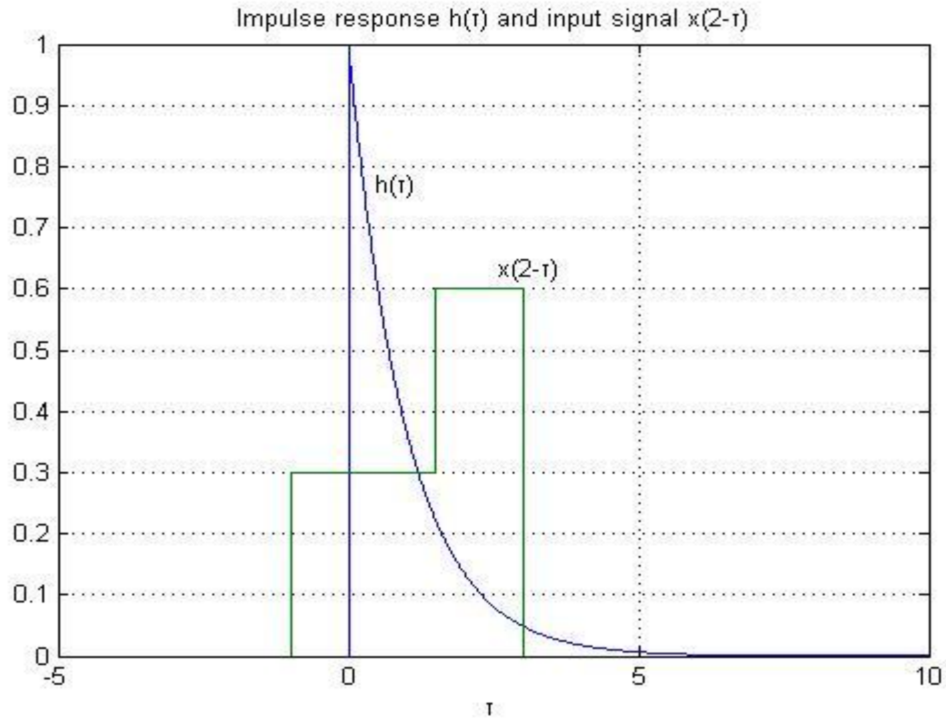
$$\boldsymbol{y(t)} = \int_{-1}^{0.5} \mathbf{0.6} * e^{-(t-\tau)}\, d\tau + \int_{0.5}^{t} \mathbf{0.3} * e^{-(t-\tau)}\, d\tau$$

$$= \mathbf{0.6} * e^{-t} \int_{-1}^{0.5} e^{\tau}\, d\tau + \mathbf{0.3} * e^{-t} \int_{0.5}^{t} e^{\tau}\, d\tau$$

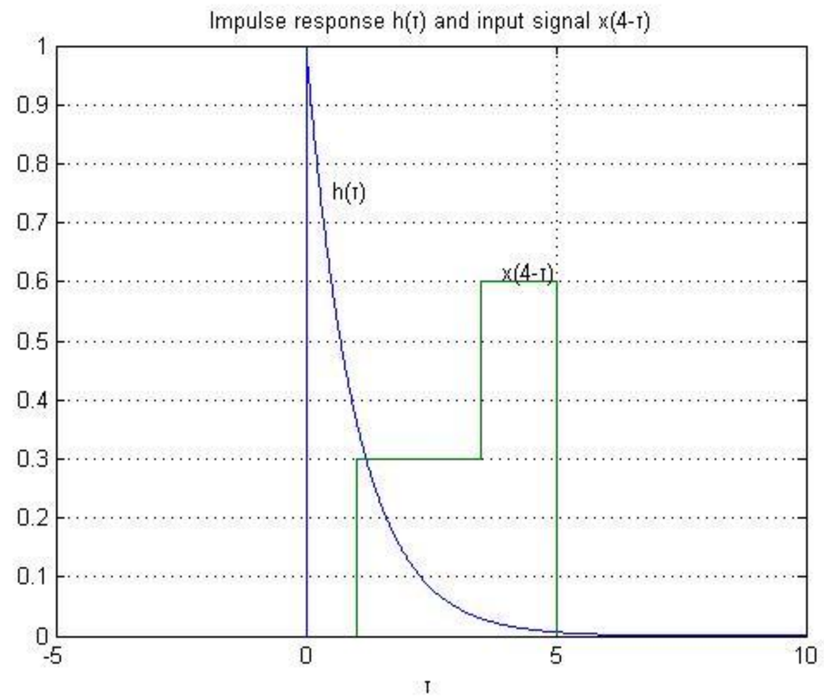
$$= \mathbf{0.6} * e^{-t}\left(e^{0.5} - e^{-1}\right) + \mathbf{0.3} * e^{-t}\left(e^{t} - e^{0.5}\right)$$

$$= \mathbf{0.3} * e^{-t}\left(1 + e^{0.5} - 2 * e^{-1}\right), \quad 0.5 < t < 3$$

```
th1=linspace(0,10,1001);
h1=exp(-th1);
h=[0 h1];
th=[0 th1];
tx=[-1 -1 0.5 0.5 3 3];
x=[0 0.6 0.6 0.3 0.3 0];
plot(th,h,2-tx,x)
grid
xlabel('τ')
axis([-5 10 0 1]) % Αλλαγή των ορίων των αξόνων
title('Impulse response h(τ) and input signal x(2-τ)')
gtext('x(2-τ)')
gtext('h(τ)')
```



- **Η τελευταία περίπτωση είναι να επικαλύπτονται πλήρως τα δύο σήματα, κάτι που συμβαίνει για t>3:**



Το ολοκλήρωμα υπολογίζεται ως εξής :

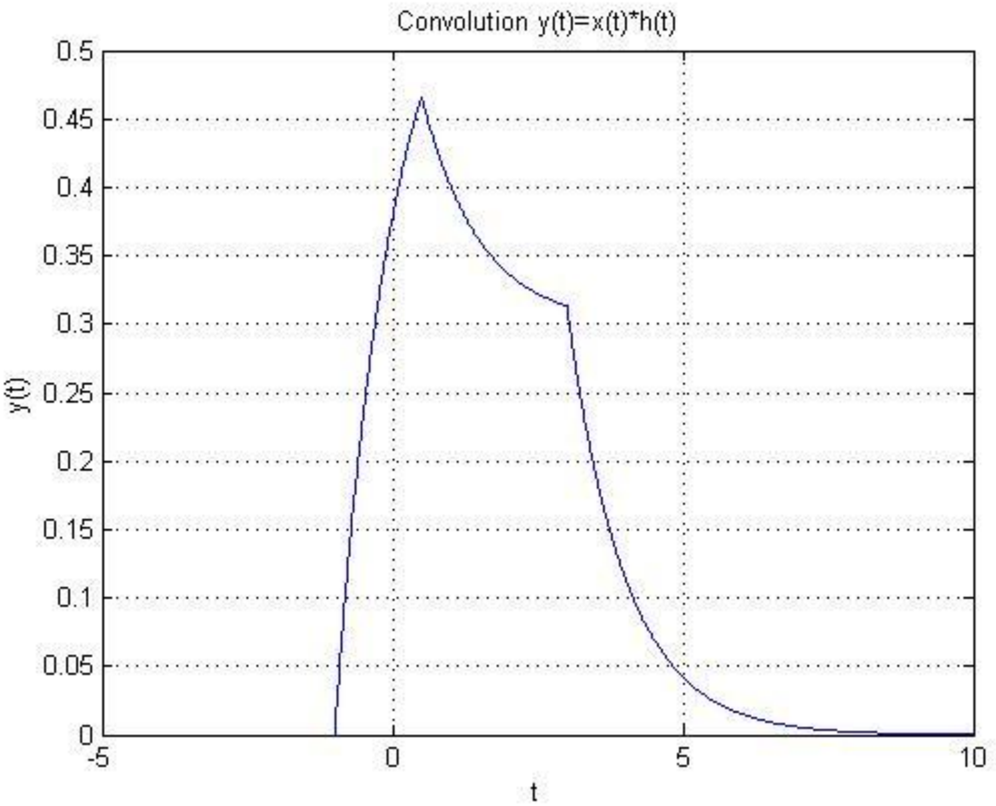
$$\boldsymbol{y(t) = \int_{-1}^{0.5} 0.6 * e^{-(t-\tau)}\, d\tau + \int_{0.5}^{3} 0.3 * e^{-(t-\tau)}\, d\tau}$$

$$\boldsymbol{= 0.6 * e^{-t} \int_{-1}^{0.5} e^{\tau}\, d\tau + 0.3 * e^{-t} \int_{0.5}^{3} e^{\tau}\, d\tau}$$

$$\boldsymbol{= 0.6 * e^{-t}\left(e^{0.5} - e^{-1}\right) + 0.3 * e^{-t}\left(e^{3} - e^{0.5}\right)}$$

$$\boldsymbol{= 0.3 * e^{-t}\left(e^{0.5} - 2 * e^{-1} + e^{3}\right),\ \ t > 3}$$

- Και να πώς μπορούμε να παραστήσουμε γραφικά το συνολικό αποτέλεσμα της συνέλιξης:



- Το τελευταίο σχήμα προέκυψε πληκτρολογώντας

```
ty1=[-1:0.01:0.5];
```

```
y1=0.6*(1-exp(-ty1-1));
ty2=[0.5:0.01:3];
y2=0.3*(1+(exp(0.5)-2*exp(-1))*exp(-ty2));
ty3=[3:0.01:10];
y3=0.3*(exp(3)+exp(0.5)-2*exp(-1))*exp(-ty3);
ty=[-5 ty1 ty2 ty3];
y=[0 y1 y2 y3];
plot(ty,y)
grid
xlabel('t')
ylabel('y(t)')
title('Convolution y(t)=x(t)*h(t)')
```

Στα παραπάνω διαθέταμε την αναλυτική περιγραφή των δύο συνελισσόμενων σημάτων και υπολογίσαμε τη συνέλιξή τους βρίσκοντας αναλυτικά το αντίστοιχο ολοκλήρωμα.

Ας δούμε τώρα πώς μπορούμε να χρησιμοποιήσουμε τις λειτουργίες του Matlab όχι μόνο για να κάνουμε τις γραφικές παραστάσεις αλλά και για ν' αποφύγουμε τον αναλυτικό υπολογισμό του ολοκληρώματος.

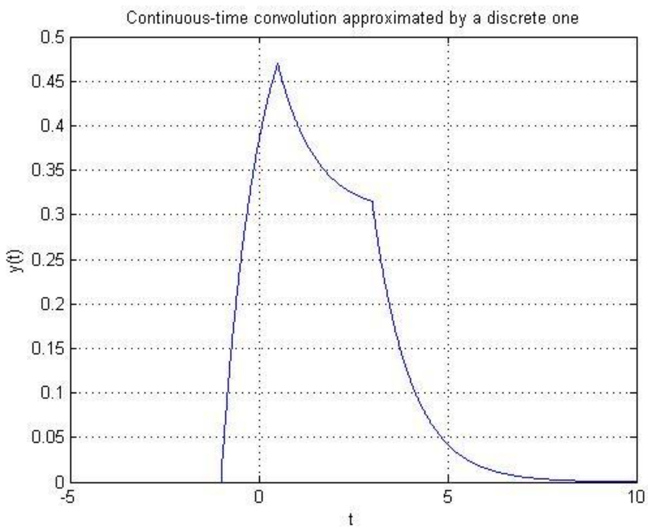
Ευτυχώς το Matlab διαθέτει έτοιμη function για τον υπολογισμό του παραπάνω αθροίσματος, την conv.

<u>Εδώ όμως πρέπει να προσέξουμε μια σημαντική λεπτομέρεια:</u> Η conv «θεωρεί» ότι τα σήματα διακριτού χρόνου x, h είναι αιτιατά. Έτσι, για να την εφαρμόσουμε στο παράδειγμά μας, θα πρέπει πρώτα να θεωρήσουμε ότι το x ολισθαίνει κατά 1 προς τα δεξιά ώστε να γίνει κι αυτό αιτιατό. Είναι πρόβλημα αυτό; Καθόλου, αφού το σύστημά μας είναι χρονικά αμετάβλητο. Θα ξέρουμε ότι και η έξοδος έχει υποστεί την ίδια ολίσθηση, άρα θα πρέπει να θυμηθούμε να την επαναφέρουμε στη θέση της.

Για το παράδειγμα των σημάτων που συζητήσαμε παραπάνω, η συνέλιξη μπορεί να υπολογιστεί με τις ακόλουθες εντολές:

```
t1=[0:0.01:1.5]; % T=0.01
t2=[1.5+0.01:0.01:4];
t3=[4.01:0.01:10]; % Θεωρούμε τα σήματα στο διάστημα [0,10]
 x=[0.6*ones(size(t1)) 0.3*ones(size(t2)) zeros(size(t3))];
 h=exp(-[t1 t2 t3]);
yc=conv(x,h)*0.01; % Προσέγγιση του ολοκληρώματος από άθροισμα
plot([-1,-1:0.01:19],[0 yc]) % Ολισθαίνουμε κατά 1 προς τ’
 % αριστερά, αλλάζοντας το διάστημα
 % [0,20] στο [-1,19].
 axis([-5 10 0 0.5])
 grid
 xlabel('t')
 ylabel('y(t)')
 title('Continuous-time convolution approximated by a discrete one')
```

Το αποτέλεσμα, που φαίνεται παρακάτω, είναι μια καλή προσέγγιση αυτού που βρήκαμε αναλυτικά:



3.4 ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4 – ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΙ LAPLACE ΚΑΙ Z

3.4.1 ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ LAPLACE

Ο μετασχηματισμός Laplace μετασχηματίζει τις διαφορικές εξισώσεις που περιγράφουν τα γραμμικά μη χρονικά μεταβαλλόμενα συστήματα συνεχούς χρόνου, σε αλγεβρικές εξισώσεις και έχει καθιερωθεί σαν βασικό εργαλείο μελέτης και σχεδίασης συστημάτων συνεχούς χρόνου.

Ορισμός : Καλούμε ευθύ μετασχηματισμό LAPLACE δοθείσης συνάρτησης f(t) με $t \in (0,\ )$ το γενικευμένο ολοκλήρωμα:

$L\{f(t)\} = F(s) = \int_0^\infty f(t) * e^{-st}\, dt$ αν υπάρχει.

Το Matlab μας δίνει τη δυνατότητα να βρούμε απευθείας το μετασχηματισμό Laplace μιας συνάρτησης με την εντολή **Laplace** ( ).

- Το Matlab επίσης μας δίνει τη δυνατότητα να βρούμε απευθείας και τον αντίστροφο μετασχηματισμό Laplace μιας συνάρτησης με την εντολή **ilaplace** ( ).

Σύνταξη εντολής Laplace:

μετασχηματισμός Laplace= Laplace(συνάρτηση)

Σύνταξη εντολής ilaplace:

αντίστροφος μετασχηματισμός Laplace= ilaplace (συνάρτηση στη μορφή Laplace)

Ας πάρουμε το <u>**παράδειγμα**</u> του σήματος

$$\boldsymbol{h(t) = \left(5e^{-2t} + 3e^{-0.1t}6te^{-5t}\right)u(t)}$$

Από την ιδιότητα της γραμμικότητας και το ζεύγος ML (δες ΤΥΠΟΛΟΓΙΟ)

$$\frac{\mathbf{t}^{\mathbf{n-1}}}{(\mathbf{n-1})!}\mathbf{e}^{-\mathbf{at}}\mathbf{u(t)} \leftrightarrow \frac{\mathbf{1}}{(\mathbf{s+a})^{\mathbf{n}}}$$

όπου ο ML έχει περιοχή σύγκλισης (ΠΣ) $\boldsymbol{Re(s) > -a}$, **βρίσκουμε το ML του παραπάνω σήματος ως εξής:**

$$\boldsymbol{H(s) = \frac{5}{s+2} + \frac{3}{s+0.1} + \frac{6}{(s+5)^2} \quad , Re(s) > -0.1}$$

Ή μετά από πράξεις

$$\boldsymbol{H(s) = \frac{8s^3 + 92.5s^2 + 277.6s + 163.7}{s^4 + 12.1s^3 + 46.2s^2 + 54.5s + 5} = \frac{8(s + 7.0553)(s + 3.7296)(s + 0.7777)}{(s + 2)(s + 0.1)(s + 5)^2}}$$

Ο κώδικας σε Matlab για τον υπολογισμό του Μετασχηματισμού Laplace ακολουθεί:

```
syms a s t w x
laplace(5*exp(-2*t)+3*exp(-.1*t)+6*t*exp(-5*t)
```

ans =

5/(s+2)+30/(10*s+1)+6/(s+5)^2

Για την αντιστροφή ρητής συνάρτησης $H\,(s)$ έχουμε τη μέθοδο της ανάλυσης σε άθροισμα απλών κλασμάτων, των οποίων ξέρουμε τον αντίστροφο Μετασχηματισμό Laplace.

Κάτι τέτοιο μπορεί να γίνει στο MATLAB με τη βοήθεια της συνάρτησης residue. Για μια συνάρτηση,

$\boldsymbol{H(s)} = \frac{\boldsymbol{B(s)}}{\boldsymbol{A(s)}}$ η εντολή του MATLAB **[R,P,K]=residue(B,A);** τη γράφει σε άθροισμα

```
[R,P,K]=residue(B,A);
```

R =

0.0264 + 0.0152i
0.0264 - 0.0152i
0.1929
-0.2216 - 0.1966i
-0.2216 + 0.1966i
-0.0163 + 0.0016i
-0.0163 - 0.0016i
-0.0251 - 0.2522i
-0.0251 + 0.2522i

0.1401 - 0.1371i
0.1401 + 0.1371i

P =

-2.4656 + 1.5851i
-2.4656 - 1.5851i
-1.0585
0.9005 + 0.5391i
0.9005 - 0.5391i
0.4989 + 0.9135i
0.4989 - 0.9135i
-0.8776 + 0.5404i
-0.8776 - 0.5404i
-0.0269 + 1.0200i
-0.0269 - 1.0200i

K =

[]

$$\boldsymbol{H(s)} = \frac{\boldsymbol{R(1)}}{\boldsymbol{s - P(1)}} + \frac{\boldsymbol{R(2)}}{\boldsymbol{s - P(2)}} + \cdots + \frac{\boldsymbol{R(n)}}{\boldsymbol{s - P(n)}} + \boldsymbol{K(s)}$$

Κάτι ανάλογο συμβαίνει όταν οι πόλοι, δηλαδή οι ρίζες του $A(s)$ δεν είναι διαφορετικές μεταξύ τους, όπως συμβαίνει στο παράδειγμά μας, όπου υπάρχει ο διπλός πόλος –5.

Ας δούμε τι μας δίνει η **residue**:

[R,P,K]=residue([8 92.5 277.6 163.7],[1 12.1 46.2 54.5 5])

R =

-0.0000
6.0000
5.0000
3.0000

P =

-5.0000
-5.0000
-2.0000
-0.1000

K =

[]

Το K **είναι κενό** αφού ο βαθμός του αριθμητή είναι μικρότερος αυτού του παρονομαστή.

Για να το επαληθεύσουμε, μπορούμε να ξαναχρησιμοποιήσουμε τη **residue**, αυτή τη φορά ανάποδα:

```
[B,A]=residue(R,P,K)
```

B =

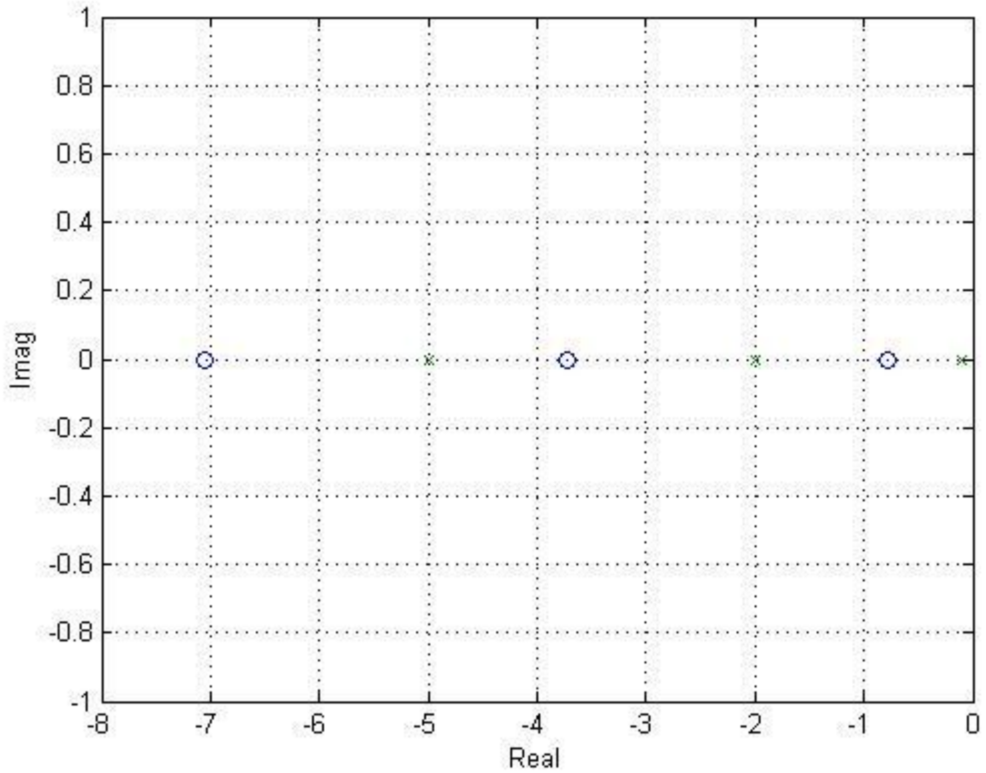
8.0000	**92.5000**	**277.6000**	**163.7000**

A =

1.0000	**12.1000**	**46.2000**	**54.5000**	**5.0000**

Μπορούμε επίσης να δούμε και **τα μηδενικά και τους πόλους** της συνάρτησης μεταφοράς στο επίπεδο s, ως εξής:

```
zeros=roots(B); % H roots βρίσκει ρίζες πολυωνύμου
poles=roots(A);
plot(real(zeros),imag(zeros),'o',real(poles),imag(poles),'x')
grid
xlabel('Real')
ylabel('Imag')
```



<u>Παράδειγμα</u>

Να βρεθεί ο μετασχηματισμός Laplace της συνάρτησης

f(t)=(e^2t)(t-2)² και στη συνέχεια ο αντίστροφος μετασχηματισμός Laplace.

Γράφουμε τον παρακάτω κώδικα

```
syms f s t
f=exp(2*t)*(t-3)^2;
laplace(f)
```

ans = **(50-42*s+9*s^2)/(s-2)^3**

ilaplace(ans)

ans = **exp(2*t)*(t-3)^2**

3.4.2 ΕΠΙΛΥΣΗ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΜΕ ΤΗ ΜΕΘΟΔΟ ΤΟΥ ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΥ LAPLACE

Τα όσα αναφέρονται παρακάτω δουλεύουν μόνο για

- *Γραμμικές Δ.Ε με σταθερούς συντελεστές*
- συναρτήσεις που είναι αθροίσματα και γινόμενα απο
    - πολυώνυμα
    - εκθετικές συναρτήσεις
    - ημίτονα και συνημίτονα
    - Βηματικές συναρτήσεις
    - Κρουστικές συναρτήσεις
- Αρχικές συνθήκες στο $t = 0$

Ας θεωρήσουμε τη Δ.Ε

$$\mathbf{y'' + 3y' + 2y = e^{-t}}\ , \qquad \mathbf{y(0) = 4, y'(0) = 5}$$

Κατ'αρχή δηλώνουμε τις συμβολικές μεταβλητές:

```
syms s t Y
```

Με το f δηλώνουμε το δεξί μέρος της εξίσωσης και βρίσκουμε τον μετασχηματισμό Laplace της τον οποίο συμβολίζουμε με F:

```
f = 'exp(-t)'
```

f= exp(-t)

```
F = laplace(f,t,s)
```

F= 1/(1+s)

Βρίσκουμε τους μετασχηματισμούς Laplace των $y'(t)$: $Y1 = s\ Y - y(0)$ και $y''(t)$: $Y2 = =s\ Y1 - y'(0)$

```
Y1 = s*Y - 4;
```

```
Y2 = s*Y1 - 5;
```

Χρησιμοποιούμε την εντολή **Solve** του Matlab για να λύσουμε ως προς Y(s)

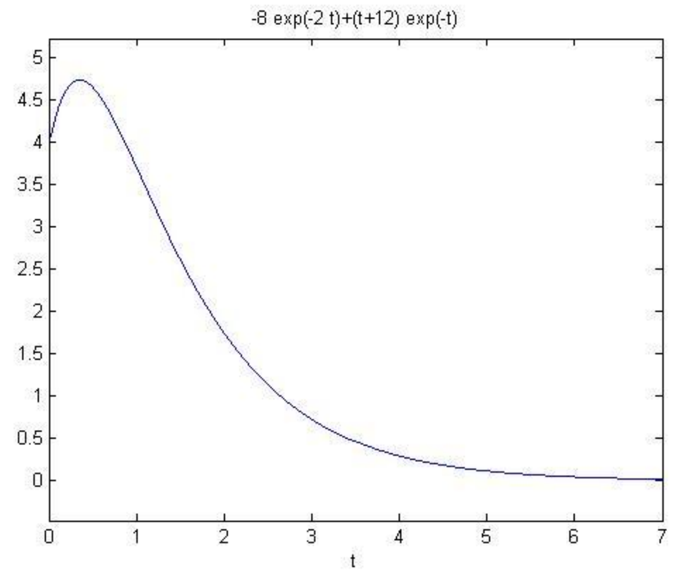
Sol = solve(Y2 + 3*Y1 + 2*Y - F, Y)

Sol=Y(s)= (21*s+4*s^2+18)/(1+s)/(3*s+2+s^2)

Για να βρούμε τη λύση της Δ.Ε δηλαδή το y(t) αρκεί να υπολογίσουμε τον αντίστροφο μετασχηματισμό Laplace του Y(s) δηλαδή

sol = ilaplace(Sol,s,t)

Sol=y(t)= -8*exp(-2*t)+(t+12)*exp(-t)



3.4.3 ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ Z

Ο μετασχηματισμός Z μετασχηματίζει τις εξισώσεις διαφορών που περιγράφουν τα γραμμικά μη χρονικά μεταβαλλόμενα συστήματα διακριτού χρόνου, σε αλγεβρικές εξισώσεις και έχει καθιερωθεί σαν βασικό εργαλείο μελέτης και σχεδίασης συστημάτων διακριτού χρόνου.

Ορισμός : Ο μονόπλευρος μετασχηματισμός Z μιας αιτιατής ακολουθίας f(k) ορίζεται ως:

$$\mathbf{Z[f(k)] = F(z) = \sum_{k=0}^{\infty} f(k) z^{-k} = ZT[f(k)]}$$

- Το Matlab μας δίνει τη δυνατότητα να βρούμε απευθείας το μετασχηματισμό Z μιας συνάρτησης με την εντολή **ztrans**( ).

- Επίσης δίνει τη δυνατότητα να βρούμε απευθείας και τον αντίστροφο μετασχηματισμό Z μιας συνάρτησης με την εντολή **iztrans**( ).

Σύνταξη εντολής ztrans:

μετασχηματισμός Z = ztrans (συνάρτηση)

Σύνταξη εντολής iztrans:

αντίστροφος μετασχηματισμός Z = iztrans (συνάρτηση)

<u>Παράδειγμα</u>

Να βρεθεί ο μετασχηματισμός Z της συνάρτησης **f(n)=n²** και στη συνέχεια ο αντίστροφος μετασχηματισμός Z.

Ο κώδικας που απαιτείται είναι:

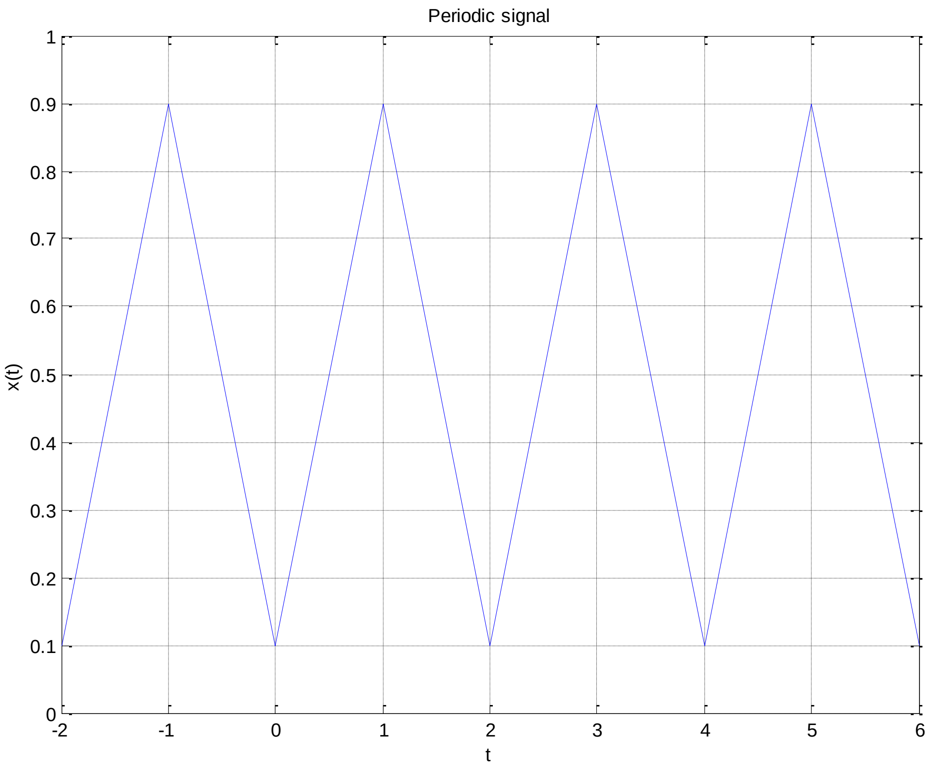
```
syms n z
f=n^2;

ztrans(f)

ans =
          z*(z+1)/(z-1)^3

iztrans(ans)

ans =
         n^2
```

3.5 ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 5 – ΣΕΙΡΑ FOURIER

Αυτή η άσκηση αφορά στη σειρά Fourier (τριγωνομετρική και εκθετική) περιοδικών σημάτων, στο φάσμα αυτών, και στην προσέγγισή τους από πεπερασμένες σειρές Fourier. Ας ξεκινήσουμε με το παράδειγμα του περιοδικού σήματος που, στη βασική του περίοδο, ορίζεται ως εξής:

$$x(t)=\begin{cases}0.1+0.8t, & 0\le t\le 1\\ 1.7-0.8t, & 1<t\le 2\end{cases}$$

Η περίοδός του είναι, προφανώς, $T=2$, άρα η θεμελιώδης (βασική) συχνότητα είναι $\Omega_0=2\pi/T=\pi$ rad/sec. Τέσσερις περίοδοι του σήματος φαίνονται στο παρακάτω σχήμα:



που προέκυψε με τις ακόλουθες εντολές:

>> x=[0.1 0.9 0.1]; % Βασική περίοδος

```
>> x=[x x x x]; % Τέσσερις περίοδοι

>> tx=[-2 -1 0 0 1 2 2 3 4 4 5 6]; % Χρονικό διάστημα

>> figure(1)

>> plot(tx,x)

>> grid

>> xlabel('t')

>> ylabel('x(t)')

>> title('Periodic signal')

>> axis([-2 6 0 1])
```

Ως περιοδικό, το παραπάνω σήμα μπορεί να παρασταθεί από τη σειρά Fourier, που στην τριγωνική της μορφή έχει ως εξής:

$$x(t) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos n\Omega_0 t + b_n \sin n\Omega_0 t\right)$$

με

$$a_0 = \frac{1}{T}\int_{t_0}^{t_0+T} x(t)dt$$

και για $n > 0$

$$a_n = \frac{2}{T}\int_{t_0}^{t_0+T} x(t)\cos n\Omega_0 t dt\ ,$$

$$b_n = \frac{2}{T}\int_{t_0}^{t_0+T} x(t)\sin n\Omega_0 t dt$$

για οποιοδήποτε t_0. Λόγω του ότι είναι άρτια συνάρτηση, δηλαδή $x(-t) = x(t)$, οι συντελεστές των ημιτόνων είναι μηδενικοί: $b_n = 0$. Για τα a_n, με $t_0 = -T/2 = -1$, έχουμε:

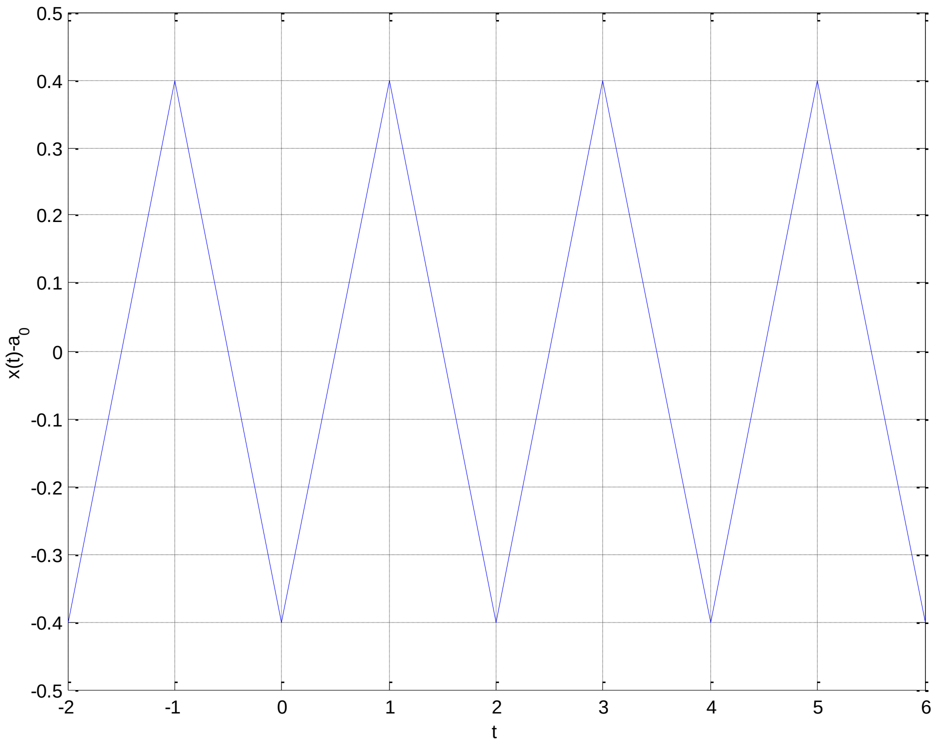
$$a_0 = \frac{1}{2}\int_{-1}^{1} x(t)dt = \int_0^1 x(t)dt = \int_0^1 \left(0.1 + 0.8t\right)dt = \left[0.1t + 0.4t^2\right]_0^1 = 0.5,$$

$$a_n = \int_{-1}^{1} x(t)\cos n\pi t dt = 2\int_0^1 x(t)\cos n\pi t dt = \frac{2}{n\pi}\left(\left[\left(0.1+0.8t\right)\sin n\pi t\right]_0^1 - \int_0^1 0.8\sin n\pi t dt\right)$$
$$= \frac{2}{n\pi}\left[0.9\sin n\pi + \frac{0.8}{n\pi}\left(\cos n\pi - 1\right)\right] = \frac{2}{n\pi}\cdot\frac{0.8}{n\pi}\left(\left(-1\right)^n - 1\right)$$

δηλαδή

$$a_n = \begin{cases} 0, & n = 2k \\ -\dfrac{3.2}{n^2\pi^2} = -\dfrac{0.3242}{n^2}, & n = 2k+1 \end{cases}$$

Άρα οι άρτιες αρμονικές δεν υπάρχουν στο σήμα αυτό. Αυτό θα έπρεπε να το αναμένουμε γιατί εκτός από την άρτια συμμετρία υπάρχει στο σήμα και μια περιττή συμμετρία, που φαίνεται αν το δούμε αφού του αφαιρέσουμε τη μέση τιμή του, a_0:



Παρατηρούμε για την $x_1(t) = x(t) - a_0$ τότε ότι $x_1(t) = -x_1(t + T/2) = -x_1(t+1)$, δηλαδή η συνάρτηση $x_1(t)$ είναι περιττή (όχι ως προς τον άξονα $t = 0$, είναι άρτια ως προς αυτόν, αλλά ως προς τον $t = -0.5$).

Ας δούμε τώρα εναλλακτικές μορφές της ίδιας σειράς Fourier. Χρησιμοποιώντας τη γνωστή μας τριγωνομετρική ταυτότητα:

$$\cos\left(n\Omega_0 t - \phi_n\right) = \cos\phi_n \cos n\Omega_0 t + \sin\phi_n \sin n\Omega_0 t$$

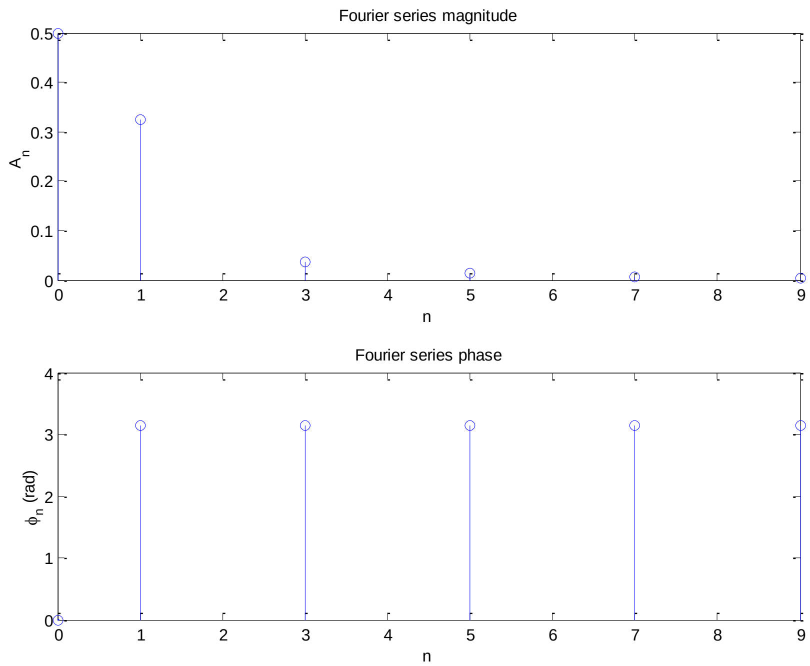
συμπεραίνουμε ότι η τριγωνομετρική σειρά Fourier πραγματικού σήματος μπορεί να γραφτεί ως σειρά συνημιτόνων:

$$x(t) = \sum_{n=0}^{\infty} A_n \cos\left(n\Omega_0 t - \phi_n\right)$$

όπου $A_0 = a_0, \phi_0 \in \{0, \pi\}, A_n = \sqrt{a_n^{\,2} + b_n^{\,2}}, \phi_n = \tan^{-1}\left(b_n / a_n\right)$, για $n > 0$.

Για το παράδειγμα που εξετάζουμε, έχουμε: $A_n = |a_n|, \phi_n = \pi$, για $n > 0$.

Η ακολουθίες A_n, ϕ_n καλούνται ***φάσματα πλάτους*** και ***φάσης***, αντίστοιχα, και φαίνονται στα παρακάτω σχήματα:



Οι αντίστοιχες εντολές είναι:

>> **a0=0.5;**

>> **ph0=0;**

>> **n=[1 3 5 7 9];**

>> **an=-3.2./(pi^2*n.^2);**

```
>> An=abs(an);

>> phn=pi*ones(size(n));

>> figure(2)

>> subplot(211)

>> stem([0 n],[a0 An])

>> xlabel('n')

>> ylabel('A_{n}')

>> title('Fourier series magnitude')

>> subplot(212)

>> stem([0 n],[ph0 phn])

>> xlabel('n')

>> ylabel('\phi_{n} (rad)')

>> title('Fourier series phase')
```

Χρησιμοποιήσαμε για τις γραφικές παραστάσεις όχι τη γνωστή μας συνάρτηση plot αλλά τη **stem,** που είναι πιο κατάλληλη όταν πρόκειται να παρασταθούν ακολουθίες. Η χρήση της είναι ανάλογη αυτής της plot. Τις άρτιες αρμονικές δεν τις συμπεριλάβαμε αφού αυτές είναι μηδενικές.

Η μιγαδική εκθετική σειρά Fourier είναι:

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\Omega_0 t}$$

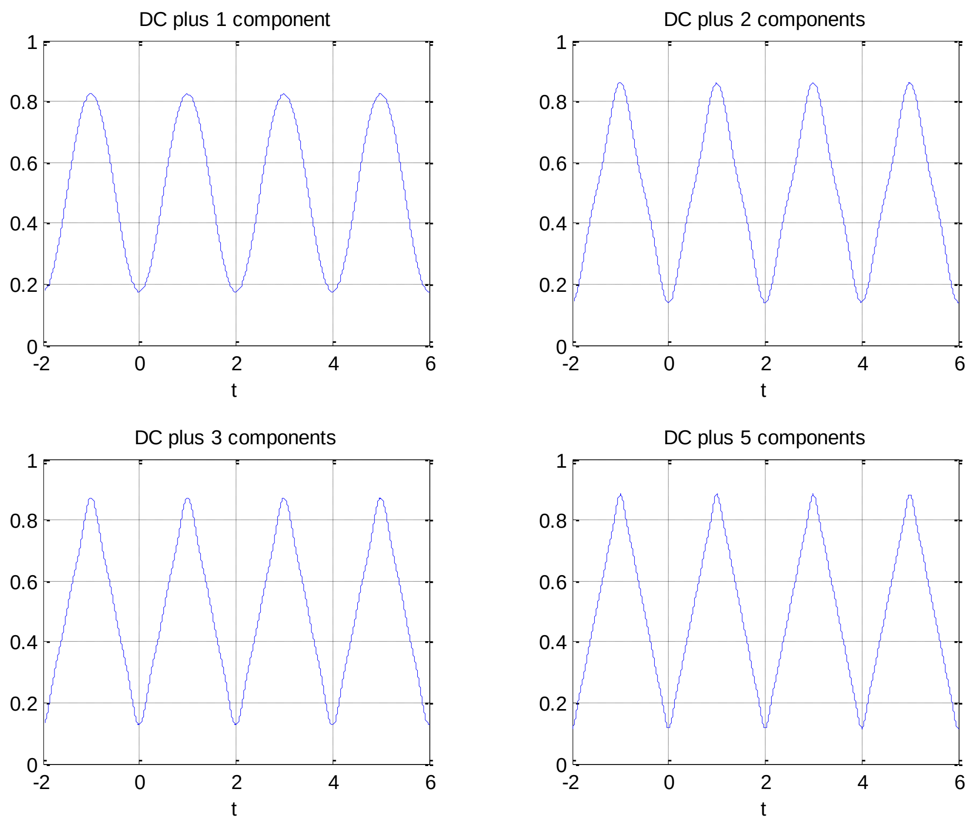
με $c_n = \frac{1}{T}\int_{t_0}^{t_0+T} x(t)e^{-jn\Omega_0 t}dt$. Η σχέση της με την τριγωνομετρική σειρά Fourier είναι: $c_0 = a_0, c_n = \frac{1}{2}\left(a_n - jb_n\right), n > 0$. Για πραγματικά σήματα: $c_n^* = c_{-n}$ και $\mathrm{Re}(c_n) = a_n/2, \mathrm{Im}(c_n) = -b_n/2$. Επίσης τότε έχουμε για $n > 0$, $|c_n| = \sqrt{a_n^2 + b_n^2}/2 = A_n/2$ και $\angle c_n = -\phi_n$. Για το παράδειγμά μας, $c_0 = 0.5, c_{\pm n} = a_n/2$. Αυτά φαίνονται στο παρακάτω σχήμα, για το οποίο χρησιμοποιήθηκαν οι εξής εντολές:

```
>> n=[1 3 5 7 9];
>> cn=abs(an)/2;
>> figure(3)
>> figure(4)
>> stem([-fliplr(n) 0 n],[fliplr(cn) a0 cn])
>> grid
>> xlabel('n')
>> ylabel('|c_{n}|')
>> title('Exponential Fourier series magnitude')
```

Παρατηρούμε ότι, όπως περιμέναμε από τη θεωρία, $A_n, c_n \to 0$ καθώς $n \to \infty$. Συνεπώς, μπορούμε να προσεγγίσουμε με όση ακρίβεια επιθυμούμε το σήμα μας με ένα *πεπερασμένο* άθροισμα Fourier:

$$x(t) = \sum_{n=0}^{N} A_n \cos\left(n\Omega_0 t - \phi_n\right)$$

Φυσικά, για να πάρουμε το σήμα *ακριβώς*, χρειάζεται να θεωρήσουμε το *άπειρο* άθροισμα. Θα δούμε πώς η προσέγγιση βελτιώνεται (και μάλιστα αρκετά γρήγορα λόγω της ταχείας μείωσης των A_n καθώς $n \to \infty$) αυξάνοντας τον αριθμό των αρμονικών συνιστωσών που συμπεριλαμβάνουμε στην άθροιση. Στο παράδειγμά μας, το αποτέλεσμα της πρόσθεσης στη συνιστώσα μηδενικής συχνότητας (a_0) (ή *DC* όπως αλλιώς λέγεται) των πρώτων 1, 2, 3 και 5 μη-μηδενικών αρμονικών φαίνεται στο ακόλουθο σχήμα.



Οι εντολές που το δημιούργησαν είναι (χρησιμοποιούμε υπάρχουσες από πριν μεταβλητές):

```
>> phi=[ph0 phn];

>> A=[a0 An];
```

```
>> n=[0 n];

>> W0=pi;

>> t=[-2:0.001:6];

>> x1=0;

>> for i=1:2, x1=x1+A(i)*cos(n(i)*W0*t-phi(i)); end

>> subplot(221)

>> plot(t,x1)

>> grid

>> xlabel('t')

>> title('DC plus 1 component')

>> x2=x1;

>> i=3;

>> x2=x2+A(i)*cos(n(i)*W0*t-phi(i));

>> subplot(222)

>> plot(t,x2)

>> grid

>> xlabel('t')

>> title('DC plus 2 components')

>> x3=x2;

>> i=4;
```

```
>> x3=x3+A(i)*cos(n(i)*W0*t-phi(i));

>> subplot(223)

>> plot(t,x3)

>> xlabel('t')

>> title('DC plus 3 components')

>> grid

>> x5=x3;

>> for i=5:6, x5=x5+A(i)*cos(n(i)*W0*t-phi(i)); end

>> subplot(224)

>> plot(t,x5)

>> grid

>> xlabel('t')

>> title('DC plus 5 components')
```

Βλέπουμε ότι γρήγορα φτάνουμε πολύ κοντά στο αρχικό μας σήμα. Αυτό οφείλεται στο ότι τα A_n μειώνονται πολύ γρήγορα (αντίστροφα του n^2) προς το μηδέν ή μ' άλλα λόγια, το φάσμα του σήματος είναι *ουσιαστικά*[1] μικρού εύρους.

Ένας εναλλακτικός τρόπος να δούμε ποσοτικά το πόσο γρήγορα παίρνουμε μια καλή προσέγγιση είναι να μετρήσουμε το ποσοστό της ισχύος του σήματος που περιέχεται στις αρμονικές που θεωρούμε. Αυτό μπορεί να γίνει με βάση το θεώρημα του Parseval για σειρές Fourier, σύμφωνα με το οποίο η ισχύς του σήματος ισούται με το άθροισμα των τετραγώνων των μέτρων των συντελεστών της εκθετικής σειράς Fourier:

$$P_x = \frac{1}{T}\int_{-T/2}^{T/2} |x(t)|^2 dt = \sum_{n=-\infty}^{\infty} |c_n|^2$$

Το πόσο καλά προσεγγίζεται το σήμα από το άθροισμα $\sum_{n=-N}^{N} c_n e^{jn\Omega_0 t}$ μπορεί να μετρηθεί από το ποσοστό της ολικής ισχύος που υπάρχει σ’ αυτό, δηλαδή $\left(\sum_{n=-N}^{N} |c_n|^2 \Big/ P_x\right) \times 100\%$. Για το παράδειγμά μας, $P_x = 0.3033$. Τα ποσοστά για τις τέσσερις περιπτώσεις που εξετάσαμε στο τελευταίο σχήμα είναι: 99.76%, 99.97%, 99.98%, και 99.99%.

Θα είχαμε μια πολύ διαφορετική κατάσταση για ένα σήμα με ευρύ φάσμα συχνοτήτων, όπως π.χ. συμβαίνει με τα σήματα που έχουν σημεία ασυνέχειας.

3.6 ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 6 – ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ FOURIER

Στην προηγούμενη εργαστηριακή άσκηση, είδαμε πως ένα περιοδικό σήμα μπορεί να αναπτυχθεί στο διάστημα $(-\infty, +\infty)$ σε μια σειρά Fourier, δηλαδή να παρασταθεί ως ένας γραμμικός συνδυασμός άπειρων αρμονικών εκθετικών σημάτων. Εδώ θα δούμε ότι τα αποτελέσματα αυτά μπορούν να επεκταθούν και για μη περιοδικά σήματα, στο διάστημα $(-\infty, +\infty)$. Ο τρόπος που μπορούμε να επιτύχουμε τα παραπάνω είναι με τη χρήση του **μετασχηματισμού Fourier.**

Ένα σήμα συνεχούς χρόνου $x(t)$ ορίζεται στο πεδίο του χρόνου t. Εφαρμόζοντας μετασχηματισμό Fourier (MF) θα «μεταφέρουμε» ή πιο απλά θα εκφράζουμε το σήμα στο πεδίο της κυκλικής συχνότητας Ω ή στο πεδίο της συχνότητας $f = \frac{\Omega}{2\pi}$.

2.6.1 Μαθηματικός ορισμός

Όπως αναφέραμε προηγουμένως, ο μετασχηματισμός Fourier μας μεταφέρει από το πεδίο του χρόνου t, στο πεδίο της κυκλικής συχνότητας Ω. Δηλαδή πλέον το σήμα (ή το σύστημα) που περιγράφεται από μια συνάρτηση $x(t)$ θα εκφράζεται από μια συνάρτηση $X(\Omega)$.

Συμβολικά το παραπάνω εκφράζεται ως:

$$X(\Omega) = F\{x(t)\} (2.6.1)$$

δηλαδή λέμε ότι ο μετασχηματισμός Fourier της $x(t)$ είναι $X(\Omega)$.

Εναλλακτικά :

$$x(t) \xrightarrow{F} X(\Omega). (2.6.2)$$

Έτσι ως μετασχηματισμό Fourier της $x(t)$ ονομάζουμε το μετασχηματισμό που εκφράζεται με τη σχέση:

$$F\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\Omega t} dt = \mathrm{X}(\Omega). (2.6.3)$$

Καταλαβαίνουμε ότι η συνάρτηση $X(\Omega)$ είναι μιγαδική ποσότητα. Στη περίπτωση που επιθυμούμε να εκφράσουμε το MF της $x(t)$ στο πεδίο της συχνότητας f έχουμε :

$$F\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt = \mathrm{X}(f). (2.6.4)$$

Προκείμενου να επιστρέφουμε από το πεδίο της κυκλικής συχνότητας Ω πίσω στο χρόνο t ορίζουμε τον αντίστροφο μετασχηματισμό Fourier . Ο **αντίστροφος μετασχηματισμός Fourier** (Inverse Fourier Transform) συμβολίζεται ως $F^{-1}\{.\}$, δηλαδή θα έχουμε:

$$x(t) = F^{-1}\{X(\Omega)\} .(2.6.5)$$

Ενναλακτικα γράφουμε: $\mathrm{X}(\Omega) \xrightarrow{F^{-1}} x(t).$

Η μαθηματική έκφραση για τον αντίστροφο μετασχηματισμό Fourier είναι:

$$F^{-1}\{X(\Omega)\} = x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\Omega)\cdot e^{j\Omega t}d\Omega. \quad (2.6.6)$$

Εάν χρησιμοποιήσουμε το f αντί του Ω:

$$F^{-1}\{X(f)\} = x(t) = \int_{-\infty}^{+\infty} X(f)\cdot e^{j2\pi ft}df \quad (2.6.7)$$

Μονάδα μέτρησης της κυκλικής συχνότητας Ω είναι το rad/s ενώ μονάδα μέτρησης της απλής συχνότητας f είναι το Hertz.

Ο MF ενός σήματος ονομάζεται και ***φάσμα*** **(*συχνοτήτων*).**

Πριν μελετήσουμε το πώς μπορεί ένας MF να υπολογιστεί στο MATLAB, αξίζει πρώτα να δούμε μέσω παραδειγμάτων τη φυσική σημασία του πολλαπλασιαστικού παράγοντα $H(\Omega)$.

Ας θυμηθούμε ότι η απόκριση του συστήματος σ' ένα μιγαδικό εκθετικό σήμα, $x(t) = e^{j\Omega_0 t}$, είναι $y(t) = H(\Omega_0)x(t)$

όπου $$H(\Omega_0) = \int_{-\infty}^{\infty} h(t)e^{-j\Omega_0 t}dt$$

Το παραπάνω μας λέει ότι τέτοια σήματα, που αποτελούνται δηλαδή από μια μόνο συχνότητα, Ω_0 (και λέγονται σήματα *απλής* συχνότητας), «περνούν» από το ΓΧΑ με μόνο τη μεταβολή στο πλάτος τους, το οποίο είναι πλέον πολλαπλασιασμένο με το (γενικά μιγαδικό) παράγοντα $H(\Omega_0)$.

Αυτό, σε συνδυασμό με το γεγονός ότι οποιοδήποτε σήμα $x(t)$ μπορεί να ειδωθεί ως άθροισμα σημάτων απλής συχνότητας, συνεπάγεται ότι με τη βοήθεια ενός ΓΧΑ μπορούμε να αλλοιώσουμε το πλάτος κάθε συχνότητας του $x(t)$.

Για <u>**παράδειγμα**</u>, αν πρέπει να μηδενιστεί η συχνότητα Ω_0 στο $x(t)$, αρκεί να το «περάσουμε» μέσα από ένα ΓΧΑ σύστημα με *απόκριση συχνότητας* $H(\Omega)$ τέτοια ώστε $H(\Omega_0) = 0$. Αν δεν θέλουμε να πειράξουμε τις άλλες συχνότητες, θα πρέπει $H(\Omega) = 1, \forall\Omega \neq \Omega_0$.

Γενικότερα, μπορούμε, με κατάλληλη επιλογή της μορφής της $H(\Omega)$ (άρα και της $h(t)$), να αλλοιώσουμε το MF του $x(t)$ (άρα και το ίδιο το $x(t)$) σύμφωνα με τις ανάγκες μας. Στο παραπάνω παράδειγμα, το ΓΧΑ σύστημα «αφήνει» να περάσουν απ' αυτό όλες οι συχνότητες εκτός μιας, που την «κόβει». Κάτι τέτοιο μας θυμίζει ένα φίλτρο, που σκοπό έχει να κατακρατήσει μέρος αυτού που του βάζουμε και ν' αφήσει να περάσει το υπόλοιπο. Σύμφωνα μ' αυτή την αναλογία, ένα ΓΧΑ σύστημα καλείται και *φίλτρο* (χωρίς απαραίτητα να «κόβει» κάποιες συχνότητες). Διακρίνουμε τέσσερα είδη φίλτρων:

1. **Κατωπερατό** (Lowpass): Περνάει χαμηλές συχνότητες και κόβει υψηλές
2. **Ανωπερατό** (Highpass): Περνάει υψηλές και κόβει χαμηλές συχνότητες
3. **Ζωνοπερατό** (Bandpass): Περνάει ένα διάστημα συχνοτήτων και κόβει όλες τις άλλες
4. **Ζώνης αποκοπής** (Bandstop): Κόβει ένα διάστημα συχνοτήτων και αφήνει όλες τις άλλες να περάσουν.

Ας δούμε ένα παράδειγμα φίλτρου ζώνης αποκοπής. Έχει την ακόλουθη απόκριση συχνότητας:

$$H(\Omega) = \frac{0.64 - \Omega^2}{(0.64 - \Omega^2) + j1.6\Omega} \cdot \frac{4 - \Omega^2}{(4 - \Omega^2) + j4\Omega}$$

ή γράφοντας την ως ρητή συνάρτηση του $j\Omega$,

$$H(\Omega) = \frac{0.64 + (j\Omega)^2}{0.64 + 1.6j\Omega + (j\Omega)^2} \cdot \frac{4 + (j\Omega)^2}{4 + 4j\Omega + (j\Omega)^2}.$$

Η παραπάνω μορφή της απόκρισης συχνότητας, δηλαδή το γινόμενο ρητών παραγόντων, είναι πολύ συχνή στην πράξη και υπαγορεύει μια υλοποίηση του αντίστοιχου φίλτρου ως σύνδεσης σε σειρά των δύο φίλτρων που αντιστοιχούν στους δύο παράγοντές της. Μια τέτοια υλοποίηση λέγεται ***σύνδεση σε σειρά* ή *cascade*.** Ο κάθε παράγοντας λέγεται και *βαθμίδα (stage)*.

Για να υπολογίσουμε την $H(\Omega)$ στο MATLAB, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **freqs**, που επιστρέφει την απόκριση συχνότητας δοσμένου *ρητού* συστήματος.

Πώς θα δώσουμε το σύστημα; Προσδιορίζοντας τους συντελεστές των πολυωνύμων ως προς $j\Omega$ στον αριθμητή και τον παρονομαστή του.

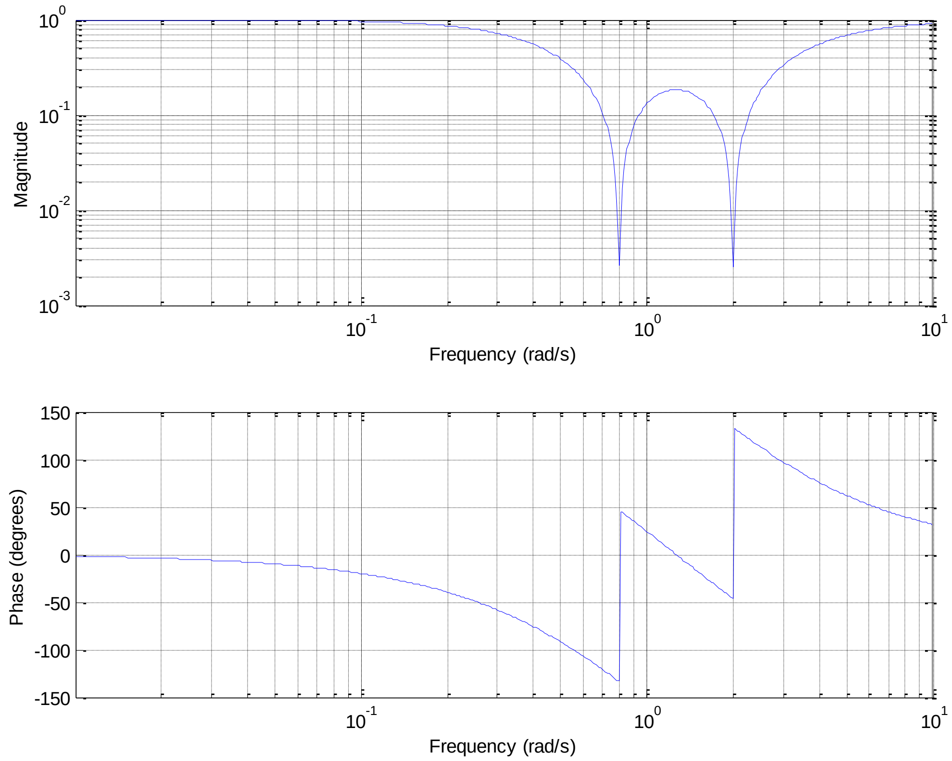
Έτσι, ο αριθμητής της πρώτης βαθμίδας έχει συντελεστές 1, 0 και 0.64, που πολλαπλασιάζουν το $(j\Omega)^2$, το $j\Omega$, και το $(j\Omega)^0$, αντίστοιχα. Παρόμοια για τα υπόλοιπα πολυώνυμα. Προκειμένου να βρούμε τον ολικό αριθμητή και τον ολικό παρονομαστή θα χρειαστούμε και τη συνάρτηση **conv** για να υλοποιήσουμε τον πολλαπλασιασμό κατά μέλη. Ας δούμε πώς:

```
>> num1=[1 0 0.64]; % Αριθμητής πρώτης βαθμίδας

>> num2=[1 0 4]; % Αριθμητής δεύτερης βαθμίδας

>> den1=[1 1.6 0.64]; % Παρονομαστής πρώτης βαθμίδας

>> den2=[1 4 4]; % Παρονομαστής δεύτερης βαθμίδας

>> num=conv(num1,num2); % Συνολικός αριθμητής

>> den=conv(den1,den2); % Συνολικός παρονομαστής

>> [H,w]=freqs(num,den,400); % Υπολογισμός της H(Ω)

                                        % σε 400 συχνότητες w
```

Αν καλέσουμε την **freqs** απλά ως

```
>> freqs(num,den,400)
```

χωρίς να της ζητήσουμε να επιστρέψει κάτι, τότε γίνεται αυτόματα η γραφική παράσταση του μέτρου και της φάσης (σε μοίρες) της $H(\Omega)$, όπως φαίνεται στο

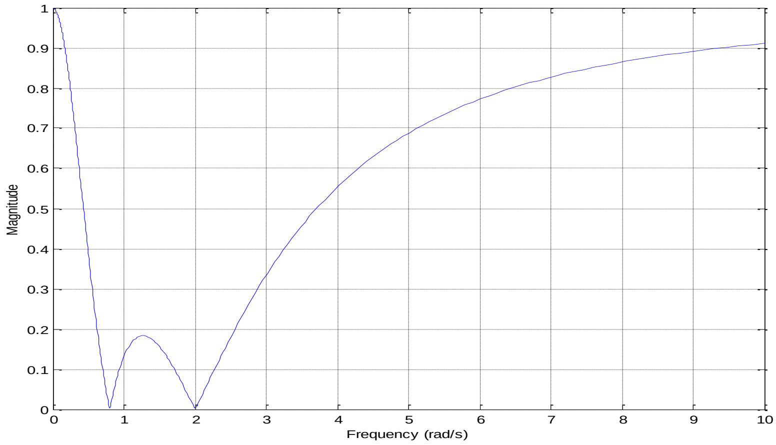


παρακάτω σχήμα

Η συχνότητα είναι σε **ακτίνια/δευτερόλεπτο** (rad/s) και δίνεται σε λογαριθμική κλίμακα. Αν θέλουμε π.χ. μόνο το μέτρο, και μάλιστα με τη συχνότητα σε γραμμική κλίμακα, τότε μπορούμε να το παραστήσουμε με τις εντολές

```
>> plot(w,abs(H))
>> grid
>> xlabel('Frequency (rad/s)')
>> ylabel('Magnitude')
```

Το αποτέλεσμα φαίνεται παρακάτω:



Όπως βλέπουμε, η **ζώνη αποκοπής** βρίσκεται **μεταξύ των 0.8 και των 2 rad/s.** Επίσης, όχι μόνο αυτές αλλά και γειτονικές τους συχνότητες εξασθενίζουν μ' αυτό το φίλτρο. Κάτι τέτοιο είναι αναπόφευκτο για φίλτρα που είναι πρακτικά υλοποιήσιμα.

Ένας δεύτερος τρόπος υλοποίησης είναι ο εξής :

```
>>  w=0:0.1:10                    % ορισμός διανύσματος συχνοτήτων
>> H=((0.64+(j*w).^2).*(4+(j*w).^2)) ./ (  (0.64 +1.6*j*w+(j*w).^2) .*  (4+4*j*w + (j*w).^2)) ;   % ορισμός της απόκρισης συχνότητας H(jω)
>>  plot(w,abs(H)) % σχεδίαση μέτρου
>> plot(w,angle(H) % σχεδίαση φάσης
```

Έχοντας υπολογίσει την απόκριση συχνότητας σ' ένα διάστημα συχνοτήτων, μπορούμε να δούμε πια την απόκριση του φίλτρου σε διάφορα σήματα απλής συχνότητας.

- Αν το σήμα εισόδου είναι γραμμικός συνδυασμός σημάτων απλής συχνότητας, στις συχνότητες $\Omega_0, \Omega_1, \Omega_2, \ldots$, δηλαδή $x(t) = A_0 e^{j\Omega_0 t} + A_1 e^{j\Omega_1 t} + A_2 e^{j\Omega_0 t} + \cdots$, τότε η έξοδος του φίλτρου θα είναι ο ίδιος γραμμικός συνδυασμός των αντίστοιχων αποκρίσεων:

$$y(t) = H(\Omega_0)A_0 e^{j\Omega_0 t} + H(\Omega_1)A_1 e^{j\Omega_1 t} + H(\Omega_2)A_2 e^{j\Omega_0 t} + \ldots$$

Γενικά ένα σήμα μπορεί να γραφεί έτσι (ίσως με άπειρες συχνότητες) όπου τα πλάτη A_i συνδέονται με το MF του. Έτσι καταλήγουμε στη γνωστή μας θεμελιώδη σχέση

$$Y(\Omega) = H(\Omega)X(\Omega)$$

Για να τη χρησιμοποιήσουμε στην πράξη, θα πρέπει να ξέρουμε να υπολογίζουμε το MF ενός σήματος συνεχούς χρόνου.

Η ακόλουθη function υπολογίζει το MF με τον παραπάνω τρόπο, στο διάστημα συχνοτήτων $\left[-\frac{\pi}{T}, \frac{\pi}{T}\right]$. Η ιδιαιτερότητά της είναι ότι το σήμα δίνεται μέσω της συνάρτησης MATLAB που το υλοποιεί. Αυτή θα πρέπει να έχει συγκεκριμένο τρόπο κλήσης.

Δημιουργήστε ένα νέο **m-file** που θα το ονομάσετε **ctft.m** (δηλαδή Μετασχηματισμός Fourier συνεχούς χρόνου) και θα το αποθηκεύσετε στον κατάλογο work του Matlab .

```
function [X,w]=ctft(x)

% CTFT  Continuous-time Fourier transform

%

% Usage: [X,w]=ctft(x)

% x must be a string containing the call to the function

% computing the signal.

% Call syntax: [y,T,N,t1]=func(parameters).
```

```
x1=['[y,T,N,t1]=',x,';'];

eval(x1); % Υπολογισμός της έκφρασης στο string x1

t2=t1+(N-1)*T;

t=t1:T:t2;

W=2*pi/(N*T);

w=[-(N/2)+1:(N/2)]*W;

X=T*exp(-j*w*t1).*fft(y);

subplot(221)

plot(t,y)

title('Signal')

subplot(223)

plot(w,abs(fftshift(X)))

title('Fourier transform magnitude')

subplot(224)

plot(w,angle(fftshift(X)))

title('Fourier transform phase')
```

Επίσης δημιουργήστε ένα **m-file** που θα το ονομάσετε **cosine.m** (δηλ. συνημίτονο) και θα το αποθηκεύσετε στον κατάλογο work του Matlab .

Για το σήμα $x(t) = \cos(\Omega_0 t)$ η συνάρτηση cosine θα είναι:

```
function [x,T,N,t1]=cosine(w0,t1,t2,N)
```

```
% COSINE    Computes cos(w0*t) for given w0

% Usage: [x,T,N,t1]=cosine(w0,t1,t2,N)

% Παρατηρείστε ότι πρέπει να ξαναεπιστρέφεται το N και το t1

% ώστε να είναι διαθέσιμα στην ctft.

T=(t2-t1)/N;

t=t1:T:t1+(N-1)*T;

x=cos(w0*t);
```

Πληκτρολογήστε στο Matlab

>> **[X,w]=ctft('cosine(5,0,15,300)');**

Το Matlab δίνει τη δυνατότητα να βρούμε απευθείας το μετασχηματισμό Fourier μιας συνάρτησης με την εντολή **fourier ( ).**

Το Matlab επίσης δίνει τη δυνατότητα να υπολογίσουμε και τον αντίστροφο μετασχηματισμό Fourier μιας συνάρτησης με την εντολή **ifourier** ( ).

Σύνταξη εντολής fourier:

Σειρά fourier = fourier (συνάρτηση)

Σύνταξη εντολής ifourier:

αντίστροφος μετασχηματισμός fourier = ifourier (μετασχηματισμένη fourier)

Παράδειγμα Να βρεθεί ο μετασχηματισμός Fourier της συνάρτησης $f(t)=e\text{-}x^2$ και στη συνέχεια ο αντίστροφος μετασχηματισμός Fourier.

Ο απαιτούμενος κώδικας είναι:

```
>> syms x

>> f=exp(-x^2);

>> fourier(f)

>> ifourier(ans)
```

ΠΑΡΑΡΤΗΜΑ 1

ΜΕΡΙΚΕΣ ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΤΟΥ SIGNAL PROCESSING TOOLBOX

- **FILTER** One-dimensional digital filter.

Y = FILTER(B,A,X) filters the data in vector X with the

filter described by vectors A and B to create the filtered

data Y.

a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)

- a(2)*y(n-1) - ... - a(na+1)*y(n-na)

- **PLOT** Linear plot.

PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,

then the vector is plotted versus the rows or columns of the matrix,

whichever line up. If X is a scalar and Y is a vector, length(Y)

disconnected points are plotted.

- **FREQZ** Digital filter frequency response.

[H,W] = FREQZ(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter:

jw -jw -jmw

jw B(e) b(1) + b(2)e + ... + b(m+1)e

H(e) = ---- = ------------------------------------

jw -jw -jnw

A(e) a(1) + a(2)e + ... + a(n+1)e

given numerator and denominator coefficients in vectors B and A. The frequency response is evaluated at N points equally spaced around the upper half of the unit circle. If N isn't specified, it defaults to 512

H = FREQZ(B,A,W) returns the frequency response at frequencies designated in vector W, in radians/sample (normally between 0 and pi).

- **ONES** Ones array.

ONES(N) is an N-by-N matrix of ones.

ONES(M,N) or ONES([M,N]) is an M-by-N matrix of ones.

- **STEM** Discrete sequence or "stem" plot.

STEM(Y) plots the data sequence Y as stems from the x axis terminated with circles for the data value.

STEM(X,Y) plots the data sequence Y at the values specified in X.

- **AXIS** Control axis scaling and appearance.

AXIS([XMIN XMAX YMIN YMAX]) sets scaling for the x- and y-axes on the current plot.

- **CONV** Convolution and polynomial multiplication.

C = CONV(A, B) convolves vectors A and B. The resulting

vector is length LENGTH(A)+LENGTH(B)-1.

- **RESIDUEZ** Z-transform partial-fraction expansion.

[R,P,K] = RESIDUEZ(B,A) finds the residues, poles and direct terms

of the partial-fraction expansion of B(z)/A(z),

```
        R(j)          R(j+1)              R(j+1-1)
  -------------- + ------------------  + ... + ------------------
  1 - P(j)z^(-1)   (1 - P(j)z^(-1))^2         (1 - P(j)z^(-1))^m
```

- **ABS** Absolute value.

ABS(X) is the absolute value of the elements of X. When

X is complex, ABS(X) is the complex modulus (magnitude) of

the elements of X.

- **ROOTS** Find polynomial roots.

ROOTS(C) computes the roots of the polynomial whose coefficients

are the elements of the vector C. If C has N+1 components,

the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).

- **ANGLE** Phase angle.

ANGLE(H) returns the phase angles, in radians, of a matrix with

complex elements.

- **FFT** Discrete Fourier transform.

FFT(X) is the discrete Fourier transform (DFT) of vector X. For matrices, the FFT operation is applied to each column. For N-D arrays, the FFT operation operates on the first non-singleton dimension.

FFT(X,N) is the N-point FFT, padded with zeros if X has less than N points and truncated if it has more.

- **SIGSHIFT** implements y(n) = x(n-n0)

  [y,n] = sigshift(x,m,n0)

- **SIGADD** implements y(n) = x1(n)+x2(n)

  [y,n] = sigadd(x1,n1,x2,n2)

  y = sum sequence over n, which includes n1 and n2

  x1 = first sequence over n1

  x2 = second sequence over n2 (n2 can be different from n1)

- **SIGFOLD** implements y(n) = x(-n)

  [y,n] = sigfold(x,n)

- **IMPZ** Impulse response of digital filter

[H,T] = IMPZ(B,A) computes the impulse response of the filter B/A choosing the number of samples for you, and returns the response in

column vector H and a vector of times (or sample intervals) in T

(T = [0 1 2 ...]').

[H,T] = IMPZ(B,A,N) computes N samples of the impulse response.

If N is a vector of integers, the impulse response is computed

only at those integer values (0 is the origin).

- **IFFT** Inverse discrete Fourier transform.

IFFT(X) is the inverse discrete Fourier transform of X.

IFFT(X,N) is the N-point inverse transform.

- **LENGTH** Length of vector.

LENGTH(X) returns the length of vector X.

- **STEPSEQ** Generates x(n) = u(n-n0); n1 <= n,n0 <= n2

[x,n] = stepseq(n0,n1,n2)

- **IMPSEQ** Generates x(n) = delta(n-n0); n1 <= n,n0 <= n2

[x,n] = ipseq(n0,n1,n2)

- **EVENODD** Real signal decomposition into even and odd parts

----

[xe, xo, m] = evenodd(x,n)

- **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.

- **ZP2TF** Zero-pole to transfer function conversion.

[NUM,DEN] = ZP2TF(Z,P,K) forms the transfer function:

```
        NUM(s)
H(s) = --------
        DEN(s)
```

- **XLABEL** XLABEL X-axis label.

XLABEL('text') adds text beside the X-axis on the current axis.

- **ZEROS** Zeros array.

ZEROS(N) is an N-by-N matrix of zeros.

ZEROS(M,N) or ZEROS([M,N]) is an M-by-N matrix of zeros.

ΠΑΡΑΡΤΗΜΑ 2

ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΜΕ OCTAVE

Π.2.1 ΕΙΣΑΓΩΓΗ

Τη δεκαετία του 80 κυκλοφόρησε ελεύθερα (public domain) στο Διαδίκτυο ένα πακέτο από FORTRAN υπορουτίνες για εύκολες και γρήγορες πράξεις με πίνακες σε υπολογιστή. Διάφορες εταιρίες βελτίωσαν και αύξησαν τις δυνατότητες του αρχικού πακέτου με την προσθήκη γραφικών και βιβλιοθηκών εξειδικευμένων συναρτήσεων χρήσιμων σε πολλούς τομείς εφαρμοσμένων επιστημών.

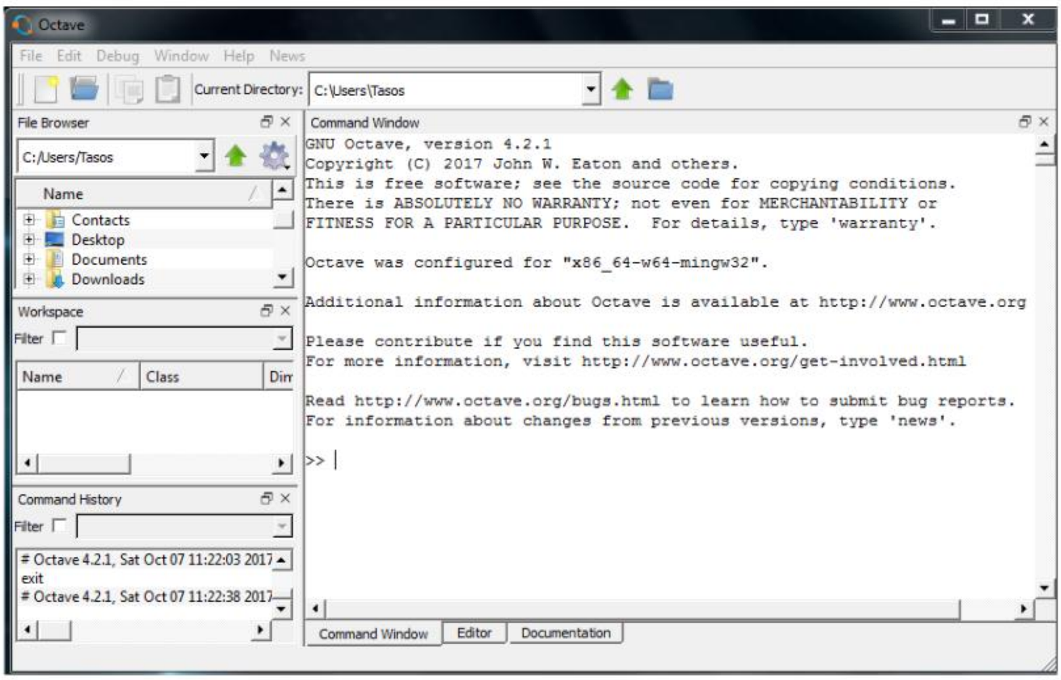
Η πιο επιτυχημένη εταιρία ήταν και είναι η Mathworks (http://www.mathworks.com) και το προϊόν της το MATLAB χρησιμοποιείται σήμερα διεθνώς σε πολλά πανεπιστήμια, ερευνητικά κέντρα και εταιρίες που ασχολούνται με την έρευνα και την τεχνολογία.

Την ίδια δεκαετία ξεκίνησε και το ρεύμα του Ανοικτού Λογισμικού (Open Source) που με τη βοήθεια του Διαδικτύου έδωσε την ευκαιρία σε πολλούς νέους ερευνητές να συνεργαστούν εθελοντικά και να δημιουργήσουν πολλές χρήσιμες εφαρμογές (π.χ. GCC C και C++ compilers και εργαλεία, λειτουργικό Linux, κλπ.).

Το MATLAB είναι αναμφισβήτητα ένα από τα πιο χρήσιμα εργαλεία που μαθαίνει ένας μηχανικός/επιστήμονας και σε όλα σχεδόν τα πανεπιστήμια των τεχνολογικά προηγμένων χωρών του εξωτερικού επιβάλλεται η διδασκαλία του από

την αρχή των σπουδών του. Εκτός όμως από το εμπορικό πακέτο MATLAB που η πλήρης έκδοση έχει αρκετά υψηλό κόστος, ερευνητές που χρησιμοποιούν εργαλεία ανοικτού λογισμικού δημιούργησαν άλλα πακέτα/εφαρμογές με παρόμοιες δυνατότητες, που έχουν το πλεονέκτημα να είναι νομίμως δωρεάν. Ένα από αυτά, το **Octave** (https://www.gnu.org/software/octave), από τα Πανεπιστήμια Wisconsin-Madison και Texas της Αμερικής, έχει σχεδιαστεί με στόχο την όσο το δυνατόν πλήρη συμβατότητα με το MATLAB. Διαθέτει πολλές από τις βασικές δυνατότητες του MATLAB και βρίσκεται σε συνεχή ανάπτυξη για περαιτέρω βελτίωση.

Η OCTAVE αναπτύχθηκε με σκοπό να αποτελέσει το «MATLAB του φτωχού» και προσφέρεται δωρεάν στα πλαίσια του ακαδημαϊκού πνεύματος του οργανισμού GNU1 που προωθεί το απεριόριστα ελεύθερο λογισμικό ανοικτού κώδικα. Αν και η υπολογιστική της μηχανή και επιδόσεις είναι η ίδια με του MATLAB (οι βιβλιοθήκες LAPACK και BLAS διανέμονται δωρεάν από την κυβέρνηση των Η-ΠΑ), τα δύο συστήματα δεν συγκρίνονται όσον αφορά το επίπεδο του περιβάλλοντος εργασίας και των προσφερομένων ευκολιών, γραφικών δυνατοτήτων, προγραμματιστικών εργαλείων, προγραμματιστικών συντομεύσεων και τεχνικής υποστήριξης, με ασυζήτητη την υπεροχή του MATLAB.



Σχήμα Π.2.1: Octave GUI περιβάλλον για έκδοση 4.2.1.

Π.2.2 MATLAB & OCTAVE

Σε γενικές γραμμές τα προγράμματα απαρτίζονται από πέντε κύριες ενότητες:

1. **Περιβάλλον εργασίας και ανάπτυξης εφαρμογών:** Πρόκειται για το σύνολο των εργαλείων που διευκολύνουν στην απρόσκοπτη χρήση των υπολογιστικών μηχανών. Στο MATLAB πολλά από αυτά τα εργαλεία είναι γραφικά (*Graphical User Interfaces- GUIs*) και περιλαμβάνουν το *περιβάλλον εργασίας* (*desktop*), το *Παράθυρο Εντολών* (*Command Window*), *αρχείο εντολών*, ένα *συντάκτη* (*editor*) και *ελεγκτή* (*debugger*) προγραμμάτων και «*περιηγητές*» (*browsers*) βοήθειας, του χώρου εργασίας και των αρχειο καταλόγου εργασίας. Στην OCTAVE οι δυνατότητες αυτές είναι μάλλον φτωχές. Υπάρχει μόνο το παράθυρο εντολών και περιορισμένες δυνατότητες βοήθειας.

2. **Βιβλιοθήκες Μαθηματικών Συναρτήσεων και Προγραμμάτων:** Πρόκειται για εκτεταμένη συλλογή υπολογιστικών αλγορίθμων, από στοιχειώδεις συναρτήσεις (π.χ. άθροισμα, ημίτονο, συνημί-τονο, μιγαδική αριθμητική κ.ά.), μέχρι εξαιρετικά περίπλοκες λειτουργίες, όπως γενικευμένη αντιστροφή πινάκων, ιδιοτιμές/ ιδιοδιανύσματα, ειδικές συναρτήσεις (π.χ. Bessel, Neumann) και μετασχηματισμούς Fourier.

3. **Γλώσσα προγραμματισμού:** Είναι σχεδόν πανομοιότυπη και για τα δύο συστήματα αλλά επειδή πρώτα αναπτύχθηκε ως μέρος του MATLAB θα την αποκαλούμε «γλώσσα MATLAB». Πρόκειται για γλώσσα υψηλού επιπέδου,2 με πλήρη σειρά εντολών ελέγχου ροής, συναρτήσεις, δομές, εντολές εισό-δου/ εξόδου και στοιχεία αντικειμενικού προγραμματισμού. Είναι εύχρηστη τόσο για «μικροπρογραμματισμό», δηλαδή ταχεία κατασκευή όχι και τόσο έξυπνα δομημένων προγραμμάτων ειδικού σκοπού, όσο και για «μακροπρογραμματισμό», δηλαδή συστηματική ανάπτυξη μεγάλων και πολύπλοκων εφαρμογών.

4. **Γραφικά:** Το MATLAB προσφέρει εξαιρετικές δυνατότητες γραφικής αναπαράστασης ανυσμάτων και πινάκων. Περιλαμβάνει GUI και συναρτήσεις για διδιάστατη και τριδιάστατη αναπαράσταση, επεξεργασία ειδώλων/ εικόνας, κατασκευή κινούμενης εικόνας (video) και παρουσίαση αποτελεσμάτων. Η OCTAVE βασίζεται κυρίως στο πακέτο GNUPLOT (το οποίο στην έκδοση για MS Windows έχει ενσωματώσει), και το οποίο αν και υπό όρους καλύπτει τις κοινότερες ανάγκες, υπολείπεται κατά πολύ της αρτιότητας και δυνατοτήτων του MATLAB. Το MATLAB και η OCTAVE παρέχουν ευρεία ποικιλία τεχνικών γραφικής παρουσίασης δεδομένων και αποτελεσμάτων, εκτύπωσης των γραφημάτων και εξαγωγής τους σε τυπικά αρχεία γραφικών για ηλεκτρονική παρουσίαση. Γενικά, η γραφική παρουσίαση των δεδομένων περιλαμβάνει μία σειρά πράξεων, οι λεπτομέρειες των οποίων διαφέρουν κατά περίπτωση. Αυτό οφείλεται εν μέρει στο γεγονός ότι ο επιλεγόμενος τύπος γραφήματος εξαρτάται από την φύση των δεδομένων και την πληροφορία που πρέπει να εξαχθεί από αυτά και να καταδειχθεί. Σε γενικές γραμμές παρέχονται αρκετοί προ-προγραμματισμένοι τύποι γραφημάτων, από συμβατικές γραφικές παραστάσεις, ιστογράμματα και pie charts, μέχρι τριδιάστατες απεικονίσεις επιφανειών. Ο βασικός τρόπος κατασκευής γραφημάτων είναι άμεσες εντολές μέσω του παραθύρου εντολών, ή με κατασκευή γραφικών προγραμμάτων μέσω της γλώσσας MALTAB. Στο παρόν κεφάλαιο θα δοθεί μόνο μία γενική εικόνα της διαδικασίας κατασκευής γραφημάτων, με παραδείγματα και λεπτομέρειες για την εκτέλεση των κυριοτέρων πράξεων.

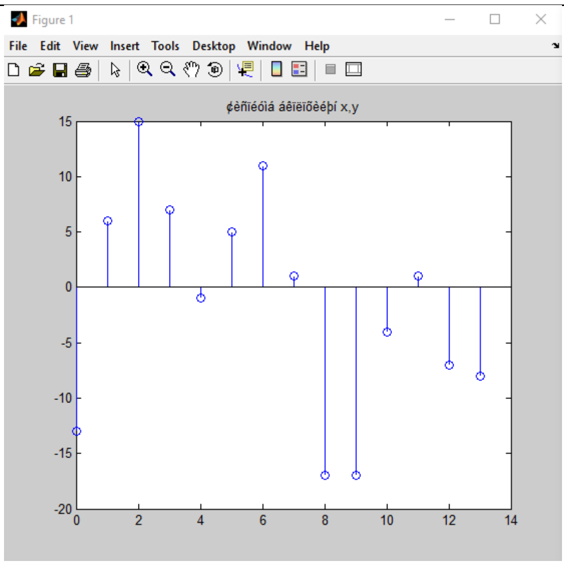
5. **Εξωτερικές διεπαφές/API**: Πρόκειται για βιβλιοθήκη που επιτρέπει στον χρήστη την συγγραφή και χρήση προγραμμάτων C και FORTRAN που αλληλεπιδρούν με το MATLAB ή την OCTAVE. Πε-ριλαμβάνει διευκολύνσεις για την κλήση συναρτήσεων και διαδικασιών των υπολογιστικών μηχανών από τα προγράμματα του χρήστη, παρέχοντας έτσι σχεδόν απεριόριστες δυνατότητες.

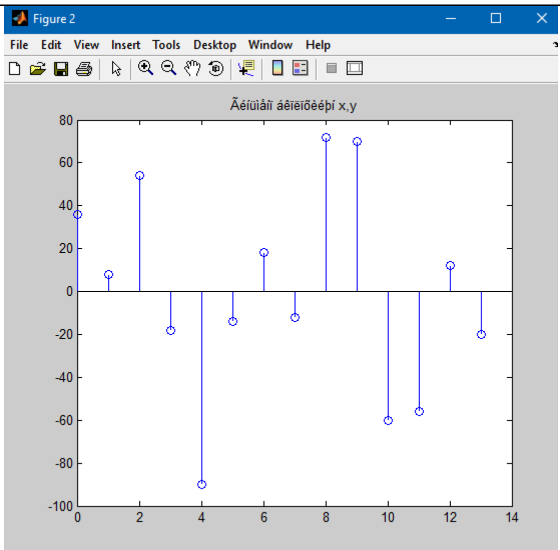
Π.2.3 ΠΡΟΤΕΙΝΟΜΕΝΕΣ ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ OCTAVE

Π.2.3.1 Εισαγωγή στα Διακριτά Σήματα

1. Να σχεδιαστούν με τη βοήθεια των εντολών της MATLAB και OCTAVE το άθροισμα και το γινόμενο των ακολουθιών x(n) και y(n). Ποιες οι μέγιστες και οι ελάχιστες τιμές του αθροίσματος και του γινομένου; Οι ακολουθίες x(n) και y(n) παράγονται με τη βοήθεια της συνάρτησης **[x,y]=sig_gen(A.M.)** όπου Α.Μ. ο πενταψήφιος αριθμός μητρώου σας. (π.χ. Αν ο αριθμός μητρώου σας είναι 12345 τότε [x,y]=sig_gen(12345)

```
n = 0:10;
[x,y] = sig_gen(39131); %Δημιουργία ακολουθιών X,Y
a = length(x); %Πλήθος όρων X
b = length(y); %Πλήθος όρων Y
nx = 0:1:a-1; %Χρόνος X
ny = 0:1:b-1; %Χρόνος Y
[athroisma,n] = sigadd(x,nx,y,ny); %Άθροισμα ακολουθιών
[ginomeno,n] = sigmult(x,nx,y,ny); %Γινόμενο ακολουθιών
figure;stem(n,athroisma)
title('Άθροισμα ακολουθιών x,y')
figure;stem(n,ginomeno)
title('Γινόμενο ακολουθιών x,y')
Athrmax= max(athroisma) %Μέγιστο αθροίσματος
Athrmin = min(athroisma) %Ελάχιστο αθροίσματος
Ginmax = max(ginomeno) %Μέγιστο γινομένου
Ginmin = min(ginomeno) %Ελάχιστο γινομένου
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE
Athrmax = 15	Athrmax = 15
Athrmin = -17	Athrmin = -17
Ginmax = 72	Ginmax = 72
Ginmin = -90	Ginmin = -90
	



2. Για την ακολουθία: z(n)=5*x(n-2)-4*x(n-2)-3*x(-n+2)

   α. να υπολογισθεί το πλήθος των όρων της ακολουθίας z(n),

   β. να σχεδιαστεί η ακολουθία z(n),

   γ. να σχεδιαστούν το άρτιο και περιττό μέρος της ακολουθίας z(n+2).

```
[x,y] = sig_gen(39131); %Δημιουργία ακολουθιών X,Y
a = length(x); %Πλήθος όρων X
b = length(y); %Πλήθος όρων Y
nx = 0:1:a-1; %Χρόνος X
ny = 0:1:b-1; %Χρόνος Y
[x1,n1] = sigshift(x,nx,2) %Δημιουργία x(n-2)
[x2,n2] = sigshift(x,nx,2) %Δημιουργία x(n-2)
[x3,n3] = sigshift(x,-nx,-2) %Δημιουργία x(-n+2)
[z1,nz1] = sigadd(5*x1,n1,-4*x2,n2); %Άθροισμα των 2 πρώτων σκελών
[z,nz] = sigadd(z1,nz1,-3*x3,n3); %Τελικό άθροισμα
figure;stem(nz,z)
title('Ακολουθία Z')
length(z) %Πλήθος όρων της Z
```

```
[z2,nz2] = sigshift(z,nz,-2); %Δημιουργία της z(n+2)
[Zeven,Zodd,m] = evenodd(z2, nz2); %Χώρισμα σε περιττά και άρτια μέρη
figure;stem(m,Zeven)
title('Even Z')
figure;stem(m,Zodd)
title('Odd Z')
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE
x1 = -9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10 n1 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 x2 = -9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10	x1 = -9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10 n1 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 x2 = -9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4

n2 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 x3 = -9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10 n3 = -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15	-10 n2 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 x3 = -9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10 n3 = -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15

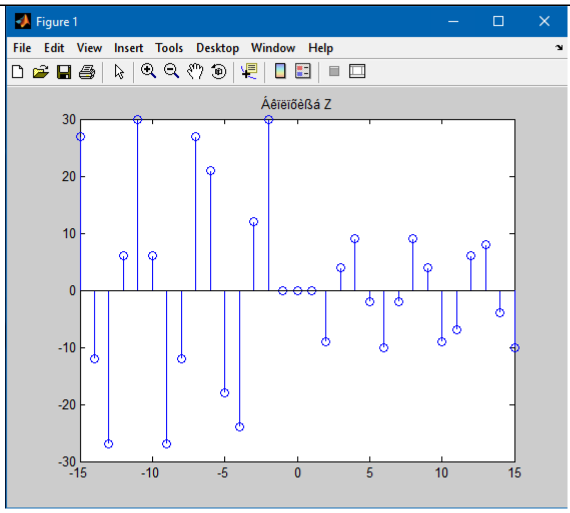


Figure 1
File Edit View Insert Tools Desktop Window Help
Áêïëïõèßá Z

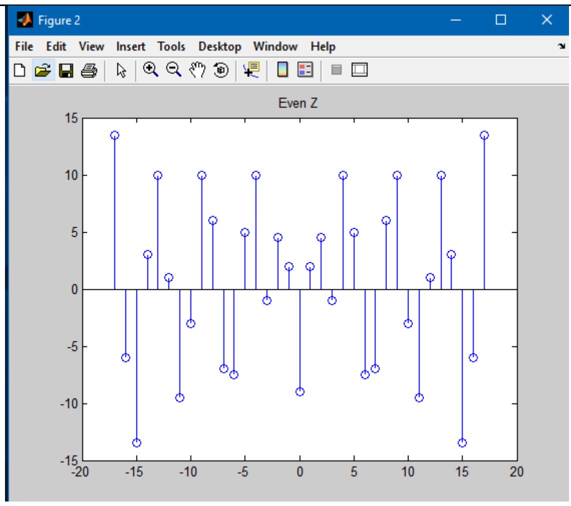


Figure 2
File Edit View Insert Tools Desktop Window Help
Even Z

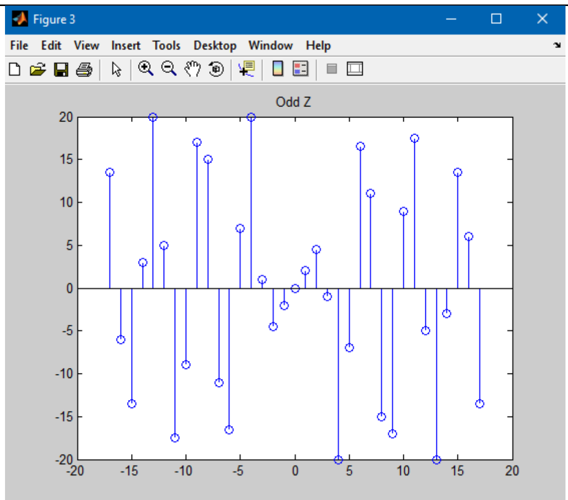


Figure 3
File Edit View Insert Tools Desktop Window Help
Odd Z

Π.2.3.2 Δημιουργία Διακριτών Σημάτων με User Interface

1. Θεωρήστε το σήμα συνεχούς χρόνου:

x(t)=2sin(3*(f1+f2)*π*t)+3cos (2*(f3+f2)*π*t-45^{0}) οπού f_1, f_2 και f_3 συχνότητες που δίνονται από τη γεννήτρια όπως περιγράφεται παρακάτω:

a. Υπάρχει κάποια από τις συχνότητες fess οι οποίες παράγονται από τη γεννήτρια η οποία μπορεί να χρησιμοποιηθεί για τη σωστή δειγματοληψία του σήματος; Αν ναι χρησιμοποιήστε την για να δειγματοληπτήσετε το σήμα στο χρονικό διάστημα $t \in [0s, 2s]$. Αν καμιά από τις παραγόμενες fs δεν είναι κατάλληλη υπολογίστε εσείς την ελάχιστη κατάλληλη συχνότητα δειγματοληψίας και δειγματοληπτήστε το σχήμα στο παραπάνω χρονικό διάστημα.

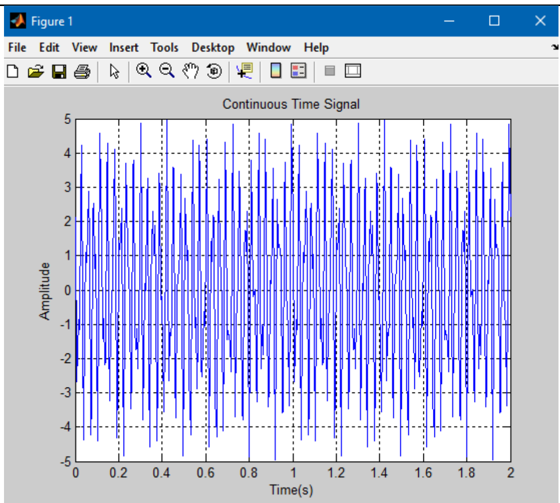
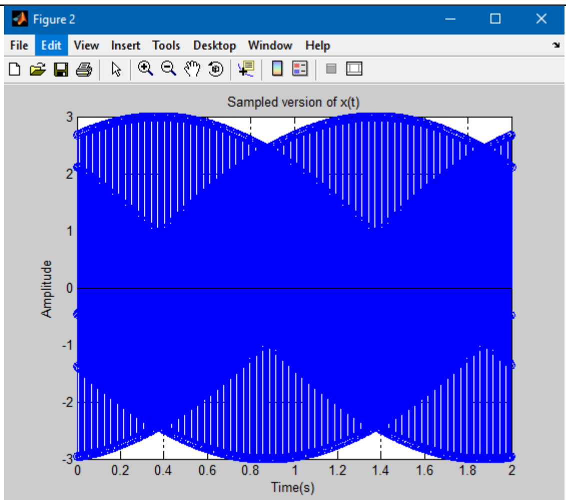
b. Σχεδιάστε το δειγματοληπτημένο σήμα με χρήση εντολών MATLAB και με χρήση του περιβάλλοντος Mastering DSP.

Οι συχνότητες του σήματος f_1, f_2 και f_3 (πινάκας f) καθώς και οι συχνότητες δειγματοληψίας fs1, fs2 και fs3 (πίνακας fs) παράγονται με τη βοήθεια της συνάρτησης [f,fs]=freq_gen(AM) οπού A.M. ο πενταψήφιος αριθμός μητρώου σας. (π.χ. Αν ο αριθμός μητρώου σας είναι 12345 τότε [x,y]=freq_gen(12345)).

```
[f,fs] = freq_gen(39131)
fs = 918; %Επιθυμητή συχνότητα δειγματοληψίας
t = [0:0.005:2];
ts = 1/fs; %Περίοδος δειγματοληψίας
x = 2*cos(2*(459)*pi*t+(pi/2))+3*cos(2*(367)*pi*t-(pi/4));
plot(t,x),grid,xlabel('Time(s)'),ylabel('Amplitude'),title('Continuous Time Signal')
%Σήμα συνεχούς χρόνου
tt = [0:ts:2];
```

```
xs = 2*cos(2*(459)*pi*tt+(pi/2))+3*cos(2*(367)*pi*tt-(pi/4));
% Δειγματοληψία αναλογικού σήματος
figure;stem(tt,xs);
grid,xlabel('Time(s)'),ylabel('Amplitude'),title('Sampled version of x(t)')
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE
f = 123 181 601 fs = 133 362 1304 f = 181 601 123 fs =	f = 123 181 601 fs = 133 362 1304 f = 181 601 123 fs =

<table>
<tr><td>362 1304 133</td><td>362 1304 133</td></tr>
<tr><td>
</td><td></td></tr>
<tr><td>
</td><td></td></tr>
</table>

Π.2.3.3 Συνέλιξη - Κρουστική Απόκριση - Υπολογισμός Απόκρισης Φίλτρου

1. Δίνεται η εξίσωση διαφορών:

$$A \cdot y(n) + 2 \cdot \Delta \cdot y(n-1) + B \cdot y(n-3) = (\mathrm{A}+\mathrm{B}) \cdot x(n) - \Gamma \cdot x(n-2) - \mathrm{E} \cdot x(n-3)$$

όπου ΑΒΓΔΕ: τα ψηφία του Α.Μ..

Παράδειγμα: Αν Α.Μ. 32104 Α=3, Β=2, Γ=1, Δ=0 & Ε=4

Να γραφεί κώδικας MATLAB ο οποίος θα απαντά στα παρακάτω ερωτήματα.

c. Να υπολογίζει και να εκτυπώνει την βηματική απόκριση του συστήματος στο χρονικό διάστημα n=0:30.

d. Να υπολογίζει και να εκτυπώνει την απόκριση του συστήματος για το παραπάνω χρονικό διάστημα αν ως είσοδος δοθεί η συνάρτηση $x(n)=0.8^{n}\cdot\cos(4\cdot 10^{5}\cdot n)$

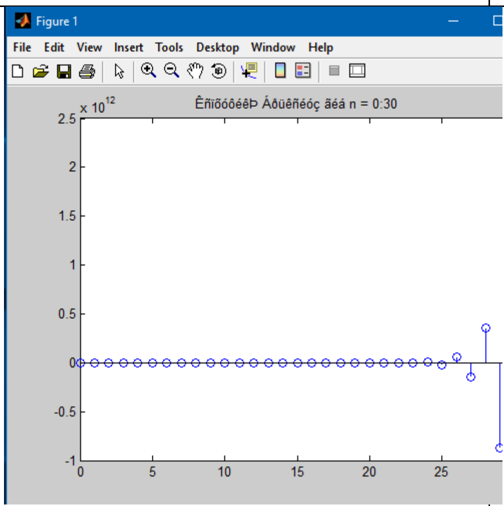
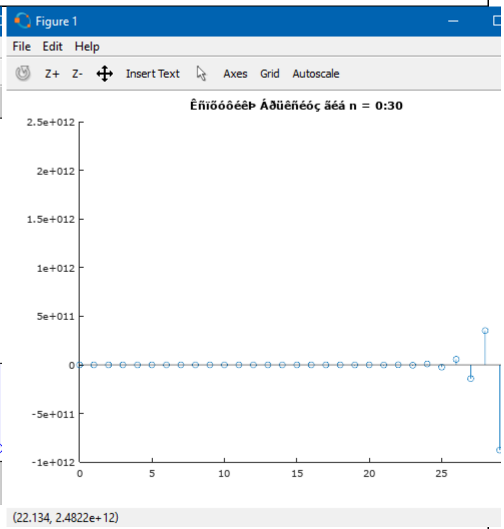
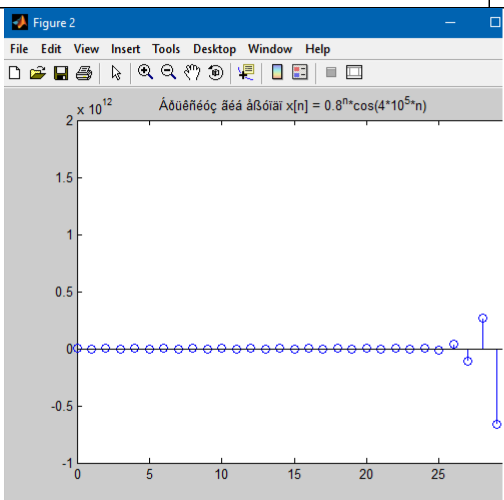
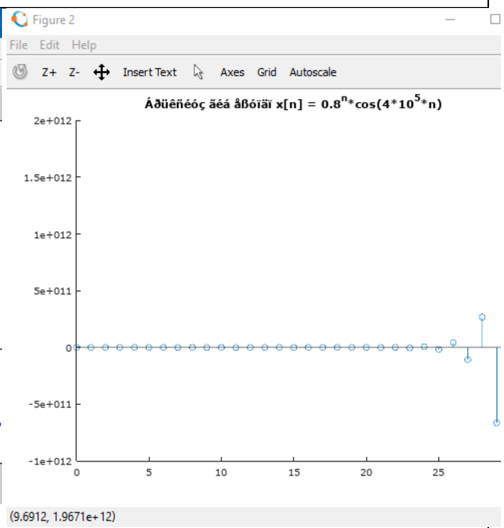
e. Να υπολογίζει και να εκτυπώνει την βηματική απόκριση με τη χρήση βρόγχου for.

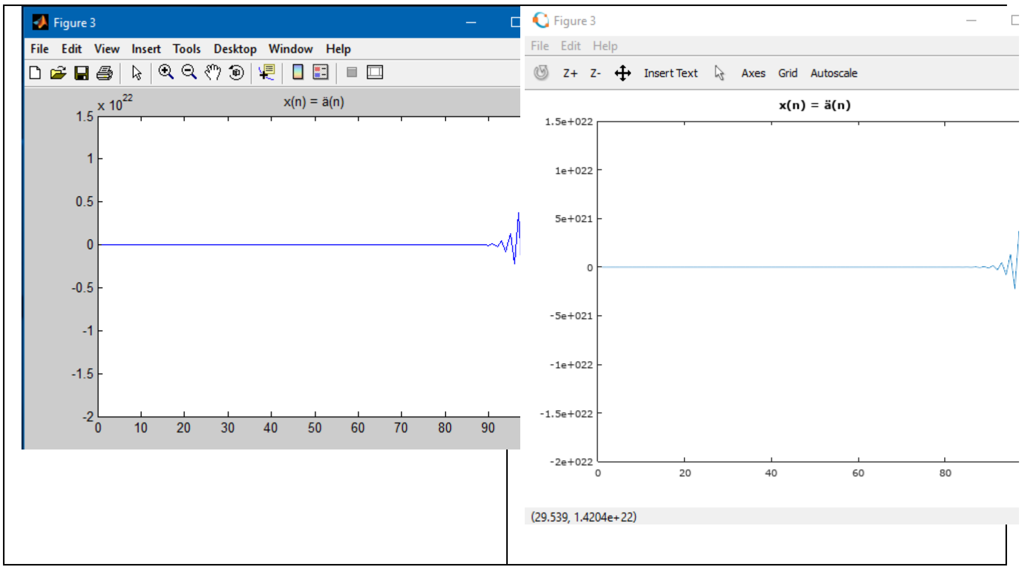
Καταρχάς Α.Μ. = 39131 οπότε η εξίσωση διαφορών μας θα είναι:

3 * y(n) + 6 * y(n-1) + 9 * y(n-3) = 12 * x(n) - 1 * x(n-2) - 1 * x(n-3)

```
ypart = [3 6 0 9]; %y meros tis exisosis
xpart = [12 -1 0 -1]; %x meros tis exisosis
n = 0:30; %xroniko diastima
d = n==0;
h = filter(xpart,ypart,d); %sxediash vimatikis apokrisis
figure;stem(n,h)
title('Κρουστική Απόκριση για n = 0:30')
x = 0.8.^n.*cos(4*10^5.*n); %eisodos synartisis 2ou erwtimatos
y = filter(xpart,ypart,x);
figure;stem(n,y)
title(Απόκριση για είσοδο x[n] = 0.8^n*cos(4*10^5*n)')
L = 100;
x = zeros(L,1);
x(1) = 0.6;
h(1) = 0.9;
h(2) = 3.15;
```

```
h(3) = 3.6;
for n = 4:L,
   h(n) = 1.5*x(n) -0.25*x(n-3) - 1.5*h(n-1) -0.5*h(n-3);
end;
figure;plot((1:L),h)
title('x(n) = δ(n)')
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE
	
	



Π.2.3.4 Απόκριση Συχνότητας Διακριτού Φίλτρου

Επίδραση της Θέσης των Πόλων στη Συνάρτηση Μεταφοράς του Φίλτρου

1. Δίνεται το σύστημα που περιγράφεται από την εξίσωση διαφορών:

$$A\cdot y(n) - B\cdot y(n-1) + \Gamma\cdot y(n-2) = \Delta\cdot x(n) - E\cdot x(n-1) + (\Delta+1)\cdot x(n-3)$$

όπου ΑΒΓΔΕ: τα ψηφία του Α.Μ..

Παράδειγμα: Αν Α.Μ. 32104 Α=3, Β=2, Γ=1, Δ=0 & Ε=4.

Να γραφεί κώδικας MATLAB ο οποίος θα απαντά στα παρακάτω ερωτήματα.

f. Να υπολογίζει και να σχεδιάζει το πλάτος και τη φάση της απόκρισης συχνότητας του συστήματος (Να περιέχονται στην απάντηση και οι γραφικές παραστάσεις).

g. Να υπολογίζει πόλους και τα μηδενικά του συστήματος. Είναι το σύστημα σταθερό; Σχεδιάστε τους πόλους και τα μηδενικά στο μιγαδικό επίπεδο.

2. Θεωρείστε την κρουστική απόκριση (h) ενός συστήματος που προκύπτει εκτελώντας την εντολή h=sig_gen(A.M.) όπου Α.Μ. ο αριθμός μητρώου σας.

a. Υπολογίστε και σχεδιάστε το πλάτος και τη φάση της απόκρισης συχνότητας του παραπάνω συστήματος.

$A \cdot y(n) - B \cdot y(n-1) + \Gamma \cdot y(n-2) = \Delta \cdot x(n) - E \cdot x(n-1) + (\Delta+1) \cdot x(n-3)$

Αντικαθιστούμε τον ΑΜ(39131)

$3y(n) - 9y(n-1) + y(n-2) = 3x(n) - x(n-1) + 4x(n-3)$

Μετασχηματισμός Z:

$3Y(z) - 9z^{-1}Y(z) + z^{-2}Y(z) = 3X(z) - z^{-1}X(z) + 4z^{-3}X(z) \Rightarrow$

$Y(z)(3 - 9z^{-1} + z^{-2}) = X(z)(3 - z^{-1} + 4z^{-3})$

$H(z) = Y(z) / X(z) = (3 - 9z^{-1} + z^{-2}) / (3 - z^{-1} + 4z^{-3})$

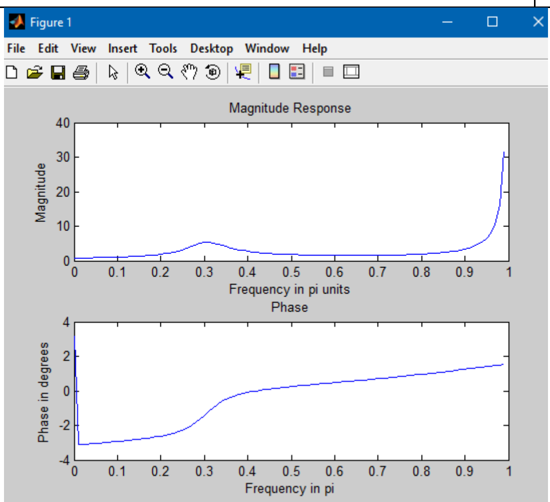
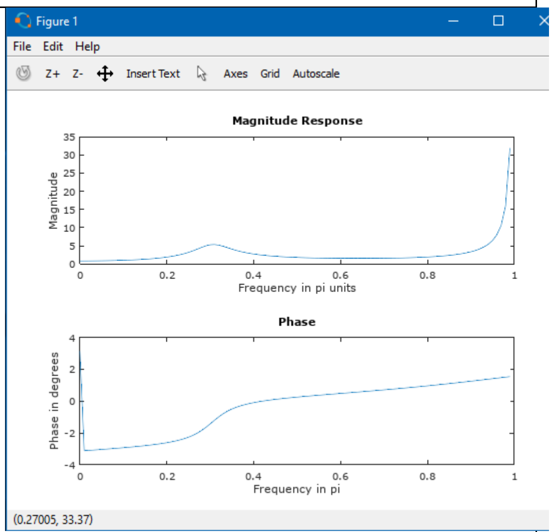
Θέτοντας όπου $z=e^{j\omega}$ παίρνουμε την απόκριση του συστήματός μας:

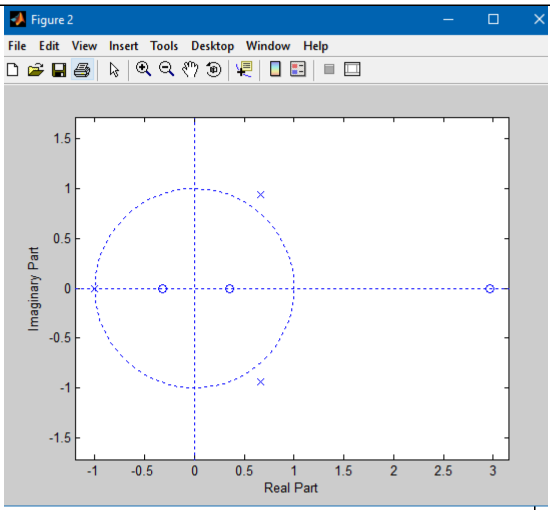
```
num = [3 -9 0 1 0];
den = [3 -1 0 4 0];
[H,w] = freqz(num,den,100); %apokrish syxnothtas
magH = abs(H); %platos
fasi = angle(H); %fash
subplot(211);
plot(w/pi, magH);
xlabel('Frequency in pi units')
ylabel('Magnitude')
title('Magnitude Response')
subplot(212);
plot(w/pi, fasi)
ylabel('Phase in degrees')
xlabel('Frequency in pi')
title('Phase')
[Z,P,K] = tf2zp(num,den); %poloi/midenika
figure;zplane(num,den) %poloi/midenika sto migadiko epipedo
h = sig_gen(39131)
w = 0:pi/255:pi;
H = freqz(h,1,w); %apokrish syxnothtas
```

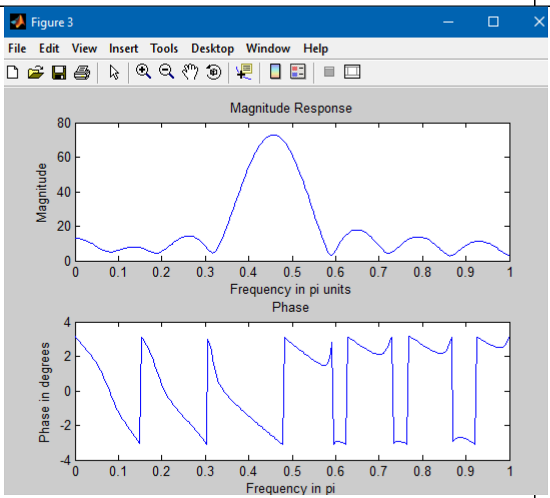
```
magH = abs(H); %platos
fasi = angle(H); %fash
figure;subplot(211);
plot(w/pi, magH);
xlabel('Frequency in pi units')
ylabel('Magnitude')
title('Magnitude Response')
subplot(212);
plot(w/pi, fasi)
ylabel('Phase in degrees')
xlabel('Frequency in pi')
title('Phase')
```

<table>
<tr><th>ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB</th><th>ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE</th></tr>
<tr><td>h =

-9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10</td><td>h =

-9 4 9 -2 -10 -2 9 4 -9 -7 6 8 -4 -10</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>





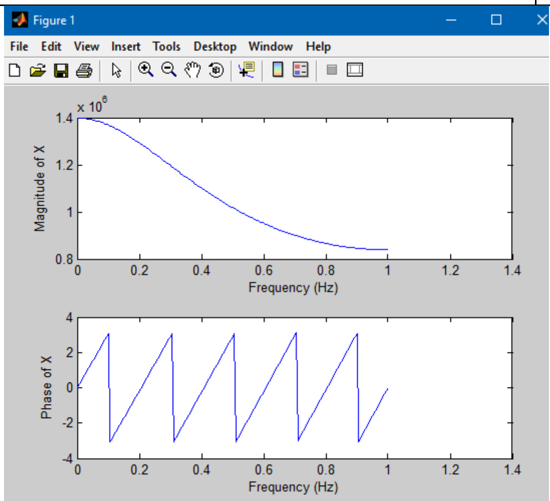
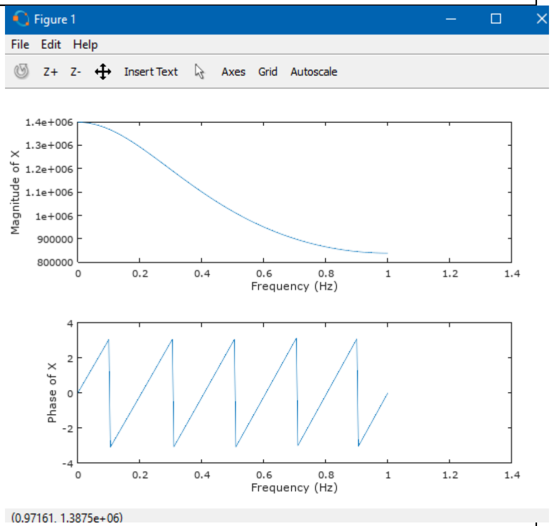
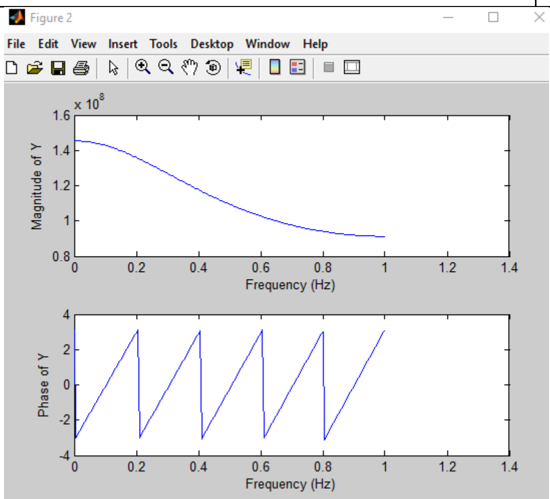
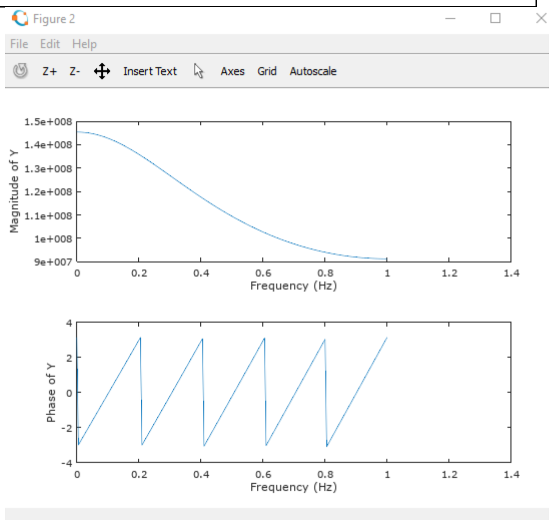
Π.2.3.5 Μετασχηματισμός Fourier Διακριτού Χρόνου

1. Δίνεται το ακόλουθο διακριτό σήμα με n= -10:10 x(n)=n * $(0.5)^{2n}$ Να γραφεί κώδικας MATLAB ο οποίος:

a) Να υπολογίζει το μετασχηματισμό Fourier του παραπάνω σήματος σε 200 συχνότητες στο διάστημα 0 – π και θα σχεδιάζει συνολικά το μετασχηματισμό (πλάτος και φάση) στο διάστημα 0 – 2π.

b) Να υπολογίζει το μετασχηματισμό Fourier για το σήμα y(n)=4*x(n-3)+5*x(n-1). Επίσης να σχεδιάζει το πλάτος και τη φάση του μετασχηματισμού.

```
n = -10:10;
k = 0:200;
x = (0.5).^(2 * n);
M = 200;
X = x * (exp(-j * pi / M)).^(n' * k);
W = (pi / M) * k;
subplot(211)
plot(W/pi,abs(X))
xlabel('Frequency (Hz)')
ylabel('Magnitude of X')
subplot(212)
plot(W/pi,angle(X))
xlabel('Frequency (Hz)')
ylabel('Phase of X')
y = 4 * x.*(n - 3) + 5 * x.*(n - 1);
Y = y * (exp(-j * pi / M)).^(n' * k);
figure;subplot(211)
plot(W/pi,abs(Y))
xlabel('Frequency (Hz)')
ylabel('Magnitude of Y')
subplot(212)
plot(W/pi,angle(Y))
xlabel('Frequency (Hz)')
ylabel('Phase of Y')
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE
	
	

2. Έστω το αναλογικό σήμα x(t)=cos(f_1 **t*)+2sin(f_2 *π **t*)-3cos(2*f_3 **t*)

Να γραφεί κώδικας MATLAB ο οποίος θα απαντά στα παρακάτω ερωτήματα.

a) Να υπολογίζει το διακριτό σήμα στο χρονικό διάστημα n=0:15 χρησιμοποιώντας την μικρότερη αλλά κατάλληλη συχνότητα δειγματοληψίας που δίνεται από τον πίνακα fs που παράγει η freq_gen. Αν καμία συχνότητα δεν ικανοποιεί τη συνθήκη Nyquist να υπολογίσετε εσείς την μικρότερη κατάλληλη συχνότητα δειγματοληψίας.

b) Να υπολογίζει το μετασχηματισμό Fourier του διακριτού σήματος και να σχεδιάζει το πλάτος και τη φάση του μετασχηματισμού για συχνότητες από 0 – 2fs. Θεωρείστε συνολικά 200 συχνότητες στο παραπάνω διάστημα.

Οι συχνότητες του σήματος f_1, f_2 και f_3 (πίνακας f) καθώς και οι συχνότητες δειγματοληψίας fs_1, fs_2 και fs_3 (πίνακας fs) παράγονται με τη βοήθεια της συνάρτησης [f,fs]=freq_gen(AM) όπου Α.Μ. ο πενταψήφιος αριθμός μητρώου σας. (π.χ. Αν ο αριθμός μητρώου σας είναι 12345 τότε [x,y]=freq_gen(12345)).

f = 323   116   251

fs = 126   219   700

Αντικαθιστώντας στο αναλογικό μας σήμα έχουμε:

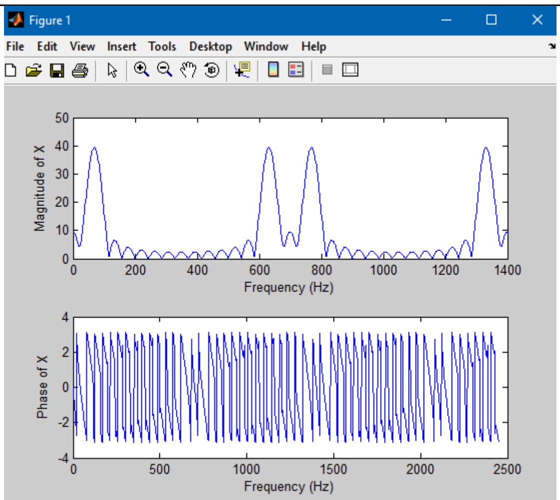
x(t) = cos(323 **t*)+2sin(116 *π **t*)-3cos(2*251 **t*) =>

x(t) = cos(323 **t*)+2sin(116 *π **t*)-3cos(502*t)

Από το θεώρημα του Nyquist έχουμε: fs>=2fmax. Το fmax ειναι το 323 * 2 = 646 άρα πληρεί την προϋπόθεση. 700>=646. Άρα χρησιμοποιούμε fs = 700.

```
[f,fs] = freq_gen(39131)
n = 0:15;
fs = 700; %συχνότητα δειγματοληψίας
k = 0:2*fs;
Ts = 1/fs;
x = cos(323 * n * Ts) + 2 * sin(116 * pi * n * Ts) - 3 * cos(502 * n * Ts);
%αναλογικό σήμα
M = 200; %συχνότητες
X = x * (exp(-j * pi / M)).^(n' * k);
F = ((fs / 2) / M) * k;
subplot(211)
plot(F,abs(X))
axis([0 2*fs 0 50])
xlabel('Frequency (Hz)')
ylabel('Magnitude of X')
```

```
subplot(212)
plot(F,angle(X))
xlabel('Frequency (Hz)')
ylabel('Phase of X')
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΤΑ ΣΤΟ OCTAVE
f = 123 181 601 fs = 133 362 1304	f = 123 181 601 fs = 133 362 1304
	

Π.2.3.6 Μετασχηματισμός Fourier Διακριτού Χρόνου

1. Θεωρείστε τις ακολουθίες πεπερασμένου μήκους x(n),y(n) οι οποίες προκύπτουν μετά την εκτέλεση της εντολής [x,y]=sig_gen(A.M.) (π.χ. Αν ο

αριθμός μητρώου σας είναι 12345 τότε [x,y]=sig_gen(12345)). Να κατασκευάσετε ένα m-file που θα δέχεται ως είσοδο τις δύο ακολουθίες x(n) και y(n) και θα υπολογίζει:

h. Την κυκλική συνέλιξη g(n)=x(n)*y(n)

i. Την κυκλική συνέλιξη με χρήση του Διακριτού Μετασχηματισμού Fourier (DFT)

j. Συγκρίνετε τα δύο αποτελέσματα μεταξύ τους υπολογίζοντας το σφάλμα που προκύπτει κατά τους υπολογισμούς. Τι παρατηρείτε;

```
[x,y] = sig_gen(39131);
length(x);
length(y);
if length(x) > length(y)
   then
   k = length(x)
else
   k = length(y)
end
ycir =circonvt(x,y,k); %κυκλική συνέλιξη
disp('Κυκλική συνέλιξη');
disp(ycir)
G1 = fft(x);
G2 = fft(y);
yc = real(ifft(G1.*G2)); %κυκλική συνέλιξη με DFT
disp('Κυκλικη  συνέλιξη με χρήση του DFT');
disp(yc)
error = max(abs(ycir-yc)); %σφάλμα
disp('Σφάλμα');
disp(error)
```

Αποτελέσματα του Matlab:

<u>Κυκλική συνέλιξη</u>

```
  Columns 1 through 13
    82   -11  -137  -122    -4    37   -21   -19   106   170    73   -47   -35

  Column 14
    32
```

<u>Κυκλική συνέλιξη με χρήση του DFT</u>

```
  Columns 1 through 8
   82.0000  -11.0000 -137.0000 -122.0000   -4.0000   37.0000  -21.0000  -
19.0000

  Columns 9 through 14
  106.0000  170.0000   73.0000  -47.0000  -35.0000   32.0000
```

<u>Σφάλμα</u>

```
  4.9738e-014
```

Παρατηρούμε πως το σφάλμα είναι πολύ μικρό. Επίσης παρατηρούμε πως και με τους 2 τρόπους κυκλικής συνέλιξης έχουμε το ίδιο αποτέλεσμα.

2. Θεωρήστε το σήμα x(t) = 2cos(2*π*f1*t) + 3sin(2*π*f2*t) - cos(2*π*f3*t) a. Να γραφεί κώδικας MATLAB ο οποίος θα απαντά στα παρακάτω ερωτήματα. Να δειγματοληπτήσετε κατάλληλα το σήμα χρησιμοποιώντας την μικρότερη αλλά κατάλληλη συχνότητα δειγματοληψίας που δίνεται από τον πίνακα f_S που παράγει η freq_gen. Αν καμία συχνότητα δεν ικανοποιεί τη συνθήκη Nyquist να υπολογίσετε εσείς την μικρότερη κατάλληλη συχνότητα δειγματοληψίας.

b. Να υπολογίζει το διακριτό μετασχηματισμό Fourier του δειγματοληπτημένου σήματος και να σχεδιάζει το πλάτος και τη φάση του μετασχηματισμού, αν σε αυτό προστεθεί στο πεδίο του χρόνου θόρυβος πλάτους 1 του οποίου τα δείγματα προέρχονται από κανονική κατανομή. Για μήκος του μετασχηματισμού χρησιμοποιήστε τη δύναμη του 2 η οποία είναι αμέσως μεγαλύτερη από το μήκος του διανύσματος του σήματος.

Οι συχνότητες του σήματος f1, f2 και f3 (πίνακας f) καθώς και οι συχνότητες δειγματοληψίας fs1, fs2 και fs3 (πίνακας fs) παράγονται με τη βοήθεια της συνάρτησης [f,fs]=freq_gen(AM) όπου Α.Μ. ο πενταψήφιος αριθμός μητρώου σας. (π.χ. Αν ο αριθμός μητρώου σας είναι 12345 τότε [x,y]=freq_gen(12345)).

Τρέχοντας το [f,fs]=freq_gen(39131) έχουμε:

```
f =

   123   181   601


fs =

   133         362         1304
```

x(t) = 2cos(2*π*f1*t) + 3sin(2*π*f2*t) - cos(2*π*f3*t)

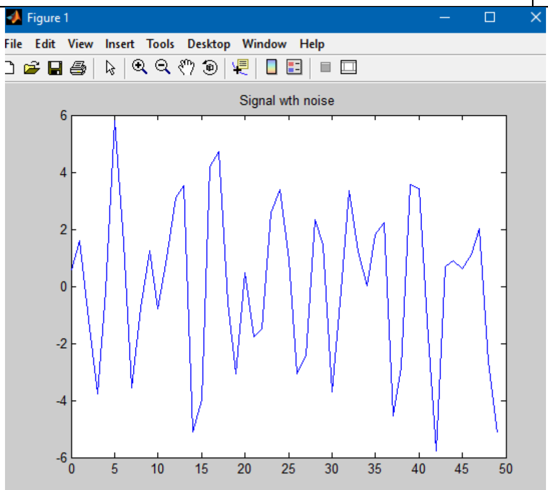
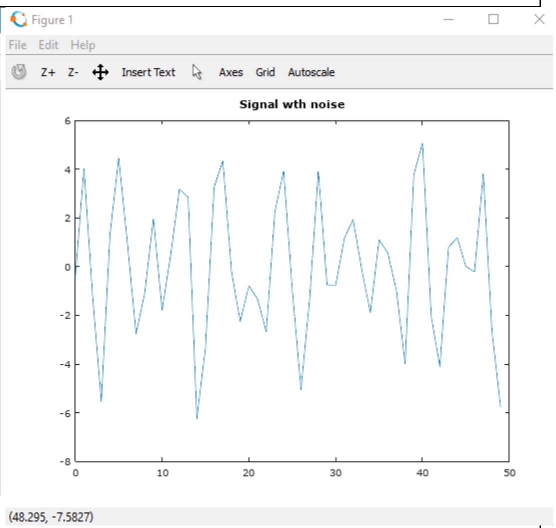
Αντικαθιστώντας έχουμε:

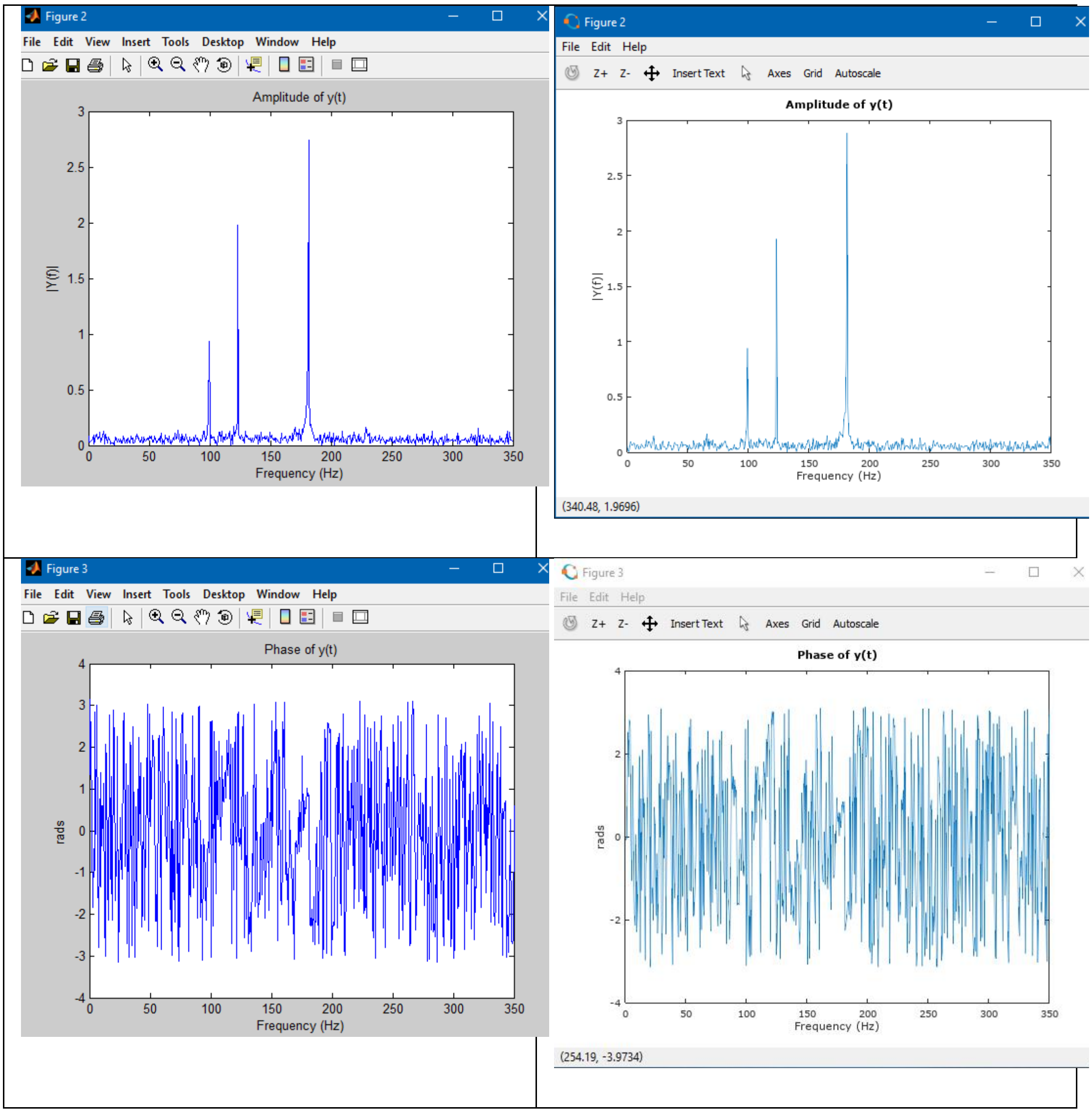
x(t) = 2cos(2*π*123*t) + 3sin(2*π*181*t) – cos(2*π*601*t)

Από το θεώρημα του Nyquist έχουμε: fs>=2fmax. Το fmax είναι το 123 * 2 = 246 άρα πληρεί την προϋπόθεση. 700>=246. Άρα χρησιμοποιούμε το fs = 700.

```
fs = 700; %συχνότητα δειγματοληψίας
T = 1/fs; %χρόνος δείγματος
L = 1000; %μήκος σήματος
```

```
t = (0:L-1)*T; %χρόνος διανύσματος
x = 2*cos(2*pi*123*t)+3*sin(2*pi*181*t)-cos(2*pi*601*t); %σήμα
y = x+randn(size(t)); %ημίτονο με θόρυβο
NFFT = 2^nextpow2(L); %επόμενη δύναμη του 2 από το μήκος του σήματος
Y = fft(y,NFFT)/L;
f = fs/2*linspace(0,1,NFFT/2);
figure;plot(fs*t(1:50),y(1:50))
title('Signal wth noise')
figure;plot(f,2*abs(Y(1:NFFT/2)))
title('Amplitude of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
figure;plot(f,angle(Y(1:NFFT/2)))
title('Phase of y(t)')
xlabel('Frequency (Hz)')
ylabel('rads')
```

<table>
<tr><th>ΑΠΟΤΕΛΕΣΜΑΤΑΣ ΣΤΟ MATLAB</th><th>ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>



Π.2.3.7 Ταχύς Μετασχηματισμός Fourier

1. Θεωρείστε τις ακολουθίες πεπερασμένου μήκους x(n),y(n) οι οποίες προκύπτουν μετά την εκτέλεση της εντολής [x,y]=sig_gen(A.M.)
(π.χ. Αν ο αριθμός μητρώου σας είναι 12345 τότε [x,y]=sig_gen(12345)).

a) Να υπολογίσετε την γραμμική συνέλιξη των ακολουθιών με χρήση της conv και του FFT. Να γίνουν οι γραφικές παραστάσεις και να συγκριθούν τα αποτελέσματα. Τι παρατηρείτε;

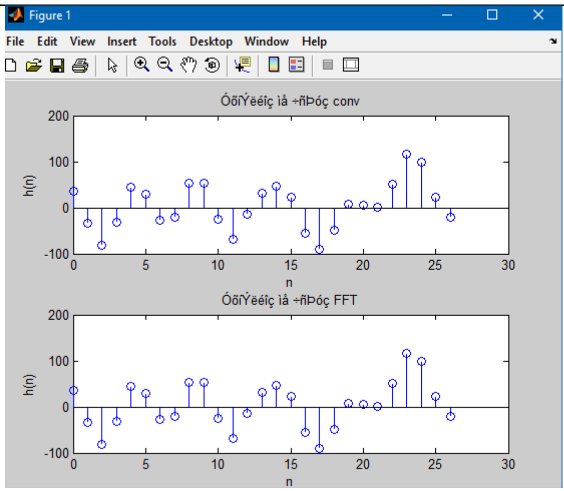
b) Να υπολογίσετε την κυκλική συνέλιξη των δύο ακολουθιών. Με το άθροισμα ποιών στοιχείων της γραμμικής συνέλιξης είναι ίσο το τρίτο στοιχείο της κυκλικής συνέλιξης των δύο ακολουθιών;

```
[x,y] = sig_gen(39131)
h = conv(x,y); %συνέλιξη με conv
n = 0:length(h)-1;
X = fft(x,length(h));
Y = fft(y,length(h));
h2 = ifft(X.*Y); % συνέλιξη με FFT
subplot(211),stem(n,h)
title('Συνέλιξη με χρήση conv')
xlabel('n')
ylabel('h(n)')
subplot(212),stem(n,h2)
title('Συνέλιξη με χρήση FFT')
xlabel('n')
ylabel('h(n)')
disp('Γραμμική συνέλιξη')
disp(h)
```

Γραμμική συνέλιξη

```
 Columns 1 through 13

      x =

      -9    4    9   -2  -10   -2    9    4   -9   -7    6    8   -4  -10



      y =

      -4    2    6    9    9    7    2   -3   -8  -10  -10   -7   -3    2

 Columns 14 through 26

       36  -34  -82  -31   45   30  -27  -20   54   54  -25  -69  -15   32   46
      23  -55  -91  -49    7    6    1   52  116   98   22

 Column 27
```

-20

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATALAB	ΑΠΟΤΕΣΕΣΜΑΤΑ ΣΤΟ OCTAVE
	

2. Δίνεται το σήμα: x(n) = sin(CBD*pi*n) + C*cos(2*ABE*pi*n) + B*sin(2*CBB*pi*n) όπου ABCDE: τα ψηφία του Α.Μ. σας (Αν Α.Μ. 32104 A=3, B=2, C=1, D=0 & E=4)

a) Να υπολογιστεί ο μετασχηματισμός FFT του σήματος. Για το σήμα σας επιλέξτε ως συχνότητα δειγματοληψίας την fs = 3 * fmax , όπου fmax η μέγιστη συχνότητα που περιέχει το σήμα σας.

Θεωρήστε χρονική διάρκεια στο διάστημα 0 έως 1 με κατάλληλο βήμα.

b) Να γίνει η γραφική παράσταση του πλάτους του FFT του σήματος. Στον οριζόντιο άξονα να παριστάνονται οι αναλογικές συχνότητες του σήματος.

Έχουμε: x(n) = sin(CBD*pi*n) + C*cos(2*ABE*pi*n) + B*sin(2*CBB*pi*n)

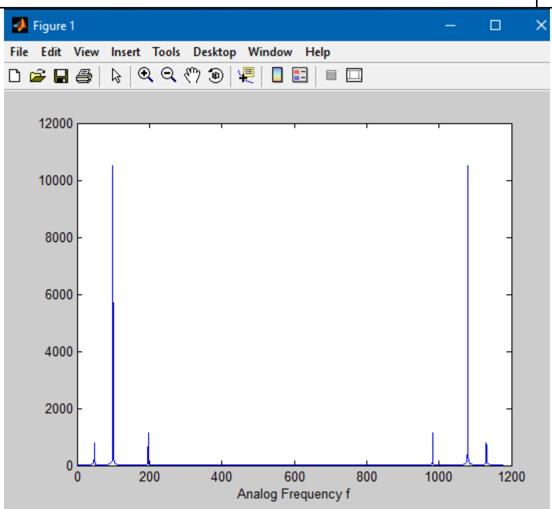
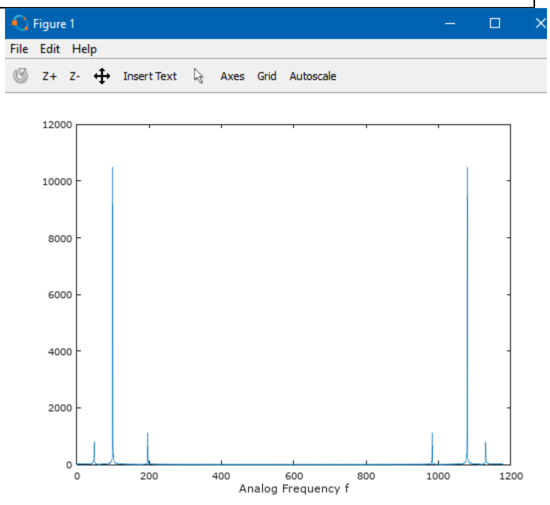
Αντικαθιστώντας με το ΑΜ έχουμε: x(n) = sin(193*pi*n) + 1*cos(2*393*pi*n) + 9*cos(2*199*pi*n)

Άρα: x(n) = sin(193 * pi * n) + cos(786*pi*n) + 9*cos(398*pi*n)

Η fmax του σήματος είναι 786 άρα fs = 3 * 786 = 2358

fs = 2358; %συχνότητα δειγματοληψίας

```
ts = 1/fs;
n = 0:ts:1; %χρονική διάρκεια με βήμα 1/fs
x = sin(193 * pi * n) + cos(786*pi*n) + 9*cos(398*pi*n); %ορισμός σήματος
X = fft(x);
k = 0:length(X)-1;
w = (k * (2*pi/length(X)/2))/ts; %υπολογισμός αναλογικής κυκλικής συχνότητας
f = w/(2*pi); %αναλογική συχνότητα σε Hz
plot(f,abs(X));
xlabel('Analog Frequency f')
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ MATLAB	ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΤΟ OCTAVE
	

ΠΑΡΑΡΤΗΜΑ 3

ΨΗΦΙΟΠΟΙΗΣΗ ΚΑΙ ΑΝΑΣΥΣΤΑΣΗ ΣΗΜΑΤΟΣ ΜΕ SIMULINK

Π.3.1 Εισαγωγή στο Simulink

Το πρόγραμμα Simulink είναι μία προέκταση του Matlab και υλοποιεί με εύκολο και πρακτικό τρόπο την μαθηματική συνάρτηση που περιγράφει το σύστημα που πρόκειται να προσομοιωθεί, χρησιμοποιώντας κατάλληλα δομικά στοιχεία (Blocks) συναρτήσεως που περιέχονται σε αντίστοιχες βιβλιοθήκες. Έτσι δίδετε η δυνατότητα στο χρήση να δομήσει γρήγορα και με ακρίβεια ένα μαθηματικό μοντέλο που αντιπροσωπεύει ένα οποιοδήποτε σύστημα απλό ή σύνθετο, γραμμικό ή μη γραμμικό.

Ένα μοντέλο Simulink μπορεί να περιλαμβάνει μεταβλητές συνεχούς και διακριτού χρόνου και να παράγει γραφικές απεικονίσεις που δείχνουν την πορεία προσομοίωσης ζωντανά, βοηθώντας έτσι σημαντικά στην κατανόηση της δυναμικής συμπεριφοράς του συστήματος.

Η συνηθισμένη διαδικασία μοντελοποίησης και προσομοίωσης μέσω ηλεκτρονικού υπολογιστή ενός δυναμικού συστήματος στο παρελθόν, ήταν η ανάπτυξη πρώτα ενός δομικού διαγράμματος και η μετάφρασή του στη συνέχεια σε πρόγραμμα χρησιμοποιώντας μία από τις γνωστές γλώσσες προγραμματισμού. Η μέθοδος αυτή εκτός από το ότι απαιτούσε διπλή προσπάθεια, (μία για τη σχεδίαση και την ανάπτυξη του δομικού διαγράμματος και μία για τον προγραμματισμό), αύξανε σημαντικά τις πιθανότητες λάθους και κυρίως το χρόνο εντοπισμού του. Το δυσκολότερο και το πιο χρονοβόρο πρόβλημα στη διόρθωση του λάθους σε μία τέτοια διαδικασία, είναι αναμφίβολα ο εντοπισμός του ίδιου του λάθους. Γιατί το λάθος μπορεί να είναι στη σχεδίαση, δηλαδή στο δομικό διάγραμμα, στο πρόγραμμα, ή στη μετατροπή του δομικού διαγράμματος σε πρόγραμμα. Με τη χρήση του Simulink αφ' ενός αποφεύγεται η διπλή προσπάθεια στην ολοκλήρωση της διαδικασίας μοντελοποίησης, άρα η μοντελοποίηση απλοποιείται στο μέγιστο δυνατό βαθμό, και αφ' ετέρου η πιθανότητα σφάλματος πρακτικά εξαλείφεται για το λόγο ότι ουσιαστικά ο προγραμματισμός εμπεριέχεται στο δομικό διάγραμμα, άρα είναι προελεγμένος και αποκλείει οποιαδήποτε πιθανότητα λάθους.

Π.3.2 Ειδικά χαρακτηριστικά του Simulink

Η βελτίωση την οποία προσφέρει αλλά και υπόσχεται για το μέλλον η μέθοδος του δομικού διαγράμματος και ειδικά το Simulink στη μοντελοποίηση και την προσομοίωση κυρίως δυναμικών συστημάτων είναι τεράστια. Αυτό προκύπτει τόσο από τα στοιχεία που έχουν ήδη αναφερθεί, όσο και για πολλούς ακόμη λόγους που προκύπτουν από την ανάλυση του αντικειμένου και πρωτίστως από την έμπρακτη ενασχόληση και την εφαρμογή του. Ένα από τα σημεία που δεν πρέπει να παραλείψουμε να αναφέρουμε είναι η σημαντική συμβολή του Simulink στη βελτίωση της παραγωγικότητας της όλης διαδικασίας μοντελοποίησης - προσομοίωσης. Από τα κρίσιμα μεγέθη που επιδρούν στην παραγωγικότητα μια τέτοιας διαδικασίας, είναι ο χρόνος και το κόστος, το οποίο και εξαρτάται από τον χρόνο. Για αυτό το θέμα θα αναφέρουμε δύο βασικές στρατηγικές που εφαρμόζει το Simulink και είναι:

- Διαδικασία ταχείας ανάπτυξης πρωτοτύπου και
- Διαδικασία ταχείας ανάπτυξης της εφαρμογής.

Διαδικασία ταχείας ανάπτυξης πρωτοτύπου είναι η εφαρμογή παραγωγικών εργαλείων για την ανάπτυξη λειτουργικών πρωτοτύπων συστημάτων ελέγχου στον ελάχιστο δυνατό χρόνο. Κριτήριο βελτιστοποίησης στη διαδικασία αυτή αποτελεί η ταχύτητα ανάπτυξης της μεθόδου, παρά η ταχύτητα εκτέλεσης ή χρήσης της μνήμης. Η διαδικασία αυτή αρχίζει με την ανάπτυξη της προσομοίωσης του φυσικού συστήματος που πρόκειται να ελεγχθεί, που αυτό μπορεί να είναι ένα μοντέλο του Simulink ή κάποια εργαλειοθήκη του Matlab, όπως για παράδειγμα την εργαλειοθήκη βελτιστοποίησης. Κατά τη διάρκεια της όλης διαδικασίας εκτελούνται λεπτομερειακές δοκιμές για να διασφαλιστεί ότι η σχεδίαση ικανοποιεί όλες τις λειτουργικές απαιτήσεις. Αφού το μοντέλο του Simulnk έχει ελεγχθεί διεξοδικά, προγραμματίζεται ο ελεγκτής σε γλώσσα συμβατή με τον βασικό υπολογιστή, στον οποίο και φορτώνεται το πρόγραμμα για να ακολουθήσουν στη συνέχεια οι δοκιμές ελέγχου της καλής συνεργασίας του ελεγκτή και του μοντέλου.

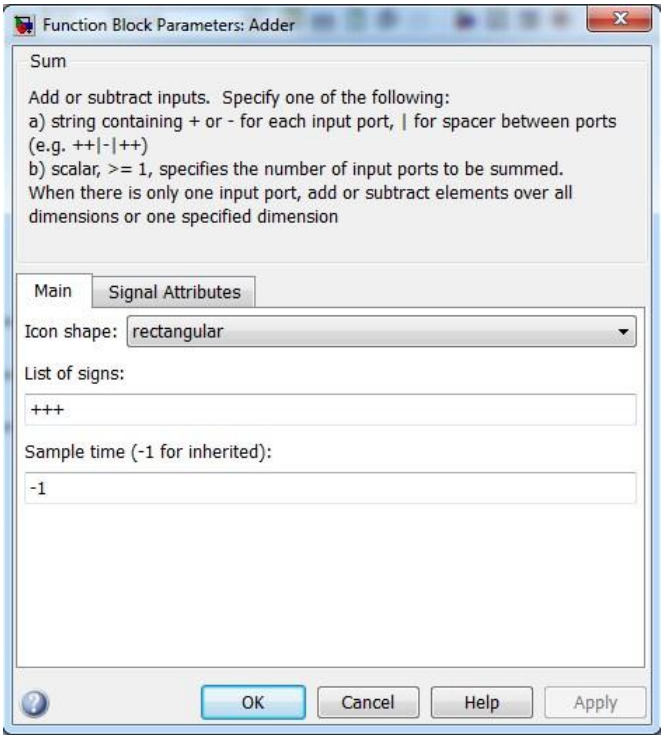
Η διαδικασία ταχείας ανάπτυξης της εφαρμογής είναι μία επέκταση της μεθοδολογίας ταχείας ανάπτυξης προτύπου, στην οποία το τελικό πρόγραμμα του υπολογιστή είναι το ίδιο το μοντέλο του Simulink, ή παράγεται αυτόματα απ' αυτό. Έτσι εξαλείφεται συνολικά η διαδικασία προγραμματισμού σε κάποιο γλώσσα. Στη περίπτωση μάλιστα που ο ελεγκτής είναι υπολογιστής συμβατός με το Simulink, τότε αυτό αποτελεί και την τελική γλώσσα προγραμματισμού του όλου συστήματος. Συχνά το μοντέλο Simulink του ελεγκτή δεν συνεργάζεται ικανοποιητικά με το υπόλοιπο σύστημα, είτε γιατί δεν αποκρίνεται αρκετά γρήγορα, είναι γιατί ο ελεγκτής μπορεί να είναι κάποιος ενσωματωμένος μικροεπεξεργαστής μ – συμβατός με το Simulink. Σε μία τέτοια περίπτωση δεν έχουμε παρά να χρησιμοποιήσουμε το εργαστήριο πραγματικού χρόνου που περιέχεται στο πακέτο Matlab/Simulink, το οποίο μετατρέπει αυτόματα το δομικό διάγραμμα του Simulink σε επιμέρους προγράμματα C τα οποία μπορούν να μεταγλωττιστούν ανάλογα και να φορτωθούν στο βασικό υπολογιστή.

Π.3.3 ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ ΨΗΦΙΟΠΟΙΗΣΗ ΚΑΙ ΑΝΑΣΥΣΤΑΣΗ ΣΗΜΑΤΟΣ ΜΕ SIMULINK

Π.3.3.1 Δημιουργία σήματος εισόδου

Αρχικά ανοίγουμε το Simulink και δημιουργούμε ένα νέο μπλοκ διάγραμμα. Στη συνέχεια από τη βιβλιοθήκη simulink / sources βάζουμε στο μπλοκ διάγραμμα τρία μπλοκ “Sine Wave” και από τη βιβλιοθήκη simulink / math operations προσθέτουμε ένα μπλοκ “Add”. Επίσης από τη βιβλιοθήκη simulink / sinks προσθέτουμε τέσσερα μπλοκ “Scope”.

Πατάμε διπλό κλικ με το ποντίκι πάνω στο μπλοκ “Add” και παραμετροποιούμε τις παραμέτρους όπως φαίνεται στο σχήμα Π.3.1. Δηλαδή προσθέτουμε ένα “+” στο πεδίο “List of sings:”.

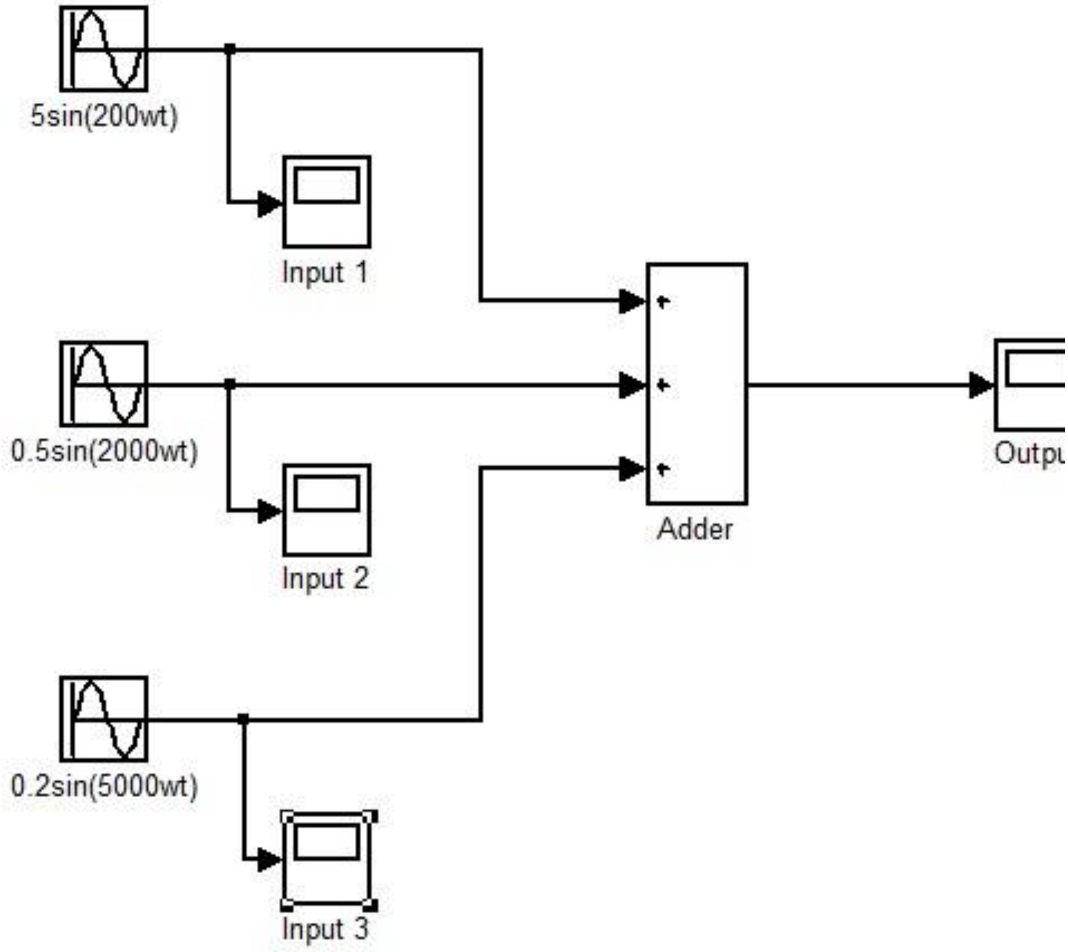


Σχήμα Π.3.1: Παράμετροι του “Add” μπλοκ

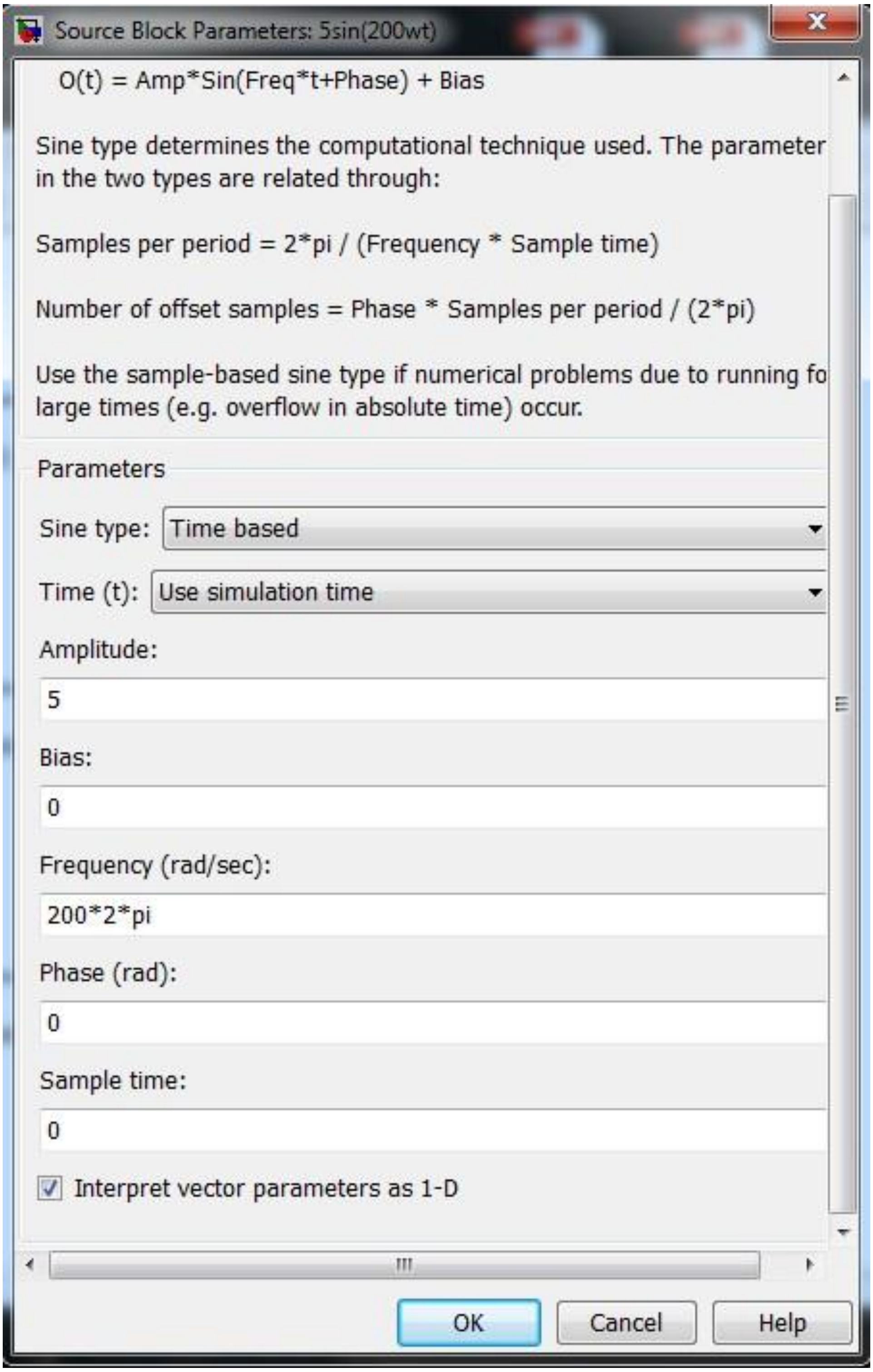
Ενώνουμε μεταξύ τους τα μπλοκ ώστε να σχηματίσουμε το διάγραμμα που φαίνεται στο σχήμα Π.3.2.

Πατώντας διπλό κλικ σε κάθε ένα από τα μπλοκ “Sine Wave” τα παραμετροποιούμε ώστε να δημιουργήσουμε τα ακόλουθα σήματα. Στα σχήματα Π.3.3, Π.3.4 και Π.3.5 φαίνονται οι αλλαγές που πρέπει να γίνουν για το κάθε σήμα.

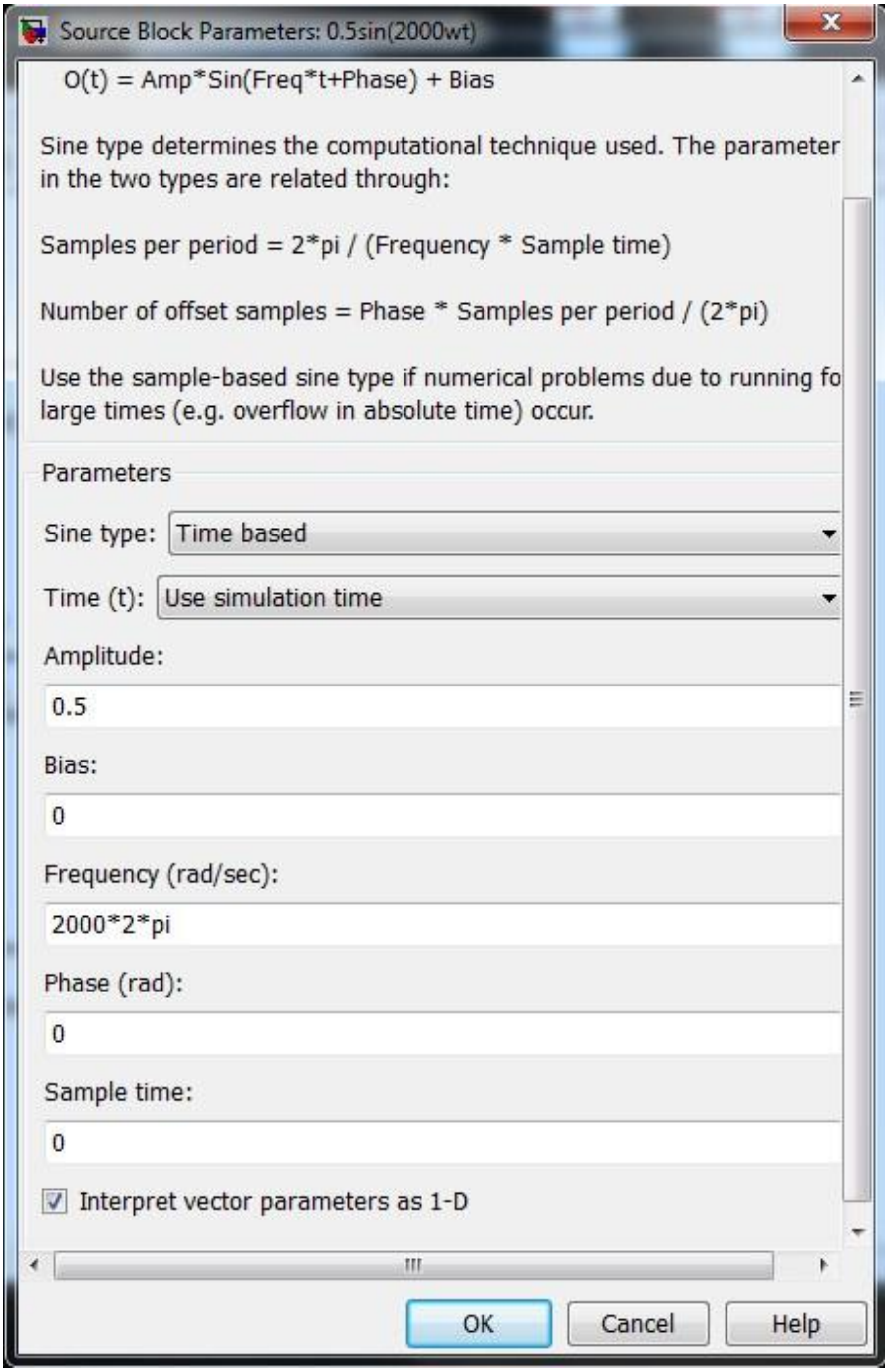
- $5\sin(200\omega t)$
- $0.5\sin(2000\omega t)$
- $0.2\sin(5000\omega t)$



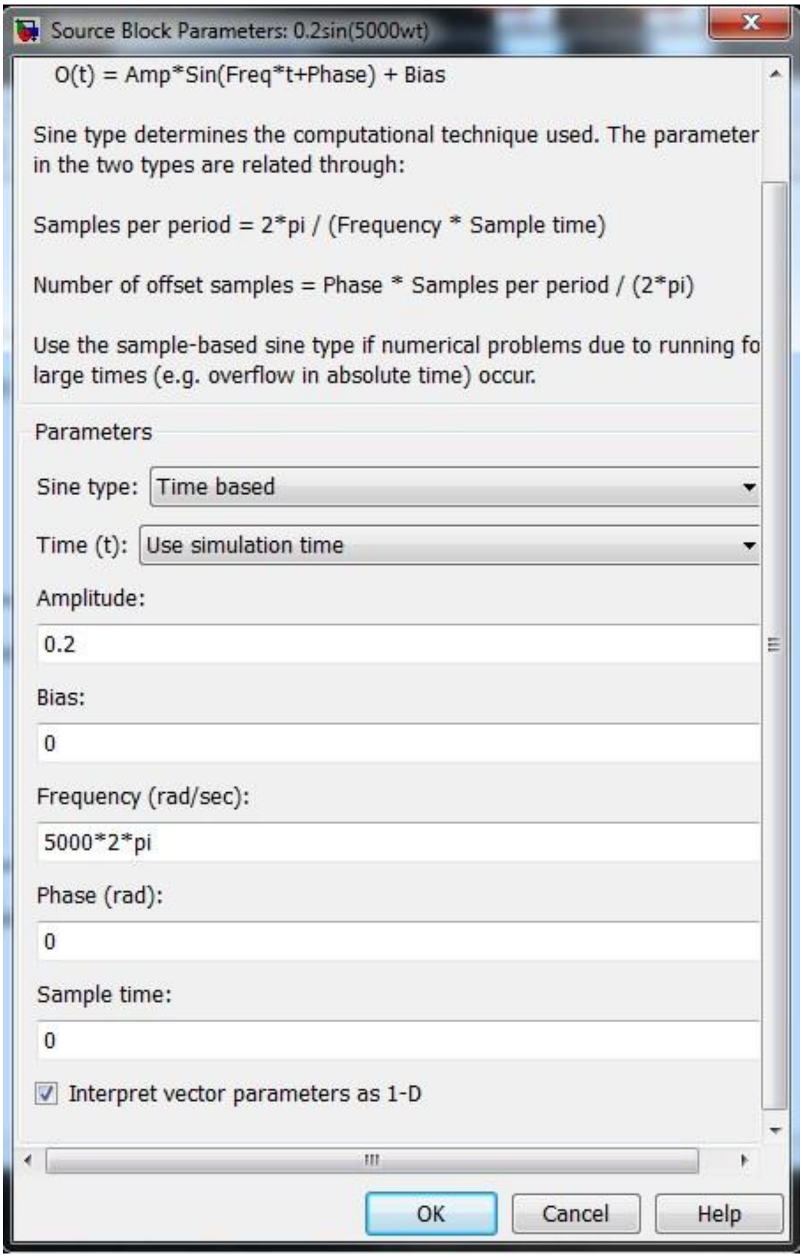
Σχήμα Π.3.2: Διάγραμμα 1ου μέρους



Σχήμα Π.3.3: Θεμελιώδες σήμα - 5sin(200ωt)



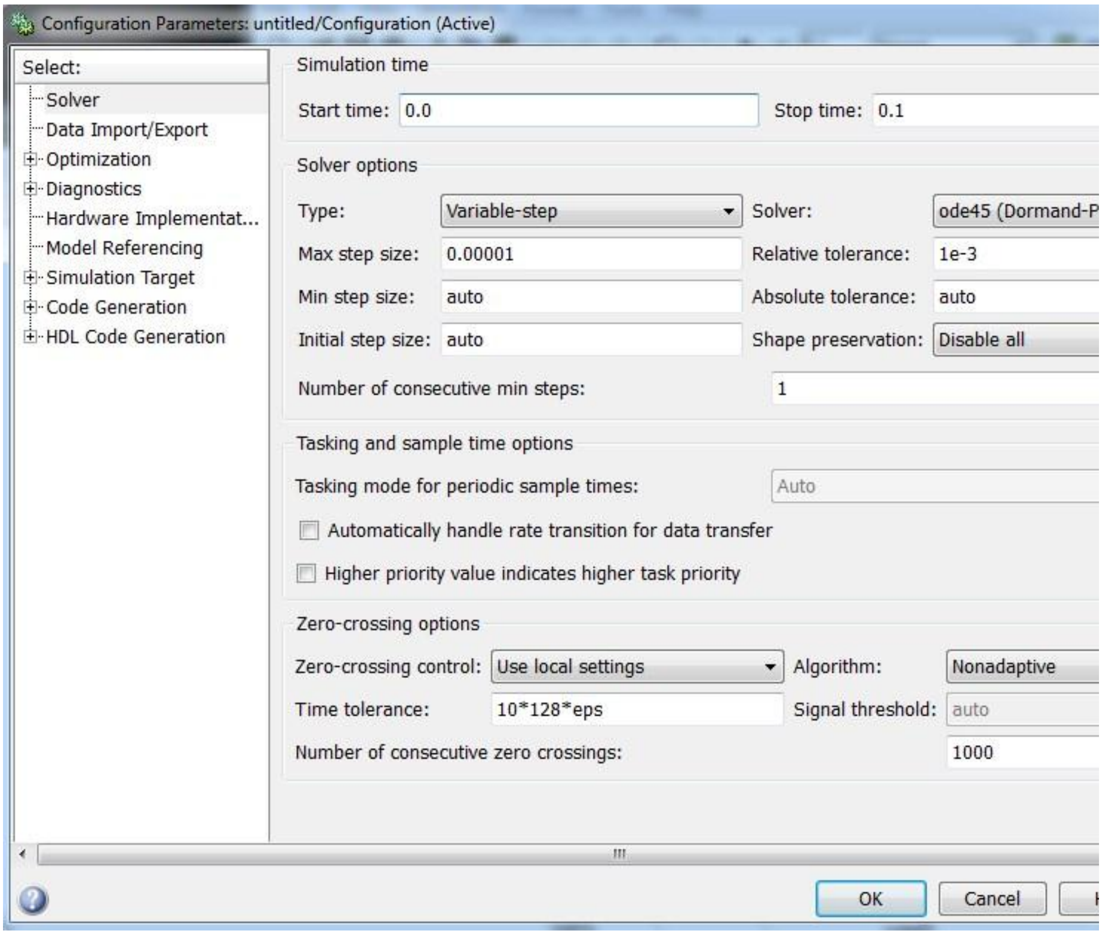
Σχήμα Π.3.4: Σήμα αρμονικών - 0.5sin(2000ωt)



Σχήμα Π.3.5: Σήμα αρμονικών - 0.2sin(5000ωt)

Για να γίνει η προσομοίωση και να δούμε από τα μπλοκ “Scope” τα σήματα πρέπει να αλλάξουμε κάποιες ρυθμίσεις στη προσομοίωση πηγαίνοντας από το παράθυρο του διαγράμματος στο simulation / configuration parameters και στο πεδίο “max step size” βάζουμε τον αριθμό που φαίνεται στο σχήμα Π.3.6. Αυτή η παράμετρος κανονικά είναι στο “auto” αλλά όπως έδηξαν οι προσομοιώσεις, τα αποτελέσματα στο “Scope” δεν ήταν σωστές. Όποτε όσο το νούμερο στο "max step size" γινόταν μικρότερο τα αποτελέσματα γίνονταν καλύτερα. Επειδή σε διαφορετικές εκδόσεις του matlab συμπεριφέρεται διαφορετικά, μπορείτε να κάνετε δοκιμές και να βρείτε τη κατάλληλη τιμή.

Επίσης μπορούμε να αλλάξουμε και το χρόνο προσομοίωσης ανάλογα με τις ανάγκες μας. Στις μετέπειτα προσομοιώσεις θα χρησιμοποιηθούν τιμές για το χρόνο προσομοίωσης από 0.01s έως και 0.1s. Αυτό γιατί τα σήματα που χρησιμοποιούμε έχουν τέτοια συχνότητα που σε προσομοιώσεις με μεγαλύτερο χρόνο δεν θα φαίνονται αναλυτικά στο "Scope".



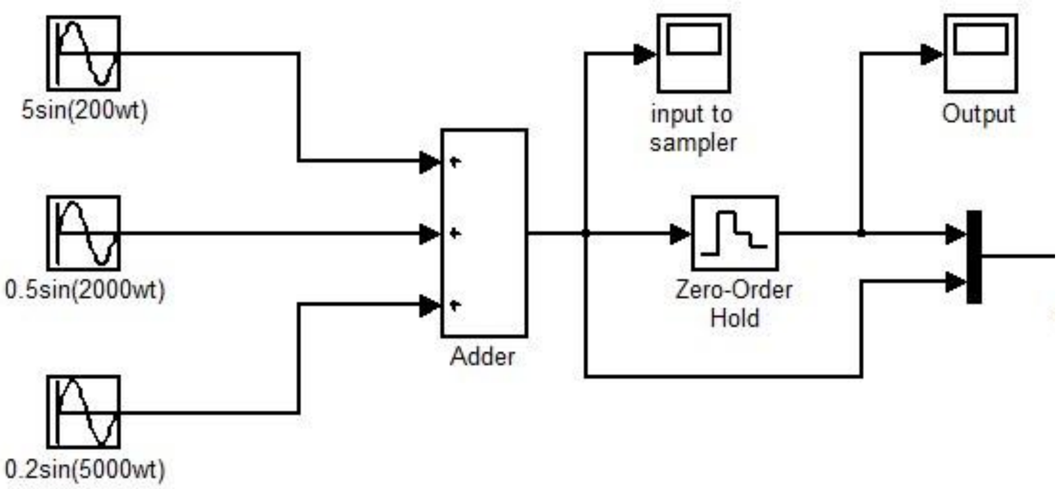
Σχήμα Π.3.6: Παράμετροι για προσομοίωση

Αφού έγιναν όλες οι ρυθμίσεις, πατάμε το σύμβολο για την εκκίνηση της προσομοίωσης ►.

Πατώντας διπλό κλικ πάνω στα "Scope" μπορούμε να δούμε το κάθε σήμα ξεχωριστά. Καλό θα ήταν να επιλέγεται το κουμπί "auto scale" όταν μπούμε στο "Scope" ώστε να προσαρμόζεται το σήμα μέσα στους άξονες.

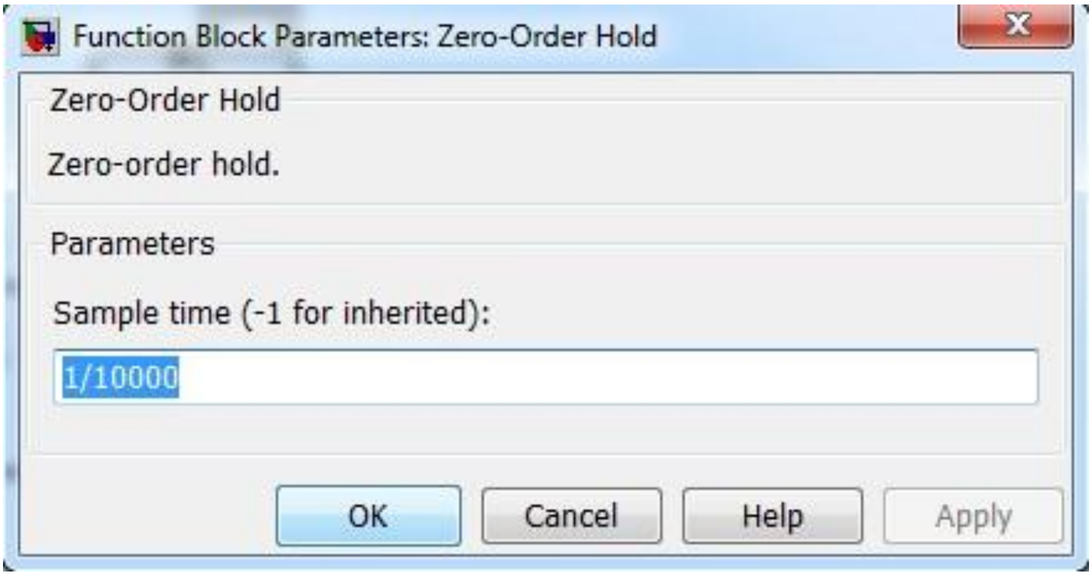
Π.3.3.2 Δειγματοληψία σήματος

Σε αυτό το μέρος θα προσθέσουμε στο μοντέλο ένα μπλοκ που ονομάζεται "zero-order hold" και μπορούμε να το βρούμε στη βιβλιοθήκη simulink/discrete/Zero-Order Hold και να το σύρουμε μέσα στο μοντέλο. Επίσης μπορούμε να αναζητήσουμε το κάθε μπλοκ γράφοντας το όνομα στο πεδίο αναζήτησης. Επιπλέον προσθέτουμε και ένα μπλοκ που ονομάζεται "mux" και βρίσκεται στη βιβλιοθήκη simulink/commonly used blocks/mux. Αυτό είναι προαιρετικό και το χρησιμοποιούμε στο μοντέλο για να εμφανίσουμε δύο σήματα στο ίδιο διάγραμμα. Προσθέτουμε και τα απαραίτητα "Scope" στο μοντέλο και ενώνουμε να μπλοκ όπως φαίνεται στο σχήμα Π.3.7.

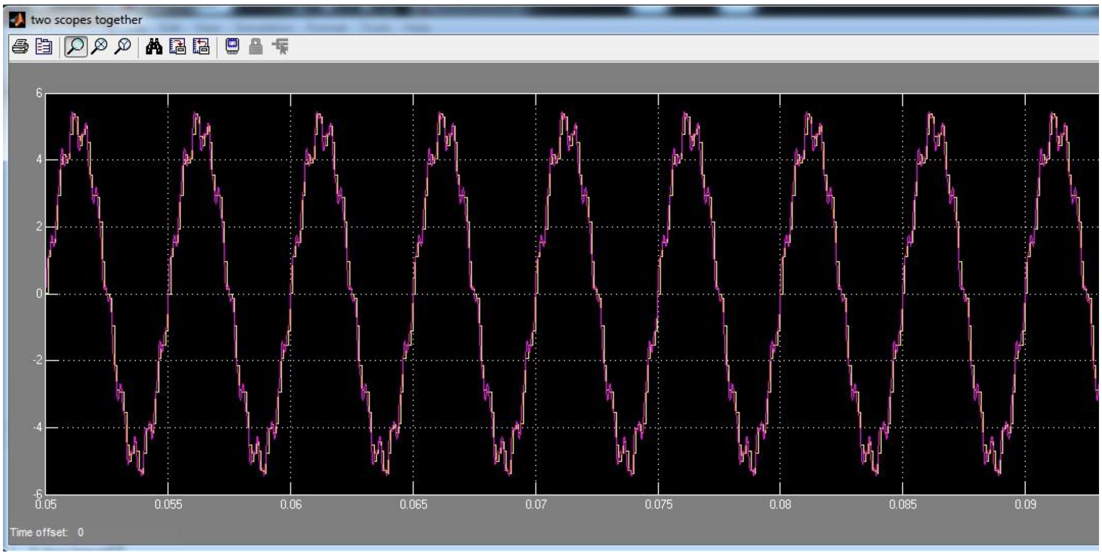


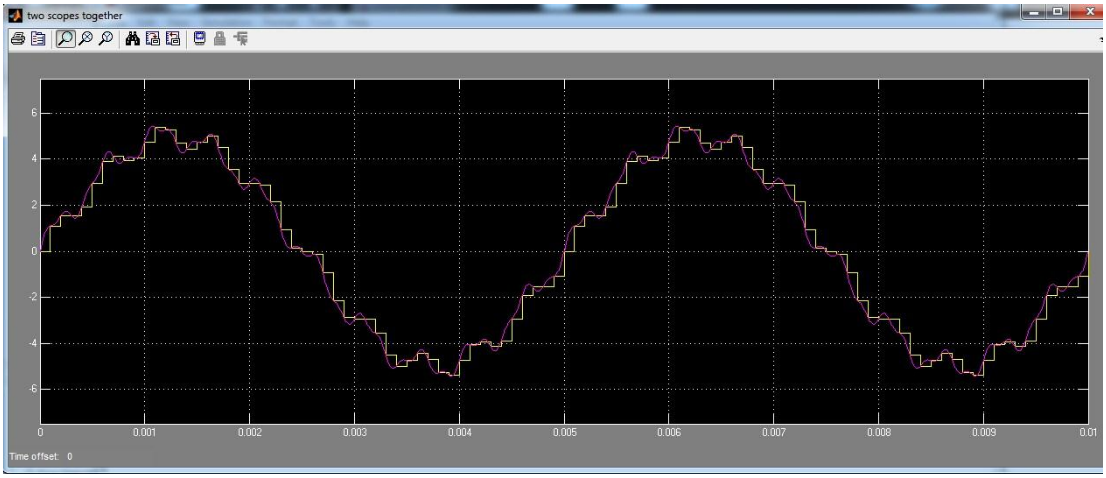
Σχήμα Π.3.7: Διάγραμμα 2^{ου} μέρους

Σε αυτή τη φάση χρησιμοποιούμε το "zero-order hold" για την δειγματοληψία του σήματος εισόδου. Κάνουμε διπλό κλικ πάνω του και στο πεδίο sample time βάζουμε τη τιμή 1/10000 όπως φαίνεται στο σχήμα Π.3.8. Αυτή η τιμή είναι ο χρόνος που περνάει από δείγμα σε δείγμα δηλαδή ο παρανομαστής είναι η συχνότητα δειγματοληψίας του σήματος εισόδου. Σύμφωνα με το θεώρημα δειγματοληψίας Nyquist – Shannon η συχνότητα δειγματοληψίας πρέπει να είναι ίση και μεγαλύτερη από το διπλάσιο της μέγιστης συχνότητα του σήματος που δειγματοληπτούμε ώστε να μην ληφθεί λανθασμένο σήμα (aliasing) και να μπορεί το σήμα να ανακατασκευαστεί σωστά στη συνέχεια.



Σχήμα Π.3.8: Παράμετροι του μπλοκ "Zero-Order Hold"





Σχήμα Π.3.9: Σήμα εισόδου και εξόδου μαζί (πάνω 0.1s, κάτω 0.01s)

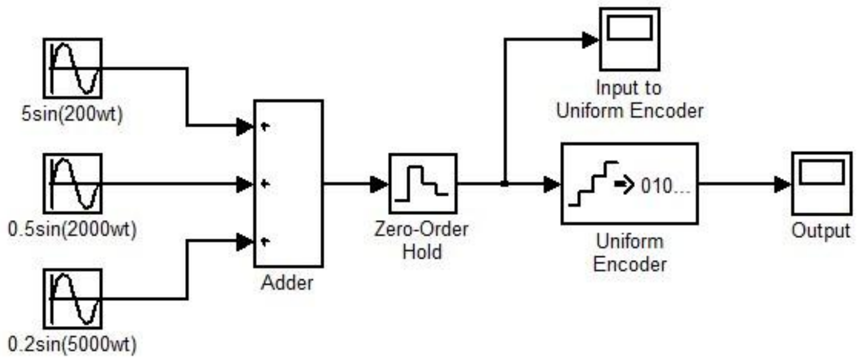
Αν παρατηρήσουμε το σήμα εξόδου προσεκτικά θα δούμε ότι έχει πάρει τιμές ανάλογα με το σημείο που παίρνει δείγμα από το σήμα εισόδου.

Η επόμενη δουλειά που πρέπει να γίνει είναι η κβαντοποίηση των τιμών αυτών σε στάθμες ακέραιας τιμής.

1.1 Π.3.3.3 Κβαντοποίηση

Προσθέτουμε στο μοντέλο ένα ακόμα μπλοκ που ονομάζεται “Uniform Encoder” και το βρίσκουμε στη βιβλιοθήκη DSP System

Toolbox/Quantizers/Uniform Encoder. Το βάζουμε στο μοντέλο μαζί με τα κατάλληλα "Scope" και ενώνουμε τα μπλοκ όπως φαίνεται στο σχήμα Π.3.10.



Σχήμα Π.3.10: Διάγραμμα 3^{ου} μέρους

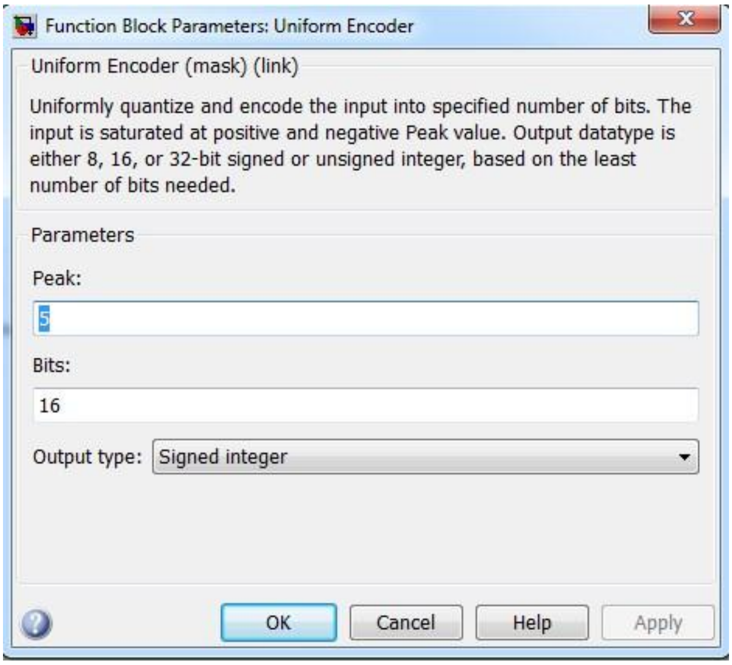
Κάνοντας διπλό κλικ πάνω στο νέο μπλοκ ανοίγει το παράθυρο με τις παραμέτρους και στα πεδία βάζουμε τις τιμές που φαίνονται στο σχήμα Π.3.11.

Στο πεδίο "Peak" βάζουμε τον αριθμό της μέγιστης τιμής που θέλουμε να υπολογίσει το μπλοκ και στο "Bits" βάζουμε πόσα bits θέλουμε να έχει η κάθε τιμή που θα υπολογίσει. Επίσης αλλάζουμε το "Output Type" σε singed Integer.

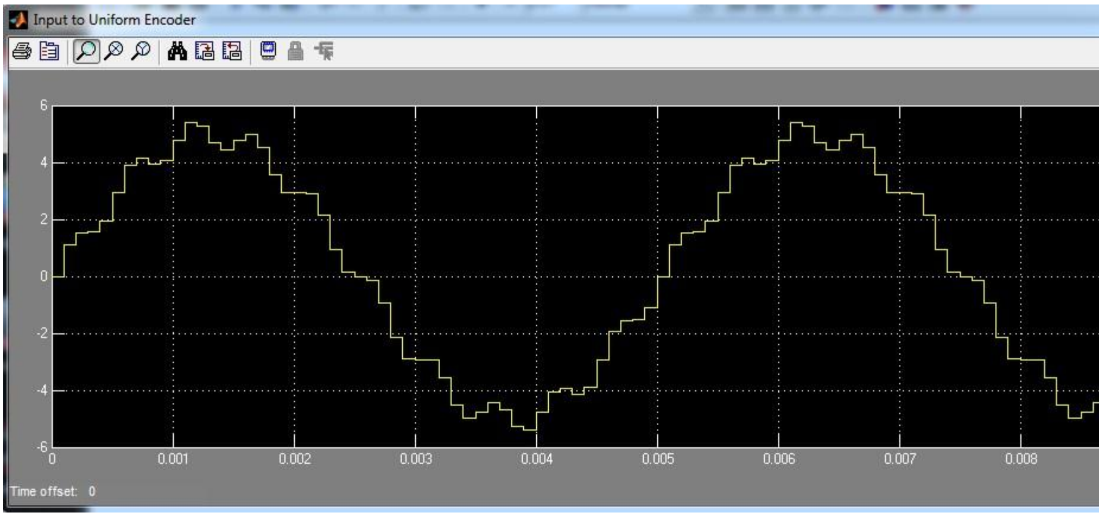
Έτσι ο κβαντιστής θα δημιουργήσει στάθμες ακέραιας τιμή με μέγεθος 16bit και θα πάρει τις τιμές του σήματος εισόδου και θα τις «στρογγυλοποιήσει» ανάλογα, στις στάθμες που έχει φτιάξει. Το αποτέλεσμα θα είναι το σήμα εξόδου να έχει «σκαλοπάτια» ακέραιας τιμής καθορισμένη από τις στάθμες. Η στάθμες λόγο του 16bitου παίρνουν μεγάλες τιμές αλλά έχουμε προϋποθέσει με τη τιμή Peak ότι αντιστοιχούν σε τιμές από -5 έως 5.

Πατάμε το κουμπί της προσομοίωσης με χρόνο 0.01s.

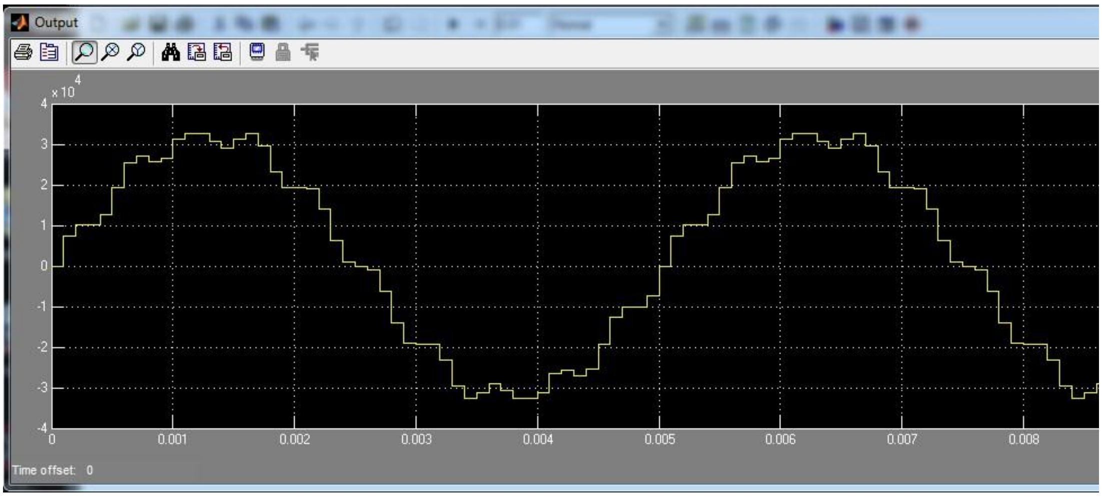
Στα σχήματα Π.3.12 και Π.3.13 φαίνονται η είσοδος και η έξοδος του κβαντιστή σε προσομοίωση με χρόνο 0.01s διότι αν γίνει σε μεγαλύτερο χρόνο δεν μπορούμε να ξεχωρίσουμε κάποια πληροφορία στη γραφική παράσταση των σημάτων.



Σχήμα Π.3.11: Παράμετροι του μπλοκ



Σχήμα Π.3.12: Σήμα εισόδου στο κβαντιστή



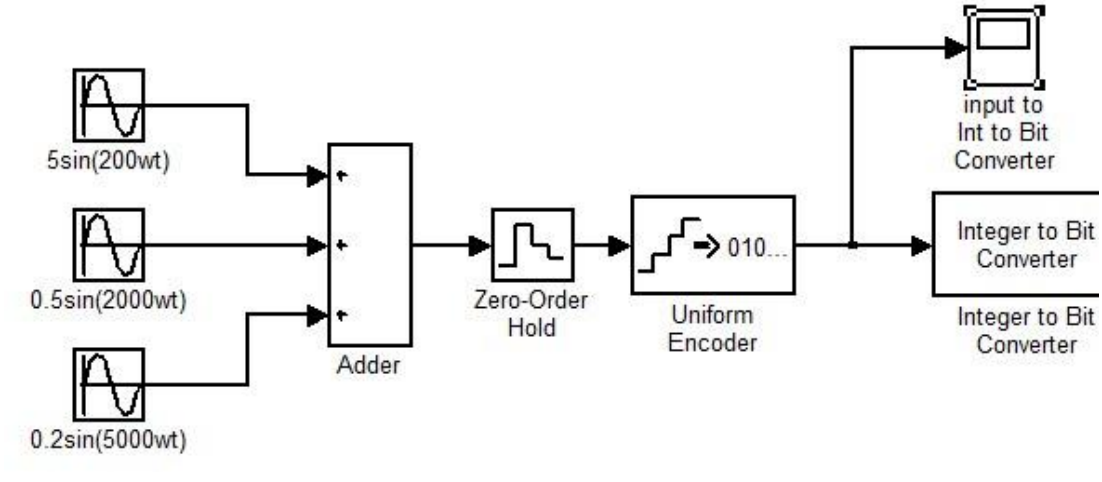
Σχήμα Π.3.13: Σήμα εξόδου από το κβαντιστή

Για να παρατηρήσουμε ακριβός τις διαφορές στα δύο σήματα, εκτός από τη διαφορά στο πλάτος, θα μπορούσαμε να χρησιμοποιήσουμε το μπλοκ “to workspace” από τη βιβλιοθήκη simulink/sinks/to workspace και να το συνδέσουμε όπου είναι συνδεδεμένα και τα “Scope” στο μοντέλο. Να αλλάξουμε τη παράμετρο “save format” σε array και να τρέξουμε τη προσομοίωση ξανά. Σαν έξοδο θα πάρουμε δύο πίνακες με τις τιμές των σημάτων εισόδου και εξόδου στο workspace του matlab. Έτσι μπορούμε αναλυτικότερα να δούμε τις διαφορές των δύο σημάτων.

Σε αυτή τη φάση λοιπόν έχουμε ένα σήμα κβαντισμένο με 16bitους ακέραιους αριθμούς. Η επόμενη φάση είναι να μετατρέψουμε αυτούς του ακέραιους σε δυαδικούς για να πάρουμε το ψηφιακό σήμα που αναζητάμε.

Π.3.3.4 Ψηφιοποίηση

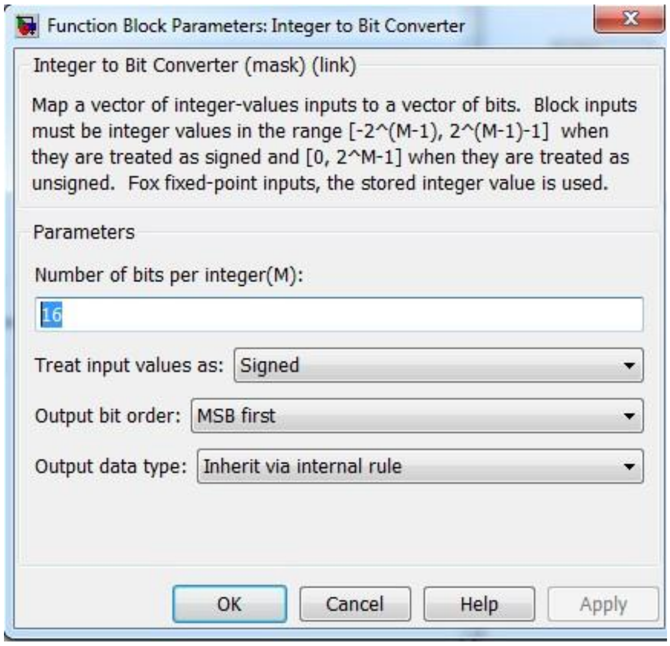
Στη βιβλιοθήκη Communication System Toolbox/Utility Blocks επιλέγουμε το μπλοκ “Integer to Bit Converter” και το σέρνουμε στο μοντέλο μας. Συνδέουμε τα μπλοκ όπως φαίνεται στο σχήμα Π.3.14.



Σχήμα Π.3.14 Διάγραμμα 4^{ου} μέρους

Αυτό το μπλοκ παίρνει τις ακέραιες τιμές των στάθμεων μετά τη κβάντωση του σήματος και τις μετατρέπει σε δυαδικό σύστημα.

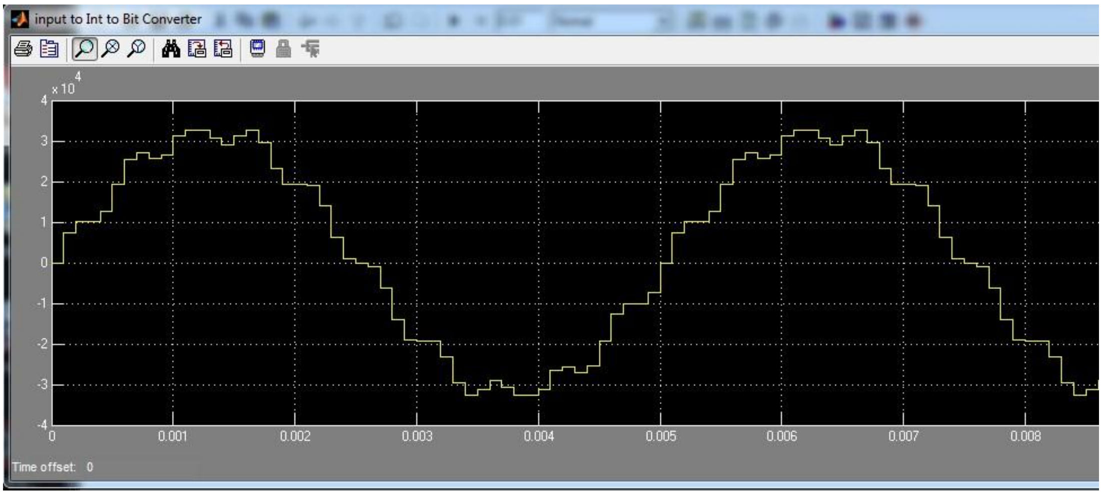
Κάνοντας διπλό κλικ στο μπλοκ μπαίνουμε στο παράθυρο των παραμέτρων και τις αλλάζουμε όπως φαίνεται στο σχήμα Π.3.15.



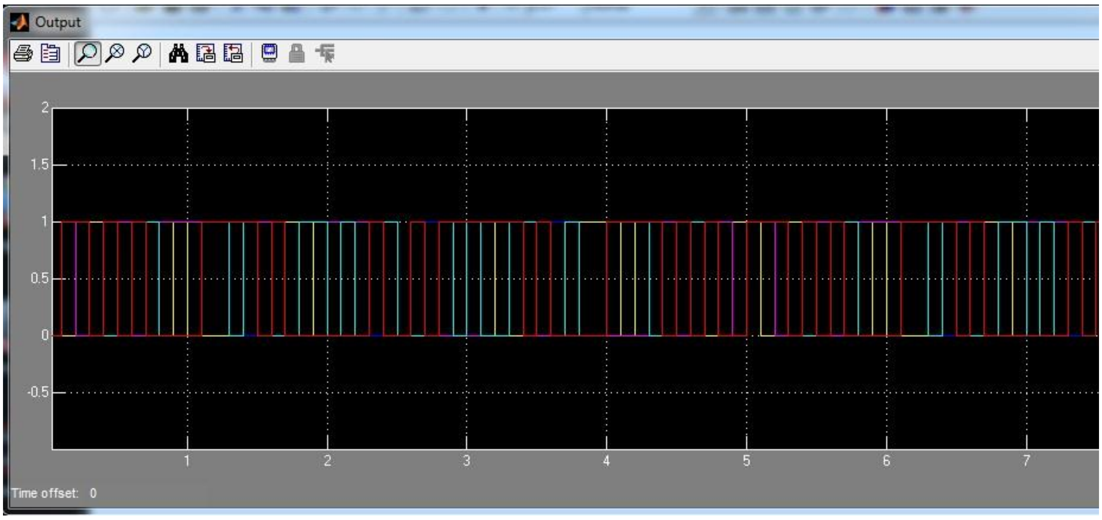
Σχήμα Π.3.15: Παράμετροι του μπλοκ Integer to Bit Converter

Το πεδίο “number of bits per integer” πρέπει να γίνει 16 για να συμβαδίζει με τα προηγούμενα μπλοκ και το “treat input values as:” πρέπει να γίνει Signed.

Πατάμε το κουμπί για τη προσομοίωση με χρόνο 0.01s και ανοίγοντας τα Scope βλέπουμε το σήμα εισόδου και εξόδου του μπλοκ που μελετάμε.



Σχήμα Π.3.16: Σήμα εισόδου στον μετατροπέα integer to bit



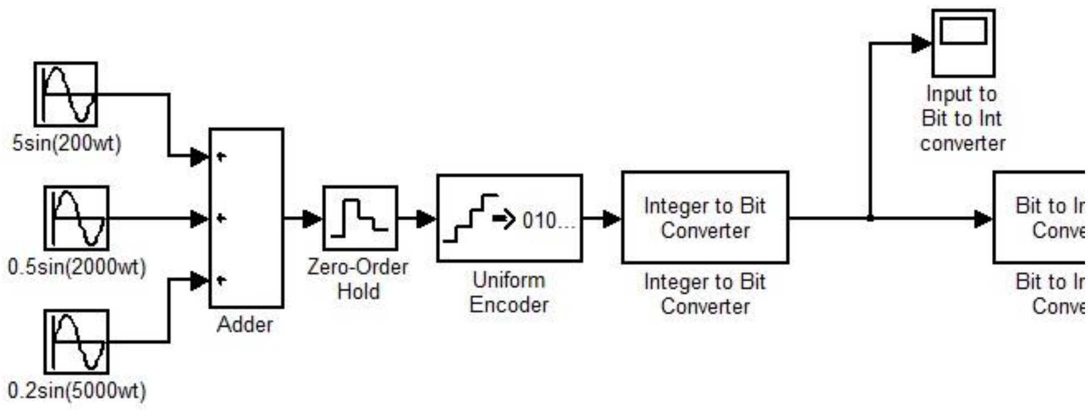
Σχήμα Π.3.17: Σήμα εξόδου του μετατροπέα integer to bit

Παρατηρούμε ότι το σήμα εξόδου έχει τιμές 0 και 1. Πλέον έχουμε το σήμα μας σε ψηφιακή μορφή. Για να παρατηρήσουμε πιο αναλυτικά το σήμα εξόδου μιας και δεν μπορούμε να πάρουμε αναλυτική πληροφορία από το Scope μπορούμε να χρησιμοποιήσουμε το «κόλπο» που κάναμε στο 3^{ο} μέρος της εργασίας με το μπλοκ “to workspace” ώστε να πάρουμε στο workspace του matlab τον πίνακα με τις τιμές του σήματος εξόδου.

Π.3.3.5 Μετατροπή δυαδικό σε ακέραιο

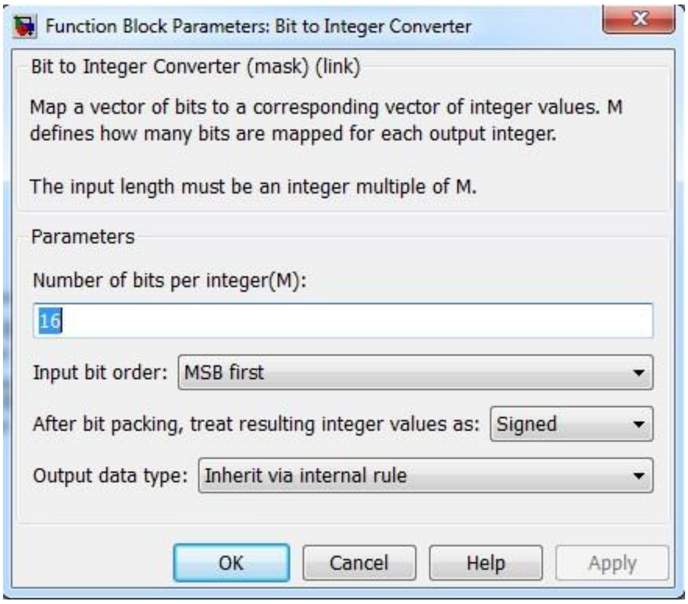
Με τη χρήση του ψηφιακού σήματος, έστω σε κάποια εφαρμογή που απαιτεί την επεξεργασία του, καλούμαστε να το επαναφέρουμε ξανά σε αναλογικό. Από το μέρος αυτό της εργασίας και μέχρι το τέλος θα δούμε τις διαδικασίες για την επαναφορά του σήματος από ψηφιακό σε αναλογικό. Θα κάνουμε δηλαδή την ανασύσταση του ψηφιακού σήματος.

Πηγαίνουμε πάλι στη βιβλιοθήκη Communication System Toolbox/Utility Blocks και επιλέγουμε το “Bit to Integer Converter” μπλοκ και το σέρνουμε στο μοντέλο. Συνδέουμε τα μπλοκ όπως φαίνεται στο σχήμα Π.3.18.



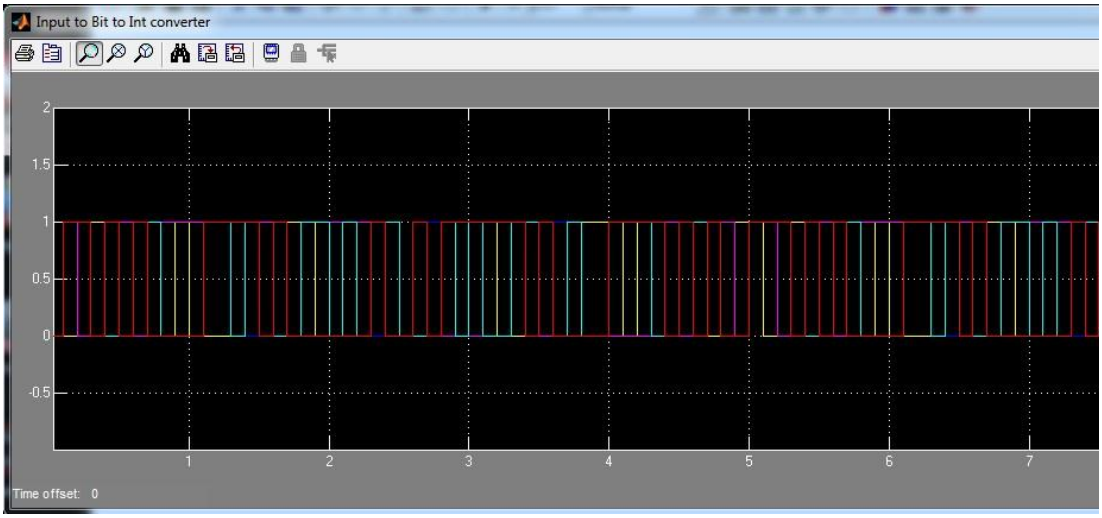
Σχήμα Π.3.18: Διάγραμμα 5^{ου} μέρους

Κάνουμε διπλό κλικ στο μπλοκ και αλλάζουμε τις παραμέτρους.

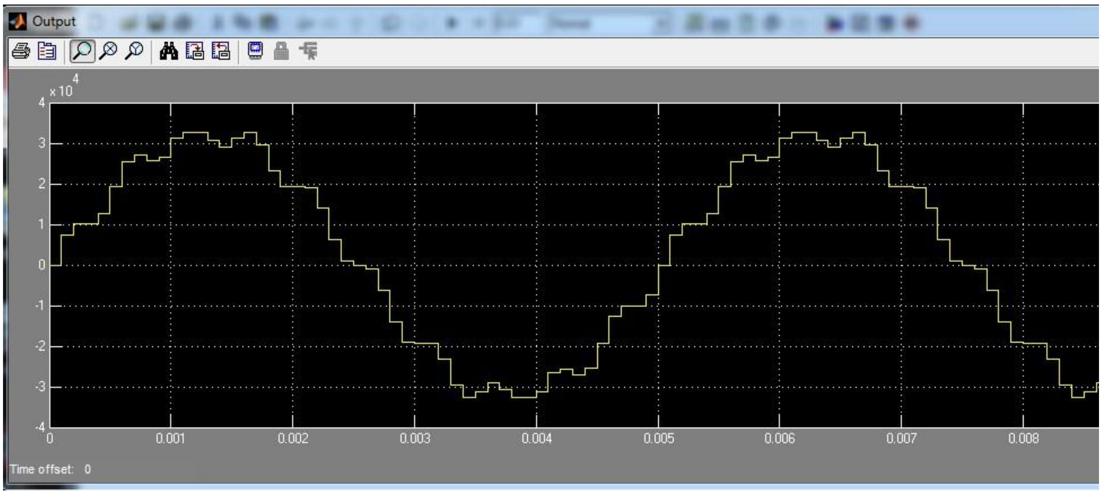


Σχήμα Π.3.19: Παράμετροι του μπλοκ Bit to integer Converter

Ανοίγοντας τα Scope παρατηρούμε την είσοδο και την έξοδο αυτού του μπλοκ.



Σχήμα Π.3.20: Σήμα εισόδου στον μετατροπέα bit to integer

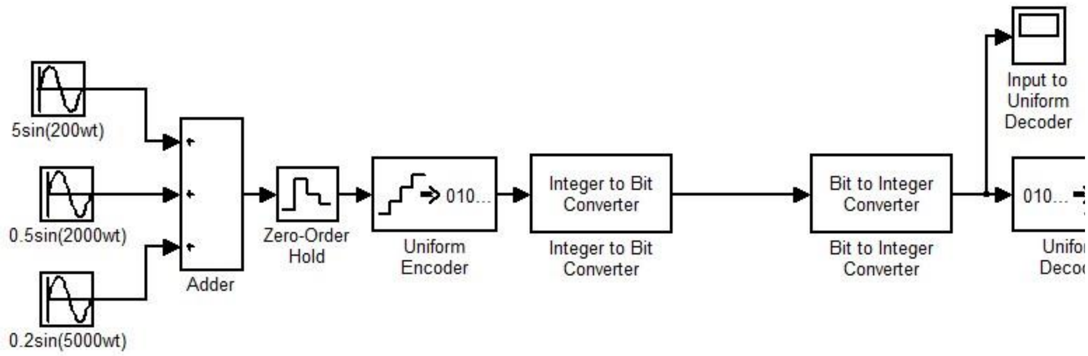


Σχήμα Π.3.21: Σήμα εξόδου από τον μετατροπέα bit to integer

Π.3.3.6 Ανασύσταση σήματος

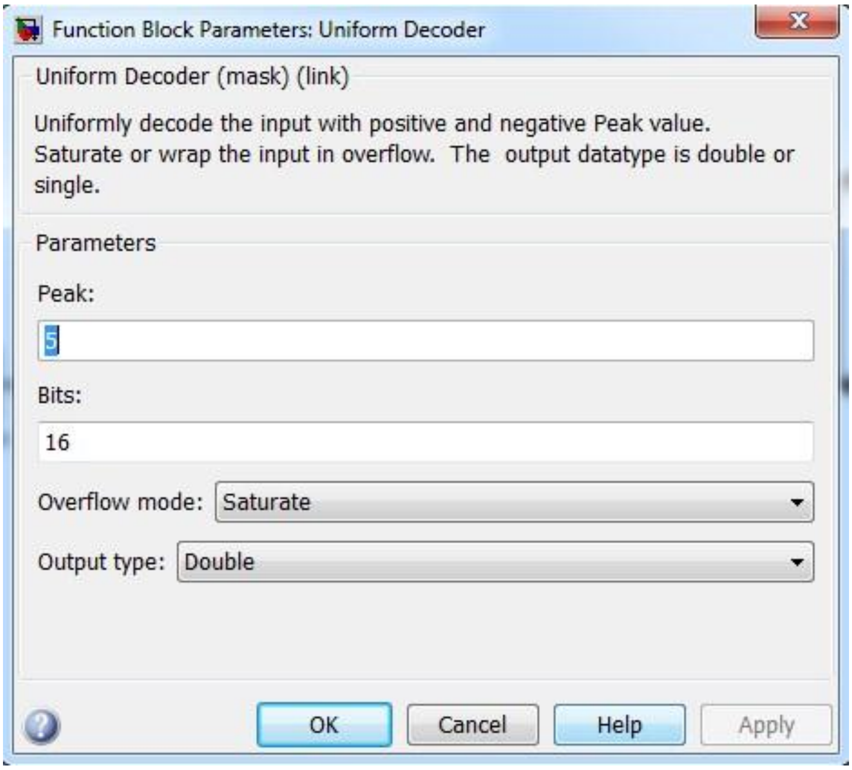
Στη φάση αυτή έχουμε ένα σήμα με στάθμες ακεραίων τιμών μεγέθους 16bit. Καλούμαστε λοιπόν να επαναφέρουμε το σήμα σε τιμές στο εύρος που είχε πριν την κβάντωσή του.

Για αυτό θα χρησιμοποιήσουμε έναν μπλοκ αποκωδικοποιητή που ονομάζεται "Uniform Decoder" και θα το βρούμε στη βιβλιοθήκη "DSP System Toolbox/Quantizers/Uniform Decoder. Θα το επιλέξουμε και θα το σύρουμε στο μοντέλο. Έπειτα θα συνδέσουμε τα μπλοκ όπως παρουσιάζεται στο σχήμα Π.3.22.



Σχήμα Π.3.22: Διάγραμμα 6^{ου} μέρους

Κάνουμε διπλό κλικ πάνω στο μπλοκ Uniform Decoder για να ρυθμίσουμε τις παραμέτρους. Αλλάζουμε τις τιμές στα πεδία όπως φαίνεται στο σχήμα 6.2.

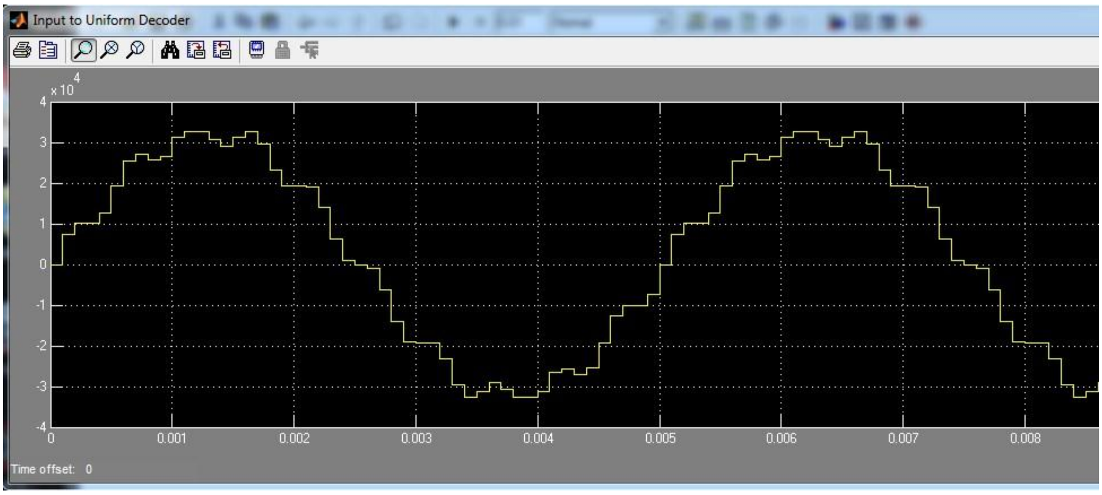


Σχήμα Π.3.23: Παράμετροι του μπλοκ Uniform Decoder

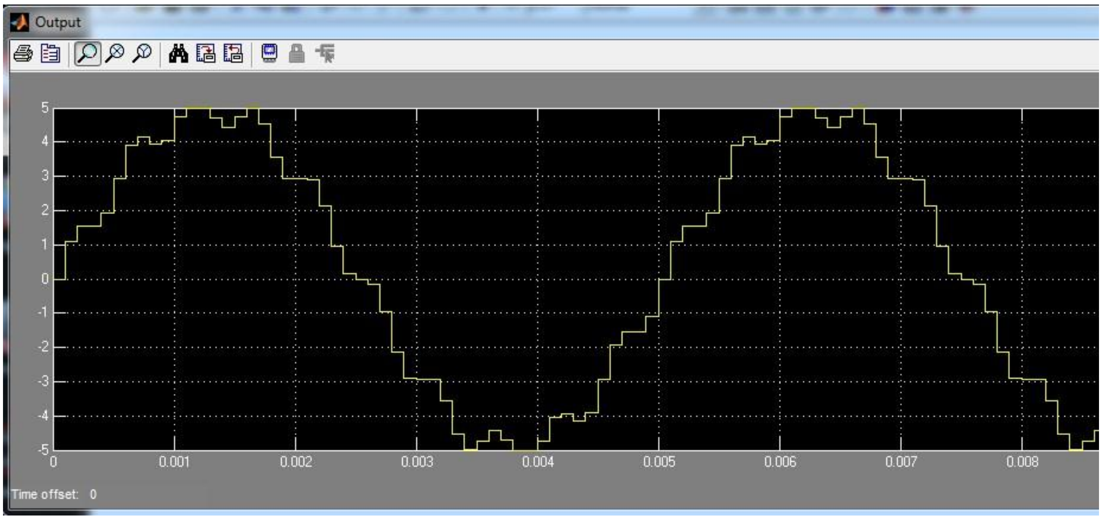
Στο πεδίο “Peak” βάζουμε τη τιμή 5 που αντιστοιχεί στο peak τιμή του σήματος εισόδου του μοντέλου. Ενώ στο πεδίο “Bits” βάζουμε τη τιμή 16 που αντιστοιχεί στα 16bit πληροφορίας που έχουμε.

Με αυτό τον τρόπο καταφέρνουμε να αντιστοιχήσουμε τις κβαντισμένες στάθμες των 16bit σε τιμές από -5 έως 5 (που είναι Volt). Κάνουμε δηλαδή mapping των τιμών της εισόδου του μπλοκ σε τιμές από -5 έως 5, ώστε να βγει το νέο σήμα στην έξοδο.

Στη συνέχεια μπορούμε να κάνουμε προσομοίωση ώστε να δούμε τα αποτελέσματα. Ανοίγοντας τα Scope μπορούμε να παρατηρήσουμε τις γραφικές παραστάσεις της εισόδου και της εξόδου του μπλοκ που μελετάμε.



Σχήμα Π.3.23: Σήμα εισόδου στο μπλοκ Uniform Decoder



Σχήμα Π.3.24: Σήμα εξόδου από το μπλοκ Uniform Decoder

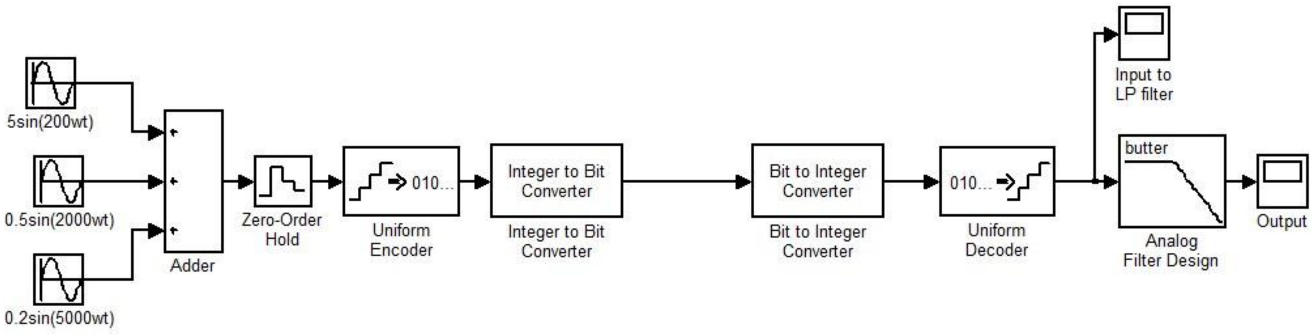
Παρατηρούμε στο σήμα εξόδου ότι έχει πάρει τιμές στο εύρος που επιλέξαμε. Τώρα το σήμα είναι έτοιμο για φιλτράρισμα ώστε να πάρουμε τις συνιστώσες που επιθυμούμε από αυτό.

Π.3.3.7 Φιλτράρισμα σήματος

Η τελευταία διαδικασία που πρέπει να κάνουμε να φιλτράρουμε το σήμα που πήραμε από το 6^{ο} μέρος ώστε να λάβουμε τελικά το θεμελιώδες σήμα που δώσαμε στην είσοδο, δηλαδή το 5sin(200ωt).

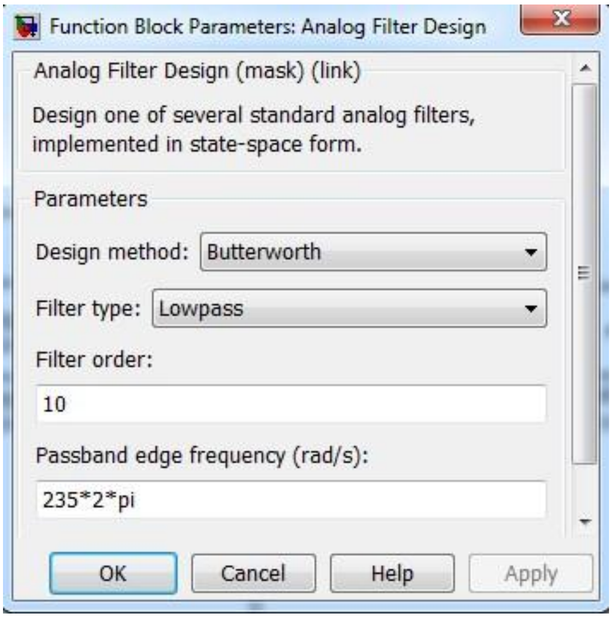
Τα σήματα των αρμονικών συνιστωσών είναι συχνότητας 2kHz και 5kHz και πρέπει να τα αποκόψουμε από το αρχικό σήμα. Αυτό θα γίνει με ένα χαμηλοπερατό αναλογικό φίλτρο (LowPass Filter). Ρυθμισμένο σε συχνότητα λίγο πάνω από τη θεμελιώδη συχνότητα των 200Hz.

Επιλέγουμε το μπλοκ “Analog Filter Design” από τη βιβλιοθήκη “DSP System Toolbox/Filter Implementation/Analog Filter Design” και το σέρνουμε στο μοντέλο. Συνδέουμε τα μπλοκ όπως φαίνεται στο σχήμα Π.3.25.



Σχήμα Π.3.25: Διάγραμμα 7^{ου} μέρους

Πατώντας διπλό κλικ πάνω στο μπλοκ μπαίνουμε στο παράθυρο των παραμέτρων και αλλάζουμε τις τιμές στα πεδία όπως φαίνεται στο σχήμα Π.3.26.



Σχήμα Π.3.26: Παράμετροι του μπλοκ Analog Filter Design

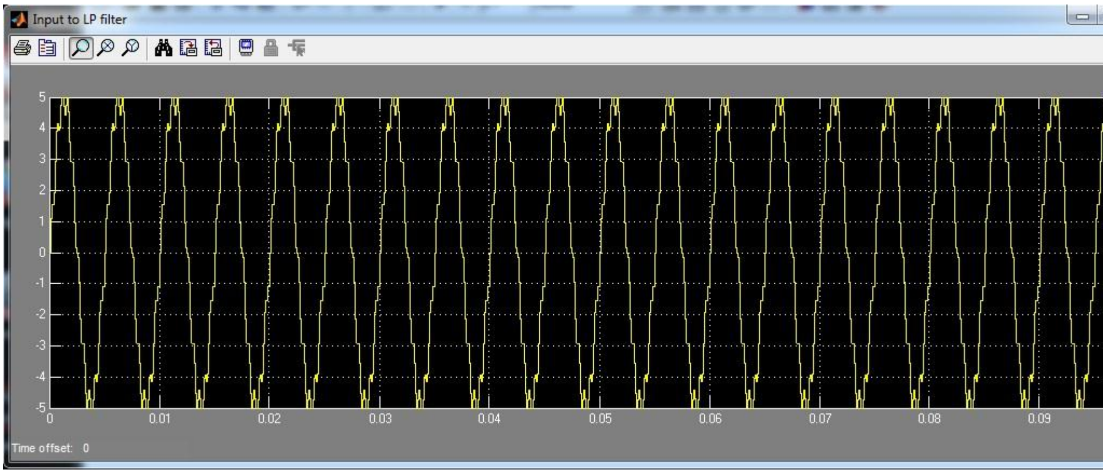
Βάζουμε στο πεδίο “filter order”, που είναι η τάξη του φίλτρου, την τιμή 10 που είναι ένα αρκετά καλό νούμερο για τη δουλειά που το θέλουμε. Όσο μεγαλύτερης τάξης είναι ένα φίλτρο τόσο πιο περίπλοκο είναι στη κατασκευή του αλλά έχει καλύτερη απόκριση.

Στο πεδίο “Passband edge frequency(rad/s)” βάζουμε τη τιμή της συχνότητας από τη οποία θέλουμε να κόβει το φίλτρο. Μετά από δοκιμές καταλήξαμε ότι η τιμή των 235Hz είναι ιδανική.

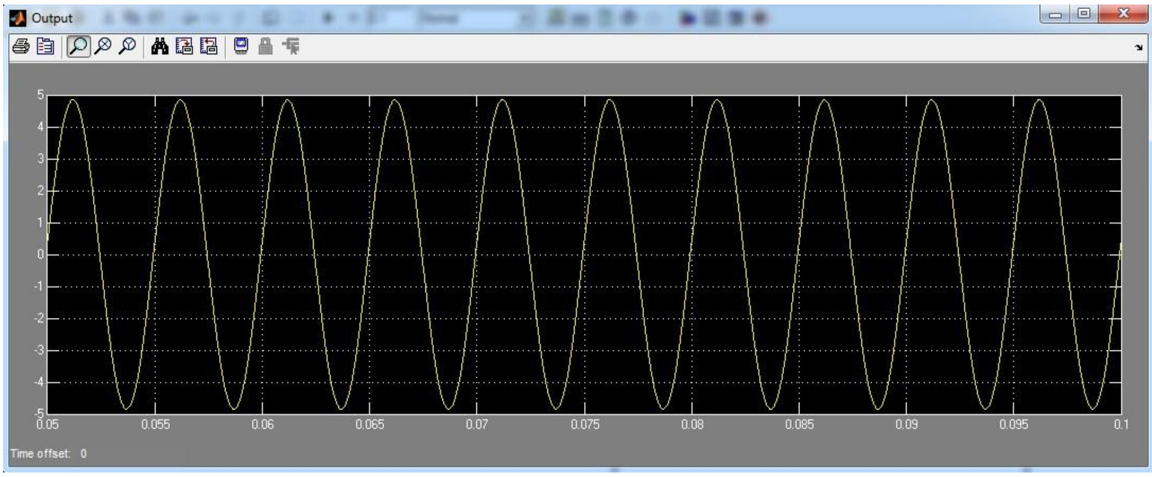
Μπορείτε να κάνετε δοκιμές αλλάζοντας τις τιμές και των δύο πεδίων ώστε να παρατηρήσετε τις διαφορές απόκρισης που θα προκύψουν.

Στη συνέχεια τρέχουμε τη προσομοίωση με χρόνο 0.1s και παρατηρούμε τα αποτελέσματα στα Scope πατώντας επάνω τους διπλό κλικ.

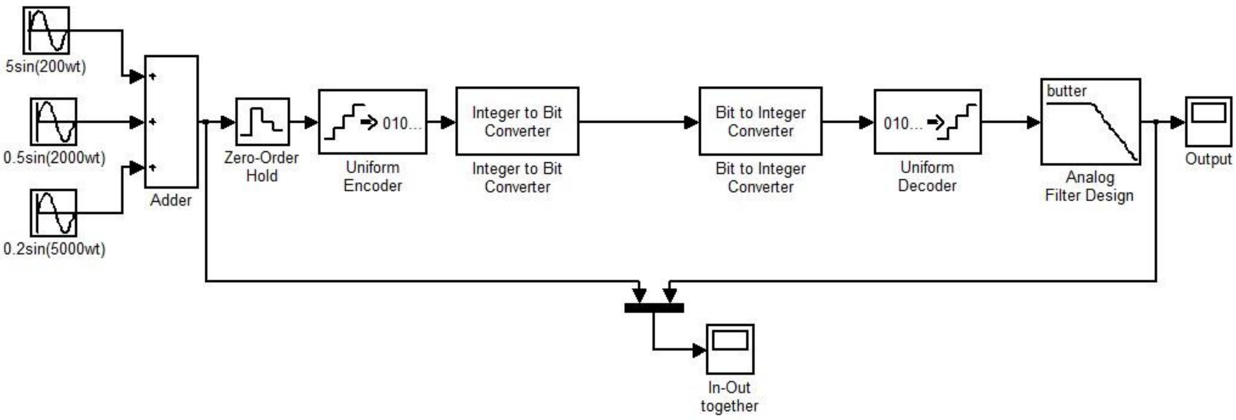
Αν εισάγετε στο μοντέλο ένα μπλοκ “mux” και ένα ακόμα Scope μπορείτε να παρατηρήσετε την είσοδο και την έξοδο του μοντέλου στους ίδιους άξονες. Ακολουθώντας τα επόμενα σχήματα μπορούμε να συγκρίνουμε την είσοδο με τις αρμονικές και την έξοδο του μοντέλου καθώς και το θεμελιώδες σήμα εισόδου με την έξοδο του μοντέλου για να δούμε πόση διαφορά έχει το ένα με το άλλο.



Σχήμα Π.3.27: Σήμα εισόδου στο φίλτρο



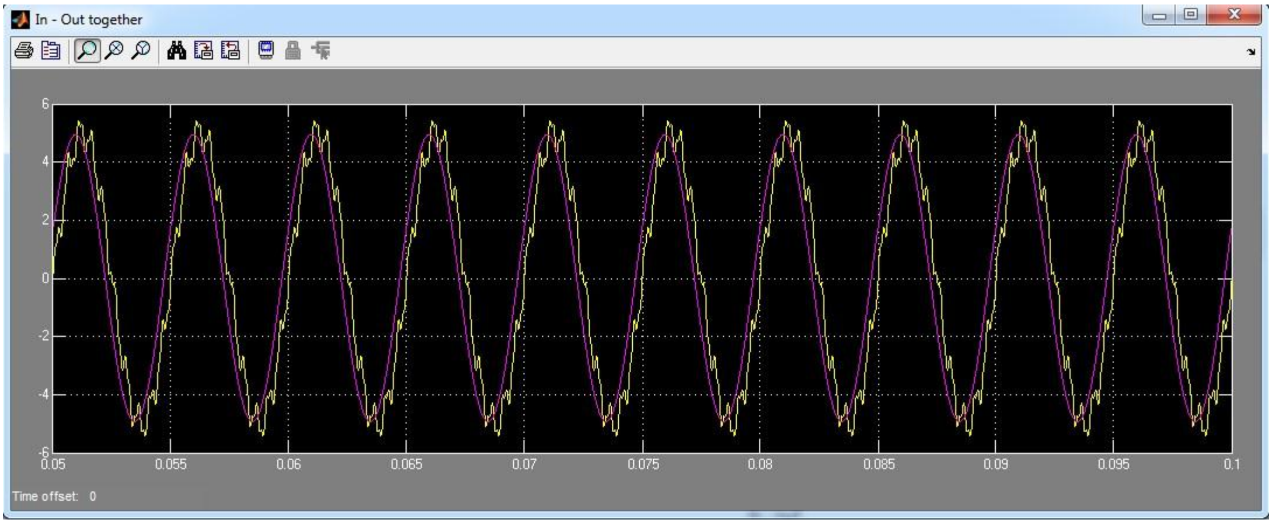
Σχήμα Π.3.28: Σήμα εξόδου από το φίλτρο



Σχήμα Π.3.29: Διάγραμμα 7^{ου} μέρους – συνέχεια 1

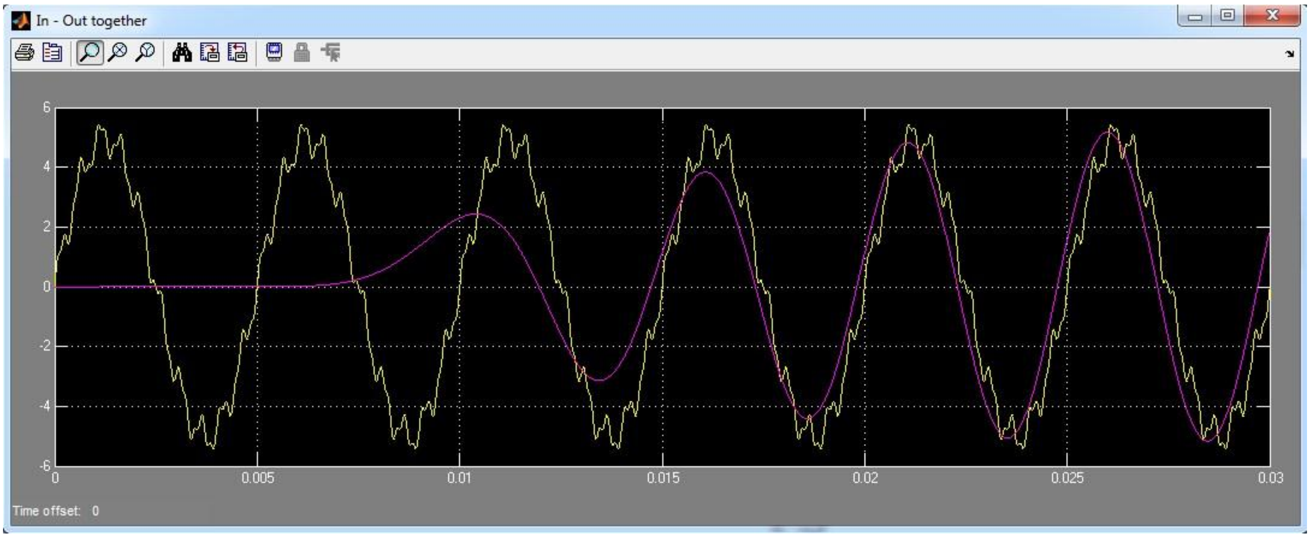
Τρέχοντας την προσομοίωση με χρόνο 0.1s παίρνουμε τα παρακάτω αποτελέσματα στο Scope που η είσοδος και η έξοδος είναι μαζί.

Το σήμα εισόδου στα επόμενα σχήματα είναι το κίτρινο ενώ το σήμα εξόδου το μωβ.



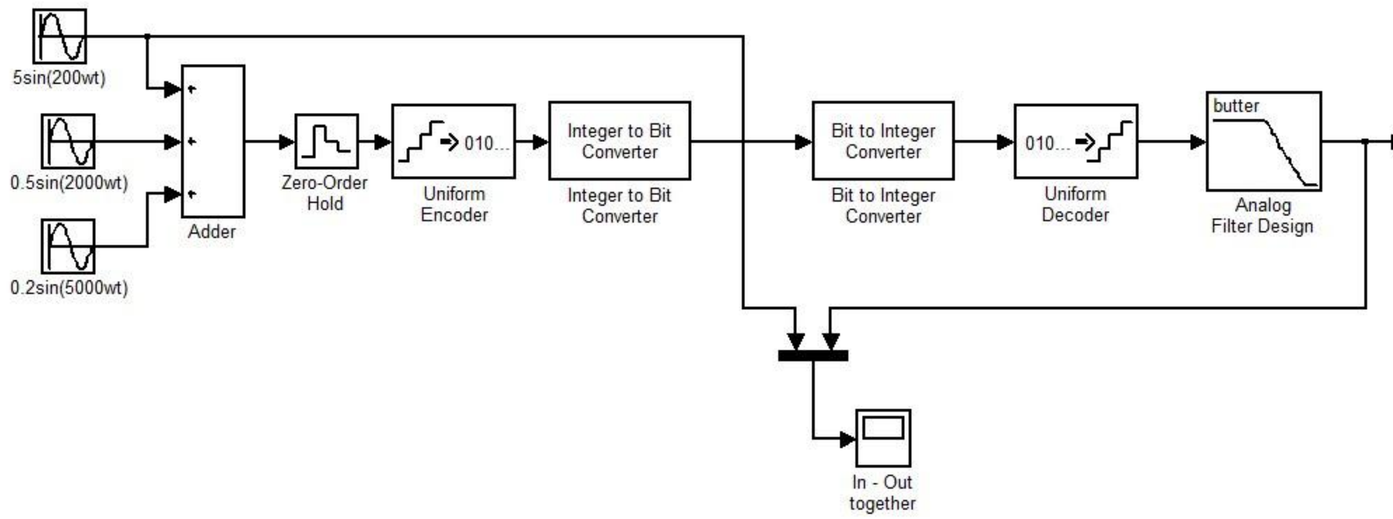
Σχήμα Π.3.30: Σήμα εισόδου και εξόδου του μοντέλου

Τρέχοντας την προσομοίωση με χρόνο 0.03s παίρνουμε τα παρακάτω αποτελέσματα:



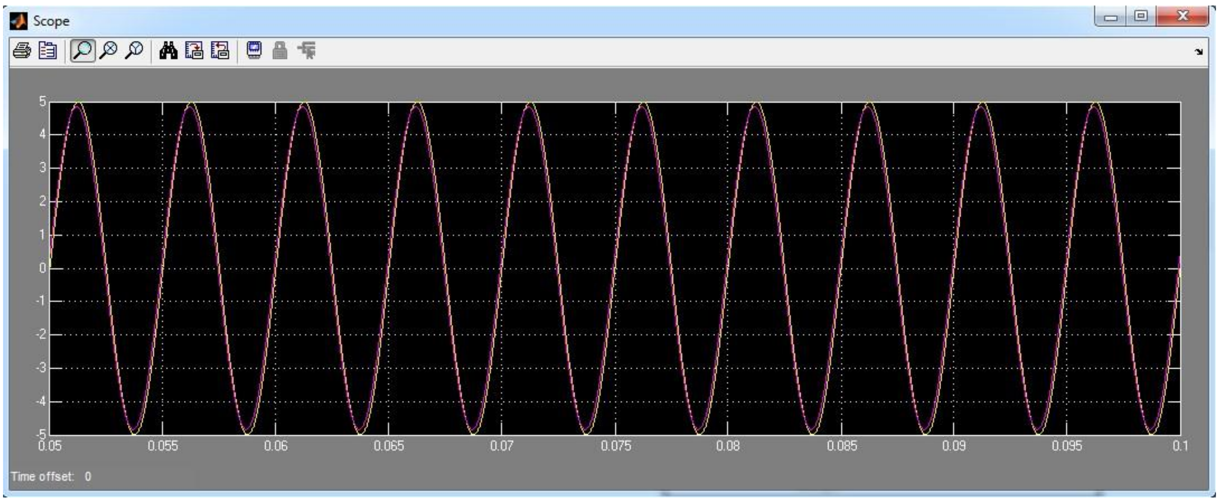
Σχήμα Π.3.31: Σήμα εισόδου και εξόδου του μοντέλου

Έπειτα συνδέουμε το Scope με την θεμελιώδη είσοδο των 200Hz όπως φαίνεται στο σχήμα Π.3.32 και τρέχουμε την προσομοίωση πάλι με χρόνο 0.1s και 0.03s.

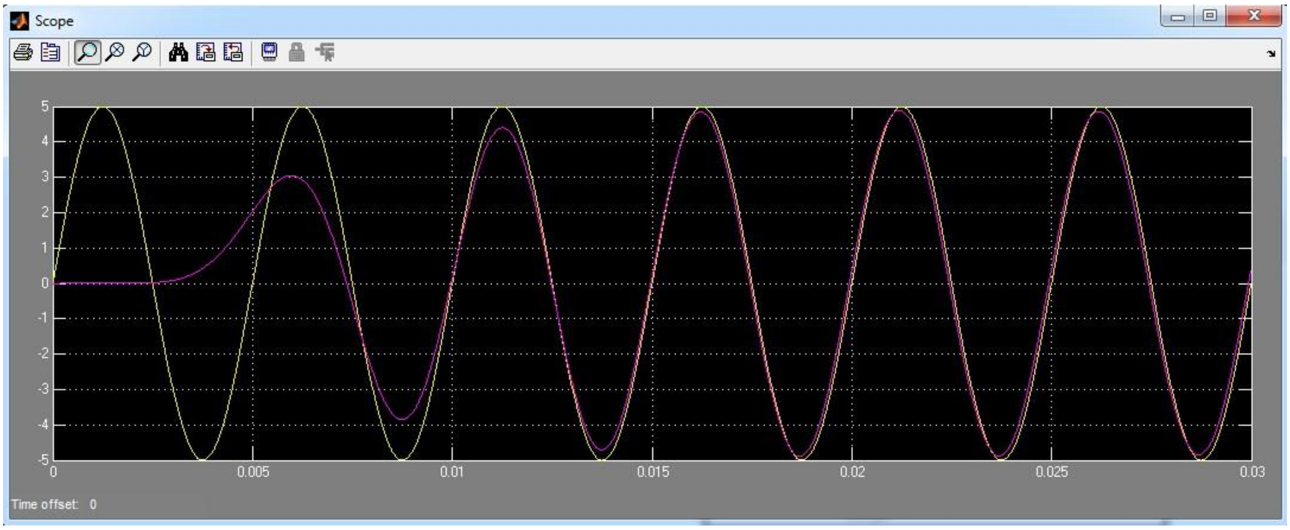


Σχήμα Π.3.32: Διάγραμμα 7ου μέρους – συνέχεια 2

Στα επόμενα σχήματα φαίνονται τα αποτελέσματα για χρόνους προσομοίωσης 0.1s και 0.03s αντίστοιχα:



Σχήμα Π.3.33: Σήμα εισόδου και εξόδου του μοντέλου



Σχήμα Π.3.34: Σήμα εισόδου και εξόδου του μοντέλου

Τα συμπεράσματα που μπορούμε να βγάλουμε είναι βάση της τελευταίας προσομοίωσης που βλέπουμε ότι η διαφορά της εισόδου με την έξοδο είναι μια πολύ μικρή διαφορά φάσης λόγο του αναλογικού φίλτρου που χρησιμοποιήσαμε. Επίσης υπάρχει μια μικρή απόκλιση από το πλάτος του σήματος εισόδου, αλλά φαίνεται αμελητέα.

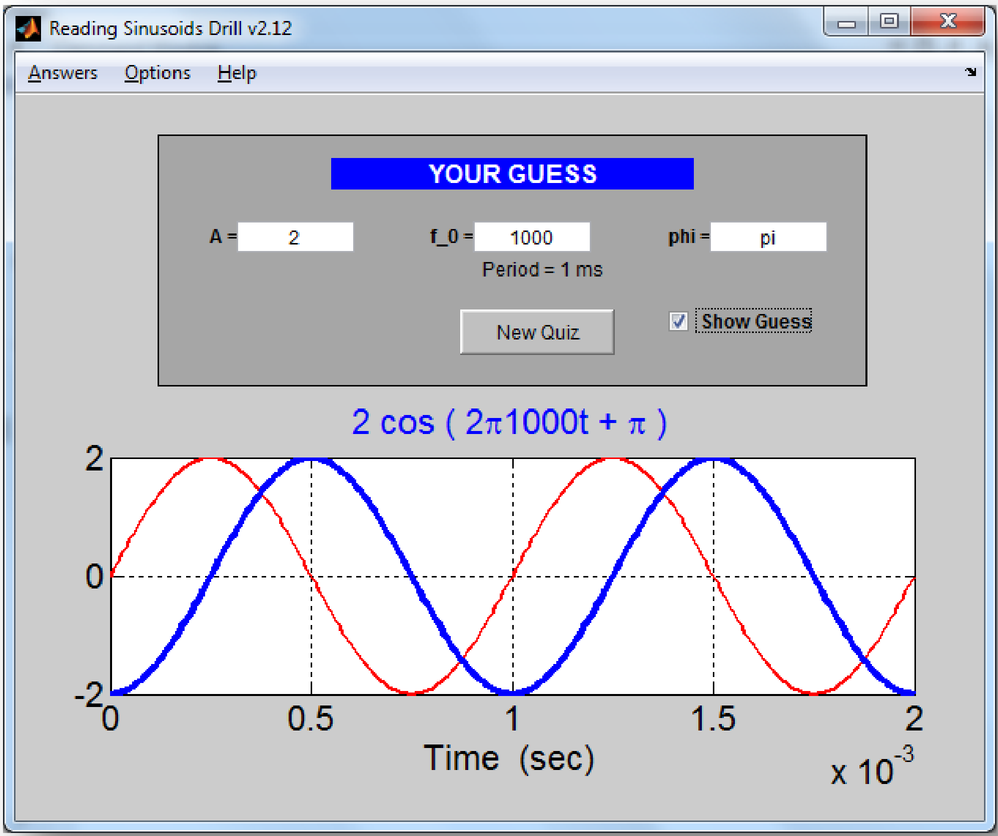
ΟΡΓΑΝΩΣΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΕ MATLAB

ΠΑΡΑΡΤΗΜΑ 4

EDUCATIONAL MATLAB GUIS

Στην ηλεκτρονική διεύθυνση http://users.ece.gatech.edu/mcclella/matlabGUIs/ βρίσκονται μερικά χρήσιμα **user interfaces** του MATLAB για τη θεωρία των Σημάτων και Συστημάτων όπως Filter Design Demo, **PEZdemo**, FourierSeriesDemo, PhasorRaces, **SinDrill**, **ZDrill**, **CLTIDemo**, DLTIDemo, Continuous-Discrete Sampling Demo, **Discrete Convolution Demo**, **Continuous Convolution Demo**.

Π.4.1 SinDrill Graphical User Interface



Το πρόγραμμα αυτό ελέγχει την ικανότητα των χρηστών να καθορίσουν τις βασικές παραμέτρους ενός ημιτονοειδούς σήματος.

Μετά την εμφάνιση ενός ημιτονοειδούς σήματος, ο χρήστης πρέπει να μαντέψει το πλάτος την συχνότητα και την φάση του.

Η γενική μορφή ενός ημιτονοειδούς σήματος είναι :

$$\begin{aligned} y(t) &= A\cos(2\pi f t + \phi) \\ &= A\cos(\omega t + \phi) \end{aligned}$$

Όπου : A = Πλάτος , f = συχνότητα και Φ = φάση.

Από την απεικόνιση του σήματος μπορεί κανείς να προσδιορίσει εύκολα αυτές τις παραμέτρους . Το πλάτος είναι απλά η μέγιστη τιμή που παίρνει το σήμα. Η συχνότητα μπορεί να υπολογισθεί από τον τύπο 1/T όπου T = περίοδος (το χρονικό διάστημα μεταξύ δύο ισοδύναμων σημείων της κυματομορφής). Η φάση μπορεί να προσδιοριστεί με τον υπολογισμό 2*π*συχνότητα*καθυστέριση όπου καθυστέρηση είναι το χρονικό διάστημα μέχρι το πρώτο peak.

SinDrill Controls

Δημιουργώντας μια νέα ερώτηση:

Όταν το πρόγραμμα ξεκινά ή όταν το κουμπί «New Quiz» πατηθεί, ένα ημιτονοειδές σήμα εμφανίζετε. Ο χρήστης θα πρέπει να καθορίσει το πλάτος, τη συχνότητα και τη φάση του ημιτονοειδούς σήματος εισάγοντας στα κατάλληλα πλαίσια πάνω από την κυματομορφή τα κατάλληλα νούμερα .

Αλλαγή Πρόβλεψης:

Το πλάτος, η συχνότητα, η φάση του ημιτονοειδούς σήματος μπορεί να αλλάξει εισάγοντας τιμές στα πλαίσια και έπειτα πατώντας το κουμπί Enter. Μπορείτε επίσης να εισάγετε οποιαδήποτε έκφραση Matlab, όπως π / 2.

Επισκόπηση Πρόβλεψης:

Όταν το «Show Guess» είναι επιλεγμένο, δημιουργείτε ένα ημιτονοειδές σήμα από τα στοιχεία πλάτος, συχνότητα και φάση τα οποία έχει εισάγει ο χρήστης και εμφανίζετε στην κυματομορφή που βρίσκετε από κάτω.

Έλεγχος απάντησης:

Πάνω αριστερά στο Menu μας δίνεται η δυνατότητα να ελέγξουμε την απάντηση. Ετσι πατώντας το κουμπί Answer εμφανίζετε ένα dropdown μενού με το σωστό πλάτος την σωστή συχνότητα και την σωστή φάση.

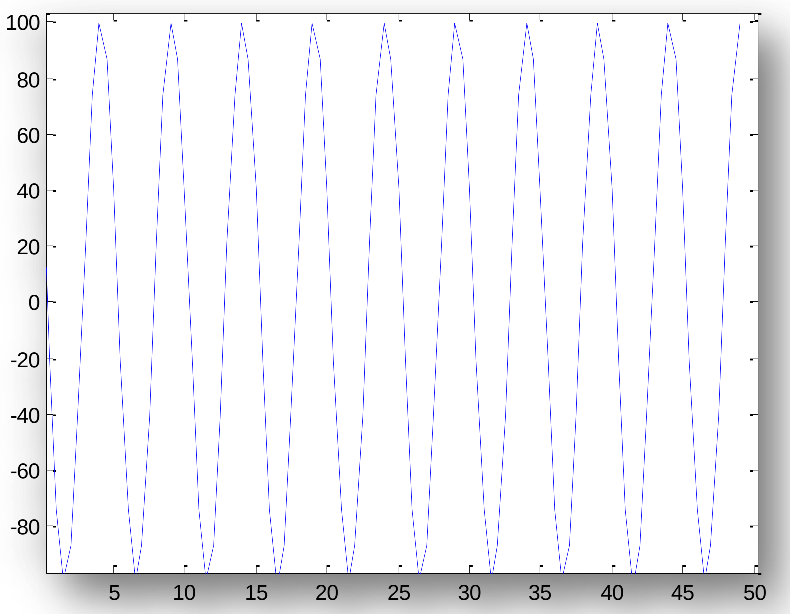
Παράδειγμα:

Συχνότητα = 0,2Hz

Πλάτος = 100

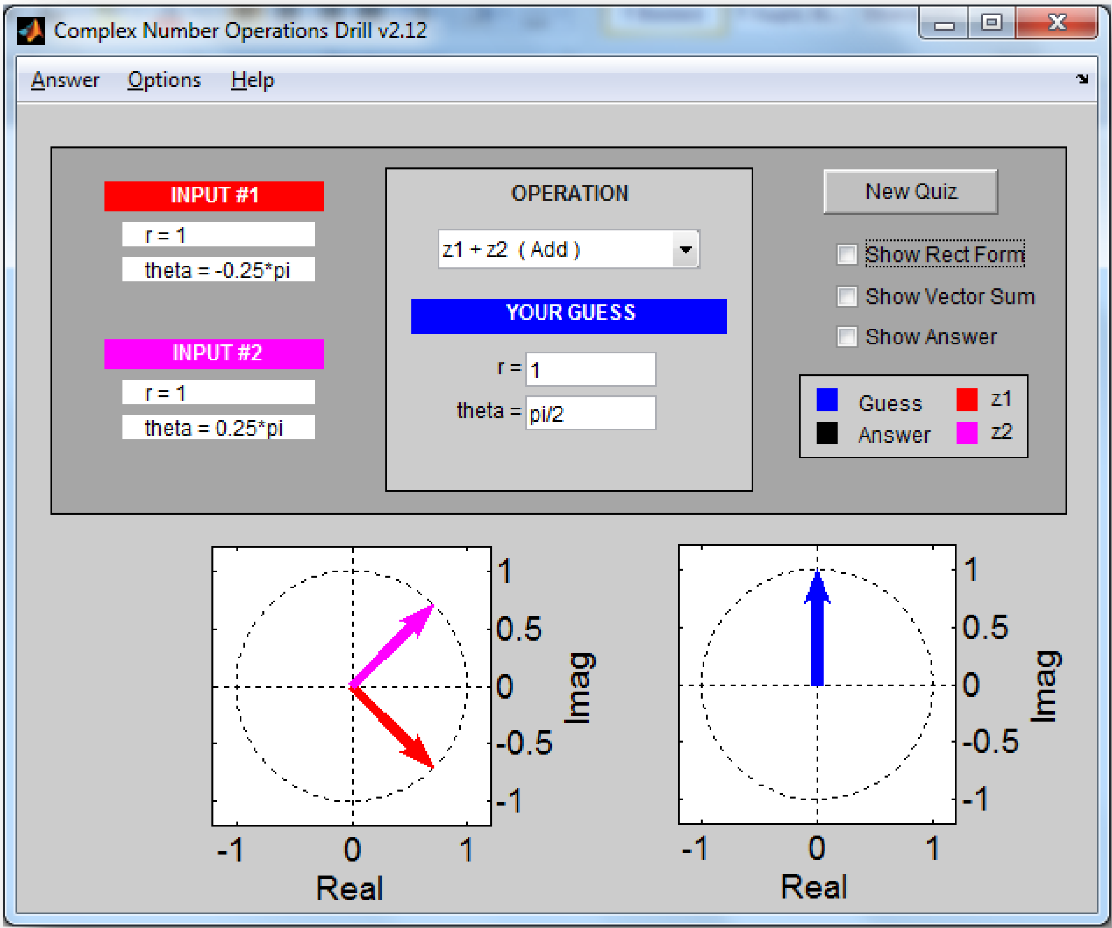
Φάση = 20

```
>> f = 0.2;
>> A = 100;
>> fasi = 20;
>> fs = 2;
>> t = 0:1/fs:49;
>> x = A*cos(2*pi*f*t + fasi);
>> plot(t,x)
```



Π.4.2 ZDrill Graphical User Interface

Το πρόγραμμα αυτό ελέγχει την ικανότητα των χρηστών να υπολογίζουν το αποτέλεσμα των απλών πράξεων των μιγαδικών αριθμών. Οι ακόλουθες έξι λειτουργίες υποστηρίζονται: Πρόσθεση, Αφαίρεση, Πολλαπλασιασμός, Διαίρεση, Αντιστροφή, Σύζευξη. Το πρόγραμμα δίνει έμφαση στην διανυσματική εμφάνιση ενός μιγαδικού αριθμού.



Η γενική μορφή για ένα μιγαδικό αριθμό είναι:

$$z = x + j*y \qquad \text{(Τριγωνομετρική μορφή)}$$

$$z = r * \exp(j*theta) \quad \text{(πολική μορφή)}$$

όπου:

x = πραγματικό μέρος του z

y = Φανταστικό μέρος του z

r = Μέγεθος του z

theta = γωνία του z

Για τη μετατροπή μεταξύ τους μπορεί κανείς να χρησιμοποιήσει τις ακόλουθες εξισώσεις:

x = r * cos (theta)

y = r * sin (theta)

r = sqrt (x ^ 2 + y ^ 2)

theta = atan2 (y, x

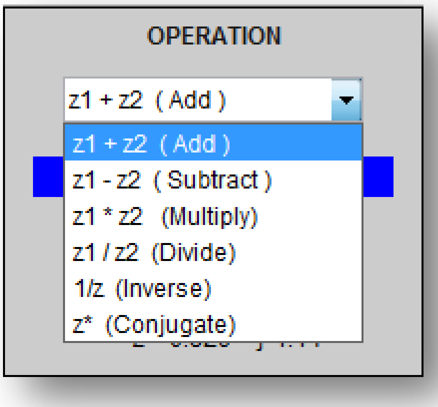
ZDrill Controls

Δημιουργώντας μια νέα ερώτηση:

Όταν το πρόγραμμα ξεκινά ή όταν το κουμπί «New Quiz» πατηθεί, δυο μιγαδικοί αριθμοί Z1 και Z2 σχηματίζονται αυθαίρετα. Οι παράμετροι τους δίδονται στην αριστερή στήλη της οθόνης και το διάγραμμα τους απεικονίζετε στο κάτω αριστερό γράφημα. (Σημείωση: μόνο το Z1 εμφανίζεται αν η λειτουργία είναι η αντίστροφη ή το συζυγές). Ο χρήστης θα πρέπει να καθορίσει την ακτίνα και την γωνία του μιγαδικού ο οποίο προκύπτει από το άθροισμα την πράξης που έχουμε επιλέξει στο Dropdown menu «Operation».

Αλλαγή της λειτουργίας:

Μπορούμε να αλλάξουμε τη λειτουργία όπου θα γίνει στους μιγαδικούς αλλάζοντας την επιλογή στο Drop Down Menu “Operation”.



Αλλαγή της εικασίας μας:

Η ακτίνα και η γωνία της εικασίας μας μπορεί να αλλάξει χρησιμοποιώντας τα δύο πεδία που βρίσκονται στη μέση της οθόνης κάτω από το μπλε πλαίσιο «Your Guess». Αφού συμπληρώσουμε αυτά τα δύο πεδία τότε στο κάτω δεξιά διάγραμμα θα εμφανιστεί το διάνυσμα της εικασίας μας. Σημείωση: μπορούμε να χρησιμοποιήσετε οποιαδήποτε έκφραση Matlab, όπως π / 2 σε αυτά τα πλαίσια.

Έλεγχος της απάντησης:

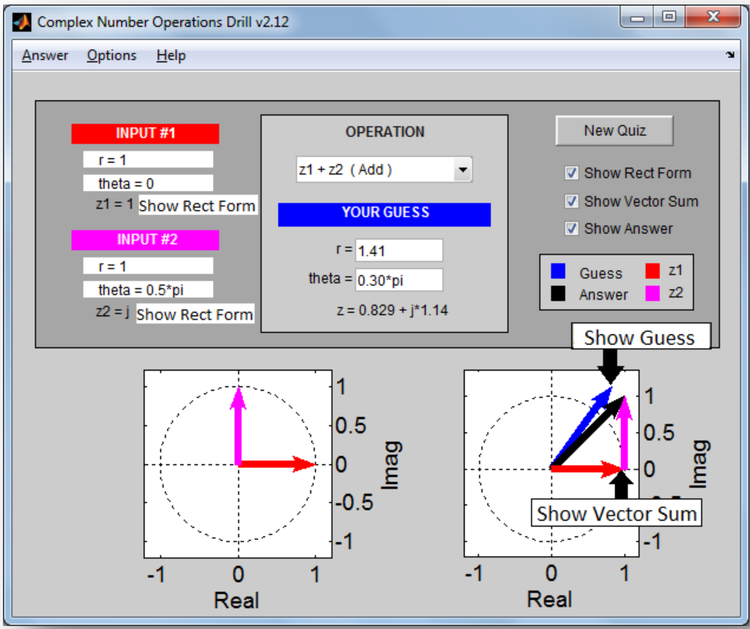
Η απάντηση για την τρέχουσα ερώτηση δίνεται πάντα στο Drop Down μενού «Answer» που βρίσκετε πάνω αριστερά . Εάν το πεδίο "Show Answer» είναι επιλεγμένο, τότε η απάντηση θα πρέπει να εμφανίζετε στο κάτω δεξιά γράφημα.

Εμφάνιση διανυσματικού αθροίσματος:

Αν η λειτουργία είναι πρόσθεση ή αφαίρεση, το πεδίο «Show Vector Sum» εμφανίζετε. Επιλέγοντας αυτό το πλαίσιο, το διανυσματικό άθροισμα ή η διαφορά θα απεικονίζετε στο γράφημα κάτω δεξιά. Μερικές φορές είναι χρήσιμο να έχουμε επιλεγμένο το πεδίο «Show answer" για να βλέπουμε πώς προκύπτει το διανυσματικό άθροισμα / διαφορά .

Εμφάνιση Τριγωνομετρικής Μορφής:

Όταν η επιλογή «Show Rect Form» είναι επιλεγμένη, η τριγωνομετρική μορφή των αριθμών εισόδου και η εικασία μας εμφανίζεται κάτω από το αντίστοιχο πλαίσιο της πολική μορφής.

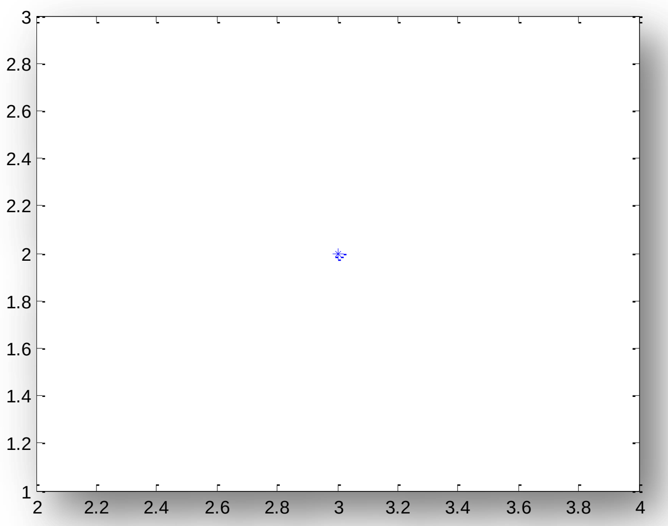


Παράδειγμα MATLAB:

Ορίστε το μιγαδικό αριθμό z = 3 + 2i και εμφανίστε το ως ένα σημείο στο μιγαδικό επίπεδο.

Εντολές:

```
>> z = complex(3, 2);
>> plot(real(z), imag(z), '*');
```



Π.4.3 CLTIDemo Graphical User Interface

Το πρόγραμμα αυτό απεικονίζει την σχέση μεταξύ της εισόδου και εξόδου ενός γραμμικού φίλτρου (LTI), όταν η είσοδος είναι μια ημιτονοειδής συνάρτηση. Ο χρήστης έχει τη δυνατότητα να ελέγχει τις παραμέτρους τόσο της ημιτονοειδούς εισόδου όσο και του φίλτρου.

Η γενική μορφή της ημιτονοειδούς εισόδου (συνεχούς χρόνου) είναι:

Input = A + B*cos(2*pi*f0*t + phi)

Όπου:

A = DC Level

B = Πλάτος ημιτονοειδούς

f0 = Συχνότητα ημιτονοειδούς

phi = Φάση ημιτονοειδούς

Επειδή το φίλτρο είναι ένα φίλτρο LTI, η έξοδος είναι :

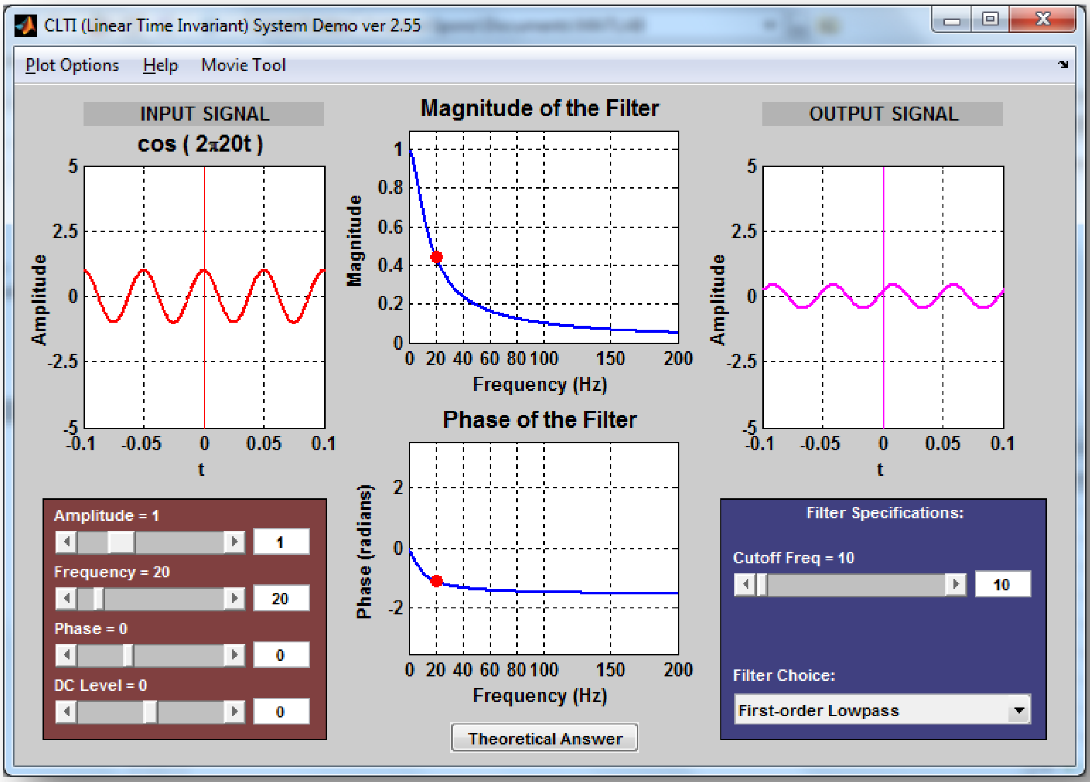
Output = A*D + B*M*cos(2*pi*f0*t + phi + P)

Όπου:

D = H DC απόκριση του φίλτρου, δηλαδή, η απόκριση συχνότητας στο f = 0.

M = Μέγεθος της απόκριση συχνότητας του φίλτρου που υπολογίζεται σε f = f0.

P = Φάση της απόκρισης συχνότητας του φίλτρου που υπολογίζεται σε f = f0.

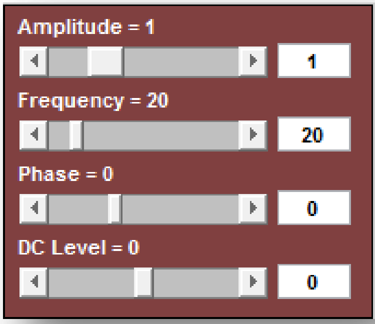


Σημειώστε ότι η έξοδος του φίλτρου εξακολουθεί να είναι ημιτονοειδής. Στην πραγματικότητα, πρόκειται για ένα ημιτονοειδές σήμα με την ίδια συχνότητα όπως η ημιτονοειδής είσοδος. Το φίλτρο αλλάζει μόνο στο επίπεδο DC, το μέγεθος, και η φάση του ημιτονικού σήματος εισόδου.

CLTIDemo Controls

Αλλαγή της ημιτονοειδούς εισόδου:

Το πλάτος, η συχνότητα, η φάση, και το DC level του ημιτονοειδούς σήματος εισόδου μπορεί να αλλάξει με τα slider αλλά και με τα πεδία που βρίσκονται στο κάτω αριστερό μέρος της οθόνης. Οι τιμές αυτές αλλάξουν τη γραφική παράσταση του σήματος εισόδου πάνω αριστερά.



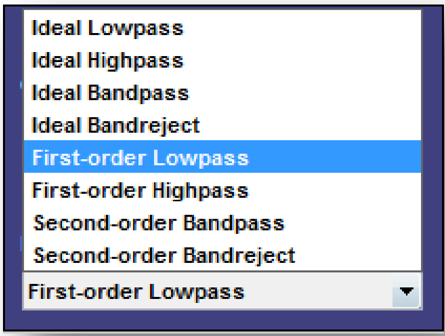
Αλλαγή του φίλτρου:

Χρησιμοποιώντας το dropdown πλαίσιο στην κάτω δεξιά γωνία της οθόνης, ο χρήστης μπορεί να επιλέξει από μια προκαθορισμένη λίστα φίλτρων. Ο χρήστης μπορεί στη συνέχεια να αλλάξετε τις παραμέτρους του φίλτρου αλλάζοντας τα πεδία που βρίσκονται κάτω από το «Filter Specification». Οι παράμετροι που ο χρήστης επιτρέπεται να αλλάξει θα εξαρτηθούν από το φίλτρο.

Η απόκριση συχνότητας του φίλτρου εμφανίζεται στα δύο κεντρικά γραφήματα. Η πάνω γραφική παράσταση είναι το μέτρο της απόκρισης συχνότητας. Ενώ το κάτω κεντρικό γράφημα είναι η φάση της απόκρισης συχνότητας.

Οι μικροί στρογγυλοί δείκτες στα διαγράμματα απόκρισης συχνότητας δείχνουν την απόκριση του φίλτρου στην ημιτονοειδή είσοδο για συχνότητα f0.

Εάν το επίπεδο της DC εισόδου είναι μη μηδενικό, τότε ένας άλλος δείκτης θα εμφανιστεί δείχνοντας την απόκριση του φίλτρου DC.

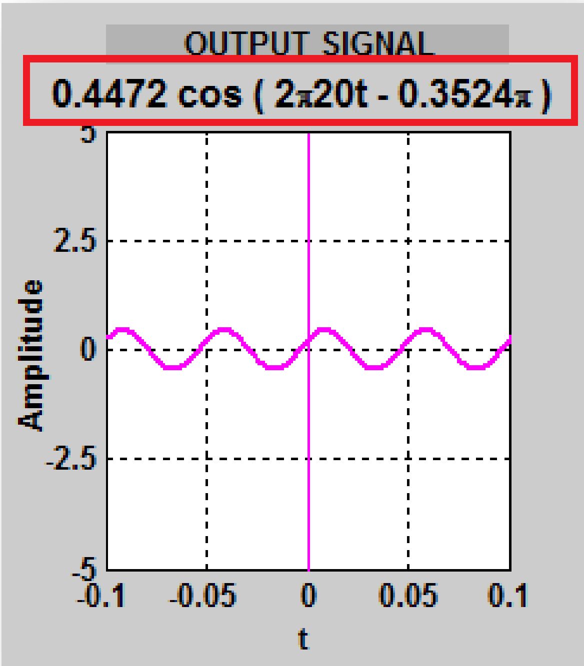


Κουμπί θεωρητικής απάντησης:

Όταν το κουμπί θεωρητικής απάντησης πατηθεί, η φόρμουλα για το σήμα εξόδου θα εμφανιστεί πάνω από το γράφημα του σήματος εξόδου.

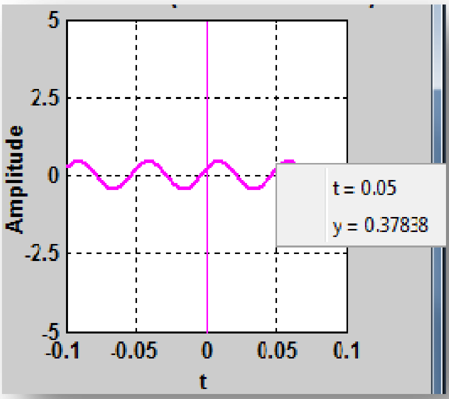
Ο τύπος για το σήμα εξόδου αρχικά είναι κρυμμένος για να ενθαρρύνει τον χρήστη να βρει την απάντηση από μόνος του χρησιμοποιώντας τα στοιχεία από τα άλλα γραφήματα.

Πιστεύουμε ότι όταν ο χρήστης μπορεί να σωστά και με συνέπεια αντλούν τον τύπο εξόδου χωρίς τη βοήθεια του κουμπιού Θεωρητική απάντηση, τότε ο χρήστης θα έχει μια πραγματική κατανόηση της θεωρίας αυτό το πρόγραμμα προσπαθεί να τονίσει.

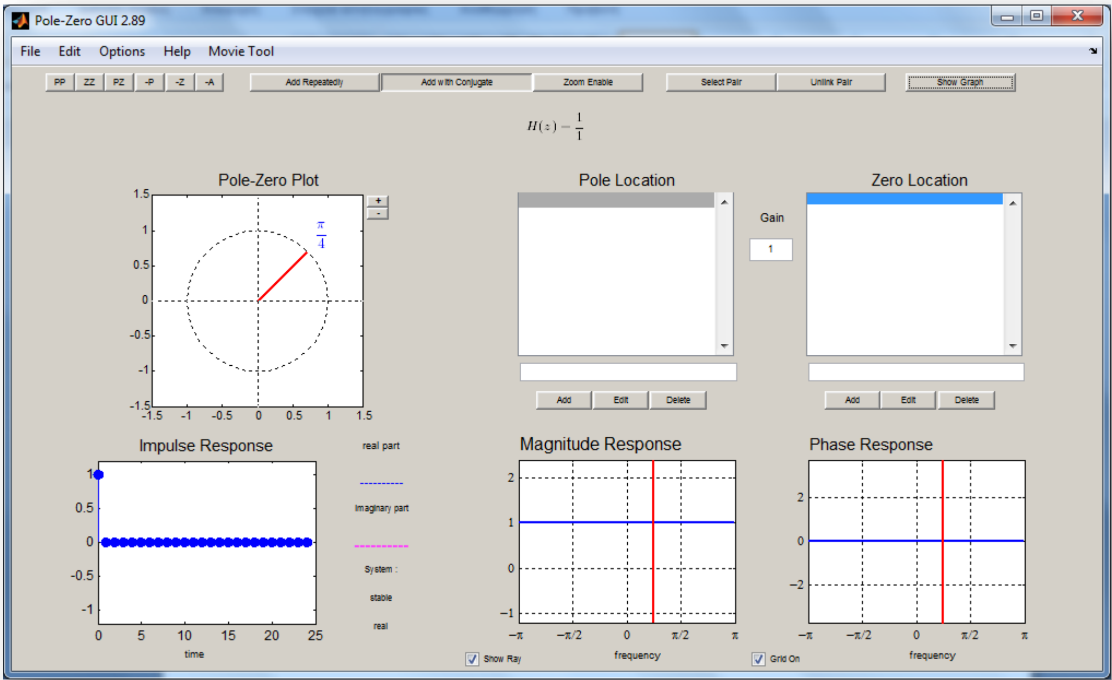


Ανάγνωση των τιμών από τις γραφικές παραστάσεις:

Κάνοντας δεξί κλικ σε μια τιμή της γραφικής παράστασης, ένα μικρό αναδυόμενο παράθυρο θα εμφανιστεί στη θέση του ποντικιού που θα δίνει τις ακριβείς τιμές x, y στο συγκεκριμένο σημείο . Αυτό είναι ιδιαίτερα χρήσιμο για να βρείτε τις ακριβείς τιμές της απόκρισης του φίλτρου στην ημιτονοειδή είσοδο .



Π.4.4 Pole-Zero Editor



Το Pole-Zero Editor είναι ένα εργαλείο το οποίο επιτρέπει στους χρήστες να δημιουργήσουν γραμμικά συστήματα διαμέσου ενός διαγράμματος μηδενικού πόλου. Η ένταση απόκρισης του γραμμικού συστήματος εμφανίζεται και ενημερώνεται καθώς ο χρήστης σύρει ατομικούς πόλους ή μηδενικά

Το Pole-Zero Editor μας βοηθά να συσχετίσουμε το πεδίο συχνότητας και το πεδίο-Ζ του συστήματος. Η κατανόηση αυτής της συσχέτισης βοηθά στην ερμηνεία των αποτελεσμάτων σε οποιοδήποτε πεδίο. Επίσης βοηθά στον προσδιορισμό της σταθερότητας του συστήματος με δεδομένη την συνάρτηση μεταφοράς H(z).

Ο μετασχηματισμός Fourier μίας ακολουθίας υπολογίζεται ως εξής:

$$X\left(e^{j\omega}\right)=\sum_{n=-\infty}^{\infty}x[n]e^{-j\omega n}$$

Και ο μετασχηματισμός - Ζ ως εξής:

$$X(z)=\sum_{n=-\infty}^{\infty}x[n]z^{-n}$$

Υπάρχει στενή σχέση μεταξύ αυτών των εξισώσεων. Αν αντικαταστήσουμε το z με e^{jw}, τότε ο μετασχηματισμός-z ανάγεται σε μετασχηματισμό Fourier. Όταν υπάρχει, ο μετασχηματισμός Fourier είναι απλά X(z) με z=e^{jw}. Παράδειγμα για |z| = 1, ο μετασχηματισμός-z αντιστοιχεί στον μετασχηματισμό Fourier. Αν όμως εκφράσουμε το z σε πολική μορφή, όπως

$$z = re^{j\omega}$$

Αυτός είναι ο μετασχηματισμός Fourier του προϊόντος της αρχικής ακολουθίας x[n] και της εκθετικής ακολουθίας r –n. Για r=1, αυτή είναι ο μετασχηματισμός Fourier.

Έτσι ο μετασχηματισμός Fourier και ο μετασχηματισμός-z ενός συστήματος μπορεί να δοθεί ως .

$$H(\omega) = \sum_{k=0}^{M} b_k e^{-j\omega k}$$

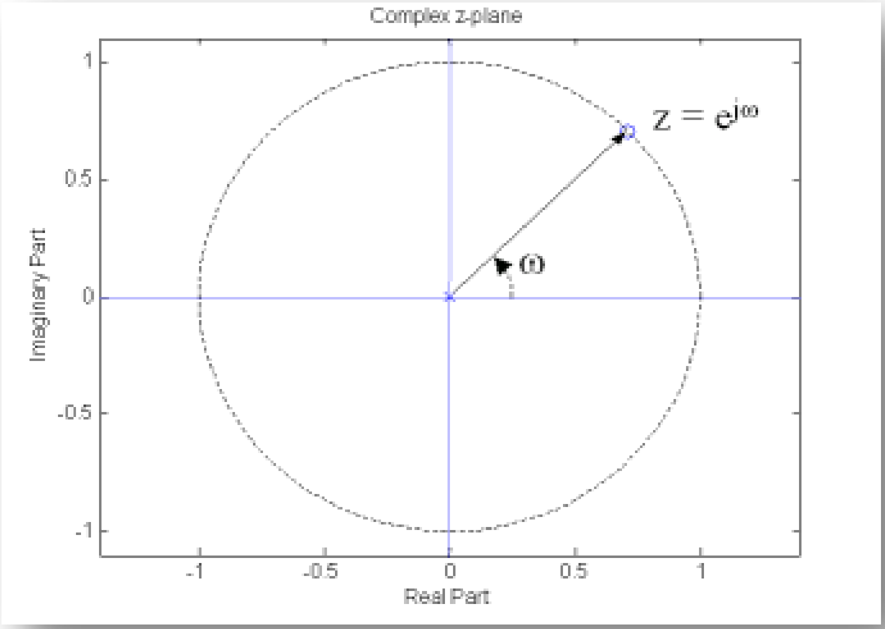
$$H(z) = \sum_{k=0}^{M} b_k z^{-k}$$

$$H(\omega) = H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}}$$

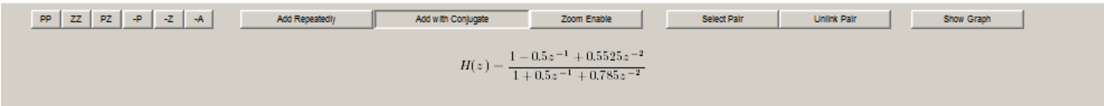
Απεικόνιση Pole-Zero:

Δεδομένου ότι ο ζ-μετασχηματισμός είναι συνάρτηση μιας σύνθετης μεταβλητής, είναι βολικό να το περιγράψουμε και να το ερμηνεύσουμε χρησιμοποιώντας το z-επίπεδο. Στο z-επίπεδο, το περίγραμμα αντιστοιχεί σε | z | = 1 είναι ένας κύκλος ακτίνας μίας μονάδας. Αυτό το περίγραμμα αναφέρεται ως ο μοναδιαίος κύκλος. Επίσης, ο z-μετασχηματισμός είναι περισσότερο χρήσιμος όταν το άπειρο άθροισμα μπορεί να εκφραστεί ως ένα απλό μαθηματικό τύπο.

Το διάγραμμα pole-zero μας δίνει ένα βολικό τρόπο απεικόνισης της σχέσης μεταξύ του τομέα συχνότητας και Ζ-τομέα. Η απόκριση συχνότητας $H(e^{jw})$ λαμβάνεται από τη συνάρτηση μεταφοράς H(z), με την αξιολόγηση της συνάρτησης μεταφοράς σε συγκεκριμένες τιμές $z = e^{jw}$. Δεδομένου ότι, η απόκριση συχνότητας είναι περιοδική με περίοδο 2π, θα πρέπει να το αξιολογήσουμε σε πάνω από μια περίοδο, όπως -p<w<p. Εάν αντικαταστήσουμε αυτές τις τιμές του w στο $z=e^{jw}$, τότε οι τιμές του z θα βρίσκονται στον μοναδιαίο κύκλο και θα κυμαίνονται από z = -1 σε όλη τη διαδρομή γύρω και πίσω στο σημείο z = -1. Αυτό φαίνεται στο παρακάτω. Από αυτό η περιοδικότητα 2π στον τομέα συχνοτήτων αντιστοιχεί σε μετακίνηση μέσω μιας γωνίας 2π στον μοναδιαίο κύκλο.



Pole-Zero Controls



Κουμπιά: PP, ZZ, PZ

Αυτή η επιλογή μπορεί να χρησιμοποιηθεί για την προσθήκη ενός ζευγαριού ή μηδενικών / πόλων ή και τα δύο, επιλέγοντας ένα από τα 3 κουμπιά,

PP - Pole

ZZ - Zero

PZ - Pole_Zero

Ένα κλικ του ποντικιού στο διάγραμμα pole-zero θα τοποθετήσει το αντίστοιχο πόλο / μηδέν σε αυτή τη θέση. Η επιλογή αυτή συνδέεται με την επιλογή «Add with Conjugate». Αν το «Add with Conjugate» έχει οριστεί, πόλοι / μηδενικά εισάγονται ως ζεύγη. Ένα ισοδύναμος τρόπος να γίνει αυτό είναι μέσω της επιλογής «Add Pair» στο μενού «Edit».

Κουμπιά: -P, -Z, -A

Κάνοντας κλικ σε ένα από αυτά τα τρία κουμπιά θα απομακρύνει όλους τους πόλους/μηδενικά ή και τα δύο ανάλογα το πάτημα.

-P - Αφαιρεί όλους τους πόλους

-Z - Αφαιρεί όλα τα μηδενικά

-A - Αφαιρεί όλους του πόλους και τα μηδενικά

Ένας ισοδύναμος τρόπος να γίνει αυτό είναι με τη χρήση του μενού «Remove» στο μενού «Edit».

Κουμπί: Add Repeatedly

Εάν έχει οριστεί αυτή η επιλογή, οι Πόλοι/Μηδενικά μπορούν να προστίθενται συνεχώς πχ. συνεχή κλικ του ποντικιού στο διάγραμμα pole-zero θα προσθέσει κατάλληλα πόλους/μηδενικά. Αν δεν έχει οριστεί, ένα από τα 3 κουμπιά PP, ZZ, PZ πρέπει να επιλεγεί πριν από κάθε κλικ για την προσθήκη μηδενικών/πόλων.

Κουμπί: Add with Conjugate

Αυτή η επιλογή βοηθά την προσθήκη πόλων / μηδενικών ως ζεύγη. Εάν αυτό έχει οριστεί, τότε πόλοι / μηδενικά προστίθενται ως ζεύγη όταν τα πλήκτρα PP, ZZ, PZ πατηθούν ή πατηθεί στο μενού η επιλογή «Add Pair» .

Κουμπί: Zoom Enable

Αυτή η επιλογή βοηθά στην μεγέθυνση του γραφήματος ώστε να μπορεί να αναγνωρίσει η ακριβή θέση των πόλων και των μηδενικών.

Κουμπί: Click Enable

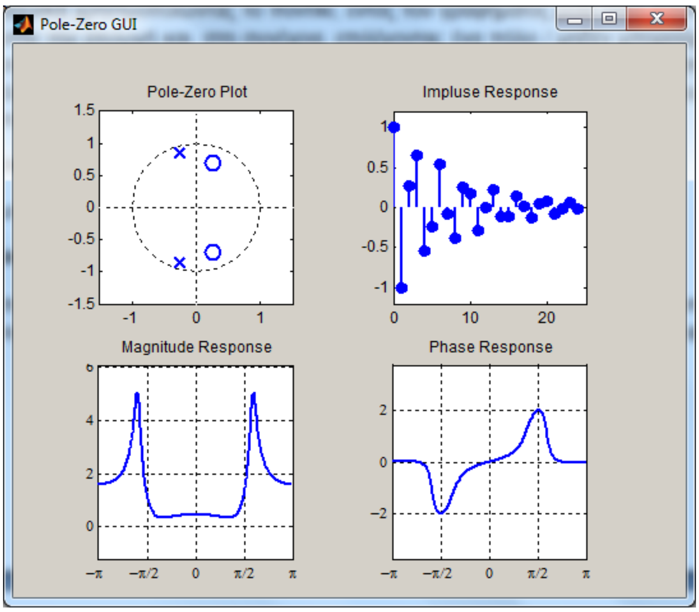
Αυτή η επιλογή βοηθά να επιλέξουμε ή να μετακινήσετε ήδη υπάρχοντες πόλους / μηδενικά χρησιμοποιώντας το ποντίκι, εντός του γραφήματος pole-zero. Επιλέγοντας αυτήν την επιλογή και, στη συνέχεια, επιλέγοντας ένα πόλο / μηδέν μπορούμε να το μετακινήσουμε σε διαφορετικές τοποθεσίες. Με την μετακίνηση των πόλων/μηδενικών, μπορούμε να δούμε τις αντίστοιχες διαφοροποιήσεις στην απόκριση του συστήματος.

Κουμπί: Unlink

Αυτή η επιλογή βοηθά να αποσυνδέσουμε ένα ζεύγος πόλων/μηδενικών.

Κουμπί: Show Graph

Επιλέγοντας αυτήν την επιλογή εμφανίζετε ένα νέο παράθυρο στο οποίο υπάρχουν συγκεντρωμένα το διάγραμμα pole-zero, η κρουστική απόκριση, μέτρο και η φάση.



Κουμπία: Show Ray, Show Grid

Το κουμπί Show Ray όταν είναι ενεργοποιημένο εμφανίζετε μια κόκκινη γραμμή στα διαγράμματα.

Το κουμπί Show Grid είναι για την εμφάνιση πλέγματος στα διαγράμματα.

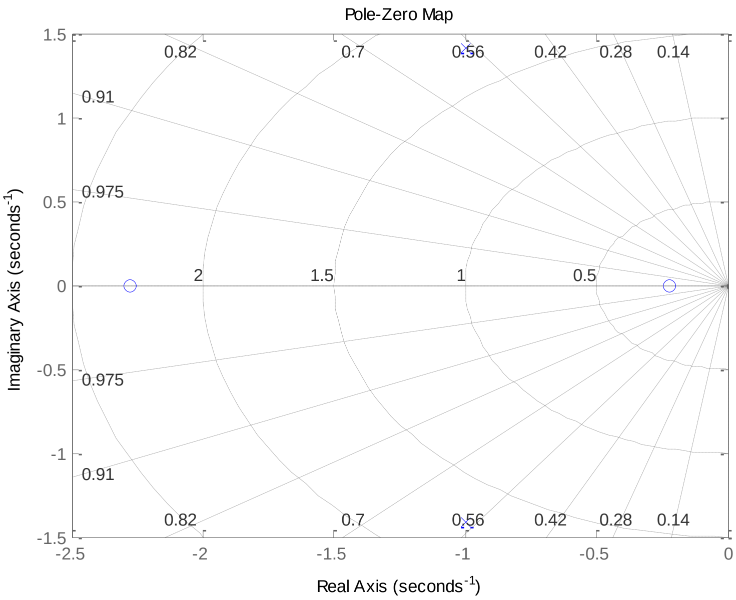
<u>Παράδειγμα MATLAB</u>

Πόλοι και τα μηδενικά του παρακάτω συστήματος συνεχούς χρόνου.

$$H(z) = \frac{2z^2 + 5z + 1}{z^2 + 2z + 3}$$

Κώδικας MATLAB:

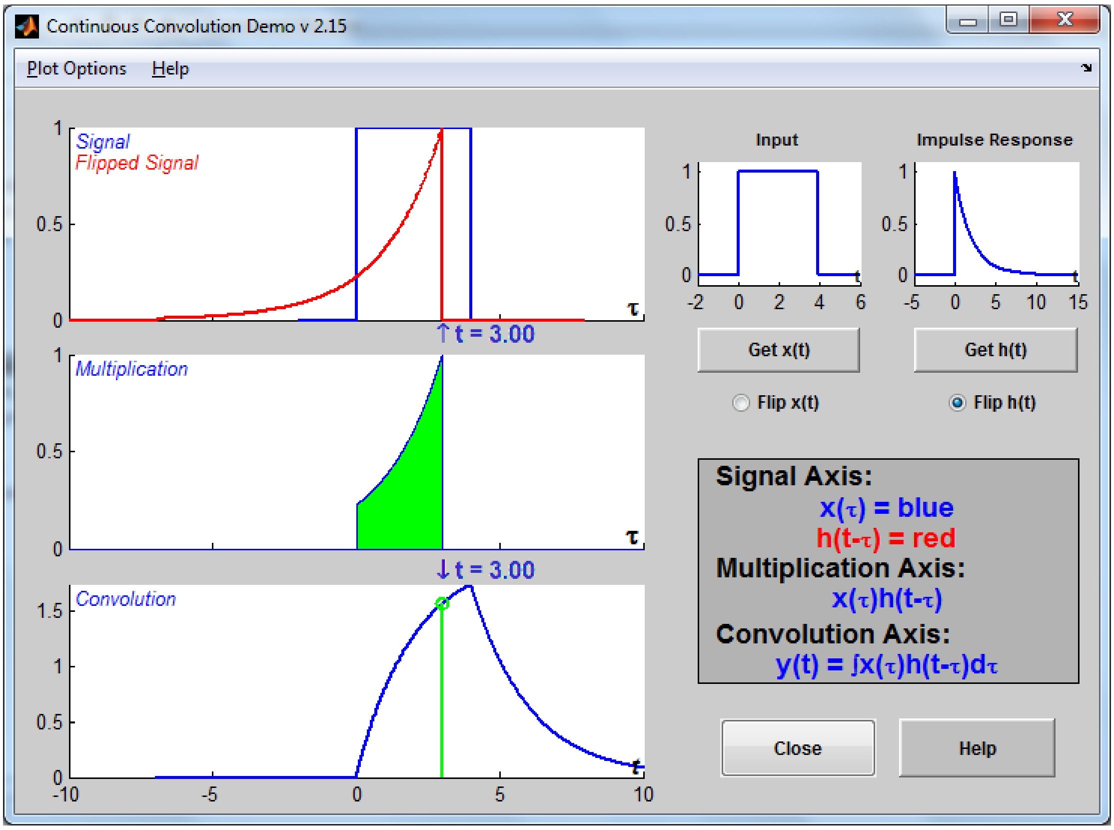
```
>> H = tf([2 5 1],[1 2 3]);
>> sgrid
>> pzmap(H)
>> grid on
```



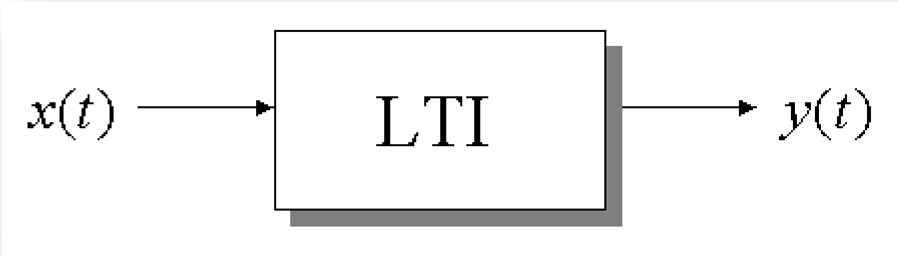
Π.4.5 Continuous Convolution Demo

Το Continuous Convolution Demo είναι ένα πρόγραμμα που μας βοηθά να καταλάβουμε τη διαδικασία της συνέλιξης συνεχούς χρόνου. χαρακτηριστικά:

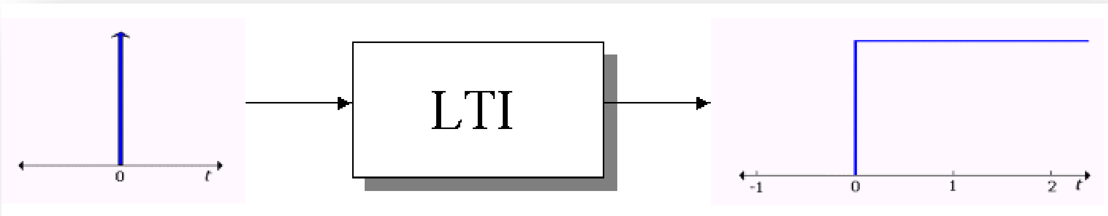
- Οι χρήστες μπορούν να επιλέξουν από μια ποικιλία από διαφορετικά σήματα.
- Τα σήματα μπορούν να συρθούν γύρω με το ποντίκι ή με τα βελάκια με τα αποτελέσματα να εμφανίζονται σε πραγματικό χρόνο.
- Επιλογή για απόκρυψη του αποτελέσματος της συνέλιξης ώστε να μπορεί να χρησιμοποιηθεί το πρόγραμμα σε μία τάξη ως προϊών εργασιών.



Συνέλιξη είναι μια πράξη με την οποία μπορεί να προσδιοριστεί η έξοδος ενός γραμμικού (LTI) συστήματος με μία γνωστή είσοδο. Παρατηρήστε το σύστημα που ακολουθεί, με συνεχή χρόνου είσοδο x (t) και y έξοδο (t). Συνέλιξη είναι απλά η διαδικασία που καθορίζει την έξοδο δεδομένου της εισόδου.



Ένα Impulse σήμα χρησιμοποιείται ως παράδειγμα εισόδου για το σύστημα που παρουσιάζεται παρακάτω. Όταν η είσοδος σε οποιοδήποτε σύστημα LTI είναι ένα Impulse σήμα, η έξοδος ονομάζεται η παλμική απόκριση και συμβολίζεται με h(t). Έτσι, σε αυτό το παράδειγμα h (t) = u (t) το σύστημα σε αυτό το συγκεκριμένο παράδειγμα είναι γνωστό ως ένα ολοκληρωτή επειδή παράγει ένα σήμα με μοναδιαίο βήμα ως έξοδο.



Η άλλη σημαντική ιδιότητα των συστημάτων συνεχούς LTI είναι η γραμμικότητα, η οποία επιτρέπει η έξοδος ενός συστήματος LTI να μπορεί να εκφραστεί ως το άθροισμα των εξόδων που λαμβάνονται από τα μεμονωμένα συστατικά που συνθέτουν το σήμα εισόδου. Η έξοδος του συστήματος μπορεί να βρεθεί μέσα από μια σχέση που είναι γνωστή ως Convolution Integral:

$$\begin{aligned} y(t) &= \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \\ &= x(t) * h(t) \end{aligned}$$

Η μεταβλητή t λαμβάνεται ως μια σταθερά για την ενσωμάτωση, η οποία γίνεται κατά τη διάρκεια της ψευδομεταβλητής “τ” για όλες τις μη μηδενικές τιμές της συνάρτησης εισόδου. Μετά την ολοκλήρωση η μεταβλητή “τ” εξαφανίζεται, αφήνοντας μία συνάρτηση χρόνου η οποία είναι η έξοδος του συστήματος.

Continuous Convolution Controls

Κουμπιά: Get x(t) | h(t)

Αυτά τα κουμπιά εμφανίζουν ένα νέο παράθυρο το οποίο μας επιτρέπει να επιλέξουμε x (t) και h (t). Ένα dropdown στο επάνω μέρος παρέχει μια περιορισμένη επιλογή των σημάτων. Αν κάποιος επιθυμεί, μπορεί να αλλάξει παραμέτρους του σήματος (π.χ., μήκος, πλάτος, κλπ ...). Κάνοντας κλικ στο κουμπί OK το επιλεγμένο σήμα εισάγετε στο κύριο παράθυρο.

Κουμπιά: Flip x(t) | h(t)

Αυτά τα κουμπιά μας επιτρέπουν να αλλάξουμε το ποίο σήμα γυρίζετε. Γυρίζοντας ένα σήμα εννοούμε ποιο σήμα αντικατοπτρίζετε στην άλλη πλευρά.

Ετικέτα: t = #

Υπάρχουν δύο ετικέτες για την τρέχουσα ώρα. Σύροντας αυτά με το ποντίκι θα αλλάξει την αξία του τρέχοντος χρόνου. Η τρέχουσα ώρα μπορεί επίσης να αλλάξει χρησιμοποιώντας τα αριθμητικά πλήκτρα 4 και 6 αλλά και τα βελάκια.

Διάγραμμα: Multiplication

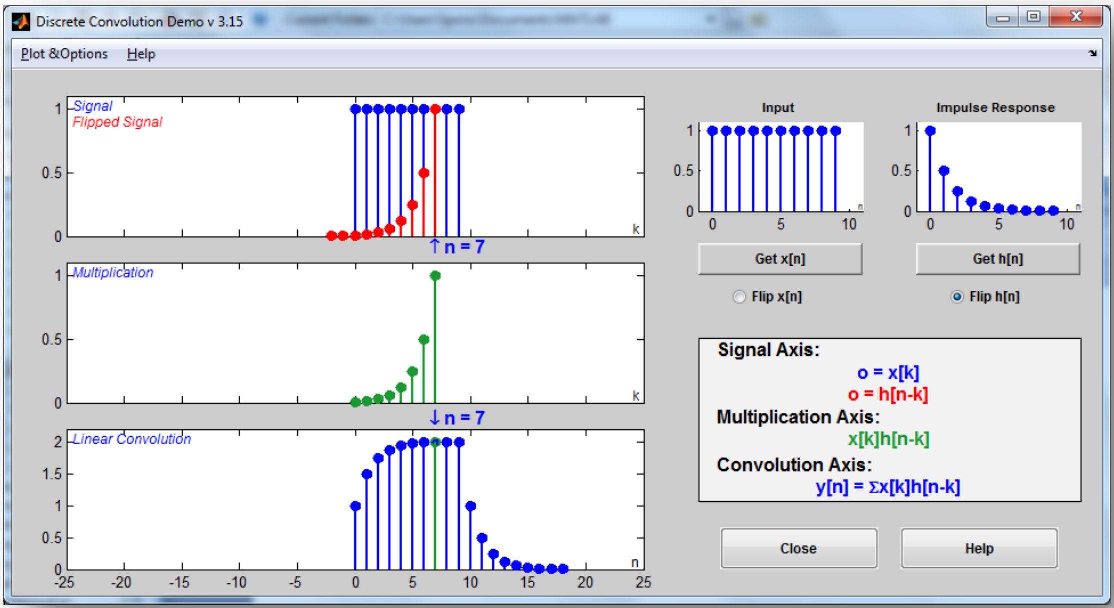
Αυτό το διάγραμμα δείχνει το αποτέλεσμα του πολλαπλασιασμού των δύο σημάτων που εμφανίζονται στο γράφημα «Signal-Flipped Signal». Το οικόπεδο αλλάζει ανάλογα με τη σχετική επικάλυψη μεταξύ των δύο σημάτων.

Διάγραμμα: Convolution

Αυτό το διάγραμμα δείχνει την έξοδο της γραμμική συνέλιξη μεταξύ του σήματος x(t) και του h(t).

Σημειώστε ότι η τιμή στην τρέχουσα χρονική στιγμή επισημαίνεται στο παραπάνω διάγραμμα. Η αριθμητική τιμή (δηλαδή, η τιμή στον άξονα y) του δείγματος προσδιορίζεται με την ενσωμάτωση της περιοχής εντός του διαγράμματος του πολλαπλασιασμού.

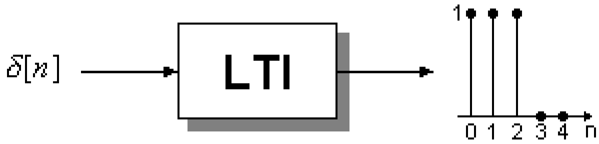
Π.4.6 Discrete Convolution Demo



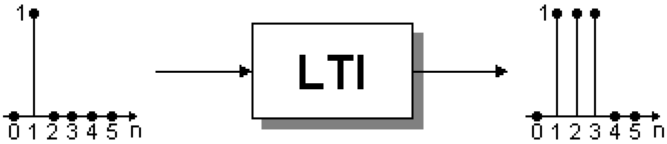
Το Discrete Convolution Demo είναι ένα πρόγραμμα που μας βοηθά να καταλάβουμε τη διαδικασία της συνέλιξης διακριτού χρόνου. χαρακτηριστικά:

- Οι χρήστες μπορούν να επιλέξουν από μια ποικιλία από διαφορετικά σήματα.
- Τα σήματα μπορούν να συρθούν γύρω με το ποντίκι ή με τα βελάκια με τα αποτελέσματα να εμφανίζονται σε πραγματικό χρόνο.
- Επιλογή για απόκρυψη του αποτελέσματος της συνέλιξης ώστε να μπορεί να χρησιμοποιηθεί το πρόγραμμα σε μία τάξη ως προϊών εργασιών.
-

Όταν η είσοδος σε ένα σύστημα είναι ένα σήμα impulse τότε καλούμε την έξοδο Impulse response (Κρουστική απόκριση). Είναι κοινή πρακτική να χρησιμοποιούμε το συμβολισμό h[n] για να υποδηλώσουμε την απόκριση. Ως εκ τούτου, y[n] = h[n] όταν η είσοδος είναι ένα σήμα Impulse. Το συγκεκριμένο σύστημα στο παρακάτω παράδειγμα έχει σαν έξοδο ένα σήμα μήκους 3.

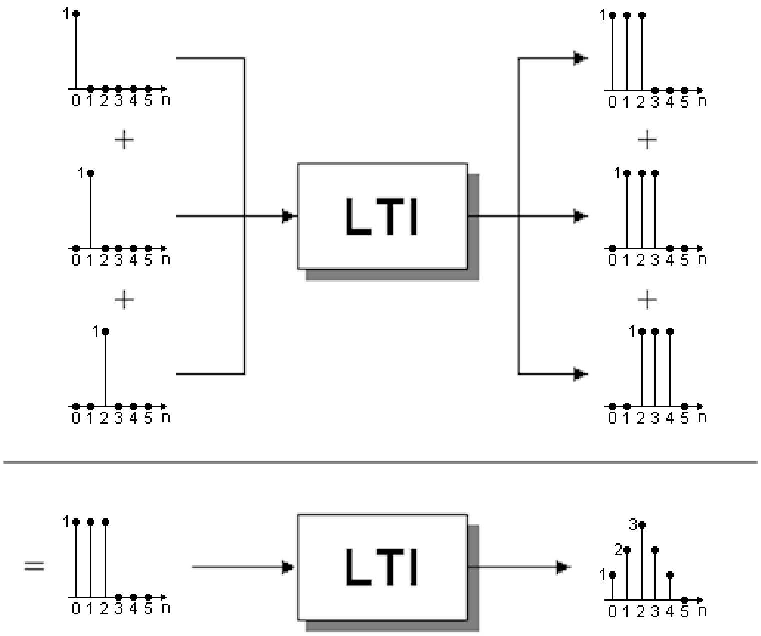


Όταν η είσοδος μετατοπίζεται κατά ένα παλμό τότε η έξοδος φαίνεται να είναι απλά μια μετατοπισμένη έκδοση. Ο λόγος είναι ότι υποθέτουμε το σύστημα σαν χρονικά αμετάβλητο. Όταν ένα σύστημα είναι χρονικά αμετάβλητο, μπορεί κάνεις να καθορίσει την απόκριση σε μια μετατοπισμένη είσοδο από την έξοδο της μη μετατοπισμένης έκδοσης. Σε αυτή την περίπτωση, η είσοδος είναι μια μετατοπισμένη έκδοση έτσι ώστε η νέα έξοδος να είναι μία ομοίως μετατοπισμένη εκδοχή.

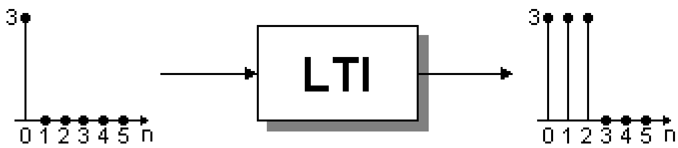


Στο παρακάτω παράδειγμα φαίνονται 3 μετατοπισμένοι παλμοί και το άθροισμα τους.

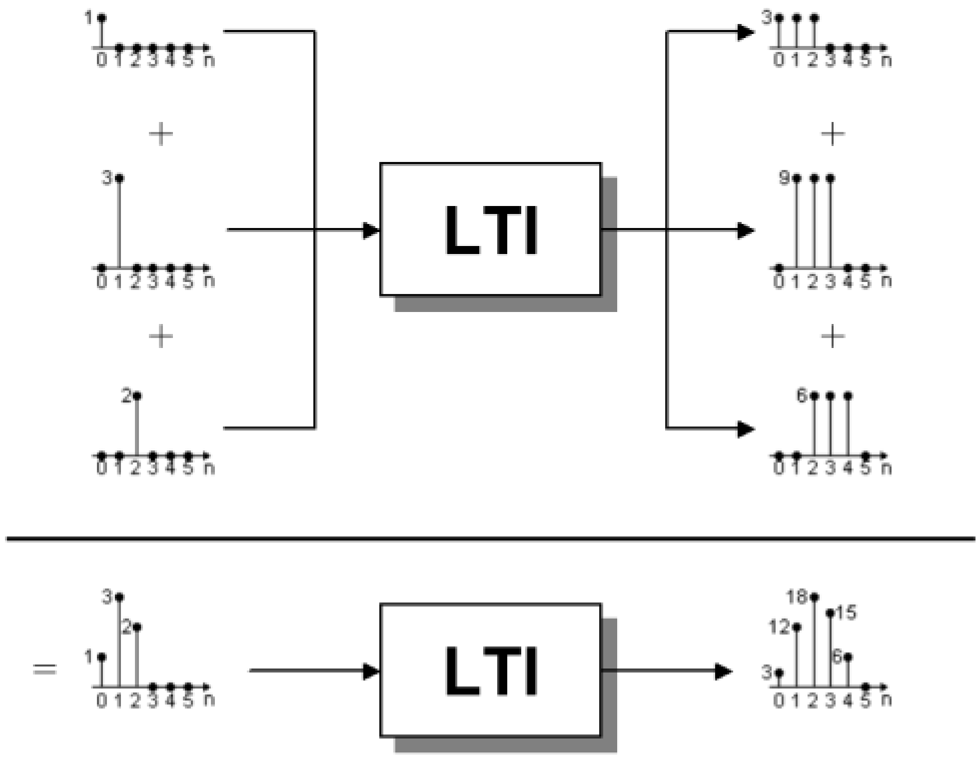
Η έξοδος με την τριγωνική μορφή είναι απλά το άθροισμα των τριών μετατοπισμένων αποκρίσεων. Όταν το σύστημα είναι γραμμικό η έξοδος ενός αθροίσματος εισόδων είναι το άθροισμα από τις επιμέρους εξόδους. Αυτή είναι η μία ιδιότητα από της δύο ιδιότητες που περιγράφουν την γραμμικότητα.



Η δεύτερη ιδιότητα της γραμμικότητας προσομοιώνετε από διάγραμμα παρακάτω. Η είσοδος είναι μία κλιμακωμένη έκδοση. Στην συγκεκριμένη περίπτωση είναι 3 φορές μεγαλύτερη από την αρχική έκδοση. Έτσι η έξοδος είναι απλά μία απόκριση κλιμακωμένη κατά τον ίδιο παράγοντα (3 στο συγκεκριμένο παράδειγμα). Έτσι μπορούμε να πούμε σας γενικό κανόνα ότι μία κλιμάκωση πηγαίνει από την είσοδο στην έξοδο.



Παρακάτω φαίνεται ένα παράδειγμα για το πώς υπολογίζουμε την έξοδο χρησιμοποιώντας τις παραπάνω ιδιότητες.



Discrete Convolution Controls

Κουμπιά: Get x(n) | h(n)

Αυτά τα κουμπιά εμφανίζουν ένα νέο παράθυρο το οποίο μας επιτρέπει να επιλέξουμε x(n) και h(n). Ένα dropdown στο επάνω μέρος παρέχει μια περιορισμένη επιλογή των σημάτων. Αν κάποιος επιθυμεί, μπορεί να αλλάξει παραμέτρους του σήματος (π.χ., μήκος, πλάτος, κλπ ...). Κάνοντας κλικ στο κουμπί OK το επιλεγμένο σήμα εισάγετε στο κύριο παράθυρο.

Κουμπιά: Flip x(n) | h(n)

Αυτά τα κουμπιά μας επιτρέπουν να αλλάξουμε το ποίο σήμα γυρίζετε. Γυρίζοντας ένα σήμα εννοούμε ποιο σήμα αντικατοπτρίζετε στην άλλη πλευρά.

Ετικέτα: t = #

Υπάρχουν δύο ετικέτες για την τρέχουσα ώρα. Σύροντας αυτά με το ποντίκι θα αλλάξει την αξία του τρέχοντος χρόνου. Η τρέχουσα ώρα μπορεί επίσης να αλλάξει χρησιμοποιώντας τα αριθμητικά πλήκτρα 4 και 6 αλλά και τα βελάκια.

Διάγραμμα: Multiplication

Αυτό το διάγραμμα δείχνει το αποτέλεσμα του πολλαπλασιασμού των δύο σημάτων που εμφανίζονται στο γράφημα «Signal-Flipped Signal». Το οικόπεδο αλλάζει ανάλογα με τη σχετική επικάλυψη μεταξύ των δύο σημάτων.

Διάγραμμα: Convolution

Αυτό το διάγραμμα δείχνει την έξοδο της γραμμική συνέλιξη μεταξύ του σήματος x(n) και του h(n).

Σημειώστε ότι η τιμή στην τρέχουσα χρονική στιγμή επισημαίνεται στο παραπάνω διάγραμμα. Η αριθμητική τιμή (δηλαδή, η τιμή στον άξονα y) του δείγματος προσδιορίζεται με την ενσωμάτωση της περιοχής εντός του διαγράμματος του πολλαπλασιασμού.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) "Signals and Systems Laboratory with Matlab", Palamides - Veloni, Taylor and Francis Gtoup - CRC, USA, 2011.

2) "Σήματα και Συστήματα με Matlab’΄, Παλαμίδης – Βελώνη – Σύγχρονη Εκδοτική - 2008.

3) "Σήματα και Συστήματα για Τεχνολόγους΄, Φωτόπουλος – Βελώνη – Σύγχρονη Εκδοτική - 2008.

4) "Signals and Systems", Poularikas - Seely, PWS - KENT Publishing Co., 1991.

5) "Digital Signal Processing", 3rd edition, Proakis - Manolakis, Prentice - Hall Intl., 1996.

6) "Modern Signals and Systems", Kwakernaak - Sivan, Prentice - Hall Intl., 1991.

7) "Continuous and Discrete Signals and Systems", Soliman - Srinath, Prentice - Hall Intl.,
1998.

8) "Signals and Systems : Continuous and Discrete", Ziemer - Tranter - Fannin, Macmillan
Publishing Co., 1990.

9) "Principles of Signals and Systems", Taylor, McGraw-Hill Intl., 1994.

10) "Introduction to Signal Processing", Orfanidis, Prentice - Hall Intl., 1996.

11) "Signals, Systems and Transforms", C. L. Phillips & J. M. Parr, Prentice - Hall Intl., 1995.

12) "Fundamentals of Signals and Systems using MATLAB", E. W. Kamen - B. S. Heek,
Prentice - Hall Intl., 1997.

13) "Digital Coding of Waveforms : Principles and Applications to Speech and Video", N.
S. Jayant, P. Noll, Prentice - Hall Intl., 1984.

14) "Signals and Systems : An Introduction", L. Balmer, Prentice - Hall Intl., 1997.

15) "Discrete - Time and Continuous - Time Linear Systems", R. J. Mayhan, Addison -

Wesley, 1984.

16) "Systemes et asservissements lineaires echantillonnes", Dunod Universite, 1969.

17) "Statistical and Adaptive Signal Processing", D. G. Manolakis, V. K. Ingle, S. K. Kogon, McGraw - Hill, 2000.

18) "Ψηφιακή Επεξεργασία Σημάτων : Θεωρία, Αλγόριθμοι, Πρακτική", Γ. Καραγιάννης,
Μ. Ραγκούση, Εκδόσεις ΣΥΜΕΩΝ.

19) "Αναγνώριση Προτύπων", Γ. Καραγιάννης,Σταϊνχάουερ, Εκδόσεις ΣΥΜΕΩΝ.

20) "Discrete - Time Signal Processing", Oppenheim - Schafer, Prentice - Hall Intl., 1989.

21) "Signals and Systems", Oppenheim - Willsky - Nawab, Prentice - Hall Intl., 1997.