ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Σχολή Μηχανικών Παραγωγής και Διοίκησης

Ανάπτυξη Υπολογιστικών Εφαρμογών ως Εργαλείο Εκμάθησης της Κλασικής Θεωρίας Ελέγχου

Διπλωματική Εργασία: Μπερσής Κωνσταντίνος

Επιβλέπων Καθηγητής: Ιψάκης Δημήτριος

Χανιά, 2023

Περιεχόμενο

ΠΕΡΙΛΗΨΗ:

Η παρούσα διπλωματική ασχολείται με την ανάπτυξη εκπαιδευτικού υλικού που θα καθοδηγεί τους προπτυχιακούς φοιτητές στην κατασκευή κώδικα σε Matlab, στο μάθημα των Συστημάτων Ελέγχου.

Πιο συγκεκριμένα, στο πρώτο κεφάλαιο θα δούμε μερικές βασικές εντολές και διαδικασίες χρήσης του Matlab. Στη συνέχεια, θα παρουσιάσουμε τρία παραδείγματα επίλυσης διαφορικών εξισώσεων. Έπειτα, στο δεύτερο κεφάλαιο, θα δούμε παραδείγματα με μετασχηματισμούς Laplace. Στο τρίτο κεφάλαιο, θα ασχοληθούμε με παραδείγματα απόκρισης στάθμης και θερμοκρασίας καθώς και παραδείγματα αναπαράστασης συστημάτων σε διαγράμματα βαθμίδων με σκοπό την εύρεση της απόκρισης των σημάτων εισόδου, εξόδου και σφάλματος σε μορφή διαγραμμάτων. Συνεχίζοντας, στο τέταρτο κεφάλαιο, θα απασχοληθούμε με την επίλυση συστημάτων 1ης και 2ης τάξης παρουσία ελεγκτή αλλά και απουσία αυτού, ενώ στο πέμπτο και τελευταίο κεφάλαιο θα παρουσιαστούν θέματα ευστάθειας, σε συστήματα ανοικτού και κλειστού βρόχου, ενώ παράλληλα θα υπάρχουν παραδείγματα που θα απαιτούν τον σχεδιασμό του γεωμετρικό τόπο των ριζών και τον συντονισμό των παραμέτρων ενός ελεγκτή PID.

Όλα τα παραδείγματα, παρουσιάζονται με αναλυτική εξήγηση της κάθε γραμμής του κώδικα ενώ υπάρχουν και επιπλέων άλυτα παραδείγματα, που έχουν ως σκοπό, την περαιτέρω εξάσκηση του φοιτητή.

ABSTRACT:

This thesis deals with the development of educational material that will guide undergraduate students in the construction of code in Matlab, in the course of Control Systems.

More specifically, in the first chapter we will look at some basic commands and procedures for using Matlab. Then, we will present examples of solving differential equations. After that, in the second chapter, we will look at examples of Laplace transforms. In the third chapter, we will deal with examples of response, as well as, examples of representing systems in step diagrams in order to find the response of the input, output and error signals in the form of graphs. Continuing, in chapter four, we will deal with the solving of 1st and 2nd order systems in the presence and absence of a controller. In the fifth and final chapter, stability problems will be presented, while there will be examples requiring the design of the geometric root locus and the coordination of the parameters of a PID controller.

All examples, are presented with detailed explanation of each line of code and there are additional unsolved problems, which are intended to further enhance the student's understanding.

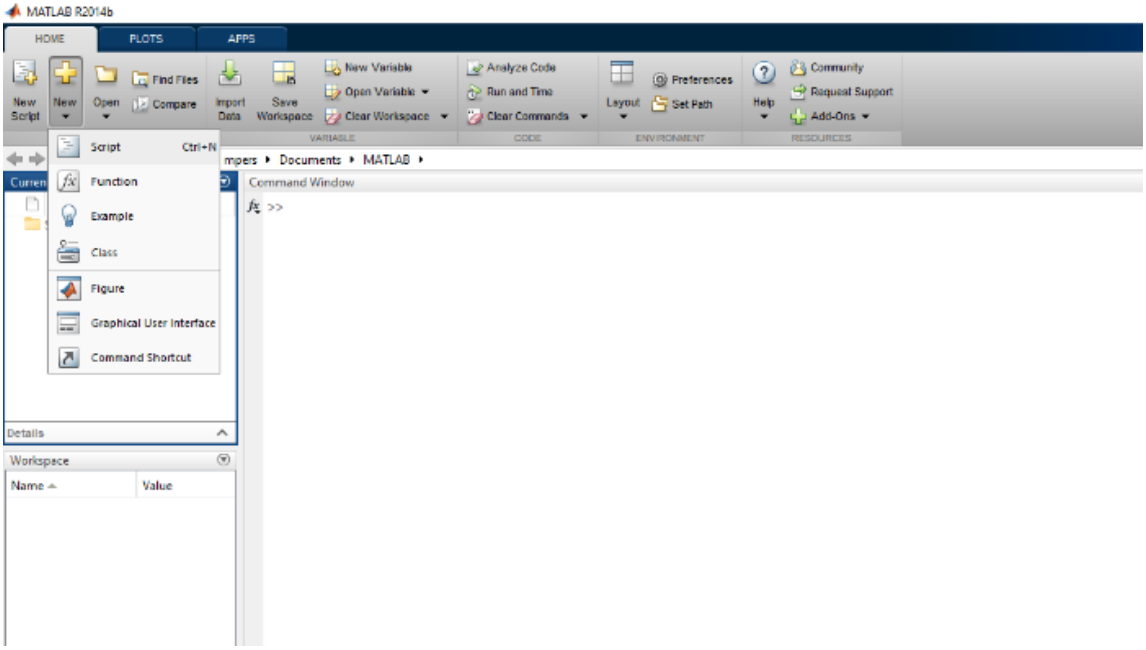
ΚΕΦΑΛΑΙΟ 1: Εισαγωγικά Στοιχεία

Στο πρώτο κεφάλαιο θα δοθεί, μια σύντομη εισαγωγή στο λογισμικό του Matlab, με επανάληψη των βασικών λειτουργιών. Έπειτα, θα ακολουθήσουν παραδείγματα προσομοίωσης μαθηματικών (δυναμικών) μοντέλων στο πεδίο του χρόνου με διαφορικές εξισώσεις 1^{ης} και 2^{ης} τάξης. Ειδικότερα, θα παρουσιαστούν τα παραδείγματα:

- *Θέρμανσης Ρευστού σε Δοχείο Πλήρους Ανάμιξης*
- *Κίνησης υπό την επίδραση συνισταμένης δυνάμεων*
- *Μελέτης Ηλεκτρικών Κυκλωμάτων RLC*

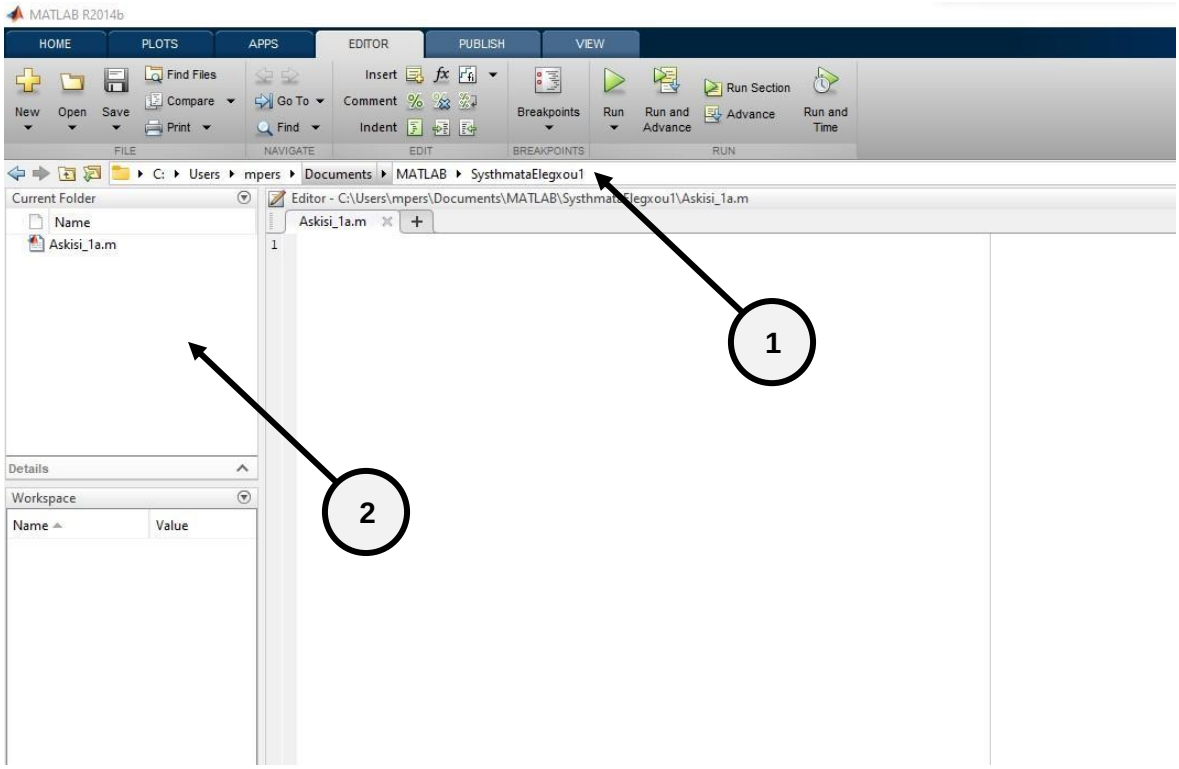
1.1 Εισαγωγή στο Matlab

Ως πρώτο στάδιο είναι η δημιουργία ενός αρχείου Matlab (m.file). Στο περιβάλλον του Matlab, πατάμε στη περιοχή New όπου βρίσκεται στο πάνω αριστερά κομμάτι της οθόνης μας και επιλέγουμε Script ή αλλιώς μπορούμε να πατήσουμε κατευθείαν τον συνδυασμό πλήκτρων Ctrl + N.



Στη συνέχεια θα ανοίξει το αρχείο m.file όπου και θα πρέπει να αποθηκευτεί πατώντας Save στην πάνω αριστερή μεριά του προγράμματος ή πατώντας τον συνδυασμό πλήκτρων Ctrl + S. Το όνομα που θα δοθεί στο αρχείο θα πρέπει να είναι με **λατινικούς χαρακτήρες** (π.χ. Askisi_1a). Παράλληλα, ο προορισμός αποθήκευσης του αρχείο θα πρέπει και αυτός να περιέχει **μόνο** λατινικούς χαρακτήρες σε ολόκληρη τη διαδρομή του. Επίσης, η διαδρομή του προορισμού, θα πρέπει να είναι η ίδια με αυτή της μπάρα που φαίνεται στη παρακάτω φωτογραφία (βέλος 1). Ειδάλλως, δεν θα μπορέσει να εκτελεστεί το αρχείο. Ο τρόπος επιλογής αυτής της διαδρομής, γίνεται με χρήση της περιοχής *Current Folder* στα αριστερά της οθόνης

μας (βέλος 2), κάνοντας διπλό κλικ στους φακέλους μέχρι να φτάσουμε τον φάκελο που μόλις αποθηκεύτηκε το αρχείο.

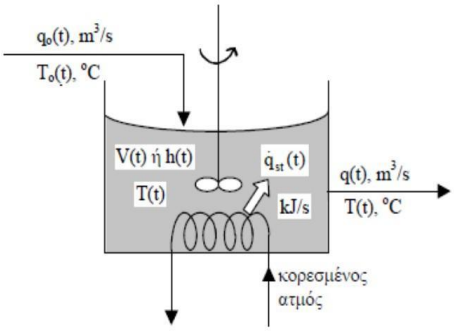


1.2 Εισαγωγικά Παραδείγματα

Στην συνέχεια, θα παρουσιαστούν παραδείγματα επίλυσης διαφορικών εξισώσεων.

Παράδειγμα 1.2.1: Θέρμανση Ρευστού σε Δοχείο Πλήρους Ανάμιξης

Εκφώνηση: Στο δοχείο του διπλανού σχήματος εισέρχεται ρευστό με παροχή q_0 (m^3/s) και θερμοκρασία T_0 (°C). Από το δοχείο εξέρχεται το ίδιο ρευστό με παροχή q (m^3/s) και θερμοκρασία T (°C). Το ρευστό εντός του δοχείου θερμαίνεται με παροχή κορεσμένου ατμού q_{st} (kJ/s). Να αναπτύξετε το μαθηματικό μοντέλο που θα περιγράφει την μεταβολή της στάθμης h (m) και την μεταβολή της θερμοκρασίας T (℃) με τον χρόνο. Θεωρήστε σταθερές τις ιδιότητες ρευστού (πυκνότητα, θερμοχωρητικότητα κτλ) και σταθερή διατομή στο δοχείο.



Οι σταθερές τιμές είναι :

$$p = 1000\frac{kg}{m^3} \qquad q = 0.8 \cdot 10^{-3}\frac{m^3}{s} \qquad q_0 = 1 \cdot 10^{-3}\frac{m^3}{s}$$

$$C_p = 4184\frac{J}{kg}K \qquad T_0 = 298K \qquad q_{st} = 50000\frac{J}{s}$$

$$A = 1m^2 \qquad T_r = 0K \qquad T(t=0) = 288K, h(t=0) = 1m$$

Επίλυση: Ξεκινάμε αναπτύσσοντας τις διαφορικές εξισώσεις που περιγράφουν το παραπάνω πρόβλημα :

$$A\frac{dh}{dt} = q_0 - q \Rightarrow$$

$$p \cdot C_p \frac{d(T \cdot V)}{dt} = p \cdot C_p \frac{d(T \cdot A \cdot h)}{dt} = p \cdot C_p \cdot q_0 \cdot (T_0 - T_r) - p \cdot C_p \cdot q \cdot (T - T_r) + q_{st} \Rightarrow$$

$$p \cdot C_p \frac{d(T \cdot V)}{dt} = p \cdot C_p \cdot [T \cdot A\frac{d(h)}{dt} + A \cdot h\frac{d(T)}{dt}] \Rightarrow$$

$$p \cdot C_p \frac{d(T \cdot V)}{dt} = p \cdot C_p \cdot q_0 \cdot (T_0 - T_r) - p \cdot C_p \cdot q \cdot (T - T_r) + q_{st} \Rightarrow$$

$$\boldsymbol{A \cdot h\frac{d(T)}{dt} = q_0 \cdot (T_0 \cdot T_r - T) - q \cdot (T_r) + \frac{q_{st}}{p \cdot C_p}}$$

Τις παραπάνω εξισώσεις θα πρέπει να τις μεταφέρουμε στο περιβάλλον του Matlab. Ξεκινάμε ανοίγοντας ένα καινούργιο m.file σύμφωνα με τις οδηγίες που αναφέρθηκαν στη προηγούμενη ενότητα.

Όπως φαίνεται παρακάτω, στην πρώτη γραμμή δηλώνεται ότι θα επιλυθεί μια συνάρτηση (function) που θα περιλαμβάνει διαφορικές εξισώσεις [dx]. Στη συνέχεια, πρέπει να δοθεί **το ίδιο ακριβώς όνομα αρχείου**. Το (Time, x) υποδηλώνει ότι θα επιλύσουμε τις μεταβλητές X (π.χ. στάθμη και θερμοκρασία) στον χρόνο (Time). Βάζοντας το σύμβολο % μπροστά, μπορούμε να γράψουμε ορισμένα σχόλια.

```
function [dx] = Simulation_Tank_Lesson_2(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature
DynamicModeling/Differential Equations)
```

Στη συνέχεια, θα δηλώσουμε τις σταθερές μεταβλητές του προβλήματός μας, όπως φαίνεται στον παρακάτω πίνακα (βλ. εκφώνηση).

```
%Parameters
r=1000;          %kg/m3
cp=4184;         %J/K*kg
```

```
A=1;              %m2
q_0=1e-3;         %m3/s
q=0.8e-3;         %m3/s
T0=298;           %K
q_steam=50000;    %J/s
Tref=0;           %K
```

Στη περίπτωση του προβλήματός μας, εκτός από σταθερές μεταβλητές έχουμε και μεταβλητές κατάστασης, δηλαδή μεταβλητές που είναι εξαρτημένες από τον χρόνο. Στο πρόβλημά μας, μεταβλητές κατάστασης αποτελούν η στάθμη (h) και η θερμοκρασία (T).

```
%State Variables
h = x(1,1);
T = x(2,1);
```

Στη συνέχεια, θα ορίσουμε τις διαφορικές εξισώσεις ακριβώς όπως τις αναπτύξαμε προηγουμένως .

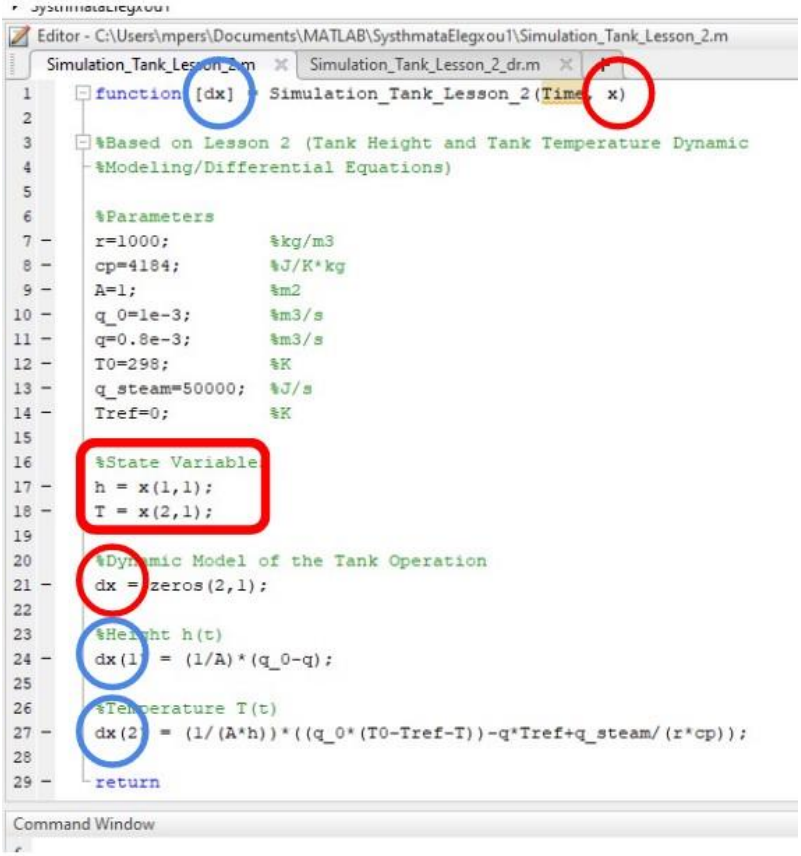
$$\frac{dh}{dt} = \frac{1}{A}(q_0 - q)$$

$$A \cdot h\frac{d(T)}{dt} = q_0 \cdot (T_0 \cdot T_r - T) - q \cdot (T_r) + \frac{q_{st}}{p \cdot C_p}$$

Δύο απαραίτητες σειρές που θα πρέπει να προσθέσουμε πριν και μετά από τις εξισώσεις μας, για την υπολογιστική μας επίλυση είναι :

1. Να δηλώσουμε το μέγεθος του προβλήματος, όπου στη περίπτωση μας έχουμε 2 διαφορικές εξισώσεις, άρα *zeros(2,1).*
2. Να τελειώσουμε γράφοντας την εντολή *return* για να σταματήσει εκεί η εκτέλεση του κώδικα

```
%Dynamic Model of the Tank Operation
dx = zeros(2,1);

%Height h(t)
dx(1) = (1/A)*(q_0-q);
%Temperature T(t)
dx(2) = (1/(A*h))*((q_0*(T0-Tref-T))-q*Tref+q_steam/(r*cp));

return
```



Είναι απαραίτητη η σύνδεση των μεταβλητών κατάστασης (State Variables) με τον ορισμό των συναρτήσεων μας στην αρχή του κώδικα.

Έπειτα, θα ανοίξουμε ένα νέο m.file, έτσι ώστε να γράψουμε τις διαφορικές εξισώσεις. Θα μπορούσαμε να τις γράψουμε και στο ίδιο αρχείο μαζί με τις μεταβλητές μας, *αλλά είναι πιο εύχρηστο να βρίσκονται ξεχωριστά.*

Ξεκινάμε γράφοντας τις εντολές *clear* και *clc* για να καθαρίσουμε το ιστορικό εκτέλεσης στο Matlab από παλαιότερα αποτελέσματα καθώς δεν γίνεται αυτόματα από το πρόγραμμα. Στη συνέχεια, δηλώνουμε τον χρόνο προσομοίωσης που θέλουμε να έχουμε, από t=0 s έως t=10000 s με βήμα 1, με χρήση της εντολής *tspan = [0:1:10000]* και τις αρχικές μας συνθήκες στάθμης και θερμοκρασίας με τις εντολές *x0(1)=1* και *x0(2)=288* αντίστοιχα, δηλαδή στάθμη h(t=0)= 1 m και θερμοκρασία T(t=0)= 288 K (βλ. εκφώνηση).

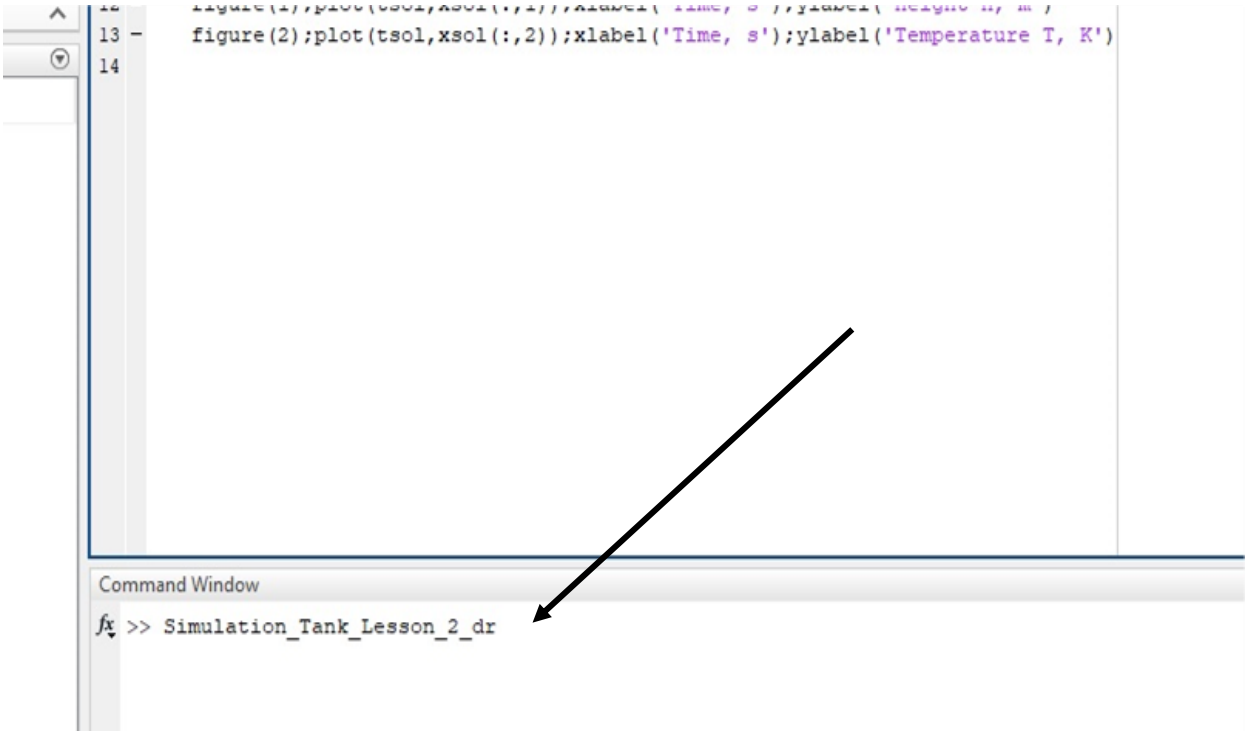
```
clear
clc

tspan = [0:1:10000];

%Initial Values for State Variables
x0(1)=1;          %m
x0(2)=288;        %K
```

Ακολουθεί η επίλυση των διαφορικών εξισώσεων με χρήση της μεθόδου Runge - Kutte(ode45), όπου είναι η στάνταρ μέθοδος επίλυσης διαφορικών εξισώσεων στο Matlab. Η τελική εντολή έχει τη μορφή *[tsol,xsol]=ode45(@Simulation_Tank_Lesson_2,tspan,x0)* όπου tsol,xsol οι

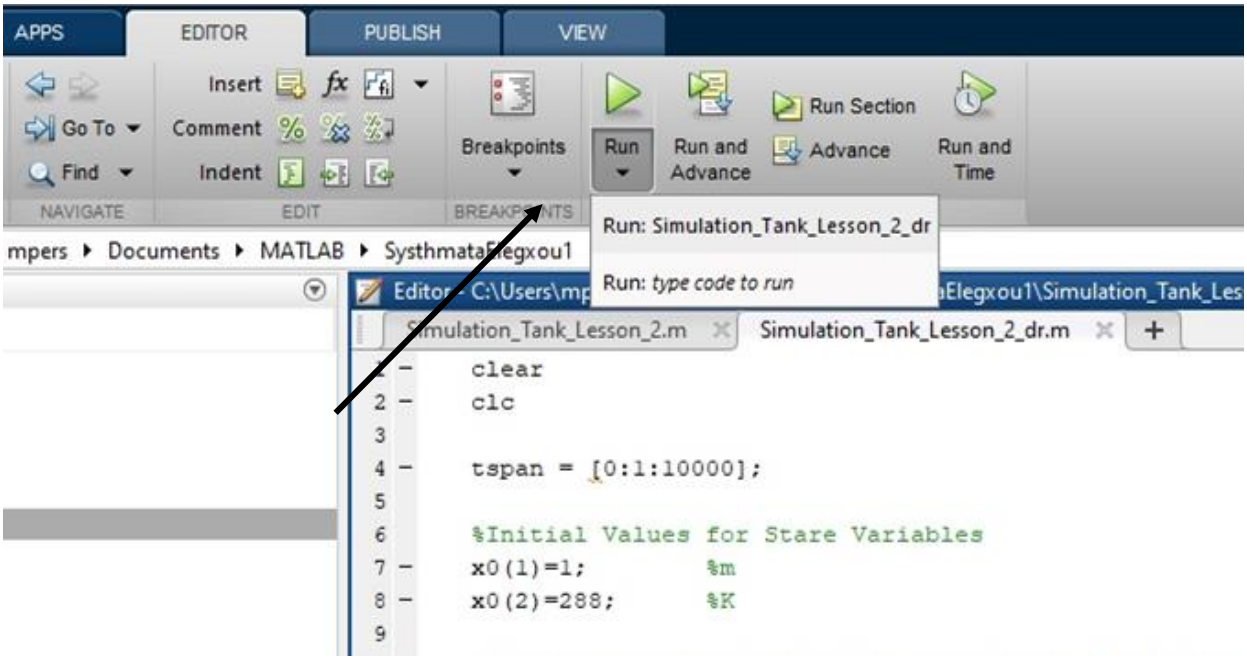
τελικές λύσεις για την θερμοκρασία και το ύψος της στάθμης αντίστοιχα. Το όνομα του αρχείου που έχουμε τις μεταβλητές μας, πρέπει οπωσδήποτε να είναι γραμμένο το ίδιο, καθώς και ο χρόνος προσομοίωσης (tspan) και οι αρχικές συνθήκες (t=0). Τέλος, ζητάμε με την εντολή *plot* να μας δώσει τα αποτελέσματα υπο μορφή διαγραμμάτων για την στάθμη και για την θερμοκρασία, ονομάζοντάς τα figure(1) και figure(2) αντίστοιχα.

```
[tsol,xsol]=ode45(@Simulation_Tank_Lesson_2,tspan,x0);

figure(1);plot(tsol,xsol(:,1));xlabel('Time, s');ylabel('Height h, m')
figure(2);plot(tsol,xsol(:,2));xlabel('Time, s');ylabel('Temperature T, K')
```

Το σύνολο του κώδικά μας έχει ολοκληρωθεί. Αρκεί να τρέξουμε το δεύτερο κώδικα για πάρουμε τα αποτελέσματά μας. Μπορούμε είτε να γράψουμε το όνομα του δεύτερου κώδικα στην περιοχή Command Window (Επιλογή 1) στο κάτω μέρος της οθόνης μας, είτε έχοντας επιλεγμένο το δεύτερο κώδικα να πατήσουμε το σύμβολο Run (Επιλογή 2) στο πάνω μέρος της οθόνης.



Επιλογή 1



Επιλογή 2

Τα αποτελέσματα εμφανίζονται στην περιοχή Workspace του περιβάλλοντος ενώ τα διαγράμματα σε ξεχωριστό παράθυρο.

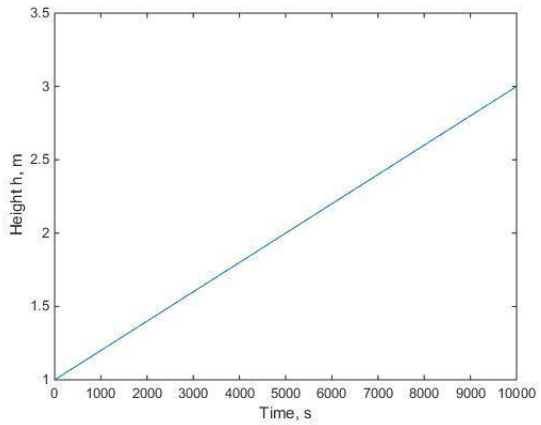


Figure 1

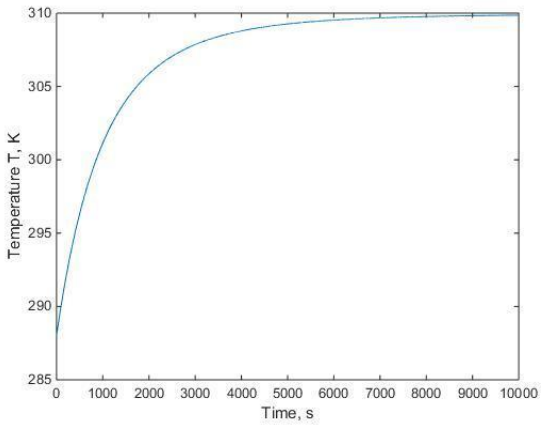
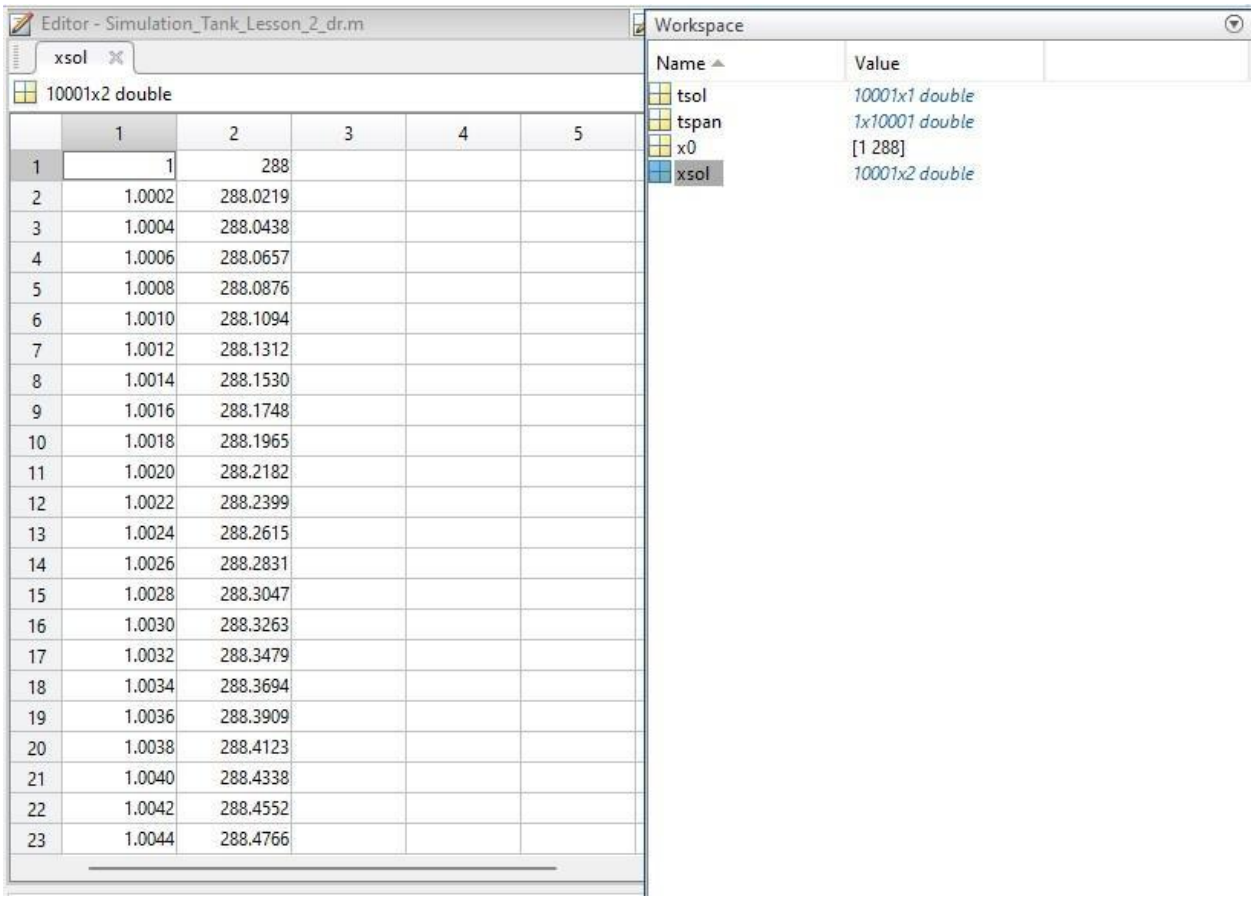


Figure 2

Με διπλό κλικ στο στοιχείο xsol της περιοχής Workspace, εμφανίζονται τα αποτελέσματα Στάθμης (στήλη 1) και Θερμοκρασίας (στήλη 2) όπως φαίνονται στην παρακάτω εικόνα.



	1	2	3	4	5
1	1	288			
2	1.0002	288.0219			
3	1.0004	288.0438			
4	1.0006	288.0657			
5	1.0008	288.0876			
6	1.0010	288.1094			
7	1.0012	288.1312			
8	1.0014	288.1530			
9	1.0016	288.1748			
10	1.0018	288.1965			
11	1.0020	288.2182			
12	1.0022	288.2399			
13	1.0024	288.2615			
14	1.0026	288.2831			
15	1.0028	288.3047			
16	1.0030	288.3263			
17	1.0032	288.3479			
18	1.0034	288.3694			
19	1.0036	288.3909			
20	1.0038	288.4123			
21	1.0040	288.4338			
22	1.0042	288.4552			
23	1.0044	288.4766			

Μπορούμε να έχουμε τα αποτελέσματα ξεχωριστά, γράφοντας στο Command Window την εντολή xsol(:,1) για να μας εμφανίσει τα αποτελέσματα της πρώτης στήλης του πίνακα, δηλαδή της στάθμης. Την εντολή xsol(:,2) για να εμφανίσει τα αποτελέσματα της δεύτερης στήλης του πίνακα, δηλαδή της θερμοκρασίας και tsol για τον χρόνο προσομοίωσης.

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 1.2.1)

Εξάσκηση

Εκφώνηση: Με βάση τα προηγούμενα, να επιλύσετε υπολογιστικά τις αποκρίσεις (δηλ. την εξάρτηση από τον χρόνο) της στάθμης και της θερμοκρασίας εάν:

1) $A = 0.1m^2$ *και όλα τα υπόλοιπα όπως πριν.*
2) $q_0 = 0.8 \cdot 10^{-3}\, m^3/s$ *και όλα τα υπόλοιπα όπως πριν.*
3) $q = 1.2 \cdot 10^{-3}\, m^3/s$ *και όλα τα υπόλοιπα όπως πριν.*
4) $q_{st} = 500\, J/s$ *και όλα τα υπόλοιπα όπως πριν.*
5) $q = \frac{h}{100} m^3/s$ *και όλα τα υπόλοιπα όπως πριν.*

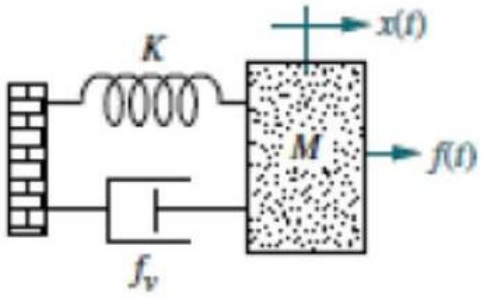
Για το ερώτημα 5 θα πρέπει να δοθεί το q αμέσως μετά τη δήλωση της μεταβλητής h όπως φαίνεται στην παρακάτω εικόνα.

```
Simulation_Tank_Lesson_2.m | Simulation_Tank_Lesson_2_dr.m | Simulation_Motion1.m

function [dx] = Simulation_Tank_Lesson_2(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
%Modeling/Differential Equations)

%Parameters
r=1000;          %kg/m3
cp=4184;         %J/K*kg
A=1;             %m2
q_0=1e-3;        %m3/s

T0=298;          %K
q_steam=50000;   %J/s
Tref=0;          %K

%State Variables
h = x(1,1);
T = x(2,1);

q=h/100;         %m3/s

%Dynamic Model of the Tank Operation
dx = zeros(2,1);

%Height h(t)
dx(1) = (1/A)*(q_0-q);

%Temperature T(t)
dx(2) = (1/(A*h))*((q_0*(T0-Tref-T))-q*Tref+q_steam/(r*cp));

return
```

Παράδειγμα 1.2.2: Κίνησης υπό την επίδραση συνισταμένης δυνάμεων

Εκφώνηση: Σώμα μάζας $M(kg)$ δέχεται την επίδραση της δύναμης $f(N)$ για να κινηθεί. Στο ίδιο σώμα, επιδρούν δύο αντίθετες δυνάμεις:

1) Η δύναμη που ασκεί το ελατήριο
2) Η δύναμη του αποσβεστήρα



Να αναπτύξετε το μαθηματικό μοντέλο που θα περιγράφει την μεταβολή της απόστασης $x(m)$ με τον χρόνο. Οι σταθερές τιμές είναι :

$$f = 5\ N \qquad f_v = 0.2\ \frac{kg}{s}$$

$$M = 100\ kg \qquad K = 1\frac{kg}{s^2}$$

$$x(t=0) = x_1(t=0) = 0m$$

Επίλυση: Ξεκινάμε αναπτύσσοντας τις διαφορικές εξισώσεις καθώς και τις βοηθητικές μεταβλητές για να μετατρέψουμε την διαφορική 2ης τάξης σε 2 διαφορικές 1ης τάξης. Οι εξισώσεις μας θα είναι :

$$f(t) = M\frac{d^2x(t)}{dt^2} + f_v\frac{dx(t)}{dt} + Kx(t)$$

$$x_1 = x$$

$$x_2 = \frac{dx}{dt} = \frac{dx_1}{dt} \Rightarrow \cdots \Rightarrow \frac{dx_2}{dt} = \frac{d^2x}{dt^2} = \frac{f}{M} - \frac{f_v}{M}x_2 - \frac{K}{M}x_1$$

Ξεκινάμε όπως ακριβώς και στο Παράδειγμα #1 δημιουργώντας ένα αρχείο m.file, δίνοντάς του όνομα και στη πρώτη γραμμή δηλώνοντας μια συνάρτηση function, προσέχοντας να έχει το ίδιο ακριβώς όνομα με το αρχείο που μόλις δημιουργήσαμε. Στη συνέχεια δηλώνουμε τις σταθερές μεταβλητές του προβλήματός μου που είναι οι f, M, f_v, K.

```
function [dx] = Simulation_Motion1(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
Modeling/Differential Equations)

%Parameters
f=5;         %N
M=100;       %m3/s
fv=0.2;      %kg/s
K=1;         %kg/s2
```

Στη συνέχεια δηλώνουμε δύο βοηθητικές μεταβλητές

```
%State Variables
x1 = x(1,1);
x2 = x(2,1);
```

Τέλος, ορίζουμε τις διαφορικές εξισώσεις όπως φαίνεται παρακάτω:

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = \frac{f}{M} - \frac{f_v}{M}x_2 - \frac{K}{M}x_1$$

ενώ παράλληλα δεν ξεχνάμε να ορίσουμε το μέγεθος του προβλήματος dx = zeros(2,1) και να τερματίσουμε τον κώδικα με την εντολή *return.*

```
%Dynamic Model of the mass-object Motion
```

```
dx = zeros(2,1);

%Help Variable x1(t)
dx(1)=x2;

%Help Variable x2(t)
dx(2)=(f/M)-(fv/M)*x2-(K/M)*x1;

return
```

Έχοντας ολοκληρώσει το πρώτο κομμάτι του κώδικα, ανοίγουμε ένα καινούργιο αρχείο m.file. Ξεκινάμε, όπως και στο Παράδειγμα #1 με τις εντολές *clear* και *clc*, για να καθαρίσουμε το ιστορικό εκτέλεσης στο Matlab από παλαιότερα αποτελέσματα. Έπειτα ορίζουμε τον χρόνο προσομοίωσης που αυτή τη φορά θα είναι από t=0 s έως t=1000 s με βήμα 1s.

```
clear
clc

tspan=[0:1:1000];
```

Συνεχίζουμε δίνοντας αρχικές τιμές, που κάποιες φορές ίσως πρέπει να είναι ένα πολύ χαμηλό νούμερο (π.χ. 0.00000001) για να αποφύγουμε θέματα αρχικοποίησης (initialization).

```
%Initial Values for State Variables
x0(1)=0+1e-8;
x0(2)=0+1e-8;
```

Χρησιμοποιείται η μέθοδος Runge - Kutte(ode45) ακριβώς όπως στο Παράδειγμα #1 της Ενότητας 1.2, και προσέχοντας το όνομα του αρχείου που έχουμε τις μεταβλητές μας, να είναι οπωσδήποτε το ίδιο με τον προηγούμενο κώδικα. Επίσης, δηλώνεται ο χρόνος προσομοίωσης (tspan) και οι αρχικές συνθήκες (t=0). Τέλος, για το αποτέλεσμα της μορφής διαγράμματος μας ενδιαφέρει η μεταβλητή $x_1 = x$ που ισούται με την πραγματική τιμή της απόστασης X:

```
[tsol,xsol]=ode45(@Simulation_Motion1,tspan,x0);

figure(1);plot(tsol,xsol(:,1));xlabel('Time, s');ylabel('Distance x, m')
```

Τρέχοντας τον αλγόριθμό μας, θα πρέπει να έχουμε το εξής διάγραμμα που δείχνει την ταλάντωση του σώματος.

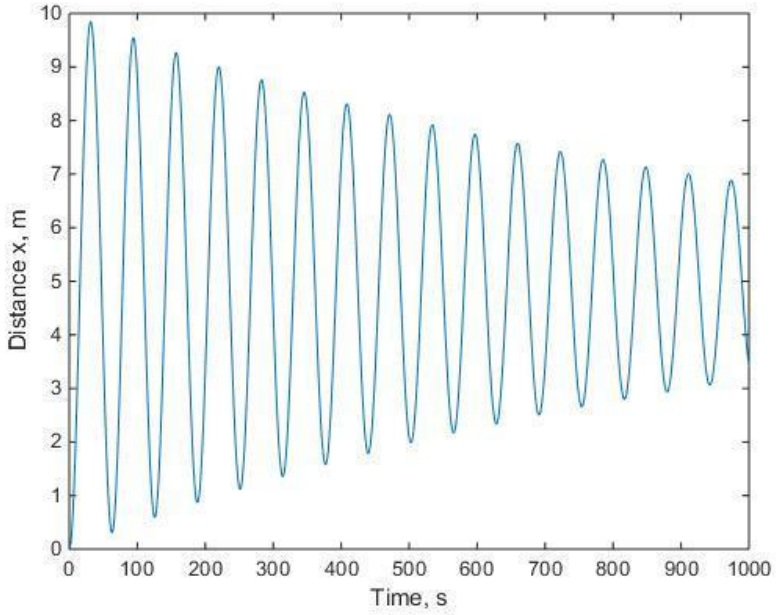
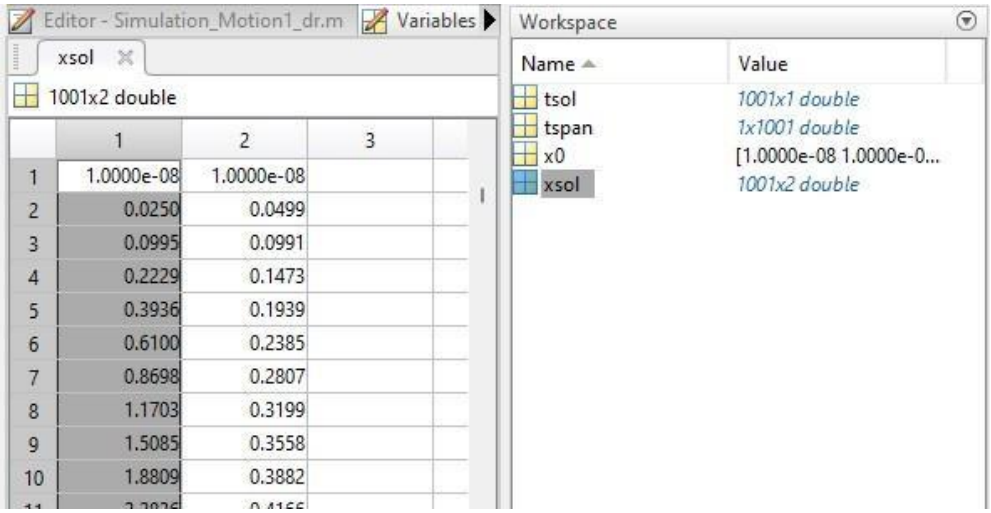


Figure 1

Παράλληλα στον πίνακα-διάνυσμα xsol, που εμφανίστηκε στην περιοχή Workspace, μας ενδιαφέρει η πρώτη στήλη όπως φαίνεται στην ακόλουθη φωτογραφία.



Αν αυξήσουμε τον χρόνο προσομοίωσης από $t = 1000\ s$ σε $t = 10000\ s$ αλλάζοντας την εντολή *tspan=[0:1:1000]* σε *tspan=[0:1:10000]*, θα δούμε ότι η ταλάντωση του αντικειμένου ολοκληρώνεται μετά από περίπου 5000-6000s, όπως γίνεται προφανές και από το αντίστοιχο διάγραμμα (μόνιμη κατάσταση).

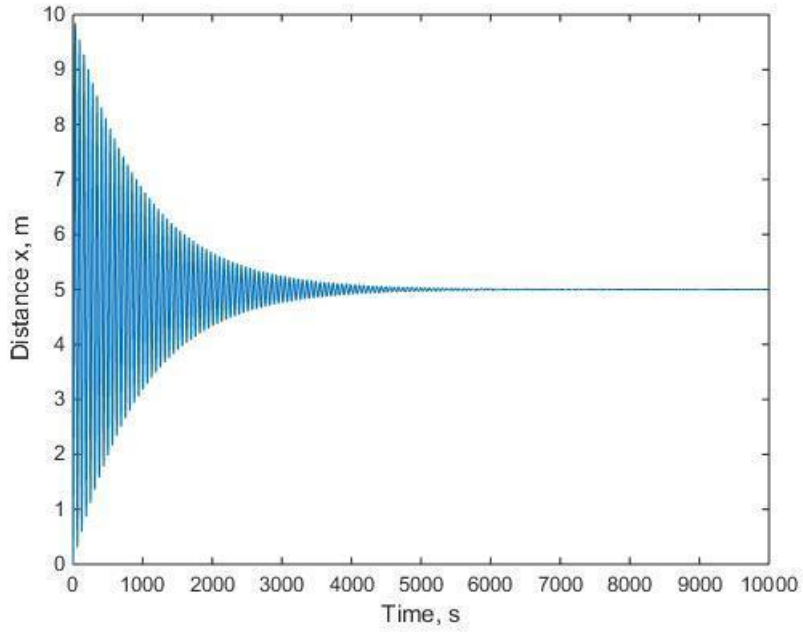


Figure 2

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 1.2.2)

Εξάσκηση

Εκφώνηση: Με βάση τα προηγούμενα, να επιλύσετε υπολογιστικά τις αποκρίσεις (δηλ. την εξάρτηση από τον χρόνο) της απόστασης εάν:

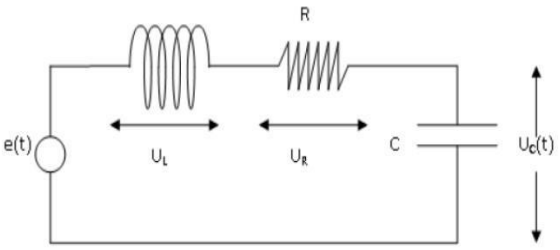
1) $M = 1kg$ και όλα τα υπόλοιπα όπως πριν.
2) $K = 100$ και όλα τα υπόλοιπα όπως πριν.
3) $f_v = 2$ και όλα τα υπόλοιπα όπως πριν.
4) $f = 500$ και όλα τα υπόλοιπα όπως πριν.
5) $f = \cos(t)$ και όλα τα υπόλοιπα όπως πριν.

Για το ερώτημα 5 θα πρέπει να δώσετε τη δύναμη $f = \cos(Time)$ όπως φαίνεται στην παρακάτω εικόνα.

```
function [dx] = Simulation_Motion1(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
%Modeling/Differential Equations)

%Parameters
f=cos(Time);     %N
M=100;           %m3/s
fv=0.2;          %kg/s
K=1;             %kg/s2

%State Variables
x1 = x(1,1);
x2 = x(2,1);

%Dynamic Model of the mass-object Motion
dx = zeros(2,1);

%Help Variable x1(t)
dx(1)=x2;

%Help Variable x2(t)
dx(2)=(f/M)-(fv/M)*x2-(K/M)*x1;

return
```

Παράδειγμα 1.2.3: Μελέτη Ηλεκτρικών Κυκλωμάτων RLC

Εκφώνηση: Το ηλεκτρικό κύκλωμα (RLC) περιλαμβάνει πηγή τάσης $e(V)$, τάση πηνίου $V_R(V)$, τάση πυκνωτή $V_c(V)$ καθώς διαπερνά ρεύμα έντασης $I(A)$. Να αναπτύξετε το μαθηματικό μοντέλο που θα περιγράφει τη μεταβολή της τάσης του πυκνωτή $V_c(V)$ με τον χρόνο.



Οι σταθερές τιμές είναι :

$$e = 5\ V \qquad C = 15\ \mu F$$
$$R = 10\ \Omega \qquad L = 230\ mH$$
$$V_C(t=0) = x_1(t=0) = 0\ V$$

Επίλυση: Ξεκινάμε αναπτύσσοντας τις διαφορικές εξισώσεις :

$$LC\frac{d^2V_C}{dt^2} + RC\frac{dV_C}{dt} + V_C = e(t)$$

$$x_1 = V_C$$

$$x_2 = \frac{dV_C}{dt} = \frac{dx_1}{dt} \Rightarrow \cdots \Rightarrow \frac{dx_2}{dt} = \frac{d^2V_C}{dt^2} = \frac{1}{LC}(e(t) - x_1 - RCx_2)$$

Όπως ακριβώς και στα προηγούμενα δύο παραδείγματα, δημιουργούμε ένα αρχείο m.file, δίνοντάς του όνομα και στη πρώτη γραμμή δηλώνοντας μια συνάρτηση function, προσέχοντας να έχει το ίδιο ακριβώς όνομα με το αρχείο που μόλις δημιουργήσαμε. Στη συνέχεια, δηλώνουμε τις σταθερές μεταβλητές του προβλήματός μου που είναι οι e, R, C, L.

```
function [dx] = Simulation_RLC(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
%Modeling/Differential Equations)

%Parameters
Voltage=5;      %V
R=10;           %?
C=15;           %?F
L=230;          %mH
```

Στη συνέχεια δηλώνουμε τις δύο βοηθητικές μεταβλητές x_1, x_2. Προαιρετικά, προσθέτουμε μία εντολή για να μας δώσει και το ρεύμα I πολλαπλασιάζοντας την σταθερά C με την βοηθητική μεταβλητή x_2.

```
%State Variables
x1 = x(1,1);
x2 = x(2,1);

i=C*x2;
```

Τέλος, ορίζουμε τις διαφορικές μας εξισώσεις όπως φαίνεται παρακάτω, ενώ παράλληλα δεν ξεχνάμε να ορίσουμε το μέγεθος του προβλήματος και να τερματίσουμε τον κώδικα με την εντολή return:

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = \frac{1}{LC}(e(t) - x_1 - RCx_2)$$

```
%Dynamic Model of the RLC
dx = zeros(2,1);

%Help Variable x1(t)
dx(1) = x2;

%Help Variable x2(t)
dx(2) = (Voltage-x1-R*C*x2)/(L*C);

return
```

Έχοντας ολοκληρώσει το πρώτο κομμάτι του κώδικα, ανοίγουμε ένα καινούργιο αρχείο m.file. Ξεκινάμε, όπως και στα προηγούμενα παραδείγματα με τις εντολές *clear* και *clc*, για να καθαρίσουμε το ιστορικό εκτέλεσης στο Matlab από παλαιότερα αποτελέσματα. Έπειτα ορίζουμε τον χρόνο προσομοίωσης που αυτή τη φορα θα είναι από t=0 s έως t=1000 s.

```
clear
clc

tspan=[0:1:1000];
```

Συνεχίζουμε δίνοντας αρχικές τιμές, προτιμώντας για άλλη μια φορά να μην είναι το απόλυτο μηδέν, αλλά ένα πολύ χαμηλό νούμερο (π.χ. 0.00000001).

```
%Initial Values for State Variables
x0(1)=0+1e-8;
x0(2)=0+1e-8;
```

Χρησιμοποιώντας τη μέθοδο Runge - Kutte(ode45) ακριβώς όπως στα προηγούμενα παραδείγματα, προσέχοντας το όνομα του αρχείου που έχουμε τις μεταβλητές μας, να είναι οπωσδήποτε γραμμένο το ίδιο, όπως επίσης και ο χρόνος προσομοίωσης (tspan) και οι αρχικές συνθήκες (t=0). Τέλος για το αποτέλεσμα της μορφής διαγράμματος μας ενδιαφέρει η μεταβλητή $x_1 = x$ που ισούται με την πραγματική τιμή της τάσης του πυκνωτή. Άρα έχουμε :

```
[tsol,xsol]=ode45(@Simulation_RLC,tspan,x0);

figure(1);plot(tsol,xsol(:,1));xlabel('Time, s');ylabel('Voltage Vc, V')
```

Τρέχοντας τον αλγόριθμό μας, θα πρέπει να έχουμε το εξής διάγραμμα.

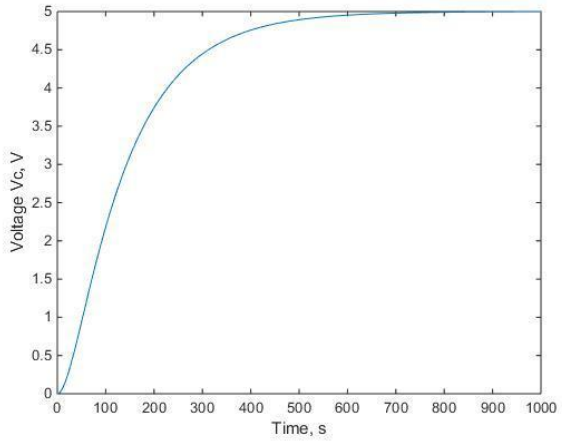
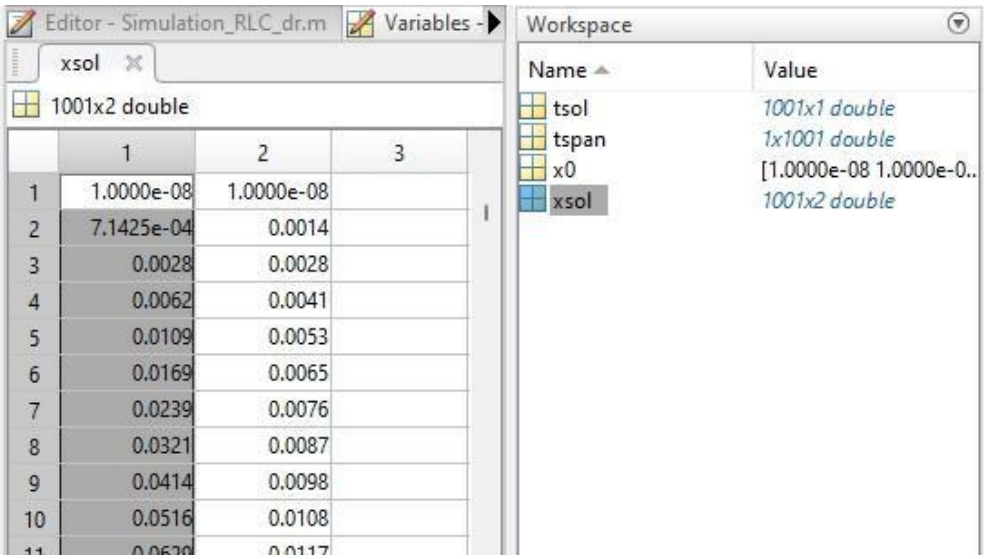


Figure 1

Ενώ στον πίνακα xsol, που εμφανίστηκε στην περιοχή Workspace, μας ενδιαφέρει η πρώτη στήλη όπως φαίνεται στην ακόλουθη φωτογραφία.



Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 1.2.3)

Εξάσκηση

Εκφώνηση: Με βάση τα προηγούμενα, να επιλύσετε υπολογιστικά τις αποκρίσεις (δηλ. την εξάρτηση από τον χρόνο) της τάσης του πυκνωτή εάν:

1) $\mathrm{R} = 100\Omega$ και όλα τα υπόλοιπα όπως πριν.
2) $\mathrm{C} = 1\,\mu\,\mathrm{F}$ και όλα τα υπόλοιπα όπως πριν.
3) $\mathrm{e} = 50\mathrm{V}$και όλα τα υπόλοιπα όπως πριν.
4) $\mathrm{e} = 50 \cdot \cos(\mathrm{t})$ και όλα τα υπόλοιπα όπως πριν.

ΚΕΦΑΛΑΙΟ 2: Laplace

Στο δεύτερο κεφάλαιο θα λυθούν υπολογιστικά οι, αντίστροφοι και μη-, μετασχηματισμοί Laplace. Επιπλέον θα προσδιοριστούν με χρήση απλών εντολών οι πόλοι (ρίζες του παρονομαστή) και τα μηδενικά (ρίζες του αριθμητή) των αντίστοιχων συναρτήσεων μεταφοράς στο πεδίο συχνότητας (Laplace). Έτσι, οι επιμέρους στόχοι του κεφαλαίου είναι:

- *Εισαγωγή στους μετασχηματισμούς Laplace στο λογισμικό του Matlab.*
- *Εισαγωγή στους αντίστροφους μετασχηματισμούς Laplace στο λογισμικό του Matlab.*
- *Εισαγωγή στην έννοια των πόλων και μηδενικών.*

2.1 Επίλυση Μετασχηματισμού Laplace

Στη συνέχεια, θα παρουσιαστούν παραδείγματα επίλυσης μετασχηματισμών Laplace.

Παράδειγμα 2.1.1: Επίλυση Μετασχηματισμού Laplace

Εκφώνηση: Να βρεθεί ο Μετασχηματισμός Laplace της συνάρτησης (μη-γραμμική και αλγεβρική σχέση) με χρήση εντολών στο Matlab.

$$\boldsymbol{h(t) = 5t + 8t^2 + sin(3t) + 5e^{-at},\ \ a = 6}$$

Επίλυση: Ξεκινάμε με την εντολή *syms* με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις συμβολικές μεταβλητές.

```
syms t
```

Έπειτα, θα δηλώσουμε την συνάρτηση $h(t)$ όπως φαίνεται παραπάνω στην εκφώνηση.

```
a=6;
h_help=5*t+8*t^2+sin(3*t)+5*exp(-a*t);
```

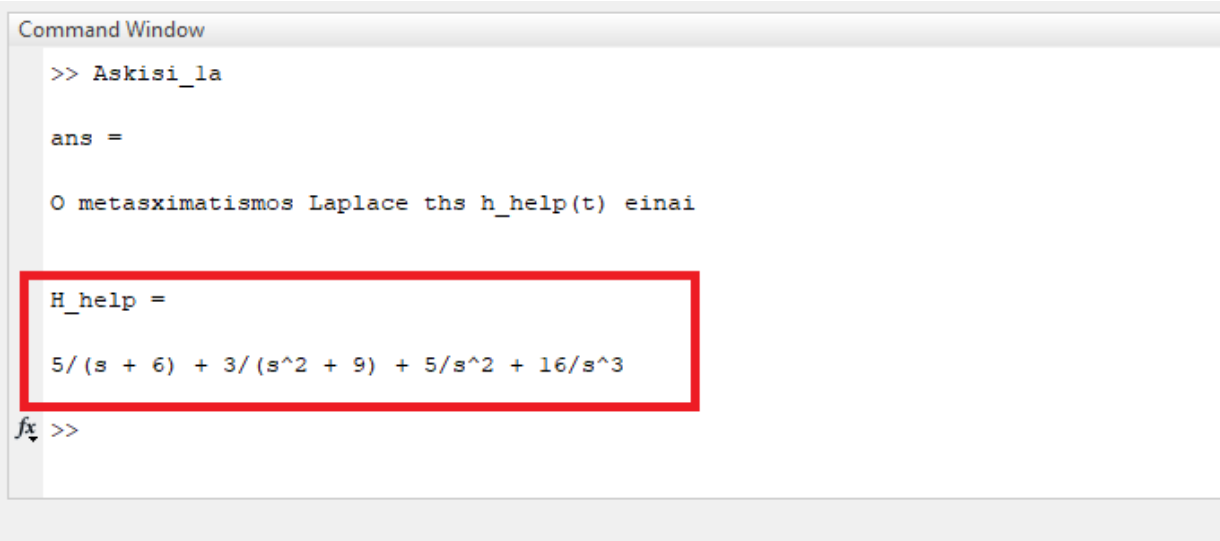
Στη συνέχεια, με την εντολή *laplace*, ζητάμε να υπολογιστεί ο μετασχηματισμός Laplace της συνάρτησης που δηλώσαμε στο προηγούμενο βήμα.

```
H_help=laplace(h_help);
```

Τέλος, χρησιμοποιώντας το σύμβολο της μονής αποστρόφου (‘) ζητάμε από το Matlab να συνοδεύσει το αποτέλεσμα με μια μικρή φράση της επιλογής μας. *Πολλές φορές, οι Ελληνικοί Χαρακτήρες δεν αναγνωρίζονται κατά την εκτέλεση, οπότε χρησιμοποιούμε και λατινικούς.*

```
'O metasximatismos Laplace ths h_help(t) einai'
H_help
```

Εκτελώντας το αρχείο, εμφανίζεται ο μετασχηματισμός Laplace (H_help) της αρχικής συνάρτησης (h_help), με την αντιστοιχία του αποτελέσματος στο πεδίο Laplace με το πεδίο του χρόνου.



Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 2.1.1)

Εξάσκηση

Εκφώνηση: Να βρείτε τους μετασχηματισμούς Laplace των παρακάτω αλγεβρικών συναρτήσεων αναπτύσσοντας κατάλληλες εντολές σε μορφή κώδικα σε ένα αρχείο m.file:

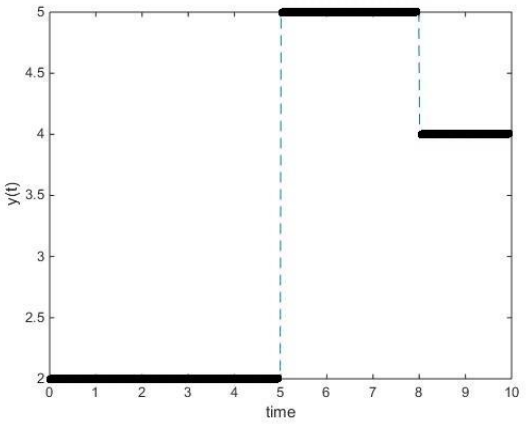
1) $h_1(t) = \sin(5 \cdot t) + 5 \cdot e^{-8t}$
2) $h_2(t) = 7 \cdot t^3 + 2 \cdot t \cdot e^{-\alpha \cdot t} + 6$, όπου $\alpha =$ 3 τελευταία ψηφία του Α.Μ. /3
3) $h_3(t) = h_1(t) \cdot e^{-b \cdot t}$, όπου $b =$ έτος γέννησης / 3 τελευταία ψηφία του Α.Μ. (στρογγυλοποιημένο σε ακέραιη μορφή)
4) $h_4(t) = h_2(t) \cdot e^{-c \cdot t}$, όπου c =[(έτος γέννησης / 3 τελευταία ψηφία του Α.Μ.) – 1] (στρογγυλοποιημένο σε ακέραιη μορφή)

Για την στρογγυλοποίηση χρησιμοποιούμε την εντολή *round* ως εξής:

```
c=(1982/025)-1;
C=round(c)
```

Παράδειγμα 2.1.2 : Επίλυση Μετασχηματισμού Laplace

Εκφώνηση: Να εκφράσετε τις παρακάτω παραστάσεις υπό μορφή συναρτήσεων στο πεδίο του χρόνου και στο πεδίο Laplace.



$$y(t) = \begin{cases} 2, & t < 5 \\ 5, & 5 \le t \le 8 \\ 4, & t > 8 \end{cases}$$

Επίλυση: Χρησιμοποιούμε(βοηθητικά) την συνάρτηση Heaviside, καθώς γνωρίζουμε ότι $H(t) = 1$. Σε αυτή τη μορφή η παράστασή μας θα είναι:

$$\mathrm{y}(t) = 2 \cdot H(t) + 3 \cdot H(t-5) - 1 \cdot H(t-8)$$

Ξεκινάμε με την εντολή *syms*, όπως και στο προηγούμενο παράδειγμα, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Δηλώνουμε την συνάρτηση Heaviside με την εντολή *heaviside* ως εξής:

```
clear all
clc

syms t s

f=2+3*heaviside(t-5)-1*heaviside(t-8);
% h
%f=2*heaviside(t)+3*heaviside(t-5)-1*heaviside(t-8);
```

Στη συνέχεια, με την εντολή *laplace* βρίσκουμε τον μετασχηματισμό Laplace, ενώ για τον αντίστροφο μετασχηματισμό χρησιμοποιούμε την εντολή *ilaplace* ως εξής:

```
F=laplace(f)
f1=ilaplace(F)
```

Για να εμφανίσουμε το διάγραμμα, αρκεί να πληκτρολογήσουμε τις εξής εντολές όπου η εντολή subs χρησιμοποιείται για να αντικαταστήσει στην y(t) τιμές του χρόνου t, ενώ η εντολή double ολοκληρώνει τον παραπάνω υπολογισμό σε πραγματικές τιμές.

```
t=0:0.01:10;
y=subs(f1,t);
y=double(y);

figure(1);plot(t,y,'--');ylabel('y(t)');xlabel('time');
```

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 2.1.2)

2.2 Επίλυση Αντίστροφου Μετασχηματισμού Laplace

Στη συνέχεια, θα παρουσιαστούν παραδείγματα επίλυσης αντίστροφων μετασχηματισμών Laplace.

Παράδειγμα 2.2.1:

Εκφώνηση: Να βρείτε τον αντίστροφο μετασχηματισμό Laplace των παρακάτω μαθηματικών εκφράσεων (θα επιλύσετε ως προς $Y(t)$).

$$Y(s) = \frac{b(s)}{a(s)} = \frac{s^2 - s - 6}{(s-1)(s+1)(s-2)} \qquad Y(s) = \frac{b(s)}{a(s)} = \frac{s+1}{s^2 - 2s + 5}$$

$$Y(s) = \frac{b(s)}{a(s)} = \frac{s^2 + 2s + 3}{(s+1)^3} \qquad Y(s) = \frac{b(s)}{a(s)} = \frac{2e^{-0.5s}}{s+1}$$

$$Y(s) = \frac{b(s)}{a(s)} = \frac{s+1}{s^2(s^2 + 4s + 5)(s-1)}$$

Επίλυση: Ξεκινάμε όπως και στην προηγούμενη ενότητα, με την εντολή *syms*. Στη συνέχεια δηλώνουμε τις συναρτήσεις, έτσι όπως μας τις δίνει η εκφώνηση.

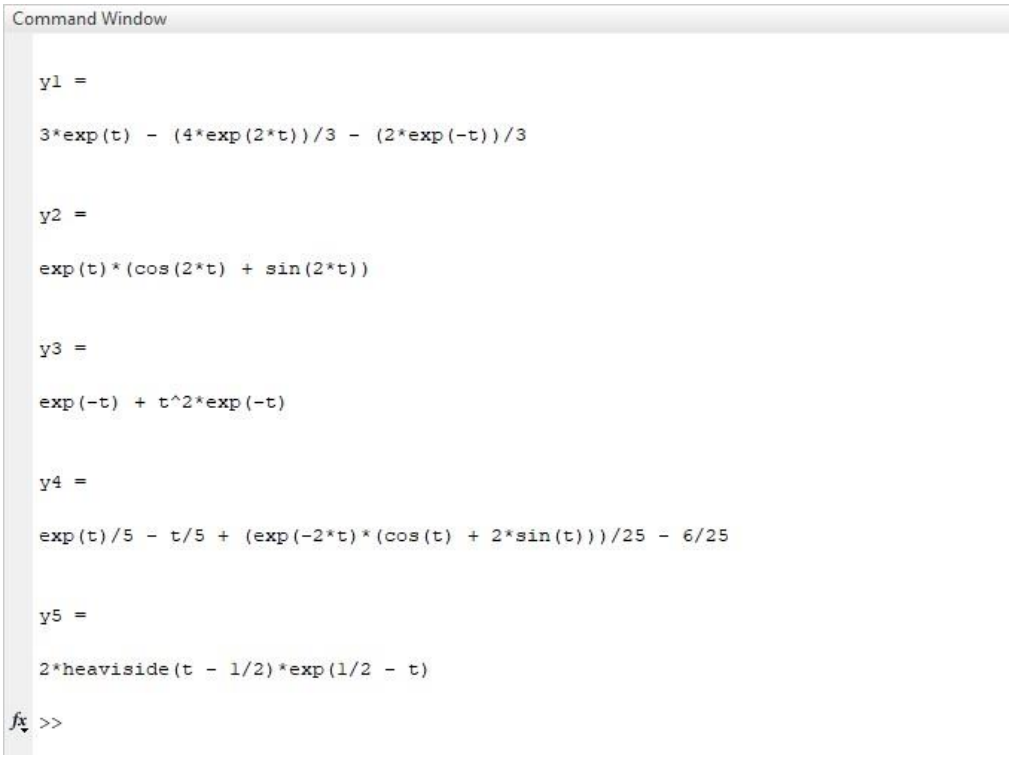
```
% Inverse Laplace
clear all
clc

syms s
Y1=(s^2-s-6)/((s-1)*(s+1)*(s-2));
Y2=(s+1)/(s^2-2*s+5);
Y3=(s^2+2*s+3)/(s+1)^3;
Y4=(s+1)/((s^2)*(s^2+4*s+5)*(s-1));
Y5=2*(exp(-0.5*s))/(s+1);
```

Τέλος, προσδιορίζουμε τον αντίστροφο μετασχηματισμό Laplace κάθε μιας συνάρτησης, χρησιμοποιώντας την εντολή *ilaplace*.

```
y1=ilaplace(Y1)
```

```
y2=ilaplace(Y2)
y3=ilaplace(Y3)
y4=ilaplace(Y4)
y5=ilaplace(Y5)
```

Εκτελώντας το αρχείο, εμφανίζονται οι αντίστροφοι μετασχηματισμοί Laplace των μαθηματικών εκφράσεων.



Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 2.2.1)

Παράδειγμα 2.2.2: Εύρεση Ριζών Αριθμητή (Μηδενικά) και Παρονομαστή (Πόλοι)

Εκφώνηση: Για τις παρακάτω συναρτήσεις κατά Laplace, να βρεθούν οι ρίζες του αριθμητή (μηδενικά) και του παρονομαστή (πόλοι).

$$Y(s) = \frac{b(s)}{a(s)} = \frac{s^2 - s - 6}{(s-1)(s+1)(s-2)} \qquad Y(s) = \frac{b(s)}{a(s)} = \frac{s+1}{s^2 - 2s + 5}$$

$$Y(s) = \frac{b(s)}{a(s)} = \frac{s^2 + 2s + 3}{(s+1)^3} \qquad Y(s) = \frac{b(s)}{a(s)} = \frac{s+1}{s^2(s^2 + 4s + 5)(s-1)}$$

Επίλυση: Ξεκινάμε με τον ίδιο ακριβώς τρόπο με το προηγούμενο παράδειγμα, δηλώνοντας τις συναρτήσεις, έτσι όπως μας τις δίνει η εκφώνηση.

```
clear all
clc

syms s
Y1=(s^2-s-6)/((s-1)*(s+1)*(s-2));
Y2=(s+1)/(s^2-2*s+5);
Y3=(s^2+2*s+3)/(s+1)^3;
Y4=(s+1)/((s^2)*(s^2+4*s+5)*(s-1));
```

Στη συνέχεια, ορίζουμε με ποια συνάρτηση (Y1,...,Y4) θέλουμε να εργαστούμε.

```
H=Y1;
```

Έπειτα, με την εντολή *numden(function)*, βρίσκουμε την συνάρτηση του αριθμητή και του παρονομαστή.

```
[num,den]=numden(H)
```

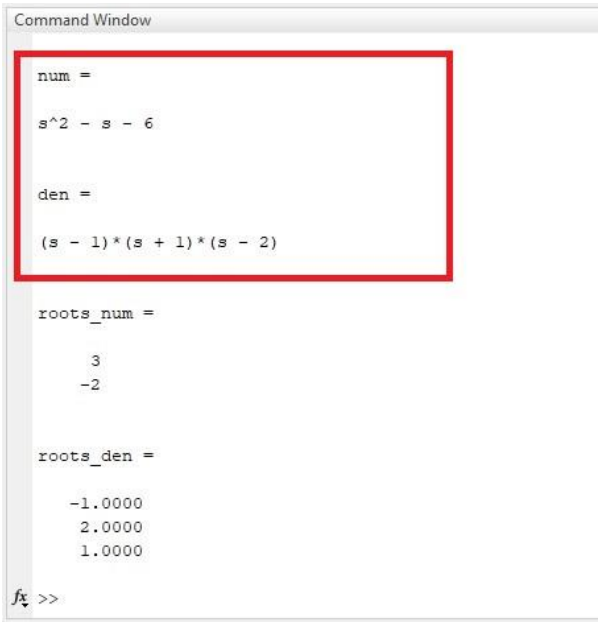
Στη συνέχεια, η εντολή *sym2poly* μας επιτρέπει να μετατρέψουμε τις συμβολικές μεταβλητές σε πραγματικές. Επομένως, ο αριθμητής και ο παρονομαστής νοείται ως πολυώνυμο πλέον, και έτσι μπορούμε να τον επεξεργαστούμε μαθηματικά.

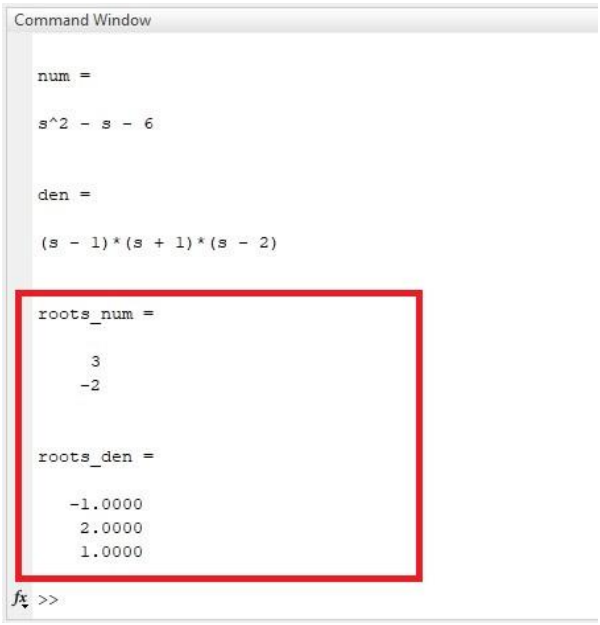
```
num=sym2poly(num);
den=sym2poly(den);
```

Τέλος, η εντολή *roots* μας επιτρέπει την εύρεση των ριζών ενός πολυωνύμου. Αντιστοιχούμε τον αριθμητή με *num* και το παρονομαστή με το *den*.

```
%MIDENIKA
roots_num=roots(num)
%POLOI
roots_den=roots(den)
```

Εκτελώντας το αρχείο, έχουμε τον αριθμητή (numerator, num) και τον παρονομαστή (denominator, den) με την μορφή Laplace (συμβολικές μεταβλητές).





Τέλος, έχουμε τις ρίζες (μηδενικά) του αριθμητή (*roots_num*) με 2 πραγματικές ρίζες και τις ρίζες (πόλοι) του παρονομαστή (*roots_den*) με 3 πραγματικές ρίζες.

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 2.2.2)

2.3 Εύρεση Μετασχηματισμού Laplace σε Διαφορικές Εξισώσεις / Επίλυση του Μετασχηματισμού Laplace και αποτελέσματα υπό μορφή Διαγραμμάτων

Στη συνέχεια, θα παρουσιαστούν παραδείγματα εύρεσης Μετασχηματισμού Laplace σε Διαφορικές Εξισώσεις και επίλυσης του μετασχηματισμού με αποτελέσματα υπό μορφής Διαγραμμάτων (εναλλακτικός τρόπος επίλυσης διαφορικών εξισώσεων n-τάξης).

Παράδειγμα 2.3.1:

Εκφώνηση: Να επιλυθεί πλήρως στο επίπεδο του χρόνου με την βοήθεια μετασχηματισμών Laplace η συνάρτηση:

$$\frac{d^3y}{dt^3} + 2 \cdot \frac{d^2y}{dt^2} + 3 \cdot \frac{dy}{dt} = 20 \cdot sin(2t)$$

Με αρχικές συνθήκες: $y(0) = y'(0) = y''(0) = 0$

Επίλυση: Ξεκινάμε με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις συμβολικές μεταβλητές.

```
clear all
clc

syms y t s Y F
```

Στη συνέχεια, δηλώνουμε την συνάρτηση που μας δίνει η εκφώνηση, μέσα στην εντολή *laplace* έτσι ώστε να υπολογιστή ο μετασχηματισμός Laplace.

```
F=laplace('diff(y(t),t,t,t)+2*diff(y(t),t,t)+3*diff(y(t),t)=20*sin(2*t)',s)
```

Μέχρι αυτό το σημείο, το αποτέλεσμα στο Matlab θα είναι:

```
F =

3*s*laplace(y(t), t, s) - 3*y(0) - 2*D(y)(0) - D(D(y))(0) - 2*s*y(0) +
2*s^2*laplace(y(t), t, s) + s^3*laplace(y(t), t, s) - s*D(y)(0) - s^2*y(0)
== 40/(s^2 + 4)
```

Έπειτα, αντικαθιστούμε την έκφραση *'laplace*$(y(t), t, s)$*'* ,που βρίσκεται στο παραπάνω αποτέλεσμα, με το Y ή με το $Y(s)$ αν θέλουμε. Για να το πετύχουμε αυτό χρησιμοποιούμε την εντολή *subs* ως εξής:

```
F=subs(F,{'laplace(y(t),t,s)'},{Y})
```

Μέχρι αυτό το σημείο, το αποτέλεσμα στο Matlab θα είναι:

```
F =

3*Y*s - 3*y(0) - 2*D(y)(0) - D(D(y))(0) - 2*s*y(0) + 2*Y*s^2 + Y*s^3 -
s*D(y)(0) - s^2*y(0) == 40/(s^2 + 4)
```

Με τον ίδιο τρόπο, αντικαθιστούμε τις εκφράσεις για τις αρχικές συνθήκες. Δηλαδή την έκφραση *'$y(0)$'* με το 0, την έκφραση *'$D(y)(0)$'* με το 0 και την έκφραση *'$D(D(y))(0)$'* με το 0.

```
F=subs(F,{'y(0)','D(y)(0)','D(D(y))(0)'},{0,0,0})
```

Μέχρι αυτό το σημείο, το αποτέλεσμα στο Matlab θα είναι:

```
F =

Y*s^3 + 2*Y*s^2 + 3*Y*s == 40/(s^2 + 4)
```

Στη συνέχεια, θα λύσουμε την παραπάνω συνάρτηση ως προς το Y, με την εντολή *solve*.

```
Y=solve(F,'Y')
```

Μέχρι αυτό το σημείο, το αποτέλεσμα στο Matlab θα είναι:

```
Y =

40/((s^2 + 4)*(s^3 + 2*s^2 + 3*s))
```

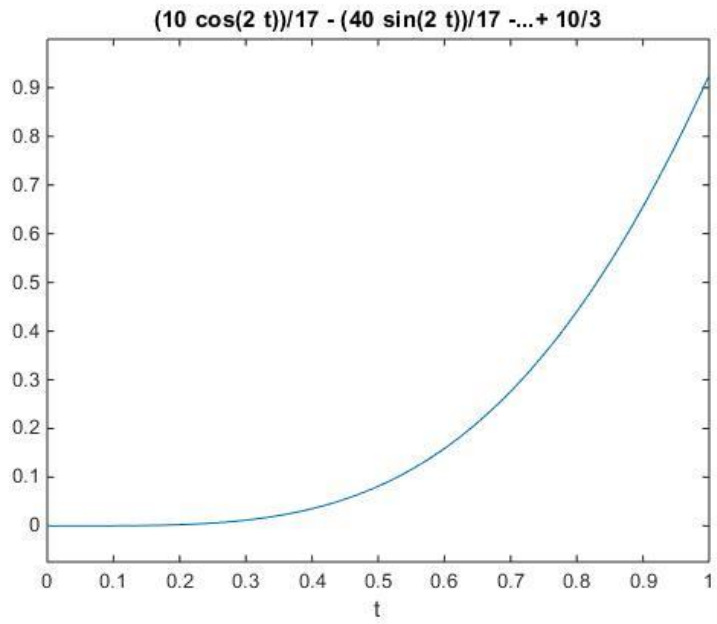
Τέλος, θα βρούμε τον αντίστροφο μετασχηματισμό Laplace ως προς τον χρόνο t της παραπάνω συνάρτησης. Δηλαδή θα λύσουμε την αρχική διαφορική εξίσωση ως προς $y(t)$. Δηλαδή την απόκριση της y στον χρόνο t. Προαιρετικά προσθέτουμε και το διάγραμμα, το οποίο θα βρει την απόκριση (επίλυση) της $y(t)$ για $t = 0 \ldots 1$

```
y=ilaplace(Y)
figure(1);ezplot(y,[0 1])
```

Οπότε το τελικό αποτέλεσμα στο Matlab θα είναι:

```
y =

(10*cos(2*t))/17-   (40*sin(2*t))/17   -   (200*exp(-t)*(cos(2^(1/2)*t)   -
(2^(1/2)*sin(2^(1/2)*t))/10))/51 + 10/3
```

Ενώ το γράφημα θα έχει τη μορφή:



Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 2.3.1)

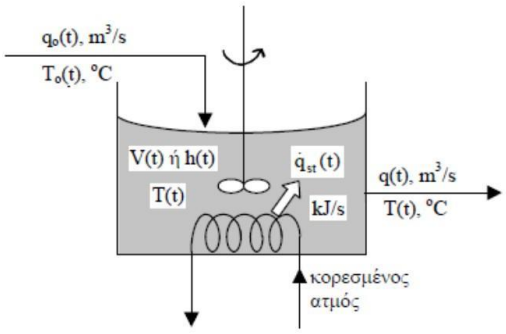
ΚΕΦΑΛΑΙΟ 3: Συνάρτηση Μεταφοράς & Σύστημα Κλειστού Βρόχου

Στο τρίτο κεφάλαιο, θα δούμε παραδείγματα απόκρισης στάθμης και θερμοκρασίας σε δοχείο καθώς και παραδείγματα αναπαράστασης συστημάτων σε διαγράμματα βαθμίδων με σκοπό την εύρεση της απόκρισης των σημάτων εισόδου, εξόδου και σφάλματος σε μορφή διαγραμμάτων. Στόχοι του κεφαλαίου, είναι :

- *Εισαγωγή στην συνάρτηση μεταφοράς (σήματα εξόδου-εισόδου)*
- *Επίλυση αποκρίσεων (συστήματα ανοικτού βρόχου) παρουσία α) εισόδου, β) εισόδου και διαταραχής*
- *Εισαγωγή στα συστήματα κλειστού βρόχου*

3.1 Δυναμικό Μοντέλο και Αναπαράσταση σε Δομικά Διαγράμματα

Όπως στο Κεφάλαιο 1.2 και παράδειγμα 1.2.1, έτσι και εδώ, έχουμε ένα δοχείο εισόδου/εξόδου στο οποίο θέλουμε να βρούμε την μεταβολή της στάθμης όπως φαίνεται στην παρακάτω εικόνα (εικόνα 3.1.1).



Εικόνα 3.1.1: Δοχείο Θέρμανσης Ρευστού

Το μαθηματικό μοντέλο που περιγράφει την μεταβολή της στάθμης με τον χρόνο είναι:

$$A\frac{dh}{dt} = q_0 - q$$

με q_0 , q μεταβλητά ή αμετάβλητα με το χρόνο. Αναπτύσσουμε την παραπάνω διαφορική εξίσωση.

$$A\frac{dh}{dt} = q_0 - q \quad \Longrightarrow \quad A \cdot [sH(s) - h(t=0)] = Q_0(s) - Q(s)$$

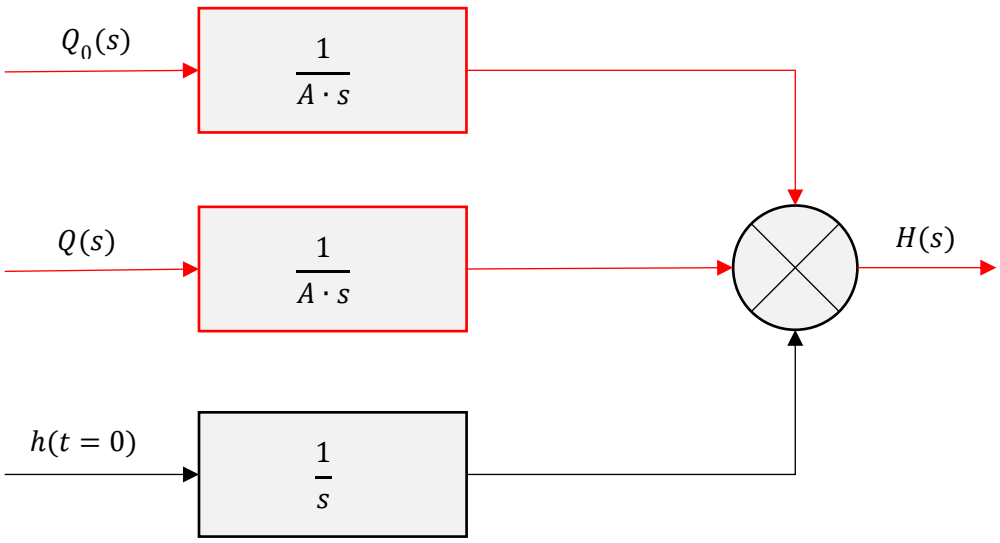
$$\Longrightarrow \quad A \cdot s \cdot H(s) = Q_0(s) - Q(s) + A \cdot h(t=0)$$

$$\Longrightarrow \quad \boldsymbol{H(s) = \frac{Q_0(s)}{A \cdot s} - \frac{Q(s)}{A \cdot s} + \frac{h(t=0)}{s}} \textit{ (εξίσ. 3.1.1)}$$

Από Θεωρία:

- Δομικό Διάγραμμα καλείται η αναπαράσταση της σχέσης εξόδου-εισόδου (ή εξόδων-εισόδων).
- Η συνάρτηση μεταφοράς περιγράφει μαθηματικά την σχέση εξόδου-εισόδου (ή εξόδων-εισόδων).

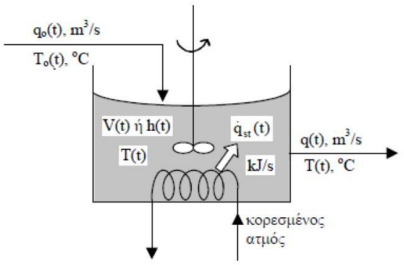
Το Δομικό Διάγραμμα που αντιστοιχεί στην εξίσωση του παραπάνω παραδείγματος (εξίσ. 3.1.1) θα είναι:



Η $h(t=0)$ δεν αποτελεί σήμα εισόδου, απλά δίνεται για λόγους κατανόησης των δομικών διαγραμμάτων.

Παράδειγμα 3.1.1: Μεταβολή στάθμης και θερμοκρασίας σε δοχείο εισόδου/εξόδου

Εκφώνηση: Στο δοχείο του διπλανού σχήματος εισέρχεται ρευστό με παροχή q_0 (m^3/s) και θερμοκρασία T_0 (ºC). Από το δοχείο εξέρχεται το ίδιο ρευστό με παροχή q (m^3/s) και θερμοκρασία T (ºC). Το ρευστό εντός του δοχείου θερμαίνεται με παροχή κορεσμένου ατμού q_{st} (kJ/s). Να αναπτύξετε το μαθηματικό μοντέλο που θα περιγράφει την μεταβολή της στάθμης h (m) και την μεταβολή της θερμοκρασίας T (℃) με τον χρόνο και στην συνέχεια να το επιλύσετε μέσω μετασχηματισμών Laplace.



Οι σταθερές τιμές είναι :

$$p = 1000\frac{kg}{m^3} \qquad q = 0.8 \cdot 10^{-3}\frac{m^3}{s} \qquad q_0 = 1 \cdot 10^{-3}\frac{m^3}{s}$$

$$C_p = 4184\frac{J}{kg}K \qquad T_0 = 298K \qquad q_{st} = 50000\frac{J}{s}$$

$$A = 1m^2 \qquad T_r = 0K \qquad T(t=0) = 288K, h(t=0) = 1m$$

Επίλυση: Ξεκινάμε με την εντολή *syms* με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις συμβολικές μεταβλητές. Στη συνέχεια δηλώνουμε τις σταθερές μεταβλητές του παραδείγματος.

```
clear all
clc

syms s

p=1000;      %kg/m3
Cp=4184;     %J/kg K
A=1;         %m2
q0=1e-3;     %m3/s
q=0.8e-3;    %m3/s
h0=1;        %m
```

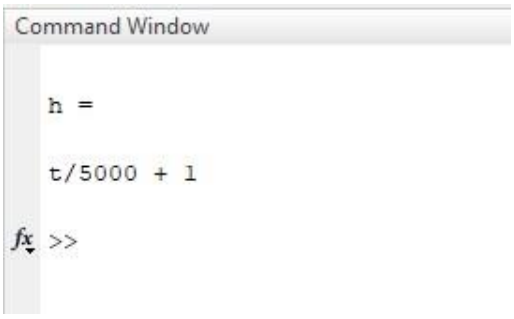
Έπειτα, θα ορίσουμε την διαφορική εξίσωση ακριβώς όπως την αναπτύξαμε παραπάνω (εξίσ. 3.1.1). Στη συνέχεια, βρίσκουμε τον αντίστροφο μετασχηματισμό με την εντολή *ilaplace*.

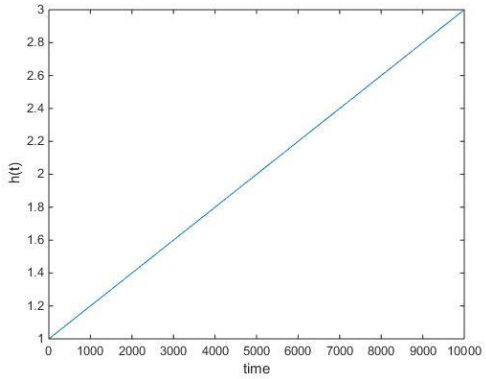
```
H=(q0/(A*s^2))-(q/(A*s^2))+h0/s;
h=ilaplace(H)
```

Εάν επιθυμήσουμε να εκφράσουμε τα αποτελέσματα μέσω διαγραμμάτων, αρκεί να πληκτρολογήσουμε τις εξής εντολές, όπου η εντολή subs χρησιμοποιείται για να αντικαταστήσει στην y(t) τιμές του χρόνου t, ενώ η εντολή double ολοκληρώνει τον παραπάνω υπολογισμό σε πραγματικές τιμές.

```
t=0:1:10000;
h=subs(h,t);
h=double(h);

figure(1);plot(t,h);xlabel('time');ylabel('h(t)')
```

Με την εκτέλεση του κώδικα παίρνουμε τα εξής αποτελέσματα:



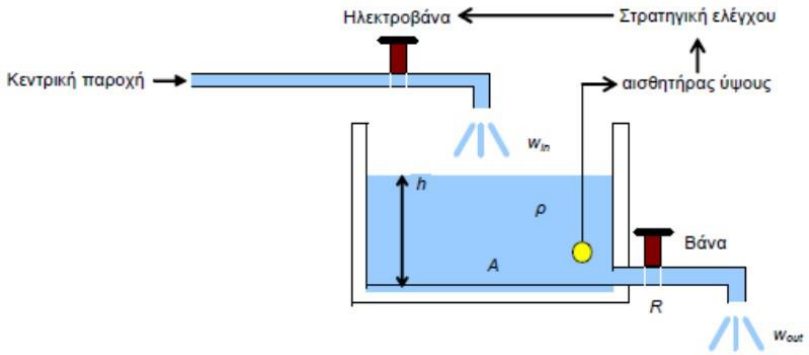


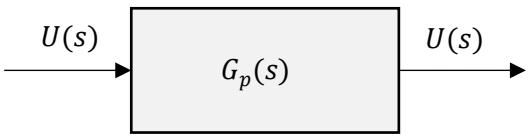
Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 3.1.1)

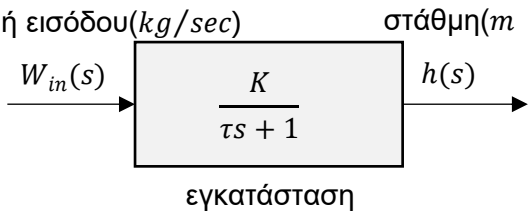
3.2 Ενδεικτικές Αποκρίσεις σε Μεταβολές Διεγέρσεων/Εισόδων

Παράδειγμα 3.2.1:

Εκφώνηση: Να βρείτε (υπολογιστικά) την απόκριση της στάθμης στο δοχείο, για τις παρακάτω εισόδους/διεγέρσεις της $win(s) = U(s)$. Θα θεωρήσετε $K = 5, \tau = 3$. Με άλλα λόγια θα προσδιορίσετε μέσω αντίστροφων μετασχηματισμών Laplace την μορφή της $h(t)$ ως προς τον χρόνο.







Σταθερή βηματική μεταβολή (step)	$u(t) = 4$ ή $U(t) = 4/s$
Μεταβολή κλίσης/ράμπας (ramp)	$u(t) = 2t$ ή $U(s) = 2/s^2$
Μεταβολή dirac	$u(t) = \delta(t)$ ή $U(s) = 1$
Τριγωνομετρική μεταβολή	$u(t) = \cos(t) + 2\sin(t)$ ή $U(s) = (s+2)/(s^2+1)$
Βηματικές/Τμηματικές μεταβολές	$u(t) = 2H(t) - H(t-4) + 2H(t-6)$ ή $U(s) = (2/s) - (1/s)e^{-4s} + (2/s)e^{-6s}$

Επίλυση: Ξεκινάμε, όπως και στο προηγούμενο παράδειγμα (βλ. Ενότητα 3.1, Παράδειγμα #1) με την εντολή *syms*, για να δηλώσουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Στη συνέχεια ορίζουμε τις διαφορετικές μορφές εισόδων/διεγέρσεων (5 σύνολο).

```
clear all
clc

syms s t

%Step Change
U1=4*heaviside(t);
U1=laplace(U1);

%Ramp Change
U2=2*t;
U2=laplace(U2);

%Dirac Change
U3=dirac(t);
U3=laplace(U3);

%Cos and Sin Change
U4=cos(t)+2*sin(t);
U4=laplace(U4);

%Step (varied) Change
U5=(2/s)-(1/s)*exp(-4*s)+(2/s)*exp(-6*s);
%U5=laplace(U5);
```

Έπειτα, επιλέγουμε όποια είσοδο/διέγερση επιθυμούμε να μελετήσουμε (μία κάθε φορά). Στη συνέχεια, ορίζουμε το σύστημα μας (π.χ. 1ης τάξης μεταβολή στάθμης σε δοχείο, 2ης τάξης κίνηση μάζας, κτλ). Συνδέουμε την έξοδο με την είσοδο μέσω της κατάλληλης συνάρτησης μεταφοράς.

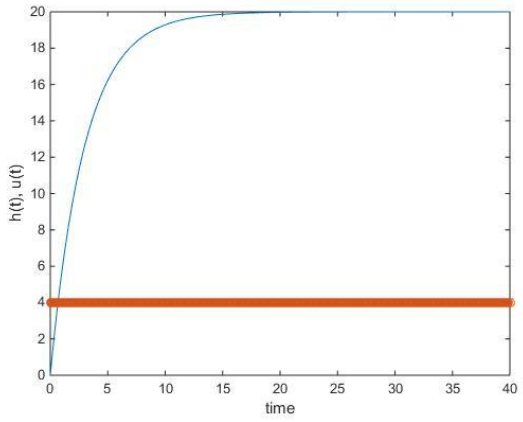
```
U=U1;

K=5;t=3;

H=(K/(t*s+1))*U;
h=ilaplace(H);
```

Τέλος, επιλύουμε ως προς το χρόνο τόσο για το σήμα εξόδου (π.χ. Στάθμη h) όσο και για το σήμα εισόδου/διέγερσης (π.χ. Βηματική μεταβολή u).

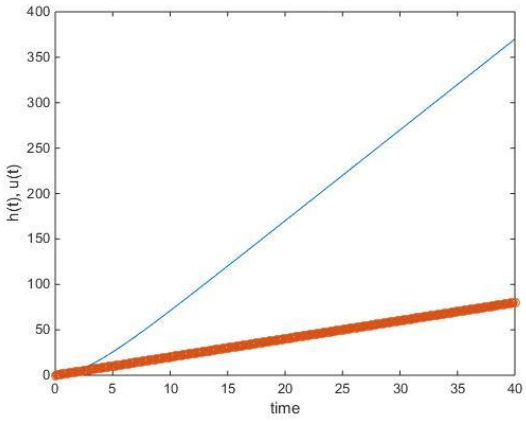
```
t=0:0.1:40;

h=subs(h,t);
h=double(h);

u=ilaplace(U);
u=subs(u,t);
u=double(u);
```

```
figure(1);plot(t,h,t,u,'o');xlabel('time');ylabel('h(t), u(t)')
```

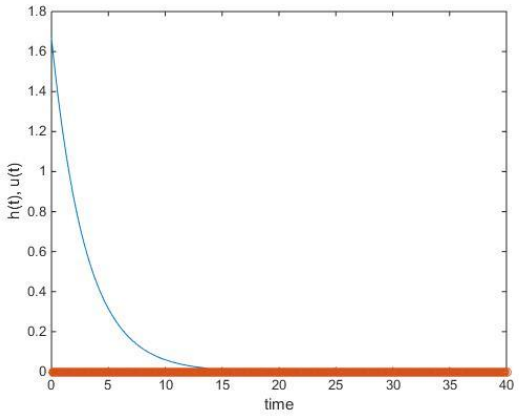
Με την εκτέλεση του κώδικα παίρνουμε τα εξής διαγράμματα, για κάθε μία από τις 5 διαφορετικές μορφές εισόδων/διεγέρσεων.



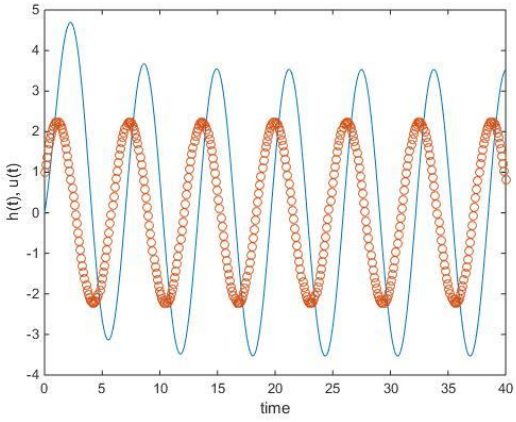
$U = U1$



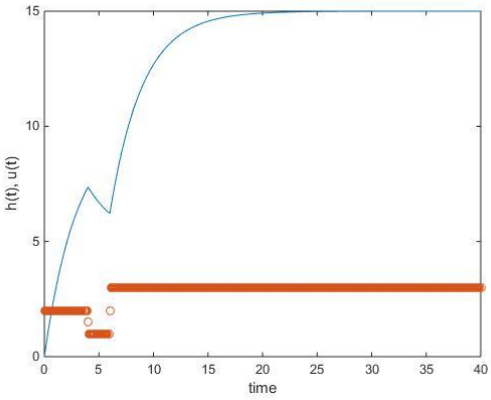
$U = U2$



$U = U3$



$U = U4$



$U = U5$

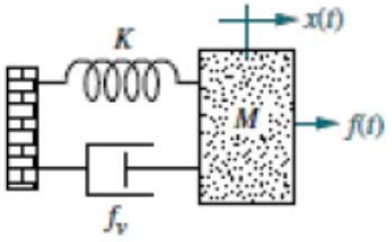
Όπου $y(t)$

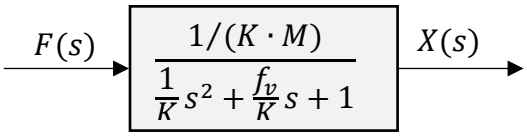
$u(t)$

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 3.2.1)

Εξάσκηση

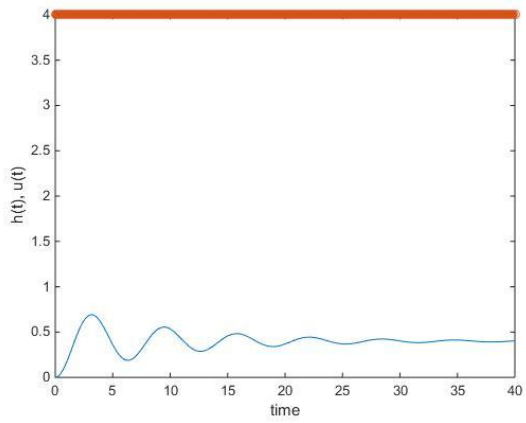
Εκφώνηση: Να βρείτε (υπολογιστικά) την απόκριση της απόστασης (κίνηση σώματος μάζας M) για δεδομένες διεγέρσεις της εισόδου (f, δύναμη). Θεωρήστε $f = 5N, M = 10kg, f_v = 0.2\,{}^{kg}/_{s}, K = 1\,{}^{kg}/_{s^2}$

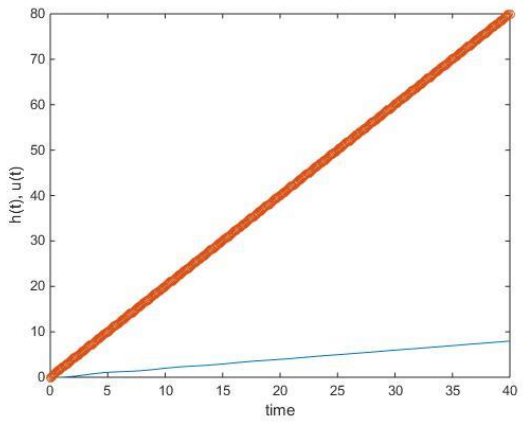




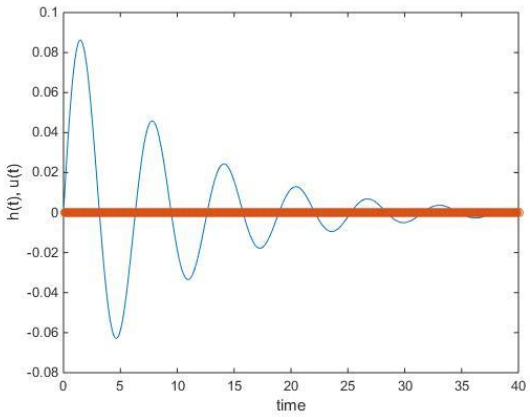
Σταθερή βηματική μεταβολή (step)	$f(t) = 4$ ή $F(t) = 4/s$
Μεταβολή κλίσης/ράμπας (ramp)	$f(t) = 2t$ ή $F(s) = 2/s^2$
Μεταβολή dirac	$f(t) = \delta(t)$ ή $F(s) = 1$
Τριγωνομετρική μεταβολή	$f(t) = \cos(t) + 2\sin(t)$ ή $F(s) = (s+2)/(s^2+1)$
Βηματικές/Τμηματικές μεταβολές	$f(t) = 2H(t) - H(t-4) + 2H(t-6)$ ή $F(s) = (2/s) - (1/s)e^{-4s} + (2/s)e^{-6s}$

Επίλυση: Μετά την εκτέλεση του κώδικα που δόθηκε πριν, τα διαγράμματα θα πρέπει να έχουν τη μορφή:

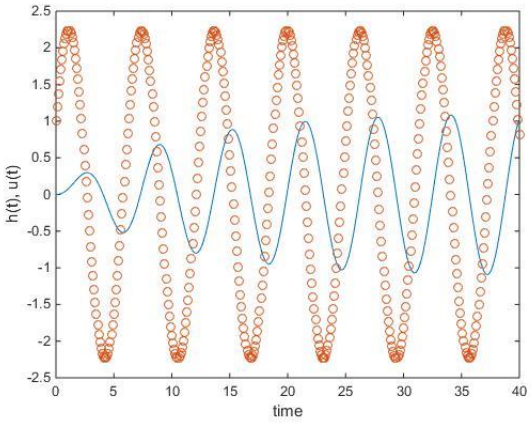




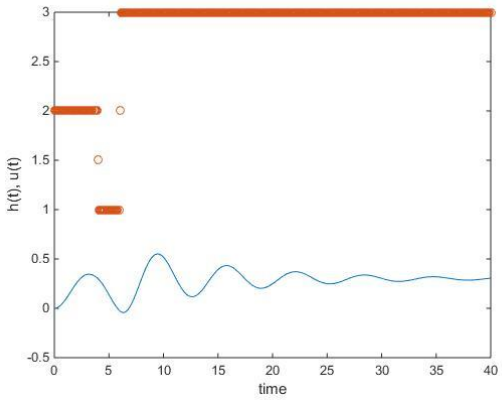
$F = F1$



$F = F2$



$F = F3$



$F = F4$

Όπου $y(t)$

$u(t)$

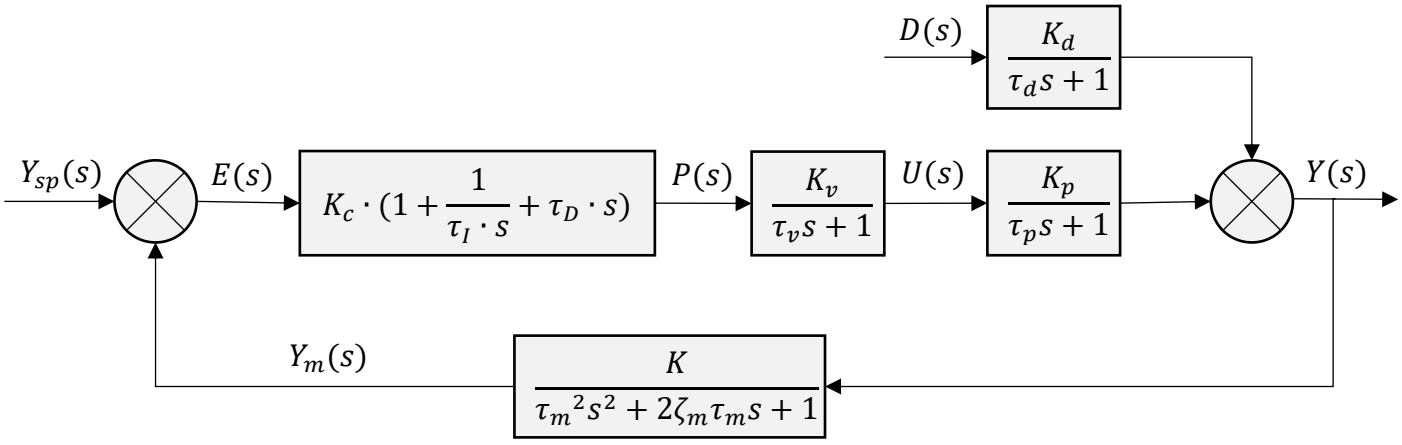
$F = F5$

3.3 Πλήρες Διάγραμμα Βαθμίδων Κλειστού Βρόχου

Παράδειγμα 3.3.1:

Εκφώνηση: Σας δίνεται το παρακάτω πλήρες (δομικό) διάγραμμα βαθμίδων που περιλαμβάνει διεργασίες (G_p), ελεγκτή (G_c), μετρητικό στοιχείο (G_m), τελικό στοιχείο ενεργοποίησης (G_v) και αρνητική ανατροφοδότηση. ***Όλες οι σταθερές K, τ είναι ίσες με 1 εκτός από $\tau_D = 5$ (για τον ελεγκτή)***

1) Να προσδιορίσετε την απόκριση του σήματος εξόδου $Y(t)$, του σήματος $U(t)$ και το σφάλμα $E(t)$ εάν: $Y_{sp} = 3 \cdot H(t) - 2 \cdot H(t-100) + H(t-300)$ και $D = 2 \cdot H(t-180) + H(t-350)$
2) Θεωρήστε τώρα ότι $\tau_I = 10000, \tau_D = 0$ (όλα τα άλλα μένουν όμοια). Τι παρατηρείτε για το σφάλμα;



Επίλυση: Ξεκινάμε με την εντολή *syms*, για να δηλώσουμε ότι θα εργαστούμε με συμβολικές μεταβλητές.

```
clear all
clc

syms s t
```

Στη συνέχεια, ορίζουμε τις συναρτήσεις μεταφοράς, που βρίσκονται στο διάγραμμα της εκφώνησης, ενώ και τις τιμές των παραμέτρων για:

- G_c Ελεγκτή
- G_v Τελικού Στοιχείου Ενεργοποίησης
- G_p Διεργασίας
- G_d Διαταραχής
- G_m Μετρητικού Στοιχείου

```
Kc=1;
tI=1;
```

```
tD=1;

Gc=Kc*(1+(1/(tI*s))+tD*s);

Kv=1;
tv=1;
Gv=Kv*(1/(tv*s+1));

Kp=1;
tp=1;
Gp=Kp*(1/(tv*s+1));

Kd=1;
td=1;
Gd=Kd*(1/(td*s+1));

Km=1;
tm=1;
zm=1;
Gm=Km*(1/((tm*s)^2+2*zm*tm*s+1));
```

Έπειτα, ορίζουμε τις σχέσεις για το σήμα αναφοράς Y_{sp} και για την διαταραχή D, οι οποίες δίνονται στην εκφώνηση της άσκησης.

```
%Ysp=3/s;
Ysp=3*heaviside(t)-(2)*heaviside(t-100)+1*heaviside(t-300);
Ysp=laplace(Ysp);
D=0*heaviside(t)+(2)*heaviside(t-180)+1*heaviside(t-350);
D=laplace(D);
```

Συνεχίζουμε ορίζοντας τις σχέσεις για το σήμα εξόδου Y, για το σήμα εισόδου στη διεργασία U και το σφάλμα E. Οι σχέσεις δίνονται:

$$Y = \frac{G_p \cdot G_v \cdot G_c}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} Y_{sp}(s) + \frac{G_d}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} D(s)$$

$$U = \frac{G_c \cdot G_v}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} Y_{sp}(s) - \frac{G_v \cdot G_c \cdot G_m \cdot G_d}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} D(s)$$

$$E = Y_{sp}(s) - Y(s)$$

```
Y=(Gp*Gv*Gc/(1+Gp*Gv*Gc*Gm))*Ysp+(Gp/(1+Gp*Gv*Gc*Gm))*D;
Error=Ysp-Y;
y=ilaplace(Y);
e=ilaplace(Error);
U=(Gc*Gv/(1+Gp*Gv*Gc*Gm))*Ysp-(Gv*Gc*Gm*Gd/(1+Gp*Gv*Gc*Gm))*D;
u=ilaplace(U);
```

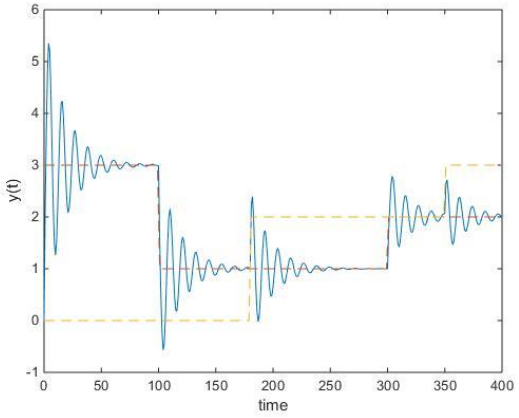
Απαραίτητες εντολές έτσι ώστε να βγούμε από τις συμβολικές μεταβλητές και να περάσουμε σε πραγματικά νούμερα είναι:

```
Total_Time=400;
t=0:1:Total_Time;

y=subs(y,t);
y=double(y);

u=subs(u,t);
u=double(u);

ysp=ilaplace(Ysp);
ysp=subs(ysp,t);
ysp=double(ysp);
d=ilaplace(D);
d=subs(d,t);
d=double(d);

e=subs(e,t);
e=double(e);
```

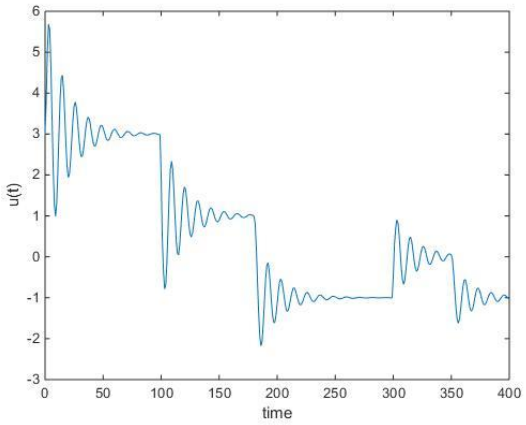
Τέλος, για να δούμε τα αποτελέσματα υπό μορφή διαγραμμάτων πρέπει να ακολουθήσουν οι εξής εντολές:

```
figure(1);plot(t,y,t,ysp,'--',t,d,'--');xlabel('time');ylabel('y(t)')
figure(2);plot(t,u,'-');xlabel('time');ylabel('u(t)')
figure(3);plot(t,e,'-');xlabel('time');ylabel('Error(t)')
```

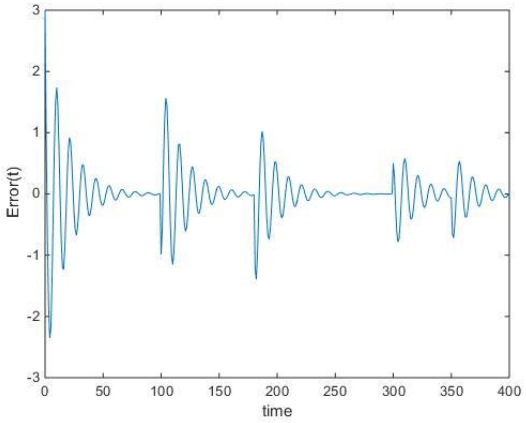
Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:



Σήμα εξόδου Y , μαζί με το σήμα αναφοράς Y_{sp} και την διαταραχή D



Σήμα εισόδου διεργασίας U



Σήμα σφάλματος E

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 3.3.1)

ΚΕΦΑΛΑΙΟ 4: Συστήματα 1ης & 2ης Τάξης

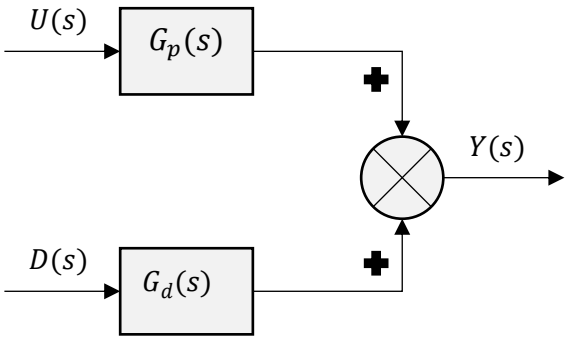
Στο τέταρτο κεφάλαιο, θα απασχοληθούμε με την επίλυση συστημάτων 1ης και 2ης τάξης παρουσία ελεγκτή αλλά και απουσία αυτού. Τα παραδείγματα που θα δούμε θα είναι με:

- *Απουσία ελεγκτή (ανοικτός βρόχος)*
- *Παρουσία αναλογικού ελεγκτή (P controller)*
- *Παρουσία αναλογικού-ολοκληρωτικού ελεγκτή (PI controller)*
- *Παρουσία αναλογικού-ολοκληρωτικού-διαφορικού ελεγκτή (PID controller)*

4.1 Συστήματα 1ης τάξης παρουσία ανοικτού και κλειστού βρόχου

Παράδειγμα 4.1.1: Δυναμικό Σύστημα 1ης Τάξης (με διαταραχή)

Εκφώνηση: Σας δίνεται το παρακάτω διάγραμμα βαθμίδων. Να προσδιορίσετε την απόκριση του σήματος εξόδου $Y(t)$, του σήματος $U(t)$ και τη διαταραχή $\mathrm{D}(t)$. ***Οι σταθερές K, τ_d είναι ίσες με 1 εκτός από $\tau_p = 10$. Οι $U(s), D(s)$ βηματικές μεταβολές (π.χ. $10/s$).***



$$\mathrm{Y(s)} = \frac{K}{\tau_p s + 1} \cdot U(s) + \frac{K_d}{\tau_d s + 1} \cdot D(s)$$

Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές.

```
%Lesson 4

clc
clear all

syms s t
```

Στη συνέχεια δηλώνουμε τις σταθερές μας καθώς και τις συναρτήσεις μεταφοράς G_p , G_d έως πρώτης τάξης, όπου σύμφωνα με την εξίσωση της εκφώνηση θα είναι:

$$G_p = \frac{K}{\tau s + 1} \qquad G_d = \frac{K_d}{\tau_d s + 1}$$

```
%Parameters
K=1;
tau=10;
Kd=1;
taud=1;

%1is taxis synartiseis metaforas
Gp=K/(tau*s+1);
Gd=Kd/(taud*s+1);
```

Έπειτα, δηλώνουμε την είσοδο και τη διαταραχή ως βηματικές μεταβολές σταθερού μεγέθους (έστω 10). Ανάλογα μπορούμε να ορίσουμε και άλλου είδους διεγέρσεις.

```
%Eisodoi / Diegerseis
%Step Change
U=(10)*heaviside(t);
U=laplace(U);

%Diataraxes
%Step Change
D=10*heaviside(t);
D= laplace(D);
```

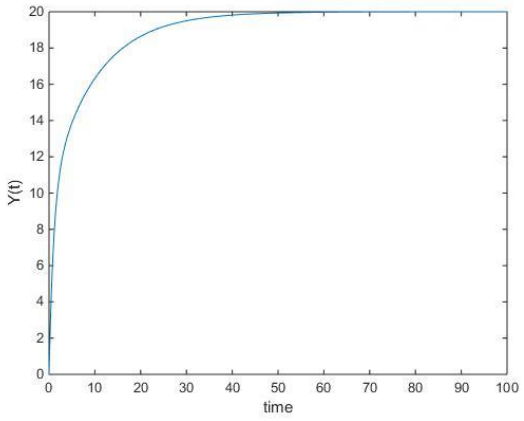
Στη συνέχεια, γράφουμε τις εντολές για να προσδιορίσουμε τα σήματα Y (σήμα εξόδου), U (σήμα εισόδου διεργασίας) και D (διαταραχή) καθώς και την σχέση που τα συνδέει, η οποία δίνεται από την εκφώνηση.

```
Y=Gp*U+Gd*D;
Y=ilaplace(Y);

t=0:0.1:100;

Y=subs(Y,t);
Y=double(Y);

u=ilaplace(U);
u=subs(u,t);
u=double(u);

d=ilaplace(D);
d=subs(d,t);
d=double(d);
```

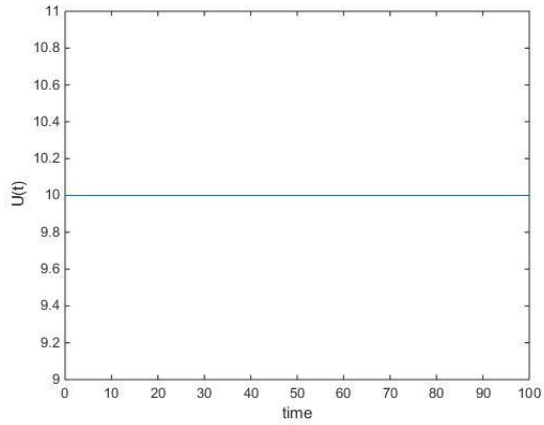
Τέλος, ακολουθούν οι εξής εντολές, έτσι ώστε να δούμε τα αποτελέσματα υπό μορφή διαγραμμάτων.

```
figure(1);plot(t,Y);xlabel('time');ylabel('Y(t)')
figure(2);plot(t,u);xlabel('time');ylabel('U(t)')
figure(3);plot(t,d);xlabel('time');ylabel('D(t)')
```

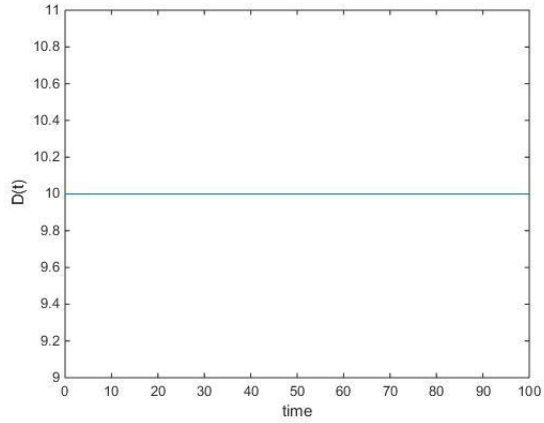
Μετά την εκτέλεση του κώδικα θα έχουμε τα παρακάτω διαγράμματα:



Σήμα εξόδου Y



Σήμα εισόδου διεργασίας U

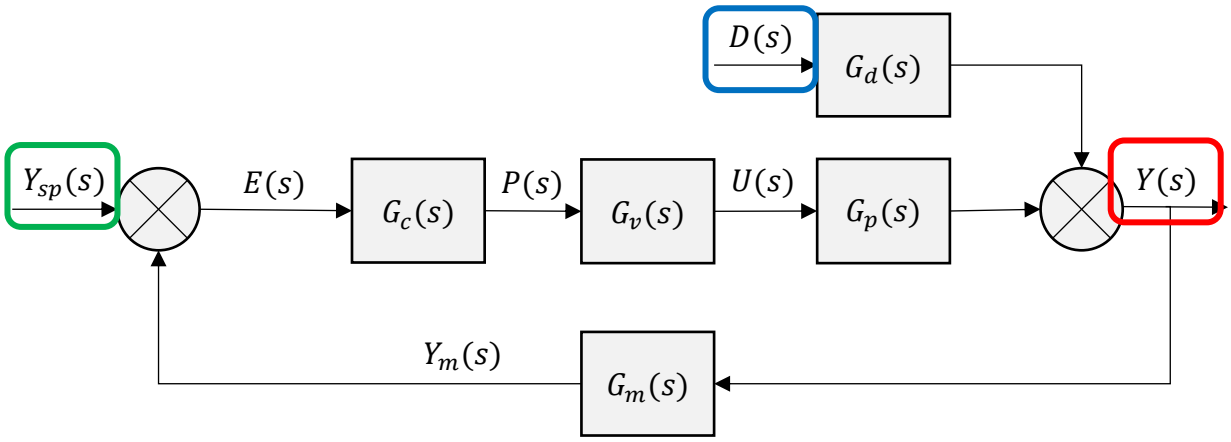


Σήμα διαταραχής D

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 4.1.1)

Από Θεωρία:

Έστω ένα πλήρες (δομικό) διάγραμμα, όπως το παρακάτω.



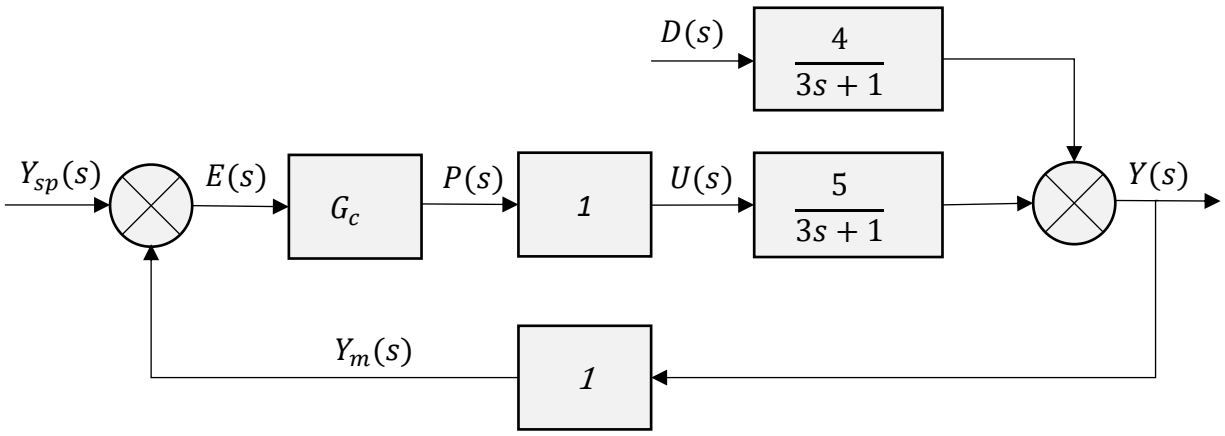
Τότε, η απόκριση κλειστού βρόχου σε ταυτόχρονες μεταβολές του σήματος αναφοράς, Y_{sp}, και της διαταραχής, D, δίνεται από τον παρακάτω τύπο για το δομικό διάγραμμα.

$$Y(s) = \frac{G_p(s) \cdot G_v(s) \cdot G_c(s)}{1 + G_p(s) \cdot G_v(s) \cdot G_c(s) \cdot G_m(s)} Y_{sp}(s) + \frac{G_d(s)}{1 + G_p(s) \cdot G_v(s) \cdot G_c(s) \cdot G_m(s)} D(s)$$

(Οι αντιστοιχίες ανάμεσα στο σχήμα και στον τύπο δίνονται χρωματικά με τα κουτιά)

Παράδειγμα 4.1.2: Ελεγκτής P, PI, PID [Συνάρτηση Μεταφοράς $G_c(s)$]

Εκφώνηση: Σας δίνεται το παρακάτω πλήρες (δομικό) διάγραμμα βαθμίδων. Να προσδιορίσετε την απόκριση του σήματος εξόδου $Y(t)$, του σήματος $U(t)$ και τη διαταραχή $D(t)$. **Για τις *σταθερές έχουμε* $K_c = 10, \tau_I = 1, \tau_D = 4, \tau_F = 0.2 \cdot \tau_D$**



Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές.

```
clc
clear all

syms s t
```

Στη συνέχεια δηλώνουμε τις σταθερές μας καθώς και τις συναρτήσεις μεταφοράς των ελεγκτών P, PI, PID (G_c).

ΠΡΟΣΟΧΗ : Για τον ελεγκτή PID της συνάρτησης μεταφοράς $G_c(s)$ θα ισχύει:

$$P(t) = p(t) - p_s = K_c \cdot E(t) + \frac{K_c}{\tau_I} \cdot \int E(t) + (K_c \cdot \tau_D) \cdot \frac{dE(t)}{dt}$$

$$\rightarrow \cdots Laplace \ldots με\ παρουσία\ φίλτρου\ (εάν\ απαιτείται) \rightarrow$$

$$\boldsymbol{P(s) = K_c \cdot \left(1 + \frac{1}{\tau_I \cdot s} + \frac{\tau_P \cdot s}{\tau_F \cdot s + 1}\right) \cdot E(s)}$$

Τέλος, επιλέγουμε ποια περίπτωση ελεγκτή θέλουμε να δουλέψουμε κάθε φορά. Στο συγκεκριμένο παράδειγμα, έχουμε επιλέξει τον ελεγκτή PID (*Gc=PID;*).

```
%Synartisi Metaforas Elegkti
Kc=10;
tI=1;
tD=4;
tF=0.2*tD;

%P controller
P=Kc;
%PI controller
PI=Kc*(1+(1/(tI*s)));
%PID controller
PID=Kc*(1+(1/(tI*s))+tD*s/(1+tF*s));

Gc=PID;
```

Έπειτα, δηλώνουμε τις συναρτήσεις μεταφοράς G_v, G_m, G_p, G_d όπως δίνονται στο διάγραμμα της εκφώνησης.

```
%Synartisi Metaforas Telikou Stoixeiou
Gv=1;
%Synartisi Metaforas Metritikou Stoixeiou
Gm=1;

%Parameters
Kp=5;
taup=3;
Kd=4;
taud=3;

%1is taxis synartiseis metaforas
Gp=Kp/(taup*s+1);
Gd=Kd/(taud*s+1);
```

Στη συνέχεια, δηλώνουμε το Σήμα Αναφοράς [$Y_{sp}(s)$] και το Σήμα Διαταραχής [$D(s)$]. Μπορούμε να επιλέξουμε διαφορετικές μεταβολές.

```
%Set Point kai Diataraxi
Ysp=1/s;
D=1/s;
```

Συνεχίζουμε ορίζοντας τις σχέσεις κλειστού βρόχου για το σήμα εξόδου Y, για το σφάλμα E και για το σήμα εισόδου στη διεργασία U που είναι αυτό που μεταβάλλει ο ελεγκτής μας. Οι σχέσεις δίνονται:

$$Y = \frac{G_p \cdot G_v \cdot G_c}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} Y_{sp}(s) + \frac{G_d}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} D(s)$$

$$U = \frac{G_c \cdot G_v}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} Y_{sp}(s) - \frac{G_v \cdot G_c \cdot G_m \cdot G_d}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} D(s)$$

$$E = Y_{sp}(s) - Y(s)$$

```
%Sxesi Kleistou Broxou
Y=(Gp*Gv*Gc/(1+Gp*Gv*Gc*Gm))*Ysp+(Gd/(1+Gp*Gv*Gc*Gm))*D;
Error=Ysp-Y;
U=(Gc*Gv/(1+Gp*Gv*Gc*Gm))*Ysp-(Gv*Gc*Gm*Gd/(1+Gp*Gv*Gc*Gm))*D;
```

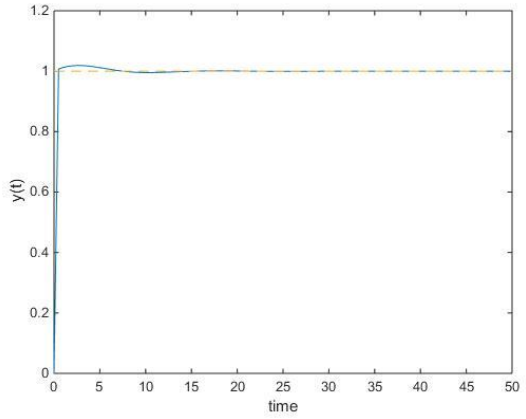
Ακολουθούν οι γνωστές εντολές για να προσδιορίσουμε τα σήματα Y (σήμα εξόδου), U (σήμα εισόδου διεργασίας), D (διαταραχή), E (σφάλμα) και Y_{sp} (σήμα εισόδου).

```
Total_Time=50;
t=0:0.5:Total_Time;

y=ilaplace(Y);
y=subs(y,t);
y=double(y);

u=ilaplace(U);
u=subs(u,t);
u=double(u);

Ysp=ilaplace(Ysp);
ysp=subs(Ysp,t);
ysp=double(ysp);

D=ilaplace(D);
d=subs(D,t);
d=double(d);

e=ilaplace(Error);
e=subs(e,t);
e=double(e);
```

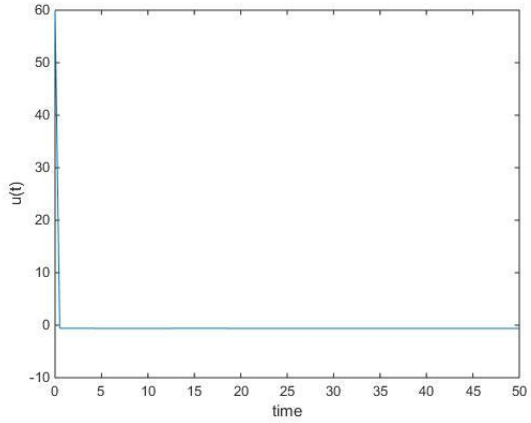
Τα σήματα αυτά τα καταγράφουμε υπό μορφή τριών διαγραμμάτων.

```
figure(1);plot(t,y,t,ysp,'--', t, d,'--');xlabel('time');ylabel('y(t)')
figure(2);plot(t,u,'-'); xlabel('time');ylabel('u(t)')
figure(3);plot(t,e,'-'); xlabel('time');ylabel('Error(t)')
```

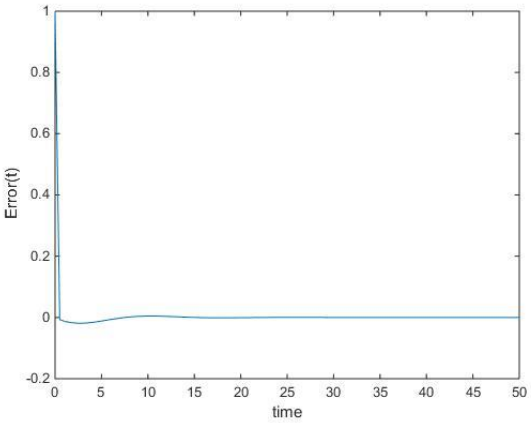
Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:



Σήμα εξόδου Y



Σήμα εισόδου διεργασίας U



Σήμα σφάλματος E

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 4.1.2)

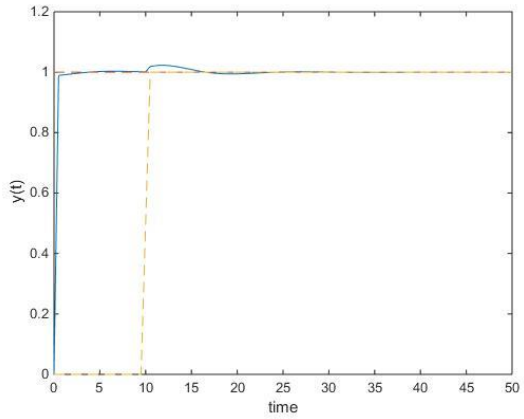
Εξάσκηση

1) *Μελετήστε την επίδραση άλλων ελεγκτών (π.χ. P, PI).*
2) *Μελετήστε την επίδραση των όρων K_c, τ_I, τ_D σε διάφορους συνδυασμούς.*
3) *Μελετήστε την επίδραση διαφορετικών σημάτων αναφοράς.*
4) *Μελετήστε την επίδραση διαφορετικών διαταραχών.*
5) *Μελετήστε την επίδραση των όρων $K_{p,}, \tau_p, K_d, \tau_d$ κτλ.*

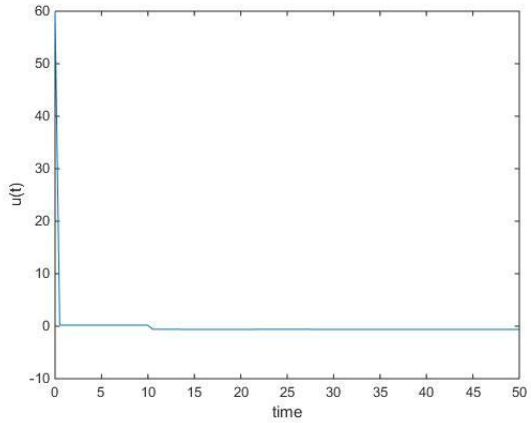
Εάν αλλάξουμε τη διαταραχή από

`D=(1/s);` σε `D=(1/s)*exp(-10*s);`

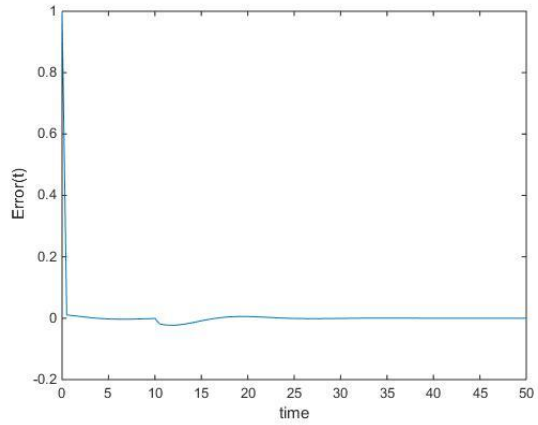
δηλαδή για $t = 10s$, τότε τα διαγράμματα θα γίνουν:



Σήμα εξόδου Y



Σήμα εισόδου διεργασίας U



Σήμα σφάλματος E

4.2 Συστήματα 2ης τάξης παρουσία ανοικτού και κλειστού βρόχου

Παράδειγμα 4.2.1: Δυναμικό Σύστημα 2ης Τάξης

Εκφώνηση: Σας δίνεται το παρακάτω διάγραμμα βαθμίδων. Να προσδιορίσετε την απόκριση του σήματος εξόδου $Y(t)$ και του σήματος εισόδου $U(t)$. ***Οι σταθερές K, τ είναι ίσες με 1 ενώ η μεταβλητή ζ είναι ίση με 0.5.***

$$\mathrm{Y(s)} = \frac{K}{\tau^2 s^2 + 2\zeta\tau s + 1} \cdot U(s)$$

Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές. Επίσης δηλώνουμε τις σταθερές μας καθώς και τις συναρτήσεις μεταφοράς G_p ως 2ης τάξης, με την εξίσωση να είναι της μορφής:

$$G_p = \frac{K}{\tau^2 s^2 + 2\zeta\tau s + 1}$$

Μπορούμε, αν θέλουμε, να ορίσουμε και διαταραχή.

```
clc
clear all

syms s t

%Parameters
```

```
K=1;
tau=1;
z=0.5;

%1is taxis synartiseis metaforas
Gp=K/((tau^2)*(s^2)+2*z*tau*s+1);
```

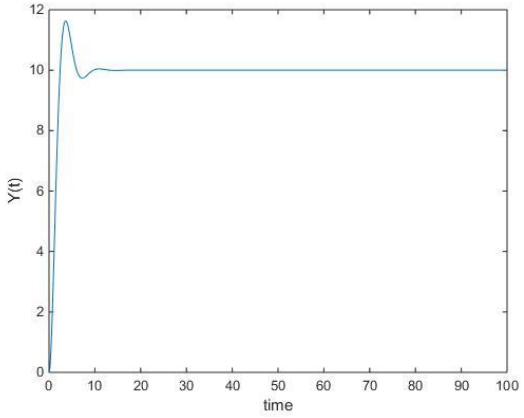
Στη συνέχεια, δηλώνουμε την είσοδο ως βηματική μεταβολή σταθερού μεγέθους (έστω 10). Ανάλογα μπορούμε να ορίσουμε άλλου είδους διεγέρσεις.

```
%Eisodoi / Diegerseis
%Step Change
U=(10)*heaviside(t);
U=laplace(U);
```

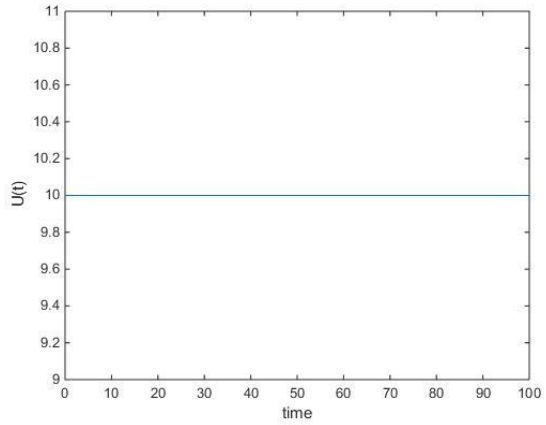
Ακολουθούν οι γνωστές εντολές για να προσδιορίσουμε τα σήματα Y (σήμα εξόδου) και U (σήμα εισόδου διεργασίας). Τα σήματα αυτά τα καταγράφουμε υπό μορφή τριών διαγραμμάτων.

```
Y=Gp*U;
Y=ilaplace(Y);

t=0:0.1:100;

Y=subs(Y,t);
Y=double(Y);

u=ilaplace(U);
u=subs(u,t);
u=double(u);


figure(1);plot(t,Y);xlabel('time');ylabel('Y(t)')
figure(2);plot(t,u);xlabel('time');ylabel('U(t)')
```

Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:

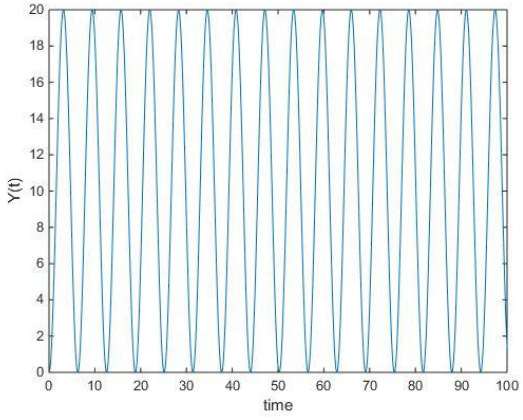


Σήμα εξόδου Y

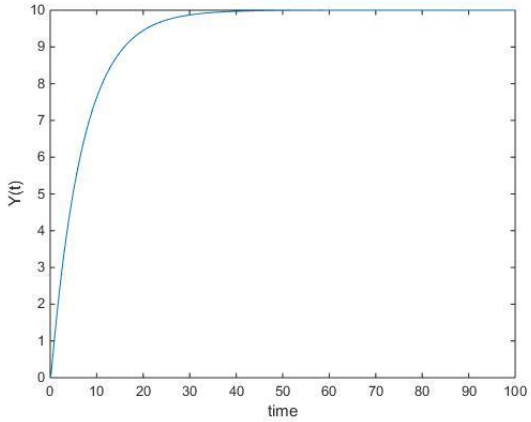


Σήμα εισόδου διεργασίας U

Για διαφορετική μεταβλητή ζ, τα διαγράμματα του Σήματος Εισόδου $Y(t)$ θα έχουν τη μορφή:



$\zeta = 0$



$\zeta = 3.5$

Ακόμη ισχύουν οι τύποι:

Χρόνος Ανύψωσης (rise time)	$t_r = (0.8 + 2.5\zeta) \cdot \tau$
Χρόνος Μέγιστης (ποσοστιαίας) υπερύψωσης ή υπέρβασης (t_p)	$t_p = \pi \frac{\tau}{\sqrt{1-\zeta^2}}$
Χρόνος Αποκατάστασης (t_s)	• $t_s = \frac{4\tau}{\zeta}$ για ζώνη ανοχής 2% • $t_s = \frac{3\tau}{\zeta}$ για ζώνη ανοχής 5%
Μέγιστη (ποσοστιαία) υπερύψωση (M_p)	$M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$

Τους προσθέτουμε στο πρόγραμμα ως εξής:

```
tr=(0.8+2.5*z)*tau
tp=pi*tau/(1-z^2)^0.5
Mp=exp(-pi*z/(1-z^2)^0.5)
ts_2=4*tau/z
ts_5=3*tau/z
```

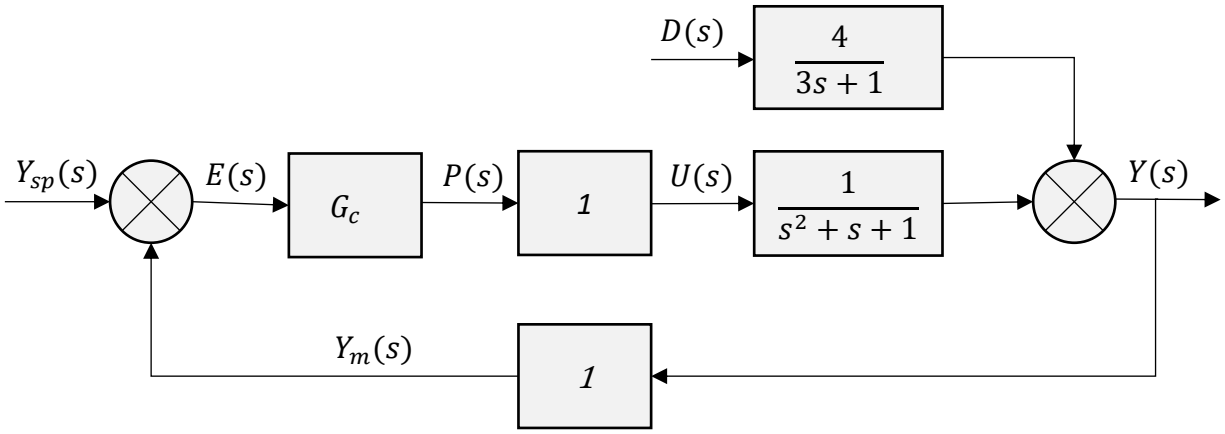
Τα αποτέλεσμα στο Matlab για $\tau = 1$ και $\zeta = 0.5$ θα είναι:

```
tr =

    2.0500


tp =

    3.6276


Mp =

    0.1630


ts_2 =

     8


ts_5 =

     6
```

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 4.2.1)

Παράδειγμα 4.2.2: Ελεγκτής P, PI, PID [Συνάρτηση Μεταφοράς $G_c(s)$]

Εκφώνηση: Σας δίνεται το παρακάτω πλήρες (δομικό) διάγραμμα βαθμίδων. Να προσδιορίσετε την απόκριση του σήματος εξόδου $Y(t)$, του σήματος $U(t)$ και τη διαταραχή $D(t)$. **Για τις *σταθερές έχουμε* $\boldsymbol{K_c = 1, \tau_I = 1, \tau_D = 4, \tau_F = 0.2 \cdot \tau_D}$**



Επίλυση: Η άσκηση έχει τον ίδιο τρόπο λύσης με το Παράδειγμα 4.1.2 της Ενότητας 4.1. Η μόνη διαφορά βρίσκεται στη δήλωση της συνάρτησης μεταφοράς G_p.

Έτσι ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές.

```
clc
clear all

syms s t
```

Στη συνέχεια δηλώνουμε τις σταθερές μας καθώς και τις συναρτήσεις μεταφοράς των ελεγκτών P, PI, PID (G_c).

<u>ΠΡΟΣΟΧΗ</u> : Για τον ελεγκτή PID της συνάρτησης μεταφοράς $G_c(s)$ θα ισχύει:

$$P(t) = p(t) - p_s = K_c \cdot E(t) + \frac{K_c}{\tau_I} \cdot \int E(t) + (K_c \cdot \tau_D) \cdot \frac{dE(t)}{dt}$$

$$\rightarrow \cdots Laplace \ldots με\ παρουσία\ φίλτρου\ (εάν\ απαιτείται) \rightarrow$$

$$\boldsymbol{P(s) = K_c \cdot \left(1 + \frac{1}{\tau_I \cdot s} + \frac{\tau_P \cdot s}{\tau_F \cdot s + 1}\right) \cdot E(s)}$$

Τέλος, επιλέγουμε ποια περίπτωση ελεγκτή θέλουμε να δουλέψουμε κάθε φορά. Σε συγκεκριμένο παράδειγμα, έχουμε επιλέξει τον ελεγκτή PID (*Gc=PID;*).

```
%Synartisi Metaforas Elegkti
Kc=1;
tI=1;
tD=4;
tF=0.2*tD;

%P controller
P=Kc;
%PI controller
PI=Kc*(1+(1/(tI*s)));
%PID controller
PID=Kc*(1+(1/(tI*s))+tD*s/(1+tF*s));

Gc=PID;
```

Έπειτα, δηλώνουμε τις συναρτήσεις μεταφοράς G_v, G_m, G_p, G_d όπως δίνονται στο διάγραμμα της εκφώνησης.

```
%Synartisi Metaforas Telikou Stoixeiou
Gv=1;
%Synartisi Metaforas Metritikou Stoixeiou
Gm=1;

%Parameters
%Parameters
Kp=1;
taup=1;
zp=0.0;

Kd=4;
taud=3;

%1is taxis synartiseis metaforas
Gp=Kp/((taup^2)*(s^2)+2*zp*taup*s+1);
Gd=Kd/(taud*s+1);
```

Στη συνέχεια, δηλώνουμε το Σήμα Αναφοράς [$Y_{sp}(s)$] και το Σήμα Διαταραχής [$D(s)$].

```
%Set Point kai Diataraxi
Ysp=(1/s)*(1+3*exp(-15*s));
D=(1/s)*(1+8*exp(-45*s));
```

Συνεχίζουμε ορίζοντας τις σχέσεις κλειστού βρόχου για το σήμα εξόδου Y, για το σφάλμα E και για το σήμα εισόδου στη διεργασία U που είναι αυτό που μεταβάλλει ο ελεγκτής μας. Οι σχέσεις δίνονται:

$$Y = \frac{G_p \cdot G_v \cdot G_c}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} Y_{sp}(s) + \frac{G_d}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} D(s)$$

$$U = \frac{G_c \cdot G_v}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} Y_{sp}(s) - \frac{G_v \cdot G_c \cdot G_m \cdot G_d}{1 + G_p \cdot G_v \cdot G_c \cdot G_m} D(s)$$

$$E = Y_{sp}(s) - Y(s)$$

```
%Sxesi Kleistou Broxou
Y=(Gp*Gv*Gc/(1+Gp*Gv*Gc*Gm))*Ysp+(Gd/(1+Gp*Gv*Gc*Gm))*D;
Error=Ysp-Y;
U=(Gc*Gv/(1+Gp*Gv*Gc*Gm))*Ysp-(Gv*Gc*Gm*Gd/(1+Gp*Gv*Gc*Gm))*D;
```

Ακολουθούν οι γνωστές εντολές για να προσδιορίσουμε τα σήματα Y (σήμα εξόδου), U (σήμα εισόδου διεργασίας), D (διαταραχή), E (σφάλμα) και Y_{sp} (σήμα εισόδου).

<u>ΣΗΜΕΙΩΣΗ</u> : Σε ορισμένες εκδόσεις του Matlab η εντολή *double* δημιουργεί καθυστερήσεις στην εκτέλεση του κώδικα. Για την επίλυση του προβλήματος αυτού μπορούμε είτε να παραλείψουμε την εντολή *double*, είτε να κάνουμε χρήση της αλληλουχίας με την εντολή *vpa* ως εξής:

```
y=ilaplace(Y);
  y=subs(y,t);
  y=double(y);
```

⟶

```
y=ilaplace(Y);
y=vpa(y);
y=subs(y,t);
y=double(y);
```

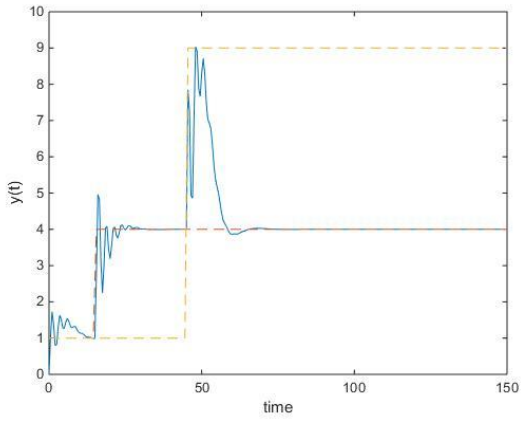
Αντίστοιχα και για τα Y_{sp}, U, D, E.

```
Total_Time=150;
t=0:0.5:Total_Time;

y=ilaplace(Y);
y=subs(y,t);
y=double(y);

u=ilaplace(U);
u=subs(u,t);
u=double(u);

Ysp=ilaplace(Ysp);
ysp=subs(Ysp,t);
ysp=double(ysp);

D=ilaplace(D);
d=subs(D,t);
d=double(d);

e=ilaplace(Error);
e=subs(e,t);
e=double(e);
```

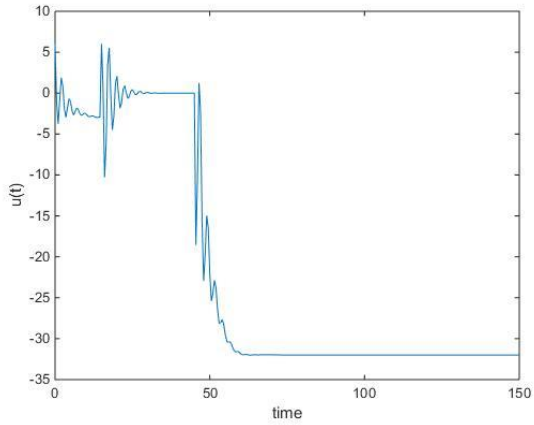
Τα σήματα αυτά τα καταγράφουμε υπό μορφή τριών διαγραμμάτων.

```
figure(1);plot(t,y,t,ysp,'--', t, d,'--');xlabel('time');ylabel('y(t)')
figure(2);plot(t,u,'-');xlabel('time');ylabel('u(t)')
figure(3);plot(t,e,'-');xlabel('time');ylabel('Error(t)')
```

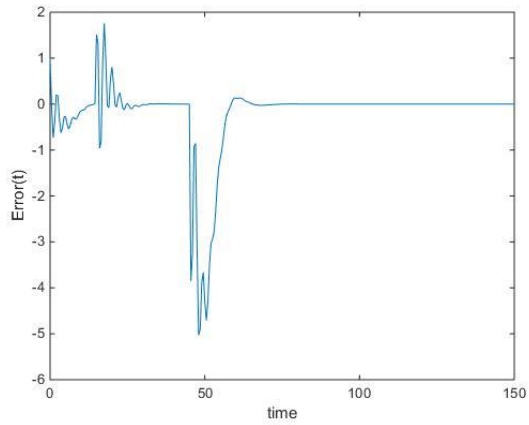
Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:



Σήμα εξόδου Y



Σήμα εισόδου διεργασίας U



Σήμα σφάλματος E

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 4.2.2)

Εξάσκηση

1) *Μελετήστε την επίδραση άλλων ελεγκτών (π.χ. P, PI).*
2) *Μελετήστε την επίδραση των όρων K_c, τ_I, τ_D σε διάφορους συνδυασμούς.*
3) *Μελετήστε την επίδραση διαφορετικών σημάτων αναφοράς.*
4) *Μελετήστε την επίδραση διαφορετικών διαταραχών.*
5) *Μελετήστε την επίδραση των όρων τ, ζ κτλ.*

ΚΕΦΑΛΑΙΟ 5: Ευστάθεια, Γεωμετρικός Τόπος Ριζών & Συντονισμός Παραμέτρων

Στο πέμπτο κεφάλαιο, θα απασχοληθούμε με θέματα ευστάθειας σε συστήματα ανοικτού και κλειστού βρόχου, ενώ παράλληλα θα μάθουμε να σχεδιάζουμε το γεωμετρικό τόπο των ριζών και να συντονίζουμε τις παραμέτρους ενός ελεγκτή PID (TUNING). Τέλος θα δούμε μερικά ειδικά θέματα Συστημάτων Ελέγχου. Τα παραδείγματα που θα δούμε αφορούν:

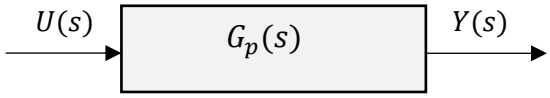
- *Ευστάθεια Συστημάτων*
- *Γεωμετρικός Τόπος Πόλων & Κρίσιμη Τιμή Ενίσχυσης*
- *Συντονισμός Παραμέτρων με τη μέθοδο Ziegler-Nichols*
- *Εύρεση Μηδενικών σε Συνάρτηση Μεταφοράς Κλειστού Βρόχου*

5.1 Ευστάθεια Συστημάτων

Παράδειγμα 5.1.1: Ευστάθεια Συστήματος Ανοικτού Βρόχου

Εκφώνηση: Να προσδιορίσετε εάν το σύστημα ανοικτού βρόχου στο παρακάτω σχήμα (χωρίς την παρουσία ελεγκτή) είναι ευσταθές. Η συνάρτηση μεταφοράς $G_p(s)$ μεταξύ εξόδου $Y(s)$ και εισόδου $U(s)$ δίνεται ως :

$$G_p(s) = \frac{1}{s \cdot (s-1) \cdot (s+1) \cdot (s+2)}$$



Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές.

```
clc
clear all

syms s
```

Στη συνέχεια, ορίζουμε την συνάρτηση μεταφοράς $G_p(s)$, έτσι ακριβώς όπως μας τη δίνει η εκφώνηση, και προσδιορίζουμε ξεχωριστά τον αριθμητή (*num*) και τον παρονομαστή (*den*) της συνάρτησης μεταφοράς $G_p(s)$ του ανοικτού βρόχου, με τη χρήση της εντολής *numden*.

```
Gp=1/(s*(s-1)*(s+1)*(s+2));
[num,den]=numden(Gp);
```

Έπειτα, μετατρέπουμε τον αριθμητή *(num)* και παρονομαστή *(den),* που βρήκαμε στο προηγούμενο βήμα, από συμβολικές μεταβλητές σε πραγματικά πολυώνυμα με χρήση της εντολής *sym2poly*. Μας ενδιαφέρει να δούμε τι συμβαίνει στο πολυώνυμο του παρονομαστή *(den)*.

```
num=sym2poly(num);
den=sym2poly(den);
```

Οι ρίζες (*roots*) του χαρακτηριστικού πολυωνύμου, δηλαδή ο παρονομαστής *den*, μας καθοδηγούν στο να αναγνωρίσουμε εάν το σύστημα είναι ευσταθές ή όχι. Το *a* στο πρόγραμμά θα καταγράψει πόσες ρίζες έχουμε (π.χ. 4).

```
ROOTS=roots(den)
[a,b]=size(ROOTS)
```

Τέλος, οι παρακάτω εντολές θα ελέγξουν ποιες από τις ρίζες (roots) του χαρακτηριστικού πολυωνύμου που βρήκαμε παραπάνω είναι:

Αρνητικές ⟶ Ευστάθεια

Θετικές ⟶ Αστάθεια

Μηδενικές ⟶ Οριακή Ευστάθεια

```
for i=1:1:a;

    if ROOTS(i) > 0
        disp('to systhma einai astathes')
    elseif ROOTS(i) < 0
        disp('to systhma einai eustathes')
    else
        disp('to systhma einai oriaka eustathes')
    end
end
```

Μετά την εκτέλεση του κώδικά τα αποτελέσματα του Matlab θα είναι:

```
ROOTS =

         0
    1.0000
   -2.0000
   -1.0000


a =

     4


b =

     1

to systhma einai oriaka eustathes
```

```
to systhma einai astathes
to systhma einai eustathes
to systhma einai eustathes
```

Όπου το a = 4 μας δείχνει πόσες ρίζες έχουμε, όπου στη περίπτωση μας είναι τέσσερις, και οι ρίζες του χαρακτηριστικού πολυωνύμου όπως και ποιες από αυτές οδηγούν σε ευστάθεια, αστάθεια και οριακή αστάθεια φαίνονται από την παρακάτω αντιστοιχία.

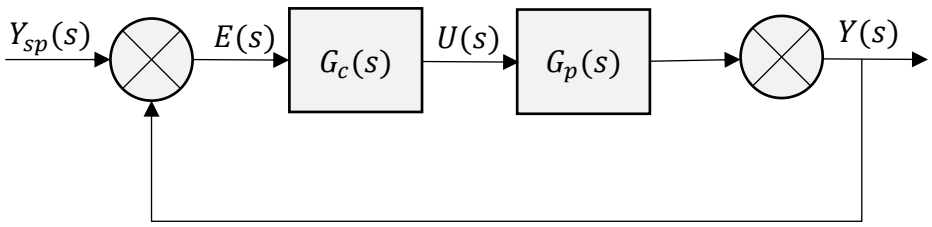
```
ROOTS =

     0
1.0000
-2.0000
-1.0000
```

```
to systhma einai oriaka eustathes
to systhma einai astathes
to systhma einai eustathes
to systhma einai eustathes
```

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 5.1.1).

Παράδειγμα 5.1.2: Ευστάθεια Συστήματος Κλειστού Βρόχου

Εκφώνηση: Εάν ο ελεγκτής είναι αναλογικός με τιμή αναλογικής ενίσχυσης $G_c = K_c = 1$, να προσδιορίσετε εάν το παρακάτω σύστημα κλειστού βρόχου είναι ευσταθές.



Η συνάρτηση μεταφοράς είναι: $$G_p(s) = \frac{1}{s \cdot (s-1) \cdot (s+1) \cdot (s+2)}$$

Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές.

```
clc
clear all

syms s
```

Στη συνέχεια, ορίζουμε την συνάρτηση μεταφοράς $G_p(s)$ και την συνάρτηση μεταφοράς του ελεγκτή $G_c(s)$, έτσι ακριβώς όπως μας δίνονται στην εκφώνηση. Ακόμη, προσδιορίζουμε ξεχωριστά τον αριθμητή (*num*) και τον παρονομαστή (*den*) του κλειστού βρόχου.

```
Gp=1/(s*(s-1)*(s+1)*(s+2));
Kc=1;
Gc=Kc;

num=Gp*Gc;
den=1+Gp*Gc;
```

Έπειτα, προσδιορίζουμε την σχέση μεταξύ της $Y(s)$ και της $Y_{sp}(s)$ που ορίζεται από την συνάρτηση *G_closed_loop*, σύμφωνα με την ακόλουθη σχέση.

$$Y(s) = G\_closed\_loop \cdot Y_{sp}(s) \quad \Longrightarrow \quad Y(s) = \frac{num}{den} \cdot Y_{sp}(s)$$

$$\Longrightarrow \quad Y(s) = \frac{G_c \cdot G_p}{1 + G_c \cdot G_p} \cdot Y_{sp}(s)$$

```
G_closed_loop=num/den;
G_closed_loop=simplifyFraction(G_closed_loop)
```

Συνεχίζουμε μετατρέποντας τον παρονομαστή *den* της *G_closed_loop*, από συμβολικές μεταβλητές σε πραγματικά πολυώνυμα, με τη χρήση της εντολής *sym2poly*. Μας ενδιαφέρει να δούμε τι συμβαίνει στο πολυώνυμο του παρονομαστή που είναι το χαρακτηριστικό πολυώνυμο, *Char_Poly*, του κλειστού πλέον βρόχου.

```
[num,den]=numden(G_closed_loop);
den=sym2poly(den);
Char_Poly=den;
```

Οι ρίζες (*roots*) του χαρακτηριστικού πολυωνύμου, δηλαδή ο παρονομαστής *den ή Char_Poly*, μας καθοδηγούν στο να αναγνωρίσουμε εάν το σύστημα στον κλειστό βρόχο, είναι ευσταθές ή όχι. Το *a* στο πρόγραμμά μας θα καταγράψει πόσες ρίζες έχουμε στον κλειστό βρόχο. Τέλος, ακολουθούν οι εντολές if, elseif, else όπως ακριβώς και στο προηγούμενο παράδειγμα.

```
ROOTS=roots(Char_Poly)
[a,b]=size(ROOTS);

for i=1:1:a;

    if ROOTS(i) > 0
        disp('to systhma einai astathes')
    elseif ROOTS(i) < 0
        disp('to systhma einai eustathes')
    else
        disp('to systhma einai oriaka eustathes')
    end
end
```

Μετά την εκτέλεση του κώδικά τα αποτελέσματα του Matlab θα είναι:

```
G_closed_loop =

1/(s^4 + 2*s^3 - s^2 - 2*s + 1)


ROOTS =

   -1.6180
   -1.6180
    0.6180
    0.6180


a =

     4


b =

     1

to systhma einai eustathes
to systhma einai eustathes
to systhma einai astathes
to systhma einai astathes
```

Το a = 4 μας δείχνει πόσες ρίζες έχουμε, όπου στη περίπτωση μας είναι τέσσερις, και οι ρίζες του χαρακτηριστικού πολυωνύμου όπως και ποιες από αυτές οδηγούν σε ευστάθεια, αστάθεια και οριακή αστάθεια φαίνονται από την παρακάτω αντιστοιχία.

```
ROOTS =

-1.6180   ⟶   to systhma einai eustathes
-1.6180   ⟶   to systhma einai eustathes
 0.6180   ⟶   to systhma einai astathes
 0.6180   ⟶   to systhma einai astathes
```

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 5.1.2).

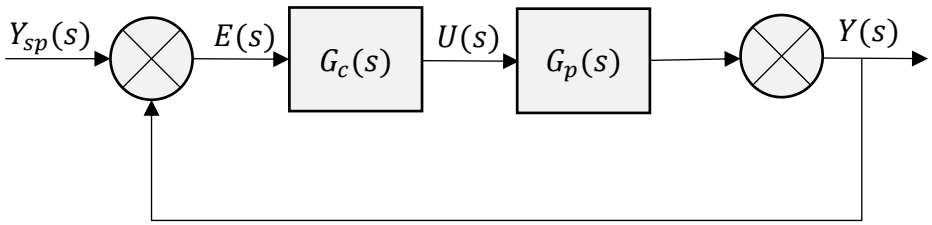
Εξάσκηση

1) Μελετήστε την επίδραση άλλων ελεγκτών (π.χ. P, PI, PID). Θα πρέπει να τροποποιήσετε την συνάρτηση μεταφοράς $G_c(s)$ στην κατάλληλη μορφή ελεγκτή P, PI, PID.
2) Μελετήστε την επίδραση άλλων $G_p(s)$ από παραδείγματα της θεωρίας.

5.2 Γεωμετρικός Τόπος Πόλων

Παράδειγμα 5.2.1: Γεωμετρικός Τόπος Πόλων / Κρίσιμη Τιμή Ενίσχυσης

Εκφώνηση: Να σχεδιάσετε τον γεωμετρικό τόπο των πόλων και να βρείτε την κρίσιμη τιμή της ενίσχυσης του αναλογικού ελεγκτή K_c, για την οποία το παρακάτω σύστημα κλειστού βρόχου εμφανίζει οριακή ευστάθεια. Ισχύει $G_c = K_c = 1$.



Η συνάρτηση μεταφοράς είναι: $$G_p(s) = \frac{1}{(s \mp 1) \cdot (s+1) \cdot (s+2)}$$

Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές. Στη συνέχεια, ορίζουμε την συνάρτηση μεταφοράς $G_p(s)$ και την συνάρτηση μεταφοράς του ελεγκτή $G_c(s)$, έτσι ακριβώς όπως μας δίνονται στην εκφώνηση.

```
clear all
clc

syms s

Gp=1/((s+1)*(s+1)*(s+2));
Kc=1;
Gc=Kc;
```

Στη συνέχεια, ορίζουμε το γινόμενο G_open_loop που απαιτείται για την εύρεση του γεωμετρικού τόπου των πόλων, σύμφωνα με την ακόλουθη σχέση

$$G_{open_{loop}} = G_p(s) \cdot G_c(s)$$

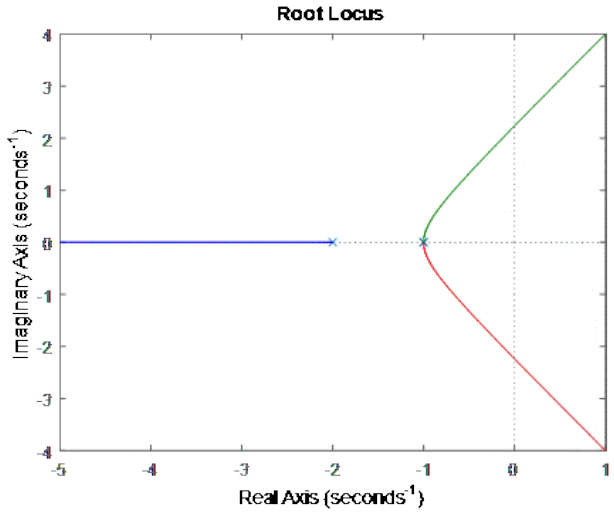
Έπειτα, προσδιορίζουμε τον αριθμητή (*num*) και τον παρονομαστή (*den*) του παραπάνω γινομένου, με τη χρήση της εντολής *numden.* Με την εντολή sym2poly, θα τους μετατρέψουμε από συμβολικές μεταβλητές σε πραγματικά πολυώνυμα.

```
G_open_loop=Gp*Gc;
G_open_loop=simplifyFraction(G_open_loop)
[num,den]=numden(G_open_loop)

num=sym2poly(num);
den=sym2poly(den);
```

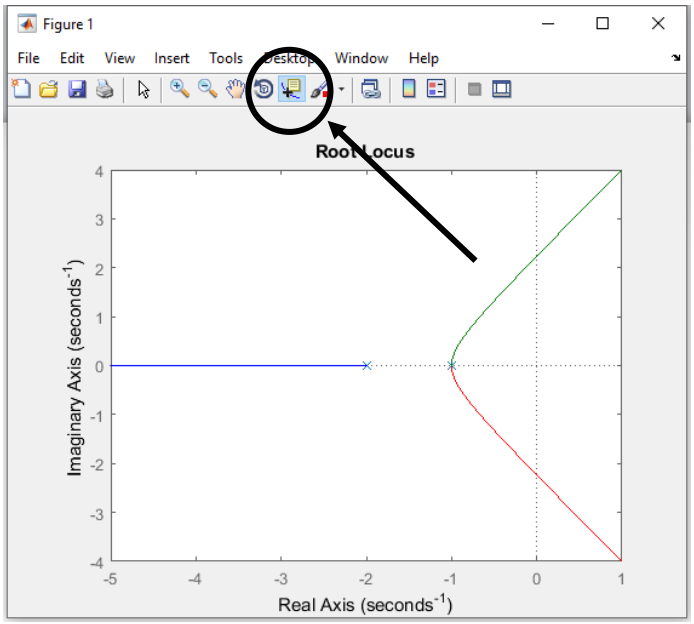
Τέλος, θα προσδιορίσουμε την συνάρτηση μεταφοράς (transfer function) με βάση τον αριθμητή και τον παρονομαστή που βρήκαμε στο προηγούμενο βήμα, με την εντολή *tf*. Η κατασκευή του γεωμετρικού τόπου των πόλων γίνεται με την εντολή *rlocus*.

```
G_root_locus=tf(num,den)

figure(1);rlocus(G_root_locus)
```

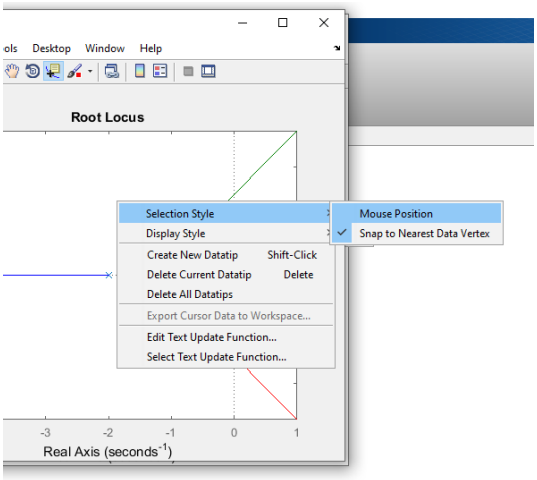
Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:



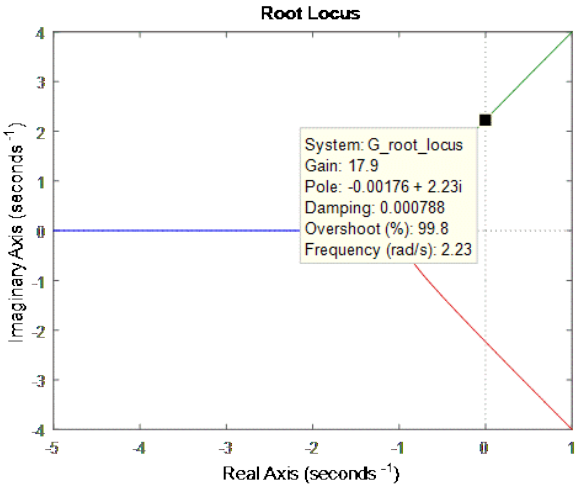
Για να πάρουμε τις απαράιτητες πληροφορίες από το δίαγραμμα, αρκεί να ακολουθήσουμε τα εξής βήματα:



1) Κάνουμε κλικ στην επιλογή που φαίνεται στη διπλανή εικόνα.



2) Κάνουμε δεξί κλίκ. Στην επιλογή *Selection Style*, επιλέγουμε το *Mouse Position*, όπως φαίνεται στη διπλανή εικόνα.



3) Κάνουμε κλικ με τον κέρσορα το σημείο οριακής ευστάθειας, το οποίο είναι το σημείο πάνω στον άξονα y που τέμνει η καμπύλη μας, όπως φαίνεται στο διπλανό σχήμα.

Η τιμή που βλέπουμε ως Gain είναι η κρίσιμη τιμή της ενίσχυσης του αναλογικού ελεγκτή, K_c, για την οποία το σύστημα κλειστού βρόχου εμφανίζει ορική ευστάθεια (πραγματικό μέρος πόλου = 0).

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 5.2.1).

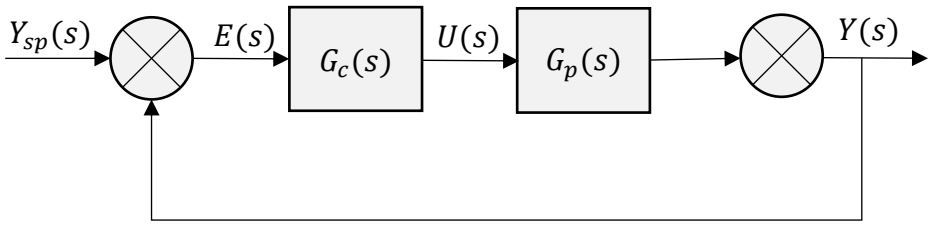
Εξάσκηση

1) Μελετήστε την επίδραση άλλων $G_p(s)$ από παραδείγματα της θεωρίας.

5.3 Συντονισμός Παραμέτρων

Παράδειγμα 5.3.1: Συντονισμός Παραμέτρων "Ziegler-Nichols"

Εκφώνηση: Με βάση την κρίσιμη τιμή της ενίσχυσης του αναλογικού ελεγκτή K_c, να βρείτε την απόκριση $y(t)$ για το σύστημα του κλειστού βρόχου εάν $Y_{sp}(s) = {}^{1}/_{s}$. Ποια είναι η κρίσιμη περίοδος σταθερής ταλάντωσης. Ισχύει $G_c = K_c = 17.9$.



Η συνάρτηση μεταφοράς είναι: $$G_p(s) = \frac{1}{(s+1)\cdot(s+1)\cdot(s+2)}$$

Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές. Στη συνέχεια, δηλώνουμε την συνάρτηση μεταφοράς της διεργασίας $G_p(s)$ και την συνάρτηση μεταφοράς του ελεγκτή $G_c(s)$ με την κρίσιμη τιμή της ενίσχυσης από τον γεωμετρικό τόπο των πόλων/ριζών και τη βηματική μοναδιαία μεταβολή της $Y_{sp}(s)$.

```
clear all
clc

syms s

Gp=1/((s+1)*(s+1)*(s+2));
Kc=17.9;
Gc=Kc;

Ysp=1/s;
```

Στη συνέχεια, βρίσκουμε τον αντίστροφο μετασχηματισμό Laplace της $Y(t)$, δηλαδή την $y(t)$. Στο συγκεκριμένο παράδειγμα δεν έχουμε διαταραχή.

```
Y=(Gp*Gc/(1+Gp*Gc))*Ysp;
y=ilaplace(Y);
```

Έπειτα, δηλώνουμε την διάρκεια του χρόνου που θέλουμε (π.χ. $10sec$) και προσδιορίζουμε αριθμητικά τις αποκρίσεις εξόδου $Y(t)$ και εισόδου $Y_{sp}(t)$.

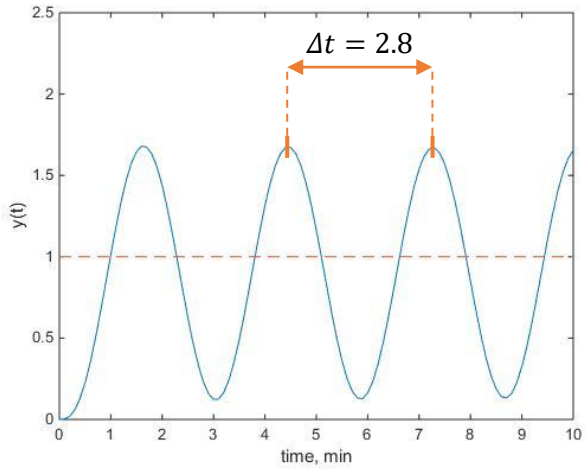
```
Total_Time=10;
t=0:0.1:Total_Time;
```

```
y=subs(y,t);
y=double(y);
ysp1=ilaplace(Ysp);
ysp1=subs(ysp1,t);
ysp1=double(ysp1);
```

Τέλος, προσδιορίζουμε τη γραφική παράσταση της απόκρισης $Y(t)$ και την απόκριση της $Y_{sp}(t)$.

```
ysp=ysp1;
figure(1);plot(t,y,t,ysp,'--');xlabel('time, min');ylabel('y(t)')
```

Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:

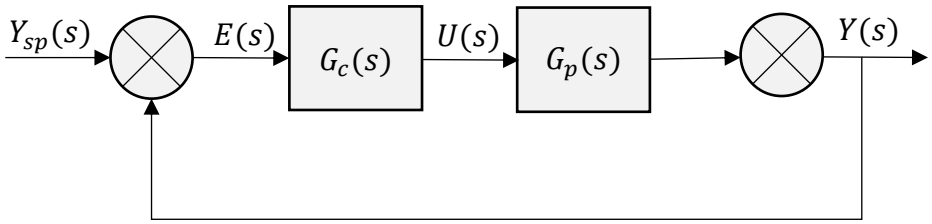


$$\boldsymbol{K_{cr} = 17.9}$$
$$\boldsymbol{P_{cr} = 2.8}$$

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 5.3.1).

Παράδειγμα 5.3.2: Συντονισμός Παραμέτρων “Ziegler-Nichols”

Εκφώνηση: Με βάση τον πίνακα Ziegler-Nichols να βρείτε την απόκριση $y(t)$ για το παρακάτω σύστημα κλειστού βρόχου με ελεγκτή P, PI, PID, εάν $Y_{sp}(s) = {}^{1}/_{s}$. Ισχύει $G_c = K_{cr} = 17.9$



Η συνάρτηση μεταφοράς είναι: $$G_p(s) = \frac{1}{(s+1)\cdot(s+1)\cdot(s+2)}$$

Πίνακας Ziegler-Nichols			
Type of Controller	$\boldsymbol{K_c}$	$\boldsymbol{\tau_I}$	$\boldsymbol{\tau_D}$
$\boldsymbol{P}$	$0.5 \cdot K_{cr}$	∞	0
$\boldsymbol{PI}$	$0.45 \cdot K_{cr}$	$\frac{1}{1.2} \cdot P_{cr}$	0
$\boldsymbol{PID}$	$0.6 \cdot K_{cr}$	$0.5 \cdot P_{cr}$	$0.125 \cdot P_{cr}$

Επίλυση: Ξεκινάμε, με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές. Με την εντολή *syms*, με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Επομένως δεν χρειάζεται να δώσουμε τιμές σε αυτές τις μεταβλητές. Στη συνέχεια, δηλώνουμε την συνάρτηση μεταφοράς της διεργασίας $G_p(s)$ καθώς και τη βηματική μοναδιαία μεταβολή της $Y_{sp}(s)$.

```
clear all
clc

syms s

Gp=1/((s+1)*(s+1)*(s+2));
Ysp=1/s;
```

Στη συνέχεια, δηλώνουμε τις διαφορετικές μορφές ελεγκτών (P, PI, PID), αφού προσδιοριστούν οι παράμετροι με βάση την τεχνική Ziegler-Nichols, όπως φαίνεται παρακάτω.

Πίνακας Ziegler-Nichols			
Type of Controller	K_c	τ_I	τ_D
P	$0.5 \cdot 17.9$	∞	0
PI	$0.45 \cdot 17.9$	$\frac{1}{1.2} \cdot 2.8$	0
PID	$0.6 \cdot 17.9$	$0.5 \cdot 2.8$	$0.125 \cdot 2.8$

```
%P Ziegler Nichols
Kc=0.5*17.9;
P=Kc;

%PI Ziegler Nichols
Kc=0.45*17.9; tI=(1/1.2)*2.8;
PI=Kc*(1+(1/(tI*s)));

%PID Ziegler Nichols
Kc=0.6*17.9; tI=(0.5)*2.8; tD=0.125*2.8;
PID=Kc*(1+(1/(tI*s))+tD*s);
```

Συνεχίζουμε, επιλέγοντας ποια περίπτωση ελεγκτή θέλουμε να δουλέψουμε κάθε φορά. Σε συγκεκριμένο παράδειγμα, έχουμε επιλέξει τον ελεγκτή P (*Gc=P;*). Έπειτα, όπως και στο προηγούμενο παράδειγμα, βρίσκουμε τον αντίστροφο μετασχηματισμό Laplace της $Y(t)$, δηλαδή την $y(t)$ και δηλώνουμε την διάρκεια του χρόνου που θέλουμε (π.χ. $30sec$) ενώ προσδιορίζουμε αριθμητικά και τις αποκρίσεις εξόδου $Y(t)$ και εισόδου $Y_{sp}(t)$.

<u>ΣΗΜΕΙΩΣΗ</u> : Σε ορισμένες εκδόσεις του Matlab η εντολή *double* δημιουργεί καθυστερήσεις στην εκτέλεση του κώδικα. Για την επίλυση του προβλήματος αυτού μπορούμε είτε να παραλείψουμε την εντολή *double*, είτε να κάνουμε χρήση της αλληλουχίας με την εντολή *vpa* ως εξής:

```
y=subs(y,t);
y=double(y);
```

⟶

```
y=vpa(y);
y=subs(y,t);
y=double(y);
```

Αντίστοιχα και για τα Y_{sp}.

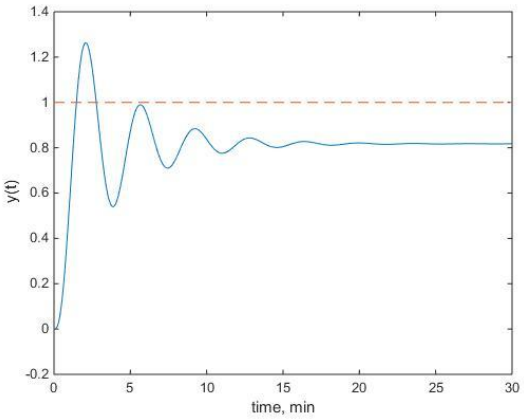
```
Gc=P;

Y=(Gp*Gc/(1+Gp*Gc))*Ysp;
y=ilaplace(Y);

Total_Time=30;
t=0:0.1:Total_Time;

y=subs(y,t);
y=double(y);
ysp1=ilaplace(Ysp);
ysp1=subs(ysp1,t);
```

```
ysp1=double(ysp1);
```

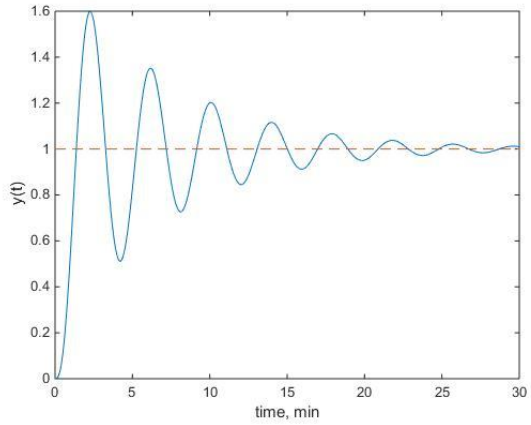
Τέλος, προσδιορίζουμε τη γραφική παράσταση της απόκρισης $Y(t)$ και την απόκριση της $Y_{sp}(t)$.

```
ysp=ysp1;
figure(1);plot(t,y,t,ysp,'--');xlabel('time, min');ylabel('y(t)')
```

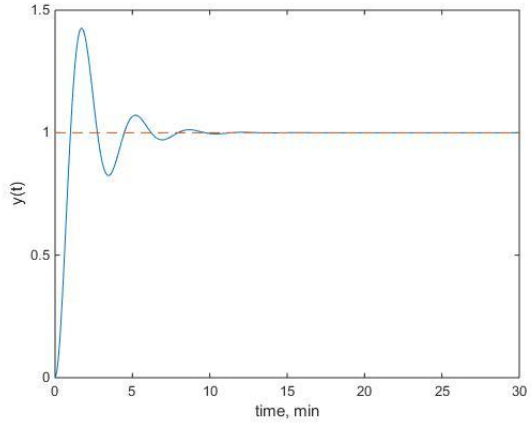
Μετά την εκτέλεση του κώδικά θα έχουμε τα παρακάτω διαγράμματα:



ελεγκτής P



ελεγκτής PI

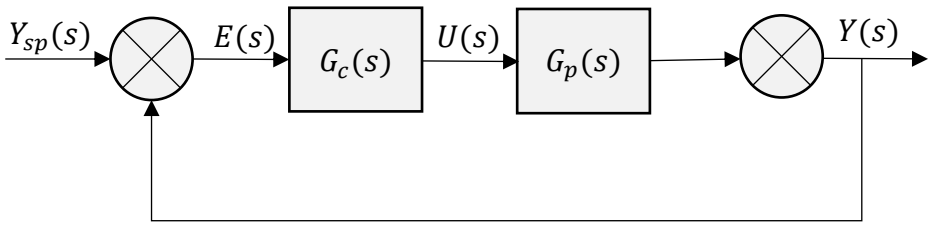


ελεγκτής PID

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 5.3.2)

Παράδειγμα 5.3.3: Εύρεση Μηδενικών σε Σύστημα Κλειστού Βρόχου

Εκφώνηση: Εάν ο ελεγκτής είναι αναλογικός-ολοκληρωτικός (PI) με παραμέτρους K_c, τ_I, να προσδιορίσετε τους πόλους και τα μηδενικά στο παρακάτω σύστημα κλειστού βρόχου.



Η συνάρτηση μεταφοράς είναι: $$G_p(s) = \frac{s+5}{s \cdot (s-1) \cdot (s+1) \cdot (s+2)}$$

Επίλυση: Ξεκινάμε όπως και στα προηγούμενα παραδείγματα με τις εντολές *clc* και *clear all* με τις οποίες καθαρίζουμε το ιστορικό από τυχόν προηγούμενες εισαγωγές καθώς και την εντολή syms με την οποία δηλώνουμε ότι θα εργαστούμε με συμβολικές μεταβλητές. Στη συνέχεια, ορίζουμε την συνάρτηση μεταφοράς $G_p(s)$, την συνάρτηση μεταφοράς του ελεγκτή $G_c(s)$ και προσδιορίζουμε ξεχωριστά τον αριθμητή (num) και τον παρονομαστή (den) του κλειστού βρόχου.

```
clear all
clc

syms s

Gp=(s+5)/(s*(s-1)*(s+1)*(s+2));

PI=0.5*(1+1/(10*s));
Gc=PI;

num=Gp*Gc;
den=1+Gp*Gc;
```

Έπειτα, προσδιορίζουμε την σχέση μεταξύ της $Y(s)$ και της $Y_{sp}(s)$ που ορίζεται από την συνάρτηση *G_closed_loop*. Συνεχίζουμε μετατρέποντας τον παρονομαστή *den* της *G_closed_loop*, από συμβολικές μεταβλητές σε πραγματικά πολυώνυμα, με τη χρήση της εντολής *sym2poly*. Κάνουμε το ίδιο και για τον αριθμητή.

```
G_closed_loop=num/den;
G_closed_loop=simplifyFraction(G_closed_loop)
[num, den]=numden(G_closed_loop);

den=sym2poly(den);
Char_Poly_den=den;
```

```
num=sym2poly(num);
Char_Poly_num=num;
```

Τέλος, προσδιορίζουμε τις ρίζες του αριθμητή (μηδενικά) και του παρανομαστή (πόλοι), κάνοντας χρήση της εντολής *roots*.

```
ROOTS_NUM=roots(Char_Poly_num)
ROOTS_DEN=roots(Char_Poly_den)
```

Μετά την εκτέλεση του κώδικά τα αποτελέσματα στο Matlab θα είναι:

```
G_closed_loop =

((10*s + 1)*(s + 5))/(20*s^5 + 40*s^4 - 20*s^3 - 30*s^2 + 51*s + 5)


ROOTS_NUM =

   -5.0000
   -0.1000


ROOTS_DEN =

  -1.6981 + 0.3140i
  -1.6981 - 0.3140i
   0.7447 + 0.5865i
   0.7447 - 0.5865i
  -0.0933 + 0.0000i
```

Η *G_closed_loop* είναι η σχέση μεταξύ $y(s)$ και $y_{sp}(s)$ που φανερώνει το χαρακτηριστικό πολυώνυμο στον παρονομαστή αλλά και το πολυώνυμο στον αριθμητή. Τα ROOTS_NUM και ROOTS_DEN παρουσιάζουν τα μηδενικά και τους πόλους αντίστοιχα.

Ολοκληρωμένος ο κώδικας του παραδείγματος βρίσκεται στο Παράρτημα (βλ. 5.3.3)

ΣΥΜΠΕΡΑΣΜΑΤΑ:

Η παρούσα διπλωματική ασχολήθηκε με την ανάπτυξη υπολογιστικών εφαρμογών, με σκοπό τη χρήση τους ως εργαλείο εκμάθησης της κλασικής θεωρίας ελέγχου.

Στο πρώτο κεφάλαιο, υπήρχε μια εισαγωγή στο τρόπο λειτουργίας του λογισμικού του Matlab. Περιγράφηκαν οι βασικές εντολές και διαδικασίες χρήσης του Matlab, που είναι απαραίτητες για την ανάπτυξη κώδικα σε αυτή. Στη συνέχεια, αναπτύχθηκαν τρία παραδείγματα μαζί με την επίλυση τους, που απαιτούσαν την ανάπτυξη των διαφορικών εξισώσεων που τα περιγράφουν καθώς και την μοντελοποίηση τους αλλά και την επίλυση τους στο περιβάλλον του Matlab. Τα παραδείγματα ήταν βασισμένα σε φαινόμενα που παρατηρούνται στον πραγματικό κόσμο ως εξής:

- Εύρεση στάθμης και θερμοκρασίας σε δοχείο πλήρους ανάμιξης με θέρμανση του ρευστού.
- Περιγραφή της μεταβολής της απόστασής με τον χρόνο σε σώμα που δέχεται την επίδραση δυνάμεων για να κινηθεί.
- Περιγραφή της μεταβολής της τάσης πυκνωτή με τον χρόνο σε ηλεκτρικό κύκλωμα.

Το δεύτερο κεφάλαιο, ασχολήθηκε με μετασχηματισμούς Laplace. Αναπτύχθηκαν δύο ολοκληρωμένα παραδείγματα που απαιτούσαν την εύρεση αντίστροφων και μη μετασχηματισμών Laplace απλών μη-γραμμικών αλγεβρικών συναρτήσεων. Υπήρχαν ακόμη παραδείγματα που χρησιμοποιόντας ένα σύνολο εξισώσεων, ζητούσαν την εύρεση των μηδενικών και των πόλων τους. Επίσης, αναπτύχθηκαν και παραδείγματα εύρεσης μετασχηματισμών Laplace σε διαφορικές εξισώσεις με την επίλυση τους σε μορφή διαγραμμάτων.

Το κεφάλαιο τρία, ασχολήθηκε με διάφορα σήματα εισόδου-εξόδου, όπως σήματα με βηματική μεταβολή, σήματα μεταβολής Dirac, τριγωνομετρικής μεταβολής αλλά και άλλα. Είχε ως στόχο την εισαγωγή στις συναρτήσεις μεταφοράς και στα συστήματα κλειστού βρόχου αλλά και στην επίλυση αποκρίσεων διαφόρων συστημάτων. Αναπτύχθηκε έτσι παράδειγμα που απαιτούσε την κατασκευή του δομικού διαγράμματος συτήματος με δοχείο υγρού, αλλά και παράδειγμα εύρεσης της απόκρισης στάθμης δοχείου. Επιπλέον, αναπτύχθηκε παράδειγμα με πλήρες δομικό διάγραμμα, που ζητούσε την εύρεση της απόκρισης σήματος εισόδου, διαταραχής αλλά και σφάλματος.

Το τέταρτο κεφάλαιο, συνεχίζει εκεί που έμεινε το προηγούμενο, με την επίλυση συστημάτων πρώτης και δεύτερης τάξης παρουσία ελεγκτή αλλά και απουσίας αυτού. Αναπτύχθηκαν τέσσερα παραδείγματα και κατασκευάστηκαν κώδικες γύρω από διαγράμματα βαθμίδων που απαιτούσαν τον προσδιορισμό της απόκρισης σημάτων εισόδου, εξόδου και σφάλματος καθώς και την παρουσίασή τους σε μορφή διαγραμμάτων.

Στο πέμπτο και τελευταίο κεφάλαιο, αναπτύχθηκαν παραδείγματα και κατασκευάστηκαν κώδικες που είχαν ως στόχο τον προσδιορισμό της ευστάθειας συστημάτων ανοικτού και κλειστού βρόχου. Ακόμη, αναπτύχθηκαν παραδείγματα για τον σχεδιασμό του γεωμετρικού τόπου των πόλων μιας συνάρτησης μεταφοράς και της εύρεση της κρίσιμης τιμής ενίσχυσης του ελεγκτή. Τέλος, υπήρχαν αναλυτικά παραδείγματα με σκοπό τον συντονισμό παραμέτρων κάνοντας χρήση της μεθόδου “Ziegler-Nichols”.

Επόμενα βήματα είναι η ανάπτυξη υπολογιστικών εφαρμογών, με σκοπό τη χρήση τους ως εργαλείο εκμάθησης, για τα:

- Μοντέρνα Θεωρία Ελέγχου
- Γραμμική/Μη-Γραμμική Θεωρία Ελέγχου
- Αποκεντρωμένος έλεγχος συστημάτων
- Ντετερμινιστικά και Στοχαστικά Συστήματα Ελέγχου

ΠΑΡΑΡΤΗΜΑ:

ΚΕΦΑΛΑΙΟ 1:

1.2.1:

```
function [dx] = Simulation_Tank_Lesson_2(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
%Modeling/Differential Equations)

%Parameters
r=1000;          %kg/m3
cp=4184;         %J/K*kg
A=1;             %m2
q_0=1e-3;        %m3/s
q=0.8e-3;        %m3/s
T0=298;          %K
q_steam=50000;   %J/s
Tref=0;          %K

%State Variables
h = x(1,1);
T = x(2,1);

%Dynamic Model of the Tank Operation
dx = zeros(2,1);

%Height h(t)
dx(1) = (1/A)*(q_0-q);

%Temperature T(t)
dx(2) = (1/(A*h))*((q_0*(T0-Tref-T))-q*Tref+q_steam/(r*cp));

return
```

```
clear
clc

tspan = [0:1:10000];

%Initial Values for Stare Variables
x0(1)=1;         %m
x0(2)=288;       %K
```

```
[tsol,xsol]=ode45(@Simulation_Tank_Lesson_2,tspan,x0);

figure(1);plot(tsol,xsol(:,1));xlabel('Time, s');ylabel('Height h, m')
figure(2);plot(tsol,xsol(:,2));xlabel('Time, s');ylabel('Temperature T, K')
```

1.2.2:

```
function [dx] = Simulation_Motion1(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
%Modeling/Differential Equations)

%Parameters
f=5;         %N
M=100;       %m3/s
fv=0.2;      %kg/s
K=1;         %kg/s2

%State Variables
x1 = x(1,1);
x2 = x(2,1);

%Dynamic Model of the mass-object Motion
dx = zeros(2,1);

%Help Variable x1(t)
dx(1)=x2;

%Help Variable x2(t)
dx(2)=(f/M)-(fv/M)*x2-(K/M)*x1;

return
```

```
clear
clc

tspan=[0:1:10000];

%Initial Values for State Variables
x0(1)=0+1e-8;
x0(2)=0+1e-8;

[tsol,xsol]=ode45(@Simulation_Motion1,tspan,x0);

figure(1);plot(tsol,xsol(:,1));xlabel('Time, s');ylabel('Distance x, m')
```

1.2.3:

```
function [dx] = Simulation_RLC(Time, x)

%Based on Lesson 2 (Tank Height and Tank Temperature Dynamic
%Modeling/Differential Equations)

%Parameters
Voltage=5;      %V
R=10;           %?
C=15;           %?F
L=230;          %mH

%State Variables
x1 = x(1,1);
x2 = x(2,1);

i=C*x2;

%Dynamic Model of the Motion
dx = zeros(2,1);

%Help Variable x1(t)
dx(1) = x2;

%Help Variable x2(t)
dx(2) = (Voltage-x1-R*C*x2)/(L*C);

return
```

```
clear
clc

tspan=[0:1:1000];

%Initial Values for State Variables
x0(1) = 0+1e-8;
x0(2) = 0+1e-8;

[tsol,xsol]=ode45(@Simulation_RLC,tspan,x0);

figure(1);plot(tsol,xsol(:,1));xlabel('Time, s');ylabel('Voltage Vc, V')
```

ΚΕΦΑΛΑΙΟ 2:

2.1.1:

```
clear all
clc

syms t

a=6;
h_help=5*t+8*t^2+sin(3*t)+5*exp(-a*t);
H_help=laplace(h_help);

'O metasximatismos Laplace ths h_help(t) einai'
H_help
```

2.1.2:

```
clear all
clc

syms t s

f=2+3*heaviside(t-5)-1*heaviside(t-8);
% h
%f=2*heaviside(t)+3*heaviside(t-5)-1*heaviside(t-8);

F=laplace(f)
f1=ilaplace(F)

t=0:0.01:10;
y=subs(f1,t);
y=double(y);

figure(1);plot(t,y,'--');ylabel('y(t)');xlabel('time');
```

2.2.1:

```
% Inverse Laplace
clear all
clc

syms s
Y1=(s^2-s-6)/((s-1)*(s+1)*(s-2));
Y2=(s+1)/(s^2-2*s+5);
Y3=(s^2+2*s+3)/(s+1)^3;
```

```
Y4=(s+1)/((s^2)*(s^2+4*s+5)*(s-1));
Y5=2*(exp(-0.5*s))/(s+1);

y1=ilaplace(Y1)
y2=ilaplace(Y2)
y3=ilaplace(Y3)
y4=ilaplace(Y4)
y5=ilaplace(Y5)
```

2.2.2:

```
clear all
clc

syms s
Y1=(s^2-s-6)/((s-1)*(s+1)*(s-2));
Y2=(s+1)/(s^2-2*s+5);
Y3=(s^2+2*s+3)/(s+1)^3;
Y4=(s+1)/((s^2)*(s^2+4*s+5)*(s-1));
%Y5=2*(exp(-0.5*s))/(s+1);

H=Y1;

[num,den]=numden(H)
num=sym2poly(num);
den=sym2poly(den);
%MIDENIKA
roots_num=roots(num)
%POLOI
roots_den=roots(den)
```

2.3.1:

```
clear all
clc

syms y t s Y F

F=laplace('diff(y(t),t,t,t)+2*diff(y(t),t,t)+3*diff(y(t),t)=20*sin(2*t)',s)

F=subs(F,{'laplace(y(t),t,s)'},{Y})
F=subs(F,{'y(0)','D(y)(0)','D(D(y))(0)'},{0,0,0})

Y=solve(F,'Y')

y=ilaplace(Y)
figure(1);ezplot(y,[0 1])
```

ΚΕΦΑΛΑΙΟ 3:

3.1.1:

```
clear all
clc

syms s

p=1000;      %kg/m3
Cp=4184;     %J/kg K
A=1;         %m2
q0=1e-3;     %m3/s
q=0.8e-3;    %m3/s
h0=1;        %m

H=(q0/(A*s^2))-(q/(A*s^2))+h0/s;
h=ilaplace(H)

t=0:1:10000;
h=subs(h,t);
h=double(h);

figure(1);plot(t,h);xlabel('time');ylabel('h(t)')
```

3.2.1:

```
clear all
clc

syms s t

%Step Change
U1=4*heaviside(t);
U1=laplace(U1);

%Ramp Change
U2=2*t;
U2=laplace(U2);

%Dirac Change
U3=dirac(t);
U3=laplace(U3);

%Cos and Sin Change
U4=cos(t)+2*sin(t);
U4=laplace(U4);
```

```
%Step (varied) Change
U5=(2/s)-(1/s)*exp(-4*s)+(2/s)*exp(-6*s);
%U5=laplace(U5);

U=U1;

K=5;t=3;

H=(K/(t*s+1))*U;
h=ilaplace(H);

t=0:0.1:40;

h=subs(h,t);
h=double(h);

u=ilaplace(U);
u=subs(u,t);
u=double(u);

figure(1);plot(t,h,t,u,'o');xlabel('time');ylabel('h(t), u(t)')
```

3.3.1:

```
clear all
clc

syms s t

Kc=1;
tI=1;
tD=1;

Gc=Kc*(1+(1/(tI*s))+tD*s);

Kv=1;
tv=1;
Gv=Kv*(1/(tv*s+1));

Kp=1;
tp=1;
Gp=Kp*(1/(tv*s+1));

Kd=1;
td=1;
Gd=Kd*(1/(td*s+1));
```

```
Km=1;
tm=1;
zm=1;
Gm=Km*(1/((tm*s)^2+2*zm*tm*s+1));

%Ysp=3/s;
Ysp=3*heaviside(t)-(2)*heaviside(t-100)+1*heaviside(t-300);
Ysp=laplace(Ysp);
D=0*heaviside(t)+(2)*heaviside(t-180)+1*heaviside(t-350);
D=laplace(D);

Y=(Gp*Gv*Gc/(1+Gp*Gv*Gc*Gm))*Ysp+(Gp/(1+Gp*Gv*Gc*Gm))*D;
Error=Ysp-Y;
y=ilaplace(Y);
e=ilaplace(Error);
U=(Gc*Gv/(1+Gp*Gv*Gc*Gm))*Ysp-(Gv*Gc*Gm*Gd/(1+Gp*Gv*Gc*Gm))*D;
u=ilaplace(U);

Total_Time=400;
t=0:1:Total_Time;

y=subs(y,t);
y=double(y);

u=subs(u,t);
u=double(u);

ysp=ilaplace(Ysp);
ysp=subs(ysp,t);
ysp=double(ysp);
d=ilaplace(D);
d=subs(d,t);
d=double(d);

e=subs(e,t);
e=double(e);

figure(1);plot(t,y,t,ysp,'--',t,d,'--');xlabel('time');ylabel('y(t)')
figure(2);plot(t,u,'-');xlabel('time');ylabel('u(t)')
figure(3);plot(t,e,'-');xlabel('time');ylabel('Error(t)')
```

ΚΕΦΑΛΑΙΟ 4:

4.1.1:

```
%Lesson 4

clc
clear all

syms s t

%Parameters
K=1;
tau=10;
Kd=1;
taud=1;

%1is taxis synartiseis metaforas
Gp=K/(tau*s+1);
Gd=Kd/(taud*s+1);

%Eisodoi / Diegerseis
%Step Change
U=(10)*heaviside(t);
U=laplace(U);

%Diataraxes
%Step Change
D=10*heaviside(t);
D= laplace(D);


Y=Gp*U+Gd*D;
Y=ilaplace(Y);

t=0:0.1:100;

Y=subs(Y,t);
Y=double(Y);

u=ilaplace(U);
u=subs(u,t);
u=double(u);

d=ilaplace(D);
d=subs(d,t);
d=double(d);

figure(1);plot(t,Y);xlabel('time');ylabel('Y(t)')
figure(2);plot(t,u);xlabel('time');ylabel('U(t)')
figure(3);plot(t,d);xlabel('time');ylabel('D(t)')
```

4.1.2:

```
clc
clear all

syms s t

%Synartisi Metaforas Elegkti
Kc=10;
tI=1;
tD=4;
tF=0.2*tD;

%P controller
P=Kc;
%PI controller
PI=Kc*(1+(1/(tI*s)));
%PID controller
PID=Kc*(1+(1/(tI*s))+tD*s/(1+tF*s));

Gc=PID;

%Synartisi Metaforas Telikou Stoixeiou
Gv=1;
%Synartisi Metaforas Metritikou Stoixeiou
Gm=1;

%Parameters
Kp=5;
taup=3;
Kd=4;
taud=3;

%1is taxis synartiseis metaforas
Gp=Kp/(taup*s+1);
Gd=Kd/(taud*s+1);



%Set Point kai Diataraxi
Ysp=1/s;
D=(1/s);

%Sxesi Kleistou Broxou
Y=(Gp*Gv*Gc/(1+Gp*Gv*Gc*Gm))*Ysp+(Gd/(1+Gp*Gv*Gc*Gm))*D;
Error=Ysp-Y;
U=(Gc*Gv/(1+Gp*Gv*Gc*Gm))*Ysp-(Gv*Gc*Gm*Gd/(1+Gp*Gv*Gc*Gm))*D;



Total_Time=50;
t=0:0.5:Total_Time;

y=ilaplace(Y);
y=subs(y,t);
y=double(y);
```

```
u=ilaplace(U);
u=subs(u,t);
u=double(u);

Ysp=ilaplace(Ysp);
ysp=subs(Ysp,t);
ysp=double(ysp);

D=ilaplace(D);
d=subs(D,t);
d=double(d);

e=ilaplace(Error);
e=subs(e,t);
e=double(e);


figure(1);plot(t,y,t,ysp,'--', t, d,'--');xlabel('time');ylabel('y(t)')
figure(2);plot(t,u,'-');xlabel('time');ylabel('u(t)')
figure(3);plot(t,e,'-');xlabel('time');ylabel('Error(t)')
```

4.2.1:

```
clc
clear all

syms s t

%Parameters
K=1;
tau=1;
z=0.5;

%1is taxis synartiseis metaforas
Gp=K/((tau^2)*(s^2)+2*z*tau*s+1);

%Eisodoi / Diegerseis
%Step Change
U=(10)*heaviside(t);
U=laplace(U);


Y=Gp*U;
Y=ilaplace(Y);

t=0:0.1:100;

Y=subs(Y,t);
Y=double(Y);

u=ilaplace(U);
u=subs(u,t);
```

```
u=double(u);


figure(1);plot(t,Y);xlabel('time');ylabel('Y(t)')
figure(2);plot(t,u);xlabel('time');ylabel('U(t)')


tr=(0.8+2.5*z)*tau
tp=pi*tau/(1-z^2)^0.5
Mp=exp(-pi*z/(1-z^2)^0.5)
ts_2=4*tau/z
ts_5=3*tau/z
```

4.2.2:

```
clc
clear all

syms s t

%Synartisi Metaforas Elegkti
Kc=1;
tI=1;
tD=4;
tF=0.2*tD;

%P controller
P=Kc;
%PI controller
PI=Kc*(1+(1/(tI*s)));
%PID controller
PID=Kc*(1+(1/(tI*s))+tD*s/(1+tF*s));

Gc=PID;

%Synartisi Metaforas Telikou Stoixeiou
Gv=1;
%Synartisi Metaforas Metritikou Stoixeiou
Gm=1;

%Parameters
%Parameters
Kp=1;
taup=1;
zp=0.0;

Kd=4;
taud=3;

%1is taxis synartiseis metaforas
Gp=Kp/((taup^2)*(s^2)+2*zp*taup*s+1);
Gd=Kd/(taud*s+1);
```

```
%Set Point kai Diataraxi
Ysp=(1/s)*(1+3*exp(-15*s));
D=(1/s)*(1+8*exp(-45*s));

%Sxesi Kleistou Broxou
Y=(Gp*Gv*Gc/(1+Gp*Gv*Gc*Gm))*Ysp+(Gd/(1+Gp*Gv*Gc*Gm))*D;
Error=Ysp-Y;
U=(Gc*Gv/(1+Gp*Gv*Gc*Gm))*Ysp-(Gv*Gc*Gm*Gd/(1+Gp*Gv*Gc*Gm))*D;


Total_Time=150;
t=0:0.5:Total_Time;

y=ilaplace(Y);
y=vpa(y);
y=subs(y,t);
y=double(y);

u=ilaplace(U);
u=vpa(u);
u=subs(u,t);
u=double(u);

ysp=ilaplace(Ysp);
ysp=vpa(ysp);
ysp=subs(ysp,t);
ysp=double(ysp);

d=ilaplace(D);
d=vpa(d);
d=subs(d,t);
d=double(d);

e=ilaplace(Error);
e=vpa(e);
e=subs(e,t);
e=double(e);


figure(1);plot(t,y,t,ysp,'--', t, d,'--');xlabel('time');ylabel('y(t)')
figure(2);plot(t,u,'-');xlabel('time');ylabel('u(t)')
figure(3);plot(t,e,'-');xlabel('time');ylabel('Error(t)')
```

ΚΕΦΑΛΑΙΟ 5:

5.1.1:

```
clc
clear all

syms s

Gp=1/(s*(s-1)*(s+1)*(s+2));
[num,den]=numden(Gp);

num=sym2poly(num);
den=sym2poly(den);

ROOTS=roots(den)
[a,b]=size(ROOTS)

for i=1:1:a;

    if ROOTS(i) > 0
        disp('to systhma einai astathes')
    elseif ROOTS(i) < 0
        disp('to systhma einai eustathes')
    else
        disp('to systhma einai oriaka eustathes')
    end
end
```

5.1.2:

```
clear all
clc

syms s

Gp=1/(s*(s-1)*(s+1)*(s+2));
Kc=1;
Gc=Kc;

num=Gp*Gc;
den=1+Gp*Gc;

G_closed_loop=num/den;
G_closed_loop=simplifyFraction(G_closed_loop)

[num,den]=numden(G_closed_loop);
den=sym2poly(den);
Char_Poly=den;

ROOTS=roots(Char_Poly)
```

```
[a,b]=size(ROOTS)

for i=1:1:a;

    if ROOTS(i) > 0
        disp('to systhma einai astathes')
    elseif ROOTS(i) < 0
        disp('to systhma einai eustathes')
    else
        disp('to systhma einai oriaka eustathes')
    end
end
```

5.2.1:

```
clear all
clc

syms s

Gp=1/((s+1)*(s+1)*(s+2));
Kc=1;
Gc=Kc;

G_open_loop=Gp*Gc;
G_open_loop=simplifyFraction(G_open_loop)
[num,den]=numden(G_open_loop)

num=sym2poly(num);
den=sym2poly(den);

G_root_locus=tf(num,den)

figure(1);rlocus(G_root_locus)
```

5.3.1:

```
clear all
clc

syms s

Gp=1/((s+1)*(s+1)*(s+2));
Kc=17.9;
Gc=Kc;

Ysp=1/s;


Y=(Gp*Gc/(1+Gp*Gc))*Ysp;
y=ilaplace(Y);
```

```
Total_Time=10;
t=0:0.1:Total_Time;

y=subs(y,t);
y=double(y);
ysp1=ilaplace(Ysp);
ysp1=subs(ysp1,t);
ysp1=double(ysp1);

ysp=ysp1;
figure(1);plot(t,y,t,ysp,'--');xlabel('time, min');ylabel('y(t)')
```

5.3.2:

```
clear all
clc

syms s

Gp=1/((s+1)*(s+1)*(s+2));
Ysp=1/s;

%P Ziegler Nichols
Kc=0.5*17.9;
P=Kc;

%PI Ziegler Nichols
Kc=0.45*17.9; tI=(1/1.2)*2.8;
PI=Kc*(1+(1/(tI*s)));

%PID Ziegler Nichols
Kc=0.6*17.9; tI=(0.5)*2.8; tD=0.125*2.8;
PID=Kc*(1+(1/(tI*s))+tD*s);

Gc=PID;

Y=(Gp*Gc/(1+Gp*Gc))*Ysp;
y=ilaplace(Y);

Total_Time=30;
t=0:0.1:Total_Time;

y=subs(y,t);
y=double(y);
ysp1=ilaplace(Ysp);
ysp1=subs(ysp1,t);
ysp1=double(ysp1);

ysp=ysp1;
figure(1);plot(t,y,t,ysp,'--');xlabel('time, min');ylabel('y(t)')
```

5.3.3:

```
clear all
clc

syms s

Gp=(s+5)/(s*(s-1)*(s+1)*(s+2));

PI=0.5*(1+1/(10*s));
Gc=PI;

num=Gp*Gc;
den=1+Gp*Gc;

G_closed_loop=num/den;
G_closed_loop=simplifyFraction(G_closed_loop)
[num, den]=numden(G_closed_loop);

den=sym2poly(den);
Char_Poly_den=den;

num=sym2poly(num);
Char_Poly_num=num;

ROOTS_NUM=roots(Char_Poly_num)
ROOTS_DEN=roots(Char_Poly_den)
```