Bölüm 1 : MATLAB hakkında genel bilgiler
Bölüm 2 : MATLAB'ın temel birimleri
Bölüm 3 : MATLAB'da denklemlerin çözümlenmesi
Bölüm 4 : MATLAB'da kontol sistemlerinin modellenmesi, modellere erişim ve birbirine dönüştürülmesi
Bölüm 5 : MATLAB'da oluşturulan modellerin birbirine bağlanması
Bölüm 6 : Kontrol sistemlerinde kullanılan giriş fonksiyonları
Bölüm 7 : MATLAB'da Routh-Hurwitz çözümü
Bölüm 8 : Kök-yer eğrilerinin incelenmesi
Bölüm 9 : MATLAB'da frekans alanı analizi
Bölüm 10 : MATLAB'da PID tasarımı
Bölüm 11 : MATLAB/Simulink'in incelenmesi ve blokların tanıtımı
Bölüm 12 : Aç-kapa tipi kontrolörün modellenmesi

İÇİNDEKİLER

BÖLÜM 1

GİRİŞ

1.1 Temel Bilgiler

MATLAB, başlangıçta bilim adamları ve mühendislerin, matrislere dayalı problemleri -program yazmaksızın- çözebilmelerini sağlamak amacıyla tasarlanmış bir sayısal çözümleme programıdır. MATris LABoratuarı adıyla ilk defa ortaya çıkmıştır. MATLAB bir yorumlayıcıdır; yani, daha ziyade el tipi hesap makinelerine benzer tarzda, program çalıştırılırken ekranda da aynı zamanda sonuçları görmek mümkündür. Neticede diğer dillerde olduğu gibi "derleme"ye ihtiyaç yoktur. MATLAB, programlamaya da izin vermesi sebebiyle yüksek düzeyde bir programlama dili olarak da kullanılabilir.

MATLAB' ın gücü sadece onun kullanıcıya yardımcı, etkileşimli bir arayüze sahip olmasından değil, aynı zamanda yüksek performanslı bilimsel/matematiksel alt programlar koleksiyonu ve güvenilir bir algoritmik temel içermesinden kaynaklanmaktadır. Bundan başka, MATLAB kullanıcılara bir grafiksel kullanıcı ara yüzü ve grafik animasyon kabiliyeti de sunmaktadır.

MATLAB,

- Optimization Toolbox (Optimizasyon kütüphanesi),
- Control System Toolbox (Denetim sistem kütüphanesi),
- Neural Network Toolbox (Yapay sinir ağları kütüphanesi),
- Fuzzy Logic Toolbox (Bulanık mantık kütüphanesi),

gibi giderek artan sayıda özelleştirilmiş kütüphaneleri içeren ve hala gelişmekte olan bir paket programdır. Bu kütüphaneler kullanıcılara uzmanlık alanlarındaki uygulamalarının her birinde geniş imkanlar sağlamaktadır.

1.2 MATLAB' in Masaüstü Birimleri

Bir Windows ortamında MATLAB yazılımını başlatmak için bilgisayarınızın masaüstündeki MATLAB sembolüne iki defa tıklamak yeterlidir.Şekil 1.1'deki pencere ekrana gelecektir.

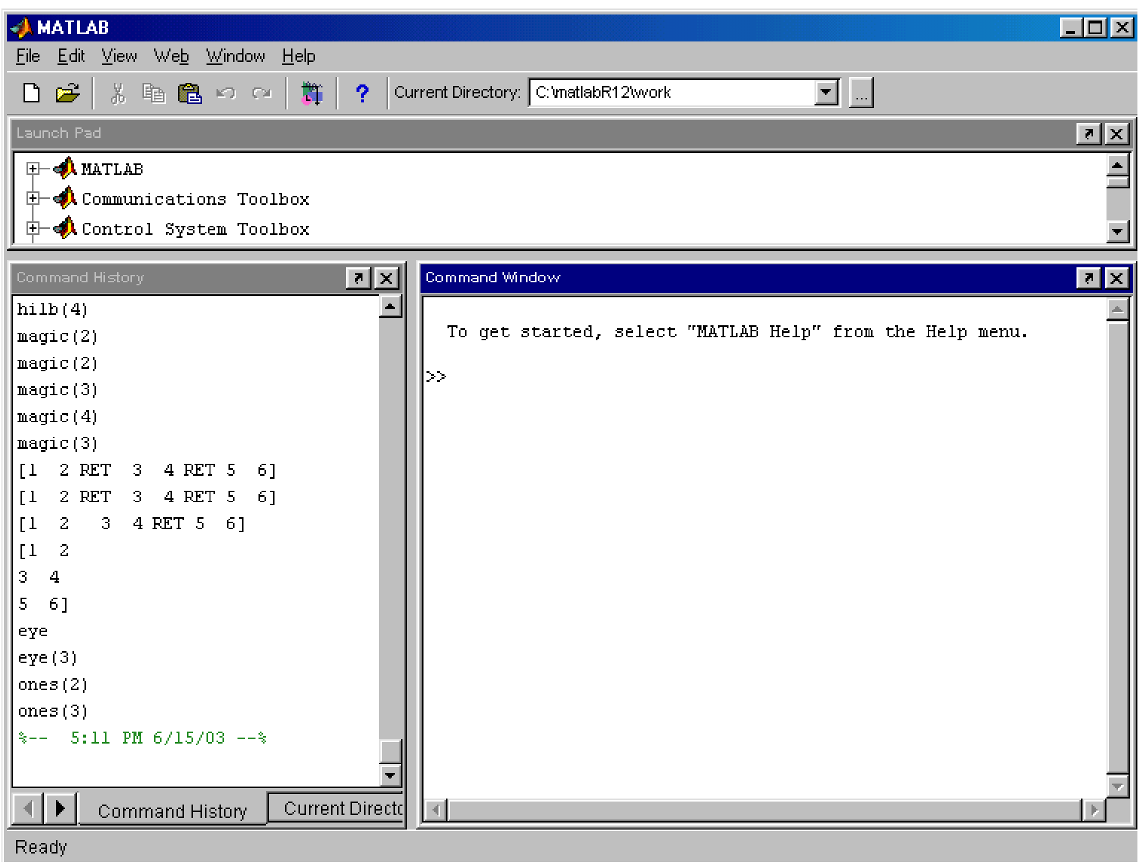
MATLAB' ın sahip olduğu çevresel birimler şunlardır; **[1]**

- Komut Penceresi(Command Window)
- Bulunulan Kütük (Current Directory)
- Komut Tarihi (Command History)
- Çalışma Alanı (Workspace)
- Dizin Erişimi (Launch Pad)

- Yardım (Help)

1.2.1 Komut Penceresi(Command Window)

MATLAB ile iletişim kurulan ana penceredir. MS Windows platformlarında, MATLAB yalnızca Windows'a ait niteliklerle özgün bir pencere sunar. MATLAB yorumlayıcısı sizden gelecek komutları kabul etmeye hazır olduğunu gösteren (») biçiminde bir ileti görüntüler. Bu alanda o an işletilmek istenilen bütün komutlar yazılabilir. Komut penceresinde komutlar satırları tek tek işletilirler. Yazılan komutun icrası için klavyedeki giriş (enter) tuşu kullanılır. Klavyede yer alan sağ ve sol ok tuşları komut satırında yazılan yanlışların düzeltilmesine olanak tanır. Herhangi bir anda Ctrl-c yada Ctrl-Pause/Break tuşlarına basarak çalışmakta olan bir program sonlandırılabilir. Bir komut deyiminin bir satıra sığmaması durumunda ard arda 3 nokta "..." ve ardından Enter tuşuna basılarak bu deyimin bir sonraki satır veya satırlarda devam ettiği beyan edilmiş olur.



Şekil 1.1 MATLAB ve Çevre Birimleri

1.2.2 Bulunulan Kütük (Current Directory)

MATLAB çalıştığında hangi kütük (directory) içinde olduğunu ve bu kütük içinde hangi kütük ve dosyaların bulunduğunu gösteren kısımdır. Sahip olduğumuz bir MATLAB dosyasını, komut penceresinde ismi ile çağırmak istediğimizde çalıştırılacak dosyanın bulunulan kütük içinde olması gerekmektedir. MATLAB kütüphaneleri bu şarta dahil değildirler.

1.2.3 Komut Tarihi (Command History)

MATLAB her açıldığında, açılan oturum için bir komut tarihi bölümü açar. Bu bölümlerde oturum açılışından kapanıncaya kadar işletilen komut satırlarının tümü kaydedilir. Bu komutlara başka oturumlarda yada içinde bulunulan oturumda tekrar ulaşılabilir. Bunun için MATLAB komut penceresinde yukarı ve aşağı ok tuşlarını kullanmak yeterlidir. Komut tarihi alanında bir komut satırının fare ile iki kere tıklanması sonucunda komut penceresinde o komut işletilir.

1.2.4 Çalışma Alanı (Workspace)

MATLAB çalışma alanı, MATLAB komut hattından kullanılabilecek değişkenler (dizinler 'arrays' olarak bilinen) takımını içerir. who veya whos komutlarını kullanarak o andaki çalışma alanı içindekiler görüntülenebilir. who komutu sadece değişkenlerin isimlerini kısa bir liste halinde verirken, buna karşılık whos komutu ayrıca boyutu ve veri türü bilgilerini de verir.

Çalışma alanı ayrıca komut penceresinde yer alan araçlar üzerindeki çalışma alanı tarayıcısı (Workspace Browser) penceresini açarak da görüntülenebilir. Tüm bilgilerin görüntülendiği bu pencerenin araçlar üzeride küp biçimde bir şekli vardır. Bu şekil üzerinde tıklandığında tarayıcı pencere açılır.

Çalışma alanında yer alan tüm değişkenleri silmek için
clear komutu kullanılır.

Çalışma alanındaki verilerin kaydedilmesi ve yüklenmesi, 'save'
ve 'load' komutları: MATLAB' m 'save' ve 'load' komutları bir oturumun her hangi bir anında MATLAB çalışma alanı içeriklerinin kaydedilmesini ve bu oturum sırasında veya daha sonraki bir oturumda kaydedilen bu verilerin tekrar çalışma alanına yüklenmesini sağlar, 'save' ve 'load' komutları aynı zamanda yazı (text) türü veri dosyalarının da çalışma ortamına ithal ve ihraç edilmesini de sağlar.

'save' komutu çalışına alanı içeriğini bir ikili sayılar (binary) MAT-dosyası olarak kaydeder. Bu dosya daha sonra 'load' komutu ile geri çağrılabilir. **[3]**

Örneğin,

```
>> save bafra
```

tüm çalışma alanı içeriğini bafra.mat dosyası içine saklar. Gerektiğinde, dosya adından sonra değişken isimlerini belirleyerek yalnızca belli değişkenlerin kaydedilmesini sağlamak mümkündür.

Örneğin,

```
>>save bafra  x  y  z
```

olduğu gibi yalnızca x, y ve z değişkeni saklanmış olur. Varsayılan biçimi ile ‘save’ komutu değişkenleri ".mat" uzantısı ile kaydetmekle beraber ASCII biçiminde de kaydetmesi de mümkündür. MATLAB komut penceresinde,

```
>>help   save
```

komutu ile bu konuda daha fazla bilgi elde edilebilir.
‘load’ komutu daha önceden ‘save’ komutu ile oluşturulan *mat-dosyasının* çalışma alanına yükler.
Örneğin

```
>>load bafra
```

komutu bafra.mat çalışma alanına yükler. Eğer *mat-dosyası*, x, y ve z değişkenlerini içeriyorsa bafra.mat yüklenmesi ile x, y ve z değişkenleri çalışma alanına yerleştirilecektir. Bu değişkenler halihazırda çalışma alanında mevcut ise yerlerine bu yüklenen değişkenler geçecektir.

1.2.5 Dizin Erişimi (Launch Pad)

Kullanıcıya sunulan tüm kütüphanelere, demolara (MATLAB’ in sahip olduğu örnek hazır programlar) ve dokümantasyona kolayca erişim imkanı sağlar.

1.2.6 Yardım (Help)

MATLAB hem yeni başlayanlara hem de uzmanlaşmış olanlara çok pratik bir yardım da sağlamaktadır. Özel bir komut hakkında bilgi edinmek için ekrandan "help komut adı" komutu girilir; burada "komut adı" ilgilendiğiniz MATLAB fonksiyonunu temsil eden komutlardır. Kullanıcı, belirli bir konuda hangi komutları kullanabileceğini bilmek isterse "lookfor kavram adı" komutunu girmesi yeter. Örneğin, "lookfor sinus" yazılıp enter tuşuna basıldığında; MATLAB her biri için kısa açıklamalarıyla birlikte sinus’e ilişkin tüm komutları sıralar. Şayet aranan bilgi bir fonksiyon ise "help sin" komutunun girilmesi aşağıdaki pencerede gösterildiği gibi ilgili komuta ait ayrıntılı bir yardımla neticelenir. **[1]**

```
>> lookfor sinus
FLATPLRS  McBryde-Thomas Flat-Polar Sinusoidal Pseudocylindrical Projection
MODSINE  Tissot Modified Sinusoidal Pseudocylindrical Projection
SINUSOID  Sinusoidal Pseudocylindrical Projection
GAUSPULS Gaussian-modulated sinusoidal pulse generator.
ROOTEIG   Computes the frequencies and powers of sinusoids via the
ROOTMUSIC   Computes the frequencies and powers of sinusoids via the
>> help sin

 SIN    Sine.
    SIN(X) is the sine of the elements of X.
 Overloaded methods
    help sym/sin.m
```

1.3 MATLAB' da Menüler

MATLAB oturumu ilk defa başlatıldığında. Şekil 1.1 deki pencere açılacaktır. MATLAB masa üstünün görünümü, kapatılarak, taşınarak veya tekrar büyüklüğü ayarlanarak istenilen şekle getirilebilir. Bu alt pencereler, masaüstünün haricine de taşınabilir. Şekil 1.1'de görüldüğü gibi MATLAB 6 adet menü başlığına sahiptir.
Bunlar sırasıyla;

File
Edit
View
Web
Window
Help' tir.

File : Bu menüde yeralan başlıklardan ilki New' dir. Bu başlık ile MATLAB' de 4 dosya modeli oluşturabiliriz. Sırasıyla M-file, Figure, Model ve GUI' dir. M-file MATLAB' in sahip olduğu programlama diliyle program yazılmasına olanak tanıyan bir dosyadır. Figure MATLAB' de grafikler için kullanılan bir penceredir. Model, MATLAB' de simülasyon için kullanılan bir yapıya sahiptir. Bir çok alanda modelleme burada mümkündür. GUI(Graphical User Interfaces)' de ise grafiksel bir programlama gerçekleştirilmek mümkündür. Open ile açılan diyalog kutusundan MATLAB' in tanımlayabildiği istenilen bir dosya açılabilir. Import Data... ile dışarıdan veri transferi mümkündür. 'save' workspace as... ile o anki çalışma alanı istenilen bir dosya ismiyle kopyalanabilir. Set path ile MATLAB, M-dosyaları ve MAT-dosyaları ile çalışmak için bu dosyaların yolunu (folder path). Preferences (tercihler) seçilerek, MATLAB masaüstünün karakteristikleri değiştirilebilir. Örneğin, Command Window'da kullanılacak olan yazı karakteristiklerini ayarlamak mümkündür.

Edit : Bu menüde bulunan Clear Command Window, Clear Command History, ve Clear Workspace başlıkları ile bu alanlarda kaydedilen bilgiler ve komutlar silinir.

View : Bu menüde yer alan başlıklarla MATLAB masaüstünde hangi çevre birimleri ile çalışılmak isteniyorsa bunlar seçilebilir.

Web : Bu menü ile MATLAB' i üreten firmanın hazırlamış olduğu çok geniş bir siteye ulaşmak ve burdan teknik destek almak mümkündür.

Help : Bu menüde yeralan başlıklarla MATLAB' in kurulumuyla gelen yardım dosyalarına ulaşılabilir. Ayrıca Demo programlara buradan da ulaşılabilir.

1.4 Hesap Makinesi Olarak MATLAB

Ekrana bir ifade yazılıp giriş (enter) tuşuna basıldığında MATLAB bu komutu hemen icra eder ve sonucu komut penceresinde ekrana basar. Eğer kullanıcı özel bir komutun sonucu olan çıktıyı görmek istemiyorsa komut bittikten sonra basitçe bir ";" işareti, yani noktalı virgül koymak yeterlidir. Açıklamalar yazılmak istendiğinde, açıklama satırının önüne % (yüzde) işareti konmalıdır. Şu halde bir satırda % işaretini takip eden herhangi bir bilgi, açıklama olarak algılanır ve icra edilmez. Bazı temel işlemler aşağıdaki çerçevede sunulmaktadır. Aşağıda görüldüğü gibi bir işlem yapılırken "=" sembolü ile bir değişken içine atanmadığında yanıt ans içine atanır. Ans, answer' ın kısaltılış şeklidir.

Örneğin,

```
» 2+5
ans =
        7
» 2 + 3;                    % Sonucun ekranda gösterilmesi noktalı
                            % virgülle önlenmiştir.
» 3^2
ans =
        9
» sin(pi/4)       % Açılar radyan cinsinden olmalıdır
                  % ve pi standart n'yi temsil eder.
ans =
        0.7071
»2*(3+4)

ans =
      14
»2+3i             % i veya j karmaşık sayı gösterimi
                  % için kullanılabilmektedir.
ans =
        2.0000 + 3.0000i
```

MATLAB' da ki temel aritmetik işlemler Tablo 1.1'de artan öncelik sırasına göre özetlenmiştir. Çarpma ve bölme aynı öncelik hakkına sahiptirler. Benzer şekilde toplama ve çıkarmanın öncelikleri eşittir. Parantez içerisindeki işlemler, en önce icra edilirler.

Tablo 1.1
MATLAB' da aritmetik işlemler.

İşlem	Sembol	Örnek
Toplama, a+b	+	2+3
Çıkarma, a-b	-	5-2
Çarpma, a*b	*	3*4
Bölme, a/b	/	14/7
Üs alma, a^b	^	2^3

MATLAB' daki ilişki işlemcileri skalerlerin ve aynı mertebede matrislerin mukayesesi için kullanılmaktadırlar. Sonuç, ilişkinin doğru veya yanlış olmasına göre sırasıyla l veya O olur. Bu işlemciler aşağıda Tablo 1.2'de listelenmiştir.

Tablo 1.2
İlişki İşlemcileri

İşlemci	Anlamı	Örnek
<	...den küçük	3<5
>	...den büyük	7>2
< =	...den küçük veya ...e eşit	$4 \leq 4$
>=	...den büyük veya ...e eşit	$5 \geq 1$
= =	Eşit	5=5
~ =	eşit değil	$3 \neq 8$

“= =” sembolü kontrol amaçlı ”eşit” anlamında, buna mukabil "=" işareti ise atama ifadelerinde; yani bazı aritmetik işlemler neticesi bulunan bir değerin bir değişkene atanması halinde kullanılmaktadır (atamalar için Bölüm 1.3'e bakınız).
Mantık işlemcileri, ilişki işlemcileriyle birlikte kullanılabilirler. Bunlar aşağıda Tablo 1.3'te özetlenmiştir. Buraya kadar verilen üç farklı sınıftaki işlemcilerin icradaki öncelik sırası aritmetik, ilişki ve mantık işlemcileri şeklindedir.

Tablo 1.3 Mantık işlemcileri

İşlemci	Anlamı	Örnek
&	AND (VE)	A&B
|	OR (VEYA)	A|B
~	NOT (DEĞİL)	~ A

Örneğin,

```
» 3 < 5        % Üç beşten küçük müdür?  Doğru (1)
ans =
        1
» a = 5= =8                 % Sonucu bir değişkene atamak da
                            % mümkündür.
                            % 5 8'e eşitse a=1 , aksi halde a=0
a =
        0
» 1|0                       % l veya O mıdır? Doğru (1)

ans =
        1
```

Mantık önermeleri mühendislik ve matematikte sıkça karşılaşılabilen konulardandır. Aşağıda, pratik bir uygulama olarak, verilen bir sayının başka bir sayıya kalansız bölünüp bölünemediği araştırılıyor. Kullanılan rem komutu, bölme işleminde kalanı verir.

```
» m=rem(5280,8)= =0          % 5280 sayısı 8'e kalansız
                             % bölünüyor ise m=l, aksi
                             % halde m=0
m =
        1
```

MATLAB aynı zamanda pek çok hazır matematik ve nümerik fonksiyonlara sahiptir. Bu fonksiyonların bazıları Tablo 1.4'te listelenmiştir.

Tablo 1.4
MATLAB'daki hazır matematik fonksiyonlar.

Fonksiyon	Sembol	Örnek
Sinüs, sin(θ)	sin	sin(pi)
Kosinüs, cos(θ)	cos	cos(pi)
Tanjant, tan(θ)	tan	tan(pi)
Arksinüs, arcsin(θ)	asin	asin(0)
Arkkosinüs, arccos(θ)	acos	acos(0)
Arktanjant, arctan(θ)	atan	atan(l)
Eksponensiyal, e^x	exp	exp(2)
Tabii logaritma	loğ	log(10)
10 tabanlı logaritma	Logl0	Log10(l0)
Kare kök, $\sqrt{x}$	sqrt	sqrt(25)
Mutlak değer, |x|	Abs	abs(3)

1.5 Değişkenler

Diğer programlama dillerin pek çoğunda olduğu gibi MATLAB da matematik deyimler şart koşmakla beraber, diğer pek çok programlama dillerinden farklı olarak bu deyimler tümüyle matrisleri kapsar.

MATLAB her hangi bir tür bildiri veya boyut ifadeleri gerektirmez. MATLAB yeni bir değişken ile karşı karşıya geldiğinde, otomatik olarak değişken oluşturur ve yeteri kadar bellek ayırır. Eğer değişken daha önceden mevcut ise MATLAB onun içeriğini değiştirir ve eğer gerekliyse yeni bellek ayırır. Örneğin,

```
» no=10;
```

komut satırı, no adı altında 1x1 lik bir matris oluşturur ve 10 değerini matrisin tek elemanı içine kaydeder.
Diğer bilgisayar dillerinde olduğu gibi MATLAB' in değişken isimleri konusunda bazı kuralları vardır. En basit değişken ismi tek bir harften (karakterden) ibarettir. Belli başlı kurallar aşağıda olduğu gibi özetlenebilir.

- Değişken isimleri küçük/büyük harf kullanımına duyarlıdır. Buna göre aynı anlama gelen fakat farklı yazılan saYi, Sayi, sAyi ve SAYI kelimeleri MATLAB için farklı değişkendirler. Türkçe karakter kullanımı mümkün olmamaktadır.
- Değişken isimleri en çok 31 karakter içerebilir. Bir değişken isminde 31 karakterden daha fazla karakter varsa hesaba katılmaz.
- Değişken isimleri daima bir harf ile başlamalı ve bunu herhangi bir sayıda harfler, rakamlar veya alt çizgi "J* izleyebilir. Noktalama işaretleri değişken ismi olarak kullanılamaz. Çünkü bunların pek çoğunun MATLAB için özel bir anlamı vardır.

Rakamlar:

MATLAB rakamlar için önünde artı veya eksi işareti ve tercihli ondalık noktası ile birlikte alışılagelmiş ondalık (decimal) işaretler sistemi kullanır. Bilimsel işaretler sistemi 10 tabanına göre kuvvet belirlemek için e harfi kullanır. Sanal rakamlar için "son takı" olarak i veya j harfi kullanır. Kurala uygun rakamlar ile ilgili bazı örnekler;

```
        6           -15           0.0003
2.6397238   1.60210e-20         -3.14159J
6.02252e23
3e5i
```

Tüm rakamlar IEEE hareketli nokta (floating-point) standart ile belirlenmiş uzun format kullanarak dahili olarak saklanır. Hareketli nokta rakamları kabaca virgülden önce 16 hanelik ondalık sayılı sonlu bir kesinliğe sahip olup bunun sonlu alanı 10^{-308} ile 10^{+380} arasındadır.

MATLAB, kullanıcıların değişkenlere değer atamasına müsaade etmektedir. Böyle bir atama yapabilmek için eşit işaretinin sol tarafına bir değişken ismi girilmelidir. Aşağıdaki örnekler değişkenlere değer ve ifadelerin nasıl atanacağını, ve MATLAB'm verdiği cevapları göstermektedirler. **[3]**

```
a = 3 , b = 2          % Virgülle ayırmak kaydıyla bir
                       % satırda birden fazla
                       % atama yapılabilir.
a =
     3
b =
     2
c = a + b
c =
     5
» d = a^b
d =
     9
```

Belirli bir program içindeki aktif değişkenlerin bir listesi arzu edilirse "who" yazıldığında, o anda kullanılan değişkenler listelenir. Bu değişkenlerin herhangi birini iptal için "clear değişken adı" komutunu kullanabilirsiniz.

```
» who
Your variables are:
a       ans      b     c     d
```

```
» clear ans                  % ans adlı değişken silindi
» who
Your variables    a    b    c     d
```

Bir değişkenin değerini görmek için değişkenin adını girmek yeterlidir.

```
»  d
d  =
         9
```

MATLAB' da, п ve çok küçük bir e sayısını temsilen, "pi' and "eps" gibi hazır değişkenler de vardır.

```
» pi
ans =
         3.1416
» eps
ans =
         2.2204e-016
```

BÖLÜM 2

MATLAB TEMEL BİRİMLERİ

2.1 Fonksiyonlar

MATLAB’ da çok miktarda hazır fonksiyon vardır. Bu fonksiyonlar kullanıcıların işlemlerini kısaltmakla birlikte varolan bazı özel fonksiyonlarda herhangi bir programlama dilinde yüzlerce satır program yazılmasıyla elde edilen programların yerini alabilmektedir.

Tablo 2.1 MATLAB Fonksiyonları **[3]**

Fonksiyon	**Tanım**
tf	Transfer fonksiyonu oluşturmak
tfdata	Transfer fonksiyonu verilerinin geri alınması
size	Çıkış/giriş/dizim boyutları veya model derecesini elde eder
bode	Verilen sistemin bode diyagramını çizer
rlocus	Verilen sistemin kök-yer (root-locus) çizer
esort	Sürekli zaman kutuplarını gerçek kısımlarına göre sıraya sokar
roots	Polinomun köklerini hesaplar
step	Basamak cevabı hesaplar
conv	İki polinomun çarpımını hesaplar

series	Seri bağlı blokları indirger
nichols	Nichols abağını çizer
ngrid	Bir nichols abağı üzerine enine-boyuna çizgileri koyar
mrgin	Kazanç ve faz paylarını inceler
sigma	Tekil değer grafiğini hesaplar, çizer
logspace	Logaritmik aralıklı bir frekans vektörü oluşturur
linspace	Düzgün aralıklı bir frekans vektörü oluşturur
gensig	Bir giriş sinyali üretir
impulse	Ani darbe cevabını hesaplar

MATLAB' da daha bir çok komut bulunmaktadır. Belki de çok azı bu tez içerisinde yer alacaktır. İleriki bölümlerde özellikle o bölüme ait olan fonksiyonlar verilecektir.
MATLAB' ın fonksiyonlar ile kullanıcıya sağladığı kolaylıklardan bir tanesi de yeni fonksiyonlar yazımına ve bunların kullanılmasına olanak sağlamasıdır.

2.2 Vektörler

MATLAB' de vektörler tek satır yada tek sütun matrisler olarak değerlendirilirler. Grafik çizimlerinde koordinat değerleri vektörsel olarak ifade edilmektedir. Bu nedenle önemi büyüktür.

MATLAB' de vektörleri üretmek için bir dizi yöntem vardır. Bunların birkaçı burada sunulacaktır. Bir vektör oluşturmanın yollarından biri, vektörün elemanlarını bir köşeli parantez içinde sıralamak ve sonucu bir değişken içine atamaktır. Satır vektörü için, elemanların arasında virgül yada boşluk olması gerekir. Sütun vektörü için elemanların arasında noktalı virgül olması yada her satırdan sonra giriş (enter) tuşuna gerekmektedir. **[9]**
Örneğin,

```
>> a=[0 1 2]

a =

    0     1     2

>> a=[0,1,2]

a =

    0     1     2

>> q=[3;4;5]
```

```
q =

     3
     4
     5

>> w=[1
2
3]

w =

     1
     2
     3
```

Bir satır transpozunu almak o vektörün elemanlarını sütun şeklinde yazmaktır. Sütun şeklinde olan vektörün transpozu da, elemanların satır şeklinde yazılmasıdır. MATLAB' de bir vektörün transpozunu almak için kesme (apostrof) kullanılır.

Örneğin,

```
b =

     1     3     5

>> b'

ans =

     1
     3
     5
```

Elemanları sabit bir olarak artan yada azalan bir vektör oluşturmak için ikin okta üst üste kullanılır. Bu şekilde vektör oluşturmanın genel ifadesi $X = X_{\min} : \Delta X : X_{\max}$ şeklindedir.Burada, X ile gösterilen elemanları $X_{\min}$ den başlayan ve ΔX artımlarıyla $X_{\max}$ ile biten bir vektördür. ΔX negatifte olabilir. Yani, azalım olabilir. Eğer artım bir olacaksa ΔX yazılmayabilir.
Örneğin,

```
>> A=[0:2:6]

A =

     0     2     4     6

>> A=[6:-2:0]
```

```
A =

    6    4    2    0
```

2.2.1 Vektörlerle İşlemler

Toplama ve çarpma gibi cebir kuralları MATLAB'daki vektör işlemlerine uygulanmaktadırlar; yalnız vektörler aynı boyutta olmalıdırlar. Vektörler üzerinde işlem yapan diğer iki mühim fonksiyon nokta (skaler) ve vektörel çarpımlardır, a ve b iki vektör olmak üzere bu iki işlem için genel şekil sırasıyla "dot(a,b}" ve "cross (a,b)"dir. Bir vektörün şiddeti "norm" fonksiyonuyla hesaplanabilir.
Örneğin,

```
>> a=[1 2 3];b=[0,1,-1;]

b =

    0    1   -1

>> a+b

ans =

    1    3    2

>> a-2*b

ans =

    1    0    5

>> dot(a,b)

ans =

   -1

>> cross(a,b)

ans =

   -5    1    1

>> norm(a)

ans =

   3.7417
```

2.3 Polinomlar

Bir polinom, genellikle bir P fonksiyonunun

$P(s) = a_n s^n + a_{n-1} s^{n-1} + ... + a_0$

şeklinde s değişkeni cinsinden ifade edilmesidir. Burada a katsayılar olup değişkenin en yüksek derecesi polinomun da derecesidir.

Polinomlar, kontrol sistemlerinde sıkça kullanılırlar. MATLAB ise polinom işlemlerini kolaylaştıracak güçlü fonksiyonları kullanıcılara sunar.

Polinomlar, MATLAB 'da, yüksek dereceli terimlerin katsayılarından başlayarak satır vektörü olarak girilirler.

Örneğin,

$P(s) = s^4 + 7s^2 + 4s^1 + 3$

polinomu, MATLAB' da şu şekilde temsil edilir.

```
>> P=[1 0 7 4 3]

P =
    1    0    7    4    3
```

2.3.1 Polinomlarla İşlemler

Bir polinomun kökleri roots komutuyla bulunabilir. Örneğin, yukarıdaki P polinomunun kökleri;

```
>> roots(P)

ans =

  -0.7131 + 2.2676i
  -0.7131 - 2.2676i
  -0.2869 + 0.6698i
  -0.2869 - 0.6698i
```

şeklinde bulunabilirler.

Bir polinomu, verilen köklerden oluşturmak mümkündür. Bunun için ise poly deyimi kullanılmaktadır. Mesela, kökleri l,0 ,1+i ve 2-i olan bir polinomu elde edelim.

```
>> k=[0 1 2+i 2-i -2]

k =

       0            1.0000            2.0000 + 1.0000i   2.0000 - 1.0000i  -2.0000

>> poly(k)

ans =

    1    -3    -1    13   -10     0
```

Cevaptan aşağıdaki polinom elde edilmiş olur;

$$P(s) = s^5 - 3s^4 - 1s^3 + 13s^2 - 10s$$

Beş kök verildiğine göre, polinomun beşinci dereceden olacağı açıktır.
Polinomun belirli değerler için hesaplanması mümkündür. Bu işlemi yapacak MATLAB deyimi ise polyval komutudur. Bu komutun kullanılışını göstermek maksadıyla, yukarıdaki P polinomunu "-1" değerinde hesaplayalım. **[2]**

```
>> polyval(k,-1)

ans =

  -3.0000 + 2.0000i
```

MATLAB'ın diğer bir kullanışlı deyimi ise polinomların türevlerim alan polyder komutudur.
MATLAB'da polinomlann çarpılması ve bölünmesi de mümkündür. Çarpmayı ve bölmeyi gerçekleştiren conv ve deconv komutlarının kullanılışları,

t=conv(a,b)
ve
[m,r]=deconv(a,b)

şeklindedir. Birincide, a ve b polinomları çarpılarak t elde edilirken, ikincide a polinomu b polinomuna bölünüp m bölümü ve r kalanı elde edilir.
Örneğin,

```
>> a=[4 5 2];b=[7 3 1];
>> t=conv(a,b)

t =

   28    47    33    11     2

>> [m,r]=deconv(a,b)

m =

   0.5714


r =

        0    3.2857    1.4286
```

2.4 MATLAB' da Grafik Çizme

MATLAB iki ve üç boyutlu verileri istenen formatta göstermeye yarayan bir dizi fonksiyon içermektedir. MATLAB'in fonksiyon çizimlerine mahsus zengin koleksiyonu, kullanıcılara bilimsel ve mühendislik uygulamalarına ait çizimleri kolayca ve etkin biçimde çizme imkanı sağlamaktadır. MATLAB'da bu amaçla yer alan komutların kısa açıklamaları için Tablo 2.2'ye bakınız.

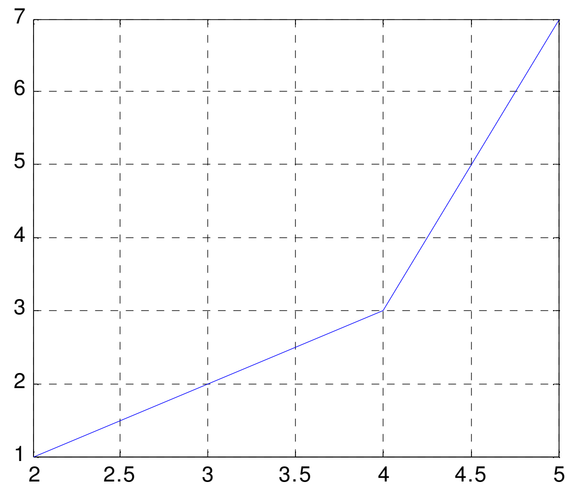
Tablo 2.2
MATLAB'daki çizim fonksiyonları **[2]**

Fonksiyon	**Açıklaması**	**Örnek**
plot	iki boyutlu çizim için temel komut	plot(xdata,ydata)
polar	Kutupsal (polar) koordinatlardaki çizimler	polar(teta,ro)
plot3	Uç boyutlu (3-D) çizimler	plot3(x,y,z)
title	Grafiğin üstüne başlık yazmak için	title('ilk grafik')
xlabel	x eksenine ait etiket	xlabel('xdata')
ylabel	y eksenine ait etiket	ylabel('ydata')
grid	Grafiği bölüntü ağlarıyla örer.	grid
subplot	Grafik penceresini bölmelere ayırır.	subplot(mnk) mxn tane şekil, ve k aktif olan şekil
text	Grafikte istenen yere bir metin yerleştirir.	text(x,y,'bir grafik') x ve y noktalarına "..."metnini yerleştirir.
gtext	Grafikte istenen noktaya bir metin yerleştirir.	gtext('ilk grafik')
ginput	Grafik üzerinde istenilen noktanın koordinatlarını belirtmede kullanılır	ginput
axis	x ve y eksenlerini yeniden ölçeklendirir	axis([Xmin , Xmax, Ymin ,Ymax])
legend	Birkaç grafik birden çizildiğinde farklı grafikleri etiketler	Legend ('isim1', 'isim2')
hold	Mevcut çizimi alıkor.	hold /hold on/ hold off
Figüre	Birden fazla grafik penceresi açar	figure(1), figure(2), vs

Örnek

```
>> a=[2 4 5];b=[1 3 7];
>> plot(a,b)
>> grid
```

Komut penceresine yukarıdaki komutlar yazılıp giriş tuşuna basıldığında figür 2.1 elde edilir.



Figür 2.1 plot (a,b) komutuna örnek

2.4.1 MATLAB Grafiklerinde Renk, Çizgi Türü ve Sembol Kullanımı

Birbiri üzerine farklı eğriler çizmenin karışıklığa yol açması mümkündür. Aynı şekilde yer alan farklı çizimleri ayırt edebilmenin basit bir yolu her eğri için farklı renkler veya farklı veri noktası sembolleri kullanmaktır. Bu imkanlar da aşağıdaki tabloda özetlenmiştir (Tablo 2.3-4). **[2]**

Tablo 2.3
Çizimde farklı renk alternatifleri

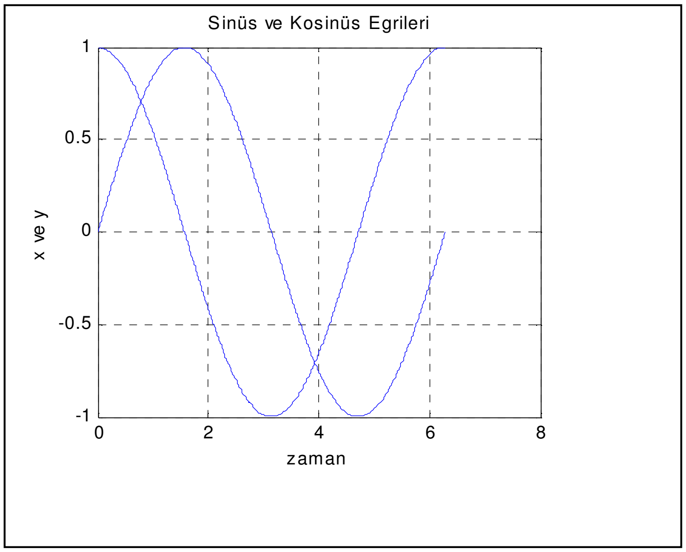
Renkler (Ingilizcesi)	Sembol
San (yellovv)	y
Eflatun (magenta)	m
Kırmızı (red)	r
Yeşil (green)	g
Mavi (blue)	b
Beyaz (white)	w
Siyah (black)	k

Tablo 2.4
Çizimde farklı çizgi alternatifleri.

Çizgi Tipi	Sembol
Düz	.
Kesikli	~
Noktalı	'.
Yıldız	*
Daire	o
Artı	+
Düz-Nokta	-.
Kare	s
Üçgenler	A, v, <, >

Renk ve çizgi tiplerinin kullanılması, çizimlerin farklı görünmesine ve birbirlerinden ayırt edilmesine yardımcı olacaktır.
Örnek :

```
>> t=[0:0.01:2*pi];              % (0,271) aralığında noktalar oluşturuldu

>> x=sin(t);y=cos(t);            % Sinüs ve Kosinüs'leri  hesaplandı

>> plot (t,x)                    % t ye göre x i çizer
>> grid                          % Bölüntü çizgilerini yerleştirir
>> xlabel ('zaman')              % Yatay eksene "zaman" etiketini koyar
>> hold                          % Çizimi alıkor yani yeni çizim mevcutun üzerine yapılır
Current plot held                % Çizim alıkonuldu
>> plot(t,y)                     % t ye göre y yi çizer
>> ylabel('x ve y')              % Düşey eksene "x ve y"  yazar
>> title ('Sinüs ve Kosinüs Egrileri')  % Figür başlığını yazar
```



Figür 2.2 Hold komutuyla 2 grafiğin aynı pencereye çizilmesi

MATLAB, kullanıcıya grafik üzerine yazı ekleme kolaylığı verdiği gibi değişik stil, sembol, üs ve alt indis kullanmak da mümkündür. Sembolleri elde etmek için, ters bölme işaretinden sonra sembolün İngilizce isminin yazılması yeterlidir. Bunlardan en çok kullanılanları aşağıda (Tablo 2.5) özetlenmiştir.

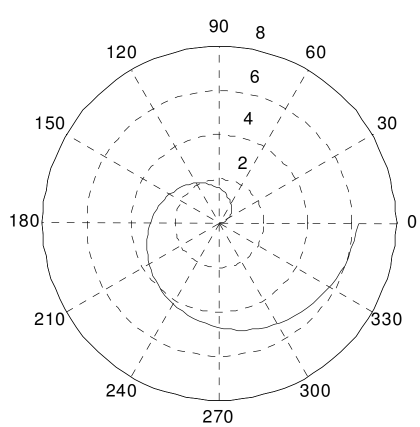
Tablo 2.5
MATLAB Grafiklerinde Sembol Kullanımı

Sembol	MATLAB Yazılımı
α	\alpha
β	\beta
δ	\delta
π	\pi
θ	\theta
ω	\omega

Örnek :

```
>> teta=linspace(0,2*pi,100);
>> c=1;
>> Ro=c*teta;
>> polar(teta,Ro,'k')
>> polar(teta,Ro,'k')
```

Yukarıdaki komut satırları yazıldığında Figür 2.3 elde edilir.

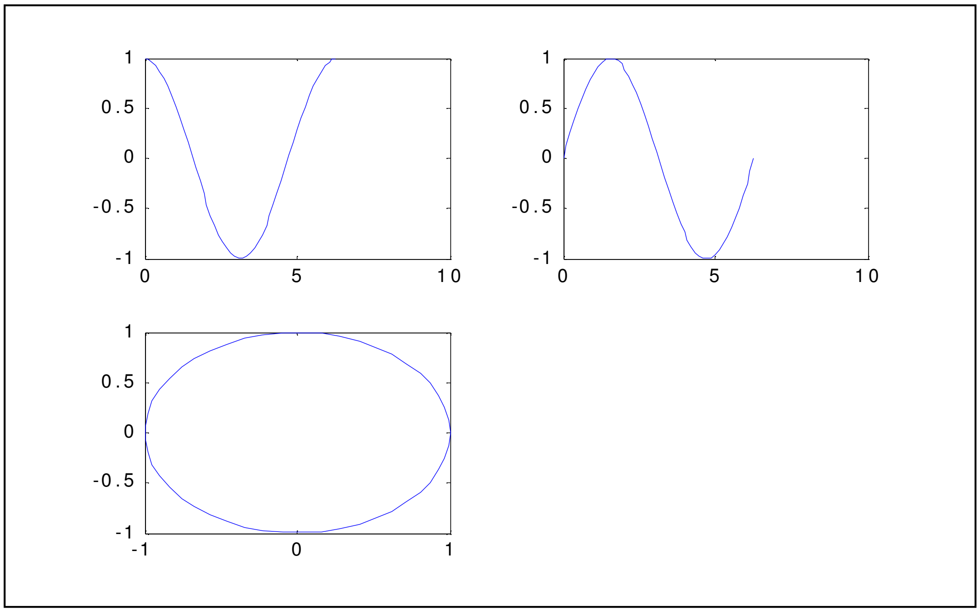


Figür 2.3 Polar koordinatlarda Arşimet spirali
Örnek :

```
>> y=cos(x);
```

```
>> z=sin(x);
>> subplot(2,2,1)
>> plot (x,y)
>> subplot(2,2,2)
>> plot (x,z)
>> subplot(2,2,3)
>> plot (y,z)
```

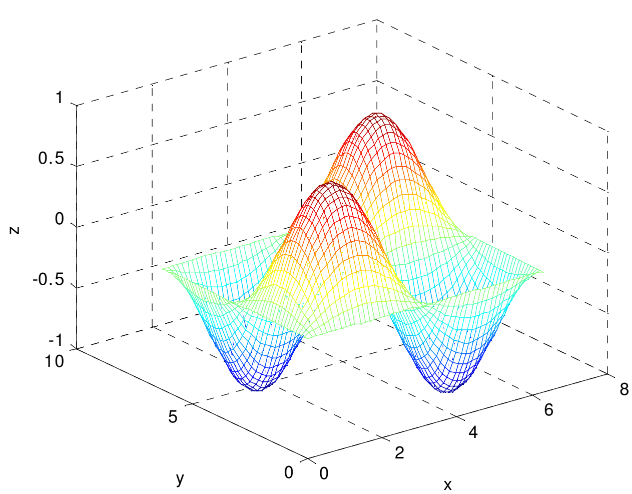
Komut satırları yazıldığında Figür 2.4 elde edilir.



Figür 2.4 Bir grafik penceresine subplot ( ) ile 3 grafiğin çizilmesi
Örnek :

```
>> x=linspace(0,2*pi,50);
>> y=linspace(0,2*pi,50);
>> [x,y]=meshgrid(x,y);
>> z=sin(x).*sin(y);          % Bu noktalarda fonksiyon hesaplandi
>> mesh(x,y,z)
>> xlabel('x'); ylabel('y'); zlabel('z');
```

Komut satırları yazıldığında Figür 2.5 elde edilir.



Figür 2.5 mesh( ) komutu ile 3 boyutlu grafik

2.5 Matrisler

Matrislere, elemanları iki indisle tanımlanan boyutlu diziler olarak bakılabilir. MATLAB' da bir matrisin elemanları, bir köşeli parantez içinde, her satırı noktalı virgüllerle ayırarak ve satır satır yazılarak girilebilir ve saklanabilir. Bir A matrisinin i'ninci satır ve j'ninci sütunundaki eleman A(i,j) şeklinde gösterilir. Mesela 3x3 lük (yani 3 satır, 3 sütun)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Şeklindeki bir A matrisi, MATLAB' da aşağıdaki gibi girilir.

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

A =

    1    2    3
    4    5    6
    7    8    9

>> A=[1 2 3              %  Yukarıdaki matris aşağıdaki komut ile de oluşturulabilir
4 5 6
7 8 9]

A =

    1    2    3
    4    5    6
    7    8    9


>> A(1,3)                        % 1.satır ve 3. sütundaki eleman

ans =

    3

>> A(:,1)                        %  ":" bütün elemanlar anlamına gelmektedir
```

```
                                % 1.sütundaki bütün elemanlar anlamına gelmektedir.
ans =

     1
     4
     7

>> A(3,:)                       % 3.satırdaki tüm elemanlar anlamına gelmektedir

ans =

     7     8     9

>> B=A([2,3],[1,2])             % Burada oluşacak yeni matris 2.satır 1.sütun ile
                                % 3.satır ve 2. sütun arasındaki elemanlardır

B =

     4     5
     7     8
```

Tablo 2.6
Matris oluşturan MATLAB hazır fonksiyonları

Fonksiyon	**Sembol**	**Örnek**
Birim matris	eye	eye(m,n)
Birler matrisi	ones	ones(m,n)
Sıfırlar matrisi	zeros	zeros(m,n)
Rasgele sayılar matrisi	rand	rand(m,n)

Aşağıdaki pencerede bazı örnekler verilmiştir.

```
>> eye(3,3)

ans =

     1     0     0
     0     1     0
     0     0     1

>> rand(3,4)

ans =

    0.4565    0.4447    0.9218    0.4057
```

```
   0.0185    0.6154    0.7382    0.9355
   0.8214    0.7919    0.1763    0.9169

>> zeros(2,4)

ans =

     0     0     0     0
     0     0     0     0
```

2.5.1 Matrislerle İşlemler

MATLAB’ da matris işlemleri oldukça kolaylaştırılmıştır. Temel matris işlemlerinden bazıları aşağıda Tablo 2.7’ de verilmiştir.

Tablo 2.7
MATLAB’ da temel matris işlemleri.

İşlem	**Sembol**	**Örnek**
Toplam	+	A+B
Çıkarma	-	A-B
Çarpma	*	A*B
Eleman-elemana çarpma	.*	A.*B
Sağdan bölme	/	A/B (=A*B $^{-1}$)
Eleman-elemana bölme	./	A./B(=A.*B $^{-1}$)
Soldan bölme	\	A\B (=A $^{-1}$ *B)
Transpozunu alma	‘	A’
Bir matrisin determinantı	det	det(A)
Bir matrisin tersi	inv	inv(A)
Bir matrisin özdeğeri ve özvektörleri	eig	eig(A) [v,d]=eig(A)

Matrislerde yapılan işlemlerin anlamlı olabilmesi ve bir sonuç verebilmesi için yapılan işlemlerdeki matris boyutlarına dikkat edilmelidir.

Aşağıda matris işlemlerine dair bazı örnekler verilmektedir. **[9]**

```
>> A=[5 2;4 -1]; B= [0 1;7 2];
>> A+B

ans =

     5     3
    11     1

>> A*B

ans =

    14     9
```

```
   -7     2

>> A.*B                  % A ve B matrisleri eleman-elemana çarpılıyor.

ans =

    0     2
   28    -2

>> A*B'                    % A matrisi ile B matrisinin transpozu çarpılıyor

ans =

    2    39
   -1    26

>> A^2                     %A matrisinin karesi alınıyor. Yani A*A
ans =

   33     8
   16     9

>> A.^2          % A matrisinin her bir elemanının(eleman-elemana karesi alınıyor)

ans =

   25     4
   16     1

>> A/B                                   % A*B^-1
```

```
ans =

   0.5714    0.7143
  -2.1429    0.5714

>> A./B                                  % A.*B^-1
Warning: Divide by zero.

ans =

      Inf    2.0000
   0.5714   -0.5000

>> A\B                                   %  A^-1 *B
ans =
   1.0769    0.3846
  -2.6923   -0.4615

>> det(A)
```

```
ans =

  -13

>> inv(A)
ans =

    0.0769    0.1538
    0.3077   -0.3846
```

2.6 M-Dosyaları (M-Files)

Basit problemler için işlemlerin MATLAB komut penceresinde hem hızlı hem de etkilidir. Fakat, bir işlem yapmak için gerekli komutların sayısı arttıkça veya değişkenlerden bazılarını değiştirip komutları tekrarlamak gerekiyorsa, işlemlerin komut penceresinde yürütülmesi pratik olmayacaktır. Etkin bir çözüm, MATLAB' ın kullanıcıya sunduğu M-dosyalarının kullanılmasıdır.

Bir M-Dosyası özel bir görevi yerine getirmek için gerekli MATLAB komutlarının saklandığı bir metin programıdır. Başka bir ifadeyle, komutla dizisi bir dosyada saklanır ve daha sonra bunları bir komut penceresinden tek tek girmek yerine, bu dosya çalıştırılarak komutlar icra edilir. Bu dosyaların MATLAB' ın çalıştığı dizinde .m uzantısıyla saklanması gerekir. MATLAB, M-Dosyalarının oluşturulması ve yazılması için bir metin hazırlayıcısı (text editor) sunmaktadır. M-Dosyaları farklı bir metin programında da yazılabilirler (Notepad,Word vb). MATLAB' in metin hazırlayıcısıı, ya komut penceresinin üst kısmında yer alan 'New M-File' düğmesi tıklanarak (Şekil 1.1) yada 'File' menüsünden 'New M-File' ibaresini seçerek etkin hale getirilebilir. **[2]**

BÖLÜM 3

3 MATLAB' DA DENKLEMLERİN ÇÖZÜMLENMESİ

3.1 Doğrusal Denklemlerin Çözümlenmesi

Tek değişkenli bir doğrusal sistemin denklemi, n. Dereceden bir polinom biçiminde;

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots.. + a_2 x^2 + a_1 x^1 + a_0$$

tanımlanır. Burada $f(x) = 0$ biçiminde denklemin köklerini bulmak için MATLAB' in 'roots' fonksiyonu kullanılır. **[2]**

Örnek 1: Aşağıda verilen 6. dereceden denklemin kökünü bulunuz.

$$f(x) = 4x^6 + 7x^5 + 5x^4 - 3x^3 + x^2 - 9x + 12$$

Çözüm : Denklemin katsayıları a değişkenine bir vektör olarak atanır ve 'roots' komutu kullanılarak denklemin kökleri bulunur. Veya 'roots' fonksiyonunun içine vektör direkt olarak girilerek de çözüme gidilebilir.

```
>> a=[4 7 5 -3 1 -9 12]

a =

    4    7    5   -3    1   -9   12

>> kokler=roots(a)

kokler =

  -1.4258 + 0.9338i          % 1. kök
  -1.4258 - 0.9338i          % 2. kök
  -0.2109 + 1.1561i          % 3. kök
  -0.2109 - 1.1561i          % 4. kök
   0.7617 + 0.4094i          % 5. kök
   0.7617 - 0.4094i          % 6. kök
```

3.2 Doğrusal Olmayan Denklemlerin Çözümlenmesi

MATLAB' da doğrusal olmayan denklemlerin kökleri 'fzero' yada 'fsolve' fonksiyonlarıyla hesaplanabilir. Her iki komutun kullanılması içinde bir m-file oluşturmak gerekmektedir.

'fzero' fonksiyonu, tek değişkenli bir denklemin kökünü araştırır. 'fsolve' fonksiyonu bir ve birden fazla değişkenli denklemlerin köklerini araştırır. Bu fonksiyonların kullanım biçimi aşağıda olduğu gibidir. **[1]**

x = fzero(fun,x0)

x = fzero(fun,x0,options)

x = fsolve(fun,x0)
x = fsolve(fun,x0,options)

Yukarıdaki fonksiyonlarda yer alan ‘fun’ kökü araştırılacak denklemin yer aldığı fonksiyon ismidir. ‘x0’ terimi MATLAB’ in ,denklemin kökünü araştırmaya başlayacağı başlangıç noktasını ifade eder. ’option’ terimi MATLAB’ in denklemi çözerken yapmış olduğu işlemlerin komut penceresinde gözükmesi işini düzenler.

‘options‘ teriminin kullanımı şu şekildedir;

options=optimset('Display','iter')

‘Display’, ‘iter’ in tanımlanma şekliyle sonuçların komut penceresine yazılmasını sağlar. Burada ‘iter’ in alabileceği dört durum vardır. ‘iter’ durumunda denklemin çözümü esnasındaki bütün deneme sonuçları komut penceresine yazılır. ‘off’ durumunda çözüm esnasındaki hiçbir sonuç komut penceresinde gösterilmez. ‘final’ durumunda çözüm esnasında denklemin çözümünün bulunduğu aralık komut penceresine yazılır. ‘fzero’ komutunun sonucu tam olarak bulamadığı zamanlar olmaktadır. Bu durum komut penceresinde bir uyarıyla kullanıcıya ifade edilmektedir. ‘notify’ durumunda komut penceresinde herhangi bir sonuç yazılmaz. Sadece, eğer denklem tam olarak çözülmemiş ise uyarı komut penceresinde görünür.

Örnek 2 : Aşağıda verilen 6. dereceden denklemin kökünü bulunuz. **[1]**

$$f(x) = e^{3x} + 2x$$

Çözüm:

Çözümün ‘fzero’ ile gerçekleştirilmesi aşağıda olduğu gibidir.

Öncelikle ‘fzero’ fonksiyonunu kullanabilmek için bu denklemi aşağıda gösterildiği gibi bir ‘m-file’ olarak yazmamız gerekmektedir.

MATLAB>File>New>M-file seçilir ve açılan pencereye aşağıdaki komutları yazılır ve saklanır.

```
function y=denklem(x)
y=exp(3*x)+2*x
```

MATAB komut penceresinde ‘option’ değerleri girilir. Burada denklem çözülürken MATLAB’ in yaptığı bütün işlemleri girmek için ‘iter’ girilir. ‘fzero’ fonksiyonu yazılıp işletildiğinde sonuç aşağıda görüldüğü gibidir.

```
>> options=optimset('Display','inter');
>> z=fzero('denklem',2,options)
Func-count    x          f(x)       Procedure
   1          2      407.429      initial
```

```
  2      1.94343     344.346     search
  3      2.05657     482.158     search
  4         1.92     321.188     search
  5         2.08     517.019     search
  6      1.88686     291.091     search
  7      2.11314     570.689     search
  8         1.84     253.315     search
  9         2.16     656.291     search
 10      1.77373     208.172     search
 11      2.22627     799.835     search
 12         1.68      157.83     search
 13         2.32     1058.27     search
 14      1.54745     106.883     search
 15      2.45255     1573.04     search
 16         1.36     61.8655     search
 17         2.64     2757.05     search
 18       1.0949      28.891     search
 19       2.9051     6101.21     search
 20         0.72     10.1111     search
 21         3.28     18776.3     search
 22     0.189807     2.14685     search
 23      3.81019       92103     search
 24        -0.56   -0.933626     search

 Looking for a zero in the interval [-0.56, 3.8102]

 25    -0.559956       -0.933513      interpolation
 26    -0.195186        0.166422      interpolation
 27    -0.250377      -0.0289199      interpolation
 28    -0.242206    -0.000869663      interpolation
 29    -0.241954     6.13983e-008     interpolation
 30    -0.241954     -9.7598e-012     interpolation
 31    -0.241954               0      interpolation
Zero found in the interval: [-0.56, 3.8102].

z =

 -0.2420
```

Yukarıda görülen Func-count MATLAB' in fonksiyonu kaç kere işlettiğini gösteren bilgidir. Görüldüğü gibi sonuca 31 denemeden sonra gidilmiştir. 'x' değişkeni MATLAB' in denklemin köklerini bulabilmesi için vermiş olduğu değerleri temsil eder. 'f(x), verilen 'x' değerleri karşılığında denklemin aldığı değerlerdir. Denklemin kökü 'z' değişkenine atanır.

Örnek 3 : Aşağıda verilen 6. dereceden denklemin kökünü bulunuz

$$f(x) = 7v^{-1}d^{0.8} + 3v^{2}d^{1.2}$$

Çözüm:

Öncelikle ‘fzero’ fonksiyonunu kullanabilmek için bu denklemi aşağıda gösterildiği gibi bir ‘m-file’ olarak yazmamız gerekmektedir.

MATLAB>File>New>M-file seçilir ve açılan pencereye aşağıdaki komutları yazılır ve saklanır.

```
function y=denklem(x)

y=7*x(1)^(-1)*x(2)^(0.8)+3*x(1)^(2)*x(2)^(1.2);
```

MATAB komut penceresinde ‘option’ değerleri girilir. Burada denklem çözülürken MATLAB’ in yaptığı bütün işlemleri girmek için ‘iter’ girilir. ‘fsolve’ fonksiyonu yazılıp işletildiğinde sonuç aşağıda görüldüğü gibidir.

```
options=optimset('Display','iter');
>> z=fsolve('denklem',[1 1],options)
Warning: Large-scale method requires at least as many equations as variables;
   switching to line-search method instead.

> In C:\MATLABR12\toolbox\optim\fsolve.m at line 194

                                                Directional
 Iteration  Func-count     Residual     Step-size    derivative
     1          2              100            1          -198
     2          9          20.8598         1.44          75.5
     3         15          20.8598       1e-008         -29.3
     4         29          18.4224        0.147         -4.68
     5         35          12.2777        0.306         -2.77
     6         41          3.53197        0.313         -28.5
     7         47          1.03833        0.279         -10.3
     8         53         0.157957        0.325         -2.87
     9         59        0.0795964        0.204         -0.45
    10         66        0.0450412        0.514         0.819
    11         72       0.00411478        0.347        -0.113
    12         78       0.00102741        0.291       -0.0121
    13         84       0.00100129       0.0126      -0.00211
    14         90       0.00024066        0.294      -0.00294
    15         96      0.000224521       0.0324     -0.000513
    16        103      0.000185047       0.0811     -0.000526
    17        109      1.84012e-006       0.388     -0.000235
    18        115      2.94422e-007       0.456     9.62e-006
    19        121      1.28724e-008       0.456     5.94e-007
    20        127      2.78181e-010       0.282    -1.06e-008
    21        133       4.4441e-012       0.196     1.33e-010
    22        139      2.79134e-012        0.12     1.85e-011
    23        145      7.86308e-013       0.107      7.3e-012
```

```
   24         151      3.64839e-013       0.0813    2.33e-012
   25         157      1.21263e-013       0.0778    9.72e-013
   26         163      5.12696e-014       0.0663    3.57e-013
   27         169      1.82399e-014       0.0611    1.41e-013
   28         175      7.30187e-015       0.0534    5.21e-014
   29         181      2.68217e-015       0.0486    1.98e-014
   30         187      1.04119e-015       0.0433    7.17e-015
   31         193      3.85597e-016       0.0398    2.57e-015
   32         199      1.45898e-016       0.0367    8.73e-016
Maximum number of function evaluations exceeded
Increase OPTIONS.maxFunEvals

z =

   0.0012 - 0.0000i  -0.0000 + 0.0000i
```

3.3 Diferansiyel Denklemlerin Çözümlenmesi

MATLAB diferansiyel denklemlerin çözümü ‘dsolve’ fonksiyonu ile hesaplanabilir.Bu fonksiyonun diğerlerinden farkı , denklemlerin fonksiyon içinde bir eşitlik şeklinde yazılmasıdır, yani ‘=’ işaretinin kullanılmasıdır. Diferansiyel denklemlerle çalıştığımız için eşitlikler içinde türevlerin ifade edilmesi gerekir. Birinci, ikinci, üçüncü, vs türevler, sırasıyla D, D2, D3... şeklinde ifade edilir. ‘dsolve’ fonksiyonu birden fazla diferansiyel denklemin çözümünde kullanılabilir. **[2]**

Örnek 4 : Aşağıda verilen 2. dereceden diferansiyel denklemi çözünüz.

$$\frac{d^2x}{dt^2} + \frac{dx}{dt} + 2x = 0$$

Çözüm :

```
>> dsolve('D2x+3*Dx+2*x=0')

ans =

C1*exp(-2*t)+C2*exp(-t)
```

$x(t) = C_1 e^{-2t} + C_2 e^{-t}$ diferansiyel denklemin çözümü olarak bulundu. Buradaki C_1 ve C_2 integral sabitlerini gösterir. Bu denklemi başlangıç şartlarıyla çözdürmekte mümkündür. Başlangıç şartlarını $\dot{x}(0) = 0$ ve $x(0) = 2$ olarak aldığımızda C_1 ve C_2 sabitlerinin değerleri başlangıç şartlarına göre bulunur.

```
>> dsolve('D2x+3*Dx+2*x=0','Dx(0)=0','x(0)=2')
```

```
ans =

-2*exp(-2*t)+4*exp(-t)
```

Örnek 5 : Aşağıda verilen diferansiyel denklemleri çözünüz.

$$\frac{dx}{dt} = 3x + 4y \text{ ve } \frac{dy}{dt} = -4x + 3y$$

Çözüm :

```
>> [x,y]=dsolve('Dx=3*x+4*y,Dy=-4*x+3*y','Dx(0)=0','Dy(0)=-1')

x=

exp(3*t)*(cos(4*t)*C1+sin(4*t)*C2)


y =

-exp(3*t)*(sin(4*t)*C1-cos(4*t)*C2)
```

$x(0) = 0$ ve $y(0) = -1$ olarak aldığımızda C_1 ve C_2 sabitlerinin değerleri bulunur.

```
>> [x,y]=dsolve('Dx=3*x+4*y,Dy=-4*x+3*y','x(0)=0','y(0)=-1')

x =

-exp(3*t)*sin(4*t)


y =

-exp(3*t)*cos(4*t)
```

BÖLÜM 4

4 MATLAB' DA SİSTEMLERİN MODELLENMESİ , MODELLERİN VERİLERİNE ERİŞİM VE MODELLERİN BİRBİRİNE DÖNÜŞTÜRÜLMESİ

4.1 MATLAB' da Sistemlerin Modellenmesi

MATLAB' in kütüphanelerinden biri olan The Control System Toolbox doğrusal, zamanla değişmeyen (MATLAB karşılığı, LTI- Linear Time Invariant) sistemleri modellemek için dört fonksiyona sahiptir. Bunlar 'tf', 'zpk', 'ss' ve 'frd' fonksiyonlarıdır. Sırasıyla; transfer fonksiyonu, sıfır-kutup-kazanç, durum denklemleri ve frekans cevabı verileri modellemelerinde kullanılırlar. **[1]**

4.1.1 Transfer Fonksiyonu Modeli

LTI sistemlerinin transfer fonksiyonu modelini oluşturmak için 'tf' fonksiyonu kullanılmaktadır. Aynı zamanda, bu fonksiyon ile durum denklemi ve sıfır-kutup-kazanç modellerini transfer fonksiyonuna çevirmekte mümkündür. Tek girişli tek çıkışlı (MATLAB karşılığı, SISO Single Input Single Output) bir sistem bir pay vektörü ve bir payda vektöründen oluşur (num/den). Çok girişli çok çıkışlı (MIMO Multiple Input Multiple Output) sistemlerde ise pay ve payda vektör dizilerinden oluşur. Yine bir giriş çıkışlı sistemi bir pay ve payda vektörü temsil eder. MATLAB' da 'tf' fonksiyonunun kullanımı aşağıda gösterildiği biçimdedir. **[1]**

```
sys = tf(num,den)
sys = tf(num,den,'Property1',Value1,...,'PropertyN',ValueN)
```

'num' pay vektörü, 'den' pyda vektörü ve 'Ts' örnekleme zamanını temsil eder.

Örnek 1 : Transfer fonksiyonu

$$G(s) = \frac{s}{s^2 + 2s + 10}$$

olarak verilen sistemin MATLAB' da transfer fonksiyonu modelini oluşturun.

Çözüm :

```
>> G=tf([1 0],[1 2 10])

Transfer function:
      s
--------------
s^2 + 2 s + 10
```

'tf' fonksiyonu içinde bulunan iki köşeli parantez çiftinden birincisi içinde transfer fonksiyonunun payı, ikincisi içinde ise paydası vardır. Pay ve payda derecesi yüksek olandan düşük olana doğru katsayıları temsil etmektedir.

Örnek 2 : İki çıkışlı bir girişli bir sitem oluşturun. Giriş akımı , çıkışlar torku ve açısal hızı temsil etsin. Sistem değişkeni ise ‘p’ olsun.

Çözüm :

```
>> num = {[1 1] ; 1}

num =

    [1x2 double]
    [         1]

>> den = {[1 2 2] ; [1 0]}

den =

    [1x3 double]
    [1x2 double]

>> H = tf(num,den,'inputn','current',...
'outputn',{'torque' 'ang. velocity'},...
'variable','p')

Transfer function from input "current" to output...
                 p + 1
 torque:  -------------
             p^2 + 2 p + 2

                         1
 ang. velocity:   ---
                         p
```

Örnek 3 : Durum denklem formu aşağıda verilen sistemin transfer fonksiyonu modelini oluşturunuz.

$$A = \begin{bmatrix} -2 & -1 \\ 1 & -2 \end{bmatrix} \;,\; B = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix} \;,\; C = \begin{bmatrix} 1 & 0 \end{bmatrix} \;,\; D = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Çözüm :

```
>> sys = ss([-2 -1;1 -2],[1 1;2 -1],[1 0],[0 1])

a =
               x1       x2
       x1      -2       -1
       x2       1       -2

 b =
               u1       u2
       x1       1        1
       x2       2       -1

c =
               x1       x2
       y1       1        0


d =
               u1       u2
       y1       0        1

Continuous-time model.
>> tf(sys)

Transfer function from input 1 to output:
s - 2.963e-016
--------------
s^2 + 4 s + 5

Transfer function from input 2 to output:
s^2 + 5 s + 8
-------------
s^2 + 4 s + 5
```

4.1.2 Sıfır-Kutup-Kazanç Modeli

LTI sistemlerinin sıfır-kutup-kazanç modelini oluşturmak için 'zpk' fonksiyonu kullanılmaktadır. Aynı zamanda, bu fonksiyon ile durum denklemi ve transfer fonksiyonu modellerini sıfır-kutup-kazanç modeline çevirmekte mümkündür. 'zpk' fonksiyonunun kullanımı aşağıda gösterildiği biçimdedir. **[1]**

sys = zpk(z,p,k)

Örnek 4 : z=1' de sıfırı ve $p_{1,2} = 2 \pm i$, $p_3 = 1$ noktalarında kutbu ve -0.5 kazancı olan bir sistemin sıfır-kutup-kazanç modelini oluşturunuz.

Çözüm :

```
>> z=[1]

z =
     1

>> p=[2-i 2+i 1]

p =

   2.0000 - 1.0000i   2.0000 + 1.0000i   1.0000

>> k=-1/2

k =

   -0.5000

>> H = zpk(z,p,k)

Zero/pole/gain:
     -0.5 (s-1)
--------------------
(s-1) (s^2 - 4s + 5)
```

Örnek 5 : Aşağıda verilen iki-giriş/iki-çıkış sistemin sıfır-kutup-kazanç modelini oluşturunuz.

$$G(s) = \begin{bmatrix} \dfrac{-1}{s} & \dfrac{3(s+5)}{(s+1)^2} \\ \dfrac{2(s^2-2s+2)}{(s-1)(s-2)(s-3)} & 0 \end{bmatrix}$$

Çözüm : Önce $(s^2 - 2s + 2)$ denkleminin çözümü yapılır. Daha sonra z, p ve k değişken atamaları yapılır ve zpk fonksiyonu ile modelleme yapılır.

```
>> roots([1 -2 2])
ans =

   1.0000 + 1.0000i
   1.0000 - 1.0000i

>> a= roots([1 -2 2])

a =
```

```
   1.0000 + 1.0000i
   1.0000 - 1.0000i

>> z={[],-5;a,[]}

z =

             []    [-5]
    [2x1 double]      []

>> p={0,[-1 -1];[1 2 3],[]}

p =

           [         0]     [1x2 double]
    [1x3 double]                      []

>> k=[-1 3;2 0]

k =

    -1     3
     2     0

>> H=zpk(z,p,k)
 Zero/pole/gain from input 1 to output...
      -1
 #1:  --
       s

       2 (s^2 - 2s + 2)
 #2:  -----------------
       (s-1) (s-2) (s-3)

Zero/pole/gain from input 2 to output...
      3 (s+5)
 #1:  -------
      (s+1)^2

 #2:  0
```

Örnek 6 : Aşağıda transfer fonksiyonu formu verilen sistemi sıfır-kutup-kazanç modeline dönüştürün.

$$G(s)=\frac{-10s^2+20s}{s^5+7s^4+20s^3+28s^2+19s+5}$$

Çözüm :

```
>> h = tf([-10 20 0],[1 7 20 28 19 5])

Transfer function:
           -10 s^2 + 20 s
----------------------------------------
s^5 + 7 s^4 + 20 s^3 + 28 s^2 + 19 s + 5

>> zpk(h)

Zero/pole/gain:
    -10 s (s-2)
----------------------
(s+1)^3 (s^2 + 4s + 5)
```

4.1.3 Durum Denklemi Modeli

Genel olarak sistemlerin durum denklemi aşağıda görüldüğü şekilde verilir. **[1]**

$$\frac{d}{dt}X(t) = AX(t) + Bu(t)$$

$$y(t) = CX(t) + Du(t)$$

X durum vektörü
u sistem girişi
y çıkış vektörü
A nxn elemanlı matris
B nxr elemanlı matris
C mxn elemanlı matris
D mxr elemanlı matris

LTI sistemlerinin durum denklemi modelini oluşturmak için 'ss' fonksiyonu kullanılmaktadır. Aynı zamanda, bu fonksiyon ile transfer fonksiyonu ve sıfır-kutup-kazanç modellerini durum denklemi modeline çevirmekte mümkündür. 'ss' fonksiyonunun kullanımı aşağıda gösterildiği biçimdedir. Fonksiyonda kullanılan a, b, c, d değişkenleri yukarıda gösterilen A, B, C, D değişkenlerine karşılık gelir.

sys = ss(a,b,c,d)

Örnek 7 : Aşağıda denklemi matrisleri aşağıda verilen sistemin modelini oluşturunuz.

$$A = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix} \ , \quad B = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \ , \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix} \ , \quad D = 0$$

Çözüm :

```
>> sys=ss([0 1;-5 -2],[0;3],[0 1],0)
```

```
a =
                  x1        x2
        x1         0         1
        x2        -5        -2


b =
                  u1
        x1         0
        x2         3


c =
                  x1        x2
        y1         0         1


d =
                  u1
        y1         0

Continuous-time model.
>> size(sys)
State-space model with 1 output, 1 input, and 2 states.
```

'ss' fonksiyonundan sonra kullanılan 'size' fonksiyonu durum denkleminin giriş, çıkış ve durum sayısını bildirir.

4.1.4 Tanımlayıcı Durum Denklemi Modeli

Durum denklemi modellerinin genelleştirilmiş biçimlerine tanımlayıcı durum denklemleri denir. Bu denklemlerin biçimi aşağıda olduğu gibidir. **[3]**

$$E\frac{dx}{dt} = Ax + Bu$$

$$y = Cx + Du$$

Buradan uygun bir çözüm elde edilebilmesi için E matrisinin tekil olmaması gerekir. Bu tür modelin eşdeğeri ise

$$\frac{dx}{dt} = (E^{-1}A)x + (E^{-1}B)u$$

$$y = Cx + Du$$

şeklindedir.

Bu tür bir sistem modeli oluşturmak için kullanılan fonksiyon ‘dss’ dir. Fonksiyonunun kullanımı aşağıda gösterildiği biçimdedir.

sys = dss(a,b,c,d,e)

Örnek 8 : MATLAB’ da bir tanımlayıcı durum denklemi modeli oluşturunuz.

Çözüm : A, B, C, D ve E durum denklemi değişkenleri girilir ve fonksiyon yazılır. Sonuç aşağıda olduğu gibidir.

```
>> A=[0 1;-5 -2]

A =
    0    1
   -5   -2

>> B=[0;3]

B =
    0
    3

>> C=[0 1]

C =
    0    1

>> D=0

D =
    0

>> E=[1 2;3 4]

E =

    1    2
    3    4

>> sys = dss(A,B,C,D,E)

a =
                x1        x2
        x1       0         1
        x2      -5        -2



b =
               u1
```

```
        x1      0
        x2      3


c =
              x1      x2
        y1     0       1

d =
              u1
        y1     0

e =
              x1      x2
        x1     1       2
        x2     3       4
 Continuous-time model.
```

4.1.5 Frekans Cevabı Verileri Modeli

LTI sistemlerinin frekans cevabı verileri modelini oluşturmak için 'frd' fonksiyonu kullanılmaktadır. Bir sistemde deney yolu ile elde edilen frekans vektörü ve frekanslara karşılık gelen cevap vektörü fonksiyonda gerekli yerlere yazıldığında sistemin frekans cevabı verileri modelini MATLAB' da elde etmiş oluruz. Fonksiyonunun kullanımı aşağıda gösterildiği biçimdedir. **[3]**

sys = frd(response,frequency,'Units',Hz')

Örnek 9 : 1000, 2000 ve 3000 Hz frekans değerlerindeki cevabı -0.81126-0.0003i, -0.1751-0.0016i ve -0.0926-0.4630i olan sistemin frekans verileri cevabını elde ediniz.

Çözüm : İlk olarak frekans ve cevap (response) vektörleri oluşturulur

```
>> freq=[1000;2000;3000];
>> resp=[-0.81126-0.0003i;-0.1751-0.0016i;-0.0926-0.4630i];
>> H=frd(resp,freq,'Units','Hz')

From input 1 to:

  Frequency(Hz)     output 1
  -------------     --------
      1000      -0.81126-0.0003i
      2000      -0.17510-0.0016i
      3000      -0.09260-0.4630i


Continuous-time frequency response data model.
```

4.1.6 Ayrık Zaman Modeli

Bir sistemin ayrık zaman modelini oluşturmak için, transfer fonksiyonu, sıfır-kutup-kazanç, durum denklemleri ve frekans cevabı verileri modellerinde kullanılan fonksiyonlara örnekleme zamanı bilgisi girilmesi yeterlidir. **[3]**

```
sys1 = tf(num,den,Ts)
sys2 = zpk(z,p,k,Ts)
sys3 = ss(a,b,c,d,Ts)
sys4 = frd(response,frequency,Ts)
```

Örnek 10 : 'tf ' fonksiyonu ile bir ayrık zaman modeli oluşturunuz.

Çözüm : 'tf' fonksiyonuna örnekleme zamanı bilgisini girerek ayrık zaman modelini elde ederiz.

```
>> h = tf([1 -0.2],[1 0.3],0.1)
Transfer function:
z - 0.2
-------
z + 0.3
Sampling time: 0.1
```

4.2 Modellerin Verilerine Erişim

Transfer fonksiyonu, sıfır-kutup-kazanç, durum denklemleri, tanımlı durum denklemleri ve frekans cevabı verileri modellerinin verilerine erişmek için sırasıyla aşağıdaki fonksiyonlar kullanılır. MATLAB ortamında mevcut bir sistem olduğunda ve bu sistemin verileri gerektiğinde bu komutlar rahatlıkla kullanılabilir ve istenilen bilgilere ulaşılabilir. **[3]**

```
[num,den,Ts] = tfdata(sys,'v')

[z,p,k,Ts] = zpkdata(sys,'v')

[a,b,c,d,Ts] = ssdata(sys,'v')

[a,b,c,d,e,Ts] = dssdata(sys,'v')

[response,frequency,Ts] = frdata(sysfr,'v')
```

Örnek 11 : Aşağıda verilen transfer fonksiyonu modelinin pay ve payda vektörlerini MATLAB ortamında elde ediniz.

$$T(s) = \frac{s+3}{s^2+2s+3}$$

Çözüm :

```
>> sys = tf([1 3],[1 2 5])

Transfer function:
   s + 3
-------------
s^2 + 2 s + 5
```

```
>> [num,den,Ts] = tfdata(sys,'v')

num =

     0     1     3

den =

     1     2     5

Ts =

     0
```

4.3 Modellerin Birbirine Dönüştürülmesi

Tablo 4.1 Sistemlerin birbirine dönüştürülmesinde kullanılan fonksiyonlar **[3]**

	Transfer Fonksiyonu	Durum Denklemi	Sıfır-Kutup-Kazanç
Transfer Fonksiyonu		tf2ss	tf2zp
Durum Denklemi	ss2tf		Ss2zp
Sıfır-Kutup-Kazanç	zp2tf poly	zp2ss	

4.3.1 Transfer fonksiyonunu Durum Denklemine Çevirme

'tf2ss' fonksiyonu transfer fonksiyonunu durum denklemine çevirir. Aşağıdaki biçimde kullanılır.

[A,B,C,D] = tf2ss(a,b)

a ve b değişkenleri transfer fonksiyonunun pay ve payda vektörleridir.
A, B, C ve D elde edilecek durum denklemi değişkenleridir.

Örnek 12 :

```
     2 s + 3
--------------
s^2 + 0.4 s + 1
```

verilen transfer fonksiyonunu durum denklemine çeviriniz.

Çözüm :

```
>> a = [2 3];
>> b = [1 0.4 1];
>> [A,B,C,D] = tf2ss(a,b)

A =

   -0.4000   -1.0000
    1.0000         0

B =

     1
     0

C =

     2     3

D =

     0
```

4.3.2 Durum Denklemini Transfer fonksiyonuna Çevirme

‘ss2tf’ fonksiyonu, durum denklemini transfer fonksiyonuna Çevirir. Aşağıdaki biçimde kullanılır.

ss2tf(A,B,C,D,iu)

‘iu’ değişkeni sistemin giriş sayısını verir. Değer verilmediğinde 1 kabul edilir.

Örnek 13 : Yukarıdaki örnekte durum denklemi değişkenleri elde edilen sistemin yeniden transfer fonksiyonu değişkenlerini elde edelim.

Çözüm :

```
>> [a,b] = ss2tf(A,B,C,D)

a =

         0    2.0000    3.0000


b =
```

```
   1.0000    0.4000    1.0000

>> tf(a,b)                              % elde edilen verilerle T.F(Transfer Fonksiyonu)' nin oluşturulması

Transfer function:
   2 s + 3
---------------
s^2 + 0.4 s + 1
```

4.3.3 Transfer fonksiyonunu Sıfır-Kutup-Kazanç Modeline Çevirme

'tf2zp' fonksiyonu transfer fonksiyonunun sıfırlarını, kutuplarını ve kazancını bulur. Aşağıdaki biçimde kullanılır.

[z,p,k] = tf2zp(a,b)

z : sıfırlar matrisi
p : kutup vektörü
k : kazanç

Örnek 14 : Bir önceki örnekte elde edilen transfer fonksiyonu verileri ile bu sistemin sıfır-kutup-kazanç modelini oluşturun.

Çözüm :

```
>> [a,b] = eqtflength(a,b)

a =

     2     3     0


b =

    1.0000    0.4000    1.0000
>> [z,p,k] = tf2zp(a,b)

z =

   -1.5000


p =

  -0.2000 + 0.9798i
  -0.2000 - 0.9798i
```

```
k =

    2
```

Yukarıdaki örnekte kullanılan 'eqtflength' fonksiyonu transfer fonksiyonunun pay ve payda vektörlerinin boyutlarını denk hale getirir.

4.3.4 Sıfır-Kutup-Kazanç Modelini Transfer Fonksiyonuna Çevirme

'zp2tf' fonksiyonu sıfır-kutup-kazanç modelinin transfer fonksiyonuna ait pay ve payda vektörlerini bulur. Aşağıdaki biçimde kullanılır.

[a,b] = zp2tf(z,p,k)

Örnek 15 : Bir önceki örnekte bulunan z,p,k verileri ile sistemin transfer fonksiyonunu bulunuz.

Çözüm :

```
>> [a,b] = zp2tf(z,p,k)

a =

    0    2    3
b =

   1.0000    0.4000    1.0000
```

4.3.5 Durum Denklemini Sıfır-Kutup-Kazanç Modeline Çevirme

'ss2zp' fonksiyonu durum denklemi verilen sistemin sıfırlarını, kutuplarını ve kazancını bulur. Aşağıdaki biçimde kullanılır.

[z,p,k] = ss2zp(A,B,C,D,i)

'i' sistemin giriş sayısını verir.

Örnek 16 : Örnek 11' de bulunulan durum denklemi değişkenlerini kullanarak bu sistemin sıfır-kutup-kazanç modelini oluşturunuz.

Çözüm :

```
>> [z,p,k] = ss2zp(A,B,C,D,1)
z =
   -1.5000
p =
  -0.2000 + 0.9798i
```

```
  -0.2000 - 0.9798i
k =
   2.0000
```

4.3.6 Sıfır-Kutup-Kazanç Modelini Durum Denklemine Çevirme

'zp2ss' fonksiyonu z, p ve k değişkenleri verilen bir sıfır-kutup-kazanç modelinin durum denklemi değişkenlerini bulur. Aşağıdaki biçimde kullanılır.

[A,B,C,D] = zp2ss(z,p,k)

Örnek 17 : Yukarıdaki örnekte bulunulan z, p ve k değerlerinden faydalanarak sistemin durum değişkenleri modelini oluşturunuz.

Çözüm :

```
>> [A,B,C,D]=zp2ss(z,p,k)
A =
  -0.4000   -1.0000
   1.0000         0
B =
     1
     0
C =
   2.0000    3.0000
D=
     0
```

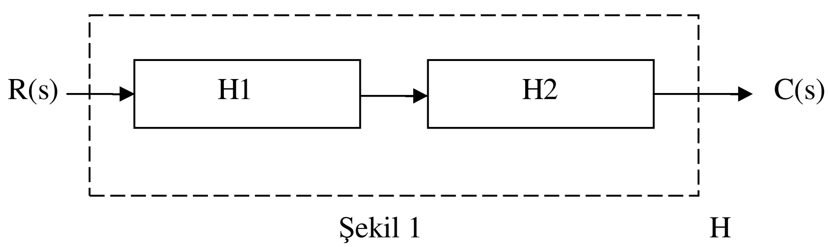
BÖLÜM 5

5 MODELLERİN BİRBİRİNE BAĞLANMASI

MATLAB/The Control System Toolbox modellerin birbirine bağlanması ve kompleks sistemlerin oluşturulabilmesi için birkaç fonksiyon kullanmaya imkan tanır. Bumlar; **[1]**

- 'series' ve 'parallel' fonksiyonları ile seri ve paralel sistemler birleştirilebilir.
- 'feedback' ve 'lft' fonksiyonları geribesleme bağlantılarını çözer
- Giriş/Çıkış bağlantıları matris yöntemi ve 'append' fonksiyonu ile yapılır.

5.1 Seri Bağlantı



Şekil 1

Yukarıda görüldüğü gibi birbirine paralel bağlı olan H1 ve H2 modelleri MATLAB ortamında 'series' fonksiyonu ile birleştirilerek H modeli oluşturulabilir. Bu sistemlerin diğer bir şekilde birleştirilmesi modellerin çarpılmasıyla mümkün olmaktadır.

Örnek 1 : Aşağıda verilen H1 ve H2 modellerini bağlayınız.

$$H_1 = \frac{s+1}{4s^2+3s+2} \quad , \quad H_1 = \frac{s}{7s^2-s+5}$$

Çözüm :

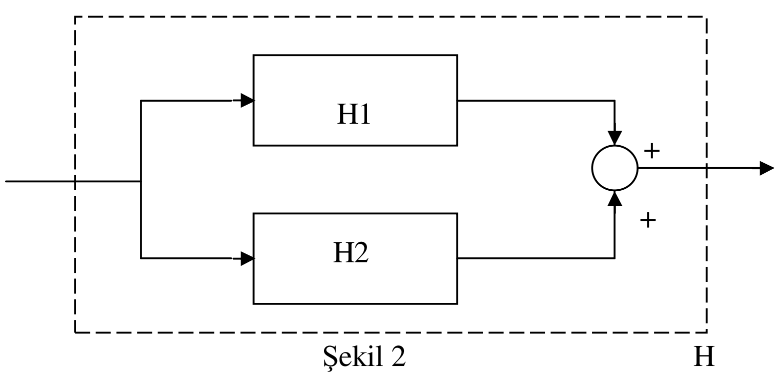
'series' fonksiyonu ile çözüm.

```
>> H1=tf([1 1],[4 3 2])

Transfer function:
     s + 1
---------------
4 s^2 + 3 s + 2

>> H2=tf([1 0],[7 -1 5])

Transfer function:
      s
-------------
7 s^2 - s + 5

>> H=series(H1,H2)

Transfer function:
             s^2 + s
------------------------------------
28 s^4 + 17 s^3 + 31 s^2 + 13 s + 10
```

Matris çarpımı (‘*’) ile çözüm

```
>> H=H1*H2

Transfer function:
             s^2 + s
------------------------------------
28 s^4 + 17 s^3 + 31 s^2 + 13 s + 10
```

5.2 Paralel Bağlantı



Şekil 2

Yukarıda görüldüğü gibi birbirine paralel bağlı olan H1 ve H2 modelleri MATLAB ortamında ‘parallel’ fonksiyonu ile birleştirilerek H modeli oluşturulabilir. Bu sistemlerin diğer bir şekilde birleştirilmesi modellerin toplanmasıyla mümkün olmaktadır.

Örnek 2 : Aşağıda verilen H1 ve H2 modellerini bağlayınız.

$$H_1 = \frac{1}{s^2 - 2s + 4} \quad , \quad H_1 = \frac{-2}{-9s^2 - 21s + 2}$$

Çözüm :

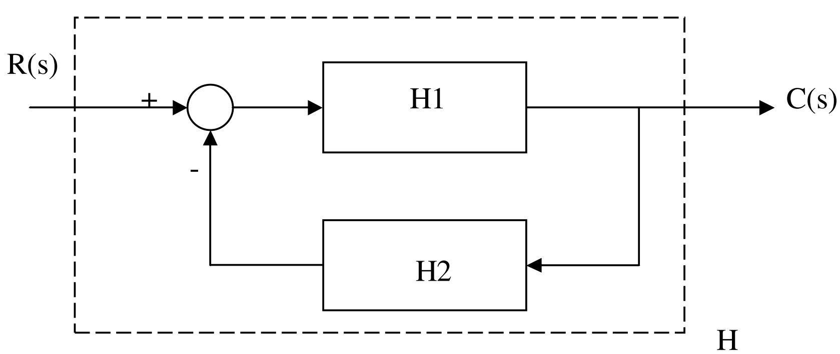
‘parallel’ fonksiyonu ile çözüm.

```
>> H1=tf([1],[1 -2 4]);
>> H2=tf([-2],[-9 -21 2]);
>> H=parallel(H1,H2)
Transfer function:
      11 s^2 + 17 s + 6
--------------------------------
9 s^4 + 3 s^3 - 8 s^2 + 88 s - 8
```

Matris çarpımı (‘+’) ile çözüm

```
>> H=H1+H2

Transfer function:
      11 s^2 + 17 s + 6
--------------------------------
9 s^4 + 3 s^3 - 8 s^2 + 88 s - 8
```

5.3 Geribeslemeli Bağlantı



Şekil 3

Yukarıda görüldüğü gibi geribesleme bağlantılı olan H1 ve H2 modelleri MATLAB ortamında ‘feedback’ fonksiyonu ile birleştirilerek H modeli oluşturulabilir. Geribesleme pozitif yada negatif geribesleme olmak üzere iki durumda olabilir. Negatif geribesleme için aşağıda gösterilen birinci fonksiyon, pozitif geribesleme için ikinci fonksiyon kullanılır. **[1]**

feedback(H1,H2)

feedback(H1,H2,+1)

Örnek 3 : Aşağıda verilen H1 ve H2 modellerini H2 negatif ve pozitif geribesleme olmak üzere her iki durum için bağlayınız.

$$H1 = \frac{5}{11s^2 - 12s + 9} \quad , \quad H2 = \frac{-2}{s-2}$$

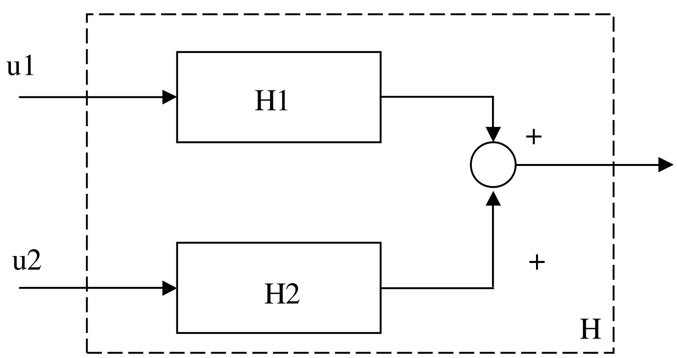
Çözüm : H2'nin pozitif geribesleme çözümü.

```
>> H1=tf([5],[11 -12 9]);

>> H2=tf([-2],[1 -2]);

>> H=feedback(H1,H2)
 Transfer function:
       5 s - 10
---------------------------
11 s^3 - 34 s^2 + 33 s - 28
```

H2'nin pozitif geribesleme çözümü.

```
>> H=feedback(H1,H2,+1)

Transfer function:
       5 s - 10
--------------------------
11 s^3 - 34 s^2 + 33 s - 8
```

5.4 Çıkışların Toplanması



$$y = \begin{bmatrix} H1 & H2 \end{bmatrix} \cdot \begin{bmatrix} u1 \\ u2 \end{bmatrix}$$

Şekil 4

Yukarıda görüldüğü gibi çıkışların bağlanması şeklinde olan H1 ve H2 modelleri MATLAB ortamında aşağıdaki şekilde birleştirilir ve H modeli oluşturulur.

H=[H1+H2]

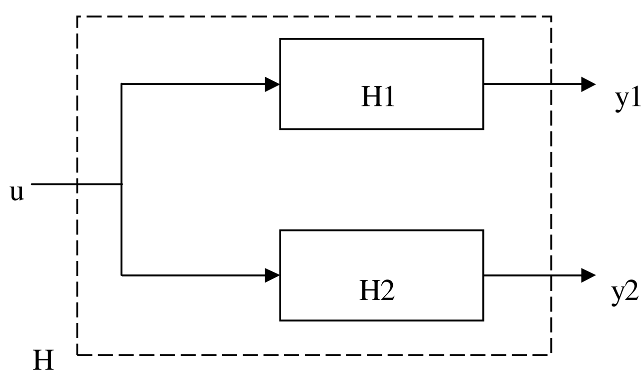
Örnek 4 : Örnek 3'te verilen H1 ve H2 modellerini çıkışların toplanması şeklinde bağlayınız.

Çözüm : H modelinde iki farklı giriş ve tek bir çıkış vardır.

```
>> H1=tf([5],[11 -12 9]);
>> H2=tf([-2],[1 -2]);
```

```
>> H=[H1,H2]

Transfer function from input 1 to output:
       5
-----------------
11 s^2 - 12 s + 9


Transfer function from input 2 to output:
 -2
-----
s - 2
```

5.5 Girişin Dağıtılması



$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} H1 \\ H2 \end{bmatrix} * u$$

Şekil 5

Yukarıda görüldüğü gibi girişin dağıtılması şeklinde olan H1 ve H2 modelleri MATLAB ortamında aşağıdaki şekilde birleştirilir ve H modeli oluşturulur. **[1]**

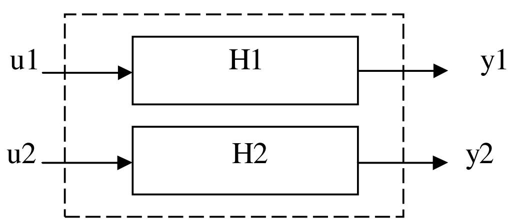
H=[H1;H2]

Örnek 5 : Girişi , örnek 3'te verilen H1 ve H2 modelleri üzerinden dağıtınız.

Çözüm : H modelinde iki farklı çıkış ve tek bir giriş vardır.

```
>> H1=tf([5],[11 -12 9]);
>> H2=tf([-2],[1 -2]);
>> H=[H1;H2]

Transfer function from input to output...
            5
 #1:  -----------------
     11 s^2 - 12 s + 9

      -2
 #2:  -----
     s - 2
```

5.6 Girişlerin ve Çıkışların Birleştirilmesi



$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} H1 & 0 \\ 0 & H2 \end{bmatrix} * \begin{bmatrix} u1 \\ u2 \end{bmatrix}$$

Şekil 6

Şekil 6'da görüldüğü gibi modellerin girişlerinin ve çıkışlarının bağlanması için 'append' fonksiyonu kullanılır. **[1]**

Örnek 6 : Önek 3'te verilen H1 ve H2 modellerini girişleri ve çıkışları birleştirerek modelleyiniz.

Çözüm :

```
>> H=append(H1,H2)

Transfer function from input 1 to output...
            5
 #1:  -----------------
     11 s^2 - 12 s + 9
```

```
 #2:  0

Transfer function from input 2 to output...
 #1:  0

          -2
 #2:  -------------
     s^2 - 2 s + 3
```

BÖLÜM 6

6 GİRİŞ FONKSİYONLARI

6.1 Basamak Cevabı (Step Response)

MATLAB' da doğrusal zamanla değişmeyen sistemlerin, transfer fonksiyonu, sıfır-kutup-kazanç ve durum denklemi modellerinin basamak cevabını hesaplamak ve grafiğini çizmek 'step' fonksiyonu ile gerçekleştirilir. Kullanım biçimi aşağıda gösterildiği gibidir.

step(sys)
step(sys,t)
'sys' değişkeni doğrusal zamanla değişmeyen bir sistemi temsil eder.
't' değişkeni zaman vektörüdür.
Birinci kullanım biçiminde 't' değişkeni girilmediği için zaman değişkenlerini MATLAB otomatik olarak seçer. **[1]**

[y,t]=step(sys,t)

Fonksiyonun yukarıda olduğu gibi kullanılması ile t vektörüne karşılık gelen basamak cevabının y vektöründe saklanması mümkün olur.

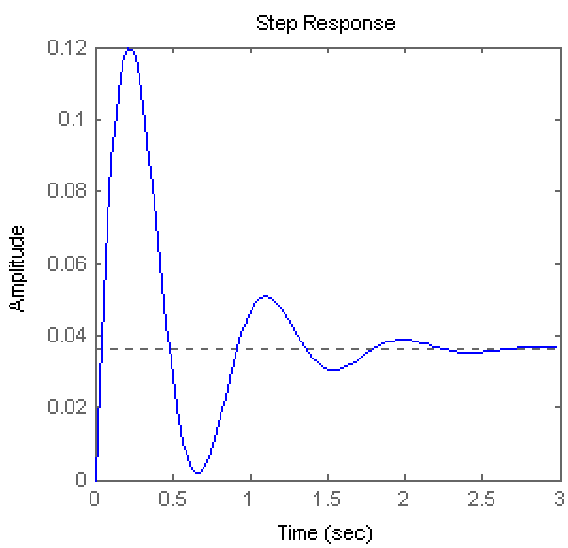
Örnek 1 : Transfer fonksiyonu aşağıda verilen sistemin basamak cevabını bulunuz.

$$T(s)=\frac{s+2}{s^2+4s+55}$$

Çözüm :

```
>> sys=tf([1 2],[1 4 55])

Transfer function:
    s + 2
--------------
s^2 + 4 s + 55

>> step(sys)
```

Yukarıdaki komut satırlarının icrası ile MATLAB' da aşağıdaki grafik çizilir ve kullanıcıya Şekil 6.1 görünür.



Şekil 6.1 örnek 1'in 'step' fonksiyonu grafiği

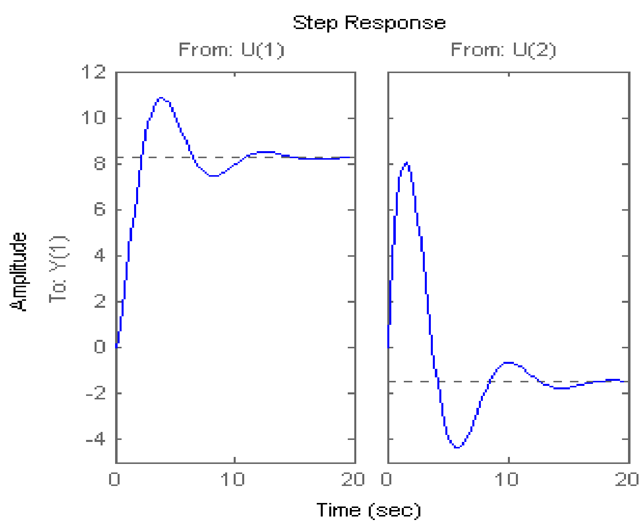
Örnek 2 : Aşağıda durum denklemi değişkenleri verilen sistemin basamak cevabını elde ediniz.

$$A=\begin{bmatrix}0.55 & 0.78\\ 0.78 & 0\end{bmatrix}\ ,\ B=\begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}\ ,\ C=\begin{bmatrix}1.96 & 6.44\end{bmatrix}\ ,\ \text{D=0}$$

Çözüm : İlk önce durum değişkenleri MATLAB' da matris olarak tanımlanır. Bu matrislerle 'ss' fonksiyonu kullanılarak durum denklemi modeli elde edilir ve 'step' fonksiyonu kullanılarak basamak cevabı hesaplanır.

```
>> a = [-0.5572   -0.7814;0.7814  0];
```

```
b = [1 -1;0 2];
c = [1.9691  6.4493];
sys = ss(a,b,c,0)

a =
              x1          x2
        x1    -0.5572     -0.7814
        x2     0.7814           0


b =
              u1          u2
        x1     1          -1
        x2     0           2

 c =
              x1          x2
        y1     1.9691      6.4493

d =
              u1          u2
        y1     0           0

Continuous-time model.
step(sys)
```



Şekil 6.2 örnek 2'deki 'step' fonksiyonu grafiği

Örnek 3 : $w_n = 1, \zeta = 0.1$ olarak verilen ikinci dereceden bir sistemin basamak cevabını elde ediniz.

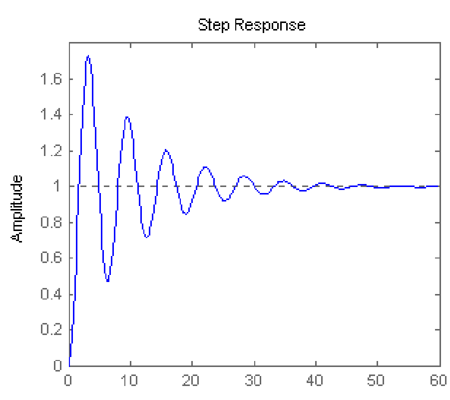
Çözüm :

Bu sorunun çözümünden önce MATLAB' da ikinci dereceden bir denklem oluşturmak için kullanılan 'ord2' fonksiyonunun anlatılmasında fayda görüyorum.
'ord2' fonksiyonunun kullanım biçimi aşağıda olduğu gibidir.

```
[A,B,C,D] = ord2(wn,z)
[num,den] = ord2(wn,z)
```

'wn' doğal frekans
'z' değişkeni ζ sönüm oranını (ksi) temsil eder
Yukarıda gösterilen fonksiyonun birinci kullanım şekliyle ikinci dereceden bir durum denklemi modeli elde edilir. İkinci kullanım şekliyle ise transfer fonksiyonu modeli elde edilir.

```
>> wn =1;
>>z =0.1;
>> [num,den] = ord2(wn,z)

num =

    1


den =

   1.0000    0.2000    1.0000

>> sys=tf(num,den);
>> step(sys)
```



Şekil 6.3 örnek 3'deki 'step' fonksiyonu grafiği

6.2 Anidarbe Cevabı (Impulse Response)

MATLAB' da doğrusal zamanla değişmeyen sistemlerin, transfer fonksiyonu, sıfır-kutup-kazanç ve durum denklemi modellerinin anidarbe cevabını hesaplamak ve grafiğini çizmek 'impulse' fonksiyonu ile gerçekleştirilir. Kullanım biçimi aşağıda gösterildiği gibidir.

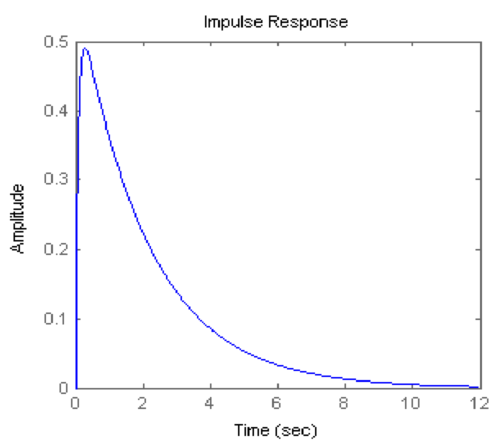
impulse(sys)

Örnek 4 : Transfer fonksiyonu aşağıda verilen sistemin basamak cevabını bulunuz.

$$T(s) = \frac{7}{s^2 + 13s + 6}$$

Çözüm :

```
>> sys=tf([7],[1 13 6])

Transfer function:
     7
--------------
s^2 + 13 s + 6

 >> impulse(sys)
```



Şekil 6.4 örnek 4'teki 'impulse' fonksiyonu grafiği

6.3 Rasgele Seçilmiş Giriş Cevabı

MATLAB' da doğrusal zamanla değişmeyen sistemlerin, transfer fonksiyonu, sıfır-kutup-kazanç ve durum denklemi modellerinin rasgele bir girişe olan cevabını hesaplamak ve grafiğini çizmek 'lsim' fonksiyonu ile gerçekleştirilir. Kullanım biçimi aşağıda gösterildiği gibidir. **[3]**

lsim(sys,u,t)

'u' değişkeni girişe uygulanan sinyaldir.
't' değişkeni zaman vektörüdür.

Bu fonksiyonun kullanılmasında 'gensig' fonksiyonunun bilinmesinin faydası olacağından kısaca 'gensig' hakkında bilgi vereceğiz. Bu fonksiyon MATLAB' da sinyal jeneratörü gibi davranır. Kullanım biçimi aşağıda olduğu gibidir. **[1]**

[u,t] = gensig(type,tau)
[u,t] = gensig(type,tau,Tf,Ts)

'type' değişkeni kullanulacak sinyalin çeşidini belirler. Bunlar;

- 'sin' sinüs sinyali
- 'square' kare dalga
- 'pulsu' darbe sinyali

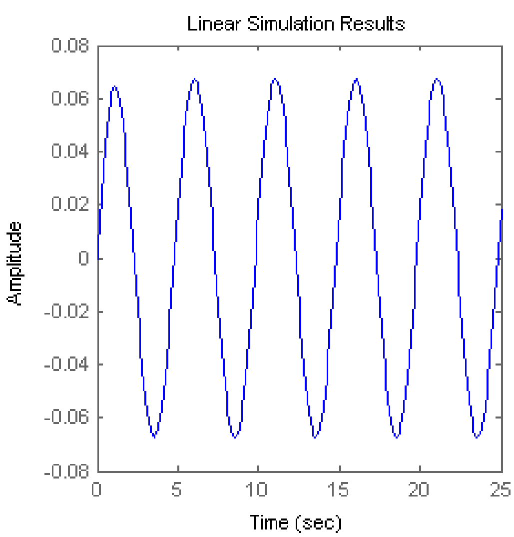
'tau' üretilen sinyalin periyodunu belirler
'Tf' sinyalin kaç saniye üretileceğini belirler
'Ts' örnekleme zamanını belirler
't' üretilen sinyalin zaman vektörü
'u' üretilen sinyalin zamana vektörüne karşılık gelen genlik değeri

Örnek 5 : Transfer fonksiyonu aşağıda verilen sisteme 25 sn boyunca, periyodu 5 sn ve örnekleme zamanı 0.1 olan sinüs dalga giriş olarak uygulandığında sistemin cevabını bulunuz.

$$T(s) = \frac{s+1}{s^2 + 11s + 21}$$

Çözüm :

```
>> sys=tf([1 1],[1 11 21]);
>> lsim(sys,u,t)
```



Şekil 6.5 örnek 5'teki 'lsim' fonksiyonu grafiği

BÖLÜM 7

7 ROUTH-HURWITZ KARARLILIK KRİTERİ

Bir sistemin kararlı olup olmadığı anlamak için kullanılan bir yöntemdir. Bu yöntem transfer fonksiyonu belli olan bir sistemin karakteristik denklemine bakılarak uygulanır. Uygulanacak sistemin öncelikle şu iki kurala uyması gerekmektedir. **[4]**

- Karakteristik denklemi oluşturan s'li ifadelerin katsayılarının '0'dan farklı olması gerekir.
- Karakteristik denklemi oluşturan s'li ifadelerin katsayılarının işaretlerinin aynı olması gerekir.

Bir sistemin karakteristik denkleminin aşağıda olduğu gibi verildiğini varsayalım.

$$KD = a_n s^n + a_{n-1} s^{n-1} + \dots\dots\dots + a_1 s^1 + a_0 = 0 \tag{7.1}$$

7.1 Hurwitz Kriteri

Routh-Hurwitz kriteri aşağıda ifade edilen kritere dayanır. (7-1) denkleminde tüm köklerin sol yarı s-düzleminde yer alması için gerek ve yeter koşul, k=1,2,.....,n için, denkleme ait tüm Hurwitz determinantları, n'den daha yüksek mertebeden ve negatif katsayılar sıfır alınmak üzere, şu şekilde türetilir:

$$D_1 = a_{n-1} \quad , \quad D_2 = \begin{vmatrix} a_{n-1} & a_{n-3} \\ a_n & a_{n-2} \end{vmatrix} \quad , \quad \dots\dots\dots$$

$$D_n = \begin{vmatrix} a_{n-1} & a_{n-3} & a_{n-5} & . & . & . & 0 \\ a_n & a_{n-2} & a_{n-4} & . & . & . & 0 \\ 0 & a_{n-1} & a_{n-3} & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & a_0 \end{vmatrix}$$

7.2 Routh Listelemesi

Bu yöntem yukarıda anlatılan Hurwitz kriterine göre uygulanması daha kolaydır. İlk olarak karakteristik denklemin katsayıları aşağıda gösterildiği gibi dizilir. **[4]**

$$\begin{matrix} a_n & a_{n-2} & a_{n-4} & . & . & . \\ a_{n-1} & a_{n-3} & a_{n-5} & . & . & . \end{matrix}$$

Sonraki adımlar, aşağıda dördüncü dereceden bir denklem için türetilen sayı dizisinden oluşur.

$$a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s^1 + a_0 = 0 \tag{7.2}$$

$$\begin{array}{c|c|cc}
s^4 & a_4 & a_2 & a_0 \\
s^3 & a_3 & a_1 & 0 \\
s^2 & b_1 & b_2 & 0 \\
s^1 & c_1 & c_2 & \\
s^0 & d_1 & &
\end{array}$$

$$\frac{a_4 a_1 - a_2 a_3}{a_3} = b_1 \quad , \quad \frac{a_4 0 - a_0 a_3}{a_3} = b_2 \quad ,$$

$$c_1 = \frac{a_3 b_2 - b_1 a_1}{b_1} \quad , \quad c_2 = \frac{a_3 0 - b_1 0}{b_1} = 0$$

$$d_1 = b_2$$

Yukarıdaki düzene Routh Listelemesi denir. Sistemin kararlılığına bakmak için yukarıda kesikli sütun içindeki elemanlara bakılır.
Routh listelemesinde birinci sütun elemanlarının işaretleri aynı ise denklem köklerinin tümü sol yarı s-düzlemindedir. Birinci sütundaki işaret değişimi sayısı pozitif gerçek kısımlı, ya da sağ yarı s-düzlemindeki köklerin sayısına eşittir. **[4]**

7.2.1 MATLAB' da Routh-Hurwitz Listelemesini Yapan Fonksiyon Hazırlamak

MATLAB' da dokuzuncu dereceye kadar olan karakteristik denklemlerin Routh-Hurwitz listesini çıkaran bir fonksiyon aşağıda olduğu gibidir..

'rliste' Fonksiyonu :

```
function [routh_table]=rliste(p)
s=size(p);
if s(1,2)<3 | s(1,2)>9,fprintf('Derece 2"den fazla 10"dan az olmali'),break,end
poz=0;
neg=0;
notr=0;
sifir=0;
yer=0;
der=0;
for i=1:s(1,2)
   if p(1,i)>0
      poz=poz+1;
   end
   if p(1,i)<0
      neg=neg+1;
   end
   if p(1,i)==0
      notr=notr+1;
   end
```

```
end
if   s(1,2)~=poz   &   s(1,2)~=neg   ,fprintf('Denklemin   Katsayilari   Gerek   Sarti
Saglamiyor.'),return,end

if s(1,2)==poz | s(1,2)==neg
   deg=s(1,2)/2;
   if deg>fix(deg)
      fdeg=fix(deg)+1;
      birler=zeros(s(1,2),fdeg+1);
      sifir=1;
   else
      sifir=0;
      fdeg=deg;
      birler=zeros(s(1,2),fdeg+1);
   end
   for i=1:fdeg
      for j=1:2
         if i<=fix(deg)
            yer=yer+1;
            birler(j,i)=p(yer);
         end
         if i==fdeg & sifir==1
            birler(1,fdeg)=p(yer+1);
            birler(2,fdeg)=0;
         end
      end
   end
end
routh=birler;
%
if fdeg-1>=0
for i=0:fdeg-1
    b(i+1)=(birler(2,1)*birler(1,i+2)-birler(1,1)*birler(2,i+2))/birler(2,1);
    routh(3,i+1)=b(i+1);
end
for i=1:2
    if routh(3,i)==0
        der=der+1;
    end
end
if der==2
    derece=s(1,2);
    routh=ydenklem(derece,routh,3);
    b(1,1)=routh(3,1);
    b(1,2)=routh(3,2);
end
der=0;

if fdeg-1==0 routh(4,1)=a(2);end
%
```

```
if fdeg-2>=0
for i=1:fdeg-1
    c(i)=(b(1)*birler(2,i+1)-b(i+1)*birler(2,1))/b(1);
    routh(4,i)=c(i);
end
for i=1:2
    if routh(4,i)==0
        der=der+1;
    end
end
if der==2
    derece=s(1,2);
    routh=ydenklem(derece,routh,4);
    c(1,1)=routh(4,1);
    c(1,2)=routh(4,2);

end
der=0;
if fdeg-2==0 routh(5,1)=b(2);end
%
if fdeg-3>=0
for i=1:fdeg-2
    d(i)=(c(1)*b(i+1)-c(i+1)*b(1))/c(1);
        routh(5,i)=d(i);
end
for i=1:4
    if routh(5,i)==0
        der=der+1;
    end
end
% fonksiyon buraya konulacak KD derecesi{s(1,2)},kacinci satir sifirlar,routh listesi
if der==4
    derece=s(1,2);
    routh=ydenklem(derece,routh,5);
    d(1,1)=routh(5,1);
    d(1,2)=routh(5,2);

end
end
if fdeg-3==0 routh(6,1)=c(2);end
der=0;
%
if fdeg-4>=0
for i=1:fdeg-3
    e(i)=(d(1)*c(i+1)-d(i+1)*c(1))/d(1);
        routh(6,i)=e(i);
end
for i=1:3
    if routh(6,i)==0
        der=der+1;
```

```
    end
end
if der==3
    derece=s(1,2);
    routh=ydenklem(derece,routh,6);
    e(1,1)=routh(6,1);
    e(1,2)=routh(6,2);

end
if fdeg-4==0 routh(7,1)=d(2);end
der=0;
%
if fdeg-5>=0
for i=1:fdeg-4
    f(i)=(e(1)*d(i+1)-e(i+1)*d(1))/e(1);
            routh(7,i)=e(i);
end
for i=1:3
    if routh(7,i)==0
        der=der+1;
    end
end
if der==3
    derece=s(1,2);
    routh=ydenklem(derece,routh,7);
    f(1,1)=routh(7,1);
    f(1,2)=routh(7,2);

end
der=0;
if fdeg-5==0 routh(8,1)=e(2);end
%
if fdeg-6>=0
for i=0:fdeg-1
    g(i+1)=(f(1)*e(i+2)-f(i+2)*e(1))/f(1);
            routh(8,i+1)=g(i+1);
end
for i=1:3
    if routh(8,i)==0
        der=der+1;
    end
end
if der==3
    derece=s(1,2);
    routh=ydenklem(derece,routh,8);
    g(1,1)=routh(8,1);
    g(1,2)=routh(8,2);

end
der=0;
```

```
if fdeg-6==0 routh(9,1)=e(2);end
%
end
end
end
end
end

fprintf('\n\n--Sistemin Routh Listesi--')
routh
```

%[1 4 8 8 7 4]

Yukarıdaki ‘rliste’ fonksiyonunun içinde çağrılan ’ydenklem’ fonksiyonu aşağıda olduğu gibidir.

‘ydenklem’ fonksiyonu

```
function [ydenklem]=ydenklem(boyut,rliste,ssatir)
yd_der=(boyut+1-(ssatir-1));
yd=rliste(ssatir-1,1:(yd_der-1));
yd1=zeros(1,yd_der);
k=1;
for i=1:(yd_der-1);
    yd1(1,k)=yd(1,i);
    k=k+2;
end
yd=polyder(yd1);
rliste(ssatir,1:yd_der-2)=yd(1,1:yd_der-2); % 2 çikartilmasinin sebebi 1 turevden + 1
yd_der'den 1 eksik olmali
ydenklem=rliste;
fprintf(' Yardimci Denklem Kullanildi. Satir No = %d',ssatir);
```

Örnek 1 : Yukarıda yazdığınız fonksiyonu karakteristik denklemi aşağıda verilen sistem üzerinde deneyiniz. **[4]**

$$2s^4 + s^3 + 3s^2 + 5s + 10 = 0$$

Çözüm : Yazdığımız fonksiyona karakteristik denklemin katsayılarını girdiğimizde Routh listesi karşımıza çıkmaktadır.

```
>> rliste([2 1 3 5 10])

--Sistemin Routh Listesi--
routh =
    2.0000    3.0000   10.0000         0
    1.0000    5.0000         0         0
   -7.0000   10.0000         0         0
    6.4286         0         0         0
   10.0000         0         0         0
         0         0         0         0
```

Örnek 2 : Yukarıda yazdığınız fonksiyonu karakteristik denklemi aşağıda verilen sistem üzerinde deneyiniz. **[4]**

$$s^5 + 4s^4 + 8s^3 + 8s^2 + 7s + 4 = 0$$

Çözüm : Fonksiyon listede 5 satırdaki tüm elemanların sıfır olduğunu tespit etti ve bunun üstesinden gelmek için bir yardımcı denklem oluşturdu. Şöyle ki ;
Sıfırlardan oluşan satırdan bir önceki satır s^2 satırıdır.
Yardımcı denklem bu yüzden aşağıda olduğu gibi oluşur.
$yd = 4s^2 + 4 = 0$
Yardımcı denklem bu şekilde oluşturulduktan sonra türevi alınır.
$(yd)' = 8$
Elde edilen sonuç sıfırlardan oluşan satıra yazılır.

```
>> rliste([1 4 8 8 7 4])
 Yardimci Denklem Kullanildi. Satir No = 5

--Sistemin Routh Listesi--
routh =

     1     8     7     0
     4     8     4     0
     6     6     0     0
     4     4     0     0
     8     0     0     0
     4     0     0     0
```

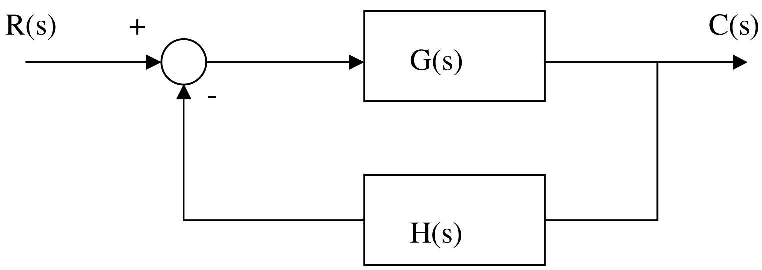
BÖLÜM 8

8 KÖK-YER EĞRİLERİ

Kök-yer eğrisi tek-giriş, tek-çıkışlı sistemler için kullanılan bir kararlılık çözümleme aracıdır. Kök-yer eğrisinin çözümlenmesi ile, sistem tasarımcısı köklerin nerede yer aldığını ve istenilen kararlılık ve cevap için transfer fonksiyonunda ne tür değişimler yapılması gerektiğini belirler.

8.1 Kök-Yer Eğrilerinin Çizimi

Şekil 1de verilen genel geribesleme sistemini ele alalım. Kapalı-döngü transfer fonksiyonun kutupların bulunması için aşağıdaki koşulların sağlanması gerekir.



Şekil 1 Kapalı-döngü sistem

$$1+G(s)H(s)=0 \tag{1.1}$$

$$G(s)H(s)= -1=1\angle 180° \ (2k+1) \tag{1.2}$$

burada,

$k=0, \pm1, \pm2,...$

dir. (1.2) nolu denklem, bir kapalı-döngü kutbu var olması halinde, açı koşulu ve modül koşulu olmak üzere iki durumu belirler. Burada G(s)H(s) bir karmaşık sayı fonksiyon olduğuna göre açı ve modül olmak üzere iki unsuru vardır. Açı koşulu,

$$\angle G(s)H(s) = 180°(2k+1) \ (k = O, \pm 1, \pm 2.....) \tag{1.3}$$

eklinde ifade edilir. Burada G(s)H(s) in açısı 180° çarpımının tek katlarıdır. Modül (büyüklük) koşulu: G(s)H(s) modülü birim değere eşit olmalı. Bu da,

$$|G(s)H(s)| = 1 \tag{1.4}$$

şeklinde gösterilir. Açı ve modül koşullarını sağlayan s değerleri karakteristik denklemin kökleri veya kapalı-döngü kutuplarıdır. Karmaşık açı koşulunu sağlayan noktaların çizdiği eğri köklerin geometrik yerinin eğrisi kısaca kök-yer eğrisidir. Kazancın belirli bir değerine karşılık karakteristik denklemin kökleri ise modül koşulundan belirlenir. **[4]**

8.2 Kök-Yer Eğrisi Çizim Kuralları [4]

Kural 1 : Kök-yer eğrisi birden fazla kollardan meydana gelebilir bu kolların sayısı karakteristik denklemin derecesine eşittir. Diğer bir deyişle kol sayısı açık-döngü kutup sayısına eşittir. Kök-yer eğrilerinin her bir bölümü veya kolu, kazancın değişimine bağlı olarak kapalı-döngü sistemin belli bir kutbunun hareketini tanımlar.

Kural 2 : Açık-döngü kutupları kök-yer eğrisinin başlama noktası (K=0) ve açık-döngü sıfırları da kök-yer eğrisinin bitiş noktasını ($K=\infty$) tanımlar. Buna göre kök yer eğrisi açık-döngü kutuplarında başlar ve sıfırlarında sona erer. Eğer G(s)H(s) in paydasının derecesi payın derecesinden büyük ise kök-yer eğrisi sonsuzda biter ve payın derece paydanın derecesinden büyükse kök-yer eğrisi sonsuzda başlar. Genellik paydanın derecesi payın derecesinden büyük olduğundan kök-yer eğrisi sonsuza gider.

Kural 3 : Gerçek eksen üzerinde yer alan kök-yer eğrisi kolları açık-döngü kutup ve sıfırlarından bulunur. Açık-döngü transfer fonksiyonunun karmaşık eşlenik kutuplarının ve sıfırlarının gerçek eksen üzerinde yer alan kök-yer eğrisi üzerinde bir etkisi yoktur. Çünkü karmaşık-eşlenik kutupların ve sıfırların gerçek eksen üzerindeki açı paylan 360° dir ve 180°(2k+l) olan açı koşulunu sağlamaz. Gerçek eksen üzerinde yer alan kök-yer eğrisinin her bir kısmı bir kutup veya sıfırdan diğer kutup veya sıfıra doğru uzanır

Kural 4 : Kök-yer eğrisinin asimptot açısı aşağıdaki denklem yoluyla bulunur.

$$\alpha = \frac{180(2k+1)}{n-m} \quad (\mathrm{k} = 0, \pm 1, \pm 2, \text{-}) \tag{1.5}$$

Burada
n= açık-döngü, G(s)H(s) kutuplarının sayısı
m= açık-döngü, G(s)H(s) sıfırlarının sayısı
Burada k=0 asimptotun gerçek eksen ile yaptığı en küçük açıya karşılık gelir. Her ne kadar k'nın sonsuz sayıda değeri olduğu kabul edilirse de k'nın artışı ile birlikte açılar kendilerini tekrarlarlar ve asimptot sayısı (n-m) değerine eşit olur.

Kural 5 : Tüm asimptotlar gerçek ekseni keser ve gerçek ekseni kestiği noktalar aşağıdaki ifade ile bulunur.

$$\sigma_a = \frac{\sum \mathrm{G(s)H(s)\ kutupları} - \sum \mathrm{G(s)H(s)\ sıfırlar}}{\mathrm{(kutup\ sayisi\ ) - (sifir\ sayisi)}} \tag{1.6}$$

Kural 6 : Kök-yer eğrisinin gerçek eksenden ayrılma noktası gerçek eksen üzerindeki kazanç katsayısı, K değerinin maksimum ve kök-yer eğrisinin gerçek eksen varış noktası K değerinin minimum olduğu noktadır. Kök-yer eğrisinin gerçek eksen etrafında simetrik olmasından dolayı, ayrılma noktaları ve varış noktalan ya gerçek eksen üzerinde yer alır ya da karmaşık kök çifti şeklinde ortaya çıkar.
Ayrılma ve varış noktalan, karakteristik denklemde K'yı çektikten sonra $s=\sigma$ koyarak hesaplanabilir daha sonra

$$\frac{dK(\sigma)}{d\sigma} = 0$$

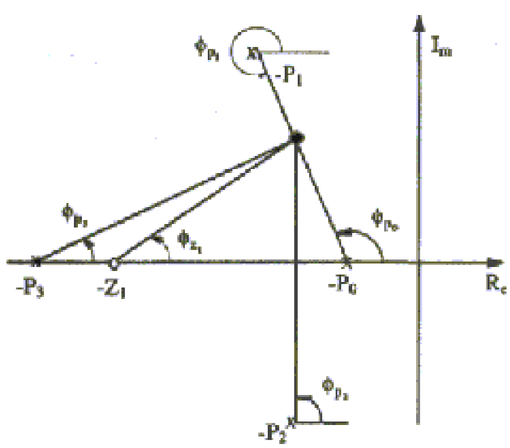
şeklinde türevi alındıktan sonra maksimum ve minimum yapan σ değerleri bulunur.

Kural 7 : Karmaşık kutuptan ayrılma açısı: Kök-yer eğrisinin bir karmaşık kutuptan ayrılma açısı (1.2) nolu açı koşulunu uygulayarak bulunur. Buna göre, Şekil 2' de gösterildiği gibi karmaşık kutbun çok yakınında bir test noktası seçilir ve diğer kutup ve sıfırlardan çizilen doğruların yatay ile yaptığı açılar toplamından 180° çıkarılarak elde edilir.

$$\varphi_{Z_1} - \left(\varphi_{P_0} + \varphi_{P_1} + \varphi_{P_2} + \varphi_{P_3}\right) = (2k+1)180$$

şeklinde ifade edilir. Simetriden dolayı diğer eşlenik kutuptan ayrılma açısı yukarıdaki değerin ters işaretlisi olur.

Kural 8 : Karmaşık sıfıra varış açısı: Kök-yer eğrisinin bir karmaşık sıfıra varış açısı yine açı koşulundan benzer şekilde bulunur.



Şekil 2 - Karmaşık kutuplardan ayrılma açısı

8.3 MATLAB' da Kök-Yer Eğrisi Çizimi

MATLAB' da doğrusal, zamanla değişmeyen bir sistemin kök-yer eğrisini çizmek için 'rlocus' fonksiyonu kullanılır. Kullanım biçimi aşağıda olduğu gibidir. **[6]**

rlocus(sys)
rlocus(sys,k)
[r,k]=rlocus(sys)

'sys' değişkeni kök-yer eğrisi çizilecek sistemin modelidir.
'k' kazanç değişkenidir.
'r' kazanç değerleri için sistemin kök yerlerini matris olarak elde eder.

Örnek 1 : Açık-döngü transfer fonksiyonu aşağıda verilen sistemin

A) Kök-yer eğrisini çiziniz.

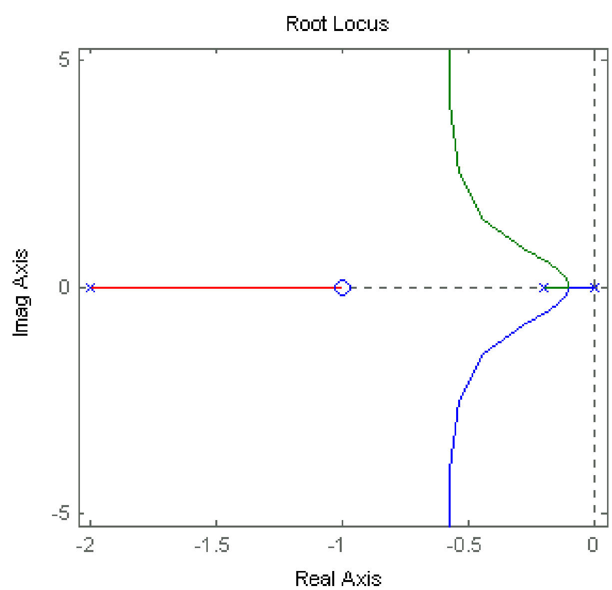
B) Çizilen grafikten sistemin kazancının 11 olduğu andaki sistemin kökünü, sönüm oranını ve frekans değerlerini bulunuz.

$$G(s)H(s) = \frac{K(s+1)}{s(s+2)(5s+1)}$$

Çözüm :

A)

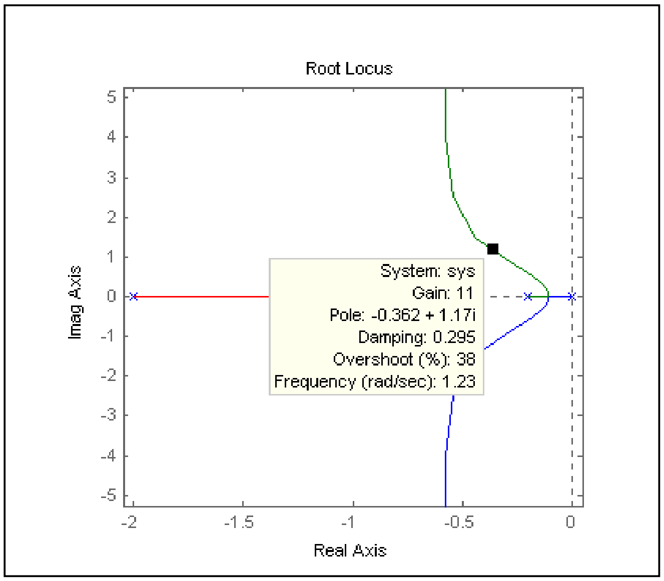
```
>> num=[1 1];
>> den=conv(conv([1 0],[1 2]),[5 1]);
>> sys=tf(num,den)

Transfer function:
        s + 1
  --------------------
  5 s^3 + 11 s^2 + 2 s

>> rlocus(num,den)
```



Şekil 3 – örnek 1'deki 'rlocus' fonksiyonu grafiği

B) 'rlocus' fonksiyonu ile MATLAB' in oluşturduğu grafikte çizilen çizgilerden herhangi bir yere fare ile tıklandığında otomatik olarak o noktanın kazanç, kutup, sönüm oranı ve frekans değerleri bir pencerede görüntülenir (Şekil 4).

Kazanç:11 , kutup: -0.362+1.17i , Sönüm oranı : 0.295 , Frekans : 1.23 (rad/sn)

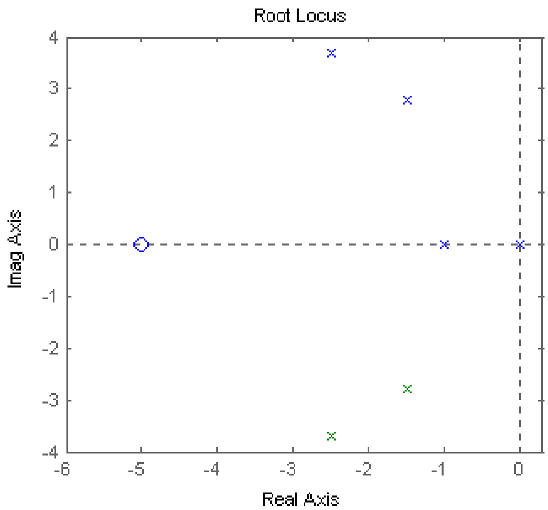


Şekil 4 kök-yer eğrisi üzerindeki bir noktanın değerleri

Örnek 2 : G(s)H(s) $= \dfrac{K(0.2s+1)}{s(s+1)}$ in kök-yer eğrisini çiziniz. K kazanç değişkeni 10 ve 20 için.

Çözüm :

```
>> num=[0.2 1];
>> den=conv([1 0],[1 1]);
>> sys=tf(num,den)

Transfer function:
0.2 s + 1
---------
 s^2 + s

 >> K=[10 20];
>> rlocus(sys,K)>> rlocus(num,den)
```



Şekil 4 – örnek 2'deki 'rlocus' fonksiyonu grafiği

Örnek 3 : $GH(s) = \dfrac{K}{s(4s+1)(0.4s+1)}$ sistemin karmaşık kök yerlerini elde ediniz. K kazanç değişken değerleri 5 ve 10 için.

Çözüm :

```
>> num=[1];
>> den=conv(conv([1 0],[4 1]),[0.4 1]);
>> sys=tf(num,den)

Transfer function:
         1
---------------------
1.6 s^3 + 4.4 s^2 + s

>> K=[5 10];
>> r = rlocus(sys,K)

r =

  -2.9051             -3.1736
   0.0776 + 1.0342i   0.2118 + 1.3873i
   0.0776 - 1.0342i   0.2118 - 1.3873i
```

BÖLÜM 9

9 FREKANS ALANI ANALİZİ

Sistemlerin frekans alanı analizinde, zaman değişimi yerine frekans değişimine karşılık gelen modül ve faz açısı değişimleri incelenir. Frekans alanı cevabı eğrilerinden sistemlerin çalışma frekansı aralığı yanında kararlılık durumlarında çözümlenir. Frekans alanı cevabında;

- Bode Diyagramı
- Nyquist Diyagramı
- Nichols Diyagramı

yöntemleri kullanılır. **[3]**
Frekans alanı çözümleme işlemi; giriş sinüzoidal frekansına karşılık bir transfer fonksiyonun genlik oranı ve faz açısı değerlerinin hesaplanmasından ibarettir.

9.1 Bode Diyagramı

Bode diyagramı, frekans değişimine karşı çizilen genlik oranı ve faz açısı eğrilerinden oluşur. Bunlarda genellikle frekans değerleri yatay eksende logaritmik ölçekte, genlikler oranının 10 tabanına göre logaritma değeri düşey eksende normal ölçekte yer alır. 10 tabanına göre genlik oranı logaritmasının çok küçük olmasından dolayı, genellikle bunun 20 katı olan desibel (dB) cinsinde değerler yer alır.
MATLAB' da bode diyagramlarının çizilmesi için 'bode' fonksiyonu kullanılmaktadır. Kullanım biçimi aşağıda olduğu gibidir. **[6]**

```
bode(sys)
bode(sys,w)
[mag,phase,w] = bode(sys) (sys,w)
```

'sys' transfer fonksiyonu, sıfır-kutup-kazanç veya durum denklemi olan bir doğrusal zamanla değişmeyen bir sistemdir.

'w' bode diyagramının çiziminde kullanılacak frekans değişkenlerinin atandığı vektördür.

'mag' sistemin frekans cevabının atandığı değişkendir.

'phase' derece cinsinden açıların atandığı değişken.

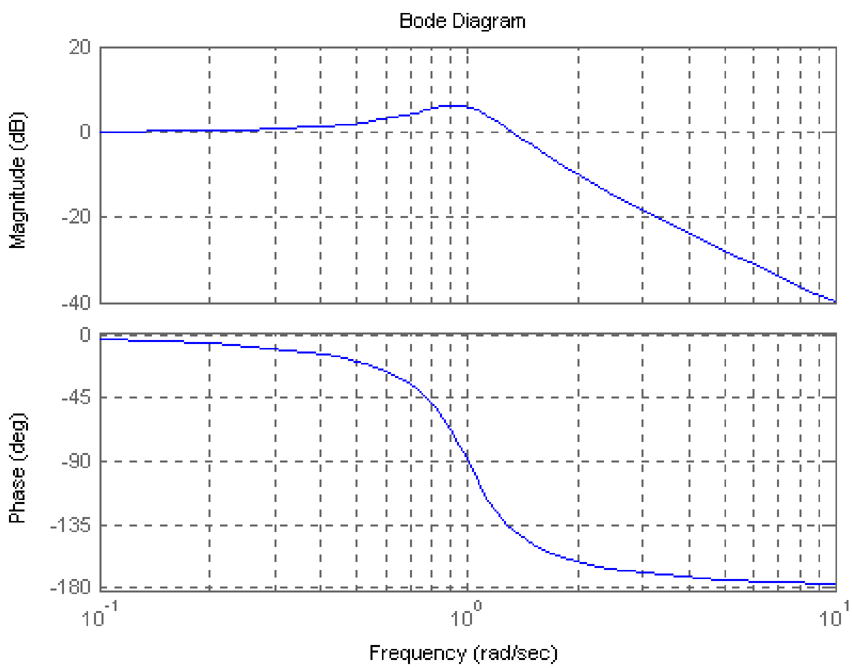
Örnek 1: Transfer fonksiyonu aşağıda verilen sistemin bode diyagramını çiziniz.

$$G = \frac{1}{s^2 + \frac{1}{2}s + 1}$$

Çözüm :

```
>> sys=tf([1],[1 1/2 1])
```

```
Transfer function:
       1
---------------
s^2 + 0.5 s + 1

>> bode(sys)
```



Şekil 9.1 örnek 1'deki bode fonksiyonu grafiği

Şekil 9.1'de görüldüğü gibi üstteki grafik sistemin genliğini, alttaki grafik ise sistemin fazını göstermektedir. Görüldüğü gibi x ekseni frekansı göstermekte logaritmik olarak artmaktadır. Faz derece cinsinden, genlik ise $20\log(|G(jw)|)$ olarak desibel cinsinden ifade edilmektedir. **[6]**

Örnek 2 : Yukarıda örnek 1'de verilen transfer fonksiyonunun genlik ve faz değerlerinin elde ediniz.

Çözüm :

```
>> bode(sys)
>> [mag,phase]=bode(sys)
```

Yukarıdaki komut satırlarının icrası ile MATLAB 50 değerli genlik ve faz değerlerini içeren iki vektör oluşturur. Aşağıda olduğu gibi.

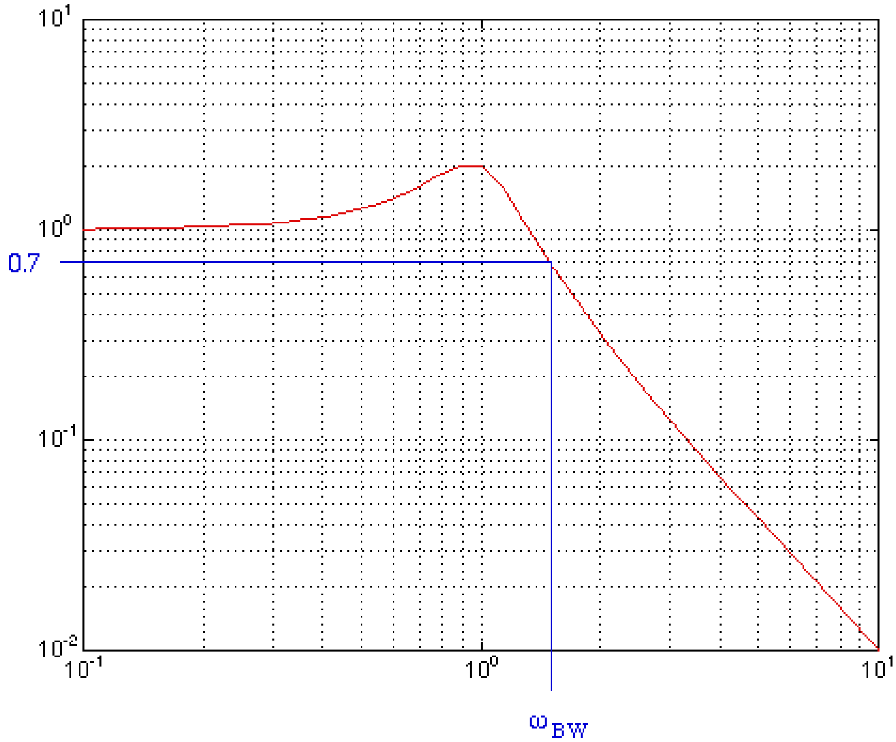
mag(:,:,1) = 1.0088	mag(:,:,21) = 1.7950	mag(:,:,43) = 0.0547	.....
mag(:,:,2) = 1.0112	mag(:,:,22) = 1.9003	mag(:,:,44) = 0.0428	phase(:,:,39) = -168.0468
mag(:,:,3) = 1.0142	mag(:,:,23) = 1.9879	mag(:,:,45) = 0.0335	phase(:,:,40) = -169.6753
mag(:,:,4) = 1.0180	mag(:,:,24) = 2.0457	mag(:,:,46) = 0.0263	phase(:,:,41) = -171.0246
mag(:,:,5) = 1.0229	mag(:,:,25) = 2.0656	mag(:,:,47) = 0.0207	phase(:,:,42) = -172.1594
mag(:,:,6) = 1.0292	mag(:,:,26) = 2.0422	mag(:,:,48) = 0.0163	phase(:,:,43) = -173.1250
mag(:,:,7) = 1.0372	mag(:,:,27) = 1.9588	mag(:,:,49) = 0.0128	phase(:,:,44) = -173.9541
mag(:,:,8) = 1.0475	mag(:,:,28) = 1.8042	mag(:,:,50) = 0.0101	phase(:,:,45) = -174.6712
mag(:,:,9) = 1.0608	mag(:,:,29) = 1.5884	phase(:,:,1) = -2.8913	phase(:,:,46) = -175.2950
mag(:,:,10) = 1.0781	mag(:,:,30) = 1.3407	phase(:,:,2) = -3.2617	phase(:,:,47) = -175.8401
mag(:,:,11) = 1.1006	mag(:,:,31) = 1.0941	phase(:,:,3) = -3.6820	phase(:,:,48) = -176.3180
mag(:,:,12) = 1.1301	mag(:,:,32) = 0.8714	phase(:,:,4) = -4.1599	phase(:,:,49) = -176.7383
mag(:,:,13) = 1.1694	mag(:,:,33) = 0.6818	phase(:,:,5) = -4.7050	phase(:,:,50) = -177.1087
mag(:,:,14) = 1.2223	mag(:,:,34) = 0.5260	phase(:,:,6) = -5.3288	
mag(:,:,15) = 1.2404	mag(:,:,35) = 0.4007	phase(:,:,7) = -6.0459	
mag(:,:,16) = 1.3065	mag(:,:,36) = 0.3531	phase(:,:,8) = -6.8750	
mag(:,:,17) = 1.3843	mag(:,:,37) = 0.2633	phase(:,:,9) = -7.8406	
mag(:,:,18) = 1.4740	mag(:,:,38) = 0.1990	phase(:,:,10) = -8.9754	
mag(:,:,19) = 1.5746	mag(:,:,39) = 0.1518	phase(:,:,11) = -10.3247	
mag(:,:,20) = 1.6834	mag(:,:,40) = 0.1167	phase(:,:,12) = -11.9532	
	mag(:,:,41) = 0.0903	phase(:,:,13) = -13.9558	
	mag(:,:,42) = 0.0702	phase(:,:,14) = -16.4787	

Band genişliği (Bandwidth) : Sistemin genliğinin 0.707 yada –3dB olduğu andaki frekans değerine band genişliği denir.

Örnek 3 : Örnek 1'de verilen sistemin band genişliğini bulunuz. **[7]**

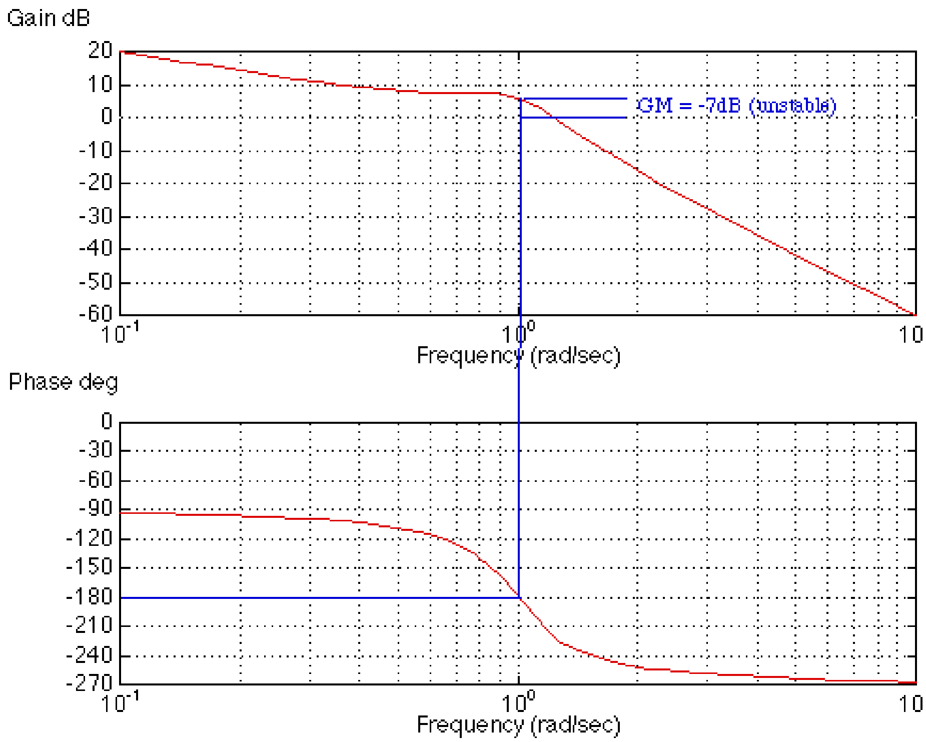
Çözüm : Öncelikle sistemin genlik ve faz değerleri bulunur. Bulunan değerlerle bir grafik çizilir ve 0.707'ye gelen genlik değerinden frekans değeri elde edilir. Buda sistemin band genişliğini verir. Şekil 9.3'te görüldüğü gibi.

```
>> numG = 1;
>> denG = [1 0.5 1];
>> [m,p,w]=bode(numG,denG);
>> loglog(w,m);
>> axis([0.1 10 0.01 10]);
>> grid;
```



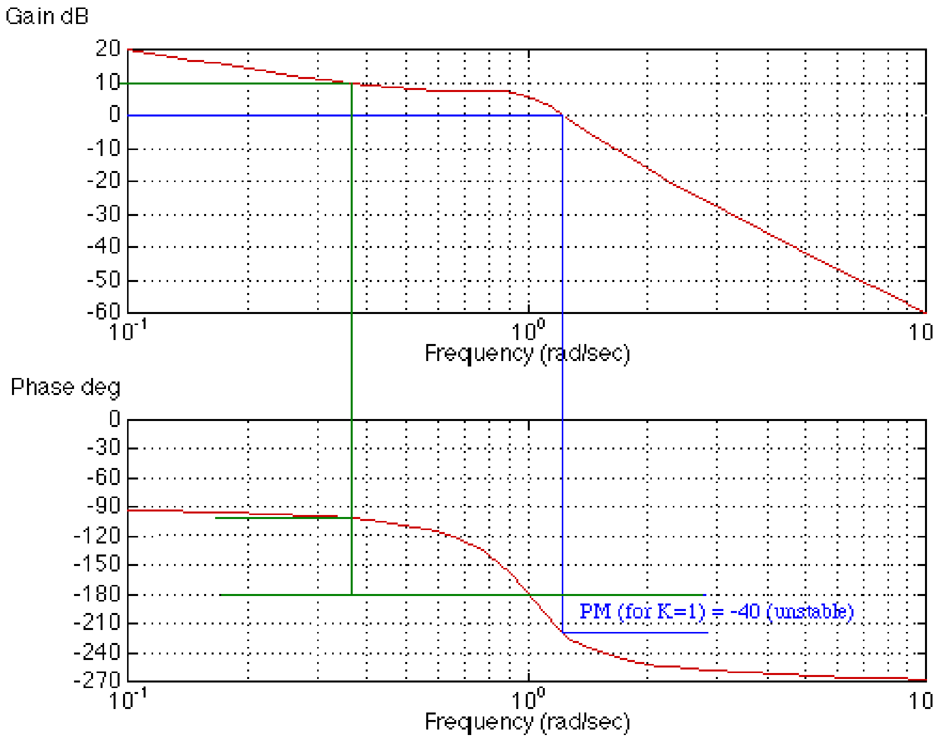
Şekil 9.2 Kazanç Payı'nın bode diyagramı üzerinde bulunması

Kazanç payı (Gain margin) : Bir sistemin girişiyle çıkışı arasındaki faz farkının –180° olduğu andaki genlikler oranı tersidir. Şekil 9.3'de GM olarak ifade edilen değer kazanç payıdır. Aşağıda da görüldüğü gibi faz açısının -180° olduğu değerden genlik grafiğine bir dik çizildiğinde elde edilen genlik değerinin sıfırdan çıkartılmasıyla GM elde edilir. **[7]**



Şekil 9.3 Kazanç Payı'nın bode diyagramı üzerinde bulunması

Faz payı (Phase margin) : Bir sistemin girişiyle çıkışı arasındaki faz farkının –180° olduğu andaki frekans değeridir. **[7]**



Şekil 9.4 Faz Payı'nın bode diyagramı üzerinde bulunması

Şekil 9.4'te görülen PM sistemin faz payını ifade etmektedir. PM, genlik değerinin sıfır olduğu andaki açı değerinin, –180° açı değeri ile arasındaki farktır.

Örnek 4 : Transfer fonksiyonu aşağıda verilen sistemin kazanç ve faz paylarını bulunuz.

$$G(s) = \frac{1}{s(s+1)(s+2)}$$

Çözüm :

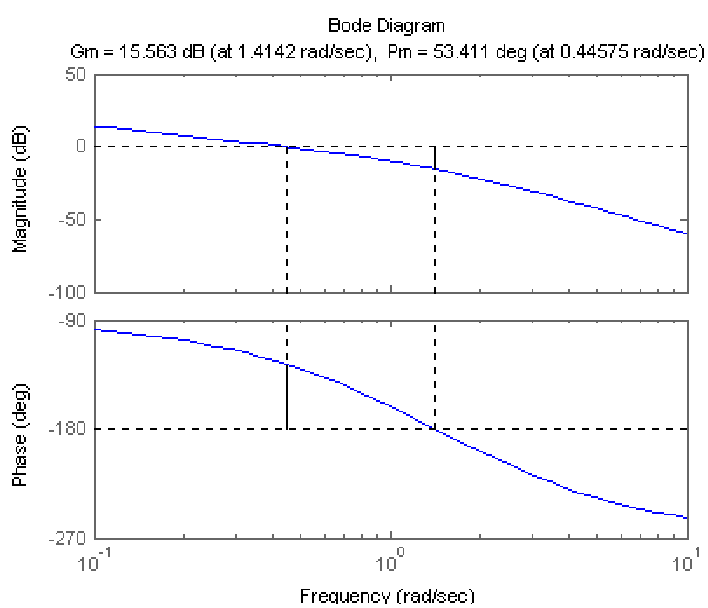
```
>> sys=zpk([],[0 -1 -2],1)

Zero/pole/gain:
     1
-------------
s (s+1) (s+2)
```

```
>> [qm,pm,wcg,wcp]=margin(sys)

qm =

    6.0000


pm =

   53.4109


wcg =

    1.4142


wcp =

    0.4457
```

Sistemin wcg ve wcp frekanslarındaki gm (kazanç payı) ve qm (faz payı) değerlerini gösterir. Aynı sonucun grafiksek olarak bulunması şu şekilde olur.

```
>> margin(sys)
```

Yukarıda ki komut satırı işletildiğinde aşağıdaki grafik oluşur.



Şekil 9.5 örnek 4'teki 'margin' fonksiyonunun grafiği

9.2 Nyquist Diyagramı

Nyquit diyagramı frekans değişimlerine karşılık gelen modül ve faz açısı değişimlerinin eğrisini verir. The Control System Toolbox'ta yer alan nyquist fonksiyonu bir frekans cevabı fonksiyonu olup, bode fonksiyonunda kullanılan giriş argümanlarını kullanır. İki fonksiyon arasındaki fark çıkış argümanlarıdır. Nyquist fonksiyonu farklı frekans değerleri için açık döngü transfer fonksiyonunun sanal bileşenine karşılık gelen gerçek bileşenin eğrisini çizer. Nyquist eğrisi genellikle kararlılık çözümlemesi için kullanılır. Kullanım biçimi aşağıda olduğu gibidir. **[6]**

```
nyquist(sys)
nyquist(sys,{wmin,wmax})
[re,im]=nyquist(sys,w)
```

'sys' transfer fonksiyonu, sıfır-kutup-kazanç ve durum denklemi alan bir sistem.
'wmin,wmax' nyquist çiziminin oluşturulacağı maksimum ve minimum frekans değerleri
're' frekans cevabını oluşturan gerçek (real) kısımlar
'im' frekans cevabını oluşturan sanal (imajiner) kısımlar

Örnek 5 : Aşağıda transfer fonksiyonu verilen sistemin nyquist diyagramını çiziniz.

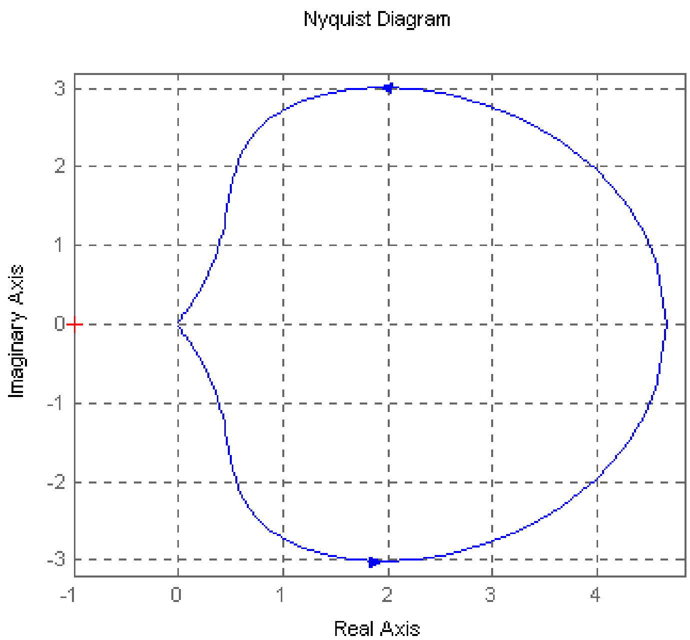
$$T(s) = \frac{7(s+2)}{(s-1)^2(s^2+2s+3)}$$

Çözüm :

```
>> sys1=zpk(-2,[1 1],7)

Zero/pole/gain:
7 (s+2)
-------
(s-1)^2

>> sys2=tf(1,[1 2 3])

Transfer function:
      1
-------------
s^2 + 2 s + 3

>> sys=sys1*sys2

Zero/pole/gain:
       7 (s+2)
---------------------
(s-1)^2 (s^2 + 2s + 3)

>> nyquist(sys)
>> grid on
>> title('Nyquist diyagrami')
```

Yukarıdaki komut satırlarının icrası ile Şekil 9.6'teki grafik çizilir. 'nyquist' fonksiyonu sistemin nyquist eğrisini çizer.'grid on' fonksiyonu grafik penceresinin ölçeklendirilmesi için kullanılır. 'title' fonksiyonu grafik için bir başlık ataması yapar.

Şekil 9.6'de görüldüğü gibi çizimde gerçek eksende (-1) değeri (+) işareti ile gösterilmiştir. Bu nokta nyquist eğrilerinin kararlılıklarını değerlendirmek açısından önemlidir.



Şekil 9.6 örnek 5'deki 'nyquist' fonksiyonunun çizimi

Örnek 6 : Durum denklemi değişkenleri aşağıda verilen sistemin nyquist diyagramını çiziniz.

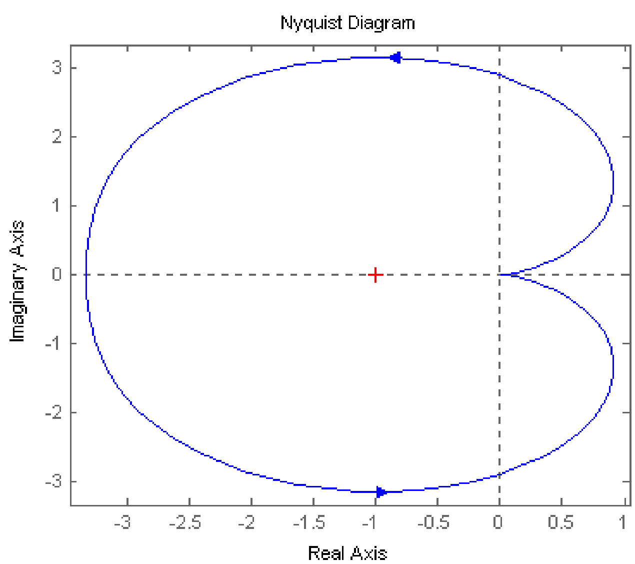
$A=\begin{bmatrix}1 & -1\\ 2 & 1\end{bmatrix}$, $B=\begin{bmatrix}0\\ 1\end{bmatrix}$, $C=\begin{bmatrix}1 & 0\end{bmatrix}$, D=0

Çözüm :

```
>> A=[1 -1;2 1];
>> B=[0 ;1];
>> C=[1 0];
>> D=0;
>> sys=ss(A,B,C,D)

a =
              x1        x2
       x1      1        -1
       x2      2         1


b =
```

```
             u1
       x1     0
       x2     1

 c =
             x1      x2
       y1    10       0

d =
             u1
       y1     0
Continuous-time model.

>> nyquist(sys)
```

Yukarıdaki komut satırlarının icrası ile Şekil 9.7'teki grafik çizilir.



Şekil 9.7 örnek

Argüman ilkesi : Bu ilke ile nyquist diyagramına bakarak sistemin kararlı olup olmadığını söyleyebiliriz. Argüman ilkesi (9.1) denkleminde ifade edilmektedir.
Argüman ilkesine göre bir sistem için Z = 0 (sıfır) ise sistem kararlıdır denir. $Z \neq 0$ ise sistem kararsızdır. **[4]**

$$N=Z-P \tag{9.1}$$

N : Nyquist diyagramında (-1)'i çevreleme sayısı . Saat yönünde çevreleme pozitif, tersi olan çevreleme negatif alınır ve toplamı N'yi verir.
Z : Kapalı sistemin transfer fonksiyonunun sağ yarım küredeki kutuplarının sayısı
P : G(s)H(s)'nin açık çevrim kutuplarının sayısı

Çevreleme : Bir karmaşık fonksiyon düzleminde eğer bir nokta yada bölge kapalı bir yolun içinde bulunuyorsa o nokta yada bölgeye çevrelenmiş denir.

Kapsama : Kapalı bir yol tarafından çevrelenen nokta yada düzlem için çevreleme yönünün solunda kalan bölge kapsanmış bölgedir.

Örnek 7 : Açık çevrim transfer fonksiyonu aşağıda verilen sistemin nyquist diyagramını çizin ve kararlı olup olmadığını bulun.

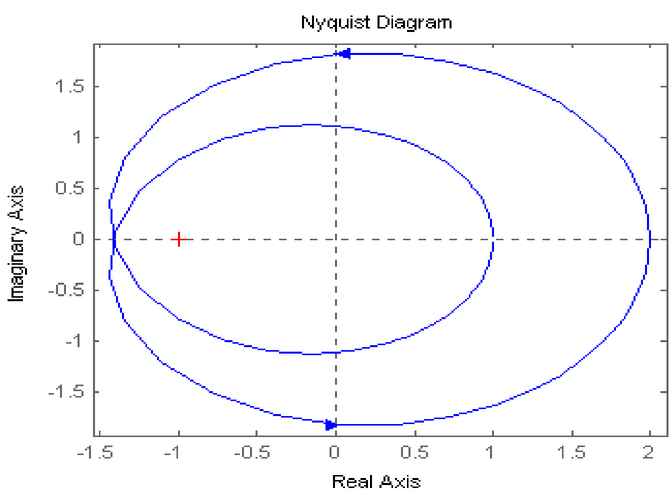
$$G(s) = \frac{s^2 + 10s + 24}{s^2 - 7s + 12}$$

Çözüm : Z = P + N denklemi yoluyla kararlılık test edilecektir. Bunun için önce sistemin açık çevrim transfer fonksiyonunun kutupları sayısına bakılır.

```
>> roots([1 -7 12])
ans =
     4
     3
```

Yukarıda görüldüğü gibi N=2'dir.
Bundan sonra sistemin nyquist diyagramı çizilir ve (-1) çevreleme sayısına bakılır.

```
>> sys=tf([1 10 24],[1 -7 12])

Transfer function:
s^2 + 10 s + 24
---------------
s^2 - 7 s + 12

>> nyquist(sys)
```



Şekil 9.8 örnek 7'daki 'nyquist' fonksiyonu grafiği

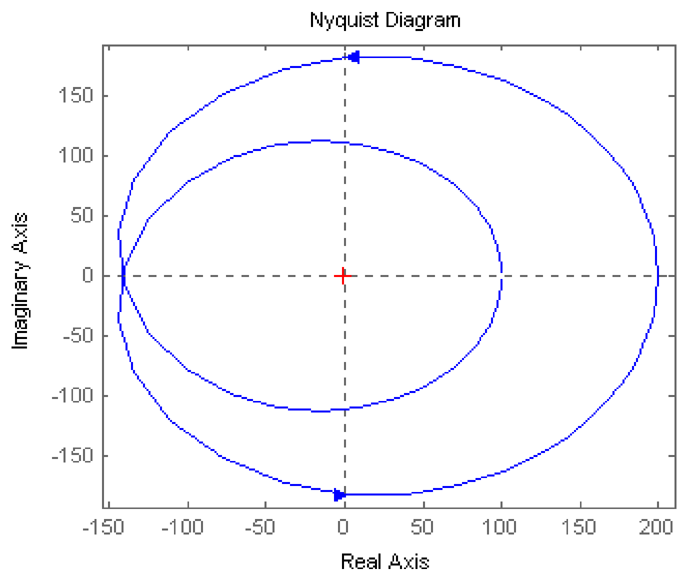
Şekil 9.8 ‘de de görüldüğü gibi (-1) noktası saatin tersi yönünde iki kere çevrelenmiştir. Bu yüzden N=-2’dir.

Z=P+N denkleminden Z=0 olduğu görülür. Bu sistemimizin kararlı olduğunu göstermektedir.

Örnek 8 : Yukarıdaki örnekte sistemimizin kazancı 1 kabul edildiğini varsayarsak kazancın 100 değeri için nyquist diyagramını elde ediniz.

Çözüm :

```
>> nyquist(sys*100)
```



Şekil 9.9 örnek 8’deki ‘nyquist’ fonksiyonu

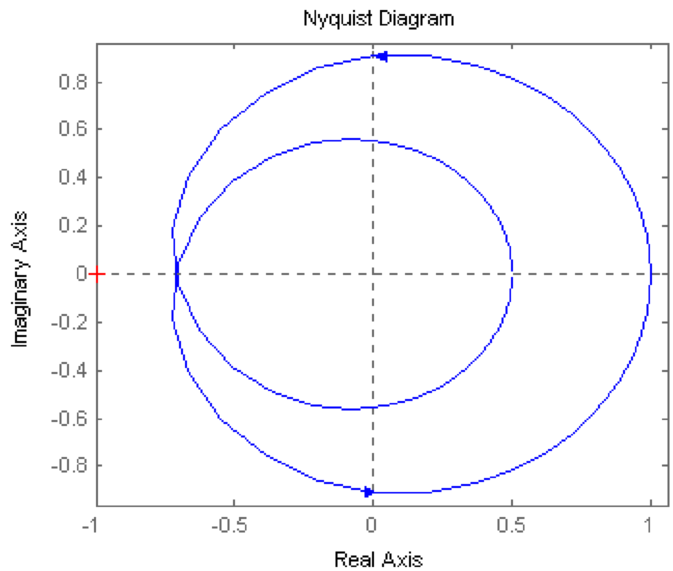
Şekil 9.9 ‘de de görüldüğü gibi sistemin kazancı arttırıldığında sistemin kararlılığı değişmemektedir.

Örnek 9 : Yukarıdaki örnekte sistemimizin kazancı 1 kabul edildiğini varsayarsak kazancın 0.5 değeri için nyquist diyagramını elde ediniz.

Çözüm :

```
>> nyquist(sys*0.5)
```

Şekil 9.8 örn



Şekil 9.10 örnek 9'deki 'nyquist' fonksiyonu grafiği

Şekil 9.10 'da da görüldüğü gibi sistemin kazancı azaltıldığında sistemin kararlılığı değişmektedir.

9.3 Nichols Abağı

G(jw) Nyquist yer eğrisinde kutupsal koordinatlarda çalışmanın en önemli sakıncası, sistemde çevrim kazancının değiştirilmesi gibi basit bir işlem yapıldığında, eğrinin ilk özgün biçimini korumamasıdır. Tasarımda genellikle, çevrim kazancını değiştirmek gerektiği gibi sisteme seri kontrolörlerde eklemek gerekebilir. Bu durumda sistemin nyquist diyagramı yeniden çizilmelidir. Mr ve BG ile ilgili tasarım işlemlerinde G(jw)'nın genlik-faz eğrisi ile çalışmak kolaylık sağlar, çünkü çevrim kazancı değiştirildiğinde tüm G(jw) eğrisi değişikliğe uğramadan yukarı aşağı kayar. G(jw)'nın faz özelliği kazançtan bağımsız değiştirildiğinde ise genlik-faz yer eğrisi sadece yatay doğrultuda etkilenir. **[4]**

MATLAB' da nichols abağının çizimini sağlayan fonksiyon 'nichols' dur. Kullanım biçimi aşağıda olduğu gibidir.

nichols(sys)
nichols(sys,w)
[mag,phase] = nichols(sys,w)

'mag' frekans cevabı genlik değerlerinin atandığı değişkendir
'phase' frekans cevabı faz değerlerinin atandığı değişkendir

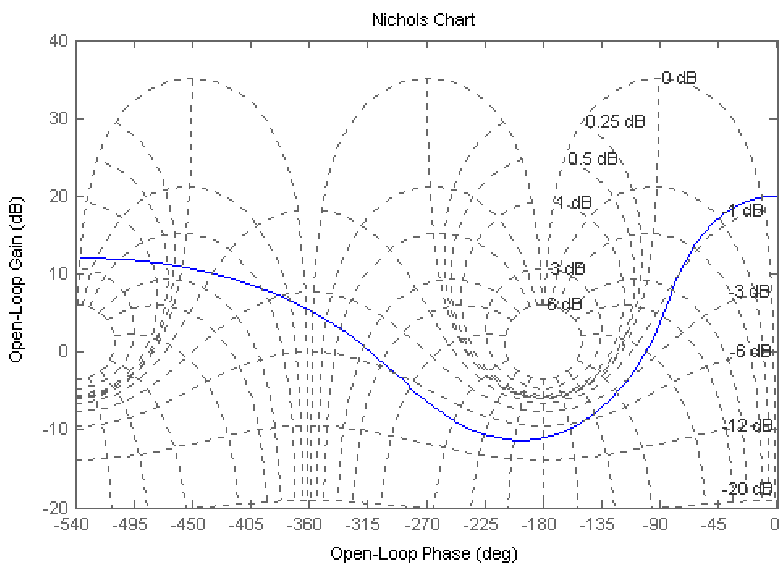
Örnek 10 : Transfer fonksiyonu aşağıda verilen sistemin nichols diyagramını çiziniz.

$$T(s) = \frac{-4s^4 + 48s^3 - 18s^2 + 250s + 600}{s^4 + 30s^3 + 282s^2 + 525s + 60}$$

Çözüm :

```
>> num = [-4 48 -18 250 600];
>> den = [1 30 282 525 60];
>> sys = tf(num,den)

Transfer function:
-4 s^4 + 48 s^3 - 18 s^2 + 250 s + 600
-------------------------------------
 s^4 + 30 s^3 + 282 s^2 + 525 s + 60

>> nichols(sys); ngrid
```

Yukarıdaki komut satırları işletildiğinde Şekil 9.11 grafiği çizilir.

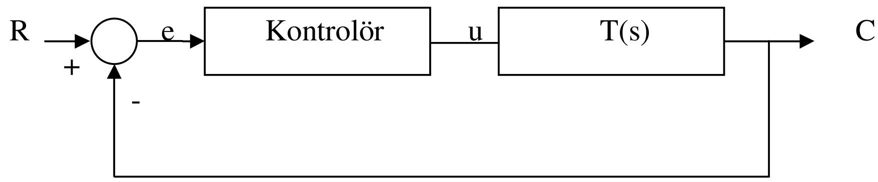


Şekil 9.11 örnek 10'daki 'nichols' fonksiyonu grafiği

BÖLÜM 10

10 MATLAB' DA PID TASARIMI

Kapalı döngü denetim sistemi iki bölümden oluşmaktadır. Birincisi denetlenen sistem, ikincisi ise kontrolördür.



Kontrolörlerde kullanılan belli başlı denetim etkileri şunlardır;

- Orantı denetim etkisi (P etki)
- İntegral denetim etkisi (I etki)
- Türev (Differansiyel) denetim etkisi (D etki)

Bu temel denetim etkilerinin bir yada birkaçının bir arada kullanılmasıyla çeşitli kontrolör sistemleri elde edilebilir. PID kontrolör' de bunlardan biridir. Yukarıda ki üç denetim etkisinin birleşimiyle oluşan kontrolöre PID denir. PID kontrolü seri bağlı PI ve PD kısımlarından oluşur. PID kontrolörünün transfer fonksiyonu aşağıdaki şekilde yazılabilir. **[4]**

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \tag{10.1}$$

K_P : Oransal kazanç
K_I : İntegral kazancı
K_D : Türev kazancı

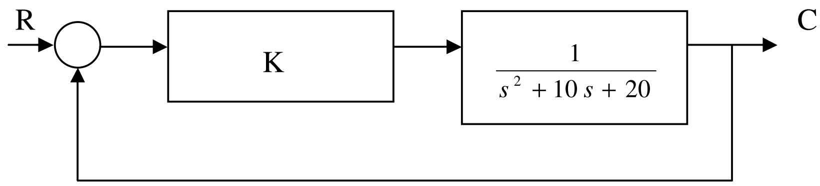
10.1 P, I ve D Kontrolörlerin Özellikleri

Tablo 10.1 P, I ve D denetim etkilerinin temel karakteristik özelliklerini içermektedir. Genellikle bu tablo sistemlerde doğrulanmakla beraber bazen bazı sistemler farklı sonuçlar verebilir, çünkü Kp, Ki ve Kd bir sistemde birbirlerini etkileyen elementlerdir.

Tablo 10.1 P,I ve D karakteristik özellikleri **[6]**

Kazanç	**Yükselme zamanı**	**Aşım**	**Yerleşim zamanı**	**Kalıcı-durum hatası**
Kp	Azalır	Artar	Etkisi az	Azalır
Ki	Azalır	Artar	Artar	Kaldırır
Kd	Etkisi az	Azalır	Azalır	Etkisi az

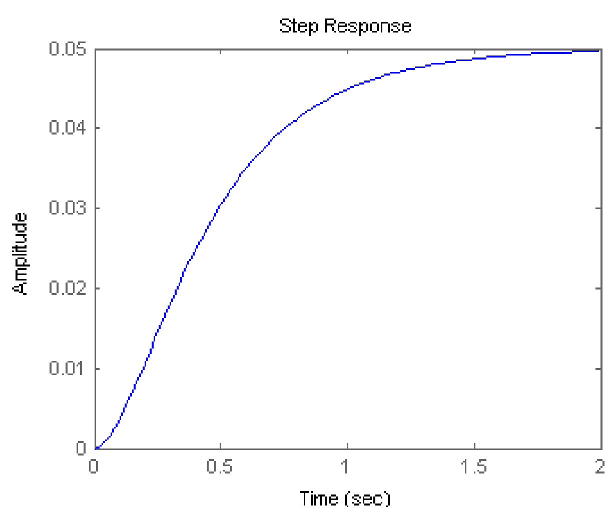
Örnek 10.1 : Aşağıda verilen sistemi MATLAB ortamında çözünüz. K sistemin kontrolörüdür. **[6]**



Çözüm :

Öncelikle, K=1 iken, yani sistemimizin kontrolör etkisi olmadan nasıl bir basamak cevabı olduğunu görelim

```
>> sys=tf([1],[1 10 20])

Transfer function:
       1
---------------
s^2 + 10 s + 20

>> step(sys)
```



Şekil 10.1 örnek 1’deki sistemin K=1 için birim basamak cevabı

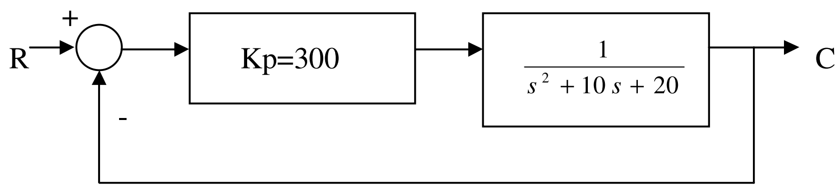
Şekil 10.1’de görüldüğü gibi transfer fonksiyonumuzun kazancı 0.05 dir. Bu, girişi birim basamak olan bir sistemin çıkışıdır. Sistemin kalıcı hal hatası yaklaşık 0.95 dir. Yükselme zamanı (the rise time) yaklaşık 1 saniyedir. Yerleşme zamanı (the settling time) 1.5 saniyedir.

Oransal kontrol (P etki)

Oransal kontrolörün transfer fonksiyonu sabit bir sayı şeklindedir ve **Kp** ile gösterilir. Oransal kontrol sistemin yükselme zamanını azaltır ve kalıcı durum hatasını azaltır, aşımı arttırır.

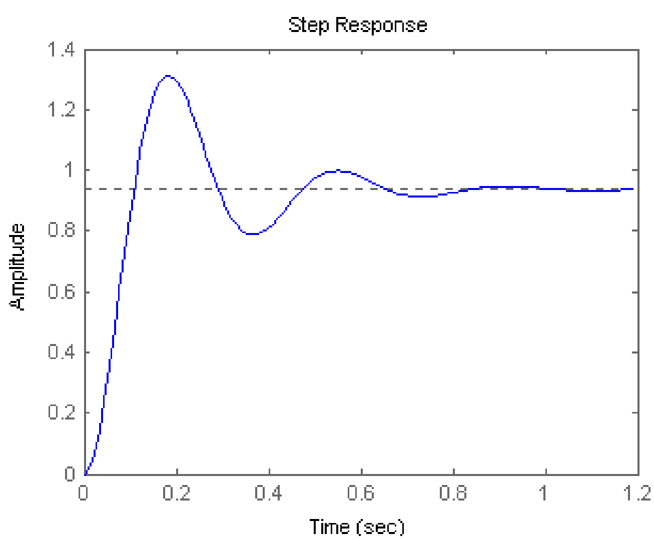
Örnek 2 : Transfer fonksiyonu yukarıda verilen sistemin oransal kontrolör kullanılarak birim basamak cevabını elde ediniz.

Çözüm : Öncelikle oransal kontrolörle sistem aşağıda olduğu gibidir. Oransal kontrolör kazancı Kp=300 alınır.




```
>> G=tf([1],[1 10 20]);
>> Kp=300;
>> Gc=G*Kp;
>> sys=feedback(Gc,1,-1)

Transfer function:
      300
----------------
s^2 + 10 s + 320

>> step(sys)
```

Yukarıdaki komut satırları icra edildiğinde birim basamak cevabı Şekil 10.2'de olduğu gibidir. Grafiğe bakıldığında oransal kontrolörün karakteristikleri görülecektir.



Şekil 10.2 örnek 2'deki sistemin Kp=300 için birim basamak cevabı

Şekil 10.2 'de görüldüğü gibi sistemin kazancı 1.3'e yükselmiştir. Aşım artmıştır. Bununla birlikte, sistemin yükselme zamanı, yerleşim zamanı ve kalıcı durum hatası azalmıştır.

Orantı-Türev Kontrol (PD etki)

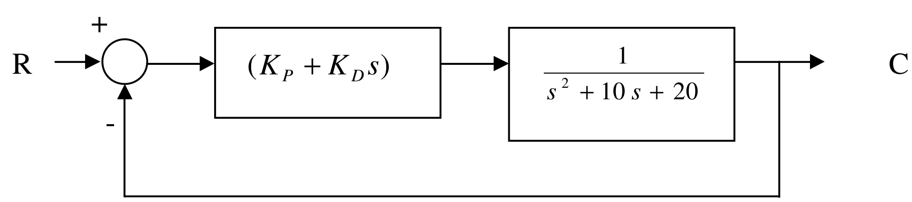
Türev kontrol hatanın türevini alarak bir kontrol sinyali üretir. Dolayısıyla kalıcı durum hatası üzerinde bir etkisi yoktur, çünkü sabit bir sinyalin türevi sıfırdır. Bu yüzden, türev etki kontrolörlerde yalnız başına kullanılmaz diğer etkilerle beraber kullanılır.

Orantı-Türev kontrolörler her iki denetim etkisinin özelliklerini taşırlar. Sistemin transfer fonksiyonu (10.2) denkleminde olduğu gibidir.

$$(K_P + K_D s) \qquad (10.2)$$

Örnek 3 : Transfer fonksiyonu örnek 1'de verilen sistemin orantı-türev kontrolör kullanılarak birim basamak cevabını elde ediniz.

Çözüm : Öncelikle orantı-türev kontrolörle sistem aşağıda olduğu gibidir. Oransal kontrolör kazancı Kp=300 ve K_D=10 alınır.



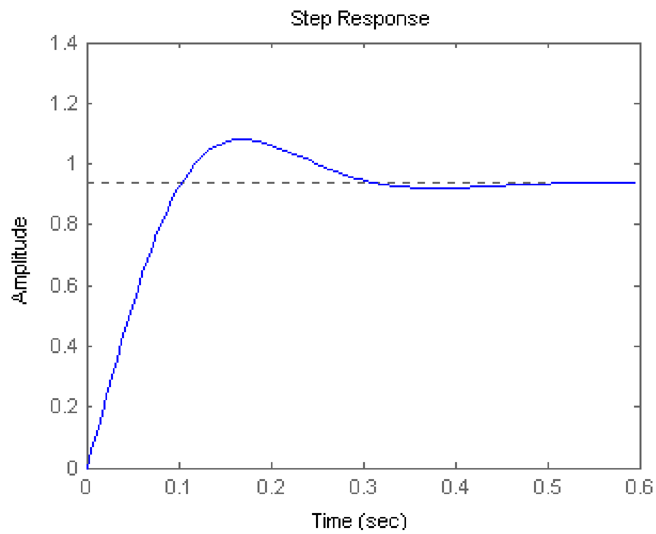
Sistemin transfer fonksiyonu aşağıda olduğu gibidir.

$$T(s) = \frac{K_P + K_D s}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Sistemin MATLAB' da yazılımı ve birim basamak cevabı aşağıda olduğu gibidir.

```
>> Kp=300;
>> Kd=10;
>> sys=tf([Kd Kp],[1 (10+Kd) (20+Kp)])

Transfer function:
  10 s + 300
----------------
s^2 + 20 s + 320

>> step(sys)
```

Yukarıdaki komut satırları icra edildiğinde Şekil 10.3'teki grafik elde edilir.



Şekil 10.3 örnek 3'deki sistemin Kp=300 ve Kd=10 için birim basamak cevabı

Orantı-İntegral Kontrolör (PI etki)

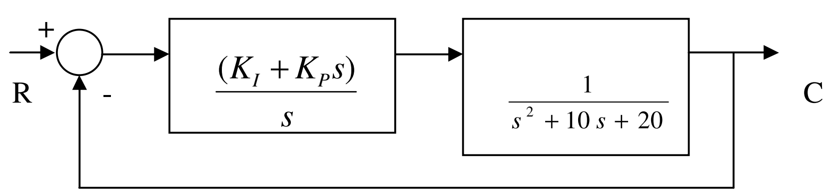
İntegral kontrolör, hata değeri sabit bir değerde kalmışsa bu hatayı gidermek üzere giderek artan bir kontrol sinyali üreterek sistem çıkışının referans değere ulaşmasını sağlar. Hata sıfır olduğunda integral çıkışı da sıfır olur. **[4]**

Orantı-Türev kontrolörler her iki denetim etkisinin özelliklerini taşırlar. Sistemin transfer fonksiyonu (10.3) denkleminde olduğu gibidir.

$$\frac{(K_I + K_P s)}{s} \tag{10.3}$$

Örnek 4 : Transfer fonksiyonu örnek 1'de verilen sistemin orantı-integral kontrolör kullanılarak birim basamak cevabını elde ediniz.

Çözüm : Öncelikle orantı-integral kontrolörle sistem aşağıda olduğu gibidir. Orantı kontrolör kazancı Kp=30 ve K_I =10 alınır



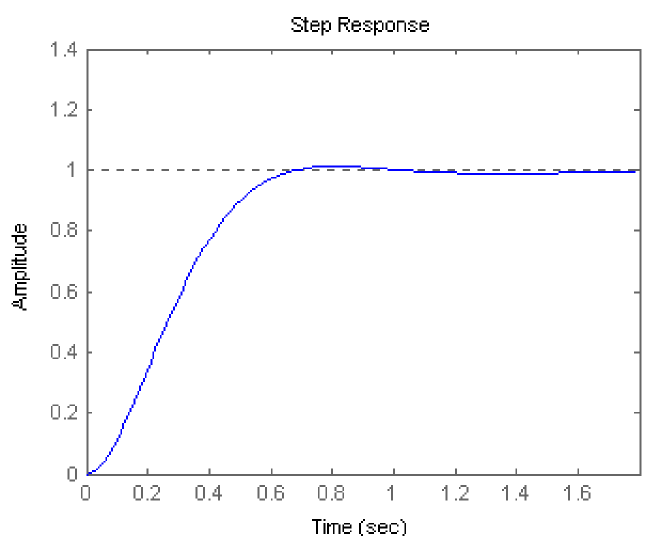
Sistemin transfer fonksiyonu aşağıda olduğu gibidir.

$$T(s) = \frac{K_I + K_P s}{s^3 + 10s^2 + (20 + K_P)s + K_I}$$

Sistemin MATLAB' da yazılımı ve birim basamak cevabı aşağıda olduğu gibidir.

```
>> Kp=30;
>> Ki=70;
>> sys=tf([Kp Ki],[1 10 (20+Kp) Ki])

Transfer function:
      30 s + 70
------------------------
s^3 + 10 s^2 + 50 s + 70

>> step(sys)
```

Yukarıdaki komut satırları icra edildiğinde Şekil 10.4'teki grafik elde edilir.



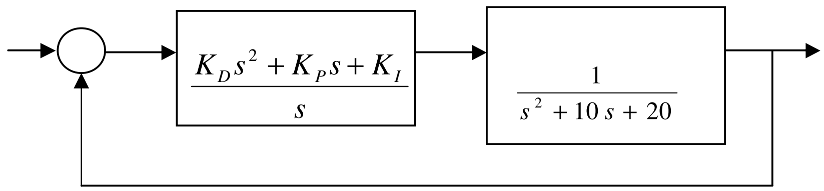
Şekil 10.4 örnek 4'deki sistemin Kp=30 ve K_I =70 için birim basamak cevabı

Orantı-İntegral-Türev Kontrolör (PID etki)

Orantı-İntegral-Türev kontrolörler her üç denetim türü etkisinin özelliklerini taşırlar. Sistemin transfer fonksiyonu (10.1) denkleminde olduğu gibidir. **[4]**

Örnek 5 : Transfer fonksiyonu örnek 1'de verilen sistemin orantı-integral-türev kontrolör kullanılarak birim basamak cevabını elde ediniz.

Çözüm : Öncelikle orantı-integral kontrolörle sistem aşağıda olduğu gibidir. Orantı kontrolör kazancı Kp=350 , K_I =300 ve K_D =50 alınır.



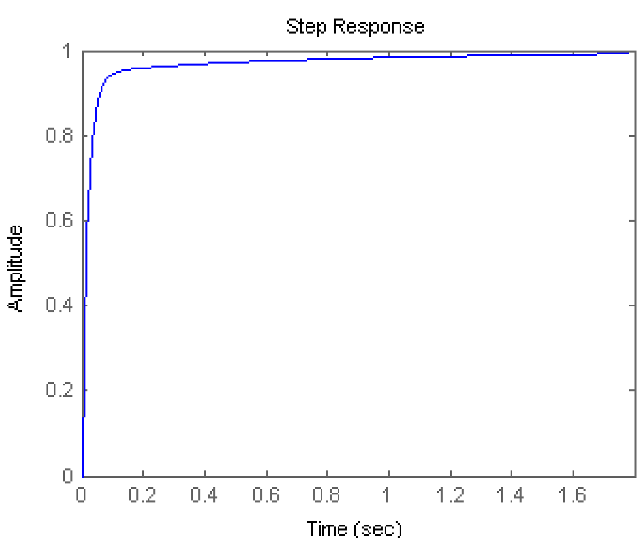
Sistemin transfer fonksiyonu aşağıda olduğu gibidir.

$$T(s) = \frac{K_D s^2 + K_P s + K_I}{s^3 + (10 + K_D)s^2 + (20 + K_P)s + K_I}$$

Sistemin MATLAB' da yazılımı ve birim basamak cevabı aşağıda olduğu gibidir.

```
>> Kp=350;
>> Ki=300;
>> Kd=50;
>> sys=tf([Kd Kp Ki],[1 (10+Kd) (20+Kp) Ki])

Transfer function:
  50 s^2 + 350 s + 300
--------------------------
s^3 + 60 s^2 + 370 s + 300

>> step(sys
```

Yukarıdaki komut satırları icra edildiğinde Şekil 10.5'teki grafik elde edilir.



Şekil 10.5 örnek 5'deki sistemin Kp=350 , K_I=300 ve K_D=50 için birim basamak cevabı

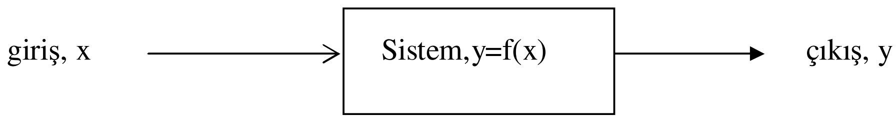
Şekil 10.5'te görüldüğü gibi aşım ve kalıcı hal hatası ortadan kalkmıştır. Yükselme zamanı ise çok küçüktür.

BÖLÜM 11

11 SİMULİNK'E GİRİŞ

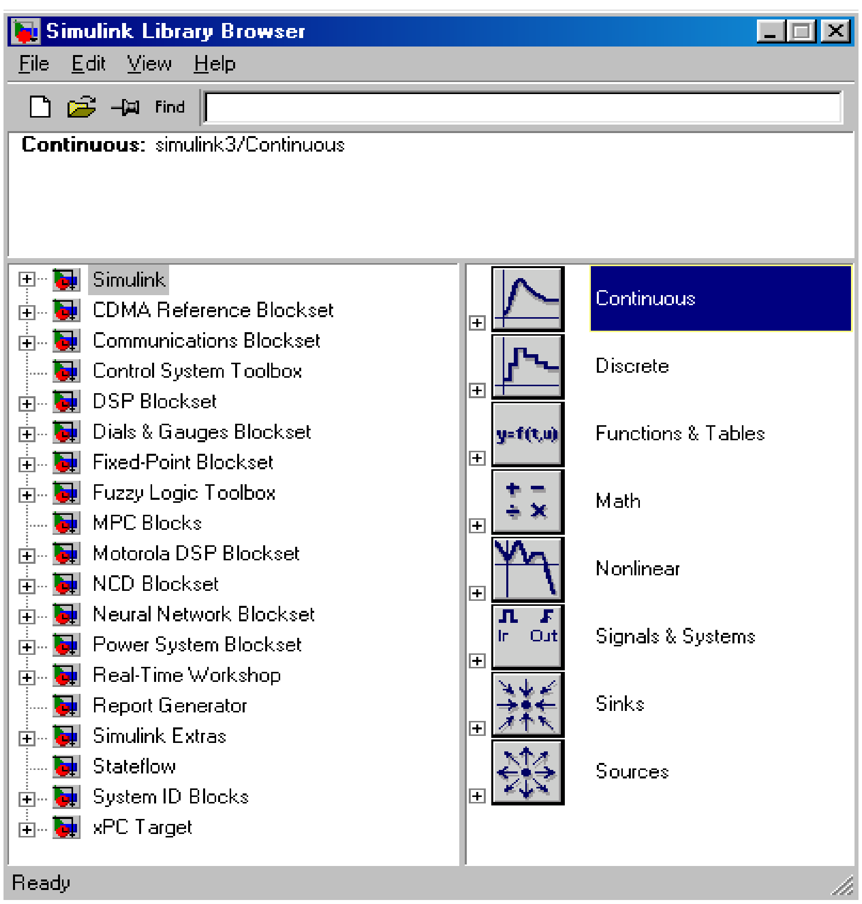
11.1 Temel Bilgiler

Simulink, MATLAB'ın bir uzantısı olup, blok diyagramlarla doğrusal ve doğrusal olmayan dinamik sistemlerin simülasyonunda menülerle çalışan bir grafik arayüz kullanır. Bir blok diyagramı, genellikle bir giriş, sistemin kendisi ve bir çıkıştan ibarettir. Blok diyagramının grafik gösterimi Şekil 11.1'de gösterilmiştir. **[2]**



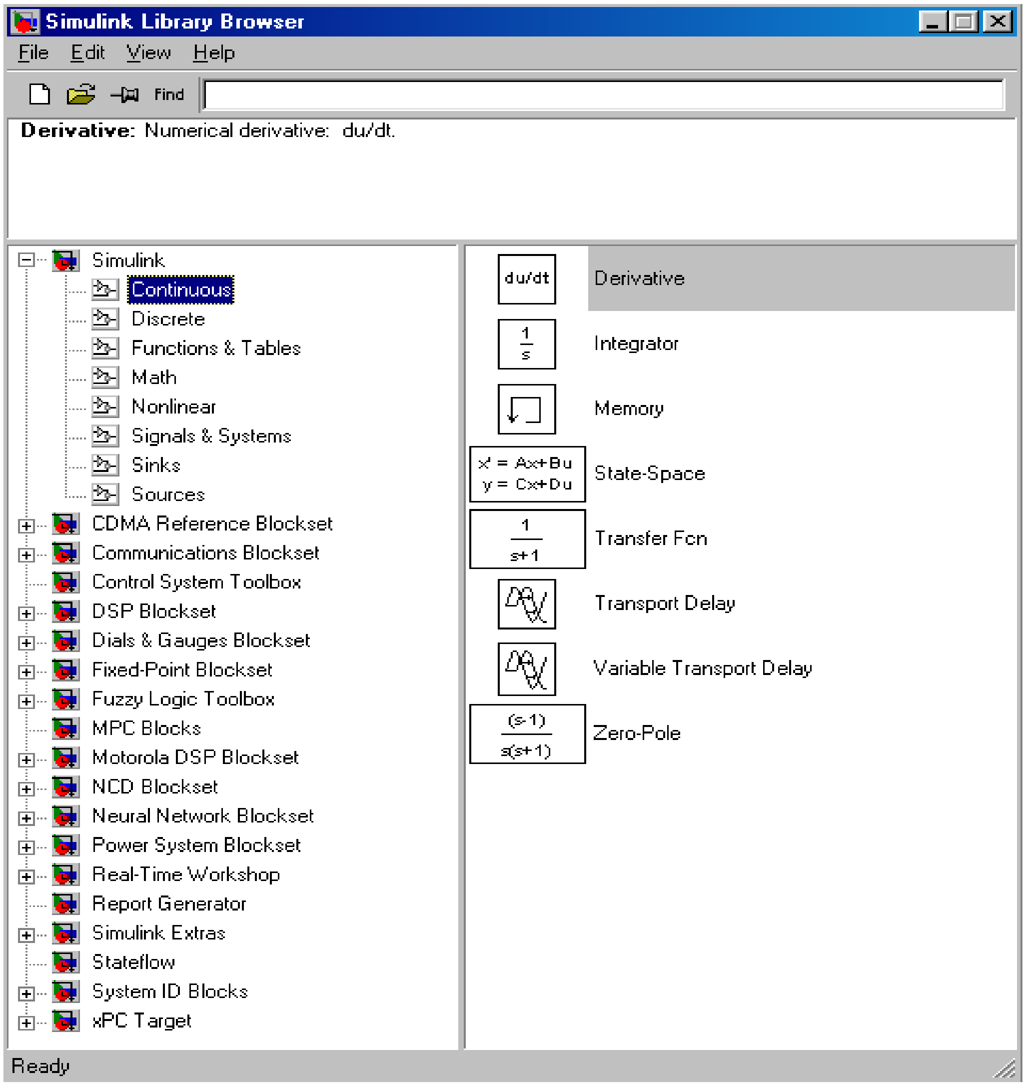
Şekil 11.1 Simulink'te blok diyagramı gösterimi

Simulink, kullanıcının karmaşık sistemlerin modellerini zahmetsizce kurmasına ve onları doğrudan bir MATLAB programı yazıp hatalarını düzeltmek zorunda kalmaksızın analizine imkan tanır. Simulink, MATLAB ile birlikte çalıştırılmasına rağmen, MATLAB komutlarının programlarının veya programlama kurallarının çok iyi bilinmesini gerektirmez. Simulink daha özelleştirilmiş bir yazılım olması nedeniyle MATLAB kadar genel ve güçlü değildir; fakat dinamik analizin pek çok tipi için kullanımı çok kolaydır ve verimli kullanmak için de genel olarak çok az bilgisayar ve programlama tecrübesi gerektirir.



Şekil 11.2 Simulink kütüphaneleri **[1]**

Simulink, sadece MATLAB içinde kullanılabilir. Dolayısıyla bir MATLAB oturumu başlatıldıktan sonra, ya açılan MATLAB komut penceresinden 'simulink' komutu girilir veya bu pencerenin üst kısmında görülen 'Simulink Library Browser' simgesine tıklanarak ulaşılabilir. Simulink başlar başlamaz kütüphane (library) isimlerini ihtiva eden Şekil 11.2'dekine benzer bir pencere açılır (sizin farklı/ilave kütüphane adlarına sahip olabilmeniz hali hariç). Bir kütüphane, dinamik modeller yapmak için kullanılan blokların topluluğudur.



Şekil 11.3 Simulink blokları ve alt blokları

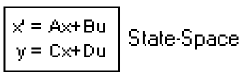
Simulink blokları sınıflara ve alt sınıflara ayrılmıştır. Alt sınıflar da başka alt sınıflara ayrılmaktadır. Bu alt sınıflan açmak ve kullanmak için bir kütüphane adının önündeki artı işaretinin tıklanması gerekir. Simulink kütüphanesinin önündeki artı işaretine basıldığında blok kütüphaneleri menüsünü içeren diğer bir pencere açılır (Şekil 11.3'e bakınız). Bu alt sınıflara tıklandığında bir dizi blok adları kullanıcıya sunulur. Bu bloklar -mesela contunious (sürekli zaman)- alt bloklara sahiptirler ve bu alt bloklardan kullanılacaklar bir simulink model ortamına fareyle sürüklenerek taşınabilirler. Yeni bir simulink model ortamının açılması için ‘Simulink Library Browser’ penceresinden *File>New>Model* seçilmesi gerekmektedir.

11.2 Sık Kullanılan Simulink Blokları [1]

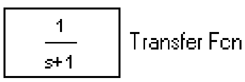
11.2.1 Sürekli Zaman Blokları (Continuous)

: Giriş sinyalinin zamana göre türevini alır.

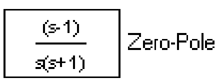
: Giriş sinyalinin zamana göre integralini alır.



: Çok girişli ve çok çıkışlı doğrusal bir sistemin durum uzayı modeli



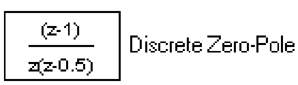
: Doğrusal bir sistemin transfer fonksiyonu modeli



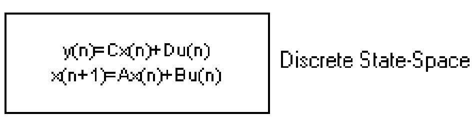
: Doğrusal bir sistemin sıfır-kutup-kazanç modeli

11.2.2 Ayrık Zaman Blokları (Discrete)

: Ayrık zamanlı transfer fonksiyonu modeli

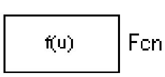


: Ayrık zamanlı Sıfır-kutup-kazanç modeli

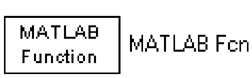


: Ayrık zamanlı durum denklemi modeli

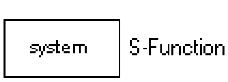
11.2.3 Fonksiyon ve Tablolar (Functions & Tables)



: Bu blok, matematik ifadeler için fonksiyon oluşturmaya yarar, ‘u’ harfi giriş sinyali için kullanılmaktadır. Örnek bir ifade ‘tan(u[l])*exp(u[2] }’ olabilir; burada u[l] and u [2] sırasıyla birinci ve ikinci giriş verileri kümelerini temsil etmektedir.

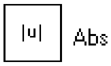


: Bir MATLAB fonksiyonuna giriş değerlerini aktarır. Bu fonksiyon MATLAB'ın hazır bir fonksiyonu veya kullanıcı tarafından yazılmış bir M-fonksiyonu olabilir.



: Kullanıcı tarafından tanımlanan bir bloktur. S-function, m-file,c,ada yada fortran dillerinde yazılabilir fakat s-function standartlarında olmak zorundadır. Kendisine ait dört adet değişkeni vardır. İstenirse kullanıcı değişken ekleyebilir.

11.2.4 Matematik Blokları (Math)

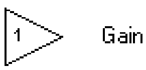


: Giriş değişkeni olan ‘u’ nun mutlak değerini alır.

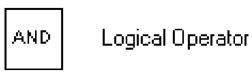
: Giriş değişkeninin büyüklüğünü ve açısını hesaplar

: Girişin sanal ve gerçek kısımlarını çıkışa verir

: Giriş vektörlerinin nokta çarpımlarını hesaplar



: Kazanç sabiti. İstenilen bir değer atanabilir.

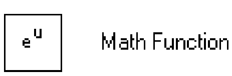


: Ve, Veya, Değil gibi bir dizi mantık işlemlerini gerçekleştirmek için kullanılabilen bir blok.

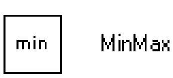
: ...den küçük,...e eşit veya ...den büyük gibi bir dizi ilişki işlemlerini uygulamak için kullanılabilen bir blok.

: Girişe uygulanan büyüklük ve açı değişkenlerini birleştirerek çıkışa verir.

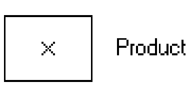
: Sanal ve gerçek olan iki girişi birleştirerek çıkışa verir.



: exp, sin, sqrt v.s. gibi bir dizi matematik fonksiyonları icra etmek üzere ayarlanabilen bir blok

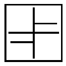


: Girişlerin minimumunu yada maksimumunu çıkışa verir.

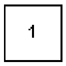


: Bir çarpma veya bölme yapmak için kullanılabilen blok (Bölme bölenin tersiyle yapılan bir çarpma olarak düşünülmelidir).

: Girişi yuvarlamak için kullanılır. MATLAB’ da kullanılan dört yuvarlama foksiyonuda kullanılabilir.

 Sign

: Çıkış değeri olarak; Pozitif giriş için l, negatif giriş için -l ve sıfır giriş için 0 değerini verir. Bu MATLAB' daki sign(x) fonksiyonuna benzerdir.

 Slider Gain

: Çalışma anında değiştirilebilen kazanç.

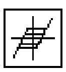
 Sum

: Girişlerin toplamını veya farkını veren bir blok. Girişlerin sayısı ve her bir girişe uygulanacak işaret, blok diyalog kutusunda ayarlanabilir.

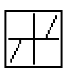
 Trigonometric Function

: Sinüs ve kosinüs gibi standart trigonometrik fonksiyonları tatbik için kullanılabilen bir blok

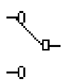
11.2.5 Doğrusal Olmayan Bloklar (Nonlinear)

 Backlash

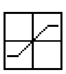
: Ölü-band genişliği sistemin durumuna göre yapılandırılır

 Dead Zone

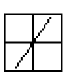
: Ölü-band bölgesinde giriş için çıkış sıfırdır

 Manual Switch

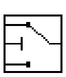
: Çalışma esnasında üzerine fareyle çift tıklanarak konum değiştiren anahtar.

 Saturation

: Sinyalin alt ve üst değerlerini sınırlanmış haliyle çıkışa verir

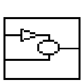
 Rate Limiter

: Giriş sinyallerinin değişimlerini sınırlar

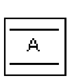
 Switch

: ‘2’ nolu giriş eşikten büyük yada eşitse ‘1’ nolu girişteki sinyal çıkışa verilir. Diğer koşullarda ‘3’ nolu giriş çıkışa verilir.

11.2.6 Sinyaller ve Sistemler (Signals & Systems)

 SubSystem

: Birden fazla bloğun tek bir blok içinde toplanmasını sağlar

 Data Store Memory

: Bir hafıza bölgesi tanımlar

: ‘Data store memory’ ile tanımlanan hafıza bölgesinden veri okumak için kullanılır.

: ‘Data store memory’ ile tanımlanan hafıza bölgesine veri saklanması için kullanılır.

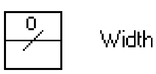
: Bir giriş sinyal vektörünü sonlu sayıda skaler çıkış sinyallerine ayıran blok (De-Multiplex için)

: Sonlu sayıda skaler giriş sinyallerini bir çıkış sinyali matrisi üretecek tarzda birleştiren blok (Multiplex için).

: Bir alt-sistem için giriş portu sağlar

: Bir alt-sistem için çıkış portu sağlar

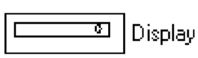
: Bir alt-sistem içerisinde tetikleme al-sistemi oluşturmak için kullanılır



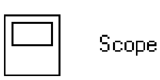
: Giriş sinyalinin genişliğini çıkışa verir.

11.2.7 Kuyu Blokları (Sinks)

Kendisine veri giren ama çıkışı olmayan bloklar burada yer alır.



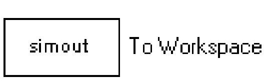
: Giriş sinyalinin o anki değerini gösterir.



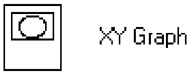
: Skaler veya vektör sinyallerini osiloskoptakine benzer tarzda grafik olarak gösteren bir blok.

: Giriş sinyali sıfırdan farklı olduğunda simülasyonu ‘u durduran blok

: Zamanı MAT dosyası olarak saklar

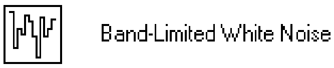


: Bir giriş sinyalini, MATLAB çalışma alanında, simülasyon bittikten sonra, erişilebilir bir MATLAB matrisinde depolayan blok.



: İki skaler girişi kullanarak bir grafik çizdiren blok. Üstteki giriş kapısına bağlanan sinyal bağımsız değişken (x ekseni) ve alttakine bağlanan ise bağımlı değişkendir (y ekseni).

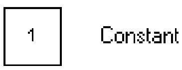
11.2.8 Kaynak Blokları (Sources)



: Rasgele ses sinyali üretir

: Zamana göre artan doğrusal bir sinyal üretir

: Mevcut simülasyon zamanından ibaret bir sinyal bloğu.



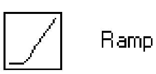
: Sabit bir sayısal değer üreten blok. Sabit, bir skaler veya vektör olabilir.

: Çalışma alanından değer okumak için kullanılır

: Simülasyonun çalışma anında bir MAT dosyasından zaman ve giriş verilerini okur

: Ayrık zamanlı puls jeneratörü

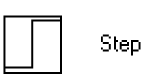
: Sürekli zamanlı puls jeneratörü



: Düzgün artan veya azalan bir sinyal üreten blok

: Sinyal jeneratörü. Çeşitli dalga şekillerini üreten blok.

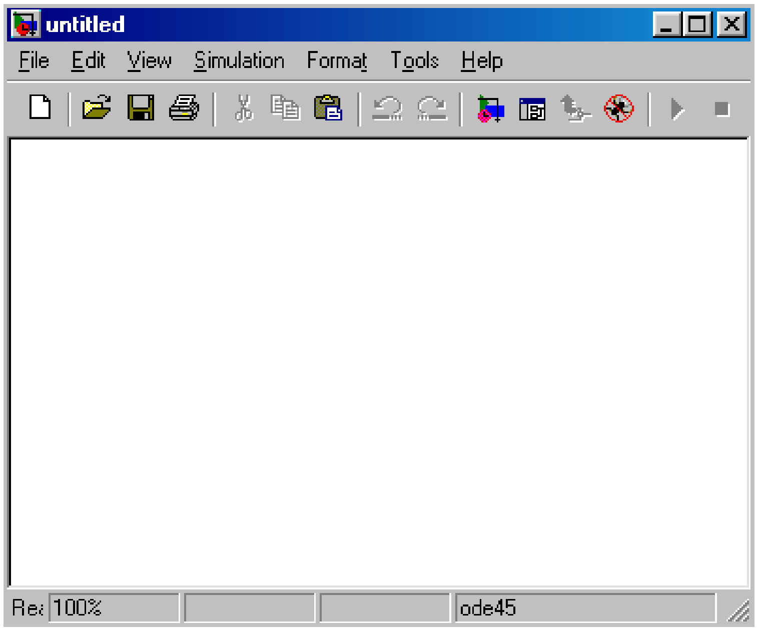
: Sinyal Üretici. Dalga şekillerini üreten blok.



: Basamak sinyali üretir

11.3 Model Kurma

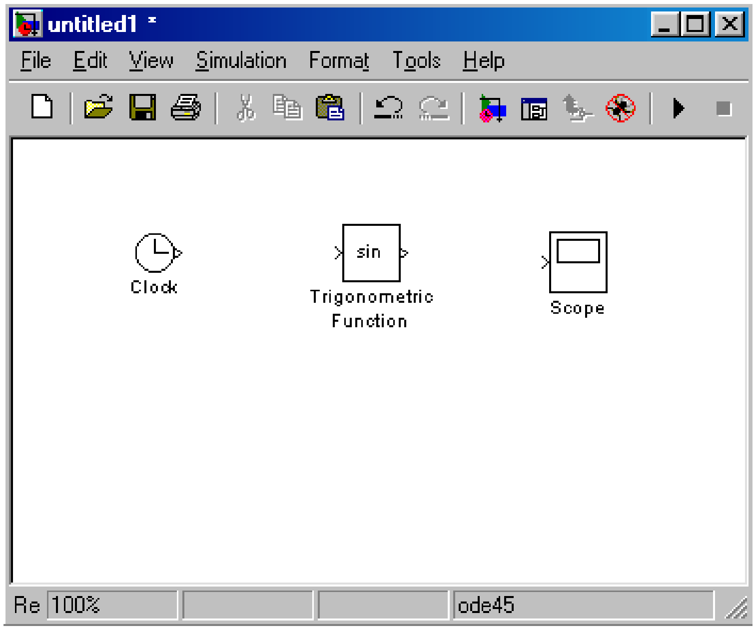
Bir model kurmak ve saklamak için Simulink'in model penceresinin açılması lazımdır. Bu işlem ya MATLAB komut penceresinin üst tarafındaki 'File' menüsünden 'New/Model' komutunu seçerek (Şekil 1.1'e bakınız) veya Simulink Kütüphane Gezgini'nin (Simulink Library Browser) üst kısmında 'yeni bir model oluştur' düğmesine tıklayarak gerçekleştirilebilir. Bu işlem neticesinde aşağıda Şekil 11.4'dekine benzer bir pencere açılacaktır. **[2]**



Şekil 11.4 Simulink Model Penceresi

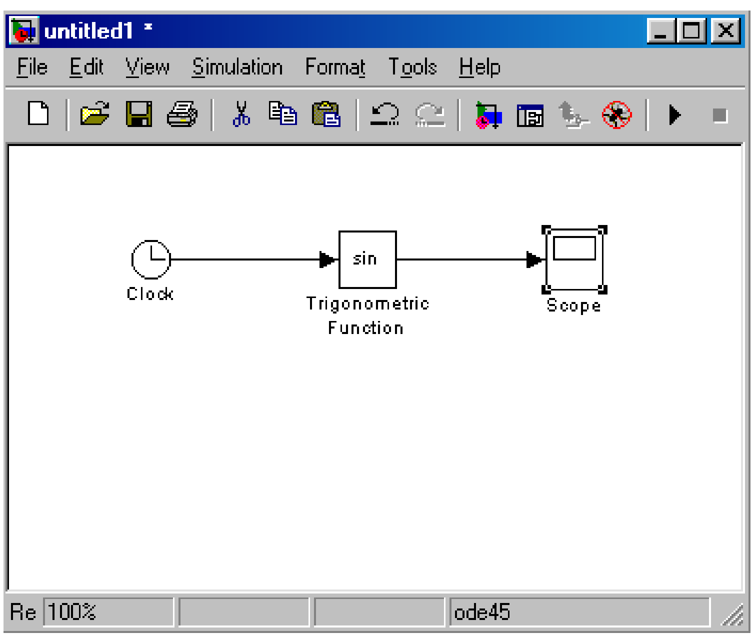
Kurulan modeli saklamak için yine 'File' menüsünden 'Save' veya 'Save As..' komutu seçilebilir ve ona bir isim atanabilir.
Örnek olarak bu bölümde, sinüs fonksiyonunun grafiğini elde etmeye yöneliktir model oluşturulacaktır. Sinüs grafiğini verecek bir blok diyagramını elde etmek için bazı blokların bu model penceresi içine sürüklenmesi gerekir.

Kaynak Blokları, Matematik Blokları ve Kuyu Blokları (Sources, Math and Sinks) sırasıyla Clock, Trigonometric Function, and Scope bloklarını kullanmaktır. Burada aynı problem için aynı sonucu verecek bir dizi kombinasyon daha olduğunu kaydedelim; fakat bu kullanıcının modeli oluşturmadaki tercihiyle ilgili bir meseledir.
Bir kütüphaneden bir bloğu model penceresine taşımak için evvela, blok, farenin sol düğmesiyle işaretlenir ve sonra model penceresine sürüklenir. Bu, basit bir 'sürükle ve bırak' işlemidir. Yukarıda bahsedilen bloklar, birbiri ardına model penceresine sürüklendiğinde model penceresi aşağıda Şekil 11.5'teki gibi görünür.



Şekil 11.5 Bazı bağlantısız blokları içeren bir model

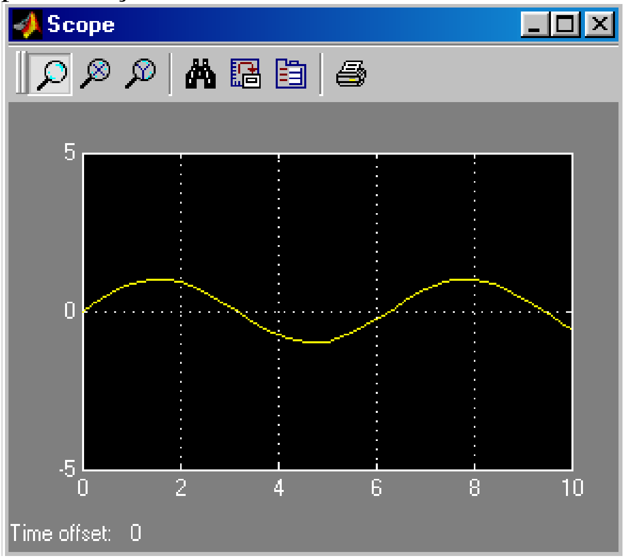
Blokları birbirine bağlamak için her bir bloğun kenarlarındaki küçük oklar kullanılmalıdır. Bir bloğun çıkış okuna farenin sol düğmesiyle tıklanıp bağlantı hattının diğer bir bloğun giriş okuna birleşene kadar sürüklenmesi bu iki blok arasında bir sinyal transferinin olmasıyla neticelenir. Bağlantı yapıldıktan sonra okların görüntülerindeki değişikliğe Şekil 11.6'da dikkat ediniz. Bir bloğun yeri onu basitçe farenin sol düğmesiyle tutup etrafta gezdirilerek değiştirilebilir. Herhangi bir blok veya bağlantı hattı silinmek istenirse önce hatta veya bloğa tıklanır ve sonra klavyede 'delete' tuşuna basılır.



Şekil 11.6 Bloklar bağlandıktan sonraki Simulink modeli

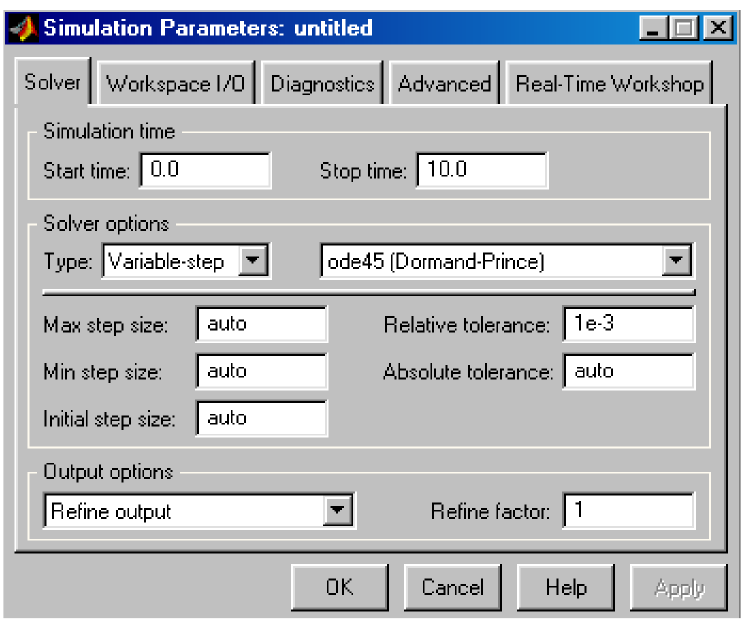
11.4 Simülasyonları Çalıştırma

Model bir kez kurulduktan sonra simülasyon model penceresinin üst kısmındaki ‘Simulation’ menüsünden ‘Start’ seçilerek başlatılabilir. Mevcut modelle hiçbir şey olmayacakmış gibi görünmektedir; zira ekranda açık hiçbir gösterge yoktur. Ancak, skop (scope) bloğuna çift tıklanırsa sinüs fonksiyonunun grafiğinin izlenebileceği Şekil 11.7'dekine benzer küçük bir pencere açılacaktır.



Şekil 11.7 Skope penceresi

Şekilde görüldüğü gibi simülasyon 10 saniye sonra sona ermiştir. Bu süre simülasyon durma zamanı için programda seçilmiş (default) bir süredir, fakat bu süre ‘Simulation’ menüsünden açılan ‘Simulation Parameters’-parametre penceresi- diyalog kutusuna girilen herhangi bir değerle ayarlanabilir (Şekil 11.8). **[2]**



Şekil 11.8 ‘Simulation Parameters’ diyalog kutusu

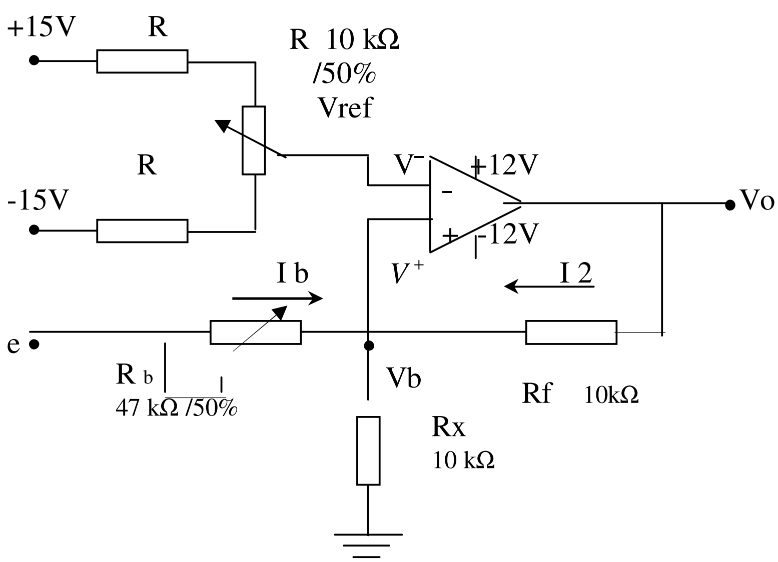
BÖLÜM 12

12 AÇ-KAPA TİPİ KONTROLÖRÜN SİMULİNKTE MODELLENMESİ

Otomatik Kontrol Dersi uygulamasının amacı, günlük hayatta ve endüstride kullanılan kontrol sistemlerinin uygulama becerilerinin öğrenciye kazandırılması olarak özetlenebilir. Dersin içeriğinde aç-kapa (On-Off) tipi, orantı tipi, integral tipi, orantı artı integral tipi,orantı artı türev tipi, orantı artı integral artı türev tipi kontrol sistemleri bulunmaktadır. Bu kontrol sistemleri arasıda yaygın olarak kullanılanlardan biri aç-kapa tipi kontrolördür. Aç-kapa tipi kontrolörler belirlenmiş olan iki seviyenin altında veya üstünde çalışırlar. Belirlenmiş olan bu iki seviye arlığına histerezis denmektedir. Yaptığımız çalışmada, histerezis genliği değişimi, kontrolörün girişinde bulunan direnç ile kontrol edilmektedir. Elde edilen analitik çözüm, deneysel olarak incelenmiş ve sonuçta müspet sonuç alınarak, simülasyon hazırlamak amacıyla matematik model çıkartılmıştır

12.1 Matematiksel Modelleme

Çalışmaya konu olan sistemin önce modeli geliştirilmiştir. Şekil 12.1'deki devre aç-kapa tipi kontrolör devresidir. Şekilde görüldüğü gibi Aç-kapa tipi kontrolör devresi, referans girişi, hata girişi, OP-AMP ve dirençlerden meydana gelmektedir. Aç-kapa tipi kontrolörün giriş-çıkış karakteristiğinde görülen histerezis pozitif geri besleme ile sağlanmıştır. Üzerinde çalışılan modelde amaç "kontrolörün histerezis genişliğini Rb direnci ile ayarlamak" tır. Bu aşamadan sonra istenilen denklemleri çıkartabilmek için OP-AMP' ın çalışma prensiplerinden faydalanılmıştır. **[10]**



Şekil 12.1 Aç-kapa tipi kontrolör devresi **[10]**

OP-AMP' ın özelliklerinden biri (+) ve (-) giriş uçlarındaki potansiyel fark sıfırdır. Giriş empedansları çok yüksek olduğundan (+) ve (-) giriş uçlarından akan akım nanoamper seviyesindedir[1].

Aşağıdaki (1) ve (2) eşitlikleri OP-AMP' ın çalışma prensipleridir.

$$V^{+} = V^{-} = V_b = 0 \tag{1}$$

$$I = I_b + I_2 \; ; \; I = \frac{V_b}{10} ; \quad I_b = \frac{(e - V_b)}{R_b} ; \quad I_2 = \frac{(V_o - V_b)}{10} \tag{2}$$

Eşitlik (1)ve (2) yardımı ile Vo, (3) eşitliği elde edilir.

$$\frac{V_b}{10} = \frac{(e - V_b)}{R_b} - \frac{(V_o - V_b)}{10}$$

$$V_o = V_b \left(\frac{2 * R_b + 10}{R_b}\right) - \frac{10 * e}{R_b} \tag{3}$$

OP-AMP' ın kazanç eşitliğinden yola çıkılarak VH bulunur.

$$n = \frac{R_f}{R_b} \tag{4}$$

$$n = \frac{[+V_{sat} - (-V_{sat})R_b]}{V_H} \; [1] \quad ; \qquad V_H = \frac{[+V_{sat} - (-V_{sat})R_b]}{n}$$

$$V_H = \frac{[(15V - (-15V))R_b]}{R_f}$$

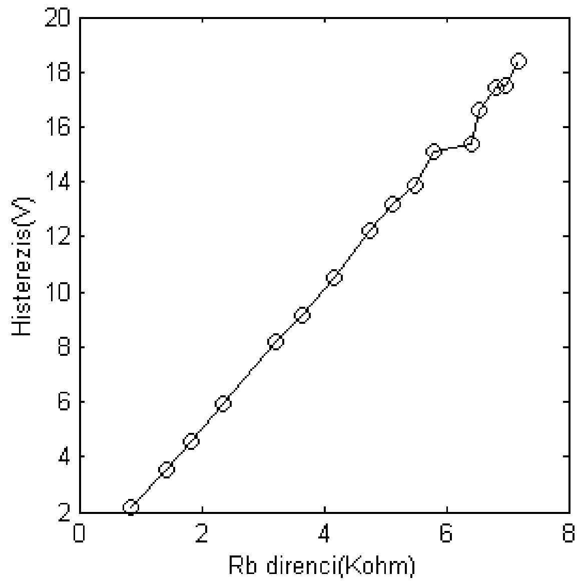
$$V_H = \frac{30 * R_b}{R_f} \tag{5}$$

Pratikte V_b tam olarak sıfır olmaz ve Rx direnci üzerinden küçük bir akım geçer $(I = \frac{V_b}{R_x})$ ve Rx üzerinde gerilim düşümü olur. VH değerini Vref=0 olduğu için az ölçüde etkiler, Yaklaşık $-0.25 * R_b$ (6)

Yukarıda VH, histerezis genliğindeki Δe 'ye, yani aç-kapa tipi kontrolörün bağıl hatasına tekabül etmektedir. Modelimizde R_b direncinin değeri belirlenir ve histerezis genişliği hesaplanır.

12.2 Deneysel Sonuçlar

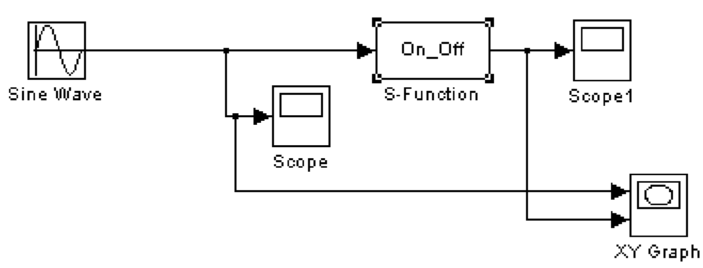
Sistemin matematiksel modeli elde edildikten sonra, sistemin istenilen şekilde davrandığını Şekil 12.1'deki düzeneğin kurulması ve yapılan deneyler sonucunda etkin bir sonuç elde ettiğimizi gördük. Marmara Üniversitesi Otomatik Kontrol Laboratuarında düzeneğimiz kurulmuş. Histerezis genişliği, iki kanallı osilaskop yardımı ile elde edilmiştir. Osialskobun bir probu Vo çıkışına diğer probuda hata girişine alındığında devrenin giriş-çıkış karakteristiği Şekil 12.2'de olduğu gibi elde edilmiştir. Performans için verimli olan bir Rb değer aralığı seçilerek Rb karşılığında ortaya çıkan histerezis genişliği tespit edilmiştir. Elde edilen grafiğin Şekil 12.3 'de görüldüğü üzere doğrusallığı, birçok sistemde, istenilen düzeye uygun olduğu görülmektedir. Yapılan deneylerde Şekil 12.2'deki Rb değerlerinin aşağısındaki ve yukarısındaki değerler için sonuç alınamamıştır.



Şekil 12.2 Rb-Histerezis değişimi

12.3 Aç-Kapa Tipi Kontrolör Simülatörü

Sistemin farklı denetim parametreleri altındaki davranışını elde edebilmek ve öğretim ortamında daha etkin bir şekilde elde edilenleri öğrenciye aktarabilmek için MATLAB yazılımının Simulink modeli kullanılmıştır. Matematiksel modeli ortaya koyabilmek için bir s-function hazırlanmıştır[4]. S-function hazırlanırken MATLAB'in m-file programlama dosyalarından faydalanılmıştır [3]. Aç-kapa tipi Kontrolör Simulink modeli Şekil 3'te görüldüğü gibidir.



Şekil 12.3 Aç-kapa tipi kontrolör simulink modeli

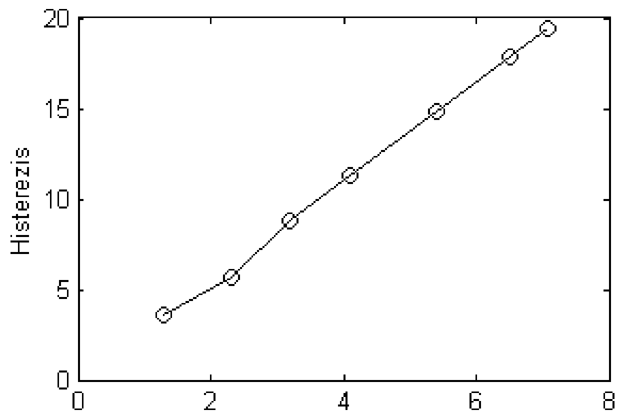
Simulink modelde oluşturulan s-function için kullanılan m-file dosyası aşağıda olduğu gibidir.

```
function [sys,x0,str,ts] = On_Off(t,x,u,flag,Rb,InputOn,InputOff)
```

```
%Variable block
global han
global devam
%/*start up condition and controlling of devam parameter
if t<1
   han=((30*Rb/10)-0.25*Rb)/2;
end
if (u >= han)
devam=1;
end
if (u <= -han)
 devam=0;
end
%end of start up condition
switch flag,
  % Initialization %
  % Initialize the states, sample times, and state ordering strings.
  case 0
    [sys,x0,str,ts]=mdlInitializeSizes(Rb,InputOn,InputOff);
  % Outputs %
  % Return the outputs of the S-function block.
  case 3
      sys=mdlOutputs(t,x,u,InputOn,InputOff,devam);
  case { 1, 2, 4, 9 }
    sys=[];
% Unexpected error handling
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
% Return the sizes, initial conditions, and sample times for the S-function.
function [sys,x0,str,ts] = mdlInitializeSizes(Rb,InputOn,InputOff)
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = -1;
sizes.NumInputs      = -1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];
% end mdlInitializeSizes
% mdlOutputs
% Return the output vector for the S-function
function sys = mdlOutputs(t,x,u,InputOn,InputOff,devam)
if (devam==1)
sys=InputOn;
end
if (devam==0)
```

```
 sys=InputOff;
end
```

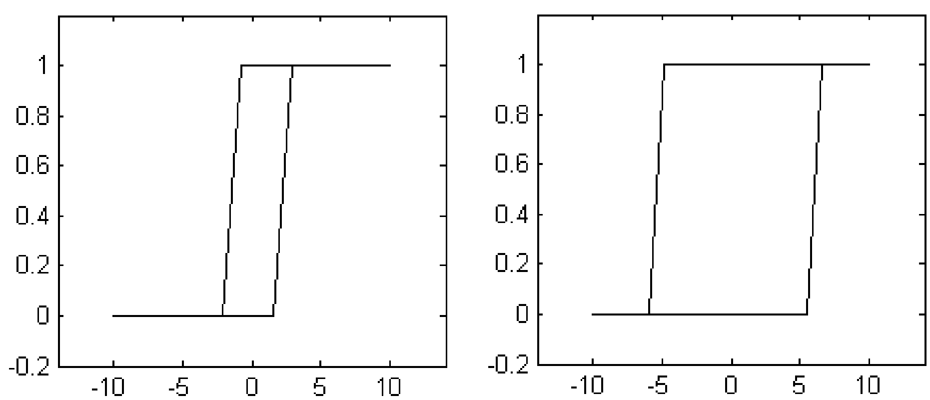
Elde edilen simülasyonda Laboratuar ortamında kullandığımız Rb değerleri kullanarak modelde elde edilen sonuçlar Şekil 12.4'de gösterilmiştir.



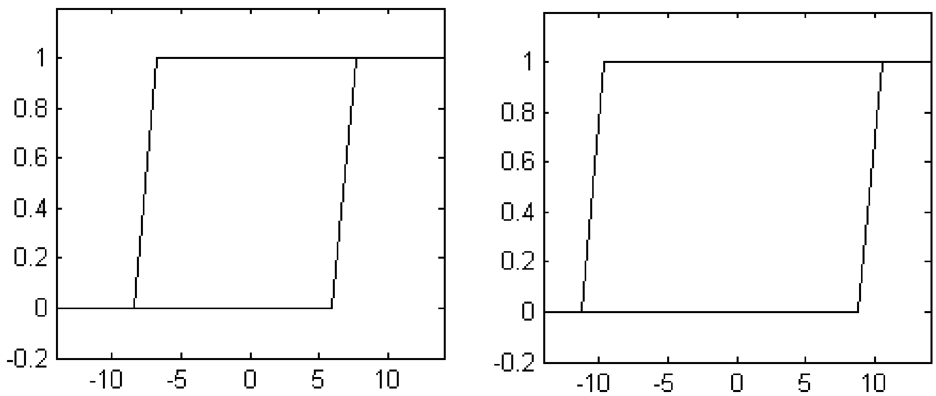
Rb Direnci

Şekil 12.4 Matematiksek modelle birlikte Rb-Histerezis değişimi

Aç-kapa tipi kontrolör simülasyonu, farklı Rb değerleri için çalıştırıldığında Şekil 12.5'daki sonuçlar elde edilmiştir.



a) Rb 1.3 KΩ b) Rb 4.1 KΩ



c) Rb 5.4 KΩ d) Rb 7.07 KΩ

Şekil 12.5 a,b,c,d grafikleri farklı Rb değerleri için , simülasyondan elde edilmiştir.

12.4 Sonuç Ve Değerlendirme

Çalışma sonucu Aç-kapa tipi Kontrolör tasarımına farklı bir yaklaşım kazandırmıştır. Kontrolörün histerezis genişliğinin kontrolör içinde hata sinyalinin uygulandığı bir değişken (devre elemanı,direnç) ile ayarlanması ve istenilen değere çekilmesi sağlanmıştır. Bu kazanım ile birlikte MATLAB/Simulink ortamında yapılan simülasyon sunulmuştur. Simülasyon farklı değerler için etkili sonuçlar vermektedir. Bu şekilde devreyi kurmadan, farklı değerler için, devre analizi yapılabilir. Bununla birlikte, laboratuar uygulamaları öncesi öğrencilerin çalışmalarına görsel bir eğitim imkanı sağlanmaktadır.

KAYNAKLAR

[1]. "MATLAB Help"
[2]. Gündoğdu, Ömer ve Kopmaz, Osman ve Ceviz, M.Akif, "Mühendislik ve Fen Uygulamalarıyla MATLAB", Paradigma Akademi, Bursa-2003
[3]. Yüksel, İbrahim, "MATLAB ile Mühendislik Sistemlerinin Analizi ve Çözümü, Genişletilmiş II. Baskı", Vipaş A.Ş., Bursa-2000
[4]. C. Kuo, Benjamin , Çeviren ve uyarlayan : Bir, Atilla, "Otomatik Kontrol Sistemleri, Yedinci Baskı", Literatür, 1999
[5]. www.mathworks.com internet adresi, 01..25/04/2003
[6]. http://www.mame.mu.oz.au/control/mcg/ctrl301/matlab/ctm/index.html
[7]. http://www-personal.engin.umich.edu/~tilbury/tutorials/me461.html
[8]. http://teal.gmu.edu/~gbeale/examples_421.html
[9]. http://www.math.mtu.edu/math/Content.html
[10]. F. Coughlin, Robert ve F. Driscoll, Frederick, Operational Amplifiers & Linear Integrated Circuits, Fourth Edition, Prentice Hall, Englewood Cliffs, NJ