T.C.
MARMARA ÜNİVERSİTESİ
TEKNİK EĞİTİM FAKÜLTESİ
ELEKTRONİK-BİLGİSAYAR BÖLÜMÜ

KONTROL SİSTEMLERİ İÇİN MATLAB'TE GUI UYGULAMALARI TASARIMI

(LİSANS BİTİRME TEZİ, 2006-2007 ÖĞRETİM YILI)

HAZIRLAYAN **:** 2306826 Kenan SAVAŞ – 4BK

TEZ DANIŞMANI **: Yrd. Doç. Dr. Hasan ERDAL**

İstanbul, 2007

T.C.
MARMARA ÜNİVERSİTESİ
TEKNİK EĞİTİM FAKÜLTESİ
ELEKTRONİK-BİLGİSAYAR BÖLÜMÜ

KONTROL SİSTEMLERİ İÇİN MATLAB'TE GUI UYGULAMALARI TASARIMI

(LİSANS BİTİRME TEZİ, 2006-2007 ÖĞRETİM YILI)

HAZIRLAYAN **:** 2306826 Kenan SAVAŞ – 4BK

TEZ DANIŞMANI **: Yrd. Doç. Dr. Hasan ERDAL**

KONTROL İMZASI **: ……...**

JÜRİ ÜYELERİ **: Yrd. Doç. Dr. Hasan ERDAL**
Yrd. Doç. Dr. Mustafa ONAT
Yrd. Doç. Dr. Özgül VAYVAY

ÖNSÖZ

Bu çalışmada MatLab programı değişik yönleri ile incelenmiştir. Konu ile ilgili literatürde çok az Türkçe kaynak olması ve konuya özellikle bilimsel araştırma yapan akademisyenlerin çoğunun bu bakımdan eksiklik yaşaması düşünüldüğünde böyle bir proje üzerinde çalışma gereği duydum. Bu amaçla konu ile ilgili literatürü çok ayrıntılı bir şekilde taramaya ve elimden geldiğince içeriğin kolay anlaşılır bir şekilde sunulmasına özen gösterdim.

Ayrıca, bu çalışmada Otomatik Kontrol alanı ile ilgili çeşitli amaçlara hizmet etmek üzere Matlab programı kullanılarak tasarlanan pek çok sayıda GUI uygulamalarına yer verilmiştir. Bu tasarımda öğrencilerin görsel uygulamalar sayesinde özellikle Kontrol alanı ile ilgili dersleri daha hızlı ve daha iyi kavramaları amaçlanmıştır. Hazırlanan GUI uygulamaları, grafiksel kolaylık ve basit bir arabirim sunması ile dersin öğrenmesinde öğrencileri komut ezberleme ya da komut ile iş yapma işkencesinden kurtararak dersi kolay anlaşılır, eğlenceli ve pratik bir zeminde öğrenme fırsatı sunacaktır. Uygulama konuları Introduction to Control Systems, Professional Mathematics, Automatic Control I ve Automatic Control II dersleri programında yer alan konular baz alınarak seçilmiştir. Bunların yanında bu çalısmanın sonunda hazırlanan GUI uygulamalarına ait algoritmaların eklenmiş, bunların benzer uygulamaların geliştirilmesi için bir kaynak olacağı düşünülmüştür.

Böyle değerli bir proje çalışmasında kendisi ile çalışmama fırsat verdiği ve başından sonuna kadar desteğini sürdürdüğü için değerli hocam Yrd. Doç. Dr. Hasan ERDAL Bey'e, değerli bilgileri ve yardımlarını esirgemeyen hocam Yrd. Doç Dr. Mustafa ONAT Bey'e, özellikle o değerli zamanlarını ayıran ve ilgisini bir an olsun esirgemeyen Muhammed ÜNAL Bey'e ve almış olduğu doktora dersleri ile ilgili bilgilerini paylaştığı için Önder DEMİR Bey'e teşekkür ederim. Routh kriteri için hazırlamış olduğu tez çalışması ile bana projemde yardımcı olan Hakan AYDIN Bey'e ve bu çalışmayı gerçekleştirmemde bana yardımcı olan ve çalışmalarından yararlandığım ve burada adını anmadığım bütün şahıslara da ayrıca teşekkür ederim

Şu an içinde bulunduğum fırsatların oluşmasında en büyük payı olan ve hayatım boyunca emeklerini ve yüreklerini unutamayacağım ve daima hatırlayacağım değerli annem ve babama şükranlarımı sunarım. Öğrenim hayatımda bu ana gelmemde emeği olan tüm hocalarıma da canı gönülden teşekkür ederim.

Haziran 2007 **Kenan SAVAŞ**

İÇİNDEKİLER

Konu **Sayfa No**

SEMBOL LİSTESİ

ÖZET

Gelişen teknoloji sayesinde günümüzde bilgisayar eğitimin çok önemli bir aracı haline gelmeye başlamıştır. Özellikle uygulamalı derslerin bilgisayarlar eşliğinde simülatörlerle işlenmesi, hem zamanı kısaltmakta, hem de öğretimin daha iyi kavranmasını sağlamaktadır. Bu amaçla hazırlanan bu çalışmada akademik çevreler tarafından yaygın bir biçimde kullanılan Matlab programı çeşitli yönleriyle açıklanmaya çalışılmıştır.

Hazırlanan çalışma toplam yedi bölüm içermektedir. Bölümlerin içerikleri şu şekilde düzenlenmiştir:

- İlk olarak Bölüm-I'de öncelikle Matlab programının kullanımı anlatılmış ve daha sonra Matlab ile ilgili temel bilgiler verilmiştir.

- Bölüm-II'de Matlab programı kullanılarak Kontrol alanı ile ilgili komutlar hakkında bilgiler verilmiş ve örnekler yapılmıştır.

- Bölüm –III'te Matlab programının bir aracı olan Simulink kullanılarak Kontrol problemlerinin nasıl çözüleceği hakkında bilgiler verilmiştir.

- Bölüm-IV'te Matlab programına dışarıdan herhangi bir dosyanın içerdiği verilerin import edilmesi için yapılacak işlemler anlatılmıştır.

- Bölüm-V'te Matlab programı kullanılarak GUI tabanlı uygulamaların nasıl hazırlanacağı hakkında bilgiler verilmiştie. Ayrıca, GUIDE aracı detalı olarak açıklanmıştır.

- Bölüm-VI'da kontrol alanı ile ilgili olmak üzere Matlab programı ile hazırlanan 10 adet GUI uygulaması tanıtılmıştır. Ayrıca, bu alana her bir uygulamaın algoritması eklenmiştir.

- Son olarak Bölüm-VII'de Matlab programı kullanılrak bilgisayar portlarının nasıl yönetileceği ile ilgili bilgiler verilmiştir.

Haziran 2007 **Kenan SAVAŞ**

ABSTRACT

Because of developments of the technology, the computer has been having a key role in education. The fact that subjects with applications that has been instructed by using such a technology leads to saving of time and increase in the quality of the education. As a result of this, this study has been done to describe the computer program of MatLab in detail in a variety of ways.

This study has totally seven captures. Tha data about these is given below.

- In the first capture, Capture 1 , Using of MatLab from beginner level and some some basic tips are given.
- Capture 2 has explanation to show how the problems related to Control Sciense can be solved by using Matlab commands.
- In Capture 3 is given the information of Simulink that is a tool of Matlab in the perspective of Control Science.
- In Capture 4, there can be found information to solve problems and find how systems is response related to Control Science.
- In Capture 5, How GUI applications can be designed using Matlab is described in detail.
- In Capture 6, ten GUI applications designed using Matlab for the problems related to Control Science is explained.
- And the last chapter, Chapter 7, some information can be given about how the ports of computers can be manged using Matlab.

June 2007 **Kenan SAVAŞ**

BÖLÜM-I
MATLAB'E GİRİŞ

1.1 Temel Bilgiler

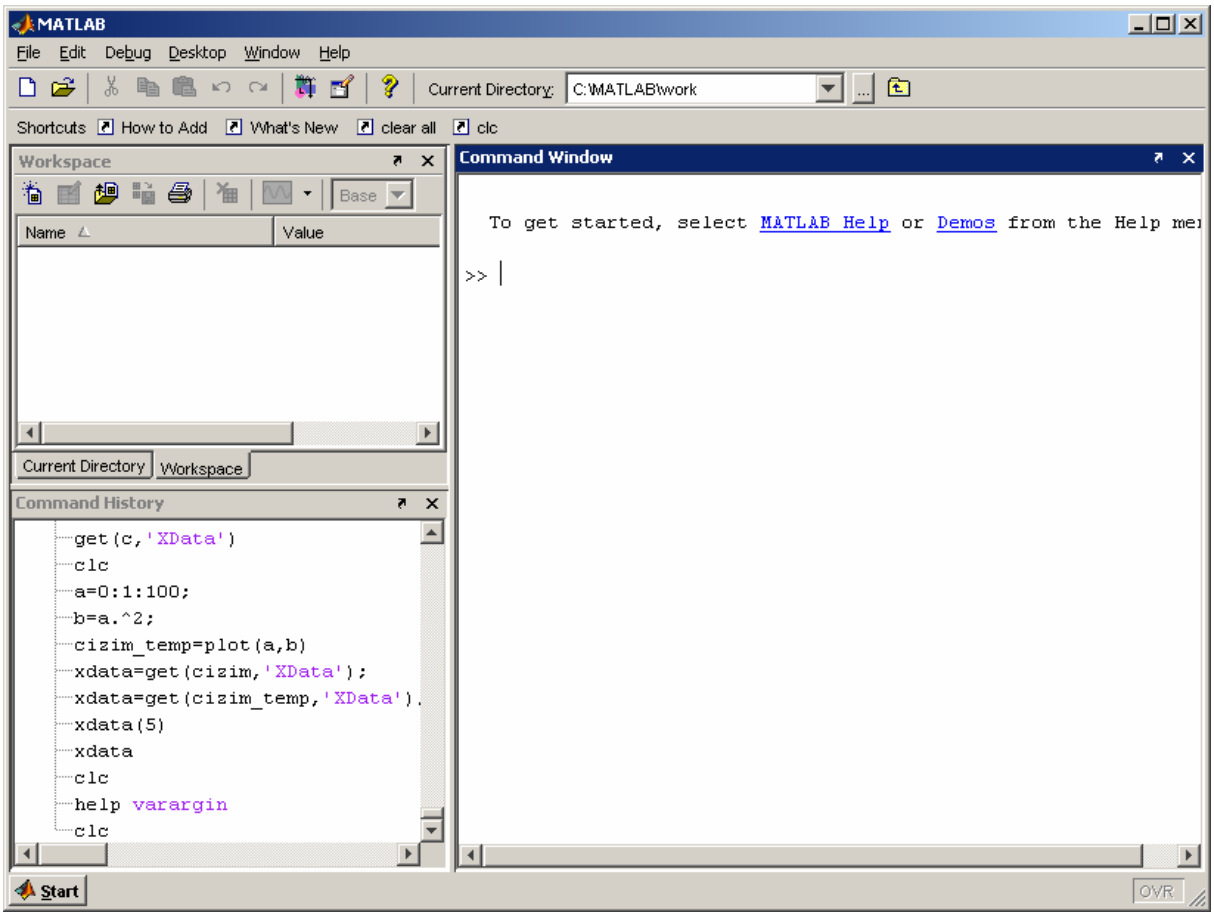
MATLAB kelime itibari ile MATrix LABoratory kelimelerinin kısaltılması ile oluşmuştur. Bu program ilk geliştirildiğinde amaç matris işlemlerinin kullanıcılar tarafından kolaylıkla yapılmasını sağlamaktır. Matlab, geliştirilmesi sonucu günümüzde basit matematiksel hesaplamalardan karmaşık analizlere varan çok çeşitli alanlarda kullanılabilir hale gelmiştir. Bu nedenle son zamanlarda Matlab özellikle bilimsel araştırmalar için tercih edilen ve popüler olarak kullanılan bir ortam haline gelmiştir.

Matlab'in bu denli popüler oluşunun altında sunduğu çok çeşitli komutların yanısıra, grafiksel arabirime sahip oluşu, kolay alışılabilir ve kullanışlı bir ortam etkileşimi sunması, çok çeşitli alanlara (örneğin Kontrol Bilimi, İnşaat Mühendisliği…gibi) hizmet eden farklı ve zengin kütüphanesinin olması yatmaktadır.

Bu proje çalışmasında MatLab'in 7.0.4 versiyonu kullanılmıştır. Ayrıca, anlatımlarda kullanıcıların Windows ayarlarının farenin sol tuşunu tek ve çift tıklama amaçlı kullandıkları varsayılmış, yani sağ elini kullananlar baz alınmıştır.

1.2 Matlab Ortamı ve Pencereler

Windows ortamında Matlab yazılımını başlatmak için Başlat Menüsünde veya Masaüstünde yer alan MatLab ikonunu tıklamak yeterlidir. Matlab açılınca karşımıza Şekil 1.1'deki gibi bir pencere gelecektir.



Şekil 1.1 Matlab Ana Penceresi

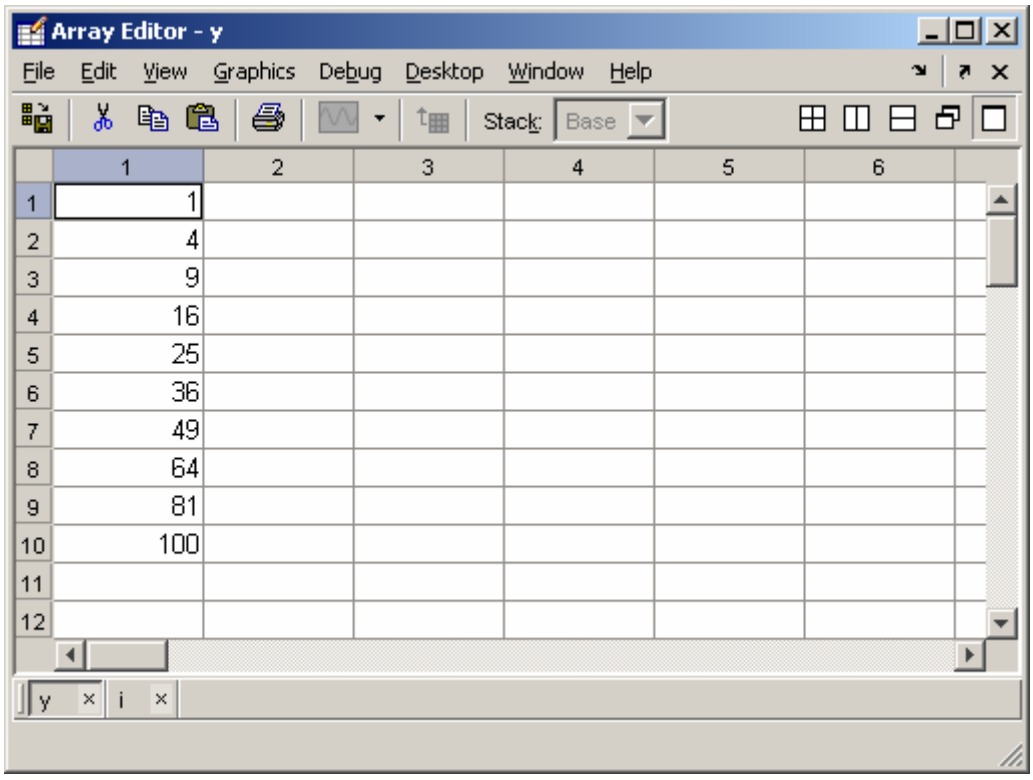
Açılan bu pencere kendi içinde çok değişik işlevleri bulunan ve kullanıcının Matlab'i rahat kullanmasını sağlayan şu pencerelerden oluşur.

- **Array Editor**
- **Command History**
- **Command Window**
- **Current Directory**
- **Demos**
- **Help**
- **Workspace**

Bu pencereleri görevleri şu şekildedir:

1.2.1 Array Editor Penceresi

Kullanıcı workspace penceresindeki herhangi bir değişkenin üzerinde çift tıkladığında Şekil 1.2'de görülen Array Editor penceresi ile karşılaşır. Bu pecere yardımıyla seçilen herhangi bir



Şekil 1.2 Array Editor Penceresi

değişkenin içeriği görülebileceği gibi yine aynı değişkenin içeriği bu pencere yardımıyla değiştirilebilir. Ancak, bazı farklı tipteki değişkenlerin içeriğini değiştirmek mümkün değildir. Bu durum komut kullanılarak gerçekleştirilebilir.

1.2.2 Command History Penceresi:

Bu pencere kullanılan tüm komutların geçmişini tutmak için kullanılır. Komut ekranından girilen her komut Matlab açıldığında geçerli zaman başlığı altındaki listeye eklenerek bu pencerede görüntülenir. Kullanıcının geçmiş komutları görmesine ve tekrar kullanmasına imkân verir.

1.2.3 Command Window Penceresi:

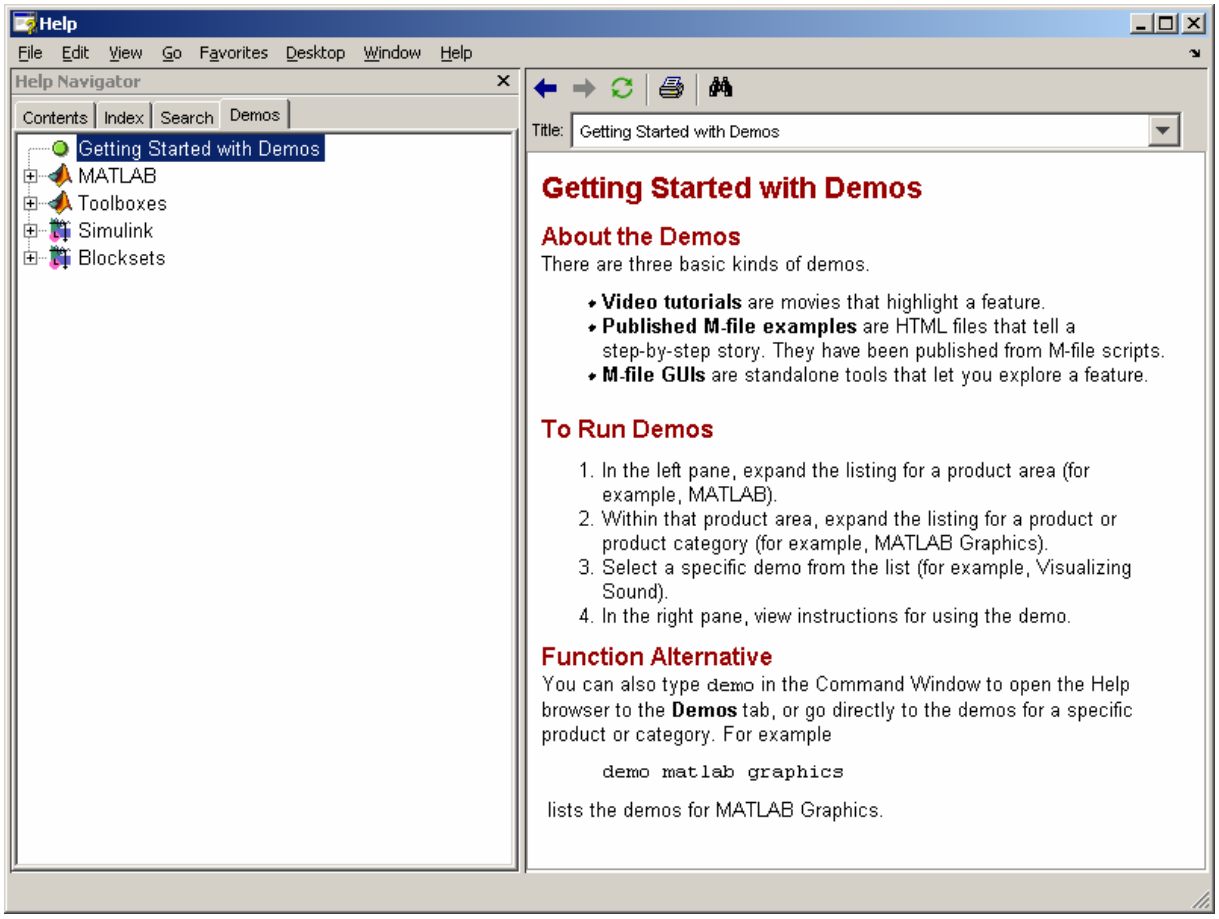
Bu pencere yardımıyla kullanıcı Matlab komutları girer. Girilen her komutun çıktısı da yine bu pencerede ve komutun girilmesinin hemen ardından görüntülenir.

1.2.4 Current Directory Penceresi:

Matlab'in herhangi bir anda aktif olarak kullandığı geçerli dizin yolunu değiştirmek, içinde bulunulan klasört içerisinde taşıma, kopyalama ve dosya silme gibi işlemleri gerçekleştirmek ve ya dosyalar hakkında bilgi edinmek için bu pencere kullanılır. Matlab daima geçerli bir yol üzerinden çalışır. Varsayılan olarak bu yol Matlab'in kurulu olduğu dizin içinde yer alan Work klasörüdür. Geçerli klasör içeriisnde yer alan kullanıcının tanımladığı fonksiyon dosyaları da bu alandan çağrılır. Eğer kullanılacak fonksiyonlar farklı ise bu pencere yardımıyla geçerli yol tanımı değiştiirlmelidir. Ayrıca, Matlab'in geçerli klasör yolu ana penceredekiaraç çubuğunda yer alan Current Directoryu bölümünden de görülebilir veya bu yol tanımı değiştirilebilir.

1.2.5 Demos Penceresi:

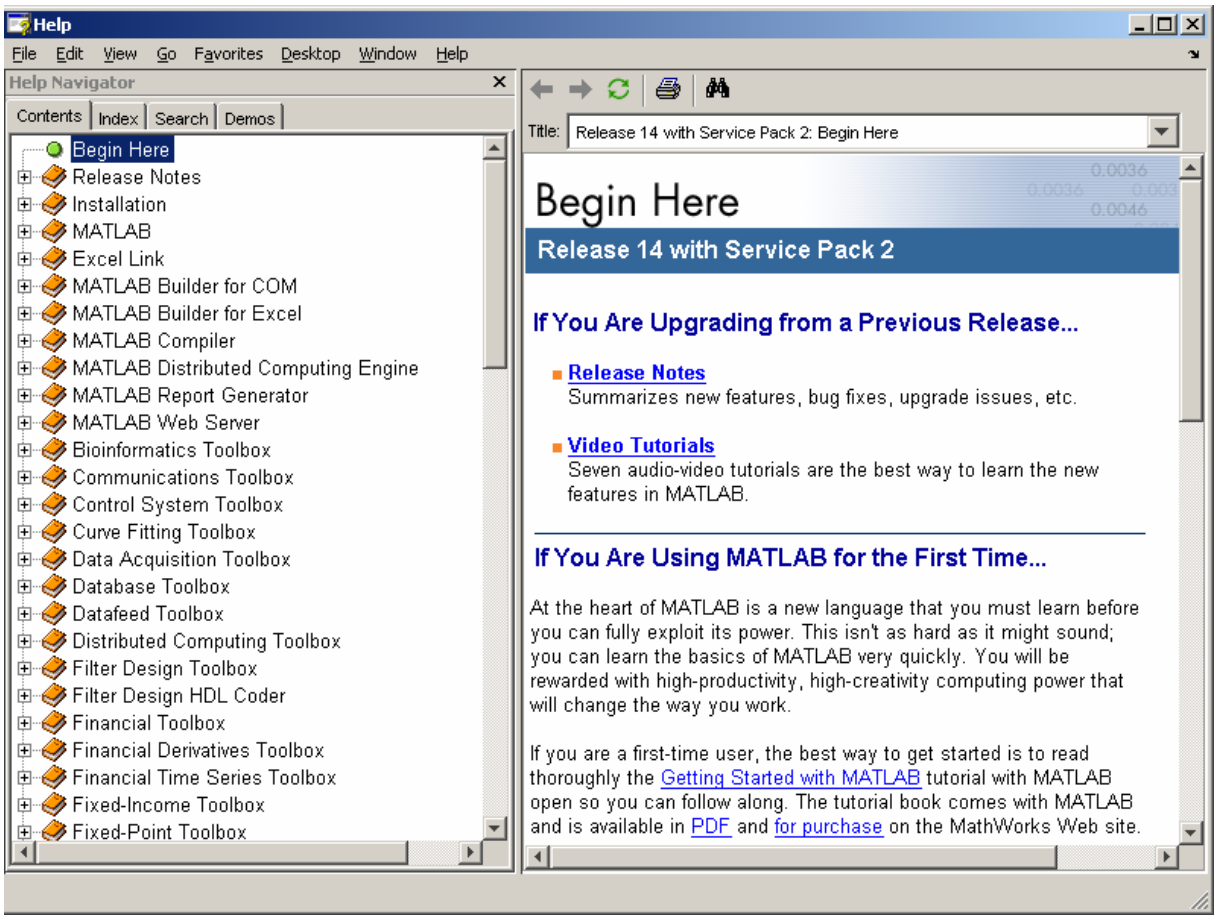
Help menüsünden Demos veya Matlab ana penceresi sol alt köşede yer alan Start butonu kullanılarak Demos komutunun verilmesi ile Karşımıza Şekil 1.3'de görülen demos penceresi gelir. Kullanıcı bu pencere yardımıyla Matlab'in kendi içinde yer alan hazır uygulamaları görebilir, kodlara bakabilir veya konu ile ilgili bilgiler edinebilir.



Şekil 1.3 Demos Penceresi

1.2.6 Help Penceresi:

Bu pencereye ulaşmak için araç çubuğundan soru işareti simgesi tıklanabilir ya da Help menüsü kullanılarak Matlab Help koömutu verilebilir. Kullanıcı Şekil 1.4'deki gibi bir ekran ile karşılaşacaktır.



Şekil 1.4 Matlab Help Penceresi

Matlab yardım penceresi ile kullanıcı sunulan çok geniş ve açıklayıcı anlatımı ile herhangi bir kaynağa gerek kalmadan Matlab kullanımı, Matlab komutları ve pek çok farklı konularda bilgiler edinebilir. Ayrıca, bu pencerenin sol tarafında yer alan Index tabı ile tüm konu başlıklarını sırasıyla görebilir ve herhangi bir konu hakkında bilgi edinebilir. Yine benzer şekilde Search tabını kullanarak hakkında bilgi edinmek istediği bir konuyu Matlab'in hkendi yardım dosyaları içerisinde aratabilir.

1.2.7 Workspace Penceresi:

Kullanıcının işlettiği bir komuta ait değişikenler veya çıktı parametreleri daima Workspace alanına atılır. Böylece kullanıcı herhangi bir anda bu pencere yardımıyla mevcut değişkenlerin listesini görebilir. Ayrıca, kullanıcı bu pencereyi kullarak içeriğini görmek istediği her hangi bir değişkenin üzerinde çift tıklayarak Array Editor pencereni açabilir.

1.3 Aritmetik İşlemler ve Operatörler

Matlab içerisinde kullanılan komutlar ile gerçekleştirilebilecek mantıksal işlemler ve her işleme ait operatör Tablo 1.1'de gösterilmiştir.

Tablo 1.1 Matlab için Aritmetik İşlemler ve Operatörler

İşlem	Sembol	Örnek	Sonuç
Toplama, a+b	+	2+3	5

Çıkarma, a-b	-	5-2	3
Çarpma, a*b	*	3*4	12
Çarpma (matris elemanlarını birebir), a*b	.*	[1 2] .* [3 4]	[3 8]
Bölme (sağdan), a/b	/	14/7	2
Bölme (sağdan) (matris elemanlarını birebir), a/b	./	[1 2] ./ [3 4]	[0.33 0.50]
Bölme (soldan), a\b	\	14\7	0.5
Bölme (soldan) (matris elemanlarını birebir), a\b	.\	[1 2] .\ [3 4]	[3 2]
Üs alma, a^b	^	2^3	8
Üs alma (matris elemanlarını birebir), a^b	.^	[1 2] .^ [3 4]	[1 16]

1.4 Karşılaştırma İşlemleri ve Operatörleri

Matlab içerisinde kullanılan komutlar ile gerçekleştirilebilecek karşılaştırma işlemleri ve her işleme ait operatör Tablo 1.2'de gösterilmiştir.

Tablo 1.2 Matlab için Karşılaştırma İşlemleri ve Operatörleri

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	$3 \neq 8$

Özellikle mantıksal operatörlerden eşittir durumuna dikkat edilmelidir. Karşılaştırma durumlarındaki eşitlik anlamında "= =" operatörü kullnılırken, bir değişkene değer atamak için "=" operatörü kullanılır.

1.5 Mantıksal İşlemler ve Operatörler

Matlab içerisinde kullanılan komutlar ile gerçekleştirilebilecek mantıksal işlemler ve her işleme ait operatör Tablo 1.3.1'de gösterilmiştir.

Tablo 1.3.1 Matlab için Mantıksal İşlemler ve Operatörler

İşlemci	**Anlamı**	**Örnek**
&	AND (VE)	A & B
&&	AND (VE, Kısa Devre)	A && B
|	OR (VEYA)	A | B

||	OR (VEYA, Kısa devre)	A || B
~	NOT (DEĞİL)	~ A

Matlab içerisinde bu koşulun mantıksal doğru (true) olması için bir değişkenin içeriğinin 0'dan farklı olması yeterlidir. Yani bu yönüyle Matlab, C diline çok benzemektedir. Benzer şekilde bir değişkenin içeriği 0 (sıfır) ise bu takdirde bu değişken mantıksal olarak yanlış (false) anlamına gelecektir. Mantıksal operaörlerden kısa devre özelliğine sahip iki operatör C dilinde kullanılan operatörlere çok benzemektedir. Bu operaörler eğer ki karşılaştırma sırasında hem değer ataması, hem de sonucun karşılaştırmaya etkisinin olduğu durumlarda sonuca etkisi açıkça görülebilir. Bu operatörlerin değer ataması olmayan işlemler için kullanımları kısa devre özellikli olmayan (normal) operatörler gibi çalışır. Matlab'te karşılaştıma operatörlerinin önceliği soldan sağa doğrudur.

1.6 Açıklama Operatörü

Matlab ile hazırlanacak fonksiyon dosyalarında herhangi bir satırda açıklama yapmak için açıklama yapılacak cümlenin başına "%" işareti koyulmalıdır. Örnek bir kullanımı aşağıda gösterilmiştir.

```
% Bu satır bir açıklama satırıdır.
```

1.7 Komutların Ekran Çıktısını Gizleme Operatörü

Yazılan bir komutun çıktısını Command Window'da görmek istemiyorsak o komutun sonuna " ; " işareti koyulmalıdır. Özellikle bu operatör fonksiyon dosyalarında arka planda yapılan işlmelerin Command Window'da görüntülenmesinin istenmediği durumlarda kullanılır. Örnek bir kullanım aşağıda sunulmuştur.

```
a=-3; b=17; c=a-b;
```

Bu kullanım sonucu ekranda herhangi bir çıktı görülmez. Ancak, workspace alanına bakılırsa girilen ve hesaplanan değişkenlerin varlığı anlaşılabilir.

1.8 Değişkenler

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 şunlardır.

- 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.

- Değişkenlerde 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 "_" izleyebilir. Noktalama işaretleri değişken ismi olarak kullanılamaz. Çünkü, bunların pek çoğunun MATLAB için özel bir anlamı vardır.

- Workspace'de yer alan değişkenlerin listesini Command Window'da görüntülemek için "who" veya "whos" komutlarından biri kullanılabilir.

1.9 Değişkenlere Değer Atama İşlemi ve Atama Operatörü

Herhangi bir değişkene değer atamak için "=" atama operatörü kullanılır. Örnek olarak aşağıdaki kullanımlara bakılabilir.

```
A=5
B=4
C=A*B
```

Yukarıdaki örnekte A ve B değişkenlere sabitdeğerler atanmakta ve yine C değişkenine atama operaötürü kullanılarak A ve B değikenlerine ait değerlerin çarpılarak sonucun C değişkenine atılması gerçekleştirilmektedir.

1.10 Rakamlar

Matlab'te rakamlar yazılırken ve kullanılırken şu hususlara dikkat edilmelidir.

- MATLAB rakamlar için önünde artı veya eksi işareti ve tercihli ondalık noktası ile
- birlikte alışılagelmiş ondalık (decimal) işaretler sistemi kullanır.

- Ondalık ayıracı için daima nokta "." karakteri kullanılmalıdır. (Türkçe'de ve Windows bölgesel ayarlarında ondalık ayıracı olarak virgül "," ve basamak gruplandırma ayıracı olarak nokta "." Sembollerinin kullanıldığına dikkat edilmelidir.)

- Bilimsel işaretler sistemi 10 tabanına göre kuvvet belirlemek için e harfi kullanır.

- 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.

- Sanal rakamlar "son takı" olarak i veya j harfi kullanır.

Kurala uygun olarak yazılan rakamlar ile ilgili bazı örnekler aşağıda sunulmuştur:

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

1.11 Hazır Matematiksel Fonksiyonlar

Matlab içerisinde kullanılan hazır matematiksel fonksyionlar Tablo 1.3.2'te gösterilmiştir.

Tablo 1.3.2 Matlab için Hazır Matematiksel 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)

1.12 Hazır Rakamlar

Matlab içerisinde kullanılan hazır matematiksel fonksyionlar Tablo 1.4'te gösterilmiştir.

Tablo 1.4 Matlab için Hazır Rakamlar

Rakam	Sembol	Açıklama
Pozitif Sonsuz	inf	Pozitif sonsuz
Negatif Sonsuz	-inf	Negatif sonsuz
Pozitif Epsilon	eps	Pozitif sıfıra çok yakın sayı (epsilon)
Pozitif Epsilon	-eps	Negatif sıfıra çok yakın sayı (epsilon)
Tanımlanmamış (Belirsizlik)	NaN	Not A Number (belirsizlik durumu, örneğin 0/0, inf/inf gibi)
Euler Sabiti (e Sayısı)	exp(1)	Euler sabiti
Pi Sayısı	pi	Pi sabiti

1.13 Vektörler (Matrisler)

Matlab içerisinde bir değişkene vektör ya da matris ataması yapılırken şunlara dikkat edilmelidir:

- Matrisleri belirtmek üzere rakamlar köşeli parantezler içinde yazılmalıdır. Ancak, otomatik artışlar belirtilmişse köşeli parantezleri yazmaya gerek yoktur.

- Matris rakamları için sütünlar arasında "," işareti kullanılmalı veya boşluk "space" konulmalıdır. Matrisin bir satırına geçildiğinde ";" işareti kullanılmalıdır.

- Bir matrisin içeriğini görüntülemek için matrisin değişken isminin yazılması yeterlidir.

- Bir matrisin transpozesini almak için matris değişken isminin en sağına " ' " operaratörü koyulmalıdır.

- Matrislerle ilgili aritmetiksel işlemler yapılırken dikkatli olunmalıdır. Eğer * kullnılacaksa matrislerde çarpma kuralı gereği ilk matrisin sütün sayısı son matrisin satır sayısına eşit olmalıdır. Eğer toplama ve çıkarma işlemleri yapılacaksa her iki matrisin boyutu da aynı olmalıdır.

- Matris elemalarının birebir çarpmak veya bölmek için ".*" ve "./" operaörleri kullanılmalıdır. Bu konuda daha detaylı bilgi için 1.3 aritmetik İşlemler ve operatörler konusuna bakılabilir.

- Matrisler için otomatik aralık tanımlama ve artış için ":" operatörü kullanılmalıdır. Kullanımı "ilk_deger:artis_miktari:son_deger" şeklindedir. Eğer artış miktari parametresi belirtilmeden iki parametreli biçimde yazılırsa otomatik artış miktarı olarak Matlab tarafından 1 değeri atanır.

- Linear artışlar için "linspace" ve logaritmaik artışlar için "logspace" komutları kullanılarak matris dizileri tanımlanabilir.

- Birim matris oluşturmak için "eye" komutu, birler matrisi oluşturmak için "ones" komutu, sıfırlar matrisi oluşturmak için "zeros" komutu, rastgele pozitif elemanlardan oluşan matris oluşturmak için "rand" komutu, rastgele hem pozitif hem negatif elemanlardan oluşan bir matris oluşturmak için "randn" komutu kullanılmalıdır.

- Bir matrisin determinatını bulmak için "det" komutu, tersini bulmak için "inv" komutu, özdeğerlerini bulmak için "eig" komutu, normunu bulmak için "norm" komutu, rankını bulmak için "rank" komutu kullanılmalıdır.

Aşağıda matrislerle ilgili örnekler yapılmıştır.

```
A=[1 2 3 4 5]        % satır vektörü (matrisi)
B=[1;2;3;4;5]        % sütun vektörü (matrisi)
C=[8,9,0;6,7,-1]     % 2 satır ve 3 sütundan oluşan bir matris

D=A*B                % çarpma işlemi için her iki matrisin uygun boyutlardadır.
E=B'                 % matrisin transpozesi alınmakta
F=D+E
G=2.*C               % C matrisinin tüm elemanları 2 sayısı ile çarpılmakta
```

```
H=0:0.1:1000      % 0.1 artımla 0 ile 1000 arası sayılardan oluşan bir dizi tanımlanmakta
K=25:75           % 25 ile 75 arasında varsayılan artımla 1'er artan bir dizi belirlenmekte
L=eye(3)          % 3 x 3 boyutlarında bir birim matris tanımlanmakta
M=rand(2,4)       % 2 satır ve 4 sütun boyutlarında rastgele pozitif değerlerle matris tan.
N=rank(M)         % M matrisinin rankı hesaplanmaktadır.
```

1.14 Polinomlar

Matlab içerisinde polinomlarla işlemler yapılırken şu hususlara dikkat edilmelidir:

- Matlab'te herhangi bir polinom ile ilgili işlem yapılacaksa polinom katsayıları sütun vektörü şeklinde düşünülür.
- Polinom çarpmaları için "conv" ve polinom bölmeleri için "deconv" komutları kullanılmalıdır.
- Bir polinomun köklerini bulmak için "roots" komutu kullanılmalıdır.
- Kökleri bilinen bir polinomu üretmek için "poly" komutu kullanılmalıdır.
- Bir polinom içinde x değişkeninin herhangi bir değeri için polinomun sonucu hesaplanmak isteniyorsa "polyval" komutu kullanılmalıdır.

Aşağıda polinomlarla ilgili örnekler yapılmıştır.

```
a=[1 4 7];b=[ 8 -5 3];          % a ve b polinomlarının tanımlanması

c=conv(a,b)                     % polinom bölme işleminin yapılamsı (a / b)
[katsayilar,kalanlar]=deconv(c,b)  % polinom bölme işleminin yapılamsı (c / b)
                                % böylelikle katsayilar değişkeni içinde tekrar a
                                % polinomu elde edilmekte
d=roots(b)                      % b polinomunun kökleri hesaplanmakta

e=poly(d)                       % d değişkenindeki köklerden polinom
                                % bulunmakta (b polinomu tekrar elde edilmekte)
polyval (a,5)                   % a polinomundaki bilinmeyen değişkenin 5
                                % değeri için polinom sonucu bulunmakta
```

1.15 Grafik Çizim Komutları

MATLAB'da grafik çiziminde kullanılan komutların kısa açıklamaları Tablo 1.5'te gösterilmiştir.

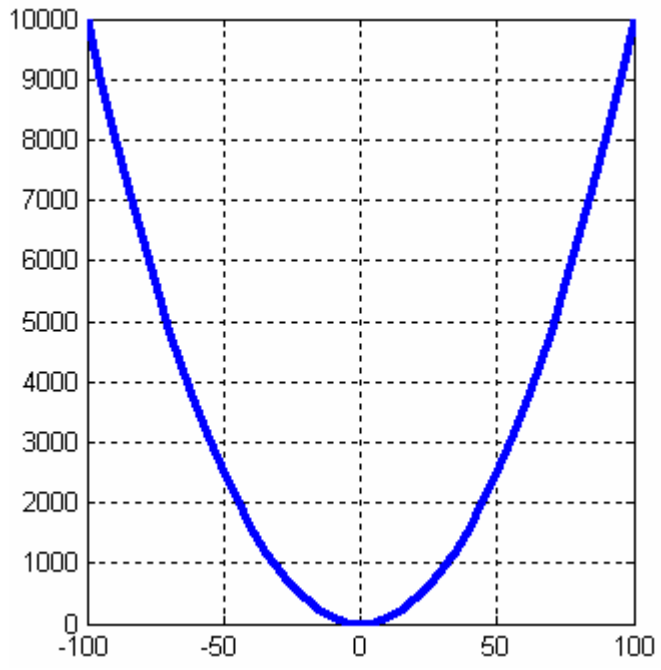
Tablo 1.5 Matlab'te Grafik Çizim Komutları

Fonksiyon	**Açıklaması**	**Örnek**
plot	iki boyutlu çizim için temel komut	plot(xdata,ydata)
polar	Kutupsal (polar) koordinatlardaki	polar(teta,ro)

	çizimler	
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

```
A = -100:5:100; b=a.^2;
plot(a,b)
grid
```

Örnek olarak komut penceresine yukarıdaki komutlar yazıldığında Şekil 1.4’teki grafik elde edilir.



Şekil 1.4 plot (a,b) Komutu ile Çizilen Grafik Örneği

1.16 Zamana Bağlı Tekrarlı İşlem Yaptırma Komutları

Matlab içinde bir komutu belirli zaman aralıklarıyla otomatik olarak yapmak istenilebilir. Bu işlem için Matlab kullancılara "timer" nesnesi sunmaktadır. Bu nesne oluşturularak ve özelliklerini değiştirmek suretiyle bir komut otomatik olarak icra edilebilir.

Matlab içinde bir timer nesnesi oluşturmak için "timer" komutu kullanılır. Ancak, timer nesnesi ile basit bir Matlab komutu ile bir fonksiyonun icra edilmesi farklılık arz etmektedir. Bu nedenle konu iki ayrı alt başlık altında aşağıda ayrıntılı olarak ele alınmıştır.

1.16.1 Timer Nesnesi ile Herhangi Bir Matlab Komutunun Çalıştırılması

Bu işlem için örnek olarak ekrana 5 saniye aralıklarla "Timer tetiklendi." Mesajını yazalım.

Aşağıda verilen komut satırları ile konu açıklanmaya çalışılsın.

```
T = timer( ' TimerFcn ', ' disp( " Timer tetiklendi. " ) ' , ' StartDelay ' , 5 , ' Period ' , 3 );
set ( T , 'ExecutionMode' , 'fixedrate' );
```

Burada"StartDelay" özelliği 5 değeri atanılarak timerın çalışmaya başlatıldıktan 5 saniye sonra teiklenmeye başlaması sağlanmıştır. Ayrıca, bu timer 3 saniye aralıklarla tetiklenecektir. 'ExecutionMode' özelliği 'fixedrate' değerine atanarak sürekli olarak timerın çalışması sağlanmıştır. Artık, programcı timerı başlatmak için "start" ve durdurmak için "stop" komutlarını kullanabilir. Ayrıca, oluşturulan bir timer nesnesi mutlaka işi bittinde veya kullanılmayacaksa silinmlei ve programdan kaldırılmalıdır. Bunun için de "delete" komutu kullanılmalıdır. Bu komutların örnek kullanımları aşağıda verilmiştir.

```
start ( T ) ;
stop ( T ) ;
delete ( T ) ;
```

1.16.2 Timer Nesnesi ile Herhangi M Fonksiyonunun Çalıştırılması

Bu işlem için örnek olarak ekrana 3 saniye aralıklarla "WebCam_Timer_Fcn" isimli bir Matlab M fonksiyonunun çalıştırılması gerektiği düşünülsün.

Aşağıda verilen komut satırları ile konu açıklanmaya çalışılsın.

```
T = timer ('Period', 3, 'StartDelay', 0);          % "StartDelay" özelliği sıfır yapılarak timerın
T.TimerFcn = {@WebCam_TimerFcn, handles} ;  % başlangıçta hiç gecikme olmadan
Set ( T , 'ExecutionMode' , 'fixedrate' );          % çalışması sağlanmaktadır.
```

Burada öncelikle "timer" komutu kullanılarak "timer" kullanılmak üzere T değişkenine timer sınıfından bir class atanmaktatır. Daha sonra ise bu sınıf bir yapı değişkeni gibi işlem göreceği için bu class ile ilgili alt özelliklere "." operatörü ile erişilebilir. Dolayısıyla oluşturulan timer sınıfını tutan T değişkeninin "TimerFcn" callback fonksiyonuna parametresine "WebCam_Timer_Fcn" M fonksiyonu ismi atnmaktadır. Burada atamanın hücre yapısı şeklinde olduğu için süslü parantezler kullanılarak yapıldığına dikkat edilmelidir.

Burada çok önemli bir durum şudur ki icra edilecek fonksiyon eğer lokal ise ya da timer komutu bir M fonksiyon dosyası içinde çalıştırılmak isteniyor ve bu fonksiyon dosyası içinde yer alan fonksiyonlardan bviri kullanılmak isteniyorsa bju udurmda komut şu şekilde kullanılmalıdır:

T.TimerFcn = {@WebCam_Timer_Fcn} ;

Burada "@" operatrü lokalde işlem yapılacağını gösterir. Daha sonra set komutu kullanılarak oluştutulan timerın sürekli çalışması ayarlanmaktadır. Kullanıcı bu adımdan sonra "start" komutu ile oluşltutulan timer nesnesini başlatabilir ve "stop" komutunu kullanarak durdurabilir. Ayrıca, oluşturulan bir timer nesnesi mutlaka işi bittinde veya kullanılmayacaksa silinmlei ve programdan kaldırılmalıdır. Bunun için de "delete" kopmutu kullanılmalıdır. "start", "stop" ve "delete" komutları için örnek kullnım şekli aşağıda verilmiştir.

```
start ( T ) ;
stop ( T ) ;
delete ( T ) ;
```

1.16.3 Timer Nesnesinin Özelliklerinin Okunması

Bir timer nesnesinin özelliklerini okumak için "get" komutu kullanılmalıdır. Örnek kullanım için aşağıdaki komutlara bakılabilir.

```
zamanlayici = timer( ' TimerFcn ', ' disp( " Timer tetiklendi. " ) ' );

get ( zamanlayici);
```

Bu komut ile oluşturulan ve "zamanlayici" değişkenine atılan timer nesnesine ait tüm özellikler Matlab komut ekranında listelenecektir.

1.16.4 Timer Nesnesinin Özelliklerinin Set Edilmesi

Bir timer nesnesinin hjerhangi bir özelliğini değiştirmek için "set" komutu kullanılır. Örmek kullanım şekli aşağıda verilmiştir.

```
zamanlayici = timer( ' TimerFcn ', ' disp( " Timer tetiklendi. " ) ' );

set ( zamanlayici , ' ExecutionMode ' , ' fixedRate ', ' BusyMode ' , ' drop ' , ' Period ' ,1);
```

"set" komutu ile bir timer nesnesi için atanabilecek tüm özellikler listesi de aşağıda sunulmuştur.

- BusyMode: [{drop} | queue | error]
- ErrorFcn: string -or- function handle -or- cell array
- ExecutionMode: [{singleShot} | fixedSpacing | fixedDelay | fixedRate]
- Name
- ObjectVisibility: [{on} | off]
- Period
- StartDelay

- StartFcn: string -or- function handle -or- cell array
- StopFcn: string -or- function handle -or- cell array
- Tag
- TasksToExecute
- TimerFcn: string -or- function handle -or- cell array
- UserData

1.16.5 Timer Nesnesinin Belirli Bir Zamanda Çalışmasının Sağlanması

Timer nesnesini belirli bir zamanda çalıştırmak için "startat" komutuna ihtiyaç vardır. Örnek kullanım şekli için aşağıdaki komutlara bakılabilir.

```
zamanlayici = timer( ' TimerFcn ', ' disp( " Timer tetiklendi. " ) ' );

startat ( zamanlayici , now+1/24 );
```

Örnek kullanım şekli ile "now" komutu ile alınan anlık zaman bilgine göre o andaki zamandan bir saat sonrasında timer devreye girecektir.

1.16.6 Timer Nesnesinin Fonksiyonları

Timer nesnesinin üç farklı çalıştırılacak fonksiyonu ya da komutu tutatn callback türü vardır. Bunlar aşağıda listelenmiştir.

- TimerFcn callback fonksiyonu her adımda çalıştırılacak fonksiyon ya da komutlar için
- StartFcn callback fonksiyonu timer nesnesi ilk çalışmaya başladığında çalıştırılacak fonksiyon ya da komutlar için
- StopFcn callback fonksiyonu timer nesnesi sonlandırıldığında çalıştırılacak fonksiyon ya da komutlar içindir.

1.16.7 Timer ile Çizim Yapılması için Yapılması Gerekenler

Eğer timer nesnesi ile bir plot benzeri çizim komut kullanılacaksa Timer nesnesi içinde "drawnow" fonksiyonu kullnılmalıdır. Bu şekilde grafik çizim alanı sürekli olarak güncel tutulacaktır.

1.16.8 Timer Nesnesinin Başlatılması ve Durdurulması

Oluştulan bir timer nesnesini başlatmak için "start" ve durudurmak için "stop" komutları kullanılır. Aşağıda bu komutların nasıl kullnılacağı gösterilmiştir.

```
zamanlayici = timer( ' TimerFcn ', ' disp( " Timer tetiklendi. " ) ' );
start ( zamanlayici ) ;
stop ( zamanlayici ) ;
```

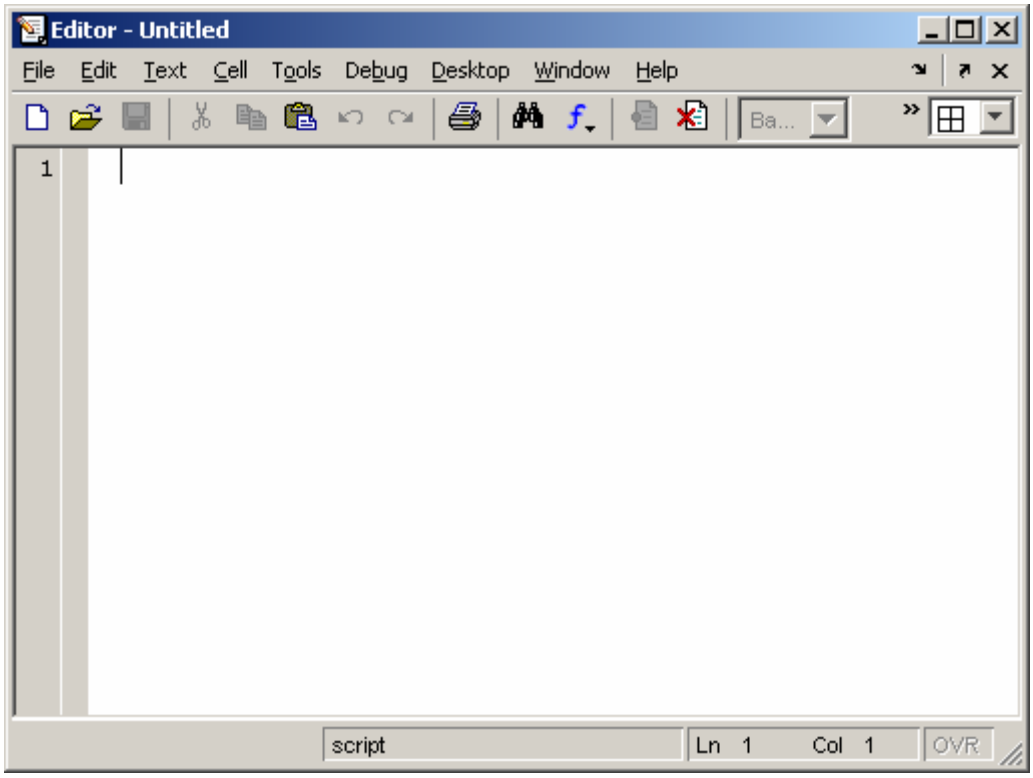
1.16.9 Timer Nesnesinin Yok Edilmesi

Oluştulan bir timer nesnesi mutlaka eğer artık çalıştırılmayacaksa bellekten kaldırılmalıdır. Bu işlem için “delete” komutu kullanılmalıdır. Örnek kullanım şekli aşağıda gösterilmiştir.

```
zamanlayici = timer( ' TimerFcn ', ' disp( " Timer tetiklendi. " ) ' );
start ( zamanlayici ) ;
stop ( zamanlayici ) ;
delete ( zamanlayici ) ;
```

1.17 Fonksiyon Dosyaları (M Dosyaları, M Files)

Kullanıcılar Matlab içinde kendilerine ait fonksiyonlar yazabilir ve kullanabilirler. Bir fonksiyon yazmak için Windows'un Notepad programı kullanılabileceği gibi Matlab'in kendisine ait "Editor" uygulaması da kullanılabilir. Bu uygulamayı açamak için komut satırından "edit" komutu verilir. Kullanıcı Şekil 1.5'teki gibi bir ekran ile karşılaşacaktır.



Şekil 1.5 Matlab Editor Uygulaması Ekran Görüntüsü

Örneğin "kare_al" isminde bir fonksiyon yazılmış olsun ve bu fonksiyon kendisine parametere olarak gönderilen sayıların karesini hesaplasın. Bunun için Editor uygulamasında aşağıdaki komutlar yazılır.

```
function sonuc=kare_al(sayi)
   sonuc=sayi.^2;
end
```

Daha sonra bu dosya Matlab'in kurulu olduğu dizin altında yer alan work klasörüne kaydedilir. Dosyanın kaydeilme esnasında isminin fonksiyon ismi ile aynı olmasına dikkat edilmelidir. Yani kare_al ismi ile kaydedilir. Böylelikle work dizini altında kare_al.m isminde bir m dosyası oluşacaktır. Daha sonra komut satırından aşağıdaki komutları girdiğimizde hazırladığımız fonksiyon kullanılarak girilen sayının karesi hesaplanacaktır.

```
x=5;
kare_al(x)
```

veya

```
kare_al(5)
```

BÖLÜM-II

MATLAB'TE KONTROL SİSTEMLERİNİN KOMUT KULLANIMI İLE ANALİZİ

Matlab kullanılarak kontrol sistemlerinin çeşitli modelleri bulunabilir. Ayrıca, varolan sistemler birbirleri ile seri veya paralel bağlanabilir veya geri beslemeli hale getirilebilir. Matlab ile bulunabilecek model türleri şunlardır:

- Transfer Fonksiyonu Modeli
- Sıfır-Kutup-Kazanç Modeli
- Durum Denklemi Modeli
- Tanımlayıcı Durum Denklemi Modeli
- Frekans Cevabı Verileri Modeli
- Ayrık Zaman Modeli

2.1 Modellerin Oluşturulması

Her bir kontrol modelinin oluşturulması için gerekli komutlar Tablo 2.1'de verilmiştir.

Tablo 2.1 Matlab'te Kontrol Modellerinin Oluşturulması

Model İsmi	**İlgili Komut**	**Giriş Parametreleri**	**Örnek Kullanım**

Transfer Fonksiyonu Modeli	Tf	pay matrisi
		payda matrisi
			pay=[1];
			payda=[1 6 5 0];
			sys=tf(pay,payda)

Sıfır-Kutup-Kazanç Modeli	Zpk	sıfırlar matrisi kutuplar matrisi kazanç katsayısı	sys = ss([-2 -1 1],[1 3 -5],1)
Durum Denklemi Modeli	Ss	a matrisi b matrisi c matrisi d matrisi	sys = ss([-2 -1;1 -2],[1 1;2 -1],[1 0],[0 1])
Tanımlayıcı Durum Denklemi Modeli	dss	a matrisi b matrisi c matrisi d matrisi e matrisi	sys = dss[0 1;-5 -2], [0;3], [0 1],0, [1 2;3 4])

Frekans Cevabı Verileri Modeli	frd	frekans matrisi
		cevap matrisi
			freq=[1000;2000;3000];
			resp=[-0.81126-0.0003i;-0.1751-0.0016i;-0.0926-0.4630i];
			H=frd(resp,freq,'Units','Hz')

Ayrık Zaman Modeli	tf, zpk, ss, dss, frd (örnekleme zamanı eklenmeli, son parametre)	seçilen modele göre giriş parametreleri	pay=[1];payda=[1 6 5 0]; sampling_time=0.1;%saniye sys=tf(pay,payda,**sampling_time**)

2.2 Model Verilerinin Elde Edilmesi

Her bir kontrol modeline ait verilerin elde edilmesi için gerekli komutlar Tablo 2.2'de verilmiştir.

Tablo 2.2 Matlab'te Model Verilerinin Elde Edilmesi

Model İsmi	**İlgili Komut**	**Giriş Parametreleri**	**Çıkış Parametreleri**	**Örnek Kullanım**
Transfer Fonksiyonu Modeli	tfdata	tf sistem	pay matrisi payda matrisi	[num,den] = tfdata(sys,'v')
Sıfır-Kutup-Kazanç Modeli	zpkdata	zpk sistem	sıfırlar matrisi kuutplar matrisi kazanç katsayısı	[z,p,k] = zpkdata(sys,'v')
Durum Denklemi Modeli	ssdata	ss sistem	a matrisi b matrisi c matrisi d matrisi	[a,b,c,d] = ssdata(sys,'v')
Tanımlayıcı Durum Denklemi Modeli	dssdata	dss sistem	a matrisi b matrisi c matrisi d matrisi e matrisi	[a,b,c,d,e]= dssdata(sys,'v')
Frekans Cevabı Verileri Modeli	frdata	fr sistem	frekans matrisi cevap matrisi	[response,frequency]= frdata(sysfr,'v')
Ayrık Zaman Modeli	tfdata, zpkdata, ssdata, dssdata, frddata	seçilen modele göre sistem	seçilen modele göre çıkış parametreleri ve örnekleme zamanı	[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')

2.3 Modellerin Birbirine Dönüştürülmesi

Kontrol modellerinin birbirlerine dönüştürülmesi için gerekli komutlar Tablo 2.3'te verilmiştir.

Tablo 2.3 Matlab'te Modellerin Birbirine Dönüştürülmesi

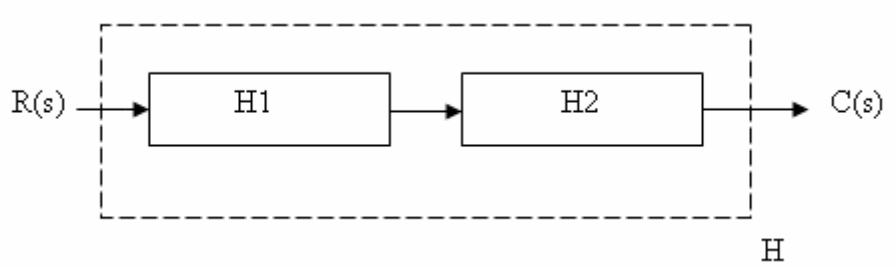
Giriş Modeli	**Çıkış Modeli**	**İlgili Komut**	**Örnek Kullanım**
Transfer Fonksiyonu	Sıfır-Kutup-Kazanç Modeli	tf2zp	[z,p,k] = tf2zp(a,b)

Modeli			
Transfer Fonksiyonu Modeli	Durum Denklemi Modeli	tf2ss	[A,B,C,D] = tf2ss(a,b)
Sıfır-Kutup-Kazanç Modeli	Transfer Fonksiyonu Modeli	zp2tf	[a,b] = zp2tf(z,p,k)
Sıfır-Kutup-Kazanç Modeli	Durum Denklemi Modeli	zp2ss	[A,B,C,D] = zp2ss(z,p,k)
Durum Denklemi Modeli	Transfer Fonksiyonu Modeli	ss2tf	ss2tf(A,B,C,D,iu) iu : sistemin giriş sayısı
Durum Denklemi Modeli	Sıfır-Kutup-Kazanç Modeli	ss2zp	[z,p,k] = ss2zp(A,B,C,D,i) iu : sistemin giriş sayısı

2.4 Modellerin Birbirine Bağlanması

2.4.1 Seri Bağlantı

Örnek olarak Şekil 2.1'de görülen sistemi ele alalım.



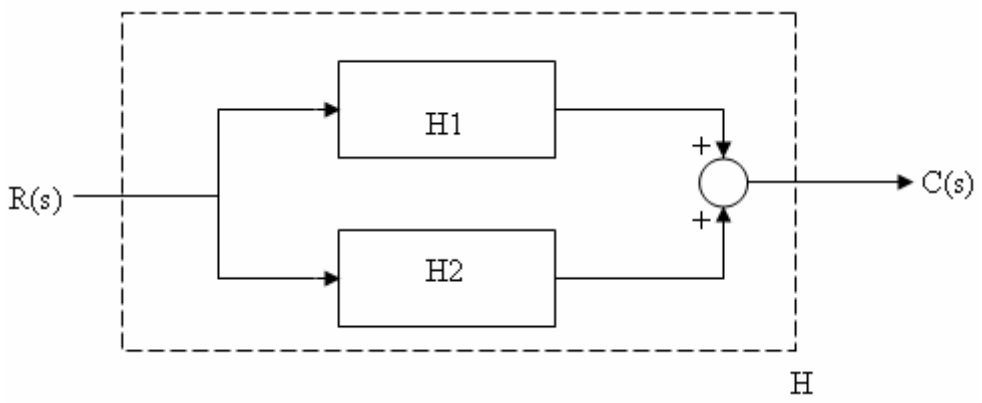
Şekil 2.1 Seri Bağlı Modellerden Tek Bir Modelin Elde Edilmesi

Bu sistemi seri bağlantılı hale getirmek için yazılması gereken komut satırları aşağıdaki gibi olacaktır:

```
H1=tf(pay1,payda1);
H2=tf(pay2,payda2);
H=series(H1,H2)       % veya çarpma işlemi ile çözüm;     H=H1*H2;
```

2.4.2 Paralel Bağlantı

Örnek olarak Şekil 2.2'de görülen sistemi ele alalım.



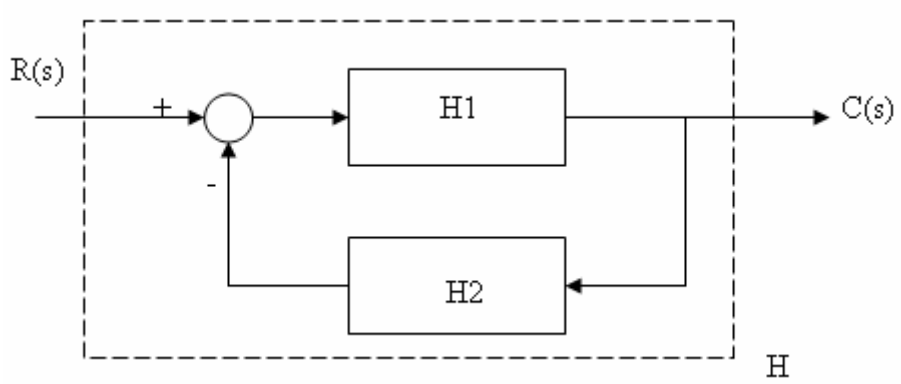
Şekil 2.2 Paralel Bağlı Modellerden Tek Bir Modelin Elde Edilmesi

Bu sistemi pararlel bağlantılı hale getirmek için yazılması gereken komut satırları aşağıdaki gibi olacaktır:

```
H1=tf(pay1,payda1);
H2=tf(pay2,payda2);
H=parallel(H1,H2)     % veya toplama işlemi ile çözüm;     H=H1+H2;
```

2.4.3 Geri Beslemeli Bağlantı

Örnek olarak Şekil 2.3'te görülen sistemi ele alalım.



Şekil 2.3 Geri Beslemeli Modelin Tek Bir Modele İndirgenmesi

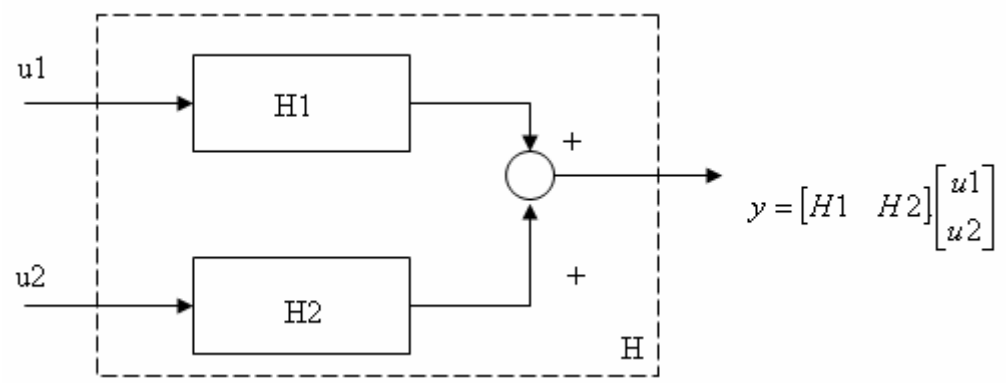
Bu sistemin geri beslemeli toplam sistem modelini bulmak için yazılması gereken komut satırları aşağıdaki gibi olacaktır:

```
H1=tf(pay1,payda1);
H2=tf(pay2,payda2);
H=feedback(H1,H2,-1)
```

Burada negatif geri beslemeli bir sistem için çözüm istenirse feedback komutunun 3. parametresi -1 olur. Ancak, bu parametre yazılmazsa da feedback komutu için varsayılan değer -1'dir, yani negatif geri besleme seçeneğidir. İndirgenen sistem pozitif geri beslemeli olursa bu parametre 1 değerini alır.

2.4.4 Çıkışların Toplanması

Örnek olarak Şekil 2.4'te görülen sistemi ele alalım.



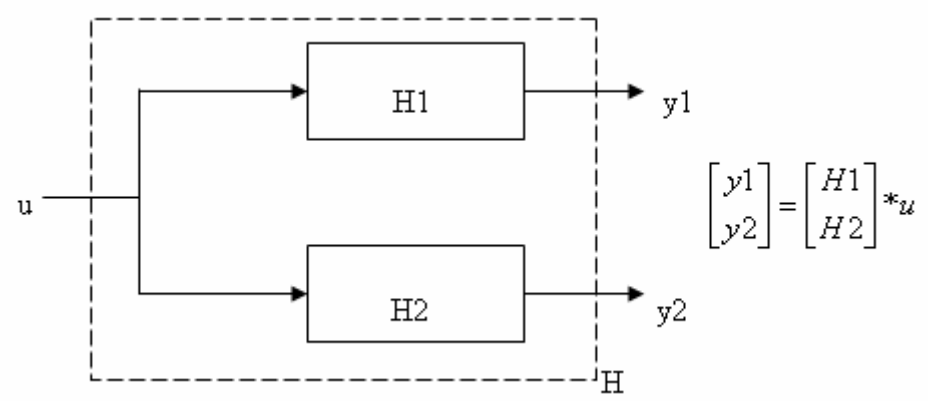
Şekil 2.4 İki Model Çıkışlarının Toplanması ile Oluşan Çıkışın Elde Edilmesi

İki modele ait çıkışların toplanması ile oluşacak toplam çıkışın yazılması için gerekli komut satırları aşağıdaki gibi olacaktır:

```
H1=tf(pay1,payda1);
H2=tf(pay2,payda2);
H=[H1,H2]
```

2.4.5 Girişlerin Dağıtılması

Örnek olarak Şekil 2.5'te görülen sistemi ele alalım.



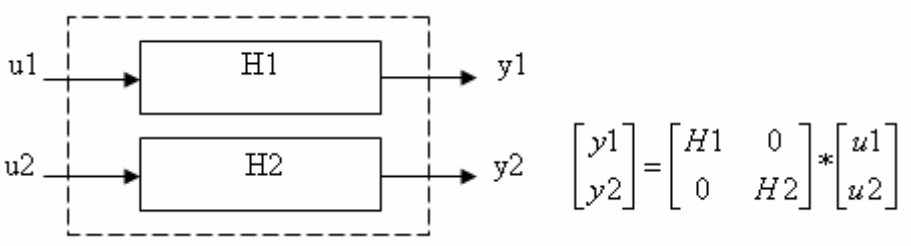
Şekil 2.5 İki Modele Ait Çıkışların Dağıtılması

İki modele ait çıkışların dağıtılması sonucu oluşacak çıkışların yazılması için gerekli komut satırları aşağıdaki gibi olacaktır:

```
H1=tf(pay1,payda1);
H2=tf(pay2,payda2);
H=[H1;H2]
```

2.4.6 Girişlerin ve Çıkışların Birleştirilmesi

Örnek olarak Şekil 2.6'da görülen sistemi ele alalım.



Şekil 2.6 İki Modele Ait Girişlerin ve Çıkışların Birleştirilmesi

İki modele ait girişlerin ve çıkışların birleştirilmesi sonucu elde edilecek çıkışların yazılması için gerekli komut satırları aşağıdaki gibi olacaktır:

```
H1=tf(pay1,payda1);
H2=tf(pay2,payda2);
H= append(H1,H2)
```

2.5 Modellerin Cevaplarının Elde Edilmesi

Kontrol sistemi modellerinin cevapları Matlab'in kullanıcıya sunduğu hazır fonksiyonlarla elde edilebilir ve grafiği çizilebilir.

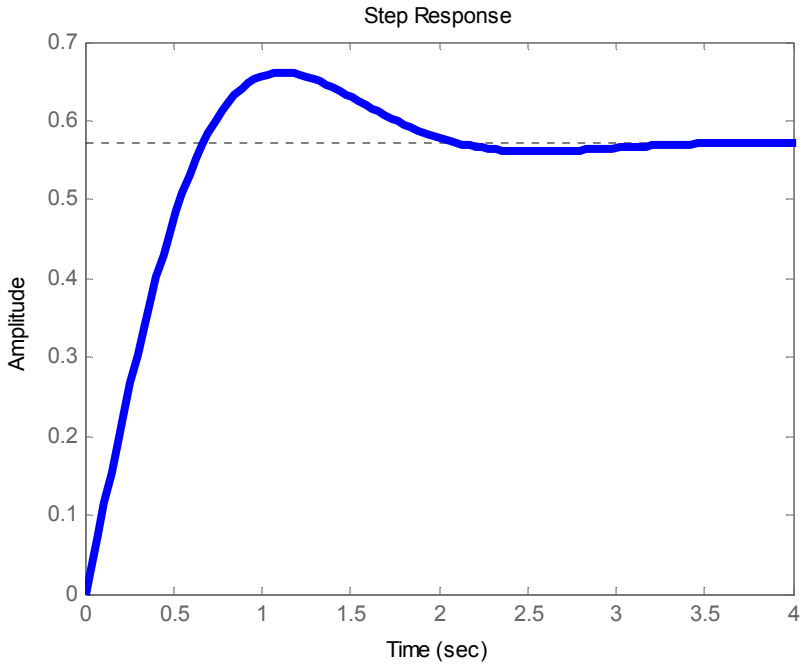
2.5.1 Adım Cevabı (Step Response)

Bir kontrol sisteminin adım cevabının gözlenmesi ve grafiğinin çizilebilmesi amacıyla "step" komutu kullanılır. Bu komuta gönderilen parametre bilgisi oluşturulan bir modelin sistem değişkenidir. (Model oluşturma ile ilgili ayrıntılı bilgi için 2.2 konu başlığına bakınız.)

Örnek olarak $\frac{C(s)}{R(s)} = \frac{s+4}{s^2+3s+7}$ sisteminin adım cevabı grafiği bulunsun. Bu işlem için yazılacak komut satırları aşağıda gösterilmiştir:

```
pay = [1 4];
payda = [1 3 7];
sistem = tf(pay,payda);
step(sistem)
```

Örnek oalınan sistemin adım cevabı grafiği Şekil 2.7'de gösterilmiştir.



Şekil 2.7 Örnek Verilen Bir Sistemin Adım Cevabı Grafiği

Ayrıca, bir sistemin adım cevabı grafiği istenilen zaman aralığında çizilebilir. Bunun için "step" komutuna verilecek 2. parametreye bir zaman aralığı matrisi değişkeninin atanması yeterlidir. Bu durum için örnek bir kullanım aşağıda gerçekleştirilmiştir.

```
pay = [1 4];
payda = [1 3 7];
sistem = tf(pay,payda);
zaman_araligi = 0:0.1:25;      % 0.1 artımla 0-25 sn arasında zaman aralığının tanımlanması
step(sistem,zaman_araligi)
```

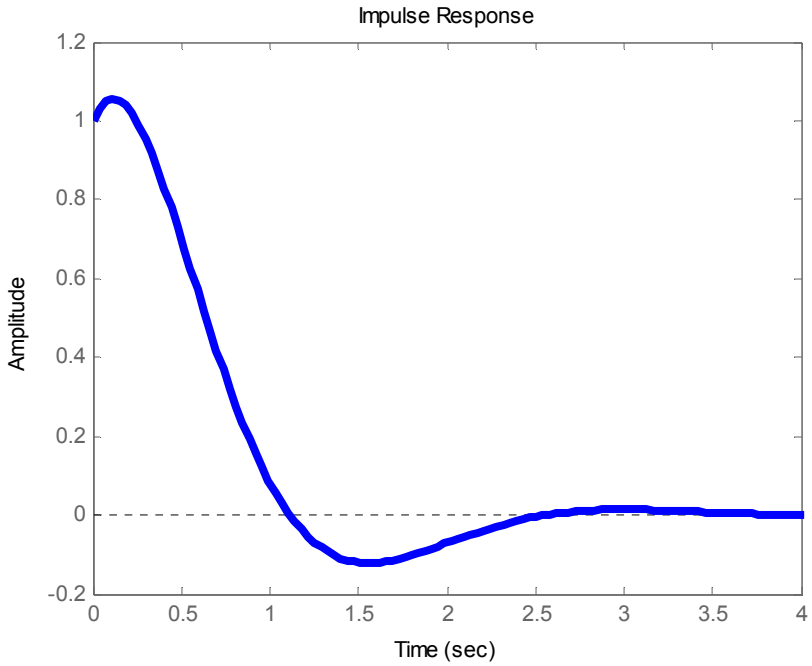
2.5.2 Ani Darbe Cevabı (Impulse Response)

Bir kontrol sisteminin ani darbe cevabının gözlenmesi ve grafiğinin çizilebilmesi amacıyla "impulse" komutu kullanılır. Bu komuta gönderilen parametre bilgisi oluşturulan bir modelin sistem değişkenidir. (Model oluşturma ile ilgili ayrıntılı bilgi için 2.2 konu başlığına bakınız.)

Örnek olarak $\frac{C(s)}{R(s)} = \frac{s+4}{s^2+3s+7}$ sisteminin ani darbe cevabı grafiği bulunsun. Bu işlem için yazılacak komut satırları aşağıda gösterilmiştir:

```
pay = [1 4];
payda = [1 3 7];
sistem = tf(pay,payda);
impulse(sistem)
```

Örnek olarak verilen sistemin ani darbe cevabı grafiği Şekil 2.8'de gösterilmiştir.



Şekil 2.8 Örnek Verilen Bir Sistemin Ani Darbe Cevabı Grafiği

Ayrıca, bir sistemin adım cevabı grafiği istenilen zaman aralığında çizilebilir. Bunun için "impulse" komutuna verilecek 2. parametreye bir zaman aralığı matrisi değişkeninin atanması yeterlidir. Bu durum için örnek bir kullanım aşağıda gerçekleştirilmiştir.

```
pay = [1 4];
payda = [1 3 7];
sistem = tf(pay,payda);
zaman_araligi = 0:0.1:25;      % 0.1 artımla 0-25 sn arasında zaman aralığının tanımlanması
impulse(sistem,zaman_araligi)
```

2.5.3 Rampa Cevabı (Ramp Response)

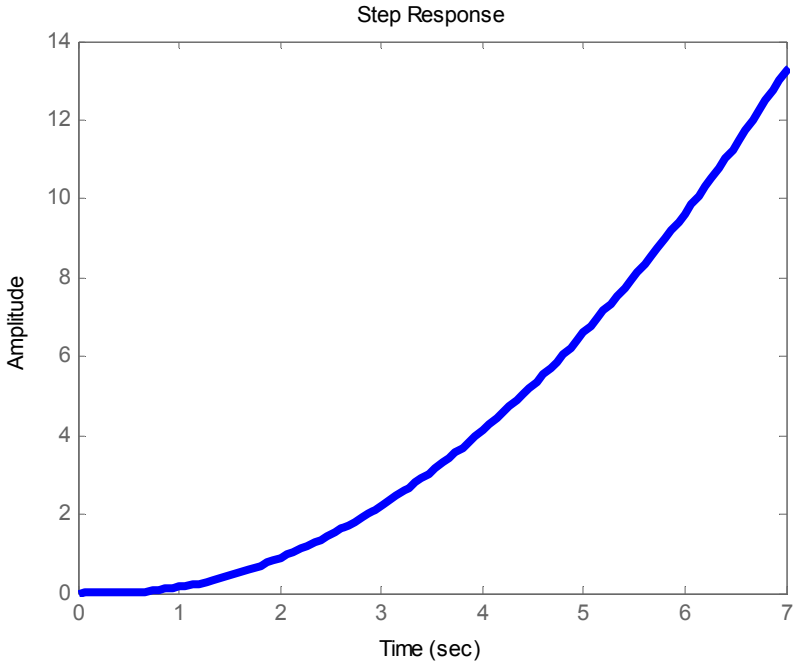
Bir kontrol sisteminin rampa cevabının gözlenmesi ve grafiğinin çizilebilmesi amacıyla Matlab'in sahip olduğu hazır bir komut yoktur. Ancak, sisteme uygulanan rampa girişine karşılık elde edilen çıkışın adım cevabının çizdirilmesi suretiyle bir sistemin rampa cevabı grafiği çizdirilebilir.

Örnek olarak $\frac{C(s)}{R(s)} = \frac{s+4}{s^2+3s+7}$ sisteminin rampa cevabı grafiği bulunsun. Bu işlem için yazılacak komut satırları aşağıda gösterilmiştir:

```
pay = [1 4];
payda = [1 3 7];
sistem = tf(pay,payda);
giris_pay = [1];
giris_payda = [1 0 0];
giris_sistem = tf(giris_pay,giris_payda);
toplam_sistem_cikisi=series(giris_sistem,sistem);
```

step(toplam_sistem_cikisi)

Örnek olarak verilen sistemin rampa cevabı grafiği Şekil 2.9'da gösterilmiştir.



Şekil 2.9 Örnek Verilen Bir Sistemin Rampa Cevabı Grafiği

Bu cevabı istenilen zaman aralığında çizmek için 2.5.1 konu başlığına bakınız.

2.6 Kontrol ile İlgili Diğer Komutlar

2.6.1 rss Komutu :

Bu komut kullanılarak giriş ve çıkış sayısı verilmek suretiyle rasgele durum uzayı modelleri oluşturulabilir. Komutun kullanımı aşağıda gösterilmiştir.

sistem = rss (mertebe)
sistem = rss (mertebe , cikis_sayisi)
sistem = rss (mertebe , cikis_sayisi , giris_sayisi)

Bu ifadelerde,

- "mertebe" değişkeni rasgele üretilecek modelin mertebesini,
- "cikis_sayisi" rasgele üretilecek modelde bulunması istenilen çıkış sayısı değerini,
- "giris_sayisi" rasgele üretilecek modelde bulunması istenilen giriş sayısı değerini

ifade eder. Eğer, ilk kullanım seçilirse tek giriş ve tek çıkışlı bir sistem üretilir. Bir sonraki kullanım için isetek giriş ve birden fazla çıkışa sabip rasglee durum uzayı modeli bulunur. Son kullanım içinse hem giriş, hem olması istenilen çıkış sayısı değerlerinde rasgele bir durum uzayı modeli oluşturulabilir. Bu komutun işletilmesi sonucu üretilen model sistem

değişkenine atanır. Eğer, bu sistemin durum uzayı matrisleri elde edilmek istenirse "ssdata" komutu kullnılabilir.

2.6.2 ord2 Komutu :

Bu komut kullanılarak Matlab'te doğal frekansı (Wn parametresi) ve sönüm oranı (zeta veya kısi parametresi) bilinen ikinci dereceden bir kontrol sistemine ait durum uzayı modeli veya transfer fonksiyonu bulunabilir. Komutun kullanımı aşağıda gösterilmiştir.

```
[A,B,C,D] = ord2(wn,z)      % durum uzayı modelinin elde edilmesi
[num,den] = ord2(wn,z)      % transfer fonksiyonunun elde edilmesi
```

Bu ifadelerde,

- "wn" değişkeni modeli bulunacak sistemin doğal frekansını
- "z" değişkeni sönüm oranını (ζ , ksi)

ifade eder.

2.6.3 gensig Komutu :

Bu komut kulanılarak tanımlanan bir zaman aralığında istenilen bir fonksiyon çıkışı elde edilebilir, yani bir sinyal jeneratörü işlevi sağlar. İki değişik kullanım şekli şöyledir:

```
[genlik,zaman] = gensig(sinyal_cesidi,sinyal_periyodu)
[genlik,zaman] = gensig(sinyal_cesidi,sinyal_periyodu,sinyal_toplam_suresi,sampling_time)
```

Bu ifadelerde,

- "sinyal_cesidi" degişkeni ile üretilecek sinyalin tipi belirlenir. Örnek olarak
  - sinüs sinyali için 'sin'
  - kare dalga sinyali için 'square'
  - darbe sinyali için 'pulse'

  verilebilir.

- "sinyal_periyodu" degişkeni ile üretilecek sinyalin periyodu belirlenir.
- "sinyal_periyodu" degişkeni ile üretilecek sinyalin periyodu belirlenir.
- "sinyal_toplam_suresi" degişkeni ile üretilecek sinyalin toplam zamanı belirlenir.
- "sampling_time" degiskeni ile sinyal üretiken kullnılacak örnekleme zamanı belirlenir.
- "genlik" degişkeni ile üretilen sinyalin zaman vektörü karşılık genlik değerleri elde edilir.
- "zaman" degişkeni ile üretilen sinyalin zaman vektörü elde edilir.

Genellikle “gensig” komutu “lsim” komutu ile birlikte kullanılır. . (Ayrıntılı bilgi için 2.6.3 konu başlığına bakınız.)

2.6.4 lsim Komutu :

Bir kontrol sistemine daima adım, ani darve v.b. gibi standart işaretlerin dışında da bir giriş uygulanabilir ve cevabı görülmek istenilebilir. Bu komut kullanılarak herhangi bir kontrol sisteminin rasgele belirlenmiş giriş değerlerine karşılık üretilen cevabı için grafik çizilebilir. Bu komutun kullanımı şu şekilde olmaktadır:

lsim(sistem,genlik,zaman)

Bu ifadede,

- “sistem” girişine sinyal uygulanacak sistemin modelidir.

  (Model oluşturma ile ilgili ayrıntılı bilgi için 2.2 konu başlığına bakınız.)
- “genlik” degişkeni sistemin girişine uygulanan sinyalin genlik vektörüdür.
- “zaman” degişkeni sistemin girişine uygulanan sinyalin zaman vektörüdür.

Genellikle “lsim” komutu “gensig” komutu ile birlikte kullanılır. (Ayrıntılı bilgi için 2.6.2 konu başlığına bakınız.)

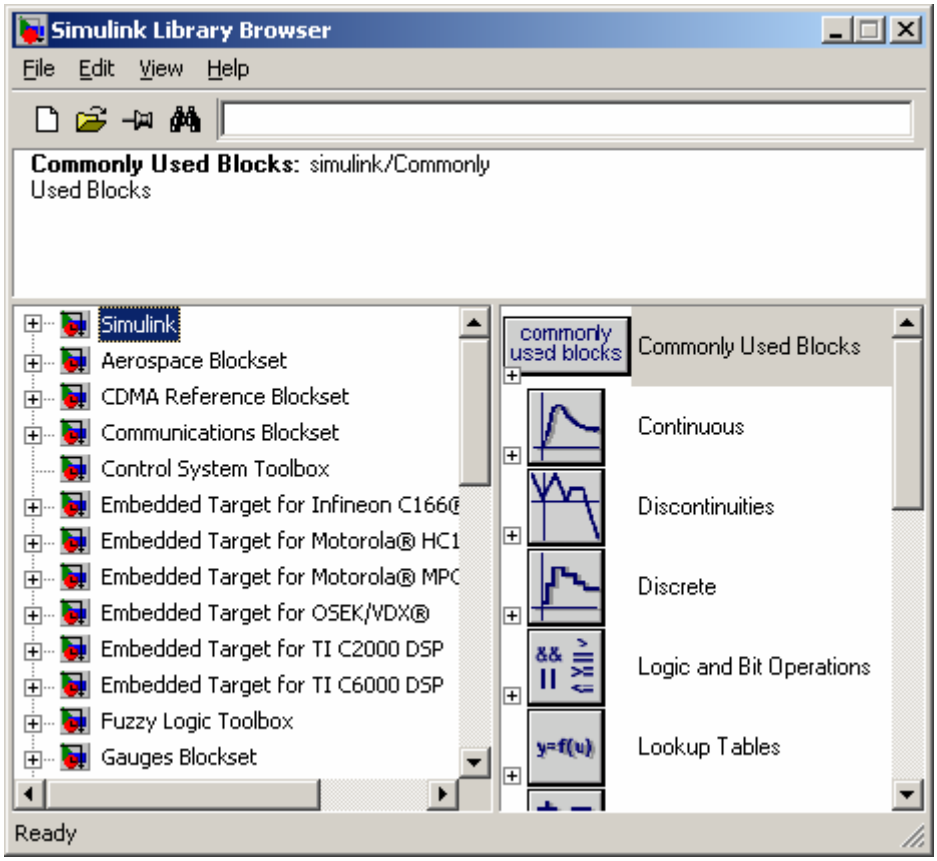
BÖLÜM-III

MATLAB'TE KONTROL SİSTEMLERİNİN SİMULİNK ORTAMINDA ANALİZİ

3.1 Simulink'e Giriş

Matlab ile pek çok işlemi komut kullanarak yapabiliriz. Ancak, bazen bu durum gerçekleştirilen işin amacına bağlı olarak uzun zaman alabilir. Bu nedenle Matlab kullnıcılara Simulink adı verilen bir araç sunmuştur. Bu araç yardımı ile komut ezberlemeksizin ve komut yazmak için harcanan uzun zamanlar yerine çok kısa sürede sadece gerekli blokları Simulink çalışma alanına ekleyerek çok değişik alanlara yönelik işlemleri gerçekleştirebiliriz.

Simulink ekranını açmak için "simulink" komutu kullanılabilir. Kullanıcı bu araç çalıştığında Şekil 3.1'deki ekran görüntüsü ile karşılaşır.



Şekil 3.1 Simulink Başlangıç Ekranı Görüntüsü

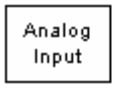
3.2 Kontrol Alanı ile İlgili Bloklar

Simulink değişik bilim dalları ve farklı alanlar ile ilgili çeşitli araç çubukları (toolboxes) içerir. Kontrol alanı ile ilgili farklı işlevlere sahip pek çok blok farklı araç kutuları altında yer almaktadır. Kontrol ile ilgili sıklıkla kullanılan bloklar ayrıntılı olarak aşağıda incelenmiştir.

3.2.1 Add Bloğu :

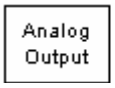
: Girişlerin toplamını veren bir blok. Girişlerin sayısı ve her bir girişe uygulanacak işaret, blok diyalog kutusunda ayarlanabilir.

3.2.2 Analog Input Bloğu :



: Paralel porttan veya bilgisayara bağlı bir DAQ kartının giriş kanalından analog veriyi okuma bloğu.

3.2.3 Analog Output Bloğu :

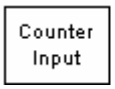


: Paralel porta veya bilgisayara bağlı bir DAQ kartının çıkış kanalına analog veriyi yazma bloğu.

3.2.4 Constant Bloğu :

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

3.2.5 Counter Input Bloğu :



: Paralel porttan veya bilgisayara bağlı bir DAQ kartının giriş kanalından gelen darbe sinyallerini sayma bloğu.

3.2.6 Demux Bloğu :

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

3.2.7 Derivative Bloğu :

: Giriş sinyalinin zamana göre türevini alır.

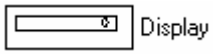
3.2.8 Digital Input Bloğu :

: Paralel porttan veya bilgisayara bağlı bir DAQ kartının giriş kanalından dijital veriyi okuma bloğu.

3.2.9 Digital Output Bloğu :

: Paralel porta veya bilgisayara bağlı bir DAQ kartının çıkış kanalına dijital veriyi yazma bloğu.

3.2.10 Display Bloğu :

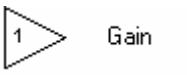


: Giriş sinyalinin o anki değerini gösterir.

3.2.11 Fcn Bloğu :

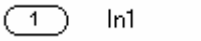
: Bu blok, matematik ifadeler için fonksiyon oluşturmaya yarar, ‘u’ harfi giriş sinyali için kullanılmaktadır. Örnek bir ifade ‘tan(u[l])*exp(u[2] }’ olabilir; burada u[l] and u [2] sırasıyla birinci ve ikinci giriş verileri kümelerini temsil etmektedir.

3.2.12 Gain Bloğu :



: Kazanç sabiti. İstenilen bir değer atanabilir.

3.2.13 In1 Bloğu :

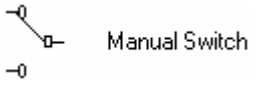


: Bir alt-sistem için giriş portu sağlar

3.2.14 Integrator Bloğu :

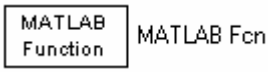
: Giriş sinyalinin zamana göre integralini alır.

3.2.15 Manual Switch Bloğu :



: Çalışma esnasında üzerine fareyle çift tıklanarak konum değiştiren anahtar.

3.2.16 MATLAB Fcn Bloğu :



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

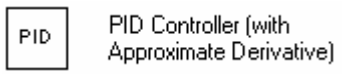
3.2.17 Mux Bloğu :

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

3.2.18 Out1 Bloğu :

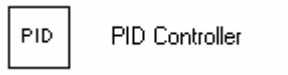
: Bir alt-sistem için çıkış portu sağlar.

3.2.19 PID Controller (with Approximate Derivative) Bloğu :



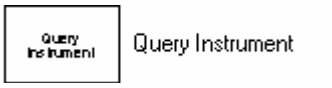
: Hazır PID denetleyici bloğu (Türevin alınmacağı birim aralık belirlemeli).

3.2.20 PID Controller Bloğu :



: Hazır PID denetleyici bloğu.

3.2.21 Query Instrument Bloğu :

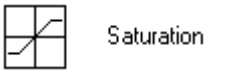


: Seri porttan bilgi okumak amacıyla kullanılan blok.

3.2.22 Relay Bloğu :

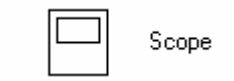
: Girş sinyalinin belli seviyeden büyük olması durumunda belirlenen bir çıkış ve belli bir değerin altında olmasında farklı bir çıkış veren ve bu iki değer arasında çıkışın önceki durumunu koruyan blok (on/off denetleyici tasarlama amacıyla kullanılır).

3.2.23 Saturation Bloğu :



: Sinyalin alt ve üst değerlerini sınırlanmış haliyle çıkışa verir.

3.2.24 Scope Bloğu :



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

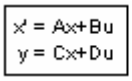
3.2.25 Signal Generator Bloğu :

: Sinyal jeneratörü. Çeşitli dalga şekillerini üreten blok.

3.2.26 Sin Wave Bloğu :

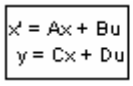
: Sinyal Üretici. Dalga şekillerini üreten blok.

3.2.27 State Space Bloğu :



: Çok girişli ve çok çıkışlı bir sistemin durum uzayı modeli

3.2.28 State-Space (with initial outputs) Bloğu :



: Çok girişli ve çok çıkışlı bir sistemin durum uzayı modeli (başlangıç çıkış değerleri belirlemeli).

3.2.29 Step Bloğu :

: Basamak sinyali üretir

3.2.30 Stop Simulation

: Giriş sinyali sıfırdan farklı olduğunda simülasyonu ‘u durduran blok

3.2.31 Subsystem Bloğu :

: Birden fazla bloğun tek bir blok içinde toplanmasını sağlar

3.2.32 Subtract Bloğu :

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

3.2.33 Sum Bloğu :

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

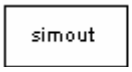
3.2.34 Switch Bloğu :

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

3.2.35 To Instrument Bloğu :

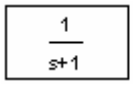
To Instrument : Seri porta bilgi göndermek için kullanılan blok.

3.2.36 To Workspace Bloğu :



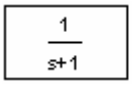
To Workspace : Bir giriş sinyalini, MATLAB çalışma alanında, simülasyon bittikten sonra, erişilebilir bir MATLAB matrisinde depolayan blok.

3.2.37 Transfer Fcn (with initial outputs) Bloğu :



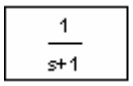
Transfer Fcn (with initial states) : Doğrusal bir sistemin transfer fonksiyonu modeli (başlangıç çıkış değerleri belirlemeli).

3.2.38 Transfer Fcn (with initial states) Bloğu :



Transfer Fcn (with initial outputs) : Doğrusal bir sistemin transfer fonksiyonu modeli (başlangıç giriş değerleri belirlemeli).

3.2.39 Transfer Fcn Bloğu :

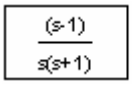


Transfer Fcn : Doğrusal bir sistemin transfer fonksiyonu modeli

3.2.40 XY Graph Bloğu :

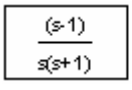

XY Graph : İki skaler girişi kullanarak bir grafik çizdiren blok. Üstteki giriş kapısına bağlanan sinyal bağımsız değişken (x ekseni) ve alttakine bağlanan ise bağımlı değişkendir (y ekseni).

3.2.41 Zero Pole Bloğu :



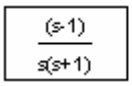
Zero-Pole : Doğrusal bir sistemin sıfır-kutup-kazanç modeli

3.2.42 Zero-Pole (with initial outputs) Bloğu :



Zero-Pole (with initial outputs) : Doğrusal bir sistemin sıfır-kutup-kazanç modeli (başlangıç çıkış değerleri belirlemeli).

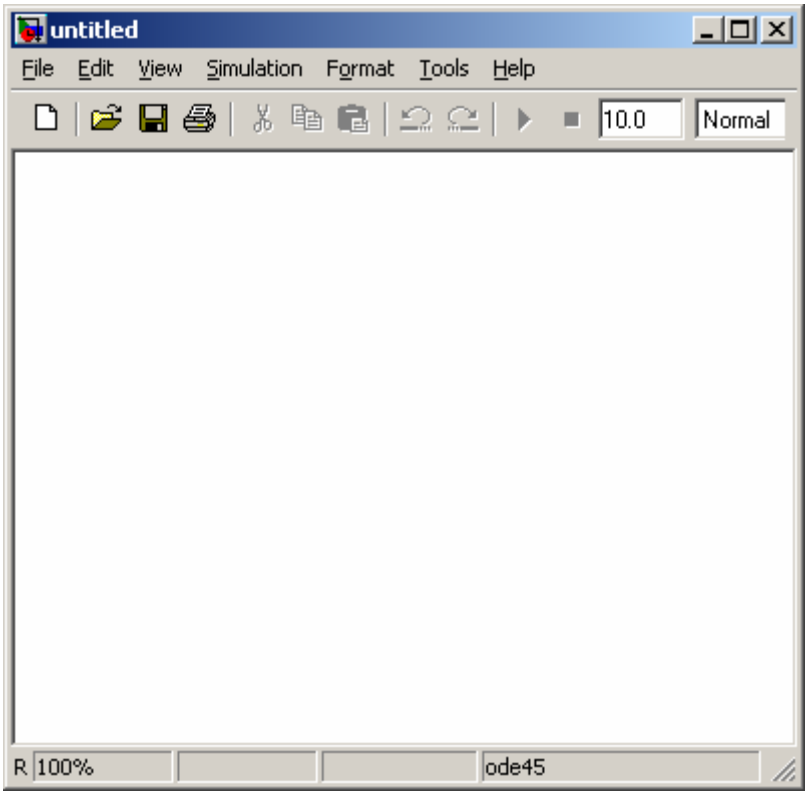
3.2.43 Zero-Pole (with initial states) Bloğu :



: Doğrusal bir sistemin sıfır-kutup-kazanç modeli (başlangıç giriş değerleri belirlemeli).

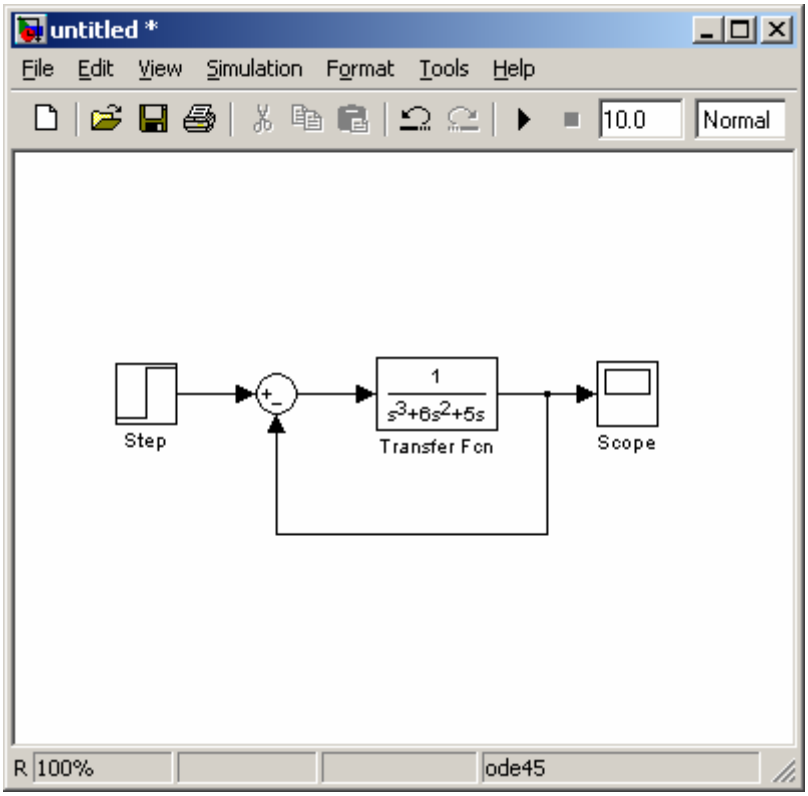
3.3 Simulink Ortamında Model Oluşturma

Simulink aracı ile model oluşturmak için öncelikle "Simulink Library Browser" penceresinden araç çubuğunda yer alan "New" simgesi tıklanır. Şekil 3.2'de görülen boş bir çalışma alanı karşımıza gelir.



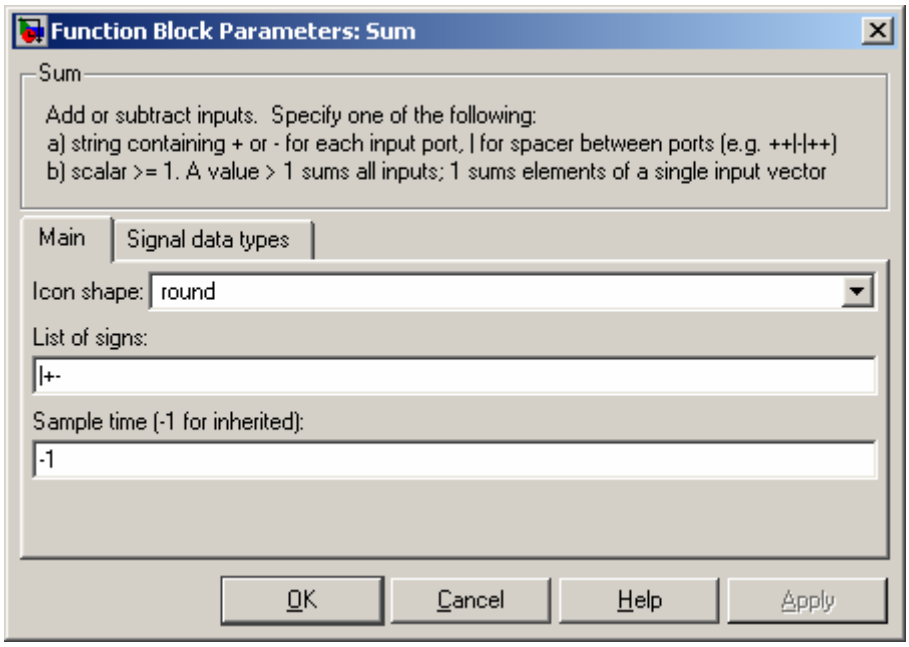
Şekil 3.2 Simulink Ortamında Boş Çalışma Alanı

Örnek olarak bir açık çevrim transfer fonksiyonu $\frac{1}{s^3+6s^2+5s}$ olan bir sistemin adım cevabına Simulink ortamında bakılsın. Bunun için "Simulink Library Browser" penceresinden ilgili ağaç dallarının altında yer alan ve Şekil 3.3'te gösterilen bir blok diyagramı kurulacak şekilde bloklar seçilerek farenin sol tuşu basılı halde Library penceresinden boş çalışma alanına sürüklenir. Çalışma alanı üzerinde istenilen noktaya gelince farenin sol tuşu bırakılır. Çalışma alanına yanlış bir blok konulmuş ise o blok üzerinde farenin sol tuşu ile bir kere tıklanılarak seçilir. Daha sonra klavyeden Del tuşuna basılır. Bloklar arasındaki hatları eklemek için hattı eklenecek bir uçta fare sol tuşu tıklanır ve basılı tutulur. Daha sonra fare işaretçisi hattın ekleneceği uca götürülür ve bırakılır. Bu şekilde hattın eklenmesi tamamlanmış olur. Ayrıca, girişten çıkışa olmak üzere önce giriş blokları fare sol tuşu ile seçilir. Sonra Ctrl klavye tuşu basılı tutulur. Çıkışı hangi bloğa bağlanacak üzere o blok farenin sol tuşu ile tıklanır. Böylelikle otomatik olarak giriş bloklarının çıkış ucu bir sonraki bloğun giriş ucu ile bağlanmış olur.



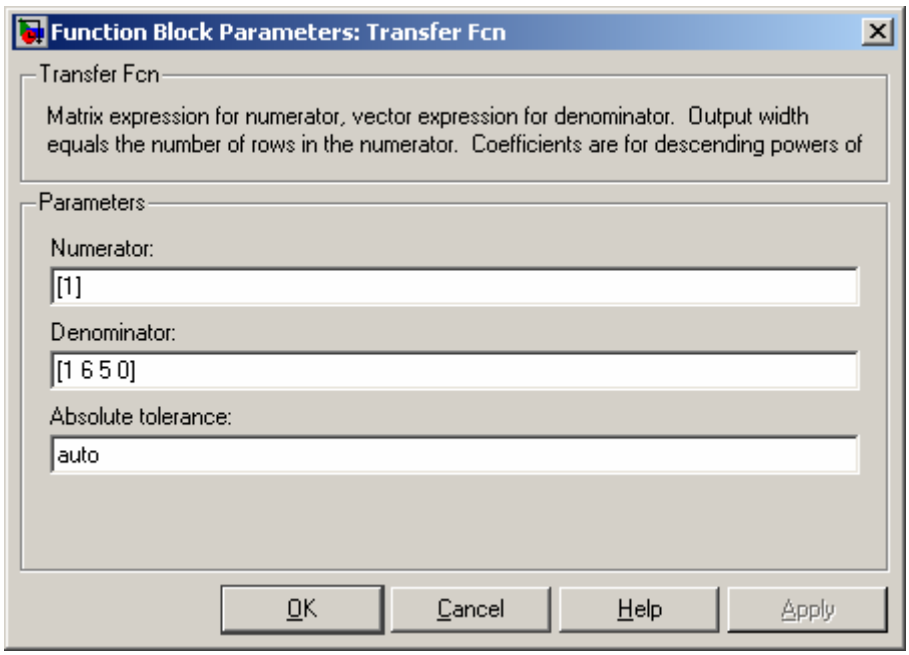
Şekil 3.3 Örnek Bir Kontrol Sisteminin Simulink Ortamında Oluşturulması

Şekil 3.3'teki blok diyagram tasarımı sırasında "Sum" bloğunun üzerinde farenin sol tuşu ile çift tıklanaılarak özellikler pencere açılır. Gelen pencerede "List of Sign" kısmının işaretleri Şekil 3.4'teki gibi değiştirilir.



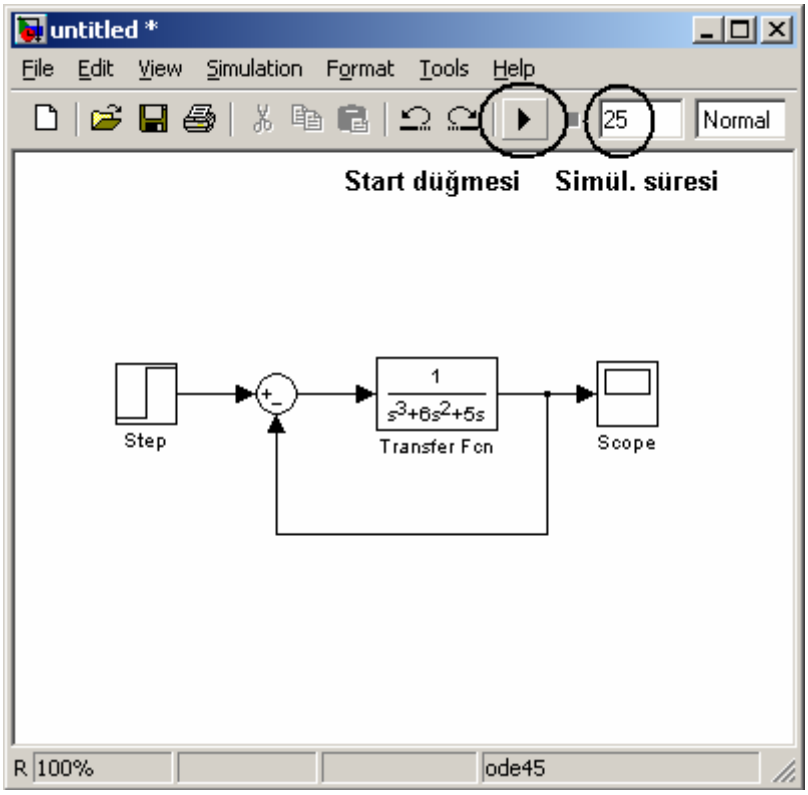
Şekil 3.4 Sum Bloğunun Özellikler Penceresi

Ayrıca, çalışma alanına eklenen “Transfer Fcn” bloğunun da benzer şekilde özellikler penceresi açılır ve buradaki ayarlar Şekil 3.5’teki gibi değiştirilir.



Şekil 3.5 Transfer Fcn Bloğunun Özellikler Penceresi

Yukarıda anlatılan tüm işlemler tamamlandıktan sonra Şekil-3.6’da gösterilen simülasyon bitiş süresi 25 olarak değiştirilir ve yine aynı pencerede gösterilmiş olan “start Simulation” butonuna basılarak oluşturulmuş olan sistem çalıştırılır (Blok diyagrmı hazırlanan sistem hızlı bir şekilde çalışacak, bu esnada önce Start butonunun şekli Stop halini alacak, ancak tekrar eski haline (Start durumuna) dönecektir. Yani simülasyon çok kısa sürede çalışıp duracaktır. Burada 25 sn denilmesine rağmen Simulink kendi içinde kısada olsa bu süreyi sisteme uygular ve sistemin cevabını bulur.).



Şekil 3.6 Simülasyon Süresinin Ayarlanması ve Simülasyonu Çalıştırma Düğmesi

Sistemin cevabına bakmak için çalışma alanındaki "Scope" bloğu üzerinde farenin sol tuşu ile çift tıklanır. Karşımıza Şekil 3.7'deki gibi bir pencere gelecektir. Bu pencerenin üst tarfında yer alan ikonlardan dürbün üzerine tıklanır. Bu şekilde garfik çizimi otomatik olarak ölçeklenmiş ve mevcut ekrana sığar bir hale getirilmiş olunur.

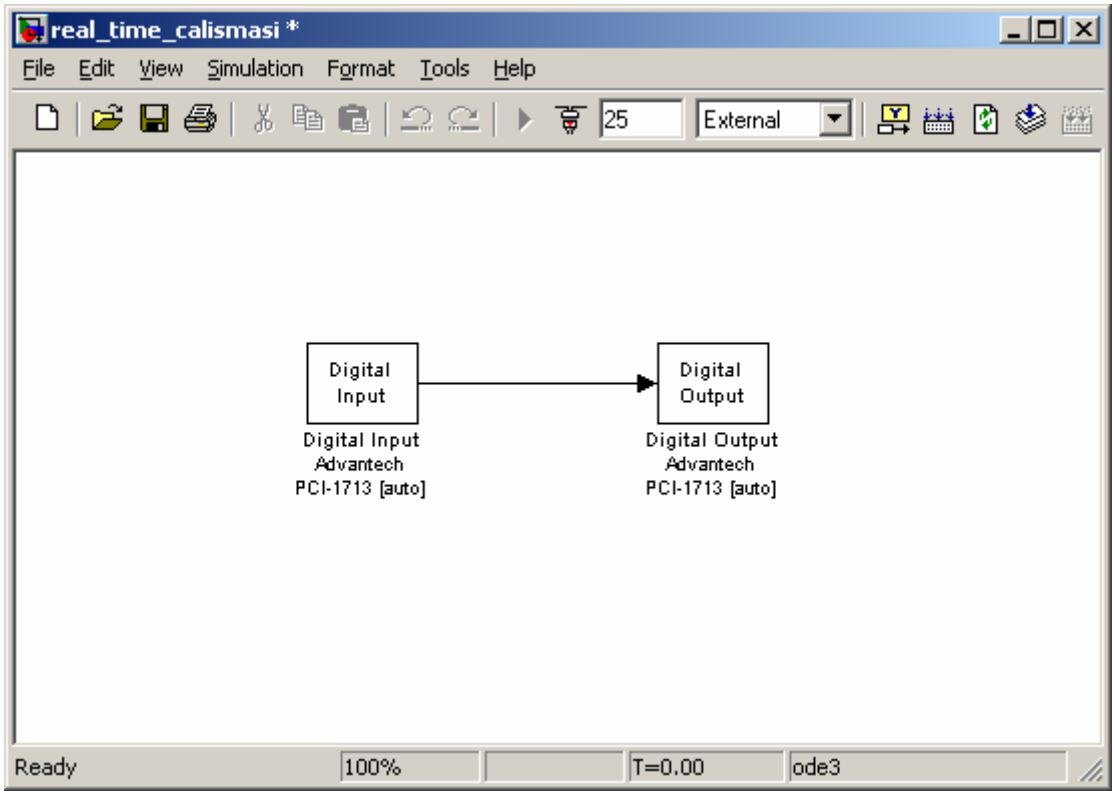
Şekil 3.7 Örnek Verilen Bir Sistemin Simulink Ortamında Adım Cevabı

İstenirse çalışma alanı penceresinde yer alan araç çubuğu kullanılarak Save ikonu tıklanmak suretiyle hazırlanan bu Simulink modeli (blok diyagramı) bilgisayara *.mdl uzantısı ile kaydedilebilir. Özetle örnek olarak verilen sistemin adım cevabı blok diyagramları ve Simulink ortamında bulunmuş olmaktadır.

3.4 Simulink Ortamında Gerçek Zamanlı (Real Time) Çalışma :

Simulink kullanılarak simülasyon amacıyla sistemler analiz edilebileceği gibi gerçek zamanlı (yani bir DAQ kartı ya da seri veya paralel port gibi bir aygıttan verilerin alınması ve aynı anda simulink ortamına alınan verilerin aktarılması ve işlenmesi sağlanmak üzere) olarak analizler yapılabilir ve sistemler kontrol edilebilir.

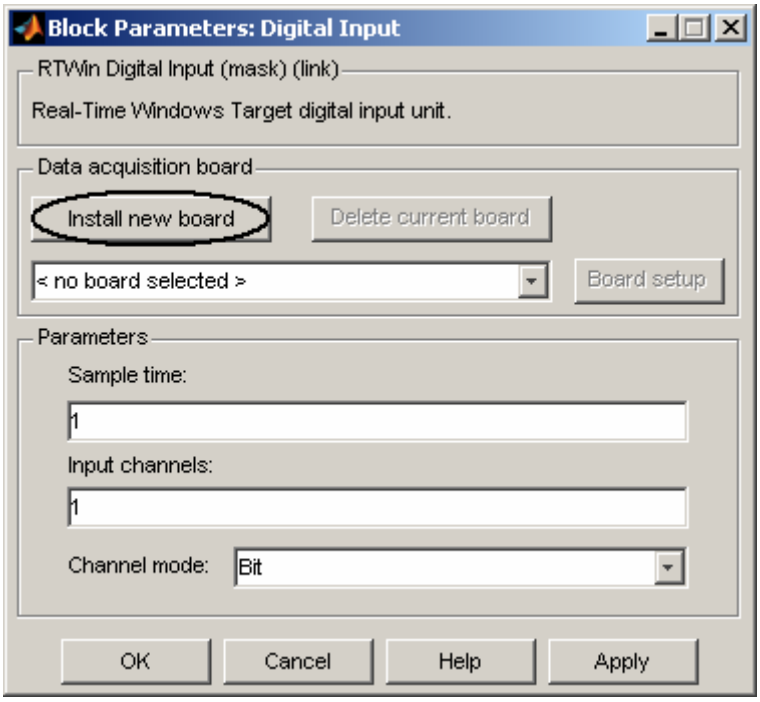
Örnek olarak elimizde Advantech firmasına ait PCI-1713 model DAQ kartı olsun. Bu DAQ kartının 2 nolu dijital giriş kanalından 0.1 sn örnekleme süresi ile alınan verilerin yine aynı kartın 5 numaralı dijital çıkış kanalından 0.1 sn öçnekleme süresiyle çıkmış olsun. Böyle bir tasarım için öncelikle boş bir çalışma alanına "Simulink Library Browser" penceresi kullanılarak "Real-Time Windows Target" ağacı altından çalışma alanına Şekil 3.8 de gösterilen blok diyagram tasarımını gerçekleştirmek üzere gerekli bloklar eklenmiş olsun.



Şekil 3.8 Simulink Ortamında Real Time Çalışma için Örnek Blok Diyagramı Tasarımı

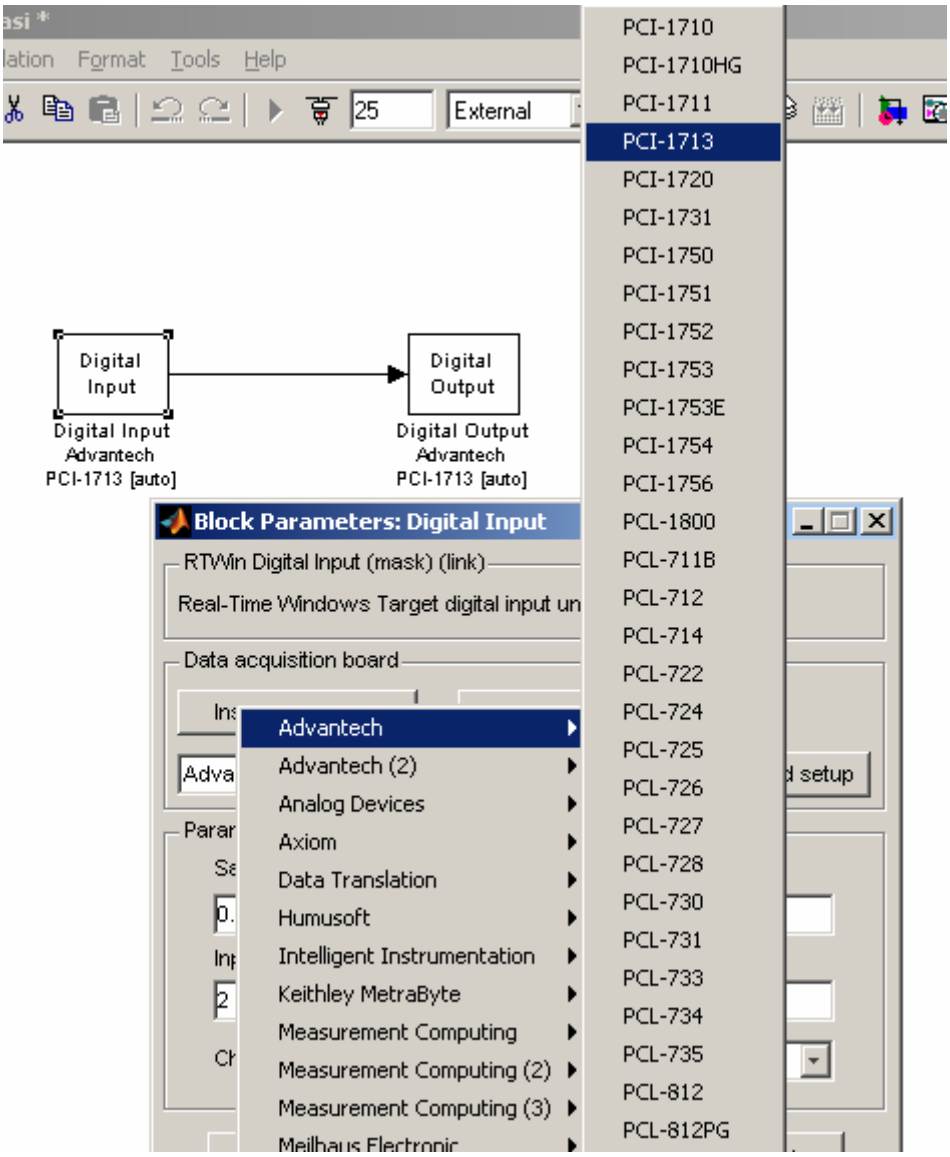
Şekil 3.8'deki gibi bir tasarıma ulaşmak için önce yapılması gereken Digital Input bloğu üzerinde farenin sol tuşu ile çift tıklayarak bu bloğun özellikler penceresini açmak ve bu

pencereden “Install New Board” butonu tıklanmalıdır. Bu duurm Şekil 3.9’da da gösterilmiştir.



Şekil 3.9 Digital Input Bloğunun Özellikler penceresi

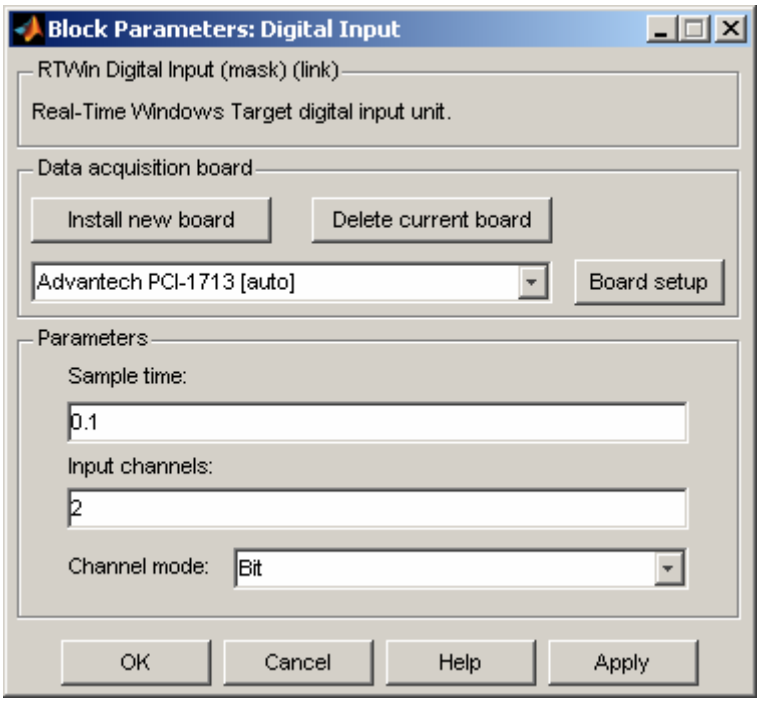
Şekil 3.9’da gösterilen buton tıklandıktan sonra Şekil 3.10’da gösterildiği gibi gelen ekranda bu örnekte kullanılacak DAQ kartı seçilir.



Şekil 3.10 Real Time için Kullanılacak DAQ Kartının Seçilmesi

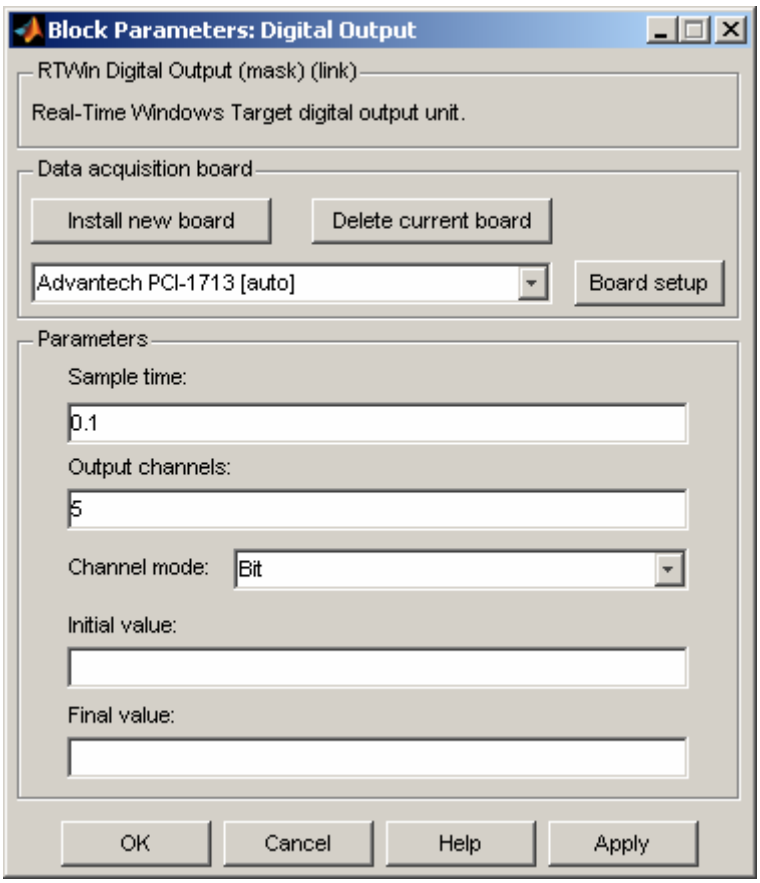
Not : Eğer paralel port kullanılmak istenirse açılan listeden "Standard Devices" listesi altında yer alan "Paralel Port" aygıtı seçilmelidir.

Daha sonra Digital Input bloğuna ait özellikler penceresinin "Data Acqusition Board " bölümüne gelen listeden bu örnekte kullanılmak üzere az önce Simulink ortamında yüklediğimiz kart seçilir. Bu pencereye ait kullanılacak giriş kanalı ve örneklem süresinin belirlenmesi ile ilgili ayarlar Şekil-11'de gösterilmiştir.



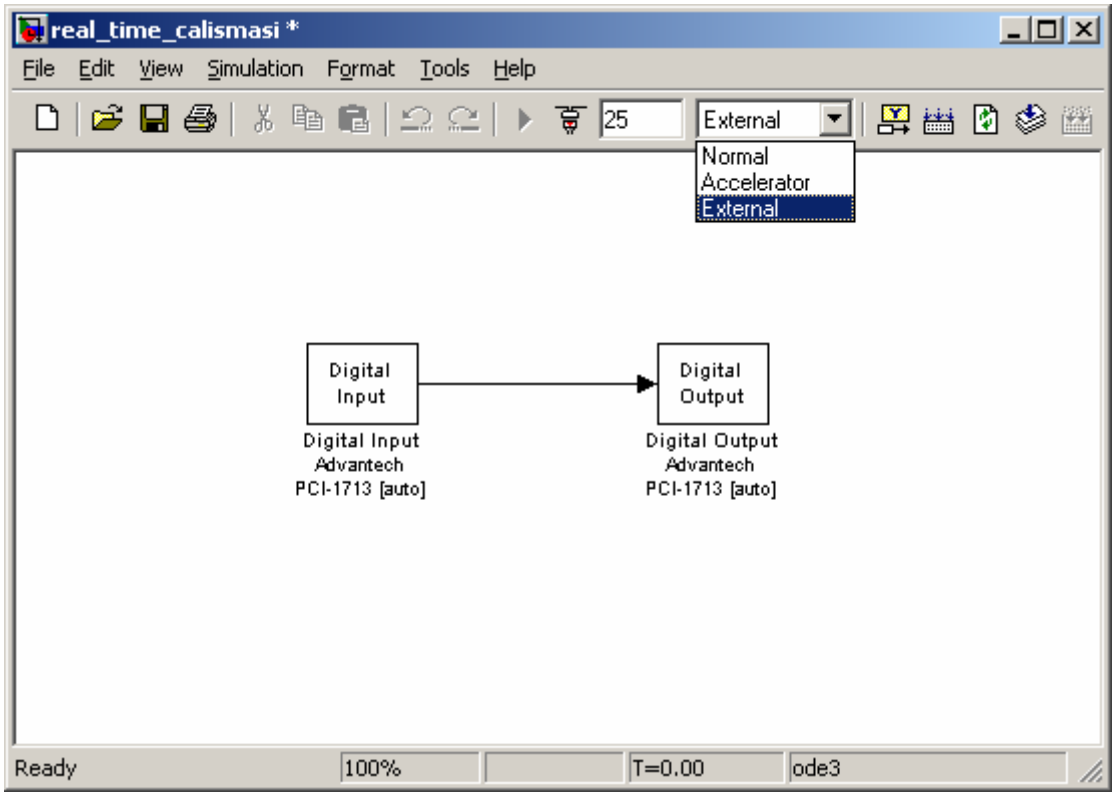
Şekil 3.11 Digital Input Ortamı İçin Kullanılacak DAQ Kartının Ayarlarının Yapılması

Daha sonra OK butonu tıklanarak "Digital Input" bloğuna ait özellikler penceresi kapatılır. Benzer işlemler "Digital Output" bloğu için de yapılır. Bu bloğa ait ayrlar Şekil 3.12'de gösterilmiştir.



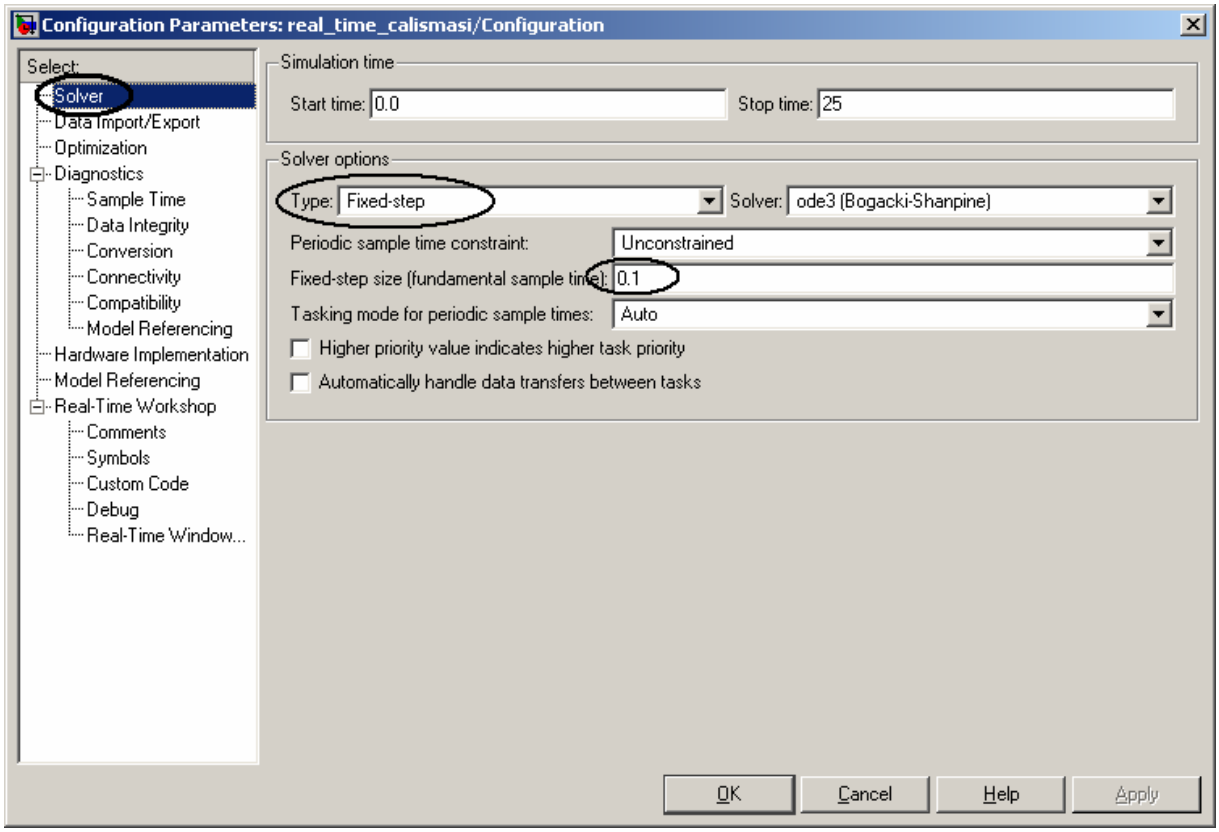
Şekil 3.12 Digital Output Ortamı İçin Kullanılacak DAQ Kartının Ayarlarının Yapılması

Tüm bu adımlar tamamlandıktan sonra gerekli blok diyagramı tasarımı sona ermişitr. Bu adımdan sonra Simulink çalışma alanında External seçeneği seçilmelidir. Bu durum için Şekil 3.13'e bakılabilir.



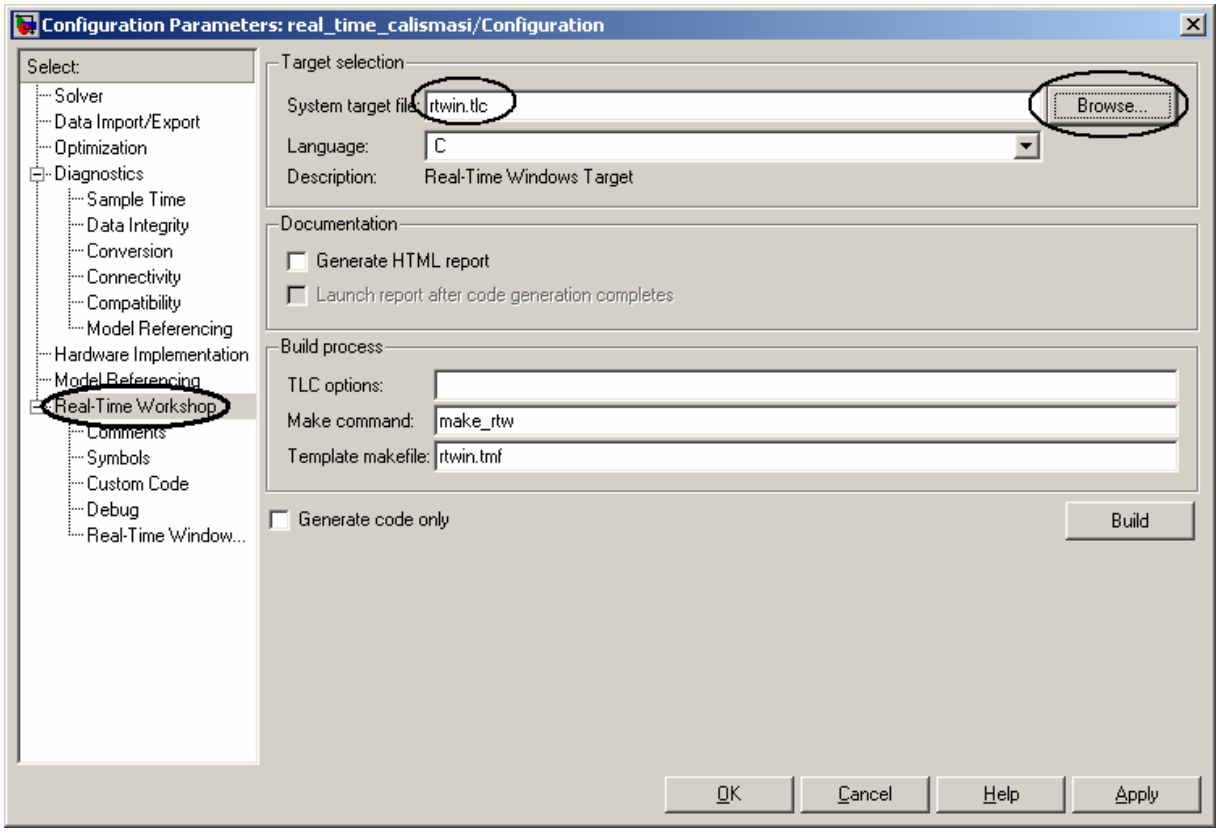
Şekil 3.13 Real Time Çalışma İçin External Seçeneğinin Aktif Edilmesi

Daha sonra "Simulation" menüsünden "Configuration Parameters" komutu kullanılarak gerekli iki ayar yapılmalıdır. Gelen pencerede sol tarafta yer alan Solver dalı seçilerek yandaki alanın "Solver options" bölümünden Type liste kutusundan "Fixed-Step" seçilir ve hemen sonra altaki giriş kutularından "Fixed-step size" alanına 0.1 değeri girilir. Bu ayarlama Şekil 3.14'ten de takip edilebilir.



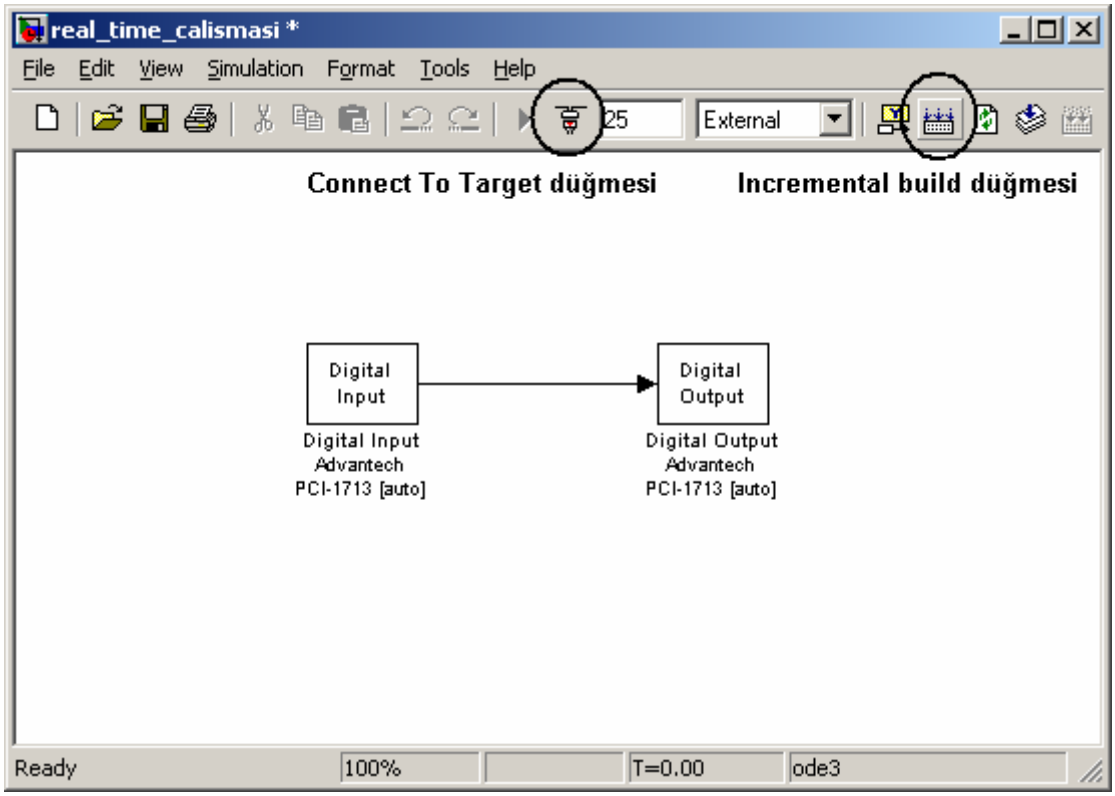
Şekil 3.15 Real Time Çalışma için Gerekli Ayarlamalar-1

Bir diğer ayar ise yine aynı pencerenin sol tarafında yer alan dallarından "Real-Time Workshop" seçilerek aynı pencerenin sağ tarafında gözüken arayüzde "Target selection" bölümü kullanılarak "Browse" düğmesi aracılığıyla "System target file" seçeneğinin "rtwin.tlc" olarak değiştirilmesidir. Bu ayar için Şekil 3.16'ya bakılabilir. Daha sonra "Configuration Parameters" penceresi OK butonu tıklanılarak kapatılır.



Şekil 3.16 Real Time Çalışma için System Target File Ayarının Değiştirilmesi

Bu işlemden sonra hazırlanan Real Time simulink modeli çalışma alanı penceresinin araç çubuğunda yer alana Save ikonu tıklanılarak ve bir isim verilerek Matlab'in kurulu olduğu dizin içinde yer alan "work" klasörüne kaydedilmelidir. Daha sonra Şekil 3.17'de gösterildiği üzere "Incremental build" butonu kullanılarak bu simulink modelinde kullanılacak DAQ kartı ile ilgili kütüphaneler kullanılması ve bu karta ait gerekli makine kodu dosyalarının üretilmesi sağlanmalıdır.



Şekil 3.17 Real Time Çalışma için DAQ Kartına Ait Makine Kodu Dosyalarının İnşası

Bu adımdan sonra arka planda çeşitli işlemler yapılacaktır. Bu işlemler "Command Window"dan takip edilebilir. "Command Window" penceresinde "succesfully builded" mesajı alındıktan kart ile bilgisayar arasında iletişimin kurulduğunu test etmek üzere Şekil 3.17'de gösterildiği üzere "Connect To Target" düğmesi kullanılabilir. Eğer iletişim kuruldu ve herhangi bir hata mesajı alınmamış ise kart ile bilgisayar arasında bağlantı doğru olarak kurulmuştur. En son olarak da Real Time uygulamsı Simulink çalışma alanından "Start simulation" düğmesi kullanılarak çalıştırılır.

3.5 Simulink Modellerinin Matlab Komutları ile Yönetilmesi

Matlab komut satırından simulink modelleri açmak, parametrelerini değiştirmek ya da çalıştırmak, durdurmak gibi işlemler yapılabilir. Bu işlemlerin nasıl yapılacağı ile ilgili detaylı bilgiler aşağıdaki alt başlıklarda anlatılmıştır.

3.5.1 Bir Simulink Modelinin Açılması Ve Kapatılması

Bir simulink model penceresini açmak için "open_system" ve kapatmak için "close_system" komutları kullanılır. Bu komutların nasıl kullanılacağı aşağıda verilmiştir.

```
open_system('deneme_model.mdl');
close_system('deneme_model.mdl',0);      % 0 parametresi pencereyi değişiklikleri modele
close_system('deneme_model.mdl',1);      % kaydetmeden kapatır, 1 ise kaydeder ve kapatır
```

Ayrıca, herhangi bir dizin altında bulunan model dosyası için kullanım şekli de şöyledir:

```
open_system('c:\models\deneme_model.mdl');
close_system('deneme_model.mdl',0);
```

3.5.2 Bir Simulink Modelinin Çalıştırılması

Bir simulink modelinin çalıştırılması için “sim” komutu kullanılır. Bu komutun örnek kullanımı aşağıda verilmiştir.

```
sim('deneme_model');
```

3.5.3 Bir Simulink Modeline Ait Parametrelerinin Değiştirilmesi

Bir simulink modeline ait pek çok farklı özelik vardır. Bu özellikler ve nasıl kullanılacağı ile ilgili örnek kullanım şekilleri aşağıda listelenmiştir.

```
set_param(‘deneme_model','SimulationCommand','Start');
set_param('deneme_model','StartTime','5','StopTime','100');
set_param('deneme_model','Solver','ode15s','MaxOrder','3');
set_param('deneme_model','SaveFcn','my_save_cb')

set_param('deneme_model','SimulationCommand','start');
set_param('deneme_model','SimulationCommand','stop');
set_param('deneme_model','SimulationCommand','pause');
set_param('deneme_model','SimulationCommand','continue');
set_param('deneme_model','SimulationCommand','step');
set_param('deneme_model','SimulationCommand','update');

set_param('deneme_model','Solver','VariableStepDiscrete',{'ode45'});
set_param('deneme_model','Solver','VariableStepDiscrete','ode45');
set_param('deneme_model','Solver','VariableStepDiscrete','ode23');
set_param('deneme_model','Solver','VariableStepDiscrete','ode113');
set_param('deneme_model','Solver','VariableStepDiscrete','ode15s');
set_param('deneme_model','Solver','VariableStepDiscrete','ode23s');
set_param('deneme_model','Solver','VariableStepDiscrete','ode23t');
set_param('deneme_model','Solver','VariableStepDiscrete','ode23tb');
set_param('deneme_model','Solver','FixedStepDiscrete',{'ode5'});
set_param('deneme_model','Solver','FixedStepDiscrete','ode5');
set_param('deneme_model','Solver','FixedStepDiscrete','ode4');
set_param('deneme_model','Solver','FixedStepDiscrete','ode3');
set_param('deneme_model','Solver','FixedStepDiscrete','ode2');
set_param('deneme_model','Solver','FixedStepDiscrete','ode1');
set_param('deneme_model','Solver','FixedStepDiscrete','ode14x');
```

3.5.4 Bir Simulink Modeline Ait Blokların Özelliklerinin Değiştirilmesi ve Okunması

Bir simulink modelinde çok farklı bloklar olabilir. Bu bloklarıon özellikleri de komut satırından değiştirilebilir. Bloktan bloğa değişik ve çok farklı özellikler bulunmaktadır. Ancak, burada temel özelliklerin nasıl değiştirileceği ile ilgili kullanım şekli verilmiştir. Bir simulink model bloğu özelliğinin değiştirilemsi için “set_param” ve özelliklerinin okunması

içinde benzere şekilde kullanıma sahip olan “get_param” komutları kullnılır. Örnek kullanım şekli aşağıda verilmiştir.

```
set_param('deneme_model/transfer_function_block','Numerator',’[ 1 ]’);
set_param('deneme_model/transfer_function_block','Denominator',’ [ 1 6 5 0 ]’);
set_param('deneme_model/constant_blok','Value', ‘5’);
set_param('deneme_model/relay_blok','OnSwitchValue',’0.4’);
set_param('deneme_model/relay_blok','OffSwitchValue',’0.2’);
```

Ancak, eğer özelliği değiştirilecek blok bir subsystem bloğu ise bu takdirde özellikler değiştiilmesi için önce bilgiler sütun vektör halşnde bir hücre matris değişkenine atanır ve daha sonra bu değişken “MaskValues” özelliğine atanmalıdır. Bu değerlerin atanabilmesi için subsystem için “SubSystem Parameters” tanımının yapılmış olması gerekir.

```
pid_values_cell = { Kp , Ki , Kd } ;
set_param ( 'pid_by_kenan_return_mdl / PID_Controller','MaskValues',pid_values_cell);
```

3.5.5 Bir Simulink Modeline Ait Verilerin Matlab Komut Satırından Erişilmesi

Simulink modelinden verilerin alınması için simulink bloklarından “To Workspace” kullanılmalıdır. Eğer, zaman bilgisi atanacaksa blok özelliklerin “Structure with Time” seçeneği seçilmelidir. Çok önemli bir nokta şudur ki simulink penceresi kapatılmadan önce mutlaka değerler “save” komutu kullanılarak bir “.mat” türü dosyay kaydedilmelidir. Bu değerler daha sonra “load” komutu kullanılarak önceden kaydedilen “.mat” dosyasından geri “Workspace” alanına yüklenebilir. Örnek kullanım şekli için aşağıdaki komut yapısına bakılmalıdır.

Simulink model çıkış bilgilerini kaydetmek için komut satırları şunlardır:

```
sim('pid_controller_model')  % mutlaka modelin içinde “To Workspace” bloğu olmalıdır.
save('pid_results');                 % “To Workspace” özelliklerinden değişken ismi “pid_output”
close_system('pid_controller_model ',0);      %yazıldığı kabul edilmiş olsun.
```

Kaydedilen bilgileri kullanmak ve bir grafik çizdirmek için komut satırları şunlardır:

```
load('pid_results');
zaman_araligi = pid_output.time;
kontrolor_cikis_degerleri = pid_output.signals.values;
plot (zaman_araligi, kontrolor_cikis_degerleri);
```

Eğer çıkış değerleri birden fazla ise bu takdirde satır vektörü yapısında bilgiler elde edilmelidir. Örnek komut satırları aşağıda verilmiştir.

```
zaman_araligi = pid_output.time;
kontrolor1_cikisi = pid_output.signals.values( : , 1 );
kontrolor2_cikisi = pid_output.signals.values( : , 2 );
plot (zaman_araligi, kontrolor1_cikisi, zaman_araligi, kontrolor2_cikisi);
```

BÖLÜM-IV

MATLAB'TE DIŞ DÜNYADAN DATALARIN ALINMASI (IMPORT EDİLMESİ)

4.1 Giriş

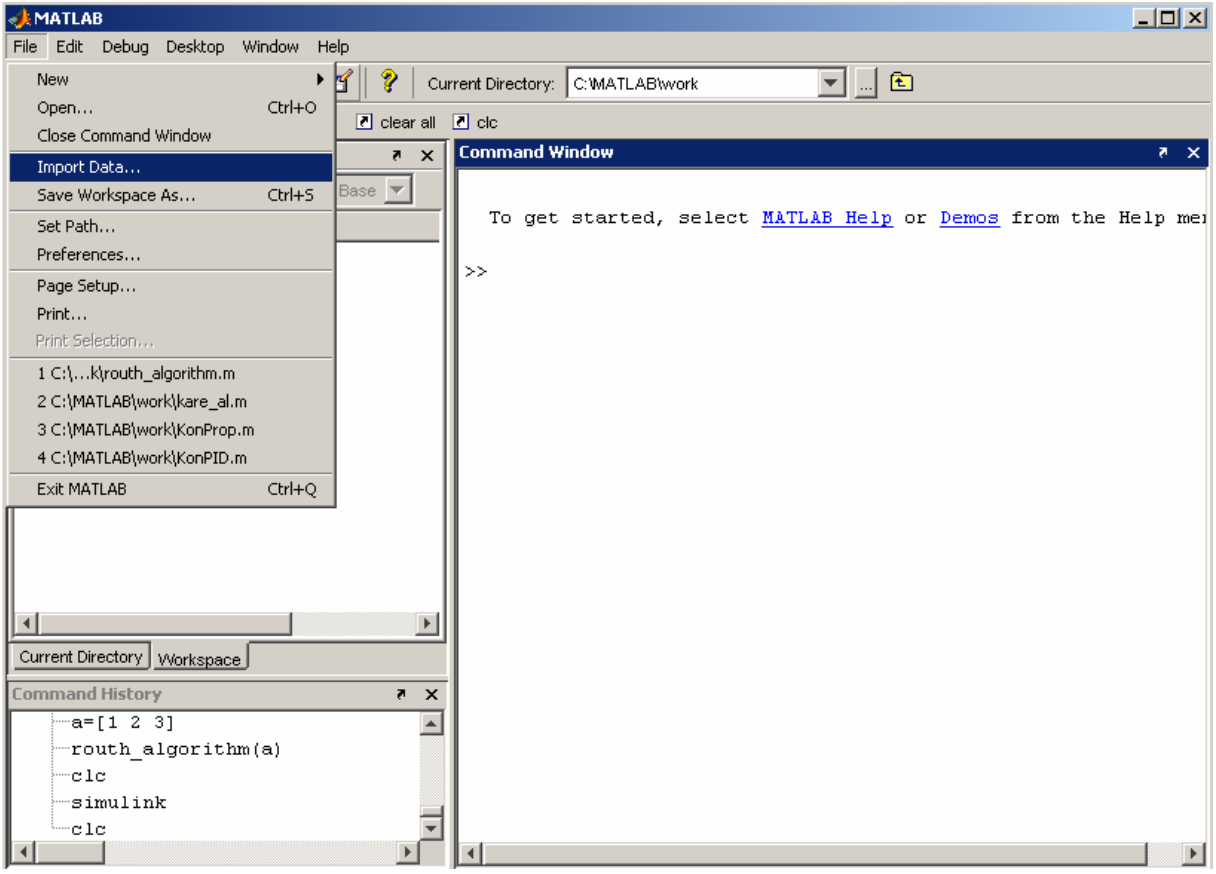
Matlab kullanılarak çok çeşitli sayıda formatlara sahip dosyalar içerisndeki veriler "Workspace"e import edilebilir (dışarıdan yüklenebilir). Böylelikle herhangi bir ortamda yer alan verilerin Matlab ile kullanılması ve işlenebilmesine olanak sağlanmış olunur.

4.2 Dataların Import Edilmesi

Herhangi bir dosyanın içerdiği verilerin Workspace'e import edilmesi iki farklı yolla olur. Bu yollar aşağıda ayrıntılı olarak anlatılmıştır.

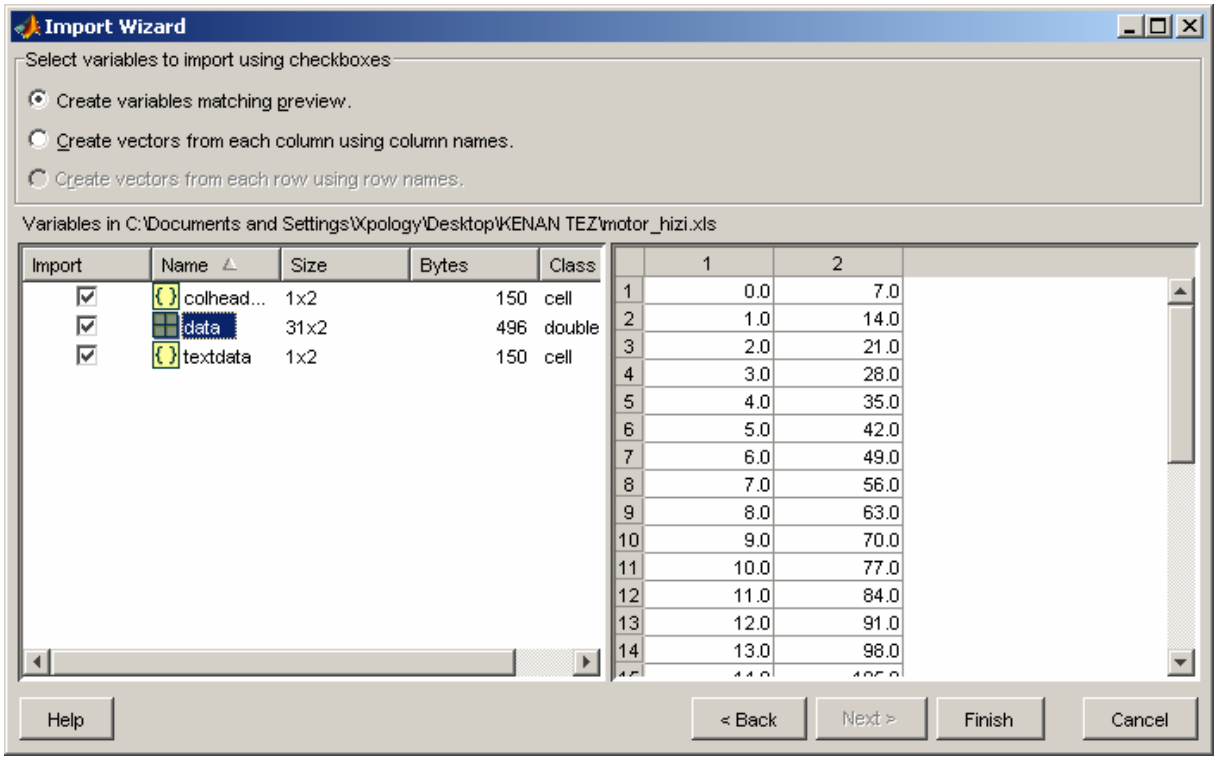
4.2.1 Dataların Import Edilmesi için 1. Yöntem

Matlab komut satırı ekranında iken "File" menüsünden "Import Data…" komutu verilir. Bu durum Şekil 4.1'de gösterilmiştir.



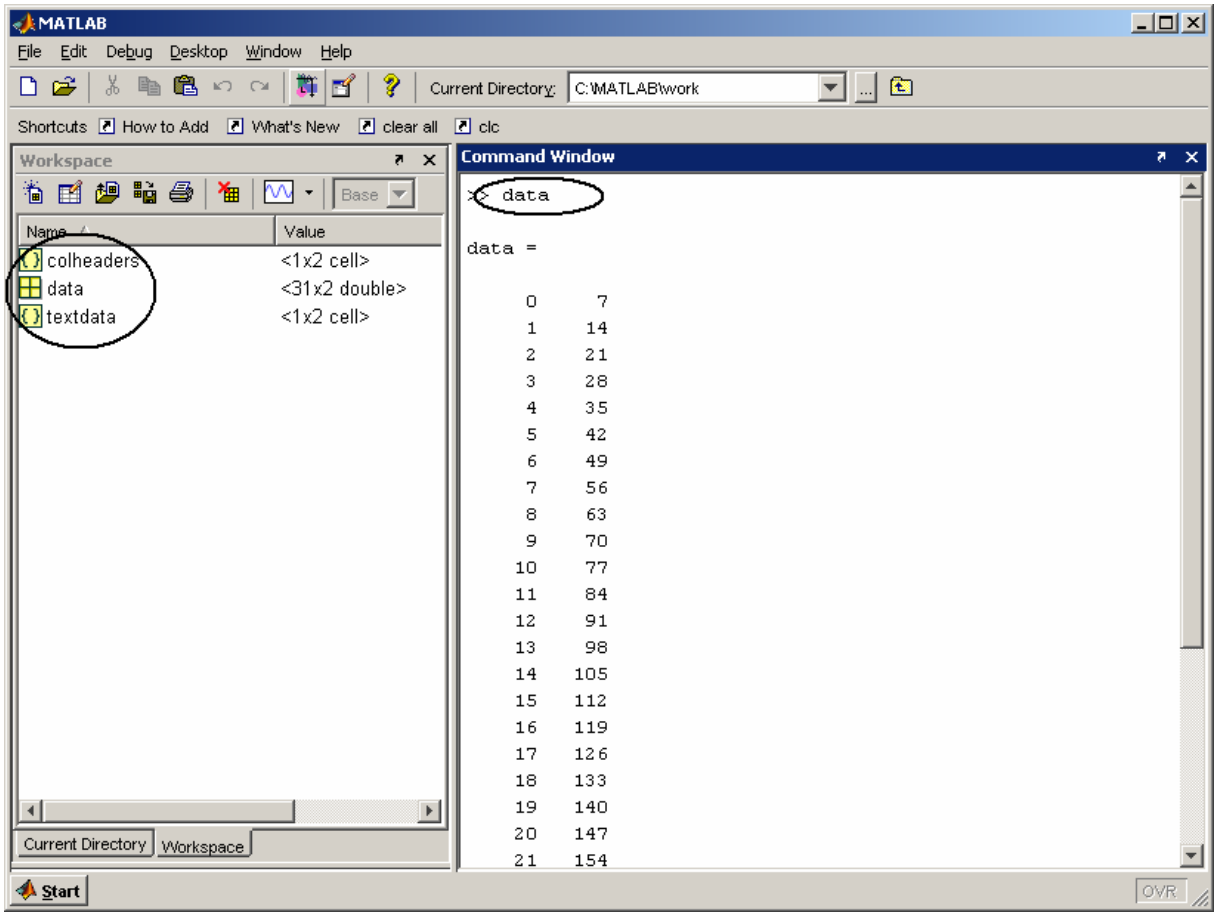
Şekil 4.1 Matlab Ortamına Verilerin File Menüsü Yardımıyla Import Edilmesi

Daha sonra ekrana gelecek “Dosya Aç” penceresinden içerisinden data import edilecek dosya seçilir ve Tamam butonuna tıklanır. Buradaki örnekte bir Excel dosyası içerisinden veriler import edilme istenmektedir. Bu adımdan sonra kullanıcı Şekil 4.2’deki ekran görüntüsüne benzer bir görüntü ile karşılaşacaktır. (Import Wizard penceresi dosyanın formatına göre farklı arayüzlerde olabilir.) Gelen pencerede gözüken alanlar ile sizin seçmiş olduğunuz dosya içindeki veri alanları farklı olacaktır. “Import Wizard” penceresinin sol tarafından içerisinden veri eklenmek istenilen dosyada bulunan tüm alanlara ait veri kümelerinden biri seçilir. Sağ tarafta o alanın verileri gözükecektir. Seçilen veriler örnekteki gibi “data” kümesinden alınmış olsun. Daha sonra bu pencere “Finish” butonu tıklanılarak kapatılır.



Şekil 4.2 Import Wizard Penceresi ile Bir Dosyadan Verilerin Import Edilmesi

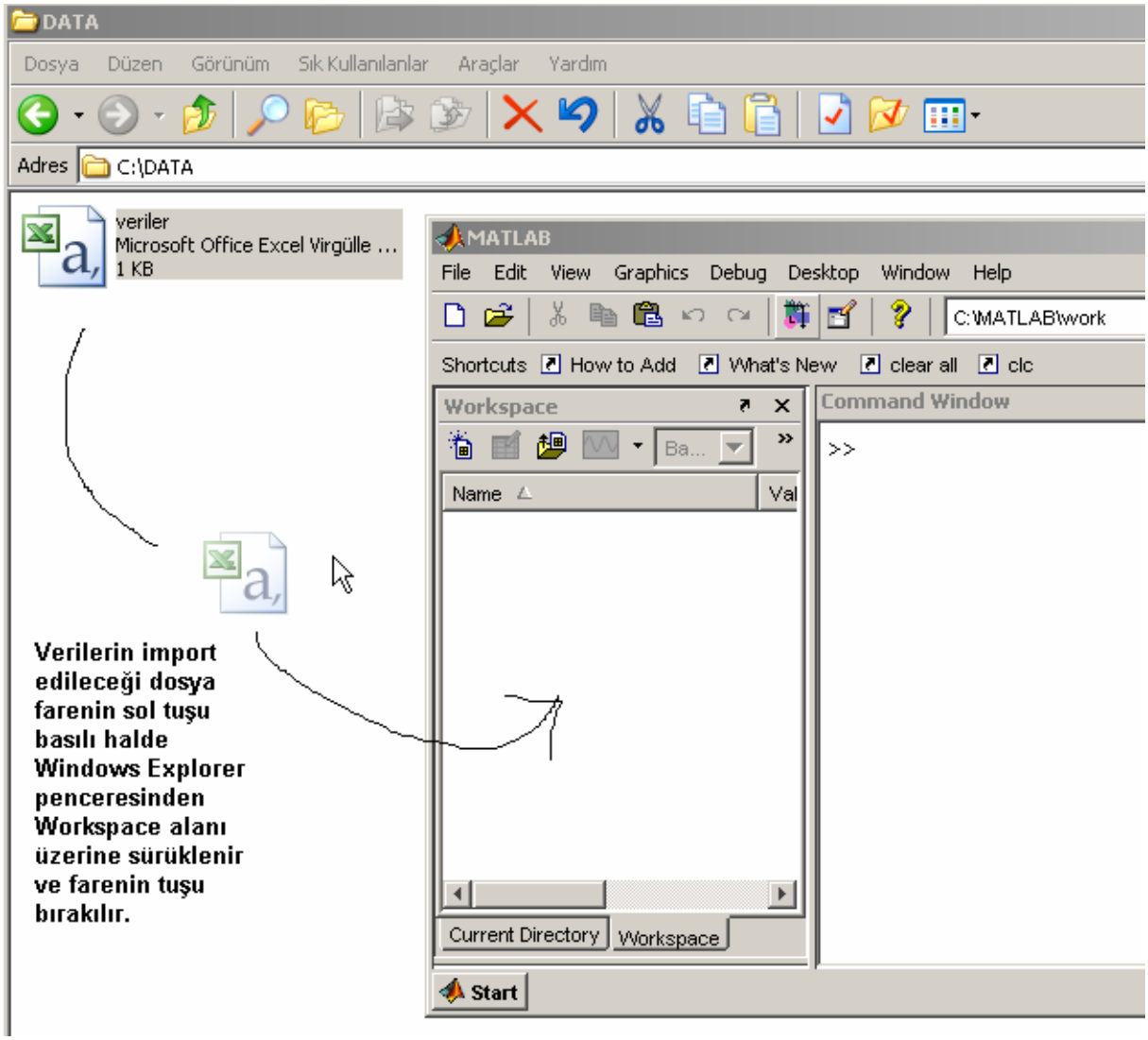
“Import Wizard” penceresi kapandığında Matlab ana penceresinden Workspace alanını kontrol ettiğimizde tüm verilerin geldiği görülür. Bu durum Şekil 4.3’te de gösterilmiştir. Komut satırından “data” değişkeninin ismi yazıldığında import edilen verilerin listelendiği görülür.



Şekil 4.3 Workspace Alanına Import Edilen Verilerin Listelenmesi

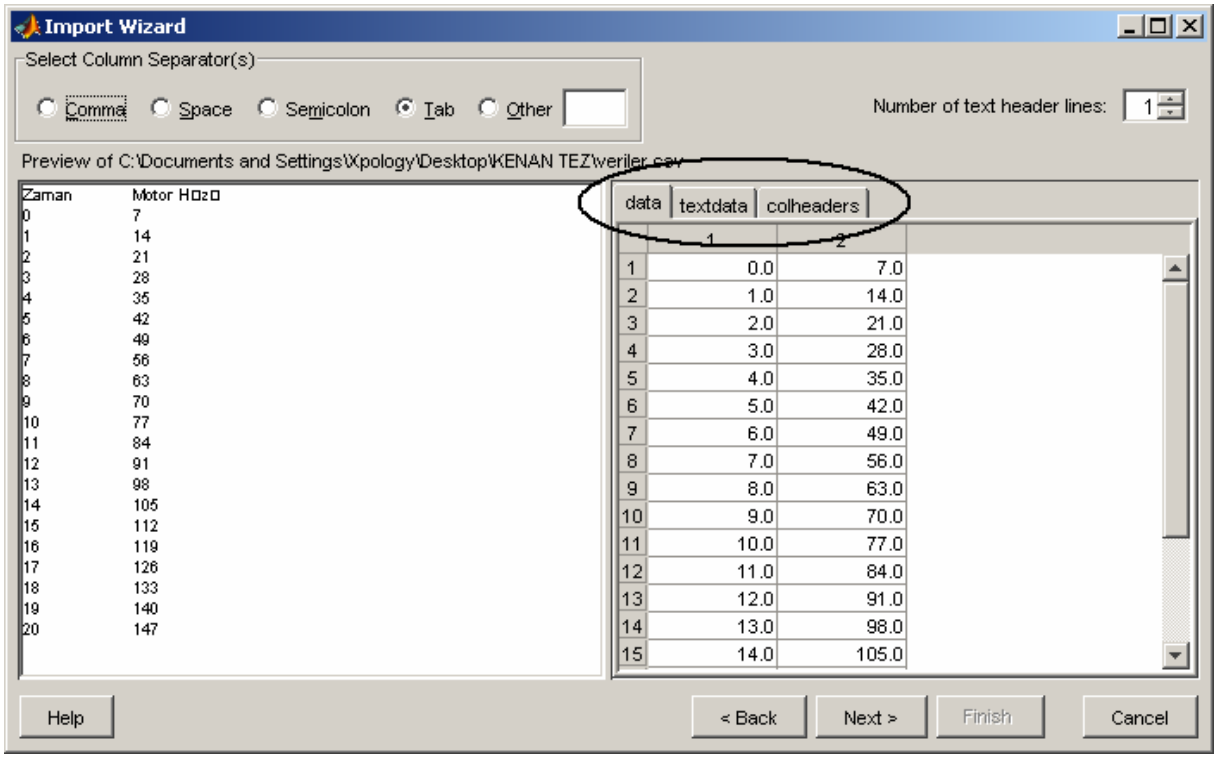
4.2.2 Dataların Import Edilmesi için 2. Yöntem

Dosyaların içindeki verileri "Workspace" alanına import etmenin bir diğer yolu da "Windows Explorer" penceresinden dosyayı farenin sol tuşu ile tıklamak ve farenin tuşunu bırakmadan basılı hale Matlab ana penceresinde "Workspace" alanının üzerinde iken farenin sol tuşunu



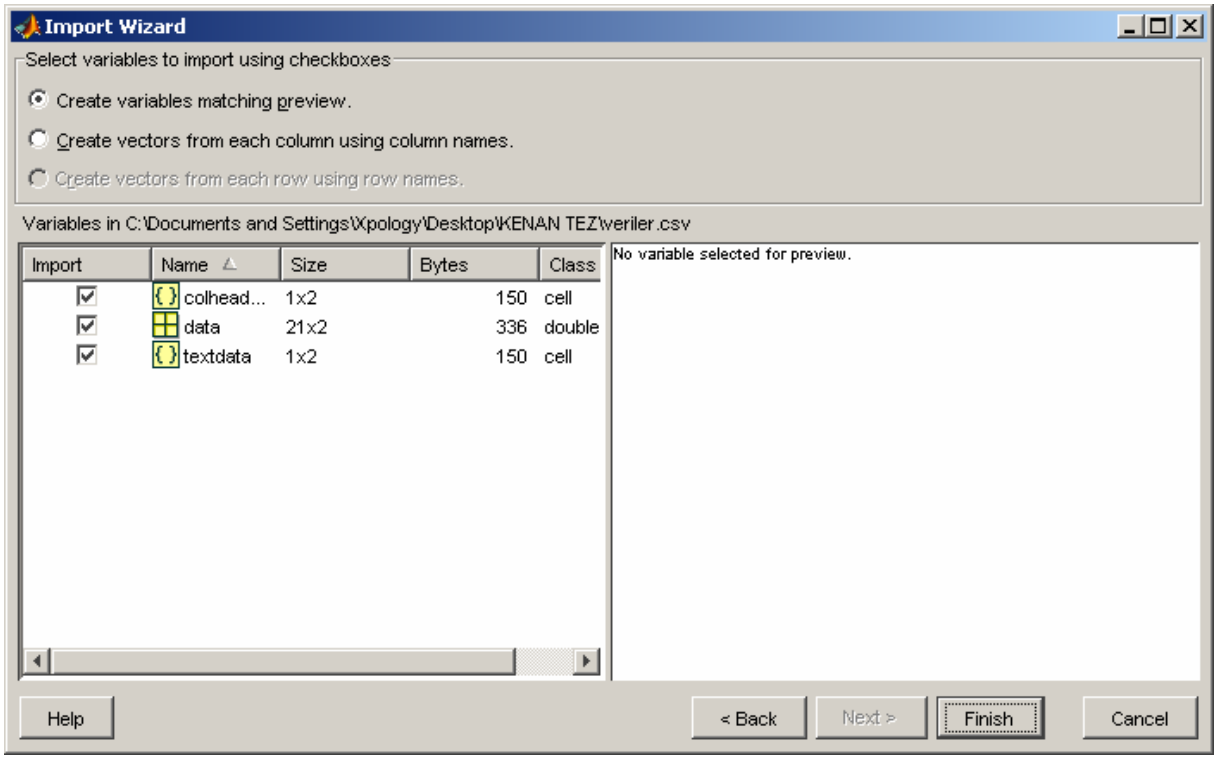
Şekil 4.4 Verilerin Import Edileceği Dosyanın Workspace Alanına Sürüklenmesi

bırakmaktır. Yani, dosyaları "Workspace" alanına sürüklemektir. Bu durum Şekil 4.4'te gösterilmiştir. Örneğin csv uzantılı bir dosya sürüklenmiş olsun. Karşımıza Şekil 4.5'teki gibi bir "Import Wizard" penceresi gelecektir. Bu pencerede şekilde de gösterildiği üzerinde içerisinde verilerin yer aldığı data kümeleri tablar halinde listelenmiştir. Daha sonra "Next"



Şekil 4.5 Csv Uzantılı Bir Dosyadaki Verilerin Matlab Ortamına Import Edilmesi

butonuna tıklanılarak Şekil 4.6'daki ekran ile karşılaşılır. Bu ekranın sol tarafında yer alan



Şekil 4.6 Csv Dosya İçerisinden Import Edilecek Alanların (Veri Kümelerinin) Seçilmesi

listeden bir önceki ekranda gösterilen data kümelerinden hangisi import edilmesi gerektiyorsa o veri kümesi ismine listeden tik konulur. En son olarak “Finish” butonu tıklanılarak dataların “Workspace” alanına import edilmesi işlemi tamamlanmış olur.

Not : Import edilecek bir dosya içerisinde sayısal veriler var ise ve bu veriler için ondalık ayıraç virgül ise (Bu duruma örnek olarak Türkçe bölgesel ayarları verilebilir.) mutlak surette o dosya içeriisndeki “ , ” karakterlerini “ . ” ile değiştirin. Çünkü, Matlab ondalık ayıraç olarak sadece “ . ” karakterini kabul etmektedir.

Not : Çok çeşitli programlama dilleri kullanılarak kontrol edilen sistemlere ait bilgilerin bir dosyaya kaydedilmesinde csv formatının seçilmesi programcının verileri başaka ortamlara aktarmasında büyük kolaylıklar sağlar. Csv dosya formatı içeriği itibari ile aslında txt bir dosyadır. Ancak, uzantısı .csv olarak kaydedilmelidir. Dosya içeriği ise aralarında “ ; ” karakteri bulunan verilerdir. Veriler sayı veya karakter olabilir. Örneğin aşağıda bir motorun hızının zamana göre değişmesini gösteren verilerin dosya içindeki formatı görülmektedir.

```
Zaman;Motor Hızı
0;7
1;14.25
2;21.41
3;28.12
4;35
5;42.85
6;49.96
7;56
8;63.47
9;70.56
10;77
```

Ayrıca, csv dosyalar MS Excel programı kullanılarak açılabilir ve içerisinde yer alan veriler listelenen hücreler şeklinde yönetilebilir.

BÖLÜM-V

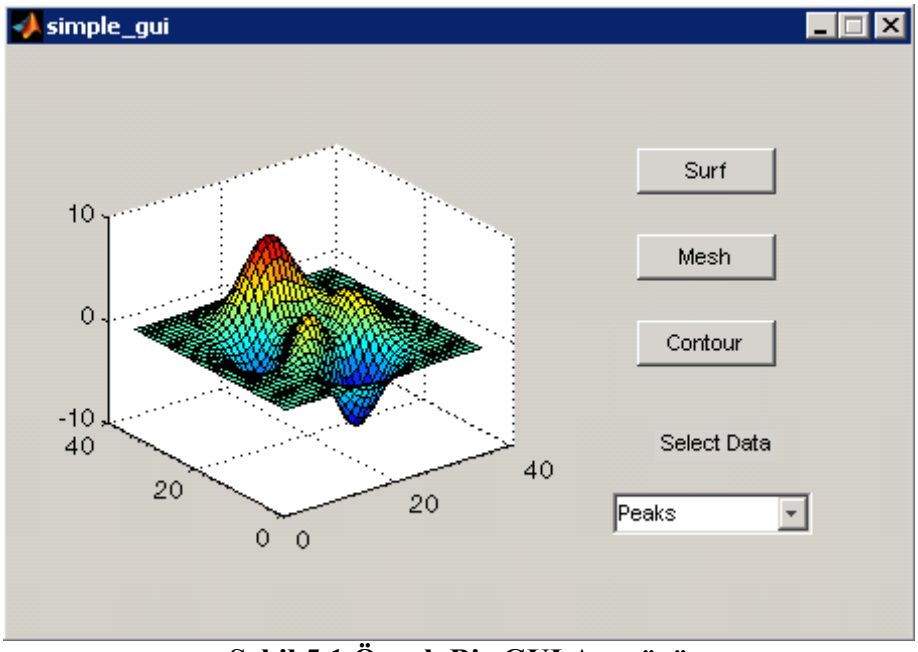
MATLAB'TE GRAFİKSEL KULLANICI ARABİRİMİ (GUI) TABANLI UYGULAMA TASARIMI

5.1 Giriş

İçeriğinde yer alan nesnelerin kullanılması ile kullanıcıya etkileşim sağlayan ve bir işin veya bir programın koşturulmasını sağlayan grafiksel bir program arayüzüdür. Açılımı Graphical User Interface (GUI) dir.

GUI nesneleri menüler, araç çubukları, radio butonlar, liste kutuları veya kaydırıcılar olabilir. Bunların yanında MATLAB GUI ile MATLAB'in sunduğu hesaplama imkânları kullanılarak da data alımı ve grafik çizimi gibi pek çok işlem gerçekleştirilebilir.

Şekil 5.1'de basitçe bir GUI arayüzü görülmektedir.



Şekil 5.1 Örnek Bir GUI Arayüzü

5.2 Grafiksel Kullanıcı Arabirimi (GUI) Nasıl Çalışır?

Her bir nesne (veya komponent) GUI için tanımlanan programlama dosyasında callback diye adlandırılan ayrı alt rutin programlama parçalarına sahiptir. Bu şekilde her bir nesnede oluşan olaylara (örnek olarak bir buton nesnesinin tıklanması ile click event oluşması gibi) GUI o olaya ait callback rutinlerini icra ettirir. Yani, GUI hem bir arayüz hem de bir program çağrılarını icra ettirme mekanizması olarak çalışır.

Yukarıda bahsedilen programlama olay tabanlı programlama diye adlandırılır. Bu tür programlamada her bir olaylara ait alt program parçaları birbirinden bağımsız olarak MATLAB GUI tarafından çalıştırılır.

5.3 Matlab'te GUI Oluşturma Yöntemleri

MATLAB GUI tasarımları iki ayrı yöntem kullanılarak yapılabilir. Bunlar,

- **MATLAB GUIDE aracı kullanılarak,**
- **M-File programlama yöntemi kullanılarak**

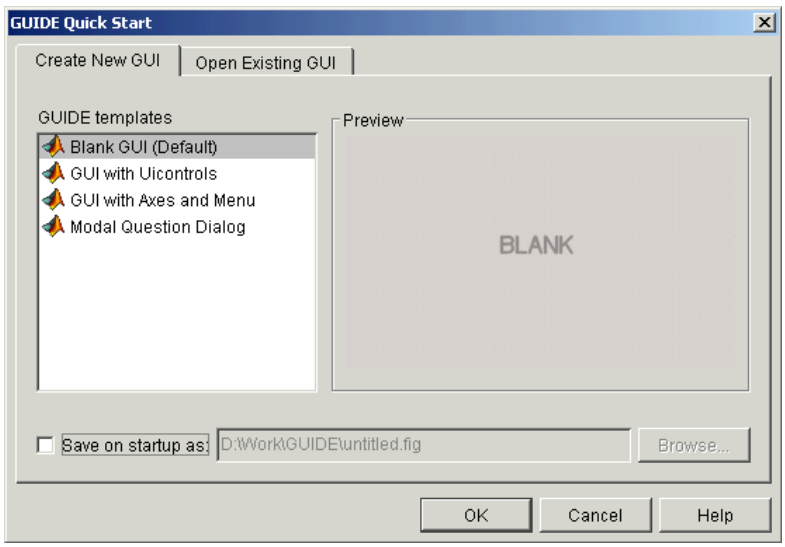
Özellikle GUI tasarımında hızlı arayüzler dizayn etmek ve bu işe ilk başlayan programcılar için MATLAB GUIDE aracının kullanılması büyük bir kolaylık sağlar. Bu aracın kullanılması ile GUI arabirimi kolaylıkla ve yorulmadan sürükle bırak ve açılan pencerelerde özelliklerin değiştirilmesine dayanan bir yöntem kullanılır. Ayrıca, bu yöntemi kullanmanın ileride var olan bir GUI nin düzenlenmesi ve değişiklik yapılması bakımından da çok yararlıdır.

M-File programlama yönteminde tüm GUI tasarımları ve callback program parçalarının yazılması tamami ile programlama kodları kullanılarak yapılır. Burada tasarımcı her şeye hakimdir ve bu teknik uzman bir programlama bilgisi gerektirir. Bu yöntem ile tasarım zamanı uzamasına rağmen programcı her türlü manipülasyonu yapabildiği için programcı açısından çok yararlıdır.

5.4 MATLAB GUIDE Aracı ile GUI Tasarımı Oluşturma

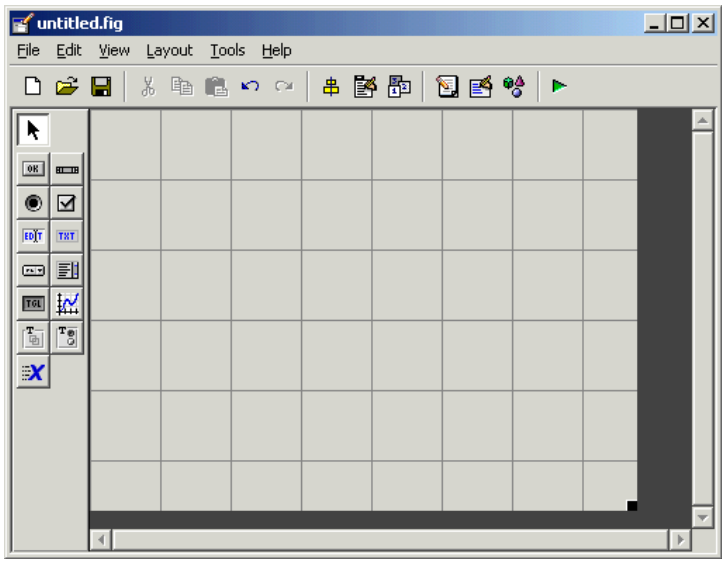
GUIDE matlabin GUI tasarımcılarına sunduğu içerisinde çeşitli araçlar içeren ve kolaylık sağlayan bir grafiksel GUI geliştirme ortamıdır. GUIDE kullanılarak tıkla ve sürükle-burak tekniği ile GUI arayüzüne nesneler (örneğin butonlar, text kutuları, liste kutuları, grafikler v.s.) kolaylıkla eklenebilir. Ayrıca, eklenen nesnelerin hizalanması, tab sırasının değiştirilmesi, görsel ayarlar üzerinde manipülasyonlar yapılması da bu ortamın tasarımcılara sunduğu imkânlardan bazılarıdır.

MATLAB GUIDE aracını tanıyalım. Bu aracını çalıştırmak için ya MATLAB komut satırından GUIDE komutu verilir ya da Start düğmesi tıklanarak MATLAB/GUIDE komutu verilir. Bu adımdan sonra karşımıza Şekil 5.2'deki gibi bir pencere gelir.



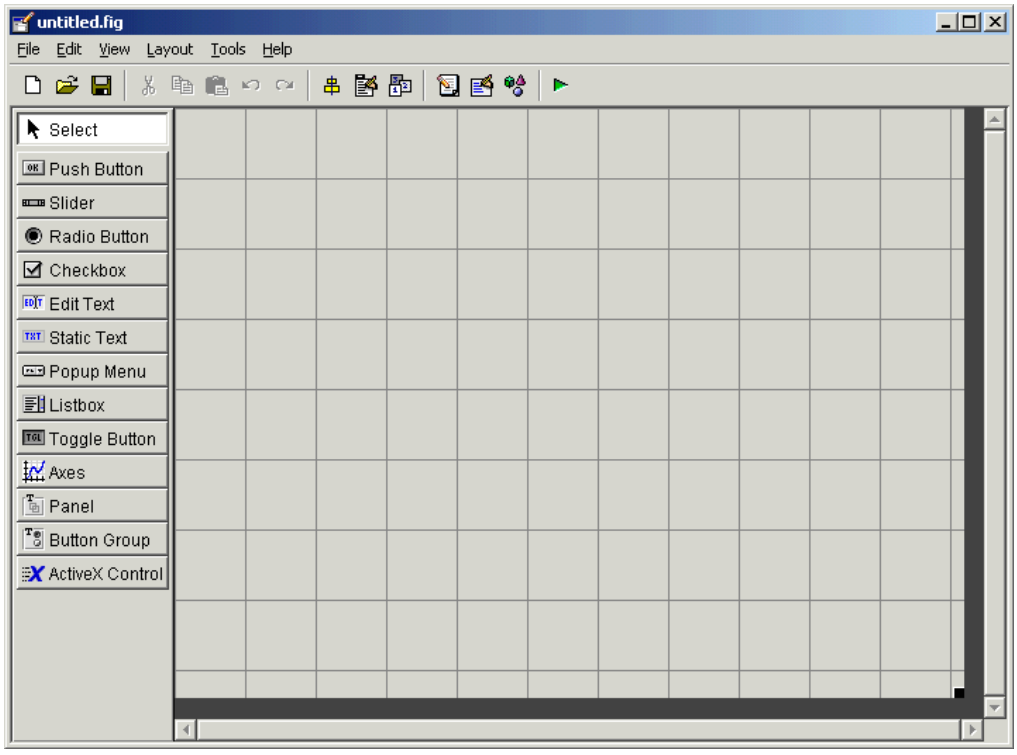
Şekil 5.2

Bu pencereden eğer yeni bir GUI tasarımı yapacak isek Blank GUI seçeneğini seçeriz. Şayet önceden yapılmış bir tasarımı açmak istiyor isek Open Existing GUı sekmesinden sonra istenilen dosyayi seçeriz. Burada yeni bir tasraım oluşturulacağını kabul edelim. Bundan sonra OK düğmesi tıklanılarak Şekil 5.3'teki GUIDE LAYOUT Editor (GUIDE Çalışma Alanı) penceresine ulaşırız.



Şekil 5.3

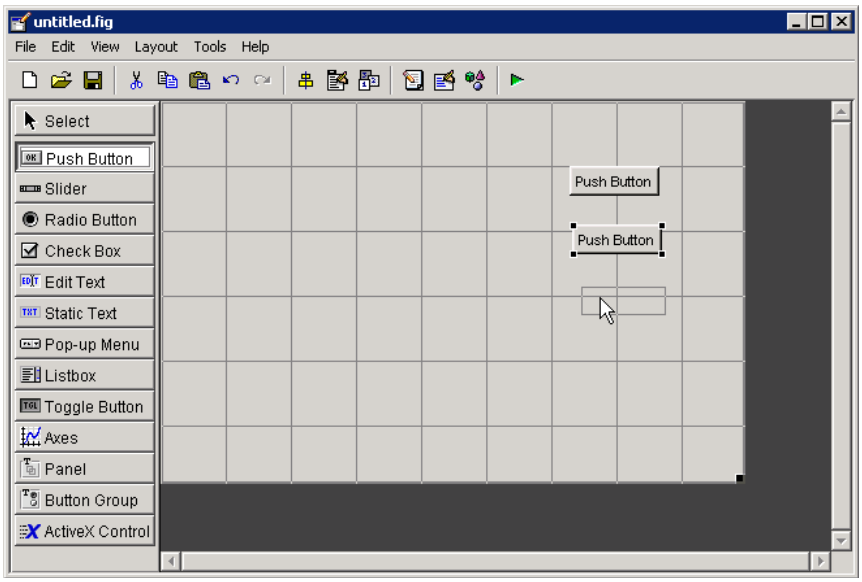
Bu adımdan sonra File/Prefences/Guide yolunu kullanılarak gelen pencereden "Show names in component palette" seçeneğini tiklayip OK düğmesine basalım. Karşımıza Şekil 5.4'teki gibi bir pencere gelecektir.



Şekil 5.4

5.4.1 Komponentleri Çalışma Alanına Ekleme

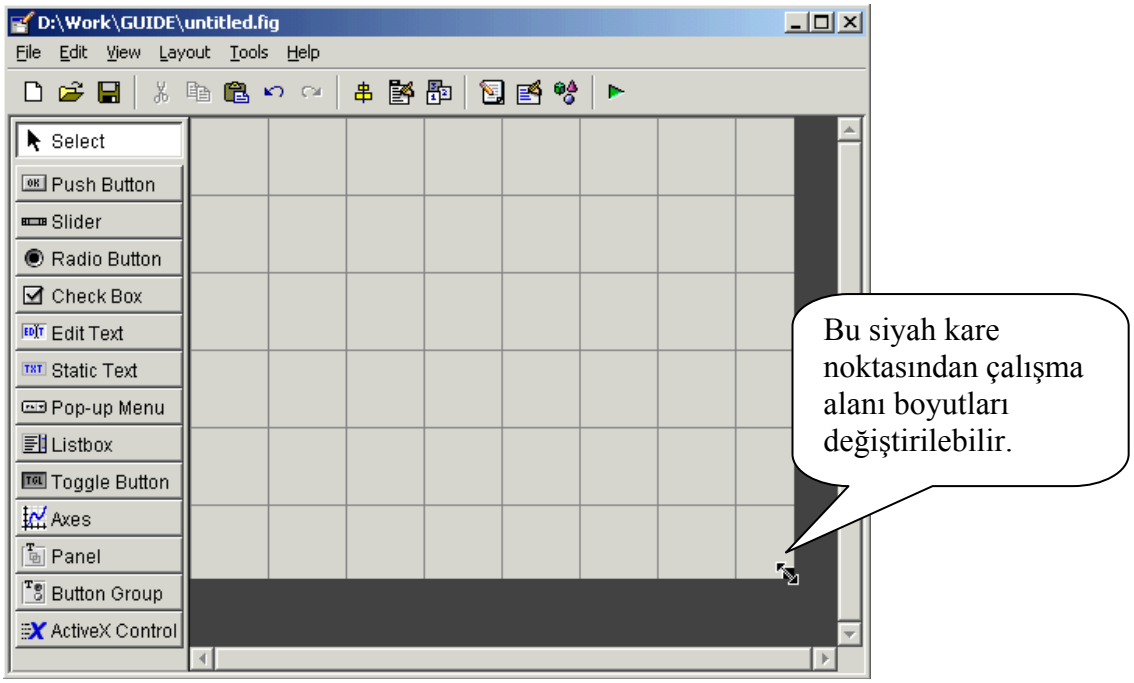
Bunun için sol tarafta bulunan nesne butonlarından istenilen nesneye ait buton tıklanır ve daha sonra çalışma alanında uygun görülen bir noktaya tıklandığında o noktaya ilgili nesne eklenmiş olacaktır. İstenirse çalışma alanındaki bir nesne farenin sol tuşu ile tıklanıp bırakılmadan çalışma alanının herhangi bir yerine sürüklenebilir. Bu durum Şekil 5.5'te de görülmektedir.



Şekil 5.5

5.4.2 Çalışma Alanının Boyutlarını Değiştirmek

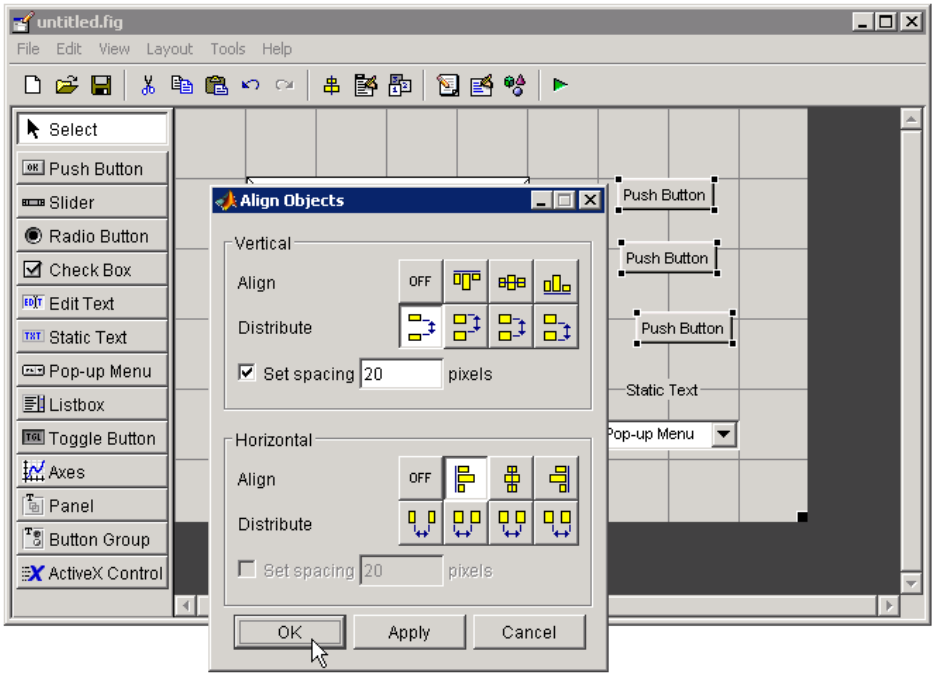
Burada da çalışma alanının sağ alt tarafında bulunan siyah karenin üzerine fare işaretçisi getirilir ve fare işaretçisi konum değiştirdiğinde farenin sol tuşu basılı tutularak çalışma alanı istenilen boyutlarda olacak şekilde düzenleme yapılabilir.



Şekil 5.6

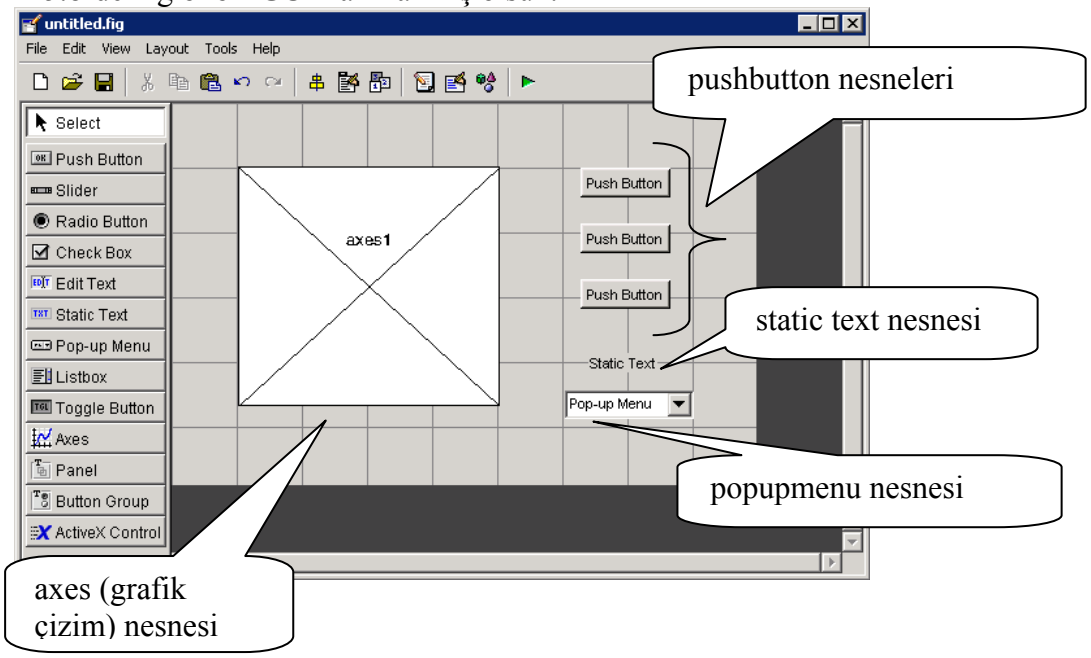
5.4.3 Nesneleri Hizalamak

Bu işlemi yapmak için öncelikle hizalanacak nesneler seçilir. Topluca seçmek için çalışma alanında fare işaretçisini herhangi bir yere tıklayıp sürükleyerek açılan kesik kenarlı pencerenin içinde nesneler kalacak şekilde hareket ettirip, hizalanacak nesneler bu çerçeve içinde kalınca farenin sol tuşunu bırakın. Bu şekilde sadece o çerçeve içinde kalan nesneler seçilmiş olacaktır. Ayrıca, nesneleri Ctrl tuşunu basılı tutarak farenin sol tuşu ile teker teker de seçme imkânı bulunmaktadır. Hizalanacak nesneler seçildikten sonra Tools/Align Objects… yolunu kullanarak Aligment Tool (Hizalama Aracı) penceresini açınız. Şekil 5.7'deki gibi bir ekran ile karşılaşırız. Burada yatay ve dikey hizalamaları kendimize göre butonlardan seçip OK butonuna bastığımız zaman nesnelerimiz hizalanmış olacaktır. Eğer ki hizalama istenilen gibi olmadı ise Ctrl + Z kısayolu ile yapılan işlemler geri alınabilir.



Şekil 5.7

Burada Şekil 5.8'deki gibi bir GUI hazırlanmış olsun.



Şekil 5.8

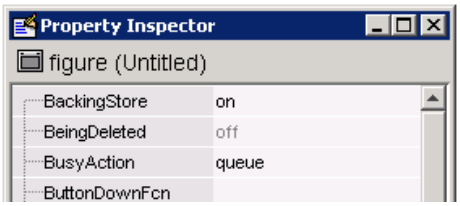
Burada GUI arayüzünde

- Bir adet grafik çizim (axes) nesnesi,
- Bir adet peak, membrane, sinc data setlerini gösteren popup menü,
- Bir adet popup menü başlığı sunan static text nesnesi,
- Üç adet surf, mesh ve contour yazılı buton nesneleri

yer almaktadır.

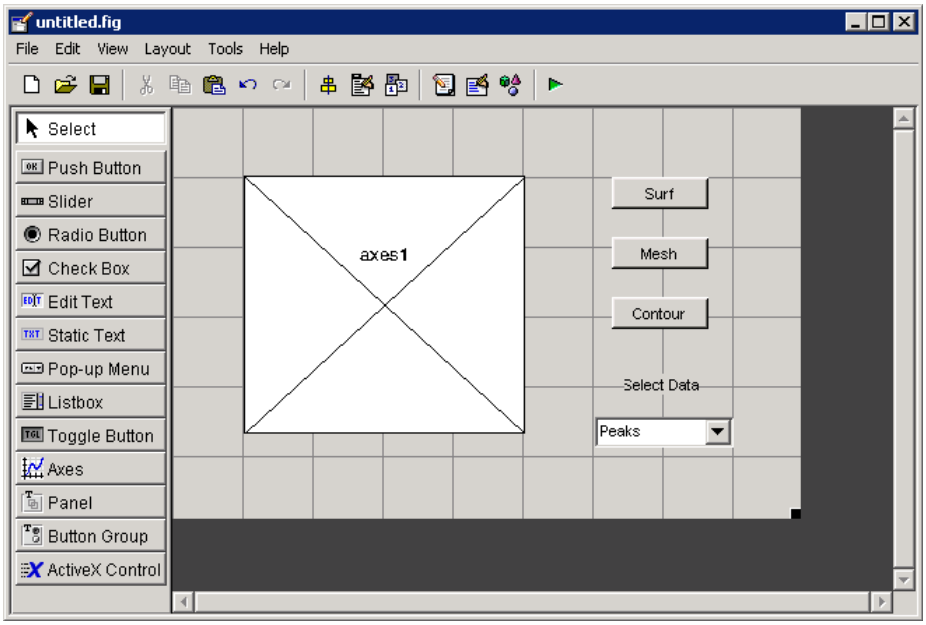
5.4.4 Nesnelere Yazı Ekleme ve Özelliklerini Değiştirme

Nesnelerin özelliklerini değiştirmek istersek ya ilgili nesne farenin sol tuşu ile çift tıklanır ya da ilgili önce seçilip daha sonra View/Property Inspector komutu ile özellikler penceresi



Şekil 5.9

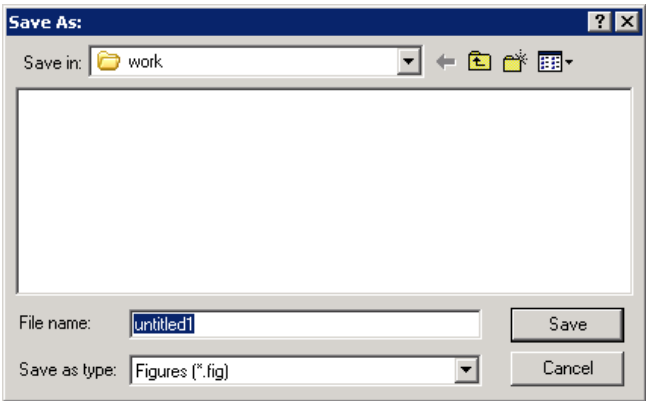
açılır. Buradan örneğimizde eklenen popup menu içeriğine Peaks, Membrane ve Sinc içeriklerini alt alta popup menu nesnesini seçtikten sonra String özelliğine ekleyiniz. Ayrıca, üç adet butonun her birine sırayla Surf, Mesh ve Contour yazıları String özelliklerine eklenmelidir. GUI arayüzü penceresi Şekil 5.10'daki gibi gözükecektir.



Şekil 5.10

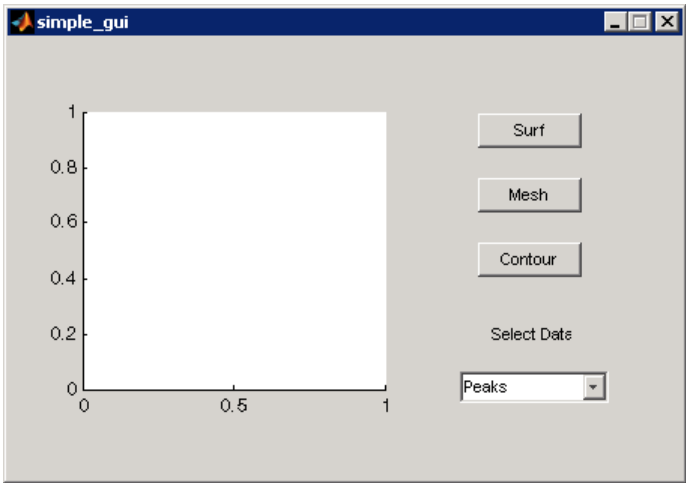
5.4.5 GUI Tasarımını Kaydetme ve Çalıştırma

Bundan sonra bitmiş olan bu GUI arayüzü çalıştırarak görmek için öncelikle Tools/Run yolundan Run (Çalıştır) komutu verilir. Daha sonra gelen pencereden çalışmamnın Run edilebilmesi için kaydedilmesi gerektiğini bildiren bir pencere çıkar Burada Yes butonuna basarız. Bu adımdan sonra MATLAB GUIDE bize tasarımın kaydedileceği dosya ismini



Şekil 5.11

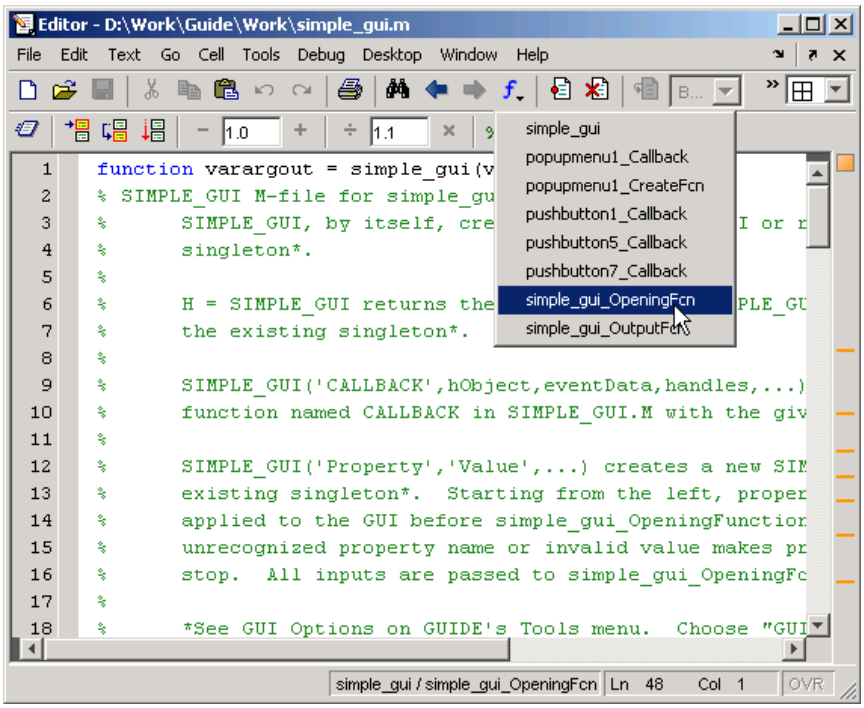
soran bir pencere getirir. Bu pencereden çalışmamıza bir isim vererek tasarımızı kaydetmiş oluruz. Ardından karşımıza Change the MATLAB Directory gibi bir ekran gelirse burada bu ekranı OK tuşuna basarak kapatabilirsiniz. Bu ekran kaydedilen dosya MATLAN tanımlıo dizinler dışında bir yere kaydedilme sözkonusu olduğunda bizi uyarmaktadır. Sonra da GUI tasarımımızın çalışması sonucu gözükecek uygulama penceresi ekranı karşımıza Şekil 5.12'deki gibi bir pencere gelecektir.



Şekil 5.12

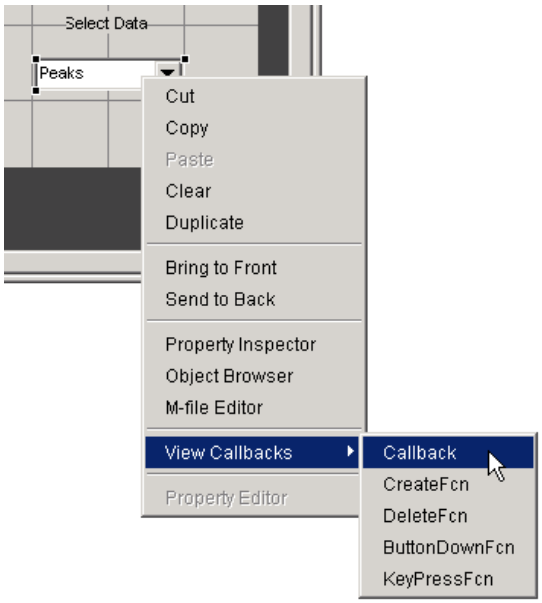
5.5 GUI Arayüzünün Programlanması

Bir GUI arayüzünün programlanması demek o çalışmanın kaydedildiği isimle aynı zamanla oluşturulan .m uzantılı dosya içerisine kodlama satırlarının eklenmesi demektir. Bu dosyanın içine görebilmek, değişiklik yapabilmek için GUIDE çalışma ekranı penceresinden View/M-File Editor komutu işletilebilir. Ardından karşımıza Şekil 5.13'deki gibi bir pencere gelecektir.



Şekil 5.13

Şekil 5.13'deki pencerede hazırlamış olduğumuz GUI tasarımına ait kodlar gözükmektedir. Burada pek çok kodun hazır eklenmiş olduğu görülecektir. Bu kodlar otomatik olarak MATLAB GUIDE tarafından eklenmiştir. Biz burada ilgili butonlara ve liste kutularına ya da istenilen bir nesneye ait callback isimli alt program parçalarına ilgili kodları yazacağız. Bir nesneye ait callback in bulunduğu satıra gitmek için araç çubuğunda yer alan f simgeli butona tıklanır ve açılan listeden ilgili nesneye ait callback in ismi seçilir. Bu durum yukarıdaki pencerede de görülmektedir. Ayrıca, GUIDE çalışma ekranından da direk istenilen bir callback satırına gidilebilir. Bunun için ilgili nesne üzerinde sağ tıklanır ve açılan pencereden View Callbacks menüsünden ilgili callback tıklanması ya da ilgili nesne seçilip View/View Callbacks yolu üzerinden açılan listeden gidilmek istenilen callback tıklanması yeterlidir.



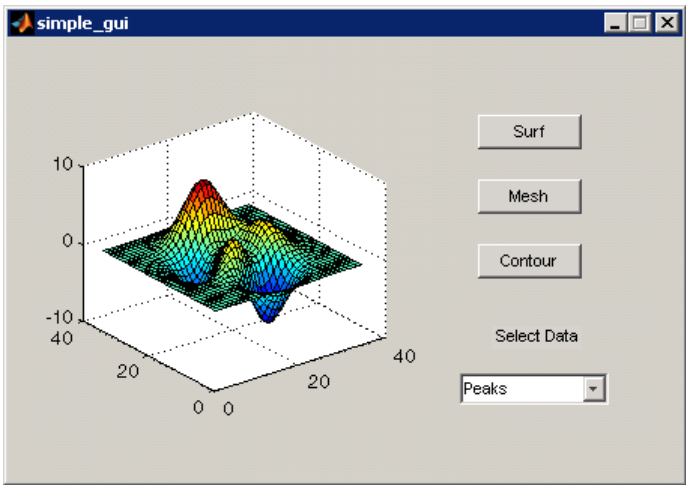
Şekil 5.14

Şimdi GUI arayüzünde yer alan tüm nesneler için View/M-File Editor yolundan kodlama penceresi açılıp, aşağıda yer alan tüm kodlar yazılsın.

```
function varargout = untitled_ilk(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled_ilk_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled_ilk_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

function untitled_ilk_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
handles.sinc = sinc;
handles.current_data = handles.peaks;
guidata(hObject, handles);
surf(handles.current_data)

function varargout = untitled_ilk_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
surf(handles.current_data);

function pushbutton2_Callback(hObject, eventdata, handles)
mesh(handles.current_data);

function pushbutton3_Callback(hObject, eventdata, handles)
contour(handles.current_data);

function popupmenu1_Callback(hObject, eventdata, handles)
```

```
str = get(hObject, 'String');
val = get(hObject,'Value');
switch str{val};
case 'Peaks' % User selects peaks.
handles.current_data = handles.peaks;
case 'Membrane' % User selects membrane.
handles.current_data = handles.membrane;
case 'Sinc' % User selects sinc.
handles.current_data = handles.sinc;
end
guidata(hObject,handles)

function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

Burada teker teker nesnelerin üzerinde sağ tıklayıp View Callback ve ilgili Callback satırına gidilip ayrı ayrı da yazılabilirdi. Burada göstermek amaçlı olduğu için kodlar bu şekilde direk verilmiştir. Kodlama satırlarında % işareti ile başlayan satırlar açıklama satırları olup, bu satırlar herhangi bir komut olarak görülmezler sadece açıklama amacı taşırlar. Tüm işlemler tamamladığına göre Tools/Run komutu ile GUI uygulamamızı çalıştırdığımızda Şekil 5.15'teki gibi bir ekran ile karşılaşılır.



Şekil 5.15

Burada yazılan kod parçalarını (callback rutinlerini) kısaca açıklayalım.

```
function varargout = untitled_ilk(varargin)
```

Yukarıdaki function bloğu GUIDE tarafından otomatik olarak oluşturulur. Burada GUI uygulamasına komut satırından gönderilen parametrelerin alınması ve GUI uygulaması çalıştıktan sonra bir fonksiyon olarak dışarıya gönderilecek parametrelerin tanımlanması ile ilgili kod satırları mevcuttur.

function untitled_ilk_OpeningFcn(hObject, eventdata, handles, varargin)

Bu fonsiyon GUI arayüzü ekrana gelmeden (visible olmadan) hemen önce çalıştırılacak kodları içerir. Örneğin böyle bir callback bir GUI uygulaması çalışmadan önce initialization işlemlerinin yapılması ya da bazı GUI nesne özelliklerinin değiştirilmesi istendiğinde kullanılabilir. Ayrıca, varargin giriş parametresi kullanılarak da MATLAB komut satırından girilen parametre değerleri GUI uygulaması içinde kullanılmak üzere bu blokta alınır.

function varargout = untitled_ilk_OutputFcn(hObject, eventdata, handles)

Bu fonksiyon bloğu bir GUI uygulaması hafızadan silinip programı sonlandırılmadan hemen önce (destroy edilmeden önce) çalıştırılacak komutlar içeriri. Ayrıca, komut satırına gönderilecek çıkış parametre değerleri de bu blok tarafından varargout değişkeni kullanılarak işleme konulur.

function pushbutton1_Callback(hObject, eventdata, handles)

Bu callback bloğu pushbutton1 isimli buton (istenilse bu buton ismi butonun Tag özelliğine özellikler penceresinden yeni bir isim verilerek de değiştirilebilir.) ki burada Surf stringine sahip olan buton tıklandığı zaman çalıştırılacak komutları içerir.

function pushbutton2_Callback(hObject, eventdata, handles)

Bu callback bloğu da benzer şekilde pushbutton2 isimli buton ki burada Mesh stringine sahip olan buton tıklandığı zaman çalıştırılacak komutları içerir.

function pushbutton3_Callback(hObject, eventdata, handles)

Bu callback bloğu da benzer şekilde pushbutton3 isimli buton ki burada Contour stringine sahip olan buton tıklandığı zaman çalıştırılacak komutları içerir.

function popupmenu1_Callback(hObject, eventdata, handles)

GUI arayüzüe eklenmiş olan popup_menu nesnesininden herhangi bir eleman tıklanıp seçildiği zaman çalışması istenilen kod parçaları bu callback altında yazılır.

function popupmenu1_CreateFcn(hObject, eventdata, handles)

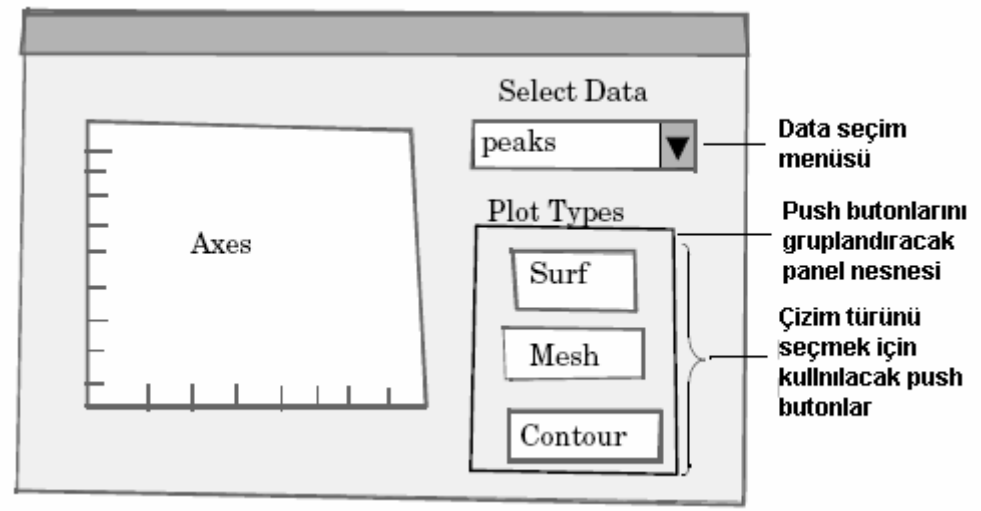
Bu callback GUIDE tarafından otomatik olarak oluşturulmuş olup, popup_menu nesnesi uygulama ekranına gelmeden (visible olmadan) ve de oluşturulmadan önce koşturulacak program satırlarını içerir.

5.6 M-File Programlama Yöntemi Kullanılarak GUI Tasarımı Oluşturma

Burada GUIDE gibi bir tasarım aracı kullanılmaz. Sadece kod satırları yazılarak hem GUI arayüzü hem de bu arayüzün koşturduğu komut satırları aynı dosya içerisinde yazılır. Bu dosyalar .m uzantısına sahiptirler.

Bir GUI arayüzünü bu yöntemle oluşturabilmek için öncelikle tasarım öncesi arayüzün bir planı taslak halinde bir kâğıt üzerine çizilmelidir. Çünkü burada tüm işlemlerin yapılması

muazzam bir çalışma ve ölçümlendirme ile belirlenen nesnelerin uygun yerlere kullanışlı bir GUI arayüzü çıkarmak üzere bir araya gelmesi tamamı ile GUI tasarım ve programcısının yazdığı kodlar ile gerçekleştirilecektir.



Şekil 5.16

Yukarıdaki pencerede görüldüğü üzere bir önceki sayfalarda anlatılan örnek GUI tasarımının taslak görüntüsü görülmektedir. Bu şekilde nesnelerin yerleri tespit edildikten sonra bir GUI uygulaması oluşturulmak üzere programlama yöntemi ile tasarıma geçilebilir.

Şimdi MATLAB komut satırından "edit" komutunu verelim. Karşımıza boş bir m file dosya gelecektir. Genel olarak programlama yolu ile tasarlanılacak GUI uygulaması komut satırları aşağıda belirtilen yapıda olmalıdır. Burada örneğin GUI uygulamamızın adı MYGUI olsun.

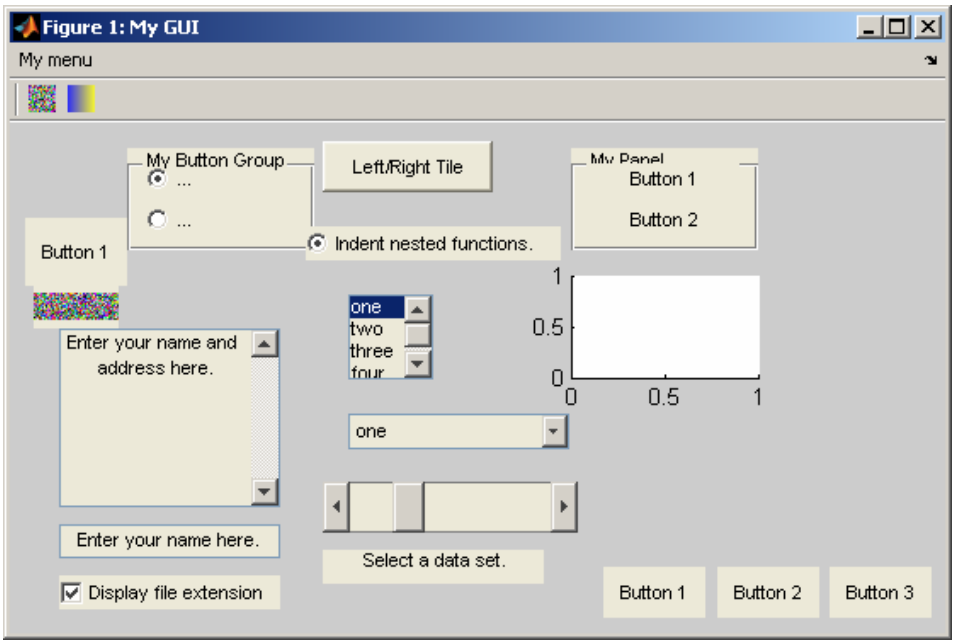
```
function varargout = mygui(varargin) MYGUI uygulaması için mygui.m dosyasi ilk satırı

% MYGUI Kısa bir GUI uyugulaması ile ilgili açıklayıcı bilgi

% Bir adet boş açıklama satırına kadar bu satir ve sonra gelen
% satırlar MATLAB komut satırında GUI uygulamasını
% açıklayıcı ve help komutu ile kullanıcıya sunulan
% yardım satırlarını içerir.

% Burada help satırlarını kod satırlarından ayırmak için bir adet boş açıklama satırı konulur.

% GUI uygulamasının giriş parametre alınması ve ilk önhazırlık işlemleri bloğu

% GUI nesnelerinin oluşturulması ile ilgili satırlar bloğu

% Callback ler öncesi önhazırlık işlemleri bloğu

% MYGUI için gerekli callback fonksiyonları

% MYGUI için kullanılacak fonksiyonlar bloğu
```

end Bu komut fonksiyon bloğunun sonunu belirtmek için konulmuştur.

Yukarıdaki yapıyı oluşturacak şekilde komutlar MYGUI isimli GUI uygulaması için mygui.m isimli dosyaya kaydedilir. Bu GUI uygulamasını çalıştırmak için de MATLAB komut satırından sadece "mygui" komutunun verilmesi yeterlidir. Bu şekilde uygulama penceresi karşımıza gelecektir. Şu aşamada herhangi bir kod yazılmadığı herhangi bir şey olmayacaktır. Ancak, yazılmış olsaydı ilgili GUI penceresi görülecekti.

5.6.1 Programlama Yoluyla Nesnelerin Eklenmesi

Yukarıda bahsedilen mygui.m dosyasının içeriğine aşağıda belirtilen kodları eklediğimizde GUI arayüzümüz şu şekilde görülecektir.



Şekil 5.17

```
function varargout = mygui(varargin)                    mygui icin fonksiyon tanımı

fh = figure('Visible','on','Name','My GUI',...          bu satırlar ekrana belirtilen boyut
'Position',[360,550,550,300]);                          ve konumda figure (GUI yüzeyi) getirme

cbh = uicontrol(fh,'Style','checkbox',...               GUI yüzeyine checkbox nesnesi
'String','Display file extension',...                   ekleme
'Value',1,'Position',[30 15 130 20]);

eth = uicontrol(fh,'Style','edit',...                   GUI yüzeyine edit kutusu ekleme
'String','Enter your name here.',...
'Position',[30 45 130 20]);

eth = uicontrol(fh,'Style','edit',...                   GUI yüzeyine çok satırlı edit kutusu
'String','Enter your name and address here.',...        ekleme (çünkü max-min>1 durumu)
'Max',2,'Min',0,...
'Position',[30 75 130 105]);
```

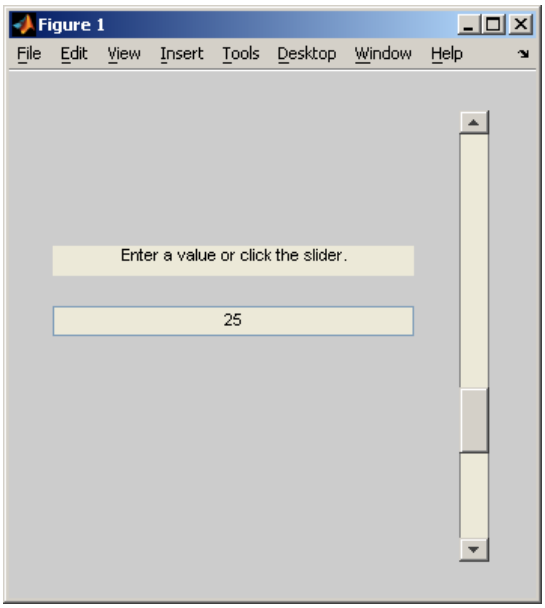
```
lbh = uicontrol(fh,'Style','listbox',...                GUI yüzeyine liste kutusu ekleme
'String',{'one','two','three','four'},...               (elemanlari 'one','two','three','four' )
'Value',1,'Position',[200 150 50 50]);

pmh = uicontrol(fh,'Style','popupmenu',...              GUI yüzeyine popup menü ekleme
'String',{'one','two','three','four'},...               (elemanlari 'one','two','three','four' )
'Value',1,'Position',[200 110 130 20]);

pbh1 = uicontrol(fh,'Style','pushbutton','String','Button 1',...    GUI yüzeyine push buton
'Position',[10 205 60 40]);                                         ekleme

img(:,:,1) = rand(16,64);                               rasgele sayılardan oluşan dizi tanımı
img(:,:,2) = rand(16,64);
img(:,:,3) = rand(16,64);

pbh2 = uicontrol(fh,'Style','pushbutton',...            GUI yüzeyine push buton ekleme
'Position',[15 180 50 25],...
'CData',img);

rbh = uicontrol(fh,'Style','radiobutton',...            GUI yüzeyine radio buton ekleme
'String','Indent nested functions.',...
'Value',1,'Position',[175 220 150 20]);

sh = uicontrol(fh,'Style','slider',...                  GUI yüzeyine kaydırıcı ekleme
'Max',100,'Min',0,'Value',25,...
'SliderStep',[0.05 0.2],...
'Position',[185 60 150 30]);

sth = uicontrol(fh,'Style','text',...                   GUI yüzeyine static text kutusu
'String','Select a data set.',...                       ekleme
'Position',[185 30 130 20]);

tbh = uicontrol(fh,'Style','togglebutton',...           GUI yüzeyine toggle (çift durumlu)
'String','Left/Right Tile',...                          buton ekleme
'Value',0,'Position',[185 260 100 30]);

ph = uipanel('Parent',fh,'Title','My Panel',...         GUI yüzeyine panel ekleme
'Position',[.60 .75 .2 .2]);

pbh3 = uicontrol(ph,'Style','pushbutton','String','Button 1',...    ph paneline push buton
'Units','normalized',...                                            ekleme
'Position',[.1 .55 .8 .3]);
pbh4 = uicontrol(ph,'Style','pushbutton','String','Button 2',...    ph paneline 2. push buton
'Units','normalized',...                                            ekleme
'Position',[.1 .15 .8 .3]);

bgh = uibuttongroup('Parent',fh,'Title','My Button Group',...       GUI yüzeyine buton grubu
'Position',[.125 .75 .2 .2]);                                       ekleme

rbh1 = uicontrol(bgh,'Style','radiobutton','String','Red',...       bgh grubuna radio buton
```

```
'Units','normalized',...                                    ekleme
'Position',[.1 .6 .3 .2]);
rbh2 = uicontrol(bgh,'Style','radiobutton','String','Blue',...   bgh grubuna 2. radio buton
'Units','normalized',...                                    ekleme
'Position',[.1 .2 .3 .2]);

ah = axes('Parent',fh,'Position',[.60 .50 .2 .2]);          GUI yüzeyine grafik çizim
                                                            alanı ekleme

b1 = uicontrol(fh,'Posit',[330 80 60 30],'String','Button 1');   GUI yüzeyine buton 1 koy
b2 = uicontrol(fh,'Posit',[350 50 60 30],'String','Button 2');   GUI yüzeyine buton 2 koy
b3 = uicontrol(fh,'Posit',[310 10 60 30],'String','Button 3');   GUI yüzeyine buton 3 koy

align([b1 b2 b3],'Right','None');                           b1, b2 ve b3 nesnelerini sağa hizala
%align([b1 b2 b3],'Left','Distribute');                     b1, b2 ve b3 nesnelerini sola dağınık hizal.
align([b1 b2 b3],'Center','Fixed',7);                       b1, b2 ve b3 nesnelerini ortala
align([b1 b2 b3],'Fixed',5,'Bottom');                       b1, b2 ve b3 nesnelerini aşağıyı doğru hiz.

set(fh,'MenuBar','figure');                                 standart araç çubuğunun gösterilmesi
set(fh,'MenuBar','none');                                   standart araç çubuğunun gizlenmesi

mh = uimenu(fh,'Label','My menu');                          GUI yüzeyi menüsüne My menu eklenm.
eh1 = uimenu(mh,'Label','Item 1');                          mh menüsüne alt menü tanımlanması
eh2 = uimenu(mh,'Label','Item 2','Checked','on');           mh menüsüne alt menü tanımlanması

set(eh2,'Separator','on');                                  mh2 menu seceneğinin üzerine ayıraç kon.

seh1 = uimenu(eh1,'Label','Choice 1','Accelerator','C',...  eh1'e kisayol (Ctrl+C) tanımı ve
'Enable','off');                                            pasif yapılması
seh2 = uimenu(eh1,'Label','Choice 2','Accelerator','H');    eh2'ye kisayol (Ctrl+H) tanımı

th = uitoolbar(fh);                                         GUI yüzeyine araç çubuğu ekleme

a = [.20:.05:0.95];                                         bu ve alt satırlarla rasgele renkleri
img1(:,:,1) = repmat(a,16,1);                               temsil etmek üzere dizilerin tanımlanması
img1(:,:,2) = repmat(a,16,1);
img1(:,:,3) = repmat(flipdim(a,2),16,1);

pth = uipushtool(th,'CData',img1,...                        th araç çubuğuna push buton ekleme
'TooltipString','My push tool',...
'HandleVisibility','off');                                  handlevisibility komut satırından erişimi ayarlar

img2 = rand(16,16,3);                                       rastgele renk dizisi tanımlama
tth = uitoggletool(th,'CData',img2,'Separator','on',...     araç çubuğuna ayırıcı ekleme
'TooltipString','Your toggle tool',...
'HandleVisibility','off');

oldOrder = allchild(th);                                    bu ve aşağıdaki satırlar ile nesnelerin tab tuşu
newOrder = flipud(oldOrder);                                ile geçiş sırası ayarlanmakta
set(th,'Children',newOrder);
```

delete(tth);	tth handleini tutan nesnenin (burada araç çubuğu) silinmesi
end	fonksiyon sonu

5.6.2 Programlama Yöntemi ile GUI Tasarımında Callback Kullanımı

Şekil 5.18'de görülen penceredeki GUI arayüzü tasarlanmış olsun. Burada amaçlanan kaydırıcının her hareketini text kutusunda göstermek, aynı zamanda text kutusuna 0-100 aralığında değerler girilip enter tuşuna basınca kaydırıcının değerini de o değere set etmektir. Şayet kullanıcı bu aralığın dışında ya da hatalı giriş yaparsa hata sayısını text kutusuna yazdırmaktır.



Şekil 5.18

Böyle bir GUI tasarımı için aşağıdaki kodlar (komut satırından edit komutu ile Editor penceresini açarak) slider_gui.m isimli dosyaya yazılsın ve kaydedilsin.

```
function slider_gui

fh = figure('Position',[250 250 350 350]);

sh = uicontrol(fh,'Style','slider',...
'Max',100,'Min',0,'Value',25,...
'SliderStep',[0.05 0.2],...
'Position',[300 25 20 300],...
'Callback',@slider_callback);

eth = uicontrol(fh,'Style','edit',...
'String',num2str(get(sh,'Value')),...
'Position',[30 175 240 20],...
'Callback',@edittext_callback);

sth = uicontrol(fh,'Style','text',...
```

```
'String',Bir değer girin veya kaydırıcıyı kullanın.',...
'Position',[30 215 240 20]);
number_errors = 0;

function slider_callback(hObject,eventdata)
set(eth,'String',...
num2str(get(hObject,'Value')));
end

function edittext_callback(hObject,eventdata)
val = str2double(get(hObject,'String'));
% text kutusuna girilen değer kaydırıcının min ve max
% değerleri arasında ise kaydırıcının yeni değerini set et.
if isnumeric(val) && length(val) == 1 && ...
val >= get(sh,'Min') && ...
val <= get(sh,'Max')
set(sh,'Value',val);
else
% Hatalı giriş söz konusu ise hata sayacını bir arttır
number_errors = number_errors+1;
set(hObject,'String',...
['Toplam hatalı giriş sayısı = ',...
num2str(number_errors)]);
end

end
```

Buradaki GUI öğreğimizde toplam iki adet callback fonksiyonu kullanılmıştır. “slider_gui” isimli fonksiyon bizim GUI ana uygulamamız için tanımlanmıştır. Burada “slider_callback(hObject,eventdata)” ve “edittext_callback(hObject,eventdata)” isimli callbackler “uicontrol” komutu ile komut satıralrının en başında nesneler GUI yüzeyine eklenirken “Callback” özelliklerine parametre olarak aktarılmıştır. Dolayısıyla hangi nesne ile ilgili bir olay (event) oluşmuş ise ona ait ve o olayla ilgili callcak ler GUI tarafından çalıştırılacaktır. Burada

```
function slider_callback(hObject,eventdata)
```

isimli callback fonksiyonu kaydırıcı nesnemizin değeri değiştirilince icra edilecek komut satırlarını içerir.

```
function edittext_callback(hObject,eventdata)
```

isimli callback fonksiyonu ise texy kutusu içerisine herhangi bir değer girilince (bu sayı da olabilir veya string tipi değerler de olabilir.) ve enter tuşuna basılınca koşturulacak olan komutları içermektedir.

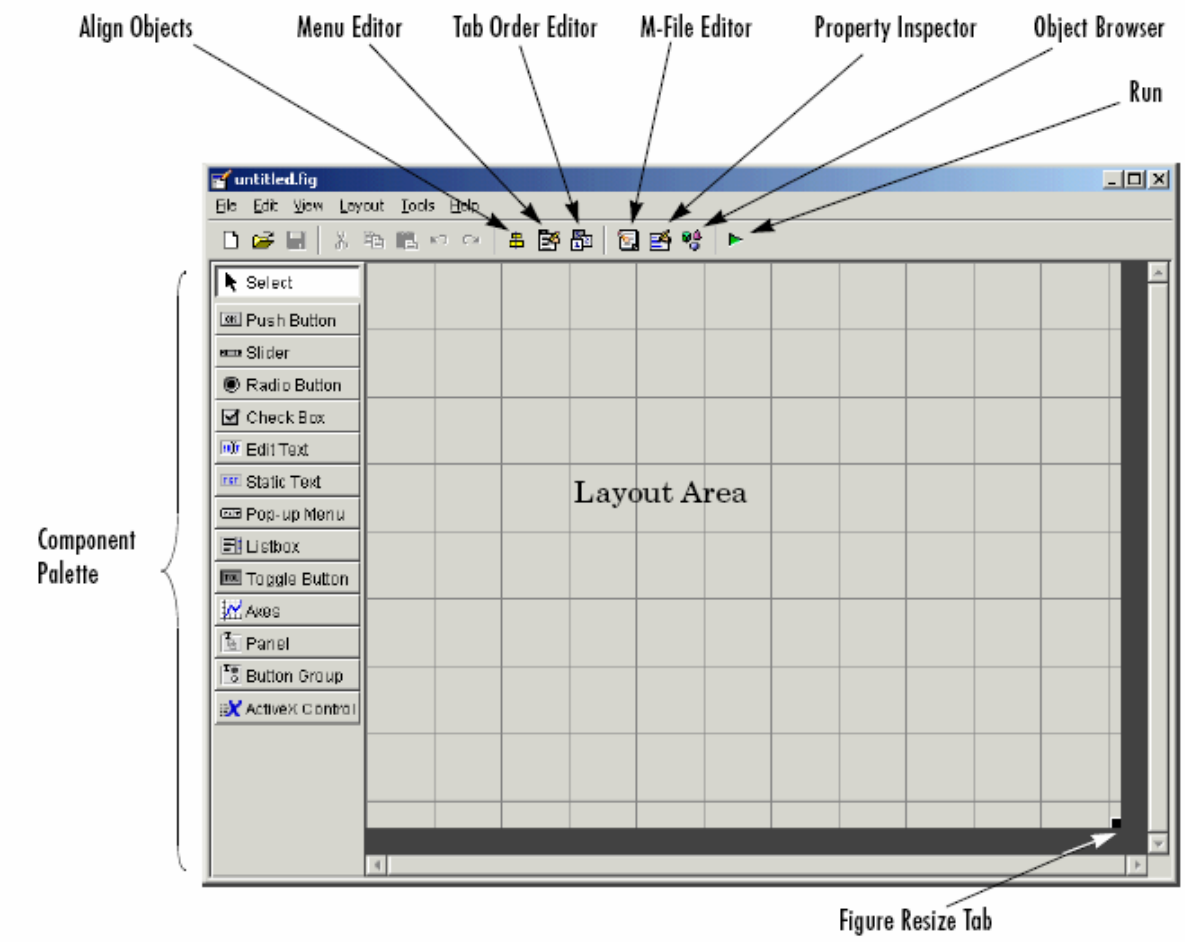
Bu programda özetle text kutusuna girilen değer

```
val = str2double(get(hObject,'String'));
```

komutu ile önce double tipinde sayısal değere çevrilmektedir. Ardından gelen if sorgusu ile girilen değer sayısal ve herhangi bir hata yoksa kaydırıcının yeni değeri set edilmektedir. Fakat girilen değer hatalı ise if sorgusunda bu durum öğrenilmekte ve de else ifadesinden sonra gelen komutlar icra edilmekte olup, burada da "number_errors" isimli genel bir değişken oluşan toplam hata sayısı için sayaç görevi görmekte ve hata durumunda değeri bir arttırılmaktadır. Daha sonra da oluşan hata durumu text kutusunun "string" özelliğinden faydalanılarak kullnıcıya gösterilmektedir.

5.7 GUIDE Aracının İncelenmesi

Bir önceki konularımızda da bu aracı kısaca incelemye çalıştık. Burada GUIDE aracı detaylı olarak incelenecektir. MATLAB komut satırından "guide" komutunu yazdığımızda ve gelen pencereden boş (blank) bir GUI tasarımını seçtiğimizde Şekil 5.19'daki pencere ile karşılaşılır.



Şekil 5.19

Bu ekrandaki araçlar ile ilgili açıklama aşağıda verilmiştir.

5.7.1 Layout Editor

GUIDE çalışma alanı ve penceresidir. Bu ekran ile GUI yüzeyine component paletten seçilen ilgili nesneler eklenebilir ya da diğer araçlar ile program kodlarının yazılması,

nesnelerin GUI yüzeyi üzerinde hizalanması, tab tuşu geçiş sırasının değiştiirlmesi gibi pek çok işlem gerçekleştirilebilir.

5.7.2 Figure Resize Tab

Bu araç GUI çalışma alanının boyutlandırılmasını sağlar. Fare işaretçisi bu alan üzerine getirildiğinde konum değiştirecektir. Bu anda farenin sol tuşu tıklanıp ileri geri hareket ettirilerek GUI yüzey alanının boyutları değiştirilebilir.

5.7.3 Menu Editor

GUI uygulamasına istenilirse File, Edit… v.b. gibi menü içeren programlarda olduğu gibi bir manü eklenmesi ve eklenen menü ile ilgili işlmelerin yapılması bu araç vasıtasıyla sağlanır.

5.7.4 Align Objects

Bu araç sayesinde GUI çalışma alanına eklenilen nesnelerin yatay ya da dikey olarak hizalanması işlemleri gerçekleştirilir.

5.7.5 Tab Order Editor

Tab Order Editor kullanılarak GUI yüzeyindeki nesnelerin birinden diğerine tab tuşu ile geçiş sırası (örneğin bir buton seçili ve aktif iken bir başka butona ya da bir liste kutusuna tab tuşu kullanılarak geçilmesi gibi.) değiştirilebilir.

5.7.6 Property Inspector

Bu pencere sayesinde de GUI uygulamasına eklenen nesnelerin özellikleri değiştirilebilir ya da var olan özelliklerinin ve değerlerinin neler olduğu gözlenilebilir.

5.7.7 Object Browser

Bu araç ile tasarımcı GUI uygulamasına eklemiş olduğu nesnelerin ve isimlerinin neler olduğu genel hali ile bakabilir.

5.7.8 Run

Bu buton yardımı ile de hazırlanmış olan bir GUI uygulaması çalıştırılabilir. Bu şekilde tasarımcı hazırlamış olduğu GUI'yi test etme imkânına sahiptir.

5.7.9 M-File Editor

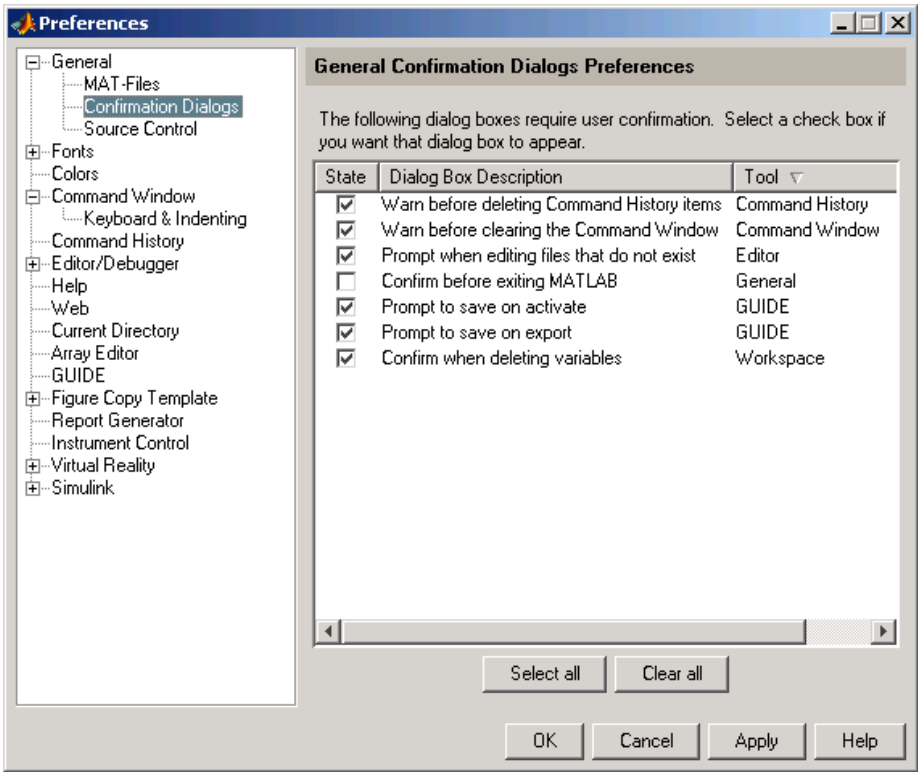
Hazırlanmış olan GUI uygulaması ile ilgili komutları görebilmek ve üzerinde değişiklik yapabilmek için bu araç kullanılır.

5.7.10 GUIDE Tercihleri

Bu tercihleri görebilmek için GUIDE ekranında File menüsünden Preferences komutu çalıştırılır.

5.7.10.1 Doğrulama Seçenekleri

Bu seçeneklere ulaşmak için General/Confirmatin yolu izlenmelidir. Karşımıza Şekil 5.20'deki gibi bir pencere gelecektir.

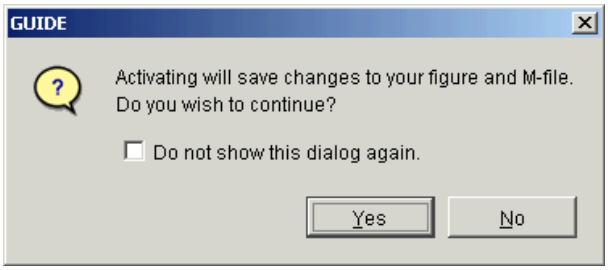


Şekil 5.20

Bu penceredeki seçeneklerden iki tanesi GUIDE ile ilgilidir. Bu seçeneklerin görevleri şu şekildedir:

- **Promp to Save on Activate**

Bu seçenek seçili ise GUIDE bir GUI uygulaması çalıştırılmadan önce onun kaydedilmesi gerektiğini bildiren Şekil 5.21'deki gibi bir pencere ile kullanıcı uyarılır.

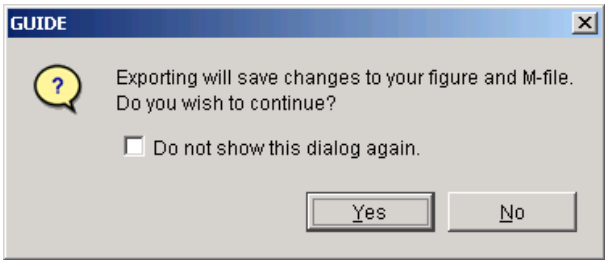


Şekil 5.21

Bu gelen ekranda evet denilerek uygulamanın kaydedişlmesi ve çalıştırılması sağlanılabilir. Eğer ki bu pencerenin her seferinde üzerinde değişiklik yapılan, fakat kaydedilmeyen bir GUI uygulaması olduğunda kullanıcıyı uyarmaması isteniyor ise "Do not show this dialog again." seçeneği işaretlenmelidir.

- **Promp to Save on Export**

Bu seçenek GUIDE çalışma ekranında iken File menüsünden Export komutu verilirse ve tasarlanan GUI uygulaması kaydedilmemiş değişiklikler içeriyor ise kullanıcıya Export işlemi öncesinde var olan değişikliklerin kaydedileceği konusunda Şekil 5.22'deki pencere ile uyarır.

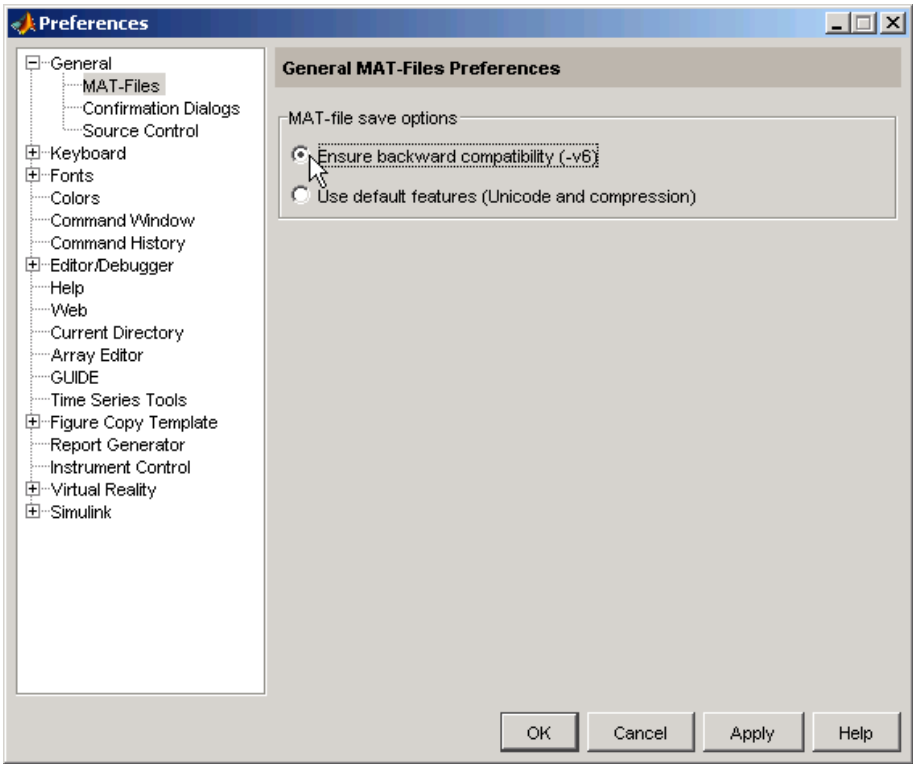


Şekil 5.22

Bu pencerede evet butonuna tıklanılarak işleme devam edilebilir. Eğer ki bu pencerenin sürekli çıkması istenmiyor ise kullanıcı evet demeden önce "Do not show this dialog again." seçeneğini işaretlemelidir.

5.7.10.2 Geriye Uyumluluk Seçeneği

Seçenekler penceresinde General>Mat-Files yolu izlenilerek gelen Şekil 5.23'teki pencereden önceki MATLAB versiyonları ile uyumlu olacak şekilde dosyaların kaydedilme format ayarı değiştirilebilir.



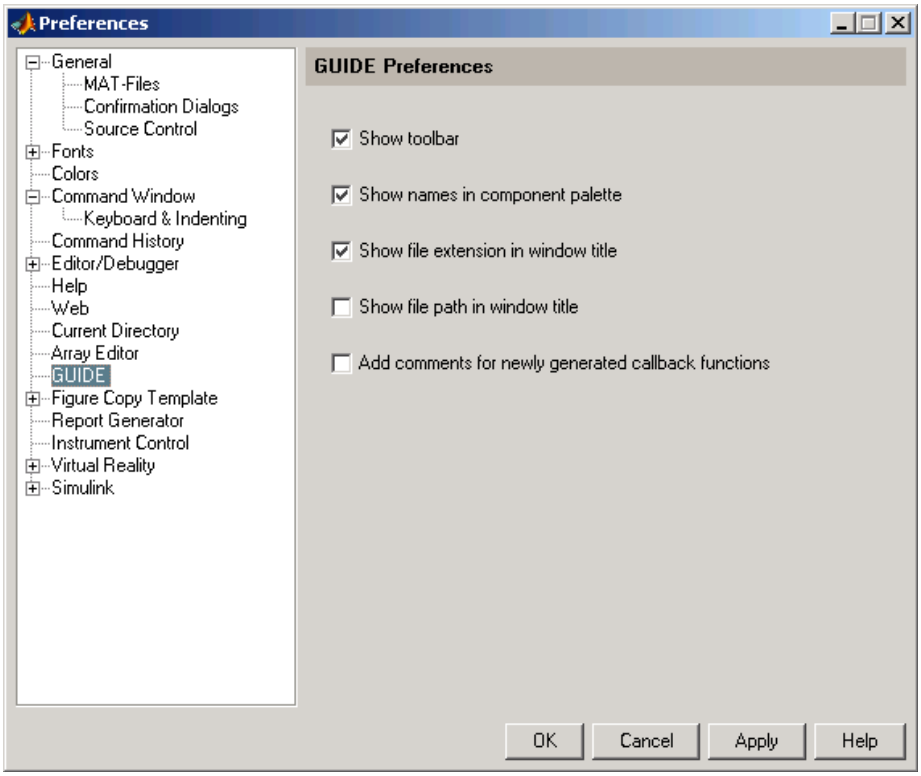
Şekil 5.23

Bu çalışmada kullanılan MATLAB versiyonu 7.0.4'tür. Bu pencere yardımıyla "Ensure backward compatibility (-v6)" seçeneği seçilirse GUIDE kullanılarak hazırlanan GUI

uygulamaları versiyon 6 nolu MATLAB GUIDE uygulamaları ile uyumlu olacaktır. Bunun anlamı hazırlanan her GUI uygulamasının iki adet dosya halinde kaydedilmesi demektir. Bu dosya türleri .m ve .fig uzantılarına sahiptir. Uyumluğu korumak için .m dosyaları GUI tasarımlarının komutlarını içermek üzere kaydedilir. Ayrıca, yine uyumluluk sağlamak için .fig uzantılı dosya formatı kullanılarak hazırlanan dosyada da GUI arayüzünün görsel ayarları ve nesnelerin görünümü ile ilgili bilgileri saklanır.

5.7.10.3 Diğer Tercihler

Preferences ekranında ayrıca GUIDE yolu altında diğer tüm GUIDE seçenekleri yer almaktadır. Karşımıza Şekil 5.24'teki gibi bir pencere gelecektir.

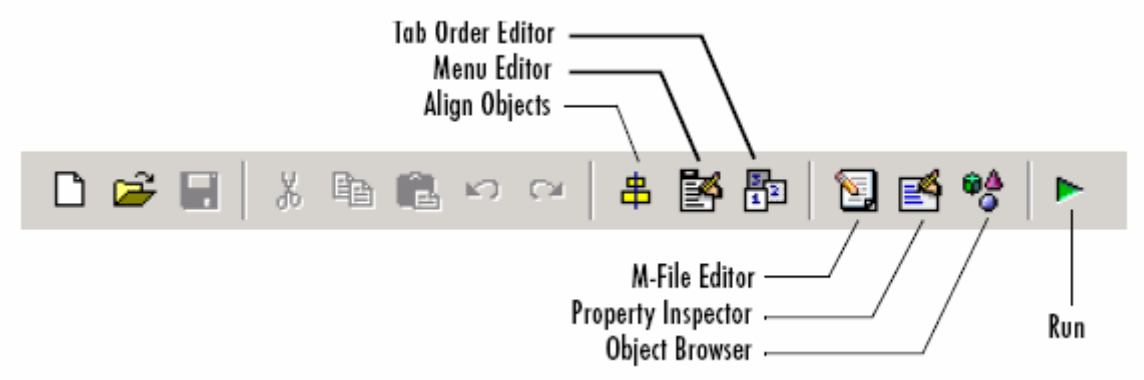


Şekil 5.24

Bu penceredeki seçeneklerin işlevleri şu şekildedir:

5.7.10.3.1 Show Toolbar:

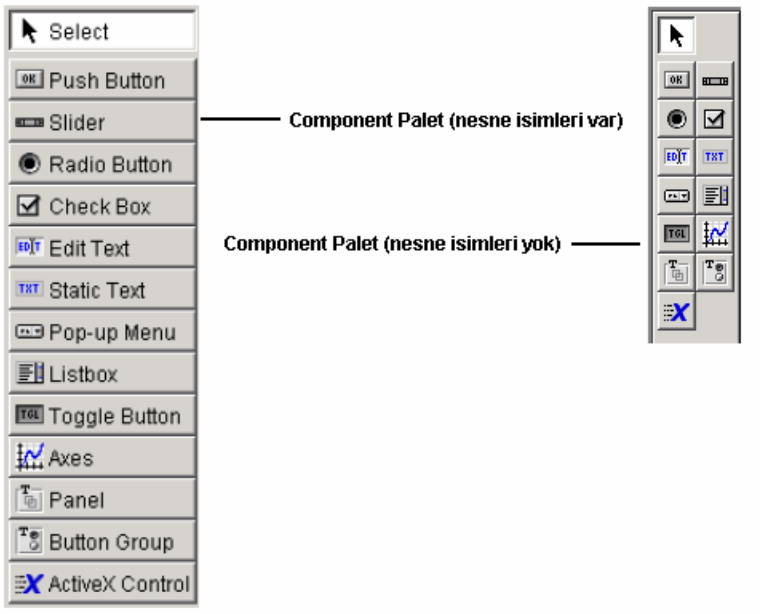
GUIDE ekranında Şekil 5.25'te görülen araç çubuğunu göstermek için bu seçenek seçili işretli olmalıdır.



Şekil 5.25

5.7.10.3.2 Show Names in Component Palette

GUIDE ekranında aşağıda da görüldüğü gibi component paletinde yer alan butonlarda nesnelerin isimlerini göstermek için bu seçenek işaretlenmelidir.



Şekil 5.26

5.7.10.3.3 Show File Extension in Window Title

GUIDE ekranının başlık çubuğunda üzerinde çalışılan GUI uygulamasının dosya isminin yanında .fig uzatısının da gösterilmesi istenirse bu seçenek işaretlenmelidir.

5.7.10.3.4 Show File Path in Window Title

GUIDE ekranının başlık çubuğunda üzerinde çalışılan GUI uygulamasının tüm dosya yolu ile birlikte dosya isminin gösterilmesi istenirse bu seçenek işaretlenmelidir.

5.7.10.3.5 Add Comments for Newly Generated Callback Functions

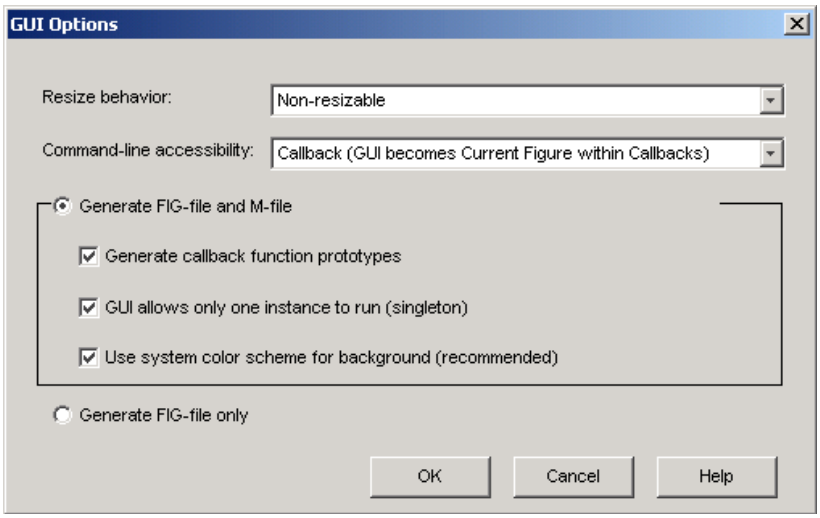
Boş bir GUI uygulaması (untitled bir döküman) ile GUI tasarıma başlandığında .m uzantılı komut satırlarının olduğu dosyaya her bir callback ve bu dosyada yer alan fonksiyonlarla ilgili

otomatik olarak açıklama satırlarının eklenmesi istenirse bu seçenek işaretli olmalıdır. GUI kodlamasında açıklama satırlarının başında % işareti yer alır ve varsayılan olarak M-File Editor'de açıklama satırları yeşil renkte gözükür. Örnek açıklama satırları aşağıda gözükmektedir.

```
% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject handle to figure1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

5.7.11 GUIDE Seçenekleri

Bu seçeneklere GUIDE çalışma penceresinde iken Tools menüsünden GUI Options komutu ile erişilebilir. Karşımıza Şekil 5.27'deki gibi bir pencere gelecektir.



Şekil 5.27

Bu penceredeki seçeneklerin görevleri şu şekildedir.

5.7.11.1 Resize Behavior

Bu seçenek üç farklı durum içerir.

- **Non-resizable**

Kullanıcılar GUI uygulaması penceresinin boyutunu değiştiremezler.

- **Proportional**

Bu seçenek seçildiğinde hem kullanıcı hem tasarımcı GUI yüzeyi ve tüm nesneleri birbiri ile orantılı olarak büyütüp küçültebilirler.

- **Other (Use ResizeFcn)**

Kullanıcılar GUI pencere boyutlarını değiştirseler bile nesnelerin boyutları aynı kalır. İstenirse tasarımcı ResizeFcn callbackini kullanarak pencere boyutlarının değiştirilmesi ile çalışan bu callback i programlayabilir.

5.7.11.2 Command-Line Accessibility

Bu seçenek ile MATLAB komut satırından kullanıcıların GUI figure penceresine erişimi ve kullanmaları engellebilir. Normalde MATLAB gcf komutu ile o an aktif olan figure gösterilebilir. Ancak, bu seçenek sadece GUI figure ekranının callbacklerden erişimine imkân verirse o takdirde aktif gure olarak komut satırından GUI uygulamasının figure ekranına erişilemeyecektir.

5.7.11.3 Generate FIG-File and M-File

Bu seçenek kullanılırsa GUI uygulaması iki dosya halinde kaydedilir ve de bu seçeneğin altındaki görevlere de erişim hakkı kazanmış olur. Alt seçeneklerin işlevleri şu şekildedir.

5.7.11.3.1 Generate Callback Function Prototypes

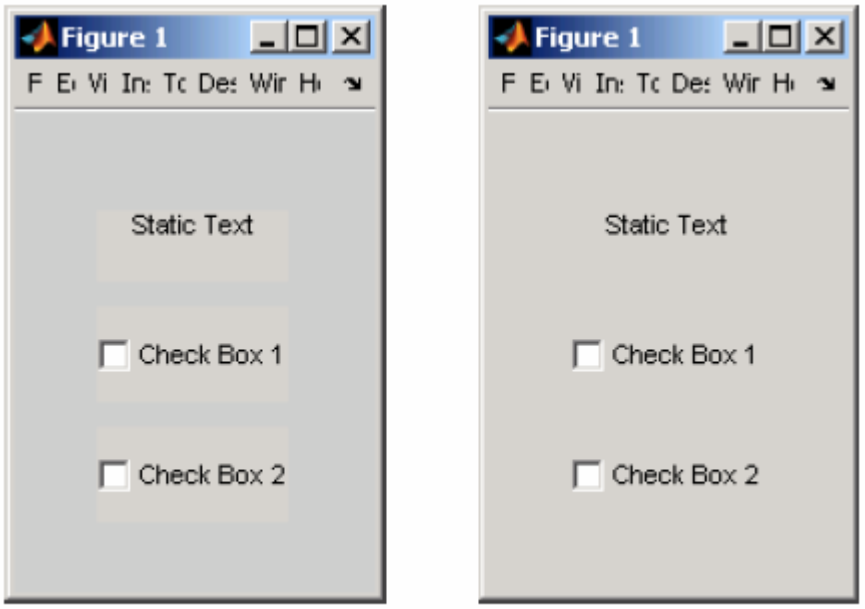
Bu seçenek ile GUI uygulaması ilk oluşturulduğunda (untitled döküman durumu) GUIDE .m uzantılı dosya içeriğine ilk bilinen şablon (template) callbacklere ait fonksiyon tanımlarını içeren komut satırlarını otomatik olarak ekler.

5.7.11.3.2 GUI Allows Only One Instance to Run (Singleton)

Bu seçenek seçilirse GUIDE ekranında tasarlanan GUI uygulaması her run edişinde sadece bir tane GUI penceresi üzerinde çalışacaktır. Ancak, bu seçenek işaretlenmezse her çalıştırmada birden fazla GUI uygulama ekranının çalışmasına izin verilecektir.

5.7.11.3.3 Use System Color Scheme for Background

Bu seçenek seçilirse GUI uygulaması yüzeyi alanının rengi sistemin genel form rengi ile uyumlu olacaktır. Ancak, bu seçenek seçilmezse aşağıda görüldüğü üzere GUI arayüzünde nesne renkleri ile uygulama pencere rengi arasında farklılıklar oluşacaktır.



Şekil 5.28

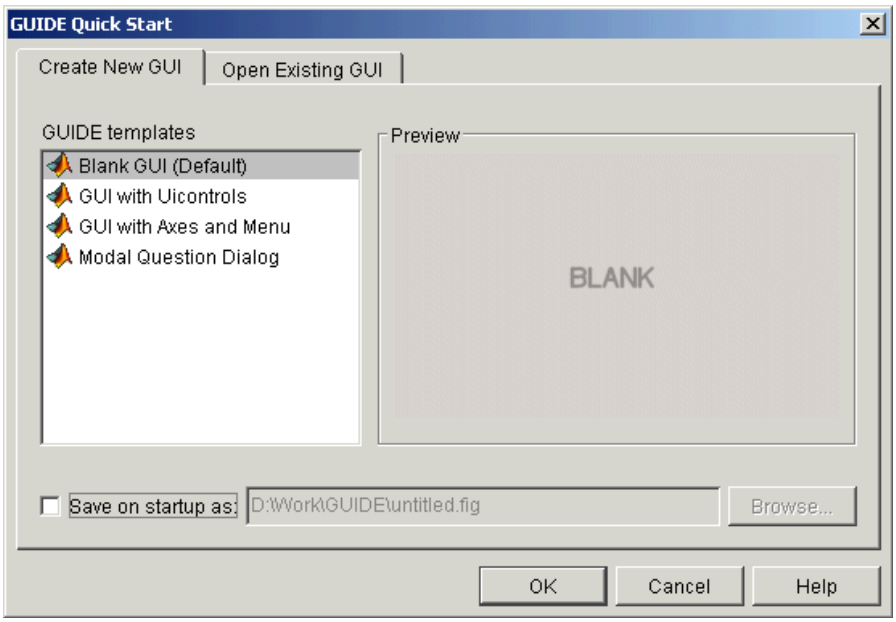
5.7.11.3.4 Generate FIG-File Only

Bu seçenek işaretlenirse GUI tasarımları sadece .fig uzantılı dosya içerisinde hem görünüm ayarlarının, hem de programlama komut satırlarının bulunması sağlanmış olur. Ancak, bu seçeneğin işaretlenmesi ile versiyon 6 GUIDE uygulamaları ile uyumluluk ortadan kalkacak ve de GUI tasarımında programcı işlemlerini yukarıda belirtilen seçenekler olmadan kısıtlı olarak sürdürecektir.

5.7.12 GUIDE Aracında Şablon (Template) Uygulamalar ile Çalışma

MATLAB GUIDE aracı GUI tasarımcılarına ilk başlayanlar için kendi içinde hazır örnek uygulamalar (templates) sunmaktadır. Bu uygulamaların arayüzü ve nesnelerin callbacklerini kullanan komut satırlarını yazma hususunda programcı ve tasarımcıya önbilgi vermesi bakımından çok yararlıdır.

MATLAB komut satırından guide yazıldığında ya daraç çubuğundan GUIDE simgesi tıklandığında karşımıza Şekil 5.29'daki gibi bir pencere gelecektir.



Şekil 5.29

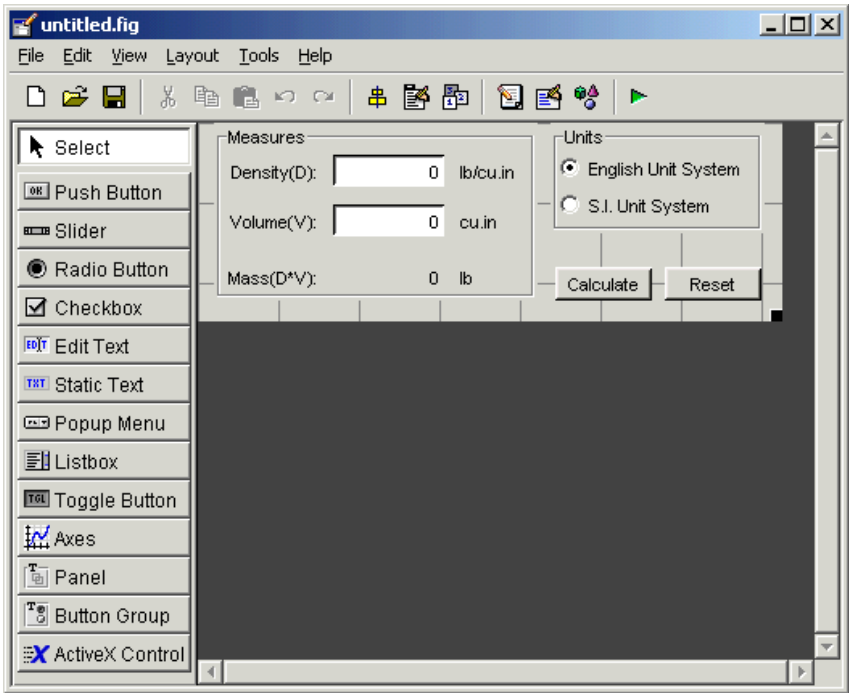
Bu pencereden template olarak karşımıza üç seçenek çıkmaktadır. Bunlar:

- **"GUI with Uicontrols"**
- **"GUI with Axes and Menu"**
- **"Modal Question Dialog"**

Aşağıda bu uygulamalr hakkında ayrıntılı bilgiler verilmiştir.

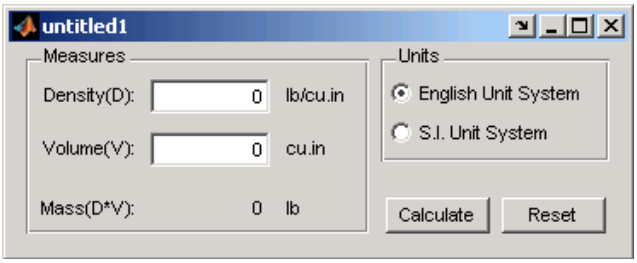
5.7.12.1 "GUI with Uicontrols" Uygulaması

Bu uygulama seçildiğinde Şekil 5.30'da gözüken GUI tasarımı gözükecektir.



Şekil 5.30

Bu uygulamayı araç çubuğundan Run butonuna basarak çalıştırdığımızda da Şekil 5.31'deki gibi bir uygulama arayüzü ekrana gelecektir.

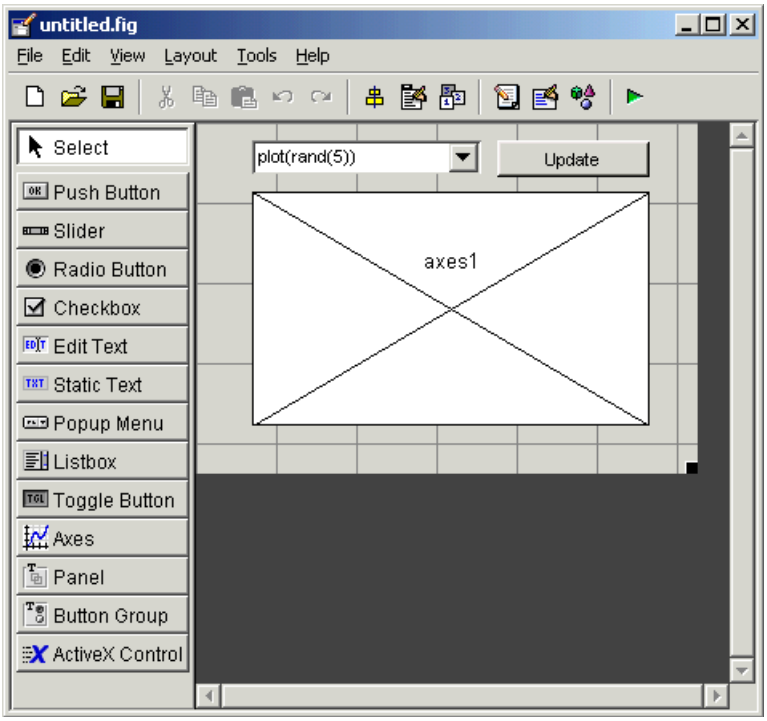


Şekil 5.31

Bu uygulamada amaç seçilen birim sistemine göre yoğunluk ve hacim değerleri girilen bir cismin kütle değerini Yoğunluk*Hacim (D*V) formülünden yola çıkarak hesaplanmasını sağlanmak ve kullanıcıya hesaplanan değeri sunmaktır. Burada push butonların ve text kutuların program kodları ve callback parçalarının kullanımına yönelik olarak programcıya bilgiler verilmektedir.

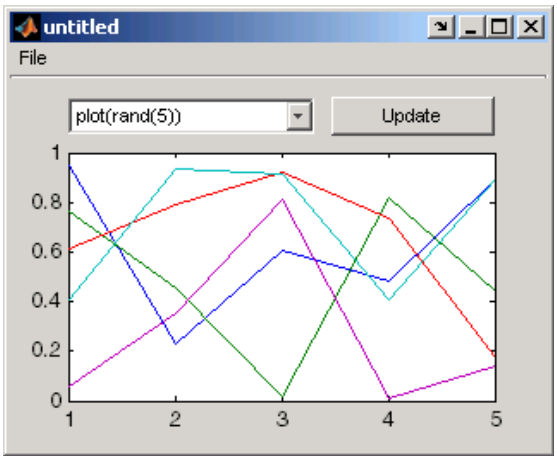
5.7.12.2 "GUI with Axes and Menu" Uygulaması

Bu seçenek seçilerek bir GUI template uygulaması açıldığında karşımıza Şekil 5.32'deki gibi bir tasarım şablonu gelecektir.



Şekil 5.32

Burada da bu GUI uygulamasının çalıştırılması durumunda programcı Şekil 5.33'teki gibi bir arayüz ile karşılaşacaktır.



Şekil 5.33

Bu çalışmada amaçlanan liste kutusundaki seçilen elemanlara göre farklı formülasyonlar kullanılarak update butonuna tıklandığında grafik nesnesi üzerinde bulunan sonuçlara göre çizimin yapılmasını sağlamaktır. Bu uygulamada ile programcıya axes (grafik çizimi) nesnesinin nasıl kullanıldığı ve de liste kutuları ile ilgili callback satırlarının nasıl işletildiği hususunda bilgiler verilmektedir. Ayrıca, bu uygulamada "rand(5)" komutunun kullanımı ve rasgele değerler içeren dizilerin nasıl üretildiği de gösterilmiştir. Bu uygulamanın bir başka faydalı yönü de uygulamanın menü içermesi ve menülerin programlanması konusunda tasarımcıya önbilgi verilmektedir. Menülerin kullanımı ile ilgili olarak

- file = uigetfile('*.fig') komutu ile bir başka .fig dosyanın bir grafik çizim nesnesinde nasıl aktarılacağı,

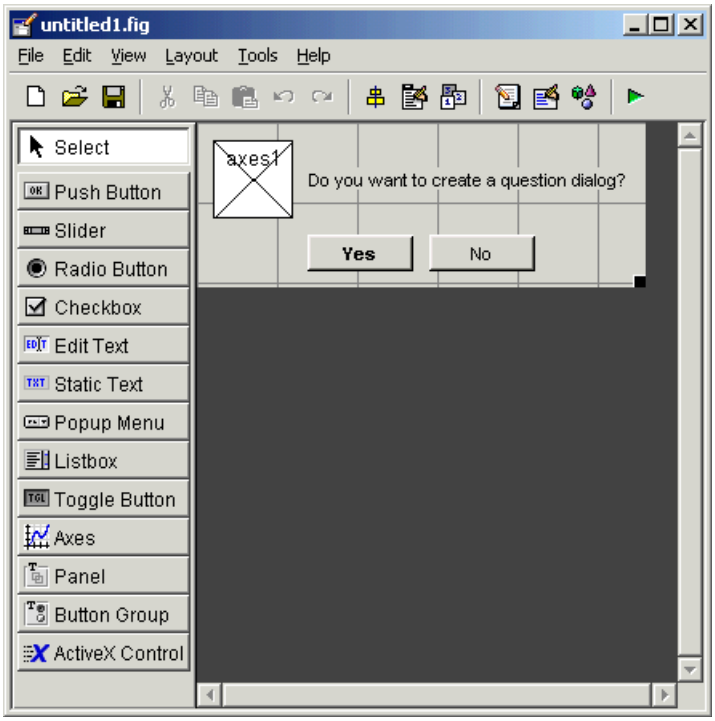
- printdlg(handles.figure1) komutu ile de varolan bir çizimin yazıcıdan nasıl çıktı alınacağı

- close komutu ile aktif bir GUI figure ekranının nasıl kapatılacağı

konularında programcıya programlama teknikleri sunulmaktadır.

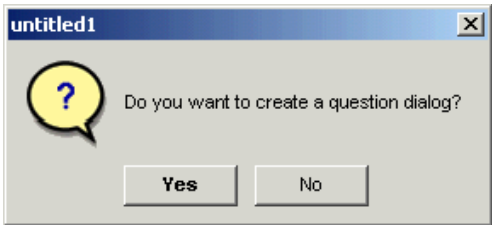
5.7.12.3 "Modal Question Dialog" Uygulaması

Bu seçenek seçildiğinde karşımıza Şekil 5.34'teki gibi bir template GUI uygulaması çıkacaktır.



Şekil 5.34

Bu uygulamayı çalıştırdığımızda da Şekil 5.35'teki gibi bir GUI arayüzü ile karşılaşılır.



Şekil 5.35

Bu GUI uygulamasında amaçlanan kendisine komut satırından verilen dış parametreleri alarak kendi içerisinde yorumlamak ve buna göre oluşturulacak GUI uygulama ekranının içeriğini (burada pencere başlığı ile text kutusunun string değerini) değiştirerek kullanıcıya sunmaktır. Ayrıca, bu uygulama ile programcı şu konularda bilgilendirilmektedir:

- varargin ve varargout komutlarını kullanarak dışarıya bilgi gönderme ve dışarıdan verilen parametreleri yorumlamak

- uiwait(handles.figure1) komutu ile bir uygulama penceresinin aktif konumunun nasıl bloklandığı ve ne iş yaptığı,

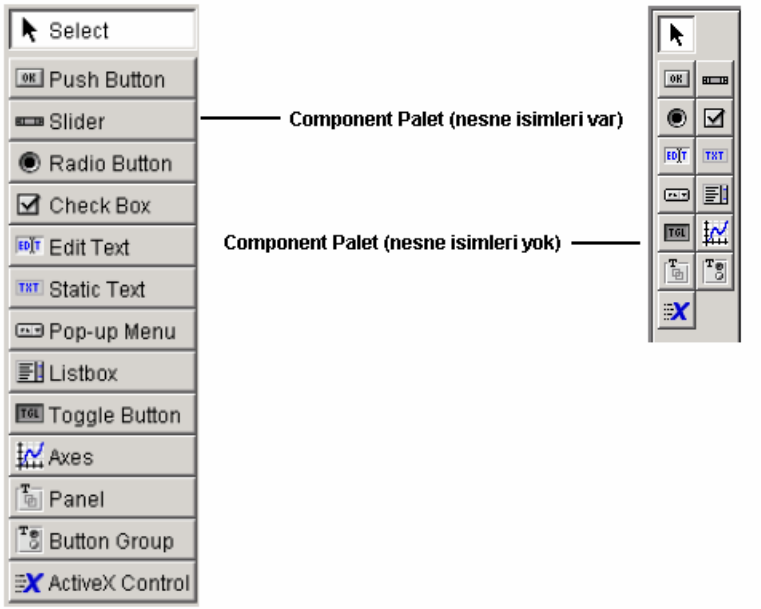
- uiresume(handles.figure1) komutu ile bloklanan bir GUI penceresinin nasıl eski haline getirildiği,

- "modal" programlama tekniği ile tasarlanan bir GUI uygulamasında açılan bir pencerenin arka taraftaki bir başka pencerenin aktif kontrolünü ele geçirmesi için gerekli programlamanın nasıl yapıldığıdır.

Ayrıca, bu uygulama MATLAB GUI tasarımlarında birden fazla form ile nasıl çalışılabileceği konusunda bir ipucu niteliği taşımaktadır. Bu uygulama ile giriş ve çıkış parametreleri kullanan GUI uygulama pencereleri vasıtasıyla kendi içinde birden fazla GUI arayüzü içeren GUI uygulamalarının programlanma tekniği tanıtılmaktadır.

5.7.13 GUI Nesnelerinin Açıklanması

MATLAB GUIDE aracı kullanarak boş (blank) bir GUI çalışma ekranını açtığımızda sol tarafta görülen component panel pek çok nesnenin kullanılabileceği görülmektedir.



Şekil 5.36

Şimdi bu nesnelerin sırasıyla özellikleri ile ilgili bilgiler verilecek ve nasıl programlanacağı gösterilecektir.

5.7.13.1 Push Button:

Normal bir buton özelliği taşımaktadır. Bir buton üzerine tıklanması ile yapılacak komutlar bu buton ile ilgili callback lerin altına yazılır.

5.7.13.2 Toggle Buton:

Çift durumlu bir buton özelliği taşıyan bu nesne ile iki farklı seçenek içeren durumlarda örneğin bu buton basılı ise bir işlemin, bu buton basılmamış ise başka işlemlerin yapılması gerektiği yerlerde tercih edilen bir nesnedir. Buton grubu nesnesi ile beraber kullanımı tavsiye edilir.

5.7.13.3 Radio Buton:

Birden fazla seçeneğin olduğu, ancak seçeneklerden sadece herhangi birinin seçilebileceği hallerde bu nesne kullanılır. Buton grupları ile kullanılması genellikle tercih edilir.

5.7.13.4 Check Box:

Kullanıcıya seçim yapabileceği ve birden fazla şıkkı işaretleyebileceği durumlarda bu nesne kullanılır.

5.7.13.5 Edit Text: 

Bir kullanıcıdan bilgi girişi ya da bir değerin alınması söz konusu olduğunda giriş elemanı olarak sıklıkla kullanılan bir nesnedir.

5.7.13.6 Static Text: 

Kullanıcıya herhangi bir bilgi verme ya da bulunan bir sonuç veya değeri gösterme amacıyla sıklıkla kullanılan bir nesnedir.

5.7.13.7 Slider:

Kullanıcıdan bir giriş değerini kaydırılmak suretiyle kolaylıkla alınmasına imkân veren bir nesnedir.

5.7.13.8 List Box: 

Kullanıcıya bilgi verme amacıyla kullanılabileceği gibi bir değeri listeden seçmek amacıyla da kullanılan sabit bir liste kutusu niteliğinde kullanılan bir nesnedir.

5.7.13.9 Pop-Up Menu: 

Kullanıcıdan alınmak istenilen bilgileri açılan bir listeden seçme özelliği taşıyan bir nesnedir.

5.7.13.10 Axes: 

Yapılan iş ile ilgili grafik çizimlerinin kullanıcıya gösterilmesini sağlayan bir nesnedir.

5.7.13.11 Panel: 

GUI yüzeyi nesnelerinin kullanıcıya daha anlamlı ve güzel gözükmesini sağlayan, ayrıca tasarımcıya GUI dizaynında kolaylık sunan bir nense olup, GUI yüzeyi nesnelerinin gruplanması ve bir arada gösterilmesi amacıyla kullanılır.

5.7.13.12 Button Group: 

Radio veya toggle tipteki buton nesnelerinin bir arada kullanılarak kullanıcının birden fazla seçenekten sadece bir tanesini seçmesini sağlamak amacıyla kullanılan bir nesnedir

5.7.13.13 ActiveX Component: 

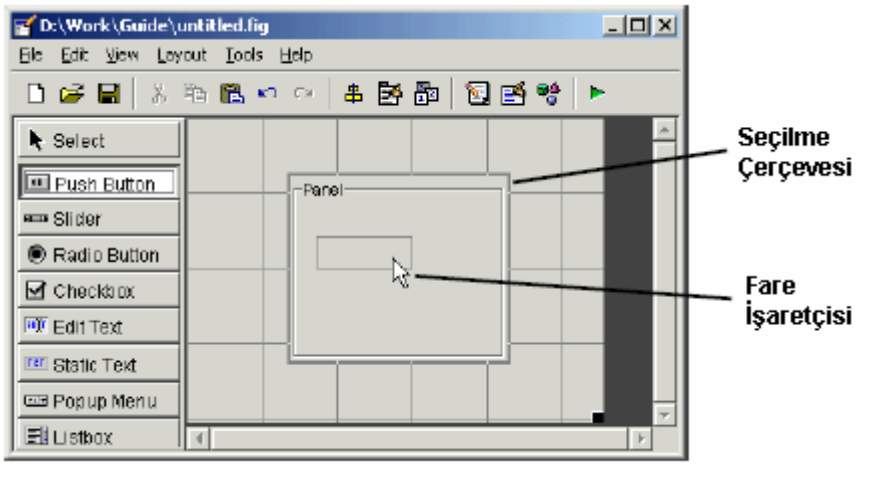
MATLAB GUI tasarımları sadece yukarıda belirtilen nesneler ile sınırlı değildir. Tasarıcı ve programcı ayrıca, ActiveX adı verilen ve değişik alternatifi olan nesnelerin kullanılmasına da imkân verir. Böylece hem tasarımcı hem tasarlanılacak GUI arayüzünün kullanımı bakımından kullanıcıya esneklik sağlanmış olunur.

5.7.14 Nesnelerin GUI Yüzeyine Yerleştirilmesi

Bir nesneyi çalışma alanına eklemek için yapılması gereken sol tarafta yer alan component panelden yerleştirilmek istenilen nesnenin butonunu tıklamak ve GUI yüzeyinde yerleştirilmek istenilen yere tıklamak ya da yerleştirilmek istenilen bölgeyi farenin sol tuşu ile basılı tutarak beliren çerçevenin nesne boyutları olacağını düşünerek yerleştirme işlemi yapılabilir.

5.7.14.1 Bir Nesnenin GUI Yüzeyinde Bulunan Bir Panele ya da Buton Grubuna Yerleştirilmesi

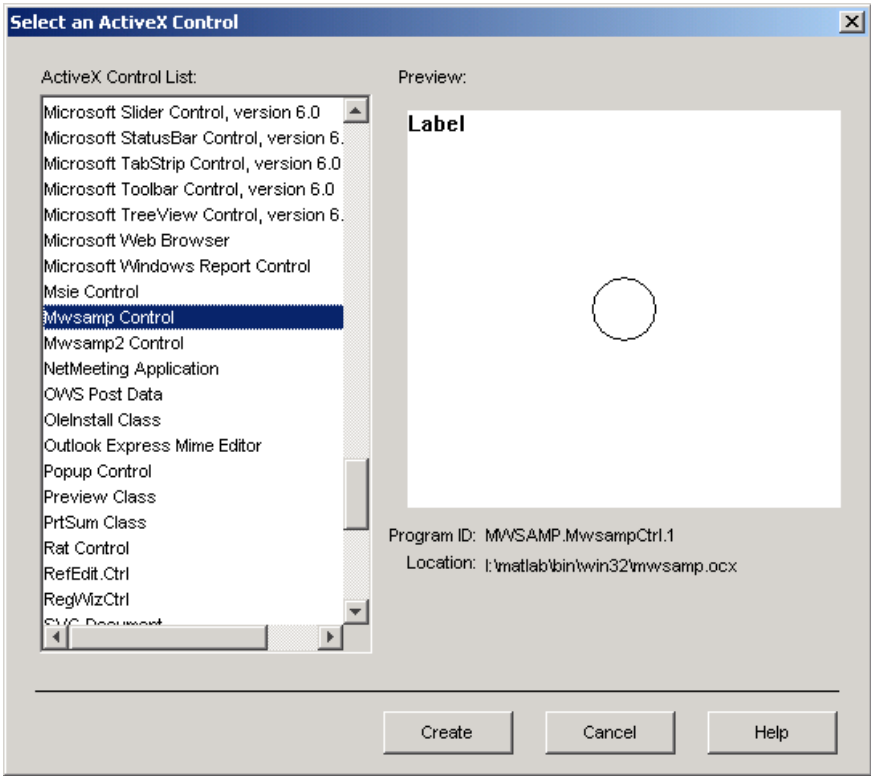
Bir nesne GUI güzeyinde daha önceden yerleştirilmiş olan bir panele ya da buton grubuna yerleştirmek için öncelikle yerleştirilecek nesne component panelden seçilir ve daha sonra fare işaretçisi yerleştirilecek panel ya da buton grubu üzerine götürülür. Bu durumda fare imlecinin üzerinde olduğu bu grup nesnesi bir anda seçili hale gelecektir. Şayet bu işlem ana GUI figure üzerine getirilirse bu sefer grup nesnesinin aktifliği kaybolacak ve figure yüzeyi seçili duruma dönüşecektir. (Bir nesnenin aktif veya seçili olduğu durumu nensnenin kenarında beliren çerçevelerin varlığı ile anlaşılabilir.) Bu durumu GUI yüzeyine yerleştirilmiş olan bir panel nesnesine bir push buton nesnesinin yerleştirilmesi örneği üzerinde Şekil 5.37'de görüldüğü üzere özetlenebilir.



Şekil 5.37

5.7.14.1.1 ActiveX Control Nesnesinin GUI Yüzeyine Yerleştirilmesi

Bir ActiveX nesnesini GUI yüzeyine eklemek için öncelikle component panelden seçilir. Daha sonra GUI alanında yerleştirilmesi düşünülen bir yere farenin sol tuşu ile tıklanır. Bu adımdan sonra karşımıza Şekil 5.38'deki gibi bir pencere gelecektir. Bu pencerede GUI yüzeyine

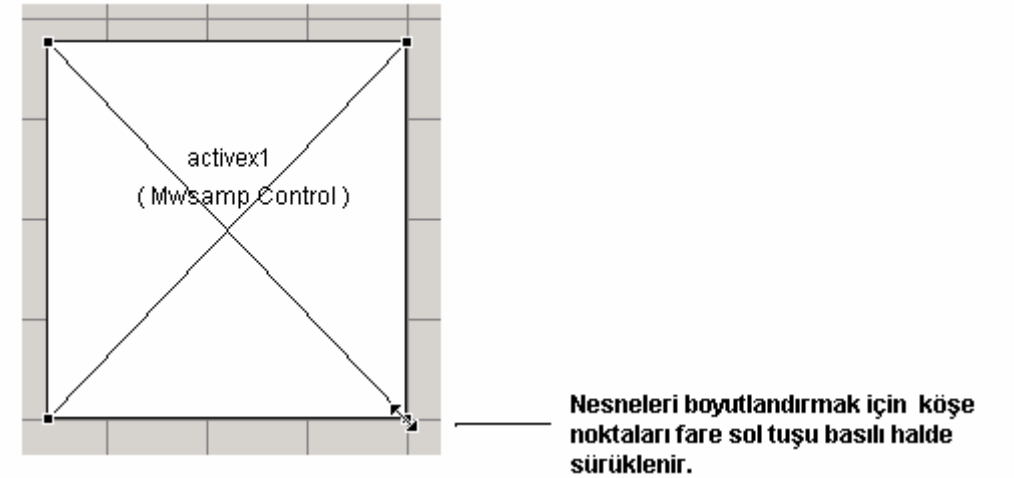


Şekil 5.38

yerleştirilmek istenmek istenilen ActiveX componenti sol taraftaki listeden seçilmeli ve ardından Create butonuna basılmalıdır.

5.7.15 GUI Yüzeyine Eklenen Nesnelerin Boyutlandırılması

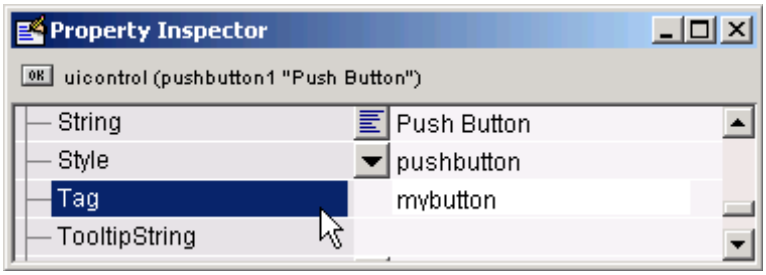
GUI alanına eklenen bir nesnenin boyutlarını değiştirilmek için ilgili nesne öncelikle farenin sol tuşu yardımı ile seçilir. Daha sonra da etrafında beliren köşe noktaları kullanılarak boyutları değiştirilebilir. Bu durum Şekil 5.39'da gösterilmiştir.



Şekil 5.39

5.7.16 Her Nesneye Tanıtıcı Bir İsim Atamak

Bir nesneye tanıtıcı ve o nesneye özel bir isim vermek için öncelikle o nesne farenin sol tuşu ile GUI tasarım yüzeyinde seçilir. Daha sonra View menüsünden Property Inspector komutu verilir. Karşımıza gelen özellikler penceresinden nesnemizin Tag özelliğine istenilen bir isim verilebilir. Ayrıca, Property Inspector penceresini açmak için nesne üzerinde farenin sol tuşu ile çift tıklanılarak da açılabilir. Şekil 5.40'da örnek olarak bir buton nesnesi için bu durum görülmektedir.



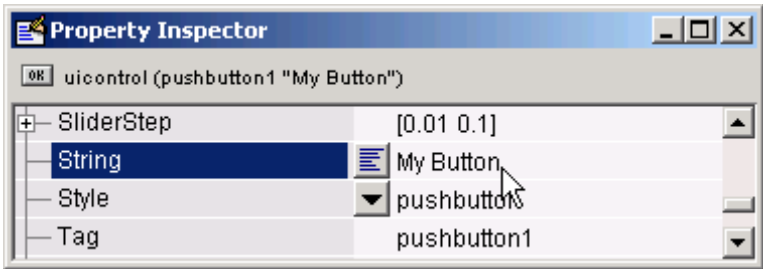
Şekil 5.40

5.7.17 Nesnelere Yazı Eklemek

Bazı nesneler özellikleri gereği kullanıcıya bilgi vermek veya bir seçenek sunmak amacıyla string bilgiler içerirler. Bu nesneler Push Button, Toggle Button, Radio Button, Check Box, Text, List Box, Popup Menu, Panel ve Button Group nesneleridir. Bu nesneler de yapıları gereği değişik özellikler içermektedir ve bu özellikleri üzerinden string değerler atamak mümkün olmaktadır. Bu nesnelere yazı bilgilerinin nasıl verildiği aşağıda sırasıyla açıklanmıştır.

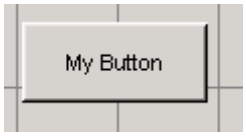
5.7.17.1 Push Button, Toggle Button, Radio Button, Check Box, Text Nesnelerine Yazı Eklenmesi

Bu nesnelere yazı eklemek için ilgili nesne seçilerek özellikler penceresinden String özelliklerine istenilen bir metin bilgisi girilebilir. Ayrıca, bu özellik programlama komut satırlarıyla da değiştirilebilir. Örnek olarak Şekil 5.41'de görülen örnekte bir push butonun üzerindeki yazının değiştirilmesi gözükmektedir. Eğer ki bu nesnelere alt alta olacak şekilde birden fazla satır içeren bilgiler girilmek istenirse bir sonraki başlık altında anlatılan teknik kullanılmalıdır.



Şekil 5.41

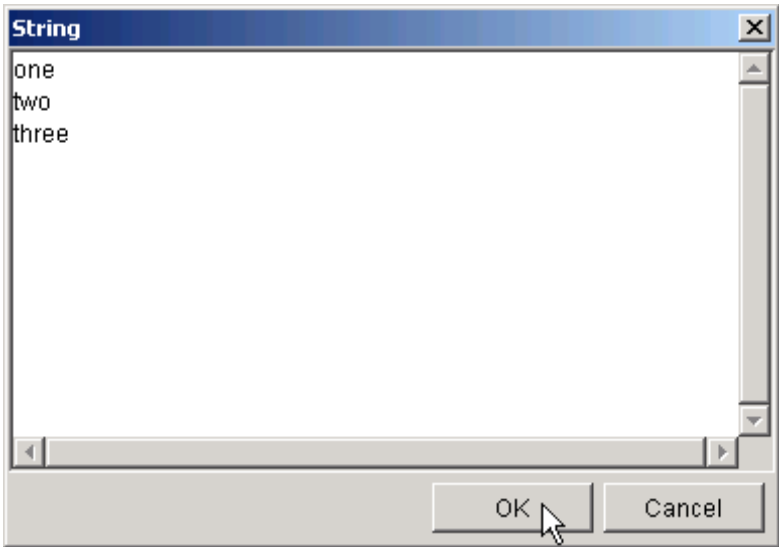
Gerekli değişiklik yapıldığında butonumuzun görüntüsü Şekil 5.42'deki gibi olacaktır.



Şekil 5.42

5.7.17.2 List Box ve Popup Menu Nesnelerine Yazı Eklenmesi

Bu nesneler çoklu string bilgiler içeren liste kutusu tarzı yapılardır. Bu nesnelere string bilgiler eklemek istenirse öncelikle nesne seçilir. Ardından String özelliğinin yanında yer alan butona tıklanır. Karşımıza aşağıdaki gibi bir pencere gelecektir. Bu pencereye gerekli bilgiler girildikten sonra OK butonuna basılır. Böylece string verilerin girilmesi işlemi tamamlanmış olur. İstenirse bu özellik programlama yoluyla da değiştirilebilir. Bu yöntem eğer ki bir önceki başlıkta belirtilen nesnelere birden fazla bilgi girilmesi istendiğinde de bu nesneler için de kullanılabilir.



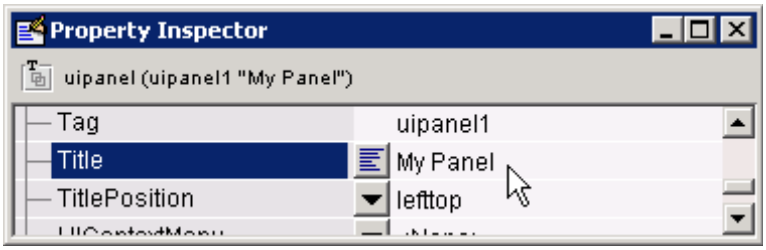
Şekil 5.43

Şekil 5.43'te görülen bilgiler girildiğinde örneğimizdeki popup menü nesnemizin görüntüsü Şekil 5.44'te görüldüğü gibi olacak ve kenarındaki buton tıklandığında da tüm seçenekler kullanıcıya sunulacaktır.

Şekil 5.44

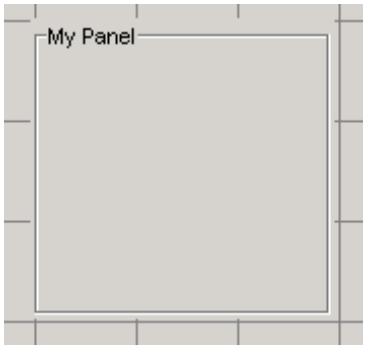
5.7.17.3 Panel ve Buton Group Nesnelerine Başlık Eklenmesi

Bu nesneler içinde yer alan pek çok nesne gruplandırma imkânına sahip olup bu nesnelere başıl eklemek için farklı bir özellik kullanılmaktadır. Bu nesnelere başlık ekmek için nesne seçili iken Property Inspector penceresinden Title özelliğine gerekli bilgi girilmelidir. Bu durumu Şekil 5.45'ten de takip edilebilir.



Şekil 5.45

Örneğimizde bir panel nesnesi kullanılmış olup başlığı My Title olarak değiştirilmiştir. Bu değişiklik yapıldıktan sonra panelimizin GUI çalışma alanındaki görüntüsü Şekil 5.46'daki gibi olacaktır.



Şekil 5.46

5.7.18 GUI Çalışma Alanında Nesneler ile Çalışma

GUI yüzeyindeki nesnelere tasarım ortamında istenildiği şekilde müdahale edilebilir. GUI çalışma alanındaki nesneler üzerinde kopyalama, silme, taşıma, öne getirme, en arkaya gönderme, hizalama, tab tuşu ile seçim sırasının değiştirilmesi, başka bir noktaya taşınması veya boyutlarının değiştirilmesi, cetvel ve ızgara kullanılarak işlemlerin yapılması, GUI uygulamasında ana menü oluşturmak ya da istenilen bir nesne üzerinde context menü oluşturmak, GUI uygulamasına araç çubuğu eklemek, GUI tasarımında kullanılan nesnelerin görülmesi gibi pek çok işlem için GUIDE tasarımcıya pek çok kolaylık sağlamaktadır.

5.7.18.1 Nesnelerin Seçilmesi

GUI yüzeyindeki nesneleri teker teker seçmek için fare işaretçisi ilgili nesne üzerine götürüp farenin sol tuşuna basmak yeterlidir. Ancak, tasarımcı birden fazla nesneyi seçmek istiyor ise o zaman Ctrl ya da Shift tuşu basılı halde iken ilgili nesneleri birer birer tıklamalıdır. Ayrıca, çoklu seçim işlemi için fare işaretçisi GUI yüzeyinin boş bir yerinde farenin sol tuşu basılı tutularak bu halde hareket ettirilir. Bu esnada oluşan pencere içerisinde seçilmesi istenilen nesneler var olduğunda farenin sol tuşu bırakılır. Bu şekilde de topluca seçim işlemi yapılmış olacaktır.

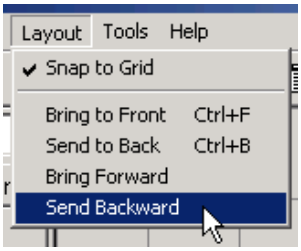
5.7.18.2 Nesneler Üzerinde Kopyalama, Silme, Taşıma, Çoğullama İşlemlerinin Yapılması

GUI yüzeyine eklenene bir nesneden kısa yoldan yeni bir kopya almak istenirse nesne üzerinde farenin sağ tuşu ile tıklanır ve açılan menüden Duplicate (Çoğalt) komutu verilir. Bu şekilde kopya alınan nesnenin tüm görünüm özellikleri aynı yeni bir kopyası oluşturulmuş olacaktır. Ayrıca, bir nesne kopyalama ve yapıştırma tekniği ile de çoğaltılabilir. Bunun için nesne üzerinde farenin sağ tuşu tıklanır açılan menüden de Copy (Kopyala) komutu verilir ya da bu işlem yerine kısaca Ctrl + C tuş kombinasyonu kullanılabilir. Daha sonra GUI yüzeyinin istenilen bir noktasına fare işaretçisi götürülür ve ardından farenin sağ tuşu tıklanır. Açılan menüden Paste (Yapıştır) komutu verilir. Bu işlem kısaca Ctrl + V tuş kombinasyonu ile de yapılabilir.

Bir nesnenin bir noktadan başka bir noktaya taşınması istenirse nesne ya farenin sol tuşu basılı halde istenilen noktaya sürüklenebilir ya da nesne üzerinde farenin sağ tuşu tıklanıp açılan menüden Cut (Kes) komutu verilir. Bu işlem kısaca Ctrl + X tuş kombinasyonu ile de gerçekleştirilebilir. Ardından istenilen noktaya fare işaretçisi götürülür ve sağ tuşu tıklanıp açılan menüden Paste (Yapıştır) komutu verilir.

5.7.18.3 Bir Nesneyi Diğer Nesneler Arasında Öne veya Arkaya Getirme

Bir nesnenin diğer nesnelere göre pozisyonunu değiştirmek için önce nesne seçilir. Daha sonra Layout menüsünden ilgili komut verilir. Bu durum Şekil 5.47'de de gözükmektedir. Buradaki komutların işlevleri şöyledir:



Şekil 5.47

- Bring to Front komutu seçili nesneyi en öne getirir.
- Send to Back komutu seçili nesneyi en arkaya gönderir.
- Bring Forward komutu seçili nesneyi bir adım öne getirir.
- Send Backward komutu seçili nesneyi bir adım arkaya gönderir.

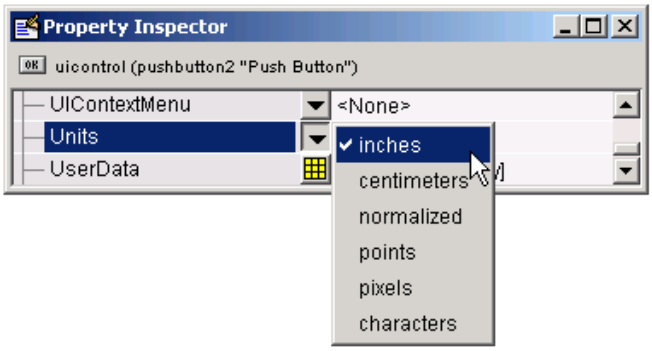
5.7.18.4 Nesnelerin GUI Çalışma Alanında Taşınması

Bir nesneyi GUI yüzeyinde farklı bir noktaya götürmek için üç farklı yöntem vardır. Bunları açıklayalım.

İlk yönteme göre bir nesne fare işaretçisi üzerinde iken farenin sol tuşu ile tıklanır ve bırakılmadan farklı bir noktaya sürüklenir. Bu şekilde nesnenin konumu kolaylıkla değiştirilmiş olacaktır.

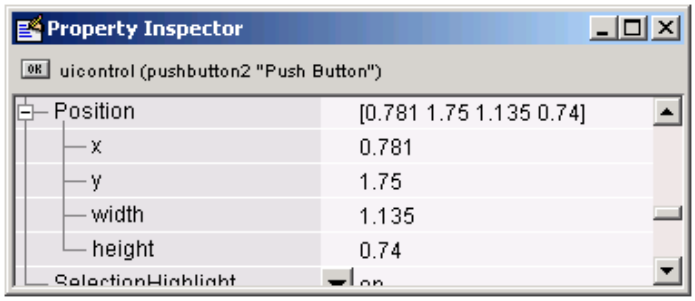
İkinci yönteme göre bir nesne önce farenin sol tuşu ile seçilir. Ardından klavyedeki yer-yön tuşları vasıtasıyla istenilen yönlerde hareket ettirilebilir. Bu yöntem özellikle bir nesne daha hassas olarak başka bir konuma yerleştirilmek istendiğinde kullanılır.

Son olarak da bir nesnenin konumu nesne seçili iken Property Inspector penceresinden Posiiton özellikleri değiştirilir. Bu şekilde de bir nesnenin konumu değiştirilmiş olacaktır. Ancak, standart ölçü birimi olarak GUIDE normalized veya characters ölçü biirmlerini kullanmaktadır. Bunlar yerine öncelikle ölçü birimini inches olarak değiştirilirse bir nesnenin konumlandırmasında tasarımcıya çok büyük kolaylık sağlayacaktır. Bu durum Şekil 5.48'de görülmektedir.



Şekil 5.48

Bu adımdan sonra da Position özelliği kullanılarak ilgili nesnenin konumu değiştirilebilir. Bu özellik hiyerarşik yapılı değişken özelliği göstermekte olup alt parametrelerinin değiştirilmesi için özellikler penceresinde Position özelliğinin yanındaki + işareti tıklanır. Böylece alt parametreler görülebilir. Örnek pencere görüntüsü Şekil 5.49'da verilmiştir. Buradaki bilgiler şu özelliktedirler:



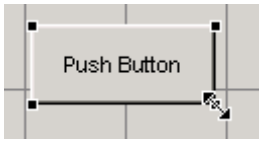
Şekil 5.49

- x bilgisi, bir nesnenin sol kenardan itibaren uzaklığnı gösterir.
- y bilgisi, bir nesnenin alt kenardan itibaren uzaklığını gösterir.
- width bilgisi, bir nesnenin yatay uzunluğunu gösterir.
- height bilgisi, bir nesnenin dikey uzunluğunu gösterir.

Bu yöntemin kullanılması x ve y parametreleri dışında width ve height özellikleri kullanılarak bir nesnenin boyutlarının da değiştirilmesi bakımından tasarımcıya yarar sağlar.

5.7.18.5 Nesnelerin Boyutunu Değiştirmek

Bir nesnenin boyutlarını değiştirmek için nesne önce farenin sol tuşu ile seçilir. Daha sonra nesnenin her bir köşesinde beliren siyah boyutlandırma kutucuklarından istenilen biri üzerine fare işaretçisi götürülür ve üzerinde iken farenin sol tuşu basılı tutulur. Daha sonra bu tuş basılı halde iken işaretçi ileri ve geri hareket ettirilir. Bu şekilde ilgili nesnenin konumu değiştirilmiş olacaktır. Bu durum Şekil 5.50'de de görülmektedir.



Şekil 5.50

Ayrıca, bir nesnenin konumunu değiştirmek için özellikler penceresi de kullanılabilir. Bir nesnenin width ve height özelliklerinin değiştirmek suretiyle nesnemizin boyutları değiştirilmiş olacaktır. Bu durum için bir önceki başlık altında bulunan bilgilere bakılabilir.

5.7.18.6 Nesnelerin Hizalanması

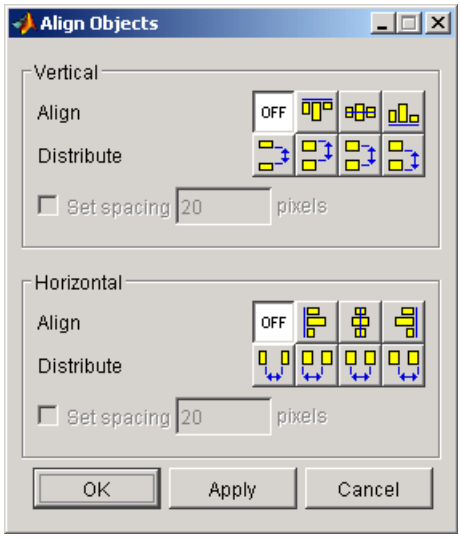
GUIDE bu konuda sunduğu çeşitli ve yararlı araçlar sayesinde tasarımcının işini kolaylaştırmaktadır. Bu araçlar şunlardır:

- Alignment Tool (Hizalama Aracı)
- Property Inspector (Özellikler Penceresi)
- Grid and Rulers (Izgara ve Cetvel)
- Guide Lines (Klavuz Çizgileri)

Aşağıda bu araçlar sırayla açıklanmıştır.

5.7.18.6.1 Alignment Tool (Hizalama Aracı)

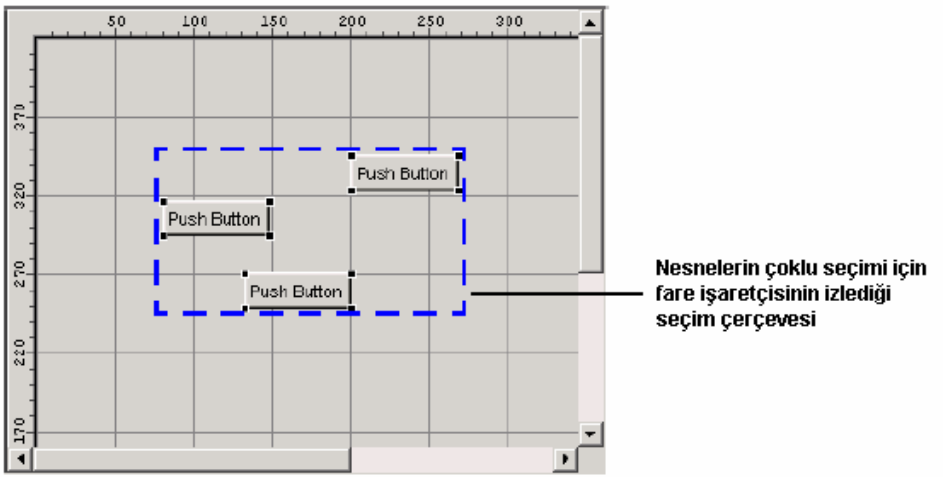
Birden çok nesneyi yatay ve dikey olarak belirlenen referans hattına göre hizlama için bu araç kullanılır. Hizalanacak olan nesneler önce çoklu olarak seçilirler. Ardından Tools menüsünden Align Objects komutu verilir. Karşımıza Şekil 5.51'deki gibi bir pencere gelir.



Şekil 5.51

Bu pencerede Vertical (Dikey) ve Horizontal (Yatay) olmak üzere nesneler hizalanabilir. Eğer ki sadece dikey hizalama yapmak isteniyorsa yatay hizalama seçeneği Off (Kapalı) durumda tutulmalıdır. Ancak isteğe göre hem dikey, hem yatay hizalama aynı anda da yapılabilir. Align seçeneği nesneleri hizalamak için kullanılırken Distribute nesneler arasında referans noktasına göre boşluk bırakmak için kullanılır. Eğer ki Distribute seçeneklerinden herhangi biri seçilirse "set spacing" seçeneği etkin olacaktır ve bu seçenek işaretlenirse yanında belirtilen kutu içerisindeki piksel sayısı kadar nesneler arasında referans noktasına göre boşluk bırakacaktır. Bu durum Şekil 5.52'de örneklendirilmiştir.

Öncelikle hizalandırılacak nesneler toplu olarak seçilir.

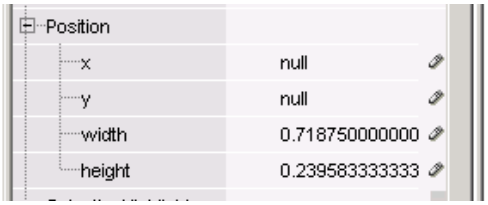


Şekil 5.52

Daha sonra Tools menüsünden hizalama aracı açılır ve gerekli seçimler yapıldığı zaman OK butonuna tıklanır. Böylece nesneler hizalanmış olacaktır. Hizalama isteğiniz gibi olmadı ise herhangi bir anda Ctrl+Z tuş kombinasyonu ile yapılan işlemleri geri alabilirsiniz.

5.7.18.7 Property Inspector (Özellikler Penceresi):

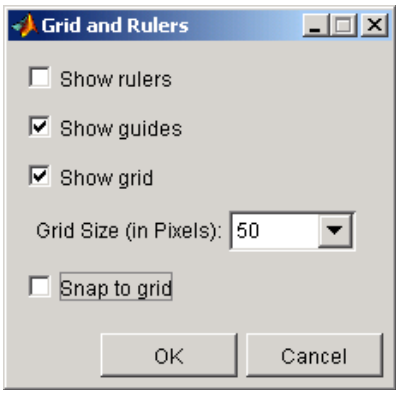
Özellikler penceresi yardımı ile de hizalama yapılabilir. Bunun için önce hizalanacak nesneler toplu olarak seçilir. Daha sonra da View menüsünden Property Inspector penceresi açılır. Gelen pencerede Position özelliği altındaki x ya da y değeri yeni bir değere set edilir. Örneğin, eğer ki nesneler sol kenardan aynı uzaklıkta olacaksa yani dikey olarak hizalanmak isteniyorsa x özelliği aynı (örneğin 2 inches gibi) değere atanır. Benzer şekilde nesneler yatay olarak hizalanmak isteniyorsa y değeri aynı değere ayarlanır. Bu şekilde de nesnelerimiz hizalandırılmış olacaktır. Bu özellikler Şekil 5.53'te de gösterilmiştir.



Şekil 5.53

5.7.18.8 Grid and Rulers (Izgara ve Cetvel)

GUI tasarımında GUIDE aracının tasarımcının nesnelerin hizalanmasında ızgara ve cetvel kullanmasına izin vermesi de tasarımcı için büyük bir kolaylık sağlamaktadır. Bir GUI tasarımında ızgara ve cetvel kullanmak için Tools menüsünden Grid and Rulers komutu verilir. Karşımıza Şekil 5.54'teki gibi bir pencere gelecektir. Buradaki seçenekler şöyledir:



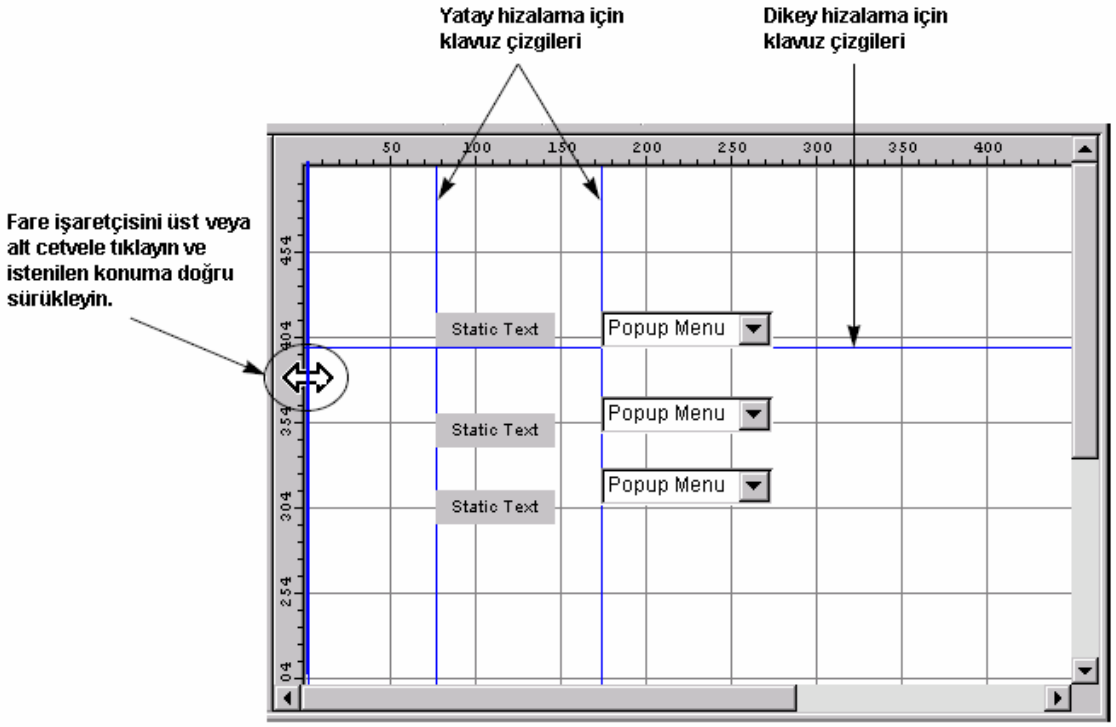
Şekil 5.54

"Show rulers" seçeneği işaretlenirse cetvelleri gösterir.
"Show guides" seçeneği işaretlenirse klavuz çizgileri gözükecektir.
"Show grid" seçeneği işaretlenirse GUI yüzeyinde ızagara belirecektir.
"Snap to grid" seçeneği işaretlenirse nesneler GUI yüzeyinde ızgara hatlarına tutturulacaktır.

İstenirse bu pencerede ızgara aralarının büyüklüğü değişitirlebilir. Bu işlem için Grid Size seçeneği kullanılabilir.

5.7.18.9 Guide Lines (Klavuz Çizgileri)

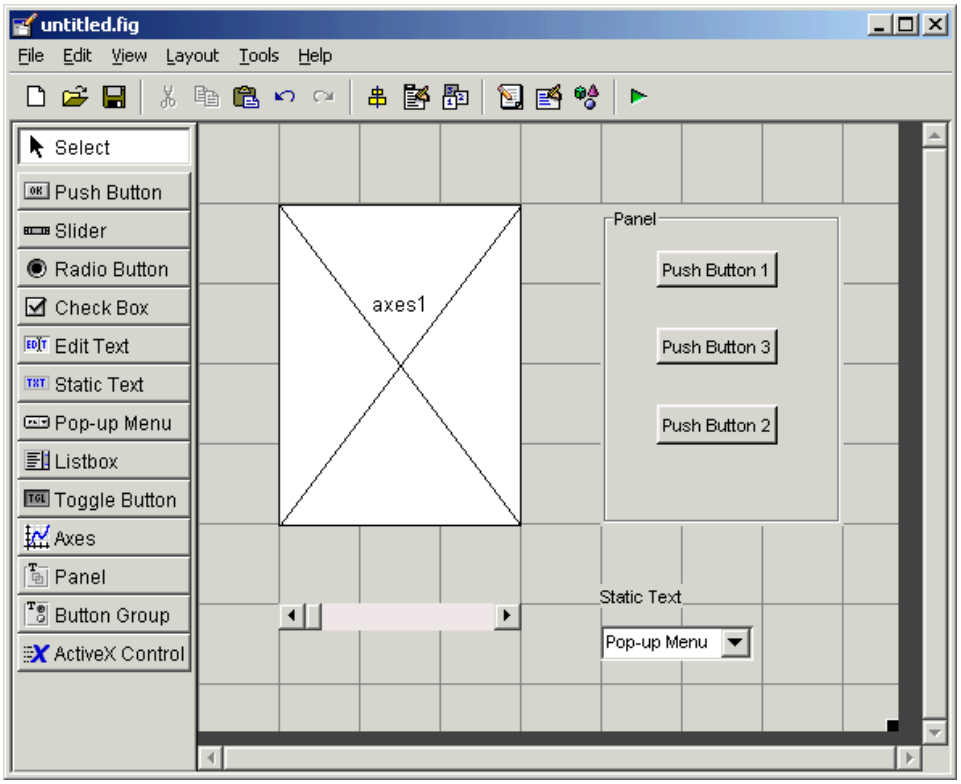
GUI yüzeyi tasarlanırken klavuz çizgileri kullanılarak nesnelerin hizalanması rahatlıkla yapılabilir. Klavuz çizgilerini çıkarmak için tasarımcı önce dikey veya yatay istediği bir cetvel üzerinde ve istediği hizadan başlayarak farenin sol tuşunu basılı tutulur. Bu durumda fare işaretçisi hareket ettirilir. İstenilen hizaya gelindiğinde farenin sol tuşu bırakılır. Böylece GUI çalışma alanında mavi renkte klavuz çizgilerini çıktığı görülecektir. Bu durum Şekil 5.55'de gösterilmiştir.



Şekil 5.55

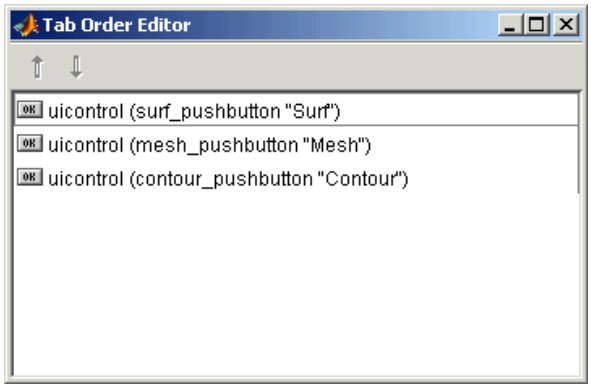
5.7.18.10 Tab Tuşu ile Geçiş Sırasının Ayarlanması

GUI yüzeyinde birden fazla buton veya liste kutuları gibi nesnelerin olduğunu düşünelim. Kullanıcı GUI uygulmasını rahat bir şekilde kullanabilmek için tab tuşu yardımı ile bir nesneden diğerine geçmek isteyebilir. Bu durum için tab sırasının uygun bir sırada değiştirilmesi GUI uygulamamızın kullanım kolaylığını artıracaktır.



Şekil 5.56

Şekil 5.56'da görüldüğü gibi bir GUI arayüzüne sahip olduğu düşünülürse bu arayüzde nesnelerin tab sırasını değiştirmek için öncelikle Tools menüsünden Tab Order editor komutu çalıştırılır.



Şekil 5.57

Daha sonra Şekil 5.57'deki gibi bir pencere ile karşılaşılır. Bu pencereden üst bölümünde yer alan yukarı ve aşağı ok butonları yardımıyla listede görülen nesneler seçilerek tab sıraları değiştirilebilir. Listenin en üstündeki nesneler daha öncelikli tab tuşu geçiş sırasına sahiptirler.

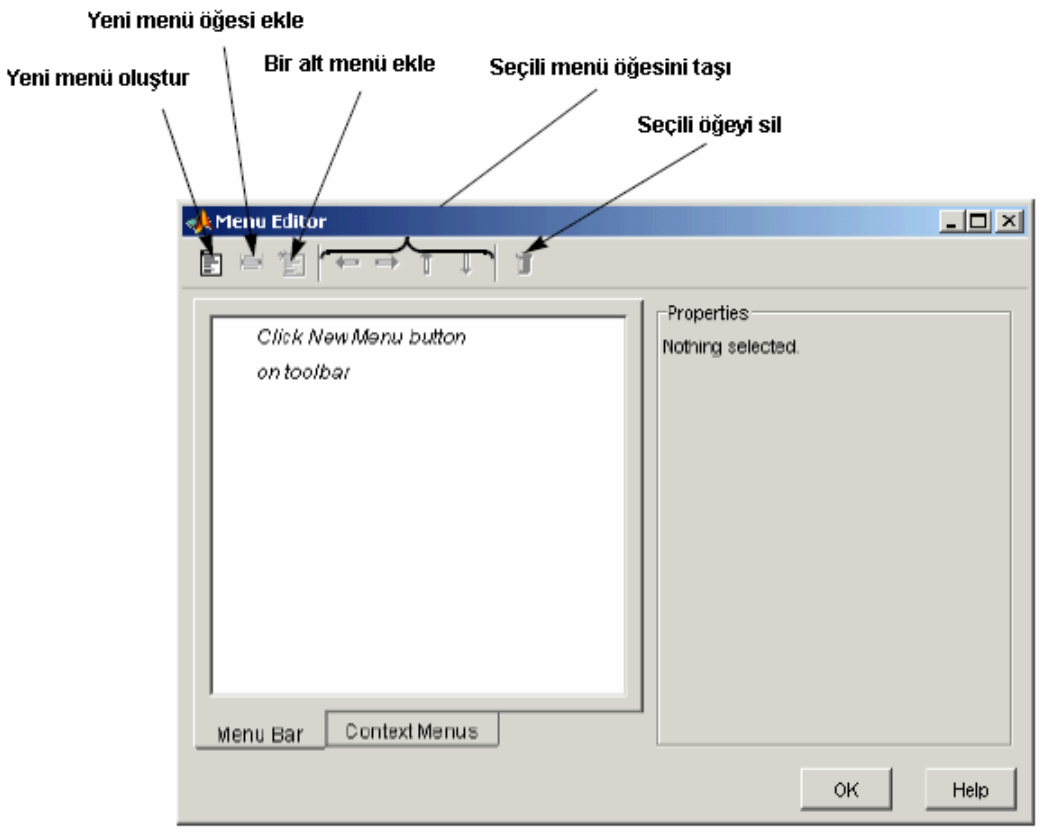
Burada dikkat edilmesi gereken bir husus da Panel, Buton Group ve Axes nesnelerinin tab tuşu sırasında bir etkisi olmaz. Çünkü bu nesneler tab tuşu ile kontrol edilebilecek bir yapıya sahip değillerdir. Ancak, bir panel ya da buton grubu içerisindeki bir nensnenin tab sırası sadece ait olduğu o grup içinde geçerlidir. Bunun için de bu grup nesnelerinden herhangi biri tıklanırsa Tab Order Editor içerisinde o grubun nesneleri gözükecektir.

5.7.18.11 GUI Uygulamasına Menü Eklemek

GUI uygulamarında iki farklı menü eklenebilir. Bunlar dan biri uygulamanın ana ekranında yer alan ve çoğu programlarda yer alan program menüsüdür. Bir diğer menü şekli ise herhangi bir nesne üzerinde farenin sağ tuşu ile tıkladığımızda açılan içerik (context) menüleridir. Aşağıda her iki menü uygulamalarının nasıl yapıldığı ile ilgili ayrıntılı bilgi verilmektedir.

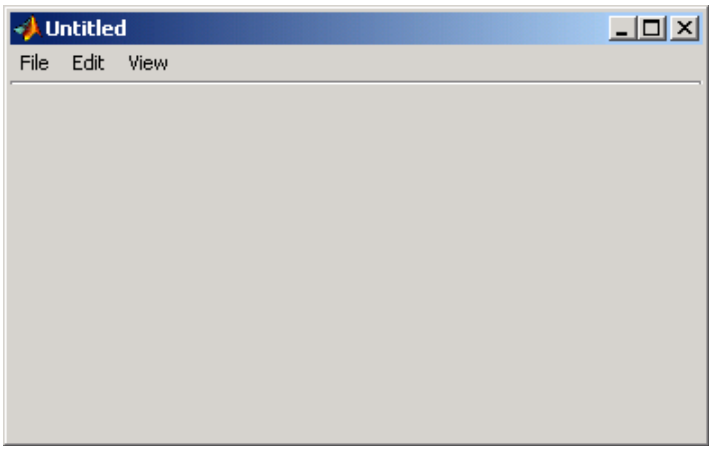
5.7.18.11.1 GUI Uygulamasına Program Menüsü Eklemek

GUI uygulamasına menü eklemek için GUIDE içinde bulunan Menu Editor aracı kullanılır. Bu aracı çalıştırmak için Tools menüsünden Menu Editor komutu verilir. Karşımıza Şekil 5.58'deki gibi bir pencere gelecektir.



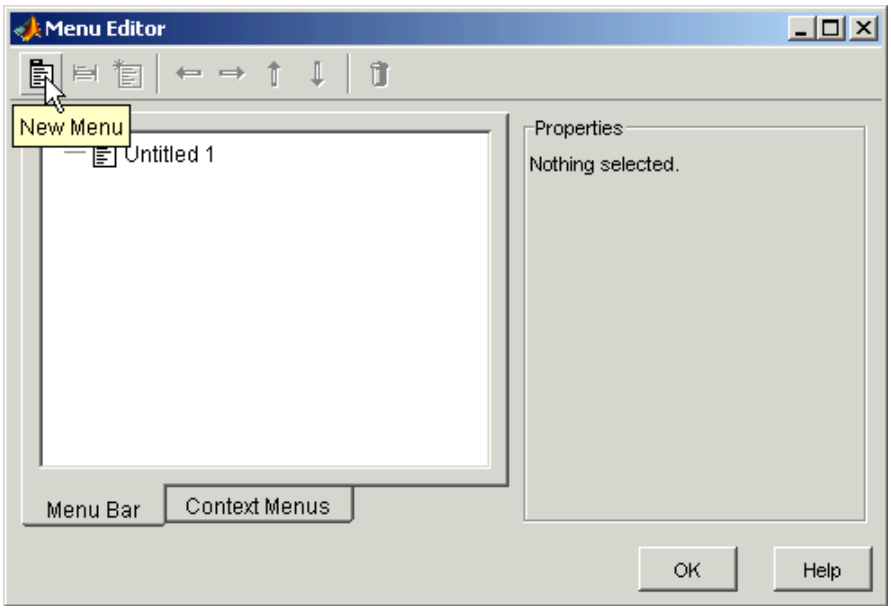
Şekil 5.58

Menu Editor penceresi ile menü ekleme, menüyü yeni bir öğe eklme, varolan bir öğeyi silme, alt menü oluşturma ve pek çok işlem yapılabilir. Örneğin Şekil 5.59'daki gibi bir GUI arayüzü tasarlayacağımızı düşünelim.



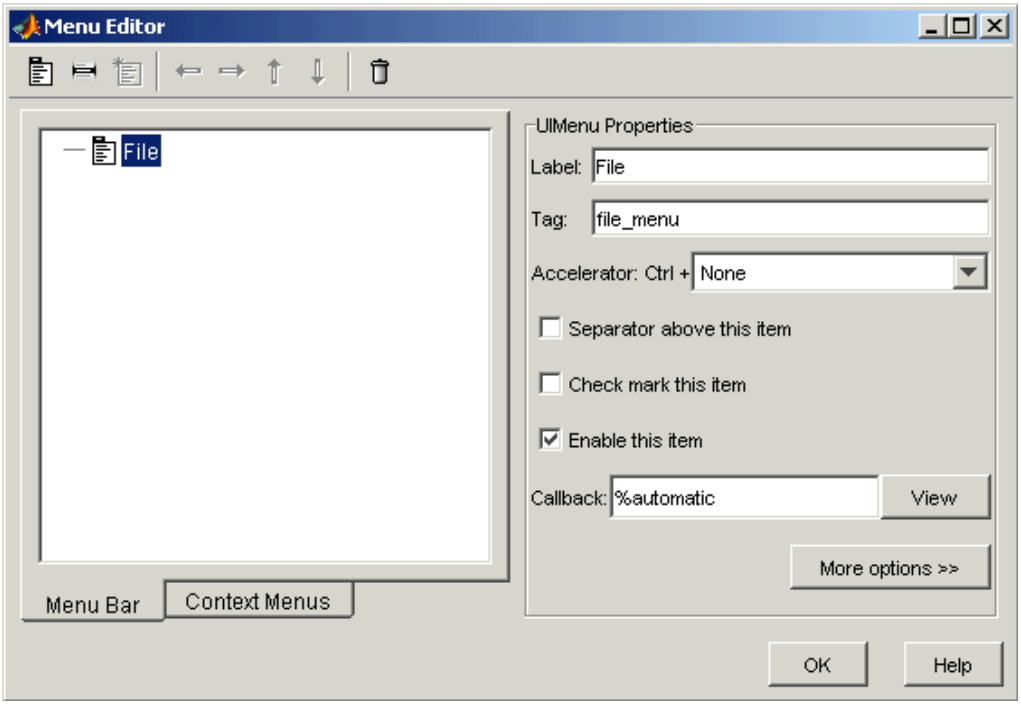
Şekil 5.59

Böyle bir tasarım için öncelikle New Menu tuşu ile yeni bir menü eklenir. Bu durum Şekil 5.60'da gösterilmiştir.



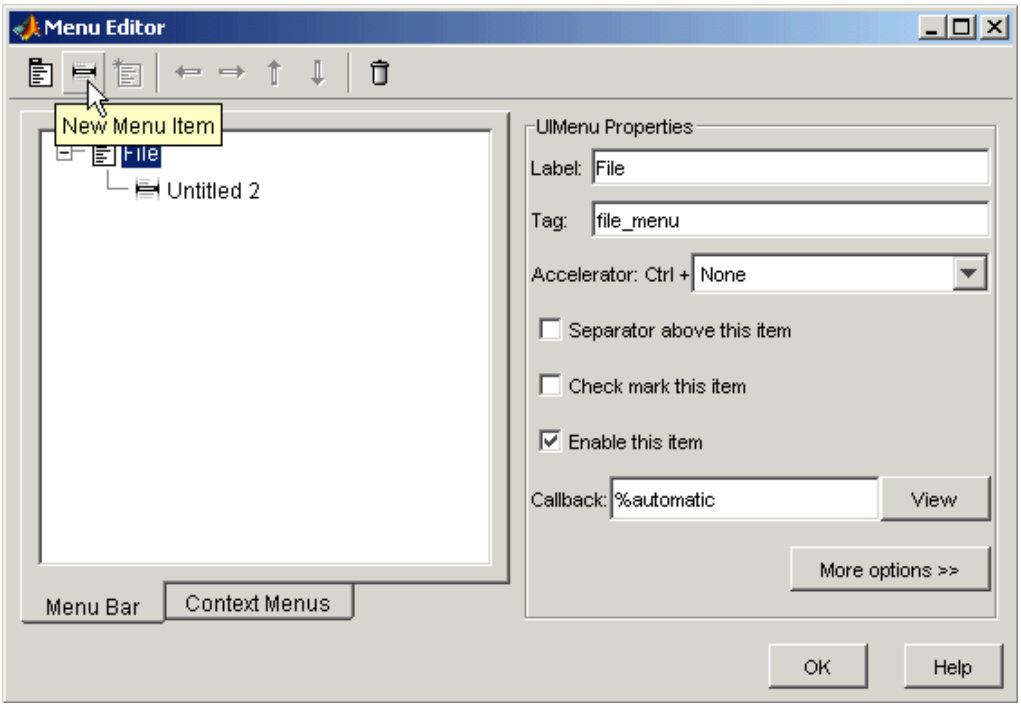
Şekil 5.60

Daha sonra bu menü listeden seçilip Properties (özellikler) panelinden bu menüye "File" etiketi verilir. Bu bilgi Label özelliğine verilir. Tag özelliğine de GUI'yi programlarken kullanılacak tanıtıcı bir isim verilmelidir. Ancak, Tag bilgisinin Label değeri ile aynı olma zorunluluğu yoktur.



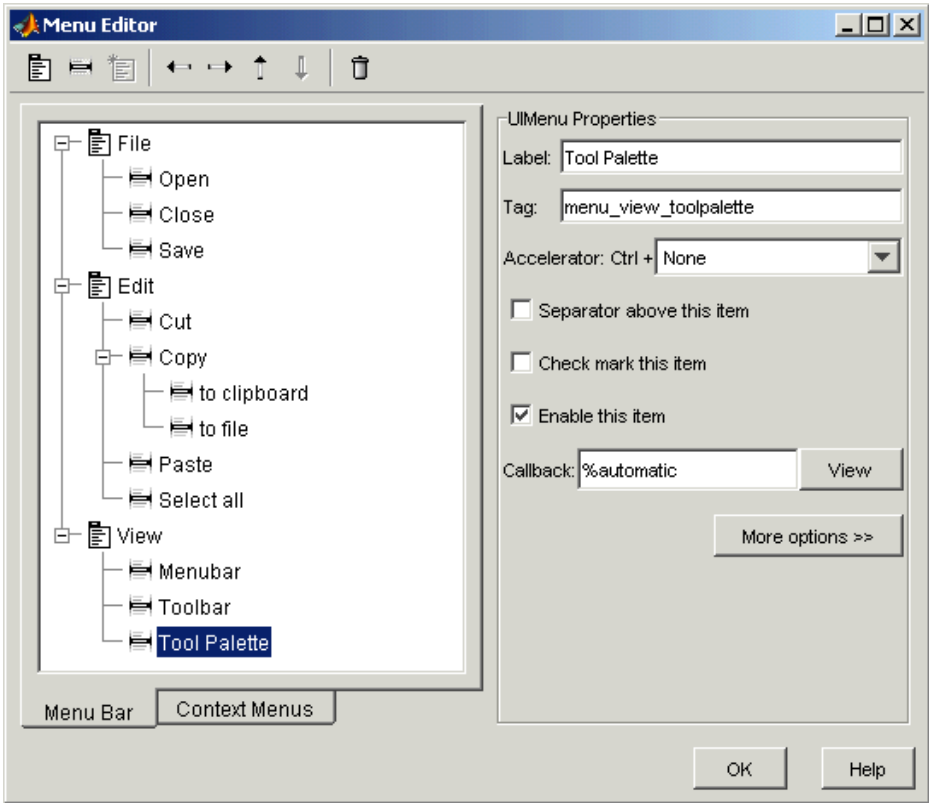
Şekil 5.61

Hazırlanan File menüsüne alt bir öğe eklemek için New Menu Item butonu tıklanır. Böylece yeni öğe File menüsüne eklenmiş olacaktır.



Şekil 5.62

Benzer şekilde bu öğeye de yeni bir etiket (label) verilir. Tüm bu işlemler tamamlandığı zaman Menu Editor penceremiz aşağıdaki gibi gözükecektir.

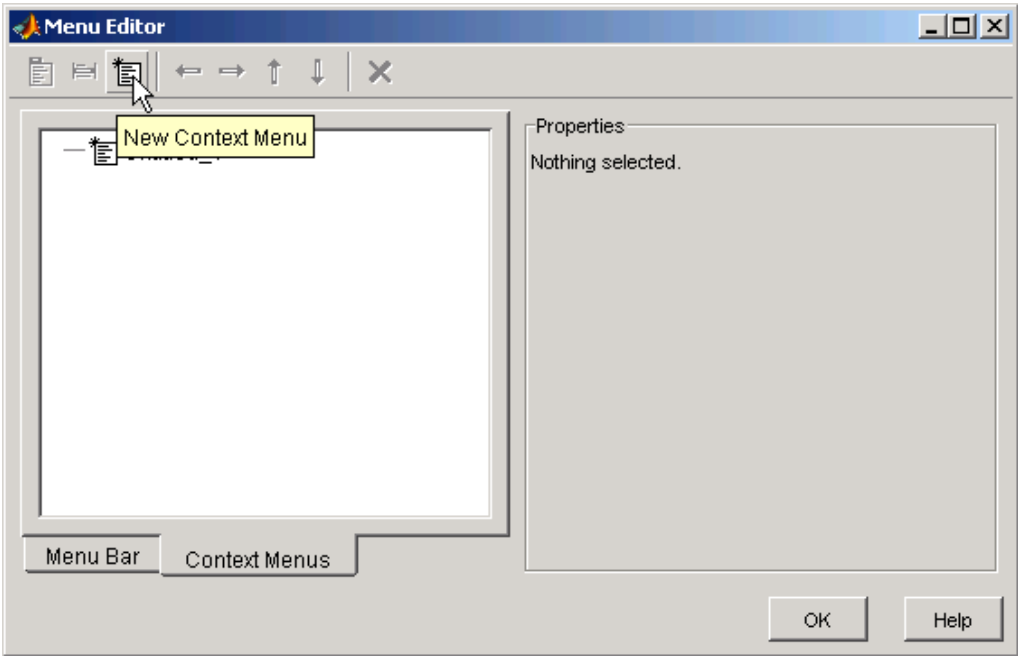


Şekil 5.63

İstenilirse bir menü öğesine Accelerator özelliği kullanılarak bir kısayol tuş kombinasyonu atanabilir. Buradaki herhangi bir menü öğesine tıklandığında çalıştırlacak callback'e gitmek için View butonuna tıklanılabilir. Gelen callback altında girilen komut satırları GUI uygulamasının çalışması durumunda o menü öğesi tıklandığında MATLAB tarafından koşturulacaktır.

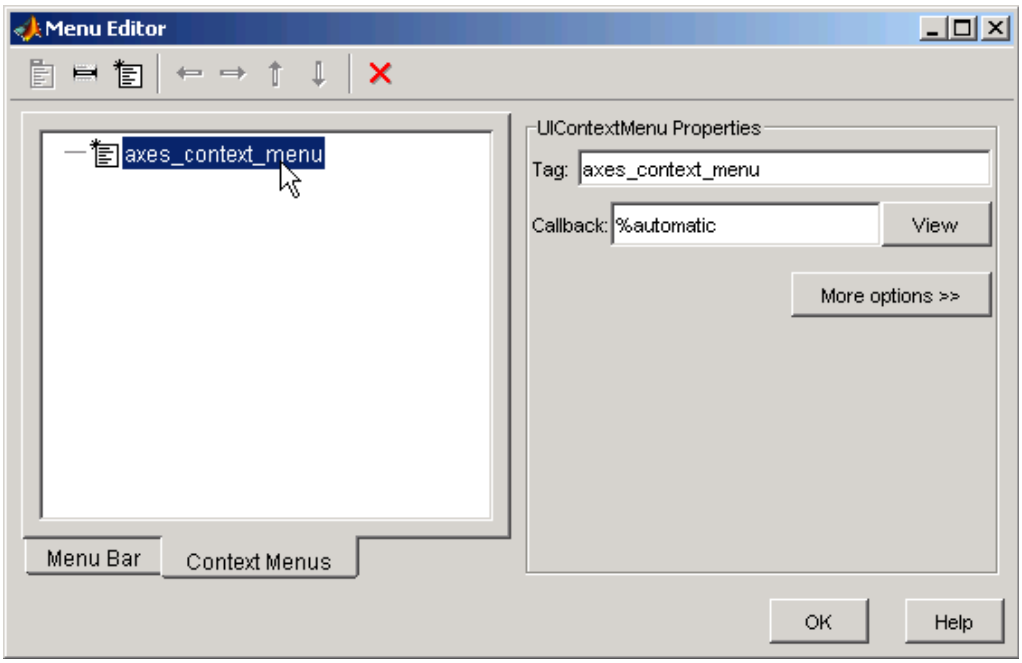
5.7.18.11.2 GUI Uygulamasına Context (İçerik) menüleri Eklemek

Bir GUI uygulamasında kullanıcıya sağlanılacak kolaylıklardan bir de bir nesne üzerinde farenin sağ tuşu ile tıklandığında gözüken context (içerik) menüleridir. Context menü oluşturma işlemleri bir önceki konu başlığı altında incelenen menülü ugyulamalar tasarlama adımlarına çok benzerlik gösterir.



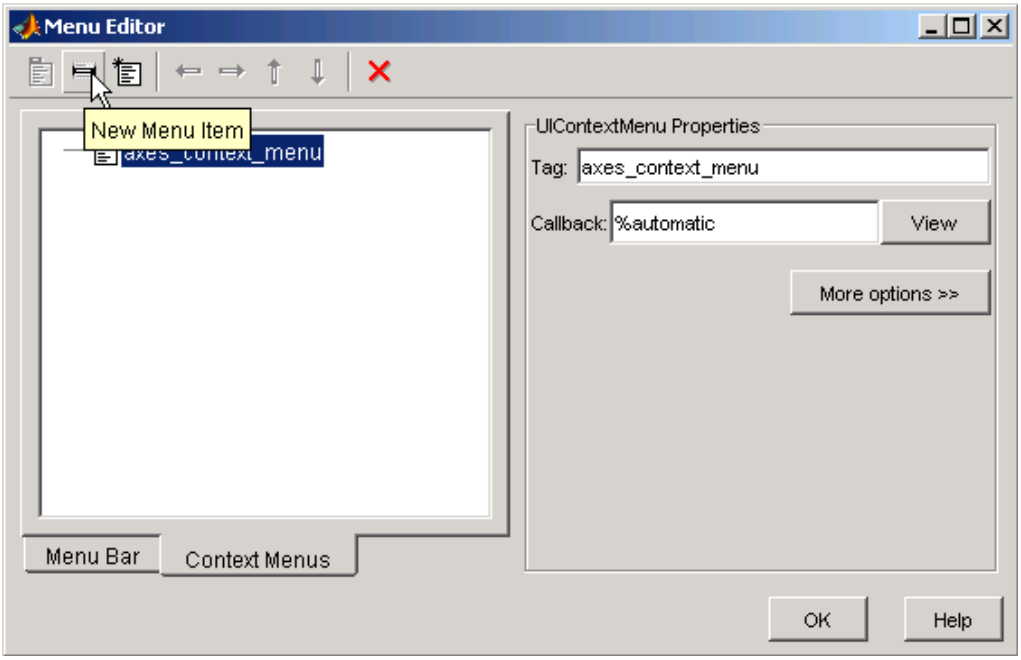
Şekil 5.64

Bir GUI uygulamasına içerik menüsü eklemek için öncelikle yapılması gereken Tools menüsünden Menu Editor komutu vermek ve karşımıza gelen ekrandan Context Menus sekmesine geçmektir. Daha sonra bu pencerenin üst tarafında bulunan New Context Menu butonunu kullanarak yeni bir içerik menüsü oluşturulur. Ardından eklenen menü listeden seçilen sağ tarafta yer alan özellikler kısmından yeni bir Tag ismi verilir. Bu isim kullanıcıya gözükmeyip sadece programlama amacıyla kullanılır.



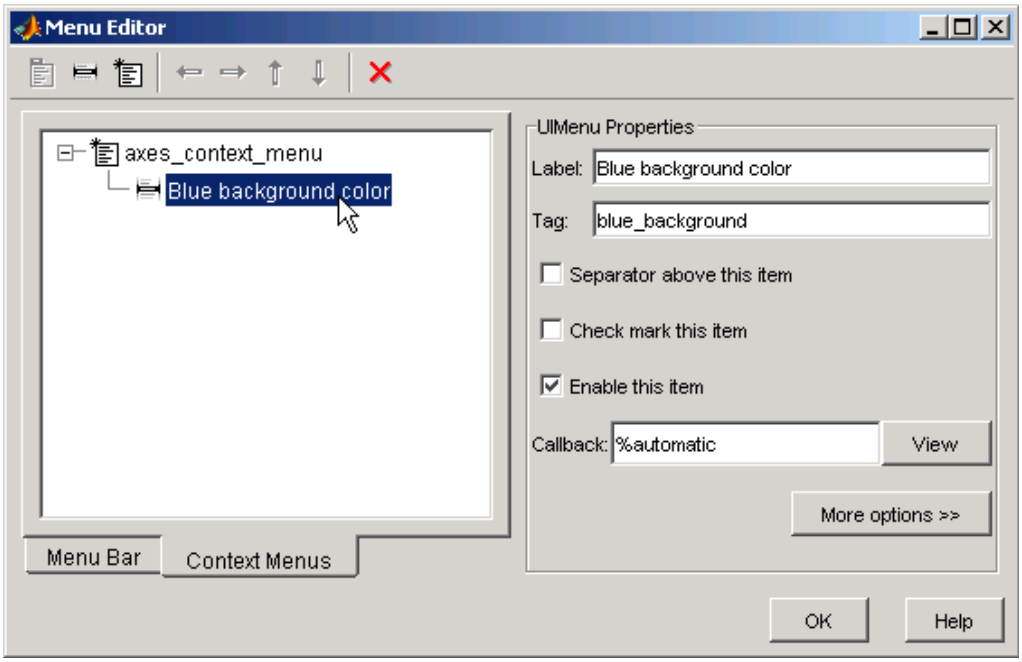
Şekil 5.65

Daha sonra bu içerik menüsüne yeni öğeler eklemek için New Menu Item butonuna basılır.



Şekil 5.66

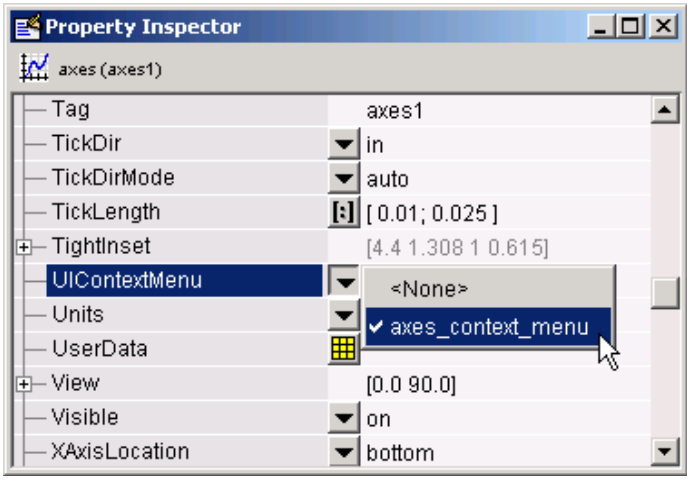
Eklenen bu menü öğesi önce listeden seçilir ve daha sonra da sağ tarafta bulunan özellikler panelinden hem Label hem de Tag bilgileri girilir.



Şekil 5.67

5.7.18.11.2.1 Context (İçerik) menüsünün Bir Nesne ile İlişkilendirilmesi

Yukarıda oluşturulan axes_context_menu isimli içerik menüsünün GUI uygulamasının çalıştırılması esnasında üzerinde farenin sol tuşu ile tıklandığında çıkacak nesne ile ilişkilendirilmesi gerekir. Bunun için ilgili nesne GUI tasarım alanında iken seçilir ve Property Inspector penceresinden UIContextMenu özelliği bu uygulamada oluşturulmuş olan içerik menüsü ile değiştirilir. Bu durum Şekil 5.68'de de görülmektedir.



Şekil 5.68

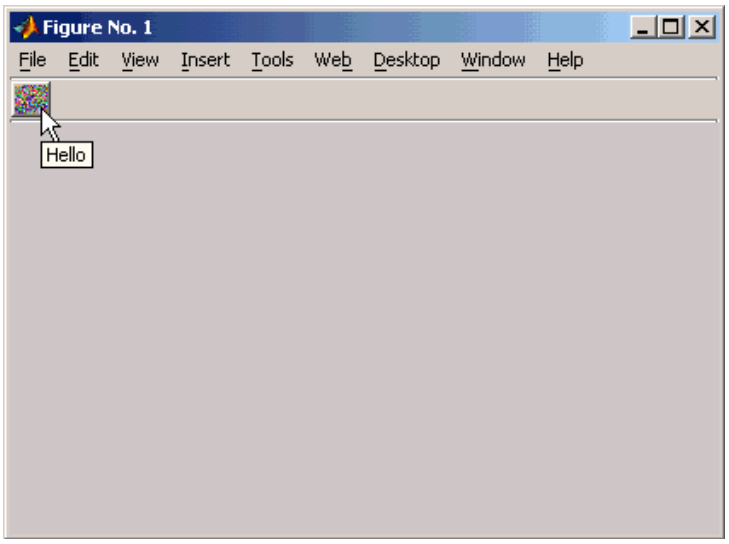
5.7.18.12 GUI Uygulamasına Araç Çubuğu Eklemek

Bir GUI uygulamasına araç çubuğu eklemek kullanıcının o uygulamayı kullanım kolaylığını arttırır. MATLAB GUI'de araç çubuğu ekleme işlemleri görsel olarak değil, komut satırları ile sağlanır. Bunun için GUI uygulamasının OpeningFcn callback bloğu kullnılır. Bu callbck GUI uygulamalarında hazırlık işlemlerinin yapılmasına imkân sağlar. GUI uygulaması daha açılmadan önce koşturulacak komut satırlarını bu blok içermektedir.

Bir GUI uygulamasına araç çubuğu eklemek için o uygulamanın OpeningFcn callback bloğuna şu komut satırları eklenmelidir.

```
ht = uitoolbar(hObject)
a(:,:,1) = rand(20);
a(:,:,2) = rand(20);
a(:,:,3) = rand(20);
htt = uitoggletool(ht,'CData',a,'TooltipString','Hello')
```

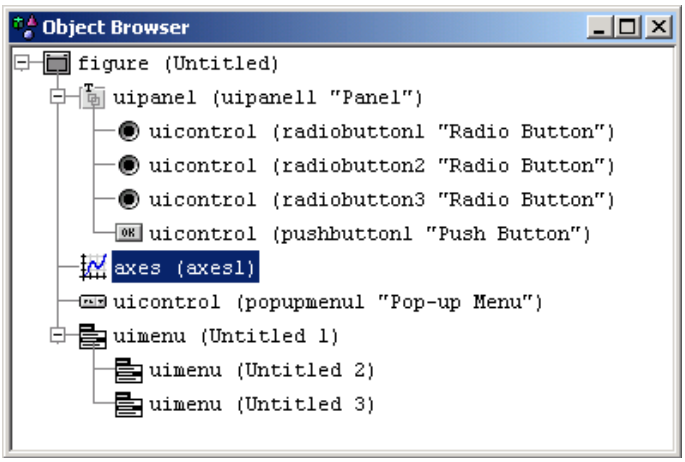
Yukarıdaki satırlar eklendikten sonra bu uygulama çalıştırıldığında Şekil 5.69'daki gibi bir GUI arayüzü ile karşılaşırız. Bu komut satırında uittolbar komutu ile yeni bir araç çubuğunun geçerli GUI figure alanına eklenmesi sağlanmaktadır. Daha sonra "a" isimli diziye eklenilecek bir buton üzerindeki renkleri göstermek üzere rasgele değerler atanmakta olup, uitoggletool komutu ile de bu araç çubuğuna ve rasgele renklerden oluşan bir görüntü ile yeni bir buton eklenmektedir. ToolTipString özelliği fare imleci bu buton üzerine geldiğinde gözükecek açıklama bilgisi için konulmuştur. Ancak, böyle bir açıklam koyma zorunluluğu da yoktur.



Şekil 5.69

5.7.18.13 Nesne Hiyerarşisinin Gösterilmesi

Tasarım esnasında programcı hangi nesneleri ve hangi isimlerle kullandığı genel hataları ile görmek isteyebilir. Bu gibi durumlar için GUIDE tasarımcıya Object Browser aracını sunar. Bu araç View menüsünden çalıştırılabilir. Bu araç Şekil 5.70'te de görülmektedir.



Şekil 5.70

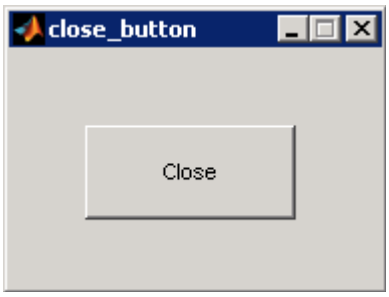
Object Browser aracı ile tasarımcı hangi nesneyi hangi Panel içinde veya hangi isim ile kullandığını kolaylıkla takip edebilir. Örneğin buradaki örnekte GUI figure içinde popup menu (açılır liste kutusu), axes (grafik çizimi), panel ile menü nesnelerinin eklendiği gözükmekte olup, panel içinde de üçü radio olmak üzere toplam dört adet buton kullanıldığı söylenilebilir.

5.7.19 GUI Nesnelerinin Programlanması

Burada her bir GUI nesnesinin programlama esnasında sıklıkla hangi özelliklerinin ve nasıl bir teknik ile kullanıldığı anlatılacaktır.

5.7.19.1 Push Button:

Şekil 5.71'deki gibi bir arayüz için butona tıklandığında GUI uygulamasını kapatacak bir Örnekle bir push buton kullanımını açıklayalım.



Şekil 5.71

Böyle bir GUI çalışma alanında fare işaretçisi push buton üzerinde iken View Callbacks/Callback komutunu verelim ve açılan callback bloğuna aşağıdaki komutları ekleyelim.

```
function pushbutton1_Callback(hObject, eventdata, handles)
display Goodbye
close(handles.figure1);
```

Böylece Close butonu tıklandığında GUI uygulaması sonlandırılmış olacaktır.

Başka bir örnekte de bir push butonun üzerine nasıl resim eklendiğini inceleyelim. Yukarıdaki örnekte Callback bloğuna önceki program satırları yerine aşağıdaki kodları yazalım.

```
a(:,:,1) = rand(16,64);
a(:,:,2) = rand(16,64);
a(:,:,3) = rand(16,64);
set(hObject,'CData',a)
```

Bu GUI uygulamasını Tools menüsünden Run komutunu kullanarak çalıştıralım ya da araç çubuğundan Run düğmesine basalım. Gelen GUI uygulamasında butona tıkladığımızda üzerine rasgele renklerden oluşan bir resimin atandığı görülecektir. Bu durum Şekil 5.72'de de görülmektedir.

Şekil 5.72

5.7.19.2 Toggle Buton:

Bir toggle buton çift durumlu çalışır. Bir kere tıklandığında basılı kalır. Bir daha tıklanırsa basılı kalmayıp eski konumuna geri döner. Böyle bir nesnede geçerli buton konumunu öğrenmek ve kullanabilmek için aşağıdaki komut satırları kullanılmalıdır.

```
function togglebutton1_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
```

```
if button_state == get(hObject,'Max')
% Toggle buton basıldığında yapılacak işlemler
...
elseif button_state == get(hObject,'Min')
% Toggle buton basılmadığı durumda yapılacak işlemler
...
end
```

5.7.19.3 Radio Buton:

Radio buton Buton Group nesnesi ile birlikte kullanıldığında daha etkili sonuçlar alınır. Ancak, kodlama yolu ile de radio butonların konumu kontrol edilebilir. Bir radio butonun basılıp basılmadığının kontrolü için şu kodlar kullanılabilir:

```
if (get(hObject,'Value') == get(hObject,'Max'))
% Radio buton basıldığında yapılacak işlemler
else
% Radio buton basılmadığı durumda yapılacak işlemler
end
```

5.7.19.4 Check Box:

Check Box nesnesinin konum kontrolü de radio butonlarınkine benzer şekildedir.

```
function checkbox1_Callback(hObject, eventdata, handles)
if (get(hObject,'Value') == get(hObject,'Max'))
% Checkbox nesnesi işaretlendiğinde yapılacak işlemler
else
% Checkbox nesnesi işaretlenmediği durumda yapılacak işlemler
end
```

5.7.19.5 Edit Text:

Bilgi girişi amacıyla sıklıkla kullanılan edit text nesnesinin string içerik bilgisini alamak için ilgili komut satırları şöyledir:

```
function edittext1_Callback(hObject, eventdata, handles)
user_string = get(hObject,'String');
% Callback bloğunun devamı ve diğer komutlar
```

Ancak, burada alınan bilgiler string tiptedir ve sayısal olarak kullanılamazlar. Sayısal olarak kullanabilmek için öncelikle edit box içerikleri sayısala dönüştürülmelidir. Daha sonra eğer ki hatalı bir giriş söz konusu ise hata kontrol deyimlerinin kullanılması ile bu durum giderilmelidir. Böyle bir durum için kullanılabilecek komut satırları aşağıda gösterilmiştir.

```
function edittext1_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
```

```
errordlg('Sayısal bir değer girilmelidir!..','Hatali Giris','modal')
return
end
% Callback bloğunun devamı ve diğer komutlar
```

Bu komut satırlarında ayrıca bir hata durumu oluştuğunda errordlg komutu ile kullanıcıya hatalı bir giriş yaptığı uyarı diyalog penceresi gösterilmektedir.

Edit Text nesnesinin callback satırları ancak kullanıcı baksa bir nesneye ya da gUI yüzeyine tıkladığı veya edit nesnesi içinde iken Enter tuşuna bastığı (çoklu giriş kutusu ise Ctrl + Enter tuş kombinasyonu kullanıldığı) zaman icra edilecektir. Aksi takdirde kullanıcı bir değer edit veya text nesnesine girerken bu callback satırları çalışmayacaktır.

5.7.19.6 Static Text:

Edit Text nesnesi ile benzer özelliklere sahiptir. Bu nesnenin edit text ten tek farkı kullanıcıdan bilgi girişi alınamamasıdır, sadece bilgi göstermek amaçlı kullanılır. Kodlama mantığı edit text nesnesi ile aynıdır.

5.7.19.7 Slider:

Bir kaydırıcı (slider) nesnesinin geçerli değerini program yoluyla okumak için gerekli komut satırları şöyle olmalıdır.

```
function slider1_Callback(hObject, eventdata, handles)
slider_value = get(hObject,'Value');
% Callback bloğunun devamı ve diğer komutlar
```

Bir kaydırıcı nesnesinin en küçük ve en büyük değerlerinin de ayarlanması bir programcı için gereklidir. Bunun için bu nesnenin Max ve Min özellikleri kullanılmalıdır.

5.7.19.8 List Box:

List Box nesnelerinin liste tipindeki string içeriğinin kullanılabilmesi için bu nesnelerin Value ve String özellikleri birlikte kullanılır. Kodlama mantığı şu şekilde olacaktır:

```
function listbox1_Callback(hObject, eventdata, handles)
index_selected = get(hObject,'Value');
list = get(hObject,'String');
item_selected = list{index_selected}; % Hücre dizisinden çevirme işlemi
% to string
```

5.7.19.9 Pop-Up Menu:

Popup menü nesnelerinde seçilen bir öğenin hangisi olduğu anlamak için bu nesnelerin Value özelliğinden yararlanılır. Kodlama örneği aşağıda gösterilmiştir.

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
```

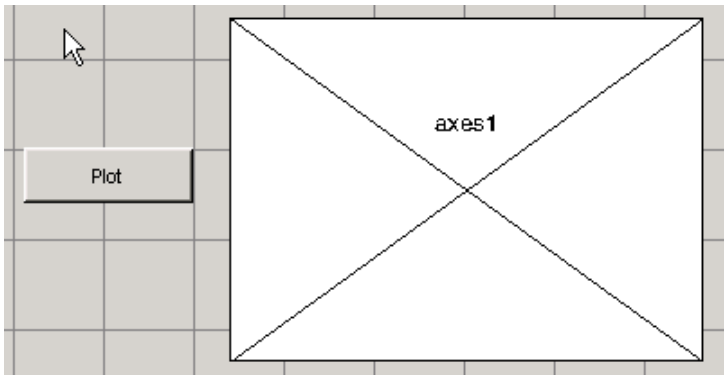
```
switch val
case 1
% Birinci öğe seçili iken yapılacak işlemler
case 2
% İkinci öğe seçili iken yapılacak işlemler
% Callback bloğunun devamı ve diğer komutlar
```

Eğer ki programcı seçilen öğenin stringini öğrenmek isterse şu komut satırları kullanılabilir:

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
string_list = get(hObject,'String');
selected_string = string_list{val}; % Hücre dizisinden çevirme işlemi
% to string
% Callback bloğunun devamı ve diğer komutlar
```

5.7.19.10 Axes:

Grafik çizimlerinin kullanıcıya sunulmasında sıklıkla kullanılan bir nesne olan axes için Şekil 5.73'teki gibi bir GUI arayüzüne sahip olunduğu düşünülmüş olsun.

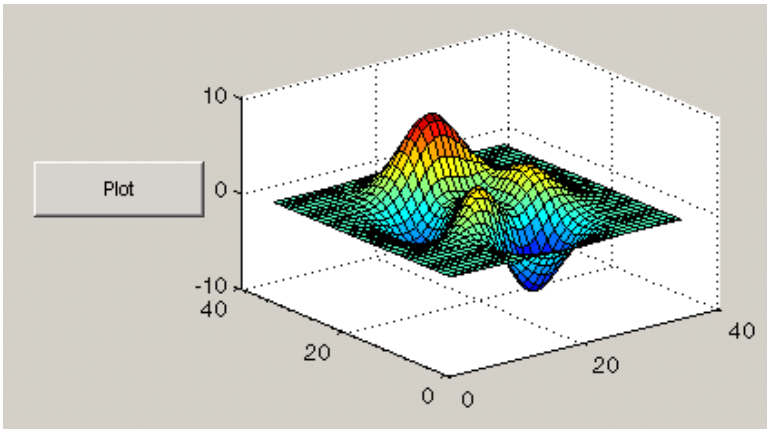


Şekil 5.73

Burada Plot butonunun callback bloğuna şu komut eklensin ve ardından Run komutu ile bu GUI uygulamasını çalıştırılsın.

```
surf(peaks(35));
```

Ekrana gelen GUI uygulama penceresinde Plot düğmesini tıkladığımızda Şekil 5.74'teki gibi bir ekran görüntüsü ile karşılaşılır.

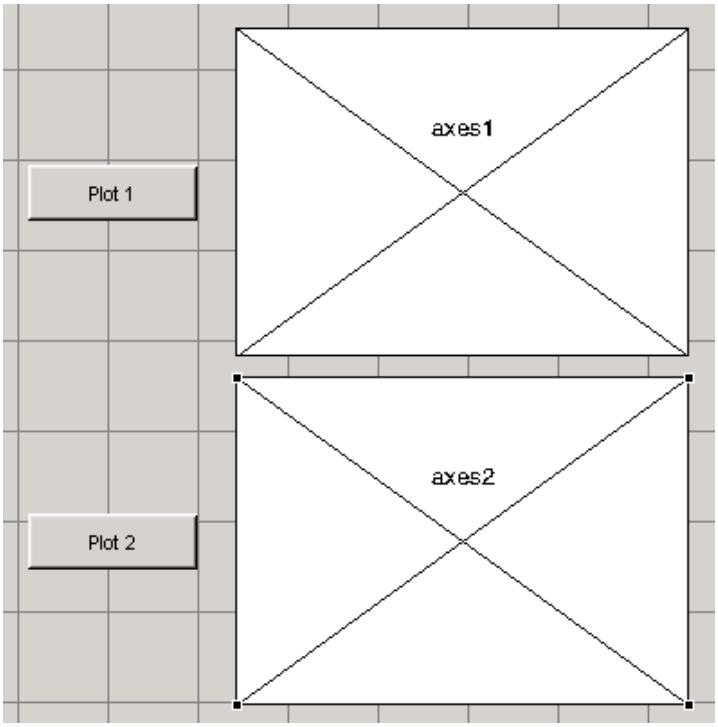


Şekil 5.74

Görüldüğü üzere tek axes nesnesi içeren GUI tasarımlarını programlamak kolay gözükmektedir. Ancak, eğer ki bir GUI arayüzü birden fazla axes nesnesine sahip ise bu takdirde hangi axes nesnesi o an aktif ise o nesne üzerinde çizimimiz gözükecektir. Aktif axes nesnesinin hangisi olacağı şu komutla belirtilir:

```
axes(handles.axes1)
```

Bir sonraki axes nesnesi ile ilgili örneğimizde çoklu axes nesnelerinin programlanmasına ait bilgilere yer verilmiştir. Bu örnek için GUI arayüzünün Şekil 5.75'teki gibi olduğu kabul edilsin. .



Şekil 5.75

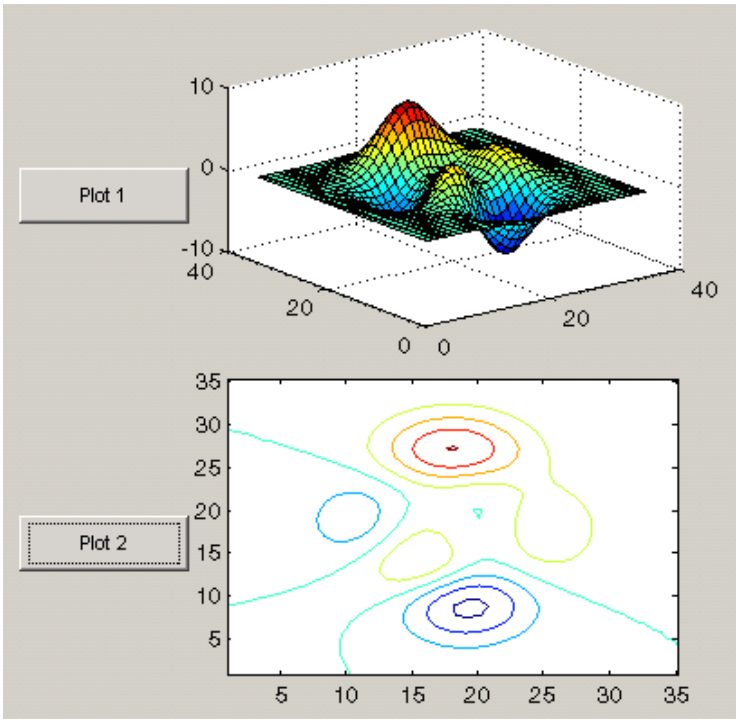
Bu uygulamada Plot 1 butonuna ait callback satırlarına

```
surf(handles.axes1, peaks(35));
```

Plot 2 butonuna ait callback satırlarına da

contour(handles.axes2, peaks(35));

komutları eklensin. Daha sonra da bu GUI uygulamsı çalıştırılsın. Gelen uygulama penceresinde ayrı ayrı Plot1 ve Plot 2 butonlarını tıklansın. Şekil 5.76'daki gibi bir görüntü ile karşılaşılır:



Şekil 5.76

Bu uygulama bize istenilen sonucu vermektedir. Ancak, aynı işlev farklı yöntem kullanılarak da gerçekleştirilebilir.

Plot 1 butonuna ait callback satırlarına önceden yazılan komut satırları yerine

axes(handles.axes1);
surf(peaks(35));

Plot 2 butonuna ait callback satırlarına önceden yazılan komut satırları yerine

axes(handles.axes2);
contour(peaks(35));

satırları yazılsın ve bu uygulama tekrar çalıştırılsın. Butonlar tıklandığında sonuç değişmeyecektir. Ancak, axes() komutunun kullanılması bir programcı olarak daha kolay ve sade bir kodlama anlamı taşıması bakımından genellikle tercih edilir.

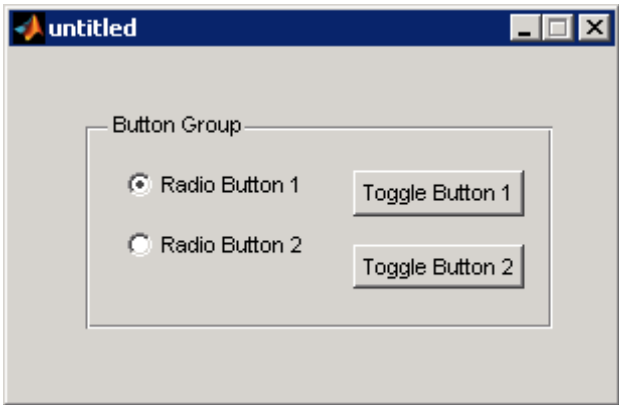
5.7.19.11 Panel:

Bu nesneler programlama amacı taşımayan yapıdadırlar. Panellerin kullanım amacı görsel olarak GUI uygulamasını zengin kılmak ve kullanıcıya yapacağı işlemler ile ilgili kullanımını kolaylaştırmaktır. Ancak, bir GUI uygulamasının koşturulması anında GUI figure alanının

boyutlarının değiştirilmesi söz konusu ise, bu takdirde panel nesnelerinin ResizeFcn metodu kullanılabilir.

5.7.19.12 Button Group:

Örnek olarak Şekil 5.77'de görülen bir GUI arayüzüne sahip olunduğu düşünülsün. Böyle



Şekil 5.77

bir durumda Buton Group nesnesi özelliğinden ötürü bu dört nesne toggle durumlu olarak (yani herhangi bir anda yalnizca biri seçili olabilir şekilde) çalışacaktır. Böyle bir uygulamada hangi buton seçili ise yaptırılmak istenilen programlama işlemleri şöyle olacaktır:

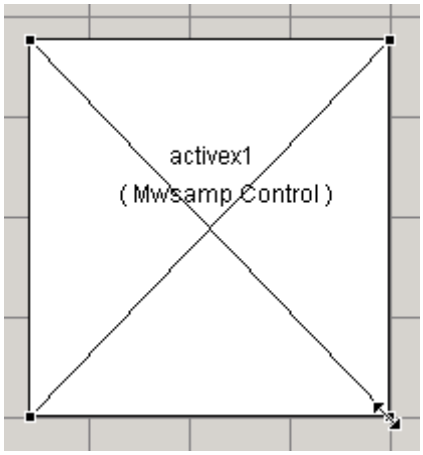
```
function uibuttongroup1_SelectionChangeFcn(hObject,...
eventdata,handles)
switch get(hObject,'Tag') % Seçili nesnenin Tag bilgisini alma
case 'radiobutton1'
% radiobutton1 seçili ise yapılacak işlemler
case 'radiobutton2'
% radiobutton2 seçili ise yapılacak işlemler
case 'togglebutton1'
% togglebutton1 seçili ise yapılacak işlemler
case 'togglebutton2'
% togglebutton2 seçili ise yapılacak işlemler
% Daha fazla sayıda buton varsa koşulların kontrolü böylece devam eder.
otherwise
% Hiçbir buton seçili değilse yapılacak işlemler
end
```

5.7.19.13 ActiveX Component:

Active X nesneleri ile MATLAB GUI uygulamalarının esnekliği arttırılmıştır. Böylece sadece GUI'nin kendi nesneleri ile tasarımcı sınırlı kalmamış olup, pek çok farklı nesneyi de alıp kullanabilir.

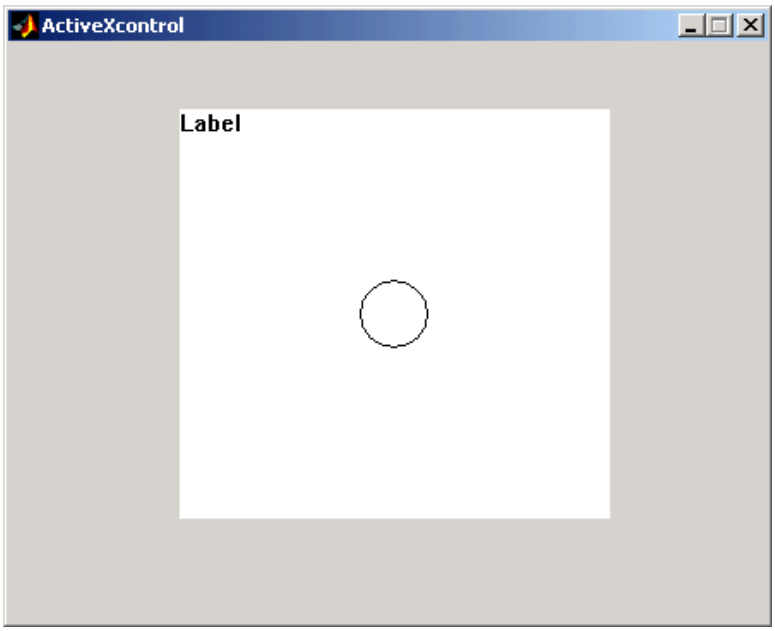
GUI yüzeyine önce bir ActiveX nesnesi ekleyelim ve karşımıza gelen component listesinden "Mwsamp Control" ü tıklayıp Create butonuna basalım. Ekelenen bu ActiveX

nesnesinin köşesinde boyutlarını değiştirmek ve büyütmek mümkündür. GUI çalışma alanındaki görüntü Şekil 5.78'deki gibi olacaktır:



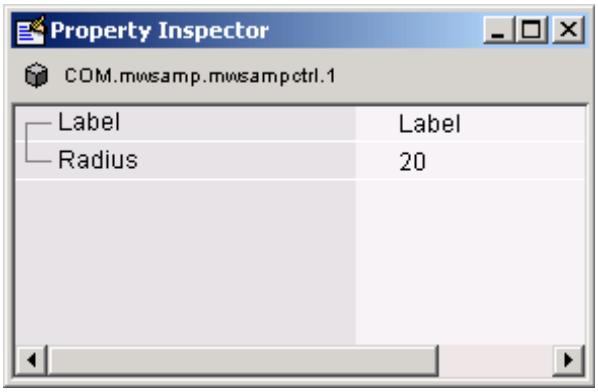
Şekil 5.78

Bu GUI uygulamasını çalıştırıldığında karşılaşılan arayüz ekranı Şekil 5.79'da gösterilmiştir.



Şekil 5.79

Tekrar tasarım ortamına geçilsin, Activex nesnesi seçilip Property Inspector penceresinden özelliklerine bakılsın.



Şekil 5.80

Bu ActiveX nesnesinin iki özelliği vardır (Bir ActiveX nesnesi ile ilgili daha kapsamlı bilgilere üreticinin hazırlanmış olduğu Kullanıcı Kılavuzundan ulaşılabilir.). Bunların işlevi şu şekildedir:

- Label özelliği, kullanıcıya ekranda sunulacak bilgidir.
- Radius özelliği, ekranda gözükecek dairenin yarıçap uzunluğudur.

Şimdi de bu ActiveX nesnesi ile ilgili programlama teknikleri açıklayalım. Öncelikle ActiveX nesnemizin Click Callback satırlarına aşağıdaki komutları ekleyelim.

```
hObject.radius = .9*hObject.radius;
hObject.label = ['Yaricap = ' num2str(hObject.radius)];
refresh(handles.figure1);
```

Böylece fare işaretçisi ile her ActiveX nesnesi üzerinde tıkladığımızda bu callback'teki komutlar icra edilecektir. Burada yapılan her tıklanmada daire yarıçapının artırılıyor olmasıdır. Ayrıca, nesnemizin Label özelliğinden faydalanılarak dairemizin yarıçap değeri ekrana yazdırılmaktadır. Burada unutulmaması gereken nokta bir ActiveX nesnesi ile ilgili özellikler değiştirildiği zaman mutlaka bunların o nesneye uygulanabilmesi ve uygulama arayüzünde yapılan değişikliklerin gözlenebilmesi için geçerli GUI figure yüzeyinin refresh edilmesi gerekliliğidir. Bunbun yanında burada Label özelliği string tipi verileri tuttuğu için yarıçap sayısal değerden num2str(komutu kullanılara string tipi bilgiye dönüştürülmekte ve ardından Label özelliğine atanmaktadır. Aksi takdirde programımızın çalıştırılması sırasında hata ile karşılaşılacaktır.

ActiveX nesnelerinin program yoluyla özelliklerinin değiştirilmesi gerektiğinde direk olarak ActiveX nesnelerinin özellikleri get metodu kullanılarak alınabilir. Bu durumla ilgili olarak şu komut satırları incelenebilir:

```
handles.activex1.radius = ...
get(hObject,'Value')*handles.default_radius;
handles.activex1.label = ...
['Radius = ' num2str(handles.activex1.radius)];
refresh(handles.figure1);
```

5.7.19.13.1 GUI Yüzeyine Eklenen Bir Activex Nesnesinin Tüm Metotları

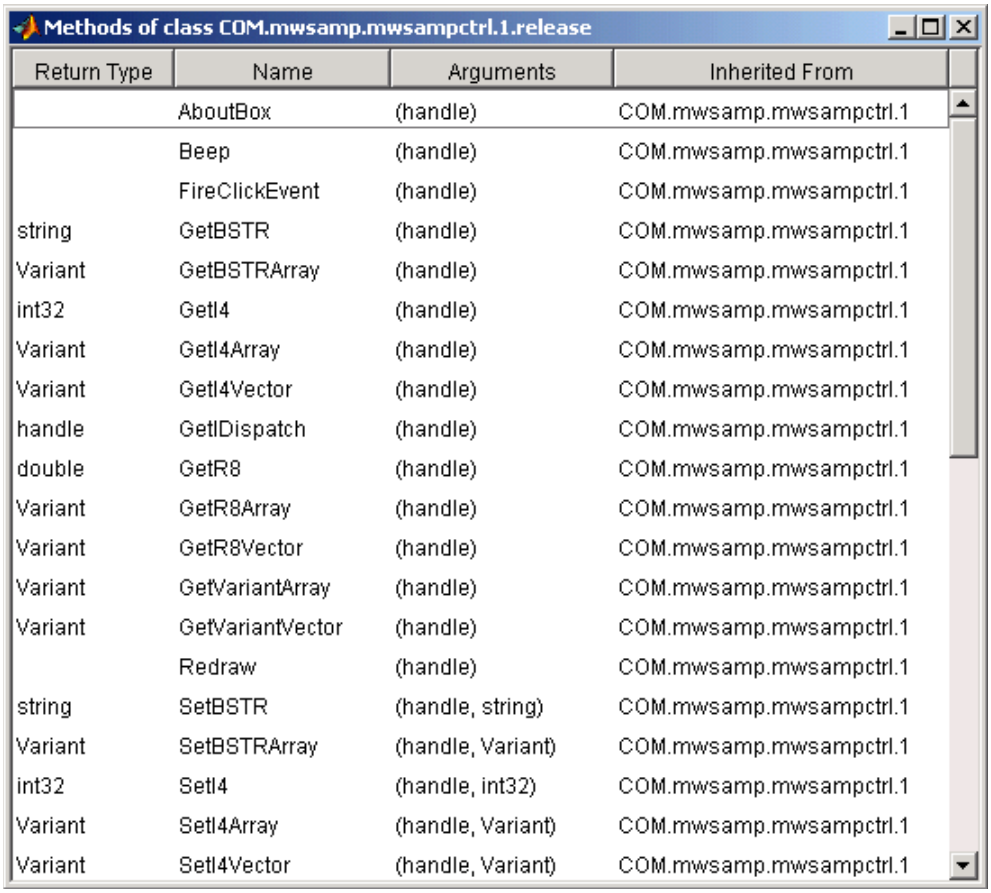
GUI yüzeyine eklenen bir activex nesnesinin tüm metotları

methodsview(hObject)

komutu ile öğrenilebilir. Örneğin bu komutu GUI çalışma alanına eklenen bir ActiveX nesnesinin Click Callback satırlarına eklenmiş olsun. Karşımıza aşağıdakine benzer bir ekran gelecektir. Ayrıca,

methods(hObject)

komutu ile bir nesnenin metotları MATLAB komut satırında da gösterilebilir. Komut çalıştırıldığında gözüken ekran görüntüsü Şekil 5.81'de verilmiştir.



Return Type	Name	Arguments	Inherited From
	AboutBox	(handle)	COM.mwsamp.mwsampctrl.1
	Beep	(handle)	COM.mwsamp.mwsampctrl.1
	FireClickEvent	(handle)	COM.mwsamp.mwsampctrl.1
string	GetBSTR	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetBSTRArray	(handle)	COM.mwsamp.mwsampctrl.1
int32	GetI4	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetI4Array	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetI4Vector	(handle)	COM.mwsamp.mwsampctrl.1
handle	GetIDispatch	(handle)	COM.mwsamp.mwsampctrl.1
double	GetR8	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetR8Array	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetR8Vector	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetVariantArray	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetVariantVector	(handle)	COM.mwsamp.mwsampctrl.1
	Redraw	(handle)	COM.mwsamp.mwsampctrl.1
string	SetBSTR	(handle, string)	COM.mwsamp.mwsampctrl.1
Variant	SetBSTRArray	(handle, Variant)	COM.mwsamp.mwsampctrl.1
int32	SetI4	(handle, int32)	COM.mwsamp.mwsampctrl.1
Variant	SetI4Array	(handle, Variant)	COM.mwsamp.mwsampctrl.1
Variant	SetI4Vector	(handle, Variant)	COM.mwsamp.mwsampctrl.1

Şekil 5.81

5.7.19.13.2 Activex Nesnesi İçeren Bir GUI Uygulamasının Compile Edilmesi

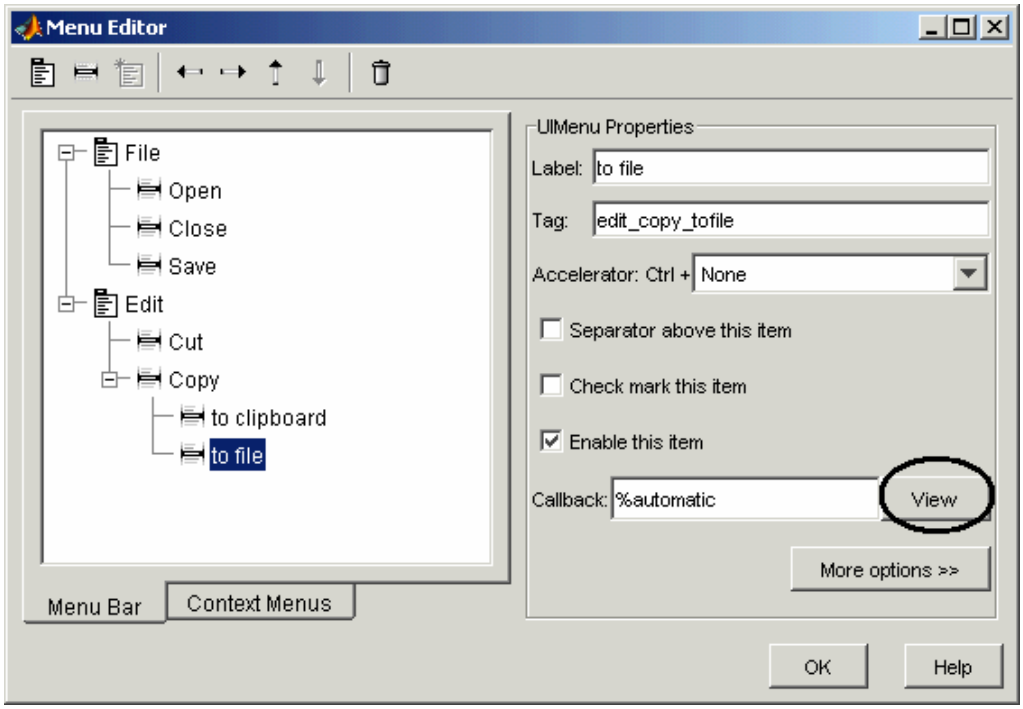
ActiveX içeren bir GUI uygulaması compile edileceği (derleneceği ve bağımsız bir çalıştırılabilir .exe uzantılı dosya edileceği) zaman aşağıdaki gibi bir yazım şekli kullanılmalıdır.

mcc -m mygui -a mygui_activex1

Her bir ActiveX nesnesi için "-a mygui_activex1" yazımı artırılmalıdır. Burada mygui adı ile kaydedilen bir GUI uygulamasının olduğu varsayılmış olup, GUI uygulamalarını derlemek üzere bu komutun uygulanacağı sistem de MATLAB Compiler aracının yüklü olduğu kabul edilmiştir.

5.7.19.14 Menü Öğeleri:

Tasarlanan bir GUI uygulamasına eklenen menünün herhangi bir öğesi tıklandığında istenilen komutların icra edilmesi istenebilir. Bunun için Menu Editor aracında ilgili öğe seçilir ve özellikler panelinden callback satırlarına gitmek için View butonu tıklanır. Bu durum Şekil 5.82'de de görülmektedir.



Şekil 5.82

5.8 GUI Uygulamalarında Callback Türleri

Callbackler önceki konularda da bahsedildiği üzere oluşan herhangi bir olaya bağlı olarak her nesne için ayrı ve olayın türüne göre icra edilen alt program parçalarıdır. Aşağıda sunulan tabloda kullnılan callbacklerin işlevi ve hangi nesneler ile birlikte kullanıldığı beliritilmiştir.

Bir m file dosya yapısı gereği bir GUI uygulaması tasarımında da aynı dosya ismini taşıyan fonksiyon ismi ile başlayan bir m function yapısına sahiptir. Ancak, giriş ve çıkış function varargout = DeneGui (varargin)
M function ilk satırında yer alan (yukarıda da ifade edilen deyimde de görüldüğü üzere) parametrelerinin dinamik olmasına dikkat edilmelidir. Yani, "varargin" deyimi ile giriş parametreleri hücre dizisi formatında birden fazla olabilir. Aynı, şekilde GUI uygulamasının kapatılacağı zaman aynı bir fonksiyon mantığı ile "varargout" dışarıya aktarılacak parametreler bu değişkene aktarılabilir.

Dışarıdan fonksiyon içerisine gönderilen giriş parametreleri ile ilgili bilinmesi gereken iki değişken vardır. Bunlar nargin ve varargin değişkenleridir.

- nargin değişkeni : Fonksiyona (ya da GUI uygulamasına) dışarıdan gönderilen toplam parametre sayısını tutar.

- vargin değişkeni: Fonksiyona gönderilen parametrelerin alınmasını sağlar. Hücre dizisi yapısında olduğundan parametrelerin alınması için dizi indislerinin "{" ve "}" işaretleri arasında yazılması gerekir.

Fonksiyonunun içinden dışarıya GUI uygulaması sonlandırılırken gönderilen çıkış parametreleri ile ilgili bilinmesi gereken iki değişkeni vardır. Bunlar nargout ve varargout değişkenleridir.

- nargout değişkeni: Fonksiyona (ya da GUI uygulamasına) dışarıdan gönderilen toplam parametre sayısını tutar.

- vargout değişkeni: Fonksiyondan dışarıya parametrelerin gönderilmesini sağlar. Hücre dizisi yapısında olduğundan parametrelerin bu değişkene atanması sırasında dizi indislerinin "{" ve "}" işaretleri arasında yazılması gerekir.

Tablo 5.1 GUI Uygulamalarında Callback Türleri

Callback Türü	Tetiklendiği Olay	Kullanıldığı Nesneler	
ButtonDownFcn	Fare göstergeci Figure veya bir nesnenin kenarlarından 5 piksel içerde olduğunda fare butonu tıklandığında oluşur. UI control için Enable özelliğinin true olması gerekmektedir.	axes	figure
		button group	panel
		user interface controls	
Callback	Nesnenin temel olayı. Örneğin bir push button tıklandığında ya da bir menü öğesi seçildiğinde oluşur.	context menu	menu
		user interface controls	
CloseRequestFcn	Figure kapanmadan önce çalıştırılır.	figure	
CreateFcn	Bir nesnenin create edilmesi anında oluşur. Nesne oluşturulduğunda initializing için kullnılabilir. Bu olay nesne create edilince ancak nesne GUI alanında gözükmeden önce icra edilir.	axes	figure
		button group	context menu
		menu	panel
		user interface controls	
DeleteFcn	Bir nesnenin kaldırılması anında oluşur. Herhangi bir nesne veya figure yok edilmeden önce temizlemeye dayalı operasyonlarda kullanılabilir.	axes	figure
		button group	context menu
		menu	panel
		user interface controls	
KeyPressFcn	Fgure veya bir nesne focus (aktif) olduğunda veya klavyeden herhangi bir tuşa basıldığında oluşur.	figure	
		user interface controls	

ResizeFcn	Panel, button group veya figure nesnenelerinin boyutları kuıllanıcı tarafından değiştiirldiğinde oluşur. Ayrıca, bu duurmun gerçekleşmesi için figure'e ait "Resize" özelliği "on" olmalıdır.	button group	figure
		panel	
SelectionChangeFcn	Button grouop nesnesi içinde kullanıcı farklı bir radio veya toggle butonu seçtiğinde bu olay tetiklenir.	button group	
WindowButtonDownFcn	Fare işaretçisi figure penceresi üzerinde iken farenin tuşuna basıldığında oluşur.	figure	
WindowButtonMotionFcn	Fare işaretçisi figure penceresi üzerinde ihareket ettirilirken oluşur.	figure	
WindowButtonUpFcn	Fare tuşu bırakıldığı zaman oluşur.	figure	
Not : User interface controls push button, slider, radio button, check box, editable textbox, static text , listbox ve toggle butonları içeren genel bir tanımlama ismidir. Bazen uicontrols olarak da isimlendirlebilirler.			

5.9 GUI Uygulamalarında Callbackler Arasında Ortak Veri Geçişini Sağlayan Yollar

GUI uygulamalarında bir değişken içeriği birden fazla callback içeriisnde kullanılmak istenebilir. Ya bir GUI callback functionun ürettiği değer başka bir function için giriş verisi olabilir. Konuyu daha geniş anlamıyla anlatılmak istenirse GUI uygulamalrında global değişken kullanım yolları öğretilmeye çalışılmaktadır. Bu durma benzer örnekler çoğaltılabilir. Bu durumu gerçekleştirmek üzere GUI uygulamalrında 6 farklı yöntem vardır.

5.9.1 Handles Yapı Değişkeni Kullanılarak Global Kullanımı

GUI uygulamalrında en sık kullanılan yöntemdir. Bu yiöntemde her callback functiona giriş parametre olarak gönderilen ve GUI uygulamalarında kullanıcı verilerinden ziyade GUI nesneleri ile ilgili handle değerlerini tutmaya yarayan “handles” yapı değişkeninden yararlanılır. Bu yapıyı çok kolaydır. Örnek olarak sistem_cikis_sayisi isminde bir değişkeni her callback içerisinden ortak olarak kullanmak isteğimiz varsayılsın ve içeriği 4 yapılsın. Daha sonra da bu değişkeni, handles yapısı içerisine koymak istediğimizi düşünelim.
Bu işlem için aşağıdaki komut satırları yazılmalıdır.

```
sistem_cikis_sayisi = 4;
Handles . sistem_cikis_sayisi = sistem_cikis_sayisi;
guidata (hObject, handles);
```

burada 1. satır ile bu deyimlerin kullanıldığı callback içinde lokal bir "sistem_cikis_sayisi" isminde değişken oluşturlmuş ve içeriği 4 olarak atanmıştır. Ancak, loakl bu değişkenin değeri ve kendisinini varlığı callback fonksiyonunun icrası tamamlandıkrtan sonra kaybolacaktır. Tekrar aynı callback'e gelindiğinde de önceki değer silinmiş olacaktır. Burada dikkat edilmesi gereken standart bir kalıp şeklinde kullanılan guidata (hObject, handles); komut satırının varlığıdır. Bu satır ile handles yapı değişkeni yeni değerlerle birlikte callback dışına çıkmadan güncellenmekte ve callback dışına çıkıldığında da veya başak acalback çağrıldığında bu yapı içerisinden kullanılmasına imkân verilmektedir.

Örneğin başka bir callback içeirisinde kurulacak bir for döngüsünün toplam sayma adedi bu değişken kadar olmak üzere aşağıda yer alan komut satırları yazılabilir.

```
For i=1:1:handles. sistem_cikis_sayisi
        % döngü içieirinde icra edilmesi düşünülen kom ut satırları
End
```

5.9.2 Global Değişken Tanımlama Deyimi

Herhangi bir callback başında global deyimi ile bir değişken ismi girildiği takdirde eğer daha önce böyle bir değişken yok ise oluşturulur ve başlangıç değeri otomatik olarak 0 (sıfır) değeri atanır. Ancak, eğer daha önceden bu callback icra edilmiş ve global deyimöi ile tranımlanan değişken bellekte bir yere sahip ve değeri var ise bu değerininh bir sonraki callback çağrısında da devam ettirecektir. Bu durumun kullanıma örnek komut satırları aşağıda gösterilmiştir.

```
Function edit1_callback( … )
global sayac;
sayac=sayac+1;
% diğer icra edilecek komutlar
if isequal(sayac,5)
sayac=0;
end
```

Yukarıdaki örnekte sayac değişkeni 0 dahil toplam 6 farklı değer almakta ve 5 olduğunda değeri tekrar sıfırlanmaktadır.

5.9.3 GUI Alanında Visible Ve Enable Özellikleri Off Yapılmış Nesne Kullanılması

Tasarımcı isterse kullanım kolaylığı sağlaması bakımından GUI alanında tasarım aşamasında görünen, fakat GUI icrası sırasında kullanıcılar tarafından görülemeyen ve kontrol edilemeyen ekstra bir nesne kullanabilir. Bunu yapmak için tek yapması gereken bu nesnenin "Enable" ve "Visible" özelliklerinin "off" yapılmasıdır. Bu nesneye ait "Value" veya "String" değerleri kullanılarak global değer kullanımına yönelik programlama yapılabilir.

5.9.4 Load Ve Save Deyimlerinin Kullanılması

Programcı herhangi bir callback içinde kullandığı değişkenleri o hali ile bir mat dosyasına kaydedebilir ve ileride ya da gerekli görüldüğü takdirde tekrar kullanmak üzere çağırabilir. Örnek olarak aşağıdaki komut satırları ele alınabilir.

```
a=5;
b=40;
save sayilar_a_ve_b;
```

Bu komutlar sonucunda a ve b değişkenleri "sayilar_a_ve_b.mat" isimli bir Matlab veri saklama dosyasına kaydedilir. İleride kullanılacağı zaman yapılması gereken çok kolaydır.

```
load sayilar_a_ve_b;
c=a*b;
```

Yukarıdaki komutların icrası ile a ve b değerleri herhangi bir an veya o an için workspace alanına veya callback'in geçici bellek alanına yüklenir. Ancak, değerleri (5 ve 40 sayıları) aynen korunur. Böylece c değişkeninin değeri 200 olacaktır.

5.9.5 Nesnelerin Userdata Özelliğini Kullanmak

Global değişken kullanmanın bir başka ve kolay yolu da her bir GUI nesnesine ait "UserData" özelliğinin kullanılmasıdır. Bu değişkenin içeriği string tipi verileri tutatbildiği gib sayısal tip verileri de direk olarak gönderme ve kullanma imkânı programcıya sunulur. Yani, aralrda num2str veya str2num komutlarının kullanılmasına gerek kalmaz.
Örneğin edit1 nesnesinin Userdata alanına Checkbox1 isimli nesnenin "Value" değerini koyan komut satırları yazılmış olsun.

```
durum = get ( handles.checkbox1 , 'Value' ) ;
set ( handles.edit1 , 'UserData' , durum ) ;
```

Örneğin Edit1'in UserData alanında bulunan bu değeri kendisinin String özelliğine atanmış olsun. Normalde bir string değer sayısal bir değişkene atandığında hata oluşur ve burada da hata oluşması beklenir Çünkü, bilindiği gibi "checkbox" nesnelerinin "Value" özellikleri sayısal tiptedir. Ancak, "edit" nesnelerinin "String" özelikleri sayısal değil, string tipindedirler.

```
userdata_icerigi = get ( handles.edit1,'UserData' );
set ( handles.edit1 , 'String' , userdata_icerigi );
```

Görüldüğü gibi bu komutların icrası sonucu herhangi bir hata oluşmamıştır (sayısal ve string değer dönüşümleri arasında). Yani, UserData özelliği değişkenler arasında uyumluluk sağlamakta ve kendisi otomatik olarak tip dönüşümlerini gerçekleştirmektedir.

5.9.6 Uygulama Datası Yöntemi

En gelişmiş, ancak kullanımı biraz zahmetli olan yöntemdir. Bu yöntem ile herhangi bir GUI nesnesinin varolan özelliklerine sanki handles yapısına yeni bir değişken ekler gibi yeni bir değişken eklenebilirç Eklelnen nesne genellikle figure olduğu için yöntem ismini buradan almaktadır. Örnek olarak toplam_boyut alanının bir herhangi bir nesne callback'inde icrası

sonucu bahsedilen değiken bu callback'e ait nesnenin boyut isminde yeni bir özelliği olarak tanımlanmış olsun. Kullanılacak komut satırları aşağıda verilmiştir.

```
toplam_boyut = 15 ;
setappdata ( hObject, 'boyut' , toplam_boyut ) ;
```

Herhangi bir anda bir nesnenin uygulama datasının alınması gerektiğinde de aşağıdaki komut satırları icra edilmelidir. Örnek olarak yukarıda içeriisnde application data saklanılan nesne edit4 isimli bir nesne olsun. Buna göre ilgili komutlar şu şekilde yazılmalıdır:

```
boyut_degiskeni = getappdata(handles.edit4, 'boyut');
```

5.10 GUI Uygulamalarında Temizleme Komutları

Herhangi bir GUI uygulamsında figure penceresi veya, komut satırı alanı ya da axes (grafik çzim) nesnesinin içeriğinin temizlenmesi gerekebilir. Ayrıca herhangi bir değişkenin workspace den atılarak bellketen temizlenmesi de gerekebilir. Bu durumlar için şu komutlar kullanılmalıdır.

- Figure alanını temizlemek için clf komutu
- Axes nesnesindeki çizimin temizlenmesi için cla komutu,
- Komut satır ekranının temizlenmesi için clc komutu,
- Worspace alanındaki tüm değişikenleri silmek ve bellekten temizlemek için clear all veya kısaca clear komutu
- Workspace alanından örneğin adet_no isimli değişken kaldırılmak istenirse clear adet_no komutu

kullanılmalıdır.

5.11 GUI Uygulamalarında Kullanılan Standart Handle Değişkenleri

GUI uyugulamalrında birden fazla nesne veyua figure alanı ile çalışıldığı düşünülürse bazen yanlış veya istenmeyen nesnelkere kontrollerin kaydığı görülebilier. Bu gibi durumlardan sakınmak için aktif axes veya grafik nesnesi gibi bazı belirli nesnenelr için özel olarak handle numarasını tutmak amacıyla tanımlanmış değişkenler Mtalab tarafından GUI tasarımcılarına sunulmuştur. Ayrıca ince bir bilgi olmasına karşılık bilmekte yarar var. Bir GUI uygulamasına ait figure nesnesinin handle numarası varsayılan olarak GUI handles yapısının içinde yer alan "output" isimli değişkende de tutulmaktadır.

GUI uygulamalarında handle numaralarını öğrenmek amacıyla sıklıkla kullanılan standart değişkenler şunlardır:

gcf : Geçerli figure nesnesinin handle numarasını verir.
gca : Geçerli axes (grafik çizim) nesnesinin handle numrasını verir.
gco : nesne_handle=gco(fig_handle) kullanımı ile GUI alanında en sok tıklanmış ya da en son aktif olan nesnenin handle numarasını verir.
gcbf : figure_bo = gcbf; kullanımı ile hangi figure nesnesine ait bir callback (veya figurein icerdiği bir nesne callback in çalışıyor olabilir) bu figure nesnesine ait handle numarası döner.
gcbo : Aktif olarak hangi nesnenin callback i çalışıyor ise o nesneye ait handle numarası

döner. Nesne_handle = gcbo olarak kullanılabileceği gibi
[nesne_handle, figure_handle] = gcbo şeklinde kullanım ile aktif nesnenin bulunduğu figure handle numrası da elde edilebilir.

5.12 Herhangi Bir GUI Uygulamasının Sonlandırılması

Bir GUI uygulamasını sonladırmak için kullanılacak komut "close" dur. İstenirse close(gcf) ile aktif GUI uygulamsı sonlandırılmak yerine başka açık bir fifure de sonlandırılabilir. Bunun için örneğin figure no su fig_no isimli bir değişkende tutulan figure nesnesinin kapatılması için close(fig_no) komutu icra edilmelidir.

5.13 Yeni Bir Figure Oluşturma Komutu

Bu amaçla "figure" komutu kullanılır. Bu komut eğer bir değişkene atanırsa bu takdirde açılan yeni figure ekranının handle numarası da saklanmış ve korunmuş olacaktır. Örnek olarak fig_handle_no = figure; şeklinde komut yazımı verilebilir.

5.14 MATLAB GUI Uygulamalarında Etkileşim Kutuları Yönetimi

MATLAB GUI ile görsel tasarım hem programcının tasarımı kolay kılmakta, hem de kullanıcı yapacağı işleri görerek ve kolaylıkla birkaç tıklama ile dahi gerçekleştirebilmektedir. GUI uygulamalarının kullanıcılara sunduğu kolaylıklardan biri de kullnıcıların yapılan veya yapılacak işler ilgili uyarılması veya bilgilendirilmesi amacıyla kullanılan etkileşim kutularının kullanılmasıdır. Örneğin bir hata oluştuğunda bu durumu en temel anlamı ile GUI arayüzünde bir nesne kullanarak ve bu nesneye ait bir özelliği (örneğin renk gibi) değiştirerek kullanıcı bilgilendirilebeilir. Ancak, bu yöntem etkin bir uyarma aracı olarak düşünülemez. Bunun yerine az önce bahsedildiği gibi örneğin bir hata penceresi ile hem kullanıcının başka işlem yapması engellenebilir (yani, uygulamanın arka plandaki arayüzü uyarı anında kilitlenebilir), hem de daha gerçekçi ve sade bir diyalog pencere görüntüsü kullanıcının Windows benzeri GUI tabanlı ortamlarda alışkanlıkların dışında tasarım ekranları ile karşlaşmamış olur. Ayrıca, etkileşim kutularının kullanımı ile programıcının da GUI arayüzünde herhangi bir tasarım değişikliğine gitmesine fırsat verilmez, dolayısıyla daha etkin GUI tabanlı uygulama geliştirme imkânı sağlanmış olmaktadır.

Matlab ile hazırlanan GUI uygulamalrında kullanılabilecek etkileşim kutuları Tablo 5.2'de gösterilmiştir.

Tablo 5.2 Matlab ile Tasarlanan GUI Uygulamalarında Kullanılabilecek Etkileşim Kutusu Türleri

Etkileşim Kutusu Türü	Açıklama
errordlg	Hata Penceresi
helpdlg	Yardım Penceresi
inputdlg	Veri Giriş Penceresi
listdlg	Liste Görünüm Penceresi
printdlg	Yazdırma Penceresi
questdlg	Sorgu Penceresi
uigetdir	Klasör Yolu Seçme Penceresi
uigetfile	Dosya Açma Penceresi
uiputfile	Dosya Kaydetme Penceresi

uisetcolor	Renk Seçim Penceresi
uisetfont	Font Seçim Penceresi
warndlg	Uyarı Penceresi
waitbar	Yükleme Çubuğu
pagesetupdlg	Sayfa Yapısı Penceresi
msgbox	Mesaj Kutusu
printpreview	Sayfa Önizleme Penceresi

Aşağıda GUI uygulamalarında çeşitli etkileşim kutularının ne amaçla ve nasıl kullanılacağı ayrıntılı olarak anlatılmıştır.

5.14.1 Hata Penceresi (Error Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında oluşan bir hata hakkında bilgilendirilmesi amacıyla kullanılır. Kullanımı şu şekildedir:

errordlg('Yanlış değer girildi.','Hata Penceresi','modal')

Hata diyalog penceresinin ekran görüntüsü Şekil 5.83'te gösterilmiştir.

Şekil 5.83

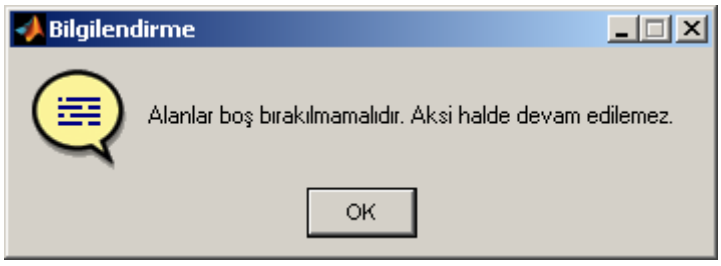
Yukarıdaki kullanımda modal seçeneğinin kullanılması seçimliktir, yani kullanılmasa da olur. Ancak, bu seçenek ile hata penceresine cevap verilmeden uygulamaya ait ilmelerin yapılması engellenmiş olmaktadır. "modal" seçeneği yerine "non-modal" kullanılarak bu özelliğin iptal edilmesi sağlanılabilir.

5.14.2 Yardım Penceresi (Help Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında eksik veya yanlış bilgi girme ya da herhangi bir konuda bilgilendirilme amacıyla kullanılır. Kullanımı şu şekildedir:

helpdlg('Alanlar boş bırakılmamalıdır. Aksi halde devam edilemez.','Yardım Penceresi','modal')

Yardım diyalog penceresinin ekran görüntüsü Şekil 5.84'de gösterilmiştir.



Şekil 5.84

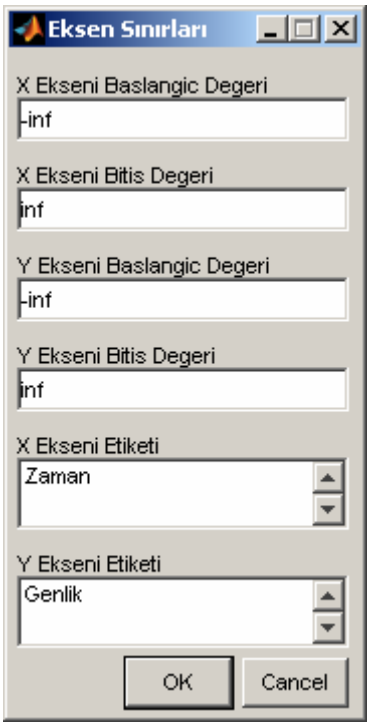
5.14.3 Veri Giriş Penceresi (Input Dialog) :

Bu diyalog penceresi, kullanıcılardan bir veya birden fazla değerin aynı anda alınması amacıyla kullanılır. Kullanımı şu şekildedir:

```
yazi_ifadeleri = { 'X Ekseni Baslangic Degeri', 'X Ekseni Bitis Degeri', ...
    'Y Ekseni Baslangic Degeri', 'Y Ekseni Bitis Degeri' ,'X Ekseni Etiketi', 'Y Ekseni Etiketi'
};
baslik = 'Eksen Sınırları';
satir_adedi = [1 1 1 1 2 2];
varsayilan_degerler = { '-inf', 'inf', '-inf', 'inf' , 'Zaman', 'Genlik'};
diyalog_donus_degeri = inputdlg (yazi_ifadeleri, baslik, satir_adedi, varsayilan_degerler)
```

bu parametrelerden “satir_adedi” parametresi açılacak diyalog penceresinde yer alan her bir text kutusunun sahip olacağı toplam satır adedini gosterir. Örnek kullanımda ekran görüntüsünde X ve Y eksenlerine ait etiket değer alanlarının 2 satırdan oluştuğu görülmektedir. Bu parametre her bir text kutusu için tanımalandığından matris şeklinde bir yapıya sahiptir. Diğer bir parametre olan “varsayilan_degerler” diyalog penceresi ilk görüntülendiğin text kutularının içinde olması gereken başlangıç değerlerini gösterir. “baslik” parametresi diyalog pencerensin caption değeri için ve yazi_iafadeleri her bir text kutusu ile ilgili bilgileri ekranda göstermek için kullanılmıştır. “yazi_ifadeleri” ve “varsayilan_degerler” isimli parametrelerin hücre dizisi şeklinde tanımlanaması gerektiğine dikkat edilmelidir (“ { ” ve “ } ” simgeleri hücre dizisi tanımlamalarında kullanılır.).

Yukarıda verilen örnek kullanıma ait veri giriş diyalog penceresi için ekran görüntüsü Şekil 5.85’te gösterilmiştir.



Şekil 5.85

Bu diyalog penceresinden geri dönüş değeri kaç tane bilgi giriş kutusu varsa, bu uzunlukta dizi şeklinde döner. Örnek olarak "y_ekseni_bitis_degeri" nin elde etmek için

y_ekseni_bitis_degeri = diyalog_donus_degeri (4)

şeklinde kullanım söz konusudur. Ancak, kullanıcı bu diyalog kutusunu "Kapat" (X simgesi) ya da "Cancel" butonlarından biri ile gönderirse boş bir hücre dizisi döner. . Eğer böyle bir durum kontrol edilecekse "isempty" fonksiyonu kullanılabilir. Bu fonksiyon bir değişken içeriği boş ise "true", dolu ise "false" üretir. "isempty(degisken_ismi)" şeklinde kullanıma sahiptir.

5.14.4 Liste Görünüm Penceresi (List Dialog) :

Bu diyalog penceresi ile kullanıcıların karşısına gelen bir listeden bir veya birden fazla liste elemanı seçmesi sağlanılşabilir. Uygulamada genellikle dosya veya dizinlerin listelenmesi ve seçilmesi amacıyla kullanılır. Örnek bir kullanımı şu şekildedir:

```
dosyalar = dir;                                % aktif dizin yolu üzer. yer alan dosyal. list. alma
 dosya_isimleri = {dosyalar.name};             % dosya_isimleri matrisine dosya isiml. atanması
 [s,v] = listdlg('PromptString',Bir dosya seçiniz : ',...      % liste diyalog pencerensin görünt.
            'SelectionMode','single',...
            'ListString',dosya_isimleri)
```

Dosyalar değişkenine "dir" fonksiyonu vaıtası ile her bir elemanı yapı dizisi olan bir dosya listesi gelir. Her bir yapı dizisinde

- name : her bir dosyanın ismini tutmak için,
- date : her bir dosyanın geçerli tarihini tutmak için,
- bytes : her bir dosyanın boyut bilgisini tutmak için,
- isdir : her bir dosyanın bir klasör olup olamdığı tutulur. Değer "true" ise bu öğe bir dizindir.

alanları bulunmaktadır. Örneğin 5. öğenin tarih bilgisi öğrenilmek istenirse
dosyalar(5).date
komut yapısı kullanılmalıdır.

Örnek kullanımda seçim modu olarak "single" seçeneği kullanılmıştır. Yani, liste elemanlarından sadece bir tanesi seçilebilir. Eğer, çoklu seçim isteniliyorsa "multiple" seçeneği kullanılabilir. Çoklu seçimde kullanıcı birden fazla öğeyi seçebilmek için öğrelrin seçimi sırasında klavyenin "Ctrl" tuşunu basılı tutmalıdır.

Liste görünüm diyalog penceresinden geri dönüş değeri olarak sadece seçilen liste elemanlarının numara bilgisi sütun vektor seklinde (burada örnek kullanım için) s değişkenine atanır. Bu atama sırasında v değişkenine daima 1 değeri atanır. Örneğin 15, 25 ve 36 numaralı liste öğreleri seçildiği düşünülürse

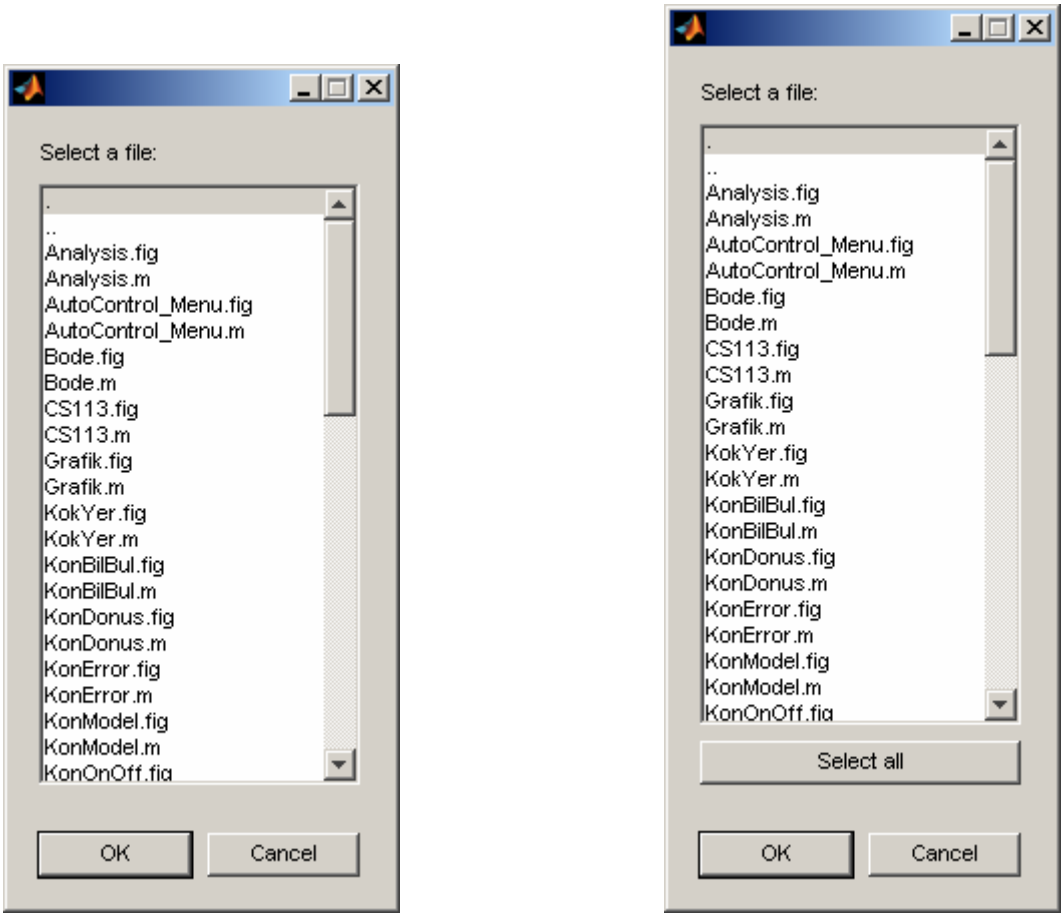
- s değişkeninin içeriği [15 25 36]
- v değişkeninin içeriği 1

olacaktır. Eğer kullanıcı bu diyalog penceresini “Cancel” veya “Kapat” (X simgesi) düğmelerini kullanarak kapatırsa geri dönüş değerleri

- s değişkeninin içeriği [] (yani boş matris)
- v değişkeninin içeriği 0

şeklinde olacaktır. S değişkeninin boş olup olmama durumu “isempty” fonksiyonu ile kontrol edilebilir. Örnek olarak isempty(s) komutu sonucu true (veya 1 değerinde) ise s değişkeninin içeriği boş demektir.

Liste görünüm diyalog penceresine ait “single” ve “multiple” modları için ekran görüntüleri Şekil 5.86’da gösterilmiştir.



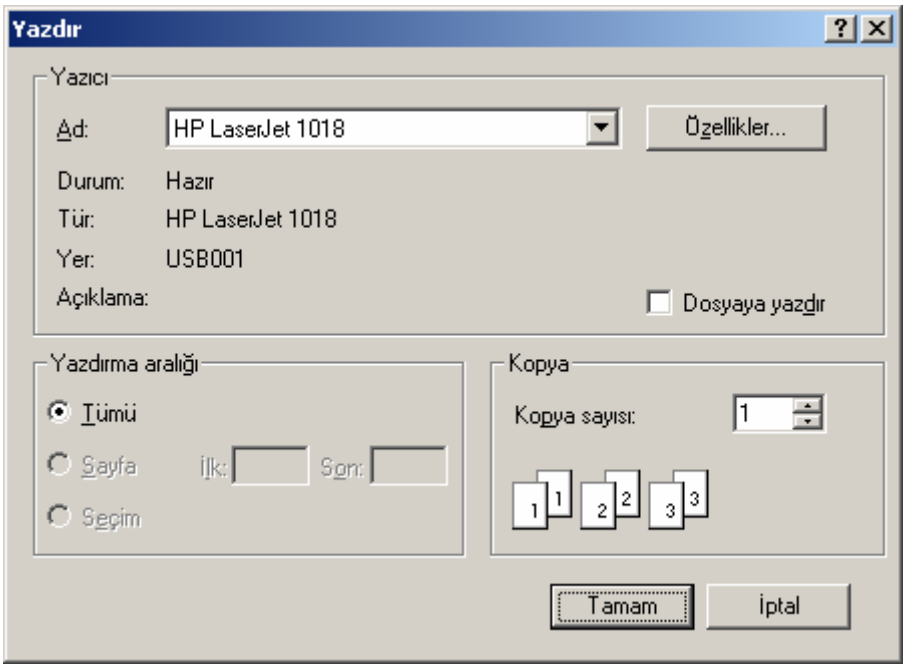
Şekil 5.86

5.14.5 Yazdırma Penceresi (Print Dialog) :

Bu diyalog penceresi, aktif olan figure alanının veya bir grafik çiziminin yazıcıdan direk çıktı alınmasını sağlar. Sayfa konumu diyalog penceresinden farkı dökümanı yazdırmadan önce herhangi bir ayar yapılmamasıdır. Ayrıca, sayfa konumu diyalog penceresinin önceden yapmış olduğu ayarları kullanarak çıktı alınması sağlar. Kullanımı şu şekildedir:

```
printdlg
```

Yazdırma diyalog penceresinin ekran görüntüsü Şekil 5.87’de gösterilmiştir.



Şekil 5.87

Şekil 5.87'deki ekranda kullanıcı "Tamam" butonunu tıkladığında yazdırılmak üzere döküman yazcıya gönderilir ve yazdırılır.

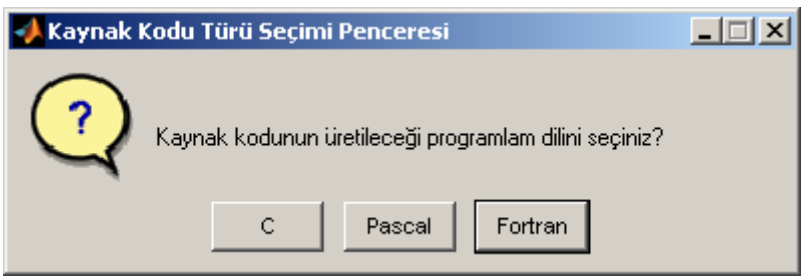
5.14.6 Sorgu Penceresi (Question Dialog) :

Bu diyalog penceresi ile uygulama sırasında kullanılacak verilerin çeşitli seçenekler içerisinden birinin seçilerek kullanıcılardan alınması sağlanır. Kullanımı şu şekildedir:

```
sorgu_sonucu = questdlg ('Kaynak kodunun üretileceği programlam dilini seçiniz?',
                         'Kaynak Kodu Seçimi Penceresi'
                         'C', 'Pascal', 'Fortran', 'Fortran')
```

Yukarıda gösterilen kullanımda son parametre varsayılan olarak seçili olacak seçeneği gösterir. Son parametre bu parametreden bir önceki üç parametreden biri olmalı veya '' şeklinde boş bırakılmalıdır. Eğer kullanıcı diyalog penceresi herhangi bir butona basmadan ve X butonu kullanarak kapatılırsa geri dönüş değeri boş bir dizi şeklindedir, yani '' şeklinde olacaktır. Eğer böyle bir durum kontrol edilecekse "isem pty" fonksiyonu kullanılabilir. Bu fonksiyon bir değişken içeriği boş ise "true", dolu ise "false" üretir. "isempty(degisken_ismi)" şeklinde kullanıma sahiptir.

Sorgu diyalog penceresinin ekran görüntüsü Şekil 5.88'de gösterilmiştir.



Şekil 5.88

Sorgu penceresinden dönen değer basitçe aşağıda gösterilen switch case yapısı ile kontrol edilebilir.

```
switch sorgu_sonucu
        case ‘C’
        case ‘Pascal’
        case ‘Fortran’
end
```

5.14.7 Renk Seçim Penceresi (Color Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında herhangi bir rengi seçmelerine imkân vermek amacıyla kullanılır. Kullanımı şu şekildedir:

```
donus_rengi = uisetcolor
```

Kullanıcı eğer “Cancel” butonu ile bu diyalog penceresi kapatırsa, yani bu diyalog penceresinden herhangi bir renk seçilmez ise “donus_rengi” değişkenine 0 (sıfır) değeri atanır. Örneğin kullanıcı kırmızı rengi seçerse “donus_rengi” değişkeninin içeriği [1 0 0] şkelinde olacaktır. Burada 3 boyutlu sütun vektör yapılı bir matris değişkeni şeklinde değerler döner. Bu matrisin sırasıyla elemanları kırmızı (red, R), yeşil (green, Y) ve mavi (blue, B) renklerinin değerlerini verir. Ancak, bu RGB değerlerinin her biri 0-255 yerine 0-1 arası değerler alır. Uygualamda 1 ile 255 arası oranlama şeklinde gerçek desimal formatta RGB değeri elde edilebilir.

Renk seçim diyalog penceresinin ekran görüntüsü Şekil 5.89’da gösterilmiştir.

Şekil 5.89

5.14.8 Font Seçim Penceresi (Font Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında herhangi bir fontu (yazı tipini) seçmelerine imkân vermek amacıyla kullanılır. Kullanımı şu şekildedir:

secilen _font_bilgisi = uisetfont

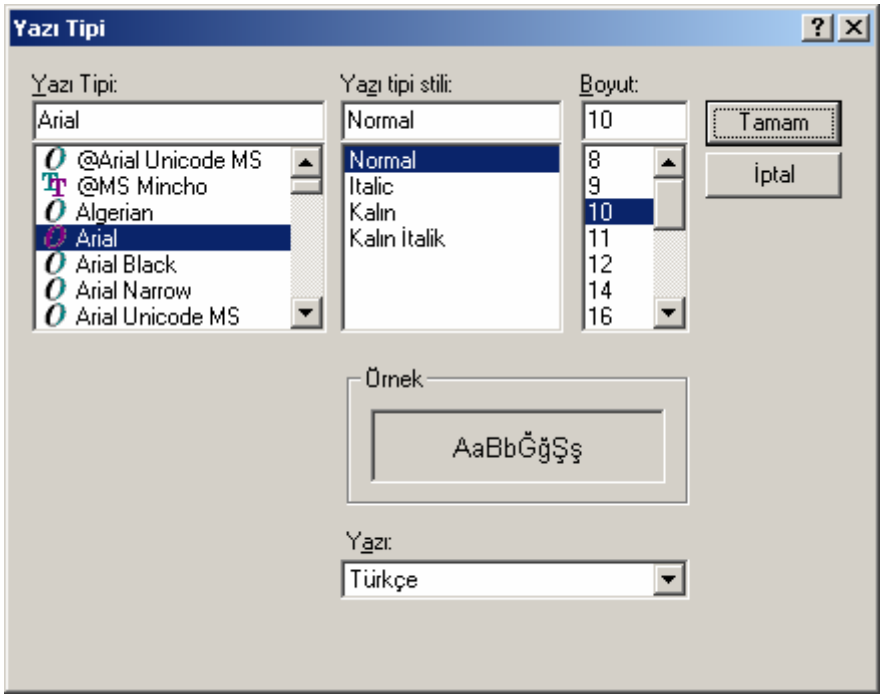
Kullanıcı eğer diyalog penceresi "İptal" butonunu kullanarak kapatmışsa geri dönüş değeri 0 (sıfır) olur. Eğer kullanıcı bir fontu seçip "Tamam" butonuna tıklayarak font seçim penceresini kapatmış ise "secilen_font_bilgisi" değişkenine

- Fontname : font ismi string bilgisi (örneğin 'Arial')
- FontUnits : font birimi string bilgisi (örneğin 'points')
- FontSize : font boyutu sayısal bilgisi (örneğin 10)
- FontWeight : font ağırlığı string bilgisi ('normal' veya 'bold' olabilir.)
- FontAngle : font açısı string bilgisi ('normal' veya 'italic' olabilir.)

alanları olan bir yapı değişkeni döner. Örneğin seçilen fontun boyutunu kullanmak için şu şekilde bir komut yapısı kullanılmalıdır:

font_boyutu = secilen_font_bilgisi. FontSize % "FontSize" ismindeki büyük ve küçük
% harflere dikkat edilmelidir.

Font seçim diyalog penceresinin seçilmiş bir font ile birlikte ekran görüntüsü Şekil 5.90'da gösterilmiştir.



Şekil 5.90

5.14.9 Uyarı Penceresi (Warn Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında eksik veya yanlış bilgi girme ya da herhangi bir konuda bilgilendirilme amacıyla kullanılır. Kullanımı şu şekildedir:

warndlg(‘Sistem kaynakları kritik seviyededir.’, ‘Tehlike’, ‘modal’)

Uyarı diyalog penceresinin ekran görüntüsü Şekil 5.91’de gösterilmiştir.

Şekil 5.91

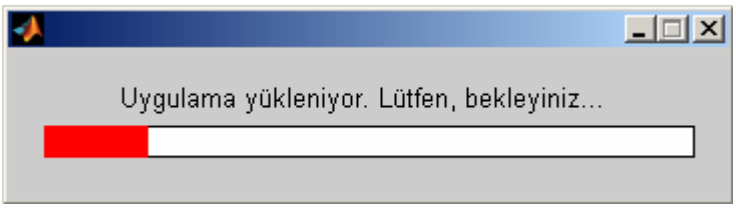
Modal parametresi ile kullanıcının bu pencereye cevap vermeden işlemlerine devam etmesi engellenmiş olmaktadır. Bu seçenek belirtilmez veya ‘non-modal’ seçeneği kullanılırsa modal özelliği devre dışı bırakılmış olur.

5.14.10 Yükleme Çubuğu (waitbar) :

Yükleme çubuğunu programcı bir uygulamanın başında kullanarak kullanıcıya uygulamanın yüklenmekte olduğunu ve ne kadarının yüklendiği görsel bir şekilde sunabilir. Kullanımı şu şekildedir:

```
yukleme_cubugu = waitbar ( 0 , ‘Uygulama yükleniyor. Lütfen, bekleyiniz…’ ) ;
for i=1:1:100              % bir üstteki satır ile waitbar nesnesi oluşturuluyor.
        % uygulamanın yüklenmesi sırasında yer alan işlemler
        waitbar ( i / 100 , yukleme_cubugu ) ; % oluşturulmuş waitbar nesnesinin günc.
end
close (yukleme_cubugu)                    % oluştur. olan waitbar nesnesin. silinmesi
```

Yükleme çubuğunun ekran görüntüsü Şekil 5.92’de gösterilmiştir.



Şekil 5.92

5.14.11 Klasör Yolu Penceresi (UIGetDir Dialog) :

Bu diyalog penceresi, kullanıcıların uygulama sırasında bir dizin yolunu seçmeleri amacıyla kullanılır. Kullanım şekilleri çeşitlidir. Bunlar aşağıda gösterilmiştir.

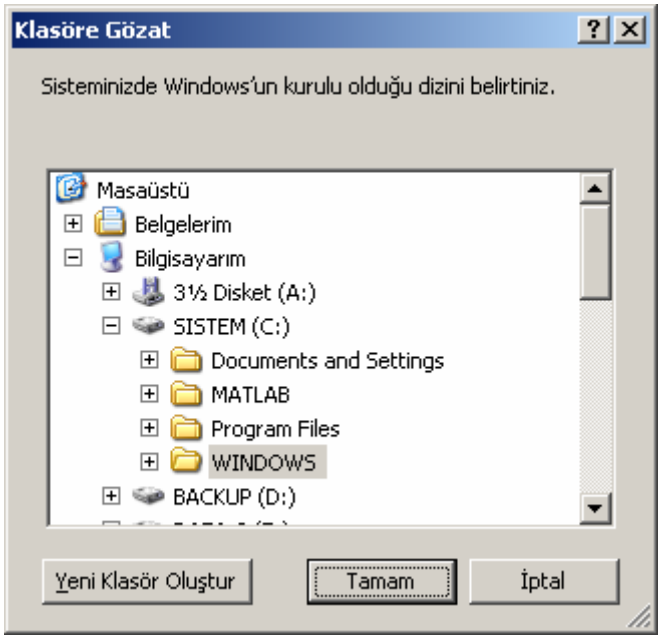
```
klasor_ismi = uigetdir
klasor_ismi = uigetdir (‘baslangic_dizin_yolu’)
klasor_ismi = uigetdir (‘baslangic_dizin_yolu’,’goruntulenecek_mesaj_stringi’)
```

Kullanımlarda “baslangic_dizin_yolu’” parameresi diyalog penceresi açıldığında ilk görüntülenecek klasörün seçilmesini ve son parametre de bu pencere ekstra görüntülenmesi istenilen mesajın çıkmasını sağlar.

Örnek olarak aşağıdaki kullanım ile bir diyalog kutusu ekrana gösterilsin.

klasor_ismi = uigetdir (‘C:\Windows’ , ‘Sisteminizde Windows’un kurulu olduğu dizini belirtiniz.’)

Örnek kullanıma ait klasör yolu diyalog penceresinin ekran görüntüsü Şekil 5.93’te gösterilmiştir.



Şekil 5.93

Bu diyalog kutusu iptal düğmesi tıklanarak kapatılırsa 0 sayısı (false cevabı) üretir. Eğer tamam butonu tıklanırsa seçilmiş olan klasöre ait tam yolu (directory path’i) geri dönüş değeri olarak gönderilir.

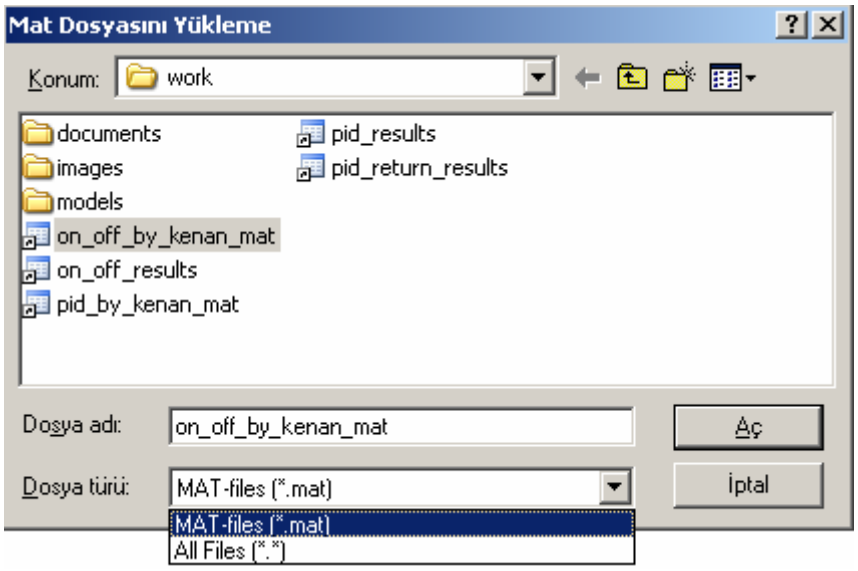
5.14.12 Dosya Açma Penceresi (UIGetFile Dialog) :

Bu diyalog penceresi ile kullanıcıdan bilgisayarda yer alan ve belirlenmiş türde dosyaların seçilerek açılması ve seçilen dosyaya ait path tanımının GUI uygulamasına aktarılması amacıyla kullanılır. Kullanımı şu şekildedir:

[dosya_ismi, dosya_yolu] = uigetfile (‘veriler.mat’ , ‘Mat Dosyasını Yükleme’)

Bu komut yapısında ekrana gelecek dosya açma diyalog penceresinin başlığı “Mat Dosyasını Yükleme” ve varsayılan dosya uzantısı olarak “.mat” olan dosyaların seçilmesi sağlanılacaktır. Bu örneğe ilişkin ekran görüntüsünde de görüldüğü üzere kullanıcı isterse *.* formatlı olmak üzere bu diyalog penceresinde dosya türünü “Tüm Dosyalar” olarak seçebilir.

Örnek olarak verilen komut yapısının işletilmesi sonucu dosya açma diyalog penceresinin ekran görüntüsü Şekil 5.94’teki gibi olacaktır.



Şekil 5.94

Kullanıcı örnek olarak verilen komutun sonucunda açılan diyalog penceresini "İptal" butonuan basarak ya da herhangi bir dosya seçmeden kapatırsa "dosya_ismi" ve "dosya_yolu" değişkenlerine 0 (sıfır) (yani mantıksal false) değeri atanır. Ancak, kullanıcı bu pencere yardımıyla bir dosyayı seçer ve "Aç" butonunu tıklayarak pencereyi kapatırsa "dosya_ismi" değişkenine seçilen dosyanın ismi ve "dosya_yolu" değişkenine bu dosyaya ait path tanımı (dizin yolu) string tipte olarak atanacaktır. Örnek olarak ekran görüntüsünde verilen "on_off_by_kenan_mat.mat" dosyası seçilmiş olduğu düşünülürse

- "dosya_ismi" değişkeninin içeriği 'on_off_by_kenan_mat.mat' string bilgisi
- "dosya_yolu" değişkeninin içeriği 'C:\MATLAB\work\' string bilgisi

olacaktır. Bu örnekte dosya açma ve Workspace'a değerlerin yüklenmesi ile desteklenirse aşağıdaki deyimler kullanılabilir.

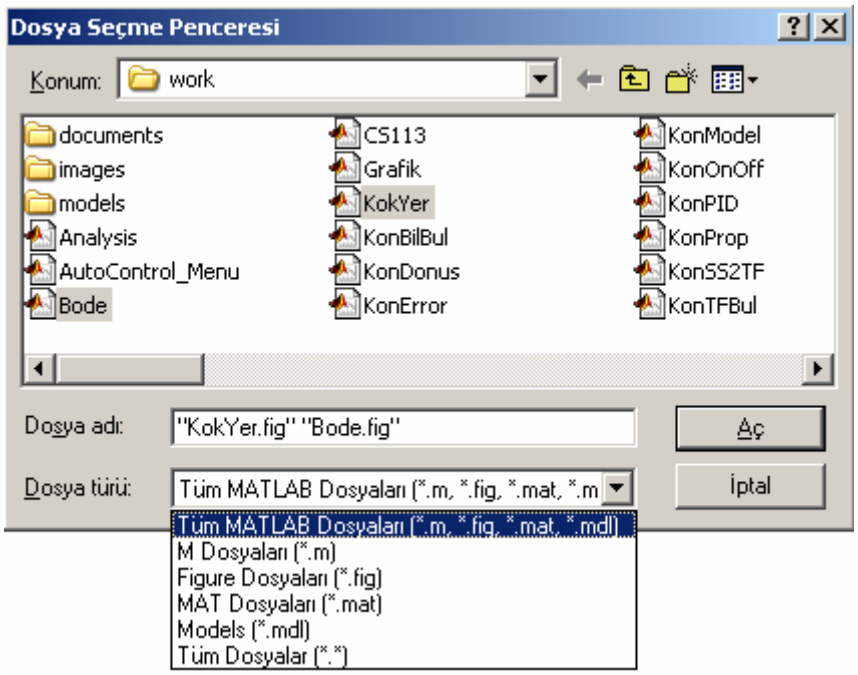
```
[dosya_ismi, dosya_yolu] = uigetfile ( 'veriler.mat' , 'Mat Dosyasını Yükleme' ) ;
if isequal ( dosya_ismi , 0)
        load ( [ dosya_yolu dosya_adi ] );
end
```

Ayrıca, kullanıcının seçmesi istenilen dosya türleri çok çeşitli ise (örneğin resim dosyaları gibi) bu durumda birden fazla dosya formatı olacak şekilde dosya açma diyalog penceresi için filtre formatlar tanımlanabilir. Çoklu dosya formatı tanımlamakla ilgili aşağıdaki örnek incelenebilir. Ayrıca, bu örnekte çoklu dosya seçimi ve yönetilmesi de gösterilmiştir.

```
[dosya_ismi, dosya_yolu, secilen_filtre_no] = uigetfile( ...
   {'*.m;*.fig;*.mat;*.mdl', 'Tüm MATLAB Dosyaları (*.m, *.fig, *.mat, *.mdl)';
   '*.m',  'M Dosyaları (*.m)'; ...
   '*.fig','Figure Dosyaları (*.fig)'; ...
   '*.mat','MAT Dosyaları (*.mat)'; ...
   '*.mdl','Models (*.mdl)'; ...
   '*.*',  'Tüm Dosyalar (*.*)'}, ...
   'Dosya Seçme Penceresi', ...
```

'MultiSelect', 'on')

Yukarıda verilen komut satırında “secilen_filtre_no” geri dönüş parametre değişkenine kullanıcı dosya açma diyalog penceresinin Dosya türü listesinden hangi filtreyi seçmiş ise bu liste öğesinin değeri numarası atanır. Seçilen öğe listenin ilk elemanı ise bu değişkenin içeriği 1 olacaktır. Yukarıdaki komut satırlarının işletilmesi ile Şekil 5.95’te yer alan ekran görüntüsü elde edilir.



Şekil 5.95

Bir dosya açma penceresi varsayılan olarak tek dosya seçmek üzere açılır. Birden fazla dosyanın seçildiği durumlar için ise “uigetfile” komutunun son iki parametresi ‘MultiSelect’,’on ’ string bilgileri olmalıdır. Bu özellik aktif edildiği takdirde kullanıcı klavyenin “Ctrl” tuşu basılı halde birden fazla dosyayı seçebilir. Örneğin ekran görüntüsünde yer alan “Bode.fig” ve “KokYer.fig” dosyalrı seçilmiş ve “Aç” butonuna basılmış olur. Bu durumda geri dönüş parametrelerin içerikleri şöyle olacaktır:

- “dosya_ismi” değişkeninin içeriği [‘Bode.fig’ ‘KokYer.fig’]
- “dosya_yolu” değişkeninin içeriği ‘C:\MATLAB\work\’

Burada örneğin 2. dosya ismi şu komut satırıyla elde edilir:

secilen_ikinci_dosya_ismi = dosya_ismi(2)

Eğer herhangi bir dosya seçilmezse her iki geri dönüş parametresine 0 (sıfır) değeri atanır.

5.14.13 Dosya Kaydetme Penceresi (UIPutFile Dialog) :

Bu diyalog penceresi ile kullanıcıdan bilgisayara kaydedilecek bir dosyanın yeri ve isminin belirlenmesi amacıyla kullanılır. Dosya aç iletişim kutusu ile aynı özelliklere sahiptir. (Dosya açma iletişim kutusu hakkında daha ayrıntılı bilgi için bir önceki konu başlığında bakılabilir.) Ancak, bu diyalog penceresi ile dosya açma diyalog penceresi arasındaki fark

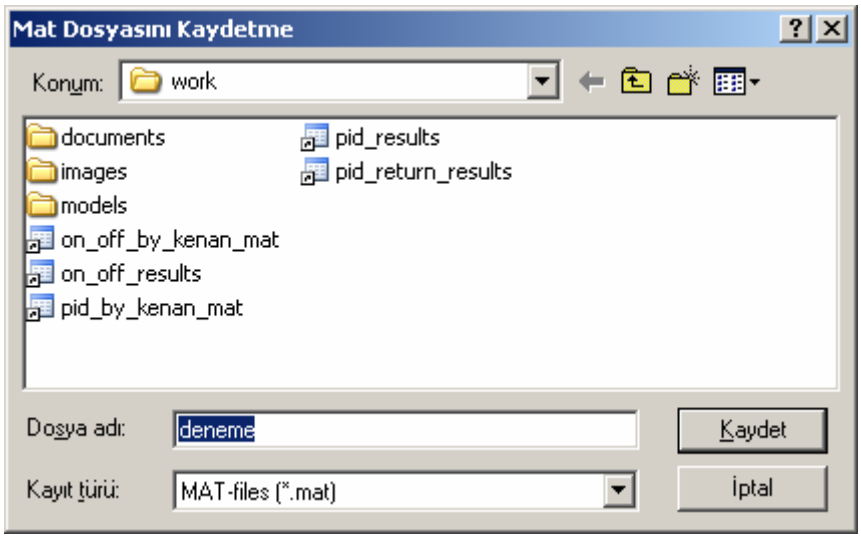
uigetfile komutu yerine uiputfile komutunun kullanılması ve dosya kaydetme kavramları içerisinde “multiselect” (yani çoklu seçim) gibi bir seçeneğin olmamasıdır. Kullanımı şu şekildedir:

[dosya_ismi, dosya_yolu] = uiputfile (‘*.mat’ , ‘Mat Dosyasını Kaydetme’)

Bu komut yapısında ekrana gelecek dosya açma diyalog penceresinin başlığı “Mat Dosyasını Yükleme” ve varsayılan dosya uzantısı olarak “.mat” olacak şekilde dosyanın kaydedilmesi sağlanır. İstenirse ‘*.mat’ parametresi boş bırakılabilir, yani bu parametre ‘’ şeklinde tanımlanabilir. Varsayılan olarak buraya girilen bir dosya ismi dosya kaydetme penceresi açıldığında görülecektir. Örneğin aşağıdaki komut çalıştırılmış olsun. Bu komuta ait ekran görüntüsü de aşağıda sunulmuştur.

[dosya_ismi, dosya_yolu] = uiputfile (‘deneme.mat’ , ‘Mat Dosyasını Kaydetme’)

Örnek olarak verilen komut yapısının işletilmesi sonucu dosya açma diyalog penceresinin ekran görüntüsü Şekil 5.96’daki gibi olacaktır.



Şekil 5.96

Yukarıdaki pencerede “deneme” ve “MAT-Files” filtresi ile “kaydet” butonu tıklandığında geri dönüş parametreleri

- “dosya_ismi” değişkeni içinde ‘deneme.mat’ string bilgisi,
- “dosya_yolu” değişkeni içinde ‘C:\MATLAB\work\’ string bilgisi

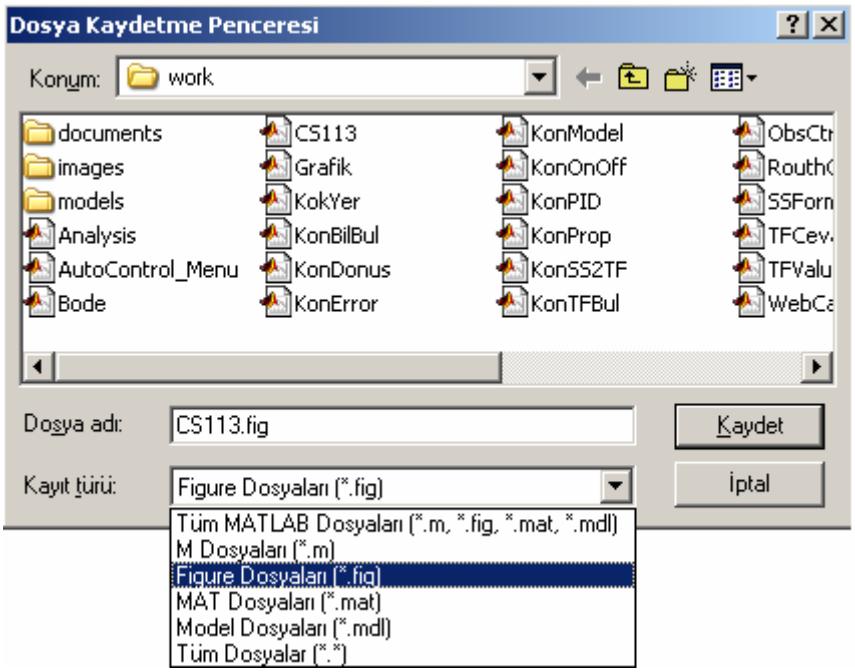
şeklinde olacaktır.

Aşağıdaki örnek birden fazla dosya formatının nasıl tanımlandığı, yani filtrelemenin nasıl yapıldığı konusunda bilgi vermektedir.

```
[dosya_ismi, dosya_yolu, secilen_filtre_no] = uiputfile( ...
   {'*.m;*.fig;*.mat;*.mdl', 'Tüm MATLAB Dosyaları (*.m, *.fig, *.mat, *.mdl)';
    '*.m',  'M Dosyaları (*.m)'; ...
    '*.fig','Figure Dosyaları (*.fig)'; ...
    '*.mat','MAT Dosyaları (*.mat)'; ...
```

'*.mdl','Model Dosyaları (*.mdl)'; ...
'*.*', 'Tüm Dosyalar (*.*)'}, ...
'Dosya Kaydetme Penceresi')

Yukarıda verilen komutun çalıştırılması sonucu Şekil 5.97'deki ekran görüntüsü gelecektir.



Şekil 5.97

Şekil 5.97'deki ekranda CS113.fig dosya ismi verilmiş ve filtre tipi olarak listenin 3. sırasında yer alan "Figure Dosyaları" işaretlenmiştir. Kullanıcı "Kaydet" butonunu tıkladığında geri dönüş parametreleri şöyle olacaktır:

- "dosya_ismi" değişkeni içinde 'CS113.fig' string bilgisi,
- "dosya_yolu" değişkeni içinde 'C:\MATLAB\work\' string bilgisi
- "secilen_filtre_no" değişkeni içinde 3 sayısal bilgisi

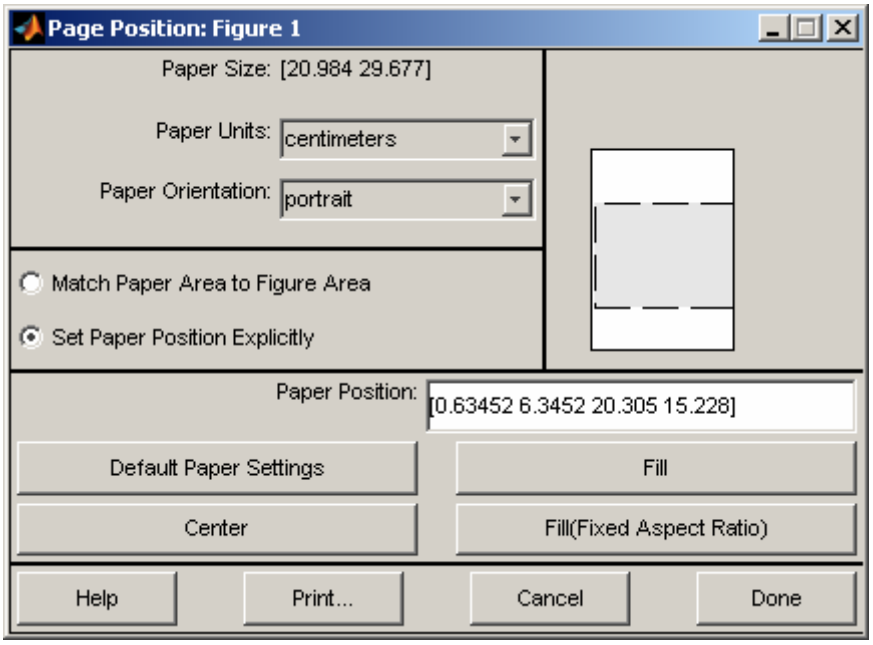
Burada seçilen filtre türlerinden 3. sıradaki filtre seçildiği için "secilen_filtre_no" değişkenine 3 sayısal değeri atanmıştır.

5.14.14 Sayfa Yapısı Penceresi (Page Dialog) :

Bu diyalog penceresi, yazıcıdan çıktı alınacak döküman ile ilgili ayarların yapılması nı ve ayrıca aktif olan figure alanının veya bir grafik çiziminin yazıcıdan çıktı alınmasını sağlar. Kullanımı şu şekildedir:

pagedlg

Sayfa yapısı diyalog penceresinin ekran görüntüsü Şekil 5.98'de gösterilmiştir.



Şekil 5.98

Kullanıcı bu diyalog penceresinde sayfa ile ilgili gerekli ayarlamaları yaptıktan sonra "Print" butonunu kullanarak aktif figure alanı veya grafik çiziminin yazıcıdan çıktı alabilir.

5.14.15 Mesaj Kutusu (MessageBox Dialog) :

Bu diyalog penceresi kullanılarak kullanıcılara herhangi bir mesaj istenilen bir resim ya da ikon dosyası ile birlikte gösterilebilir. Çok çeşitli kullanımlara sahip bir diyalog penceresi olup, aşağıda bu kullanım çeşitleri gösterilmiştir.

```
msgbox (mesaj)                % Şekil  …...ya bakınız.
msgbox (mesaj, baslik)        % Şekil  …...ya bakınız.
msgbox (mesaj, baslik, ikon) % Şekil  …...ya bakınız.
msgbox (mesaj, baslik, 'custom', resim_data_degiskeni, resim_colormap_degiskeni)
msgbox (mesaj, ……., olusturulma_modu)
```

Bu kullanımlardaki parametrelerin tipi ve görevleri şu şekildedir:

- mesaj : Mesaj kutusunda gözükecek string bilgi
- baslik : Mesaj kutusunun pencere başlığında gözükmesi istenilen string bilgi
- ikon : 'none', 'error', 'help', 'warn' veya 'custom' değerlerinden biri olabilir. Varsayılan değeri
  - 'none' olup, Matlab'in kendi içinde kullandığı hazır standart ikonlkarın gösterilmesi
  - bu parametre ile saplanır. Bu paramerenin kullanımı ile ilgili örnek ekran
  - görüntğlerine aşağıdan bakılabilir.
- resim_data_degiskeni : import edilen bir resim dosyasının veri (data) dizis. tutan değiş.
- resim_colormap_degiskeni : import edilen bir resim dosyasının colormap dizis. tutan değiş.
- olusturulma_modu : Bu parametre msgbox komutuna gönderilecek son

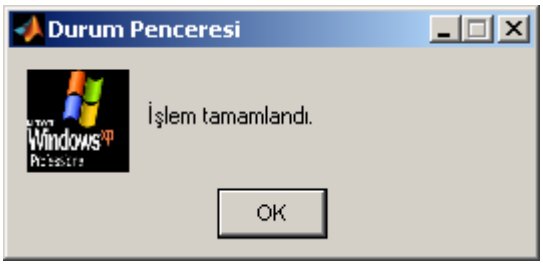
parametre bilgisi olmalıdır. ‘modal’ veya ‘non-modal’ string bilgilerinden biri olabilir. Varsayılan değer ‘non-modal’ seçeneğidir. Eğer modal seçilirse kullanıcı mesaj kutusuna cevap vermeden arka plandaki uygulamaya dönemz, yani arkapalnın kilitlenmesi sağlanır. ‘non-modal’ seçeneği ise bu durumun tam tersidir.

msgbox (mesaj, baslik, ‘custom’, resim_data_degiskeni, resim_colormap_degiskeni)

Yukarıdaki kullnım şekli için için öncelikle bir resmin “imread” komutu ile Matlab’in “Workspace” alanına yüklenmesi ve bu fonksiyonun geri dönüş parametrelerinin daha sonra kullanılması gereklidir. Örnek kullanım için aşağıdaki komut satırları incelenebilir.

```
[resim_data_degiskeni, resim_colormap_degiskeni] = imread (‘C:\Windows\winnt256.bmp’);
mesaj = ‘İşlem tamamlandı.’;
baslik = ‘Durum Penceresi’ ;
msgbox (mesaj, baslik, ‘custom’, resim_data_degiskeni, resim_colormap_degiskeni)
```

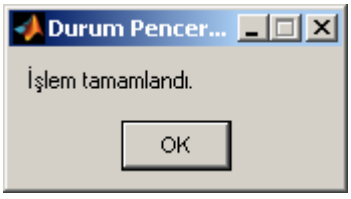
Yukarıda verilen komut satırıçalıştırıldığında Şekil 5.99’daki gibi bir ekran görüntüsü oluşacaktır.



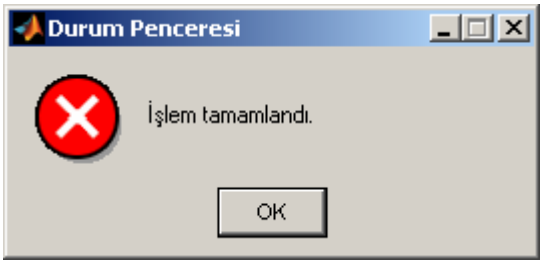
Şekil 5.99

Çeşitli mesaj kutularına ait ekran görüntüleri Şekil 5.100, Şekil 5.101, Şekil 5.102, Şekil 5.103 ve Şekil 5.104’te verilmiştir.

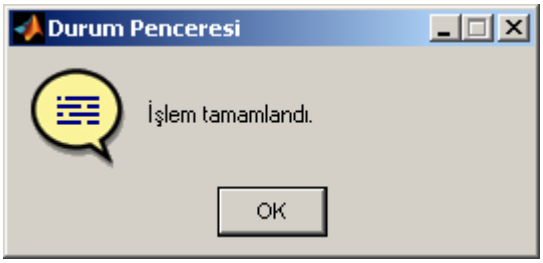
Şekil 5.100



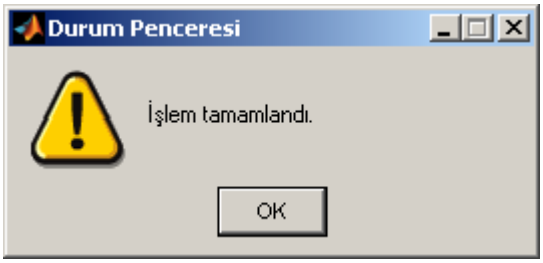
Şekil 5.101



Şekil 5.102



Şekil 5.103



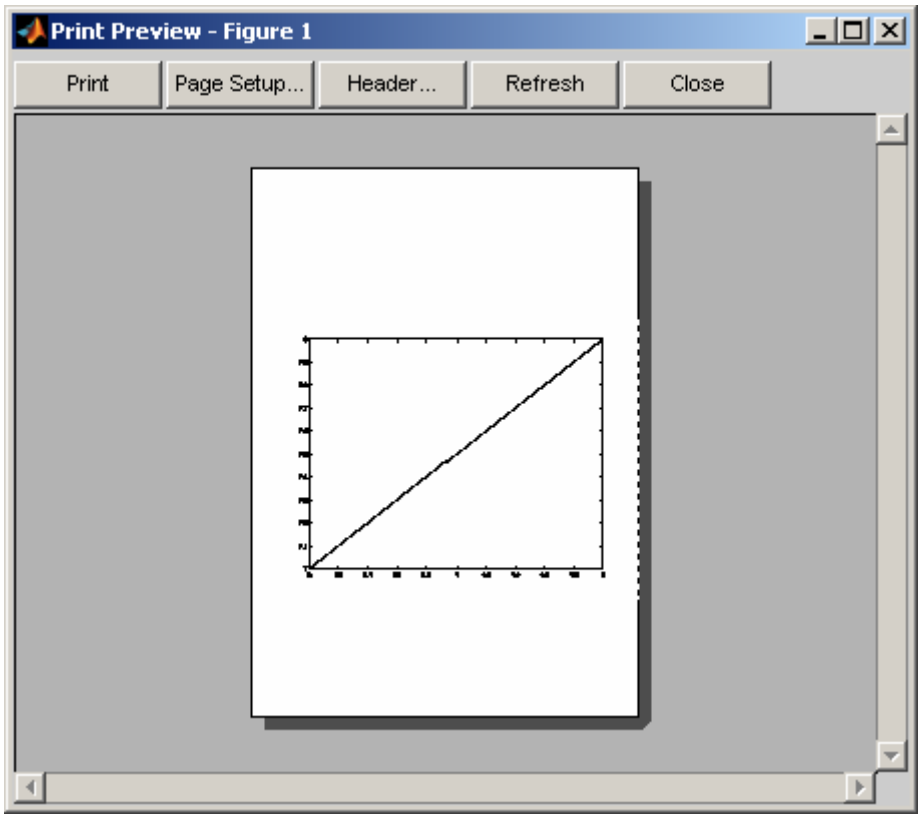
Şekil 5.104

5.14.16 Sayfa Önizleme Penceresi (PrintPreview Dialog) :

Bu diyalog penceresi ile kullanıcılar yazıcıya aktif figure alanını veya aktif axes (grafik çizim nesnesi) içeriğini göndermeden önce sayfanın önizlemesini görebilirler. Ayrıca, sayfa ile ilgili ayarlamaları yapabilir ve sayfanın yazıcıdan çıktısını alabilir. Kullanımı şu şekildedir:

```
printpreview
```

Sayfa önizleme diyalog penceresinin örnek bir uygulama için ekran görüntüsü Şekil 5.105'te gösterilmiştir.



Şekil 5.105

Kullanıcı bu ekranda "Print" butonu ile sayfayı yazdırabilir. "Header" butonunu kullanarak isterse kullanıcı sayfaya istediği bir fontta başlık ekleyebilir. "Page Setup" butonu kullanılarak gelen pencereden (bu pencere ile ilgili bilgi almak için önceki konu başlıklarından "Sayfa Yapısı Penceresi" bölümü kullanılarak detaylı bilgiler alınabilir.) sayfa ile ilgili ayarlamalar yapılabilir. Kullanıcı bu sayfayı "Close" butonunu kullanarak kapatabilir.

BÖLÜM-VI
MATLAB'TE KONTROL ALANI İLE İLGİLİ TASARLANAN GUI UYGULAMALARI

Bu bölümde Matlab kullanılarak hazırlanmış olan ve Otomatik Kontrol derslerinde kullanılabilme amacıyla hazırlanan çok çeşitli GUI uygulamalarının kullanımı, tasatımı ve progranması hakkında bilgi verilecektir.

Burada anlatılan GUI uygulamaları tez kitabının arka kabağının iç yüzüne eklenmiştir.

6.1 Uygulama-1'in Tanıtılması

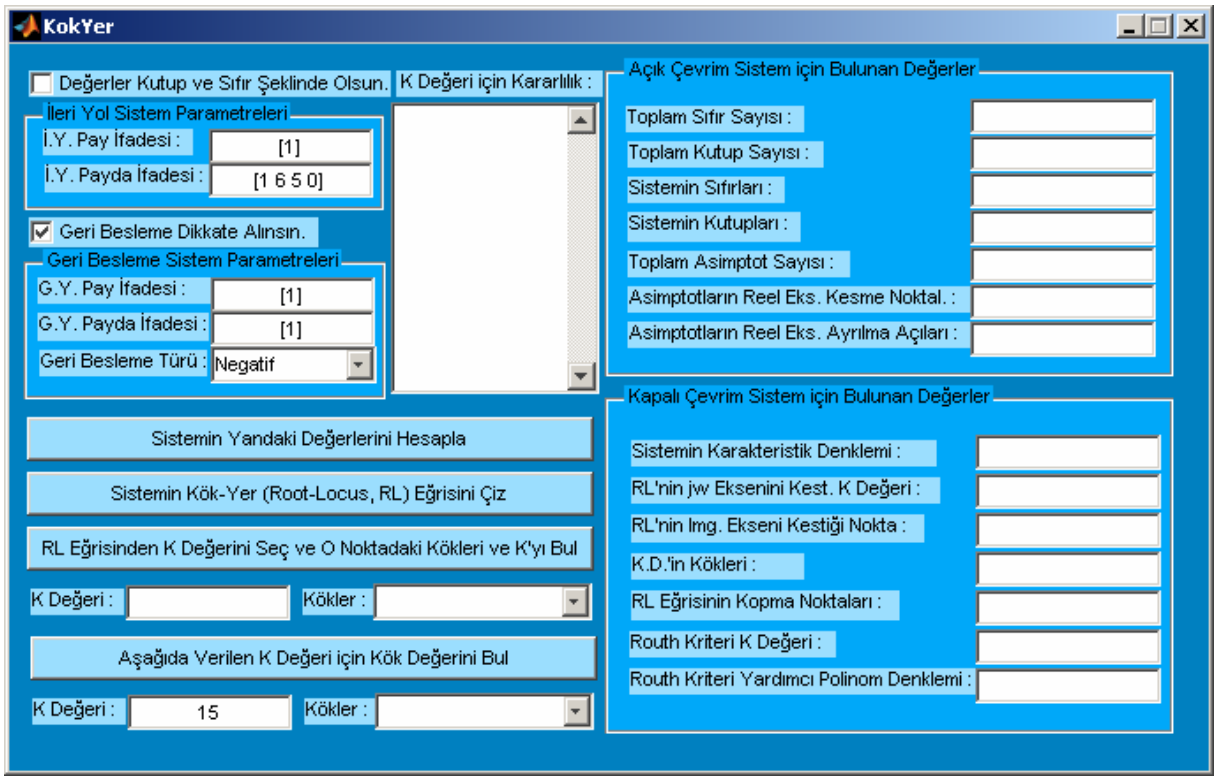
6.1.1 Uygulamanın Adı : Kök-Yer Eğrisinin Çizimi ve Hesaplamalarının Yapılması

6.1.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerine uygulanan kazanç değerine bağlı olarak sistemin kararlığının nasıl olacağı hakkında bilgi edinilmesini sağlayan yöntemlerden biri olan Kök-Yer eğrisi ile ilgili bilinmeyen parametrelerin kendi içerisinde yer alan formülasyon kullanılarak hesaplanması ve bu eğrinin çizilerek kullanıcıya gösterilmesini sağlamaktır. Ayrıca, Kök-Yer eğrisi ile ilgili hesaplamaların programlanarak nasıl elde edilebileceği konusunda kullanıcıları bilgilendirmek amaçlanmıştır.

6.1.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.1'de görülmektedir.



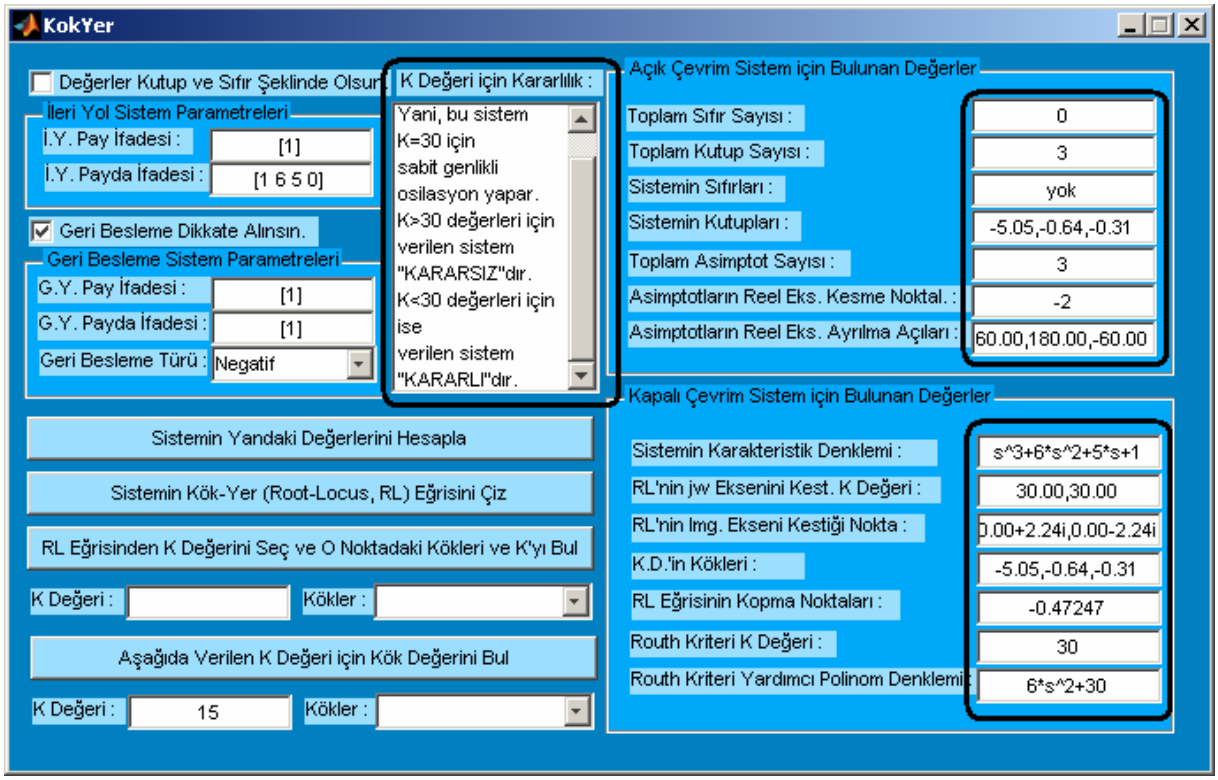
Şekil 6.1 Uygulama 1 Arayüzü

6.1.4 Uygulamanın Kullanılışı

Uygulamanın kullnılabilmesi için öncelikle kullanıcının ileri yol ve geri besleme bloklarına ait parametreelri girmesi gerekmektedir. Bunun için uygulama arayüzünde yer alan "İleri Yol Sistem Parametreleri" ile "Geri Yol Sistem Parametreleri" kısmından bloğa ait pay ve payda katsayıları girilmelidir. Kullanıcı isterse pay ve payda katsayıları yerine her bir bloğun sıfır ve kutup değerleri girerek de işlem yapabilir. Bunun için "Değerler sıfır ve kutup şeklinde olsun." Seçeneğinin işaretlenmesi gerekmektedir. Bu durumda kullanıcı ileri ve geri yol sitsem parametrelerinin pay ve payda kısımlarına kullanılacak sıfır ve kutup değerlerini girebilir. Bunun yanında "Geri Besleme Türü" bölümü kullanılarak kullanıcı seçtiği sistemin geri besleme tipini belirleyebilir. İsterse "Geri Besleme Dikkate Alınsın." Seçeneği işaretlemeyerek sistemin açık çevrim transfer fonsktiyonu ile ilgili işlem yapabilir. Programda örnek olarak varsayılan değerler ileri sistem bloğu için pay " [1] " ve payda " [1 6 5 0] " ile geri isistem için pay " [1] " ve payda " [1] " olarak seçilmiştir. Sistemde geri blok ileri bloğa negatif geri besleme ile bağlı olarak belirlenmiştir. Programın anlatılmasında örnek olması amacıyla bu değerler kullanılacaktır. Uygulama temel olarak 4 farklı işleve sahiptir. Bu işlevlerin programın kullanılarak nasıl gerçekleştirileceği ile ilgili ayrıntılı bilgiler aşağıda verilmiştir.

6.1.4.1 Kök-Yer Eğrisi İle İlgili Hesaplamaların Yapılması ve Eğrinin Yorumlanması

Bu işlem için kullanıcı "Sistemin Yandaki Değerlerini Hesapla" butonuna basmalıdır. Bu işlem gerçekleştirildiğinde hesaplanan değerler örnek sistem için Şekil 6.2'de gösterilmiştir.

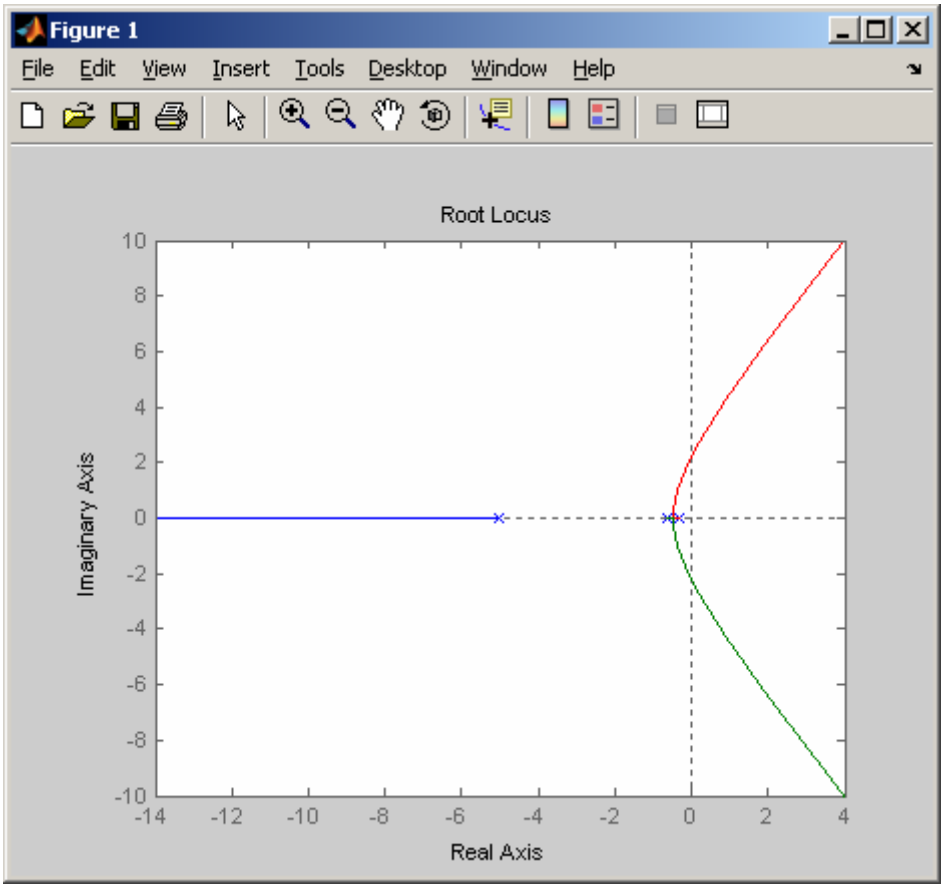


Şekil 6.2 Uygulama 1 Kullanım Ekranı 1

Program kendi içinde sistemin hangi K aralığı için kakrlı olup olmadığı hakkında yorum da yapabilmektedir. Örnek olarak az önce hesaplanan değerlere göre ele alınan sistem "K>30" değerleri için "KARARSIZ" ve "K<30" değeri için "KARARLI" ve son olarak "K=30" değeri için KRİTİK KARARLI'dır.

6.1.4.2 Kök-Yer Eğrisinin Çizdirilmesi

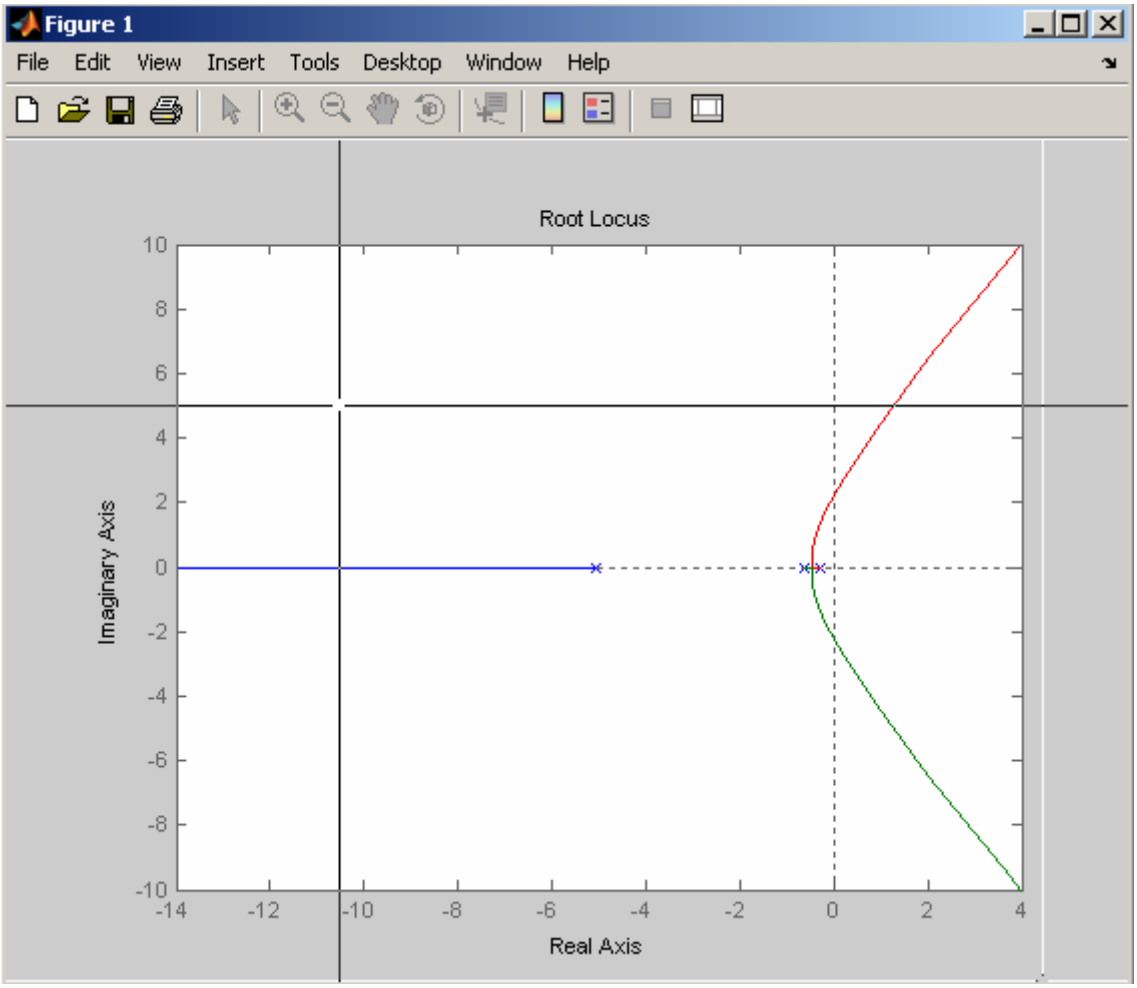
Bu işlem için "Sistemin Kök-Yer Eğrisini Çiz" butonuna basılmalıdır. Bu işlem yapıldığında ikinci bir pencere açılacak ve bu pencerede Kök-Yer eğrisi çizilecektir. Örnek kabul edilen sistem Kök-Yer eğrisi çizimi Şekil 6.3'te gösterilmiştir.



Şekil 6.3 Uygulama 1 Kullanım Ekranı 2

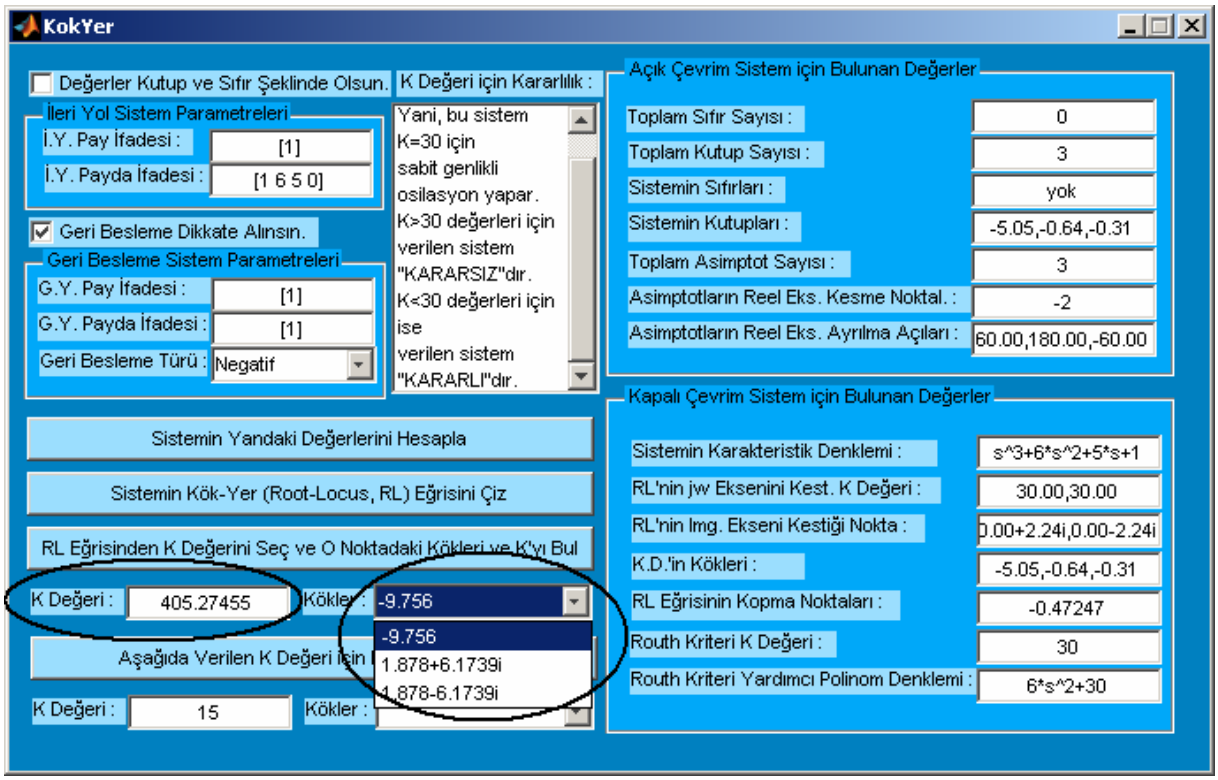
6.1.4.3 Kök-Yer Eğrisinde Seçilen Bir K Değerine Ait Köklerin Kullanıcıya Gösterilmesi

Bu işlemi gerçekleştirmek için "RL Eğrisinden K Değerini Seç ve O Noktaki Kökleri Bul" butonu kullnılmlaıdır. Bu işlem yapıldığında kullancıı Şekil 6.4'deki gibi fare ile seçebileceği yeri gösteren biri yatay, diğeri dikey iki çizgi hattı belirmektedir. Kullanıcı istediği bir noktayı



Şekil 6.4 Uygulama 1 Kullanım Ekranı 3

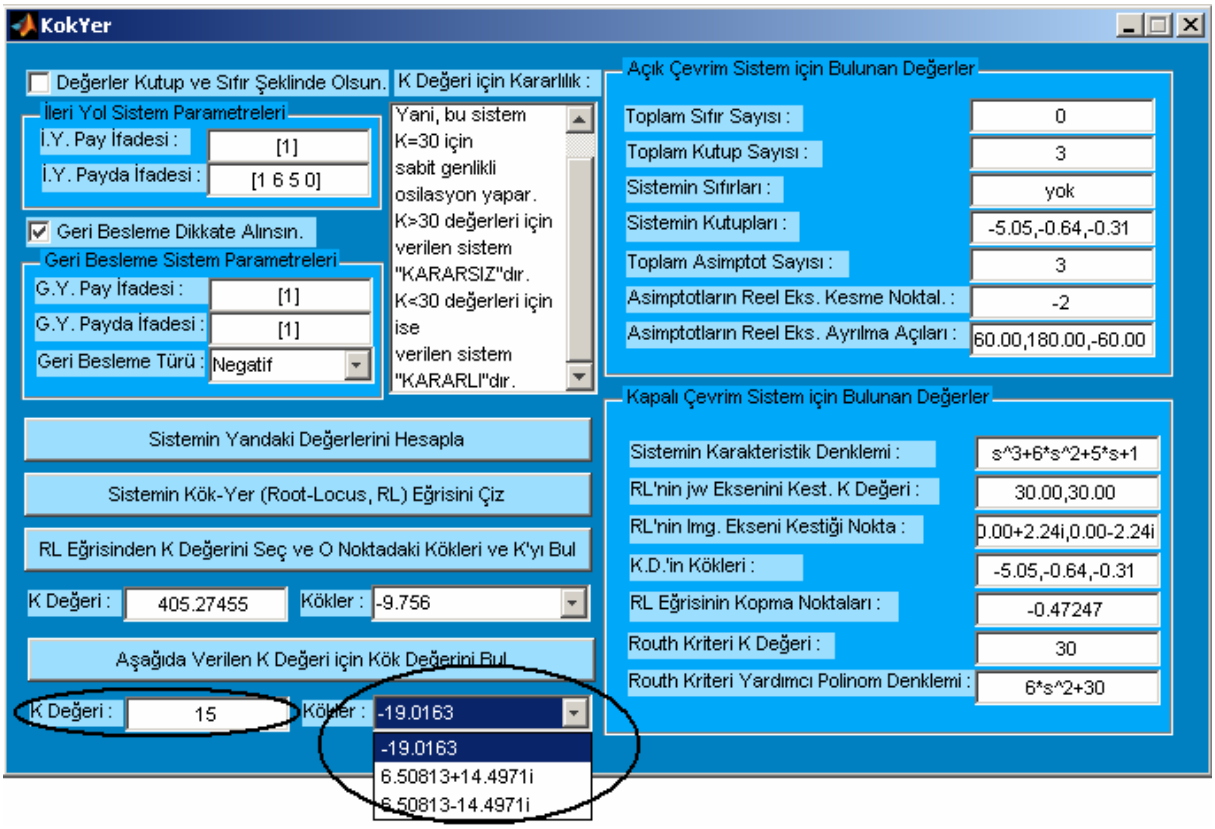
seçerek bu nokta farenin sol tuşuna bastığında o noktaya ait K değerini ve bu noktaki sistemin kökleri programda gösterilir. Bu durum Şekil 6.5'te de görülmektedir.



Şekil 6.5 Uygulama 1 Kullanım Ekranı 4

6.1.4.4 Kök-Yer Eğrisi İçin Verilen Bir K Değerine Ait Köklerin Bulunması

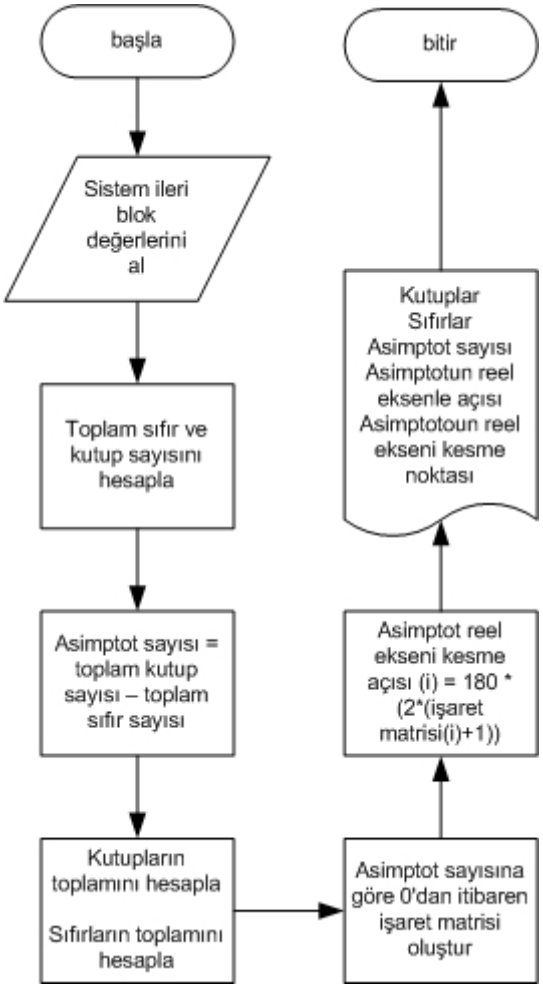
Kullanıcı "Aşağıda Verilen K Değeri için Kökleri Bul" butonunu kullanarak K Değeri yazılı alana girilen bir K katsayısına karşılık sistemin köklerini hemen bu alanın sağında yer alan Kökler kısmından öğrenebilir. Bu durum Şekil 6.6'da gösterilmiştir.



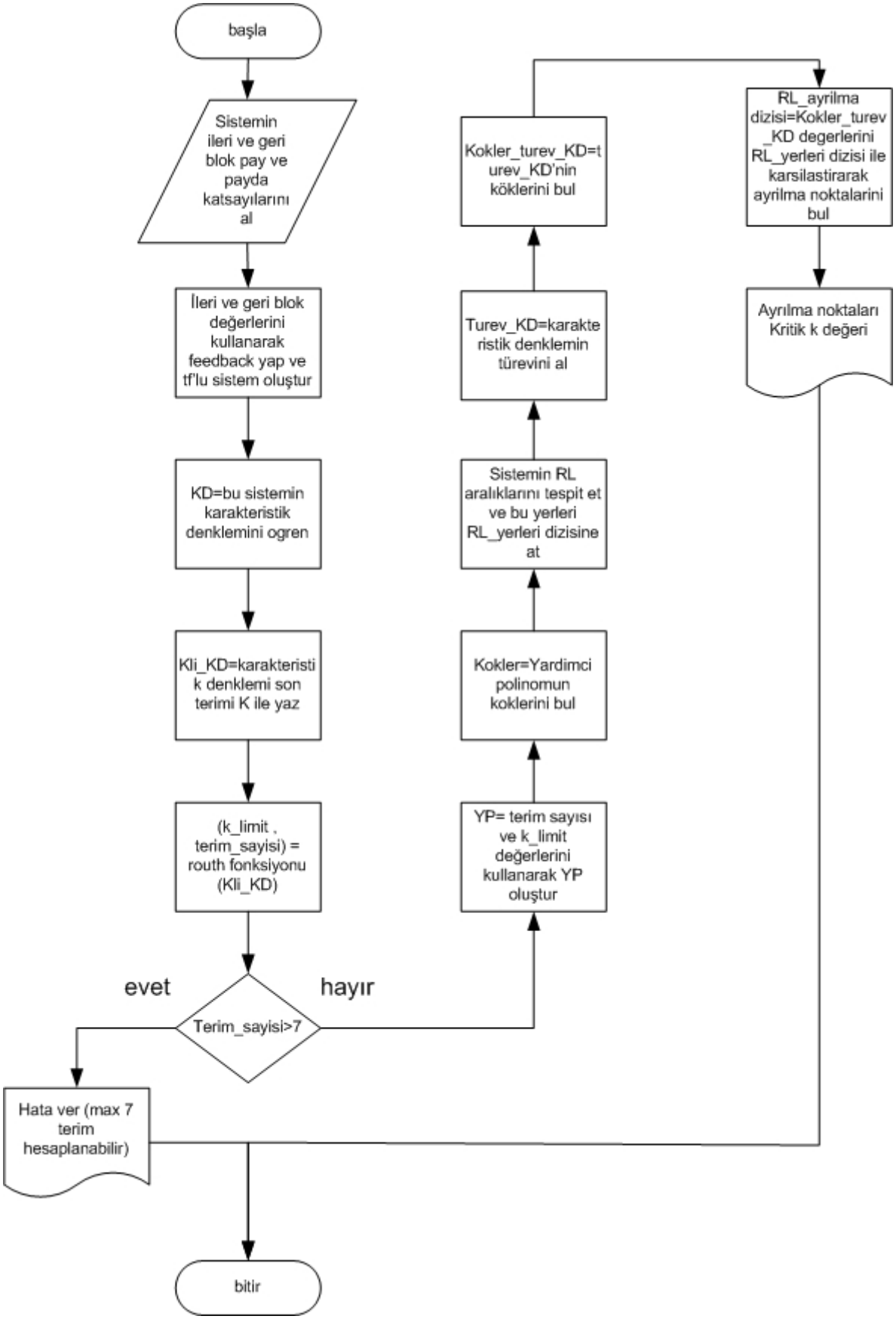
Şekil 6.6 Uygulama 1 Kullanım Ekranı 5

6.1.5 Uygulamanın Algoritması

Bu uygulamada açık çevrim sistem değerlerinin hesaplanması ile ilgili algoritma Algoritma 6.1.'de gösterilmiştir. Ayrıca, bir sistemin kök-yer eğrisi çizilirken dikkate alınan kapalı çevrim yapısının kullanırak bulunması gerekli değerlerin bulunması için de Algritma 6.2.'ye bakılabilir.



Algoritma 6.1 Uygulama 1 Algoritması 1



Algoritma 6.2 Uygulama16 Algoritması 2

6.2 Uygulama-2'nin Tanıtılması

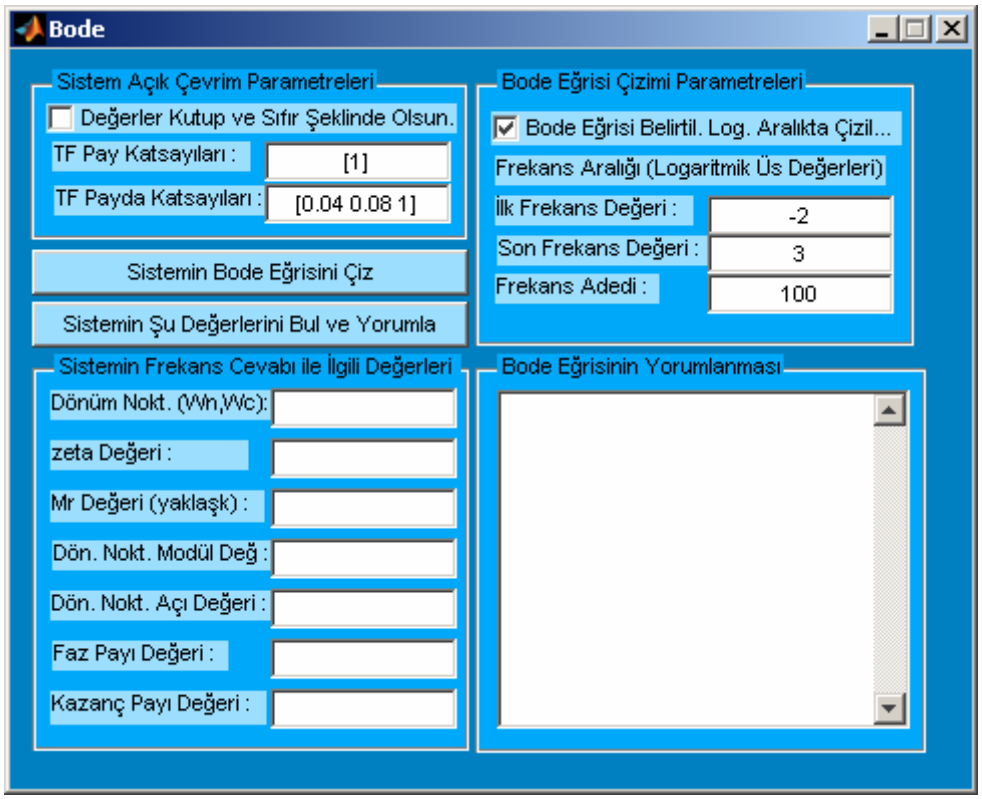
6.2.1 Uygulamanın Adı : Bode Eğrisinin Çizimi ve Hesaplamalarının Yapılması

6.2.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerine uygulanan frekans değerine bağlı olarak sistemin kararlığının nasıl olacağı hakkında bilgi edinilmesini sağlayan yöntemlerden biri olan Bode eğrisi ile ilgili bilinmeyen parametrelerin kendi içerisinde yer alan formülasyon kullanılarak hesaplanması ve bu eğrinin çizilerek kullanıcıya gösterilmesini sağlamaktır. Ayrıca, Bode eğrisi ile ilgili hesaplamaların programlanarak nasıl elde edilebileceği konusunda kullanıcıları bilgilendirmek amaçlanmıştır.

6.2.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.1'de görülmektedir.



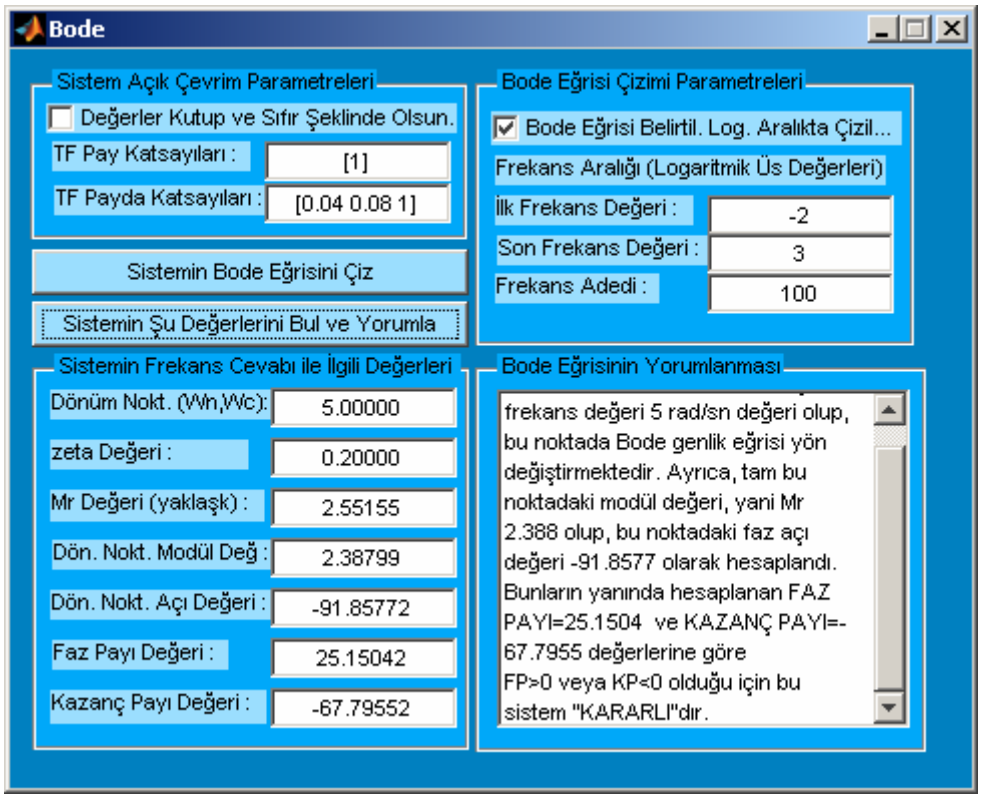
Şekil 6.7 Uygulama 2 Arayüzü

6.2.4 Uygulamanın Kullanılışı

Uygulamanın kullnılabilmesi için öncelikle kullanıcının iBode eğrisi analizi yapacağı sisteme ait açık çevrim transfer fonksiyonunu girmelidir. Bunun için uygulamanın "Sistem Açık Çevrim Parametreleri" bölümü kullanılarak tranfer fonksiyonunun pay ve pay katsayıları girilebilir. İstenirse "Değerler Kutup ve Sıfır Şeklinde Olsun" seçeneği işaretlenerek pay ve payda katsayıları yerine sitemin sıfır ve kutup değerleri girilebilir. Kullanıcı çizdirilecek Bode eğrisinin çizileceği frekans logaritmak aralığına ait sınırları ve hangi artım miktarıyla çizileceğini belirleyebilir. Bu işlem için kullanıcı "Bode Eğrisi Beliritilen Logaritma Aralığında Çizilsin" seçeneğini seçmelidir. Eğer bu seçeneği seçmezse eğri varsayılan aralık değerleriyle çizdirilecektir. Aralık değerlerini girmek için "Bode Eğrisi Çizimi Parametreleri bölümünden" aralığın ilk ve son frekans değeri LOG10 tabanı baz alınarak tamsayı değerleri şeklinde girilmelidir. Ayrıca, bu aralıkta kaç tane değer olacağı da "Frekans Adedi" kutucuğuna yazılmalıdır. Bu değer arttıkça eğri daha hassa ve doğru olarak çizilecektir. Programda örnek olarak varsayılan değerler transfer fonskyonu için pay " [1] " ve payda " [0.04 0.08 1] " olarak seçilmiştir. Programın anlatılmasında örnek olması amacıyla bu değerler kullanılacaktır. Uygulama temel olarak 2 farklı işleve sahiptir. Bu işlevlerin programın kullanılarak nasıl gerçekleştirileceği ile ilgili ayrıntılı bilgiler aşağıda verilmiştir.

6.2.4.1 Bode Eğrisi Çizi ile İlgili Değerlerin Bulunması ve Eğrinin Yorumlanması

Bu işlem için kullanıcı "Sistemin Şu değerlerini Bul ve Yorumla" butonuna kullanmalıdır. Bu işlem gerçekleştirildiğinde hesaplanan değerler örnek sistem için Şekil 6.8'de gösterilmiştir.

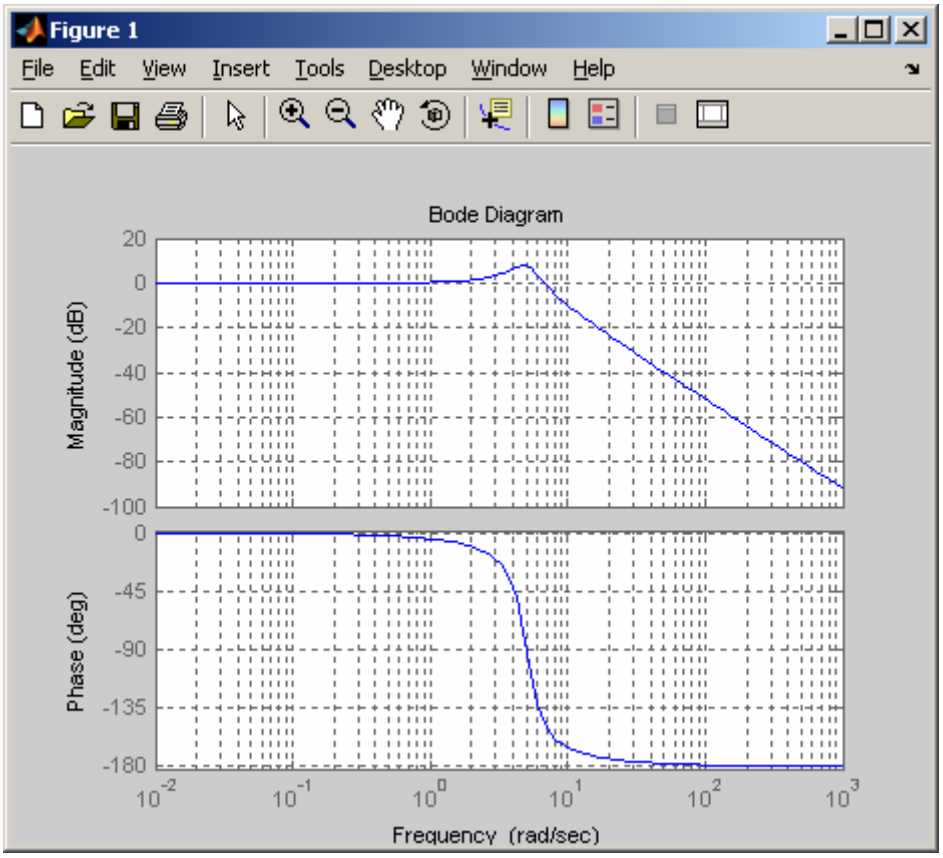


Şekil 6.8 Uygulama 2 Kullanım Ekranı 1

Ayrıca, programda burada antılan işlevin geröekleşrtirilmesi ile bulunan FAZ PAYI ve KAZANÇ PAYI değerleri yorumlanarak sistemin KARARLI ya da KARARSIZ olup olmadığı yorumunu program yapmaktadır.

6.2.4.2 Bode Eğrisinin Çizdirilmesi

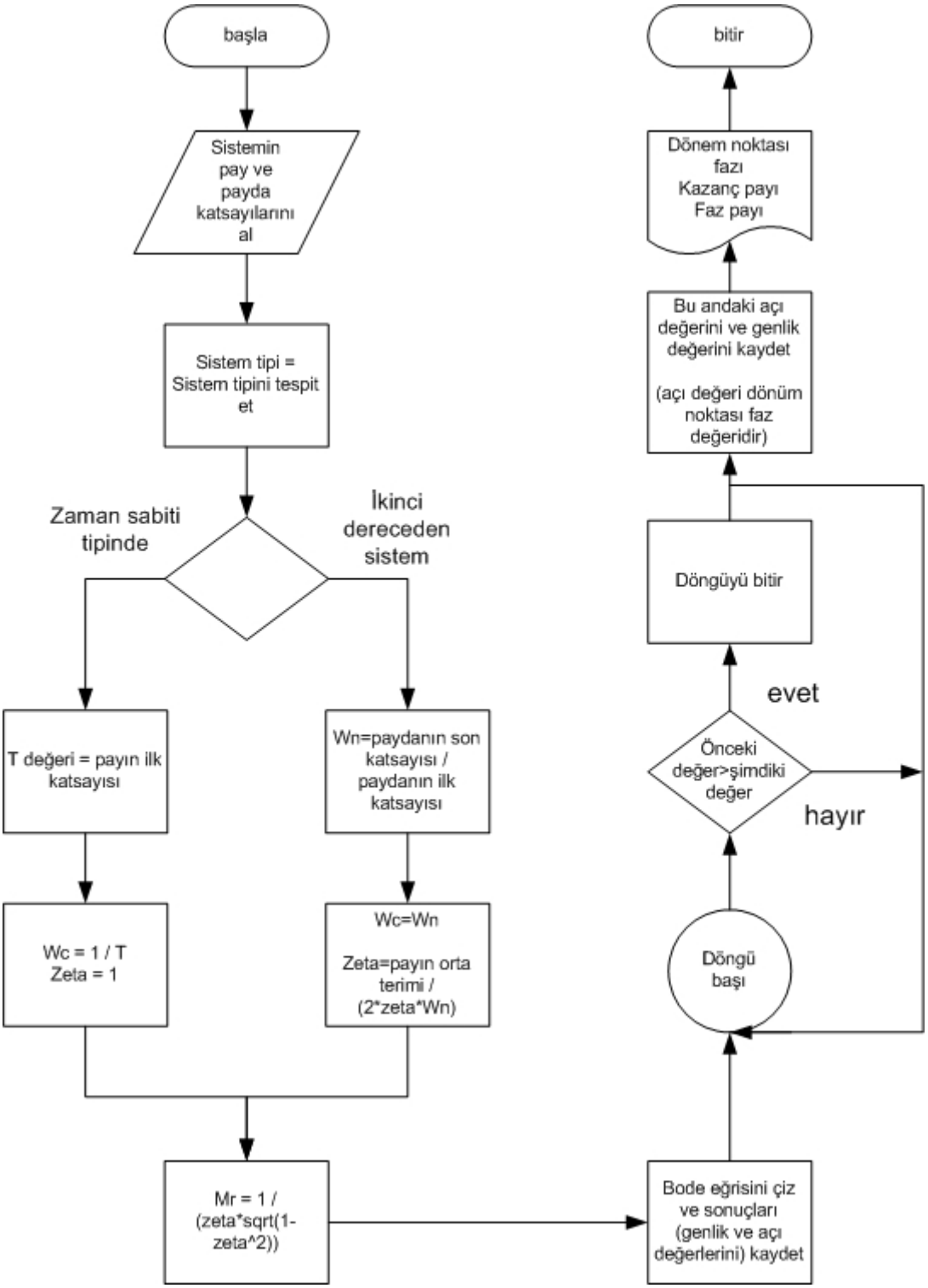
Bu işlem için "Sistemin Bode Eğrisini Çiz" butonuna basılmalıdır. Daha sonra ikinci ayrı bir pencere açılacak ve bu pencerede önceden beliritilen sistemin Bode eğrisi çizdirilecektir. Bu pencere Şekil 6.9'da gösterilmiştir.



Şekil 6.9 Uygulama 2 Kullanım Ekranı 2

6.2.5 Uygulamanın Algoritması

Bu uygulamada Bode eğrisi kullanılarak hesaplanan faz payı, kazanç payı ve sistemin dönüm noktası gibi değerlerin hesaplanması ile ilgili algoritma Algoritma 6.3'te gösterilmiştir.



Algoritma 6.3 Uygulama 2 Algoritması

6.3 Uygulama-3'ün Tanıtılması

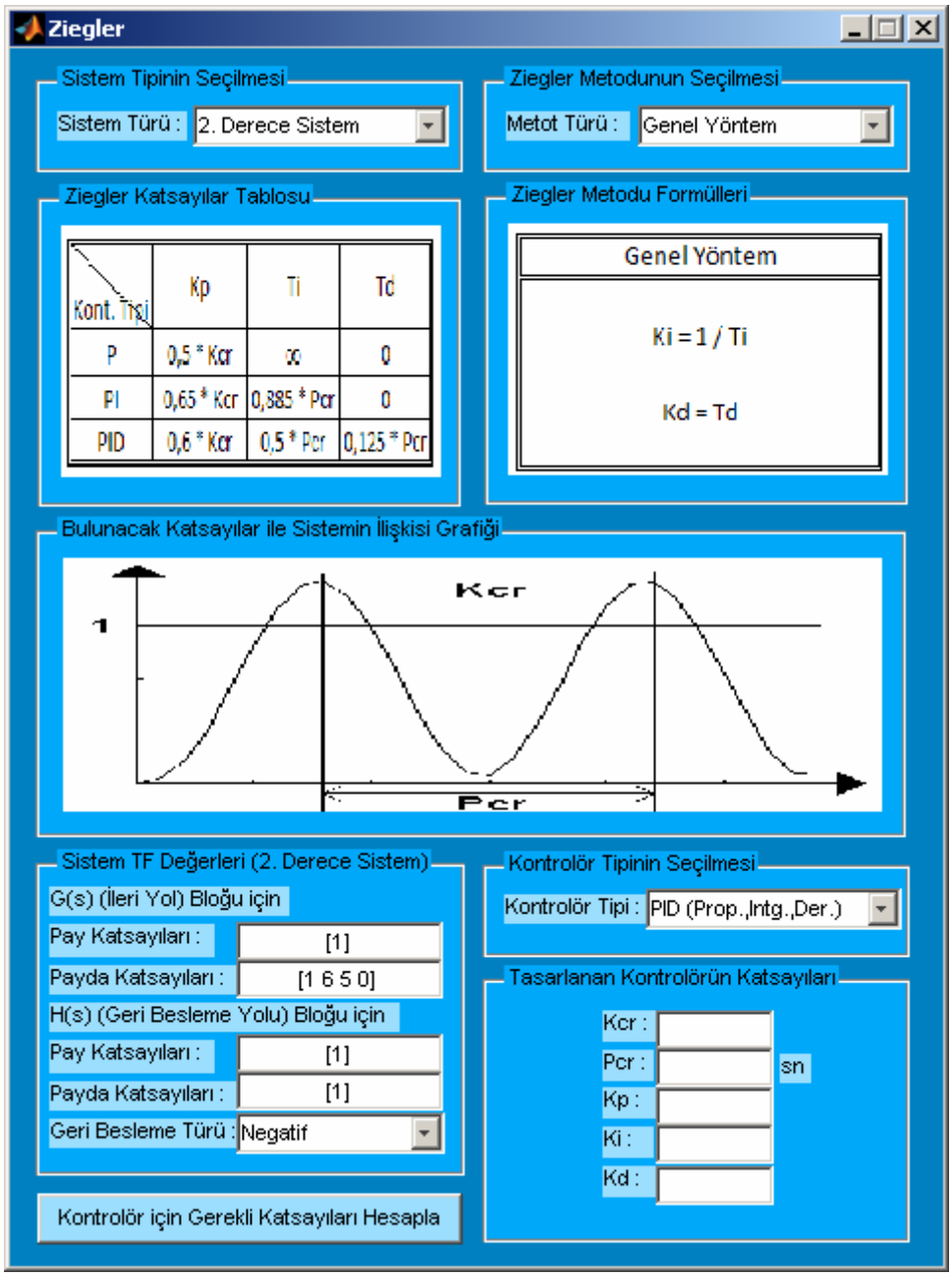
6.3.1 Uygulamanın Adı : Ziegler-Nichols Yöntemi Uygulaması

6.3.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerini kontrol etmek üzere tasarlanacak oransal, oransal ve integral veya PID denetleyicilere ait Kp, Ki, ve Kd gibi kontrolör katsayılarının bulunması amacıyla Ziegler-Niichols yönteminin kullanılarak hesaplanması ve bu şekilde kullanıcıların zaman harcamalarını engellemektir. Ayrıca, yöntemin nasıl programlanacağı konusunda da kullnıcılara bilgi vermek amaçlanmıştır.

6.3.3 Uygulamanın Arayüzü

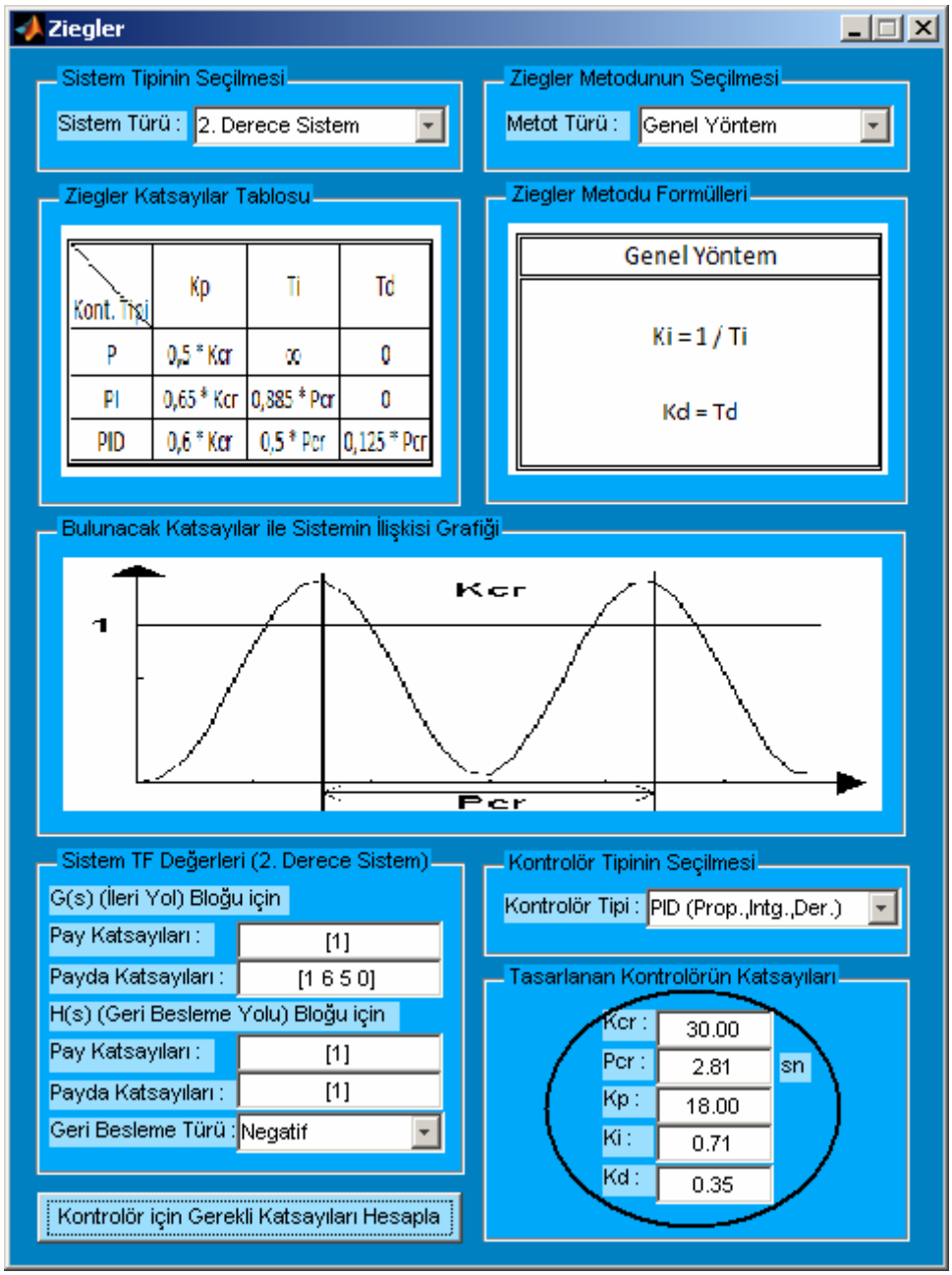
Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.10'da görülmektedir.



Şekil 6.10 Uygulama 3 Arayüzü

6.3.4 Uygulamanın Kullanılışı

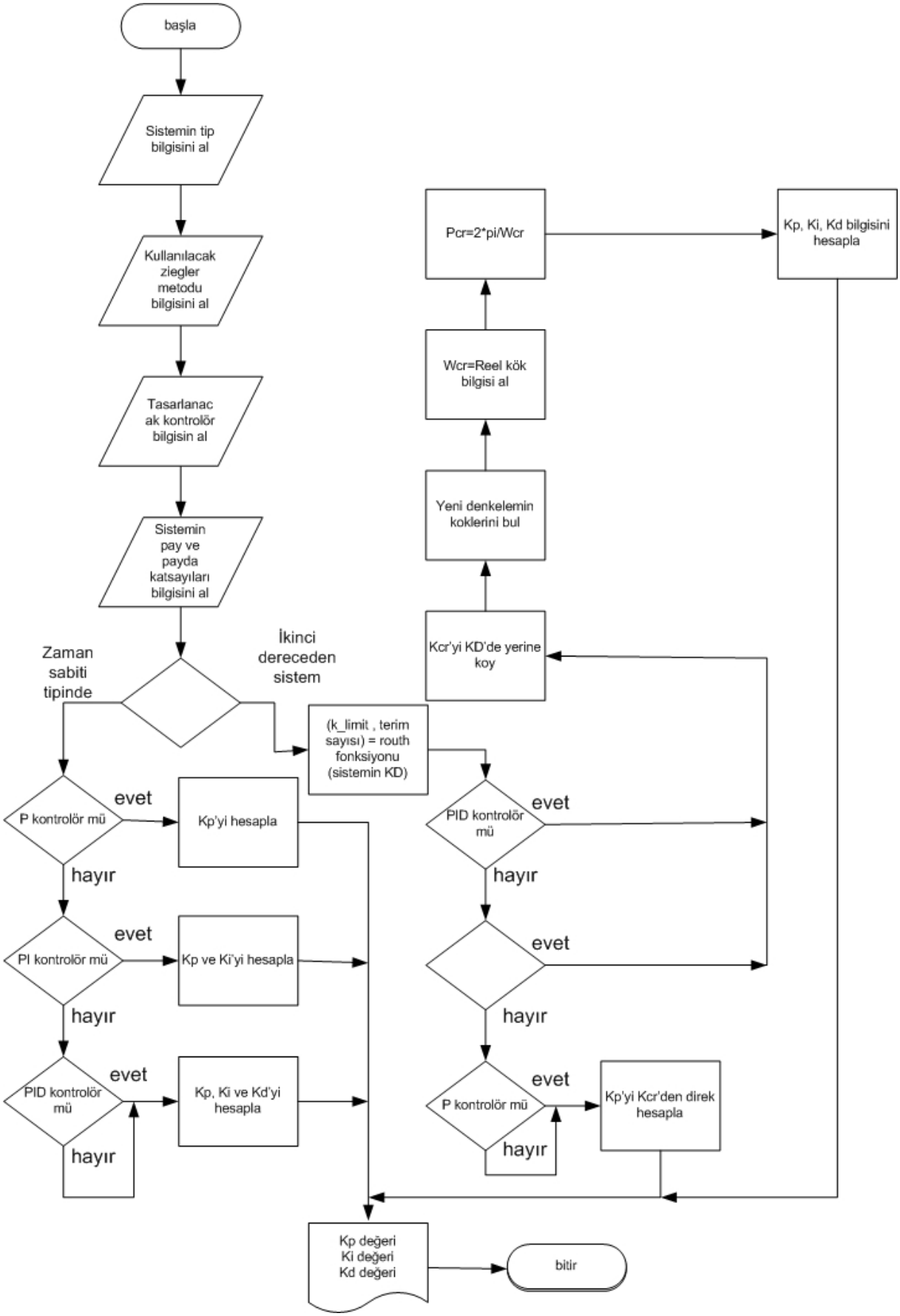
Kullanıcı öncelikle Ziegeler Nichols yöntemini uygulayacağı sistemi "Sistem Tipinin Seçilmesi" bölümünden belirlenemelidir. Burada eğer sistem 2. dereceden ise "2. Derece Sistem" veya zaman sabiti tipinde ise "Zaman Sabiti Tipinde" seçeneği seçilmelidir. Bu seçenekler seçildiğinde programda kullanılacak sistem ile Ziegler-Nichols yönteminin uygulanması sırasında kullanılacak katsayı ve katsayıların nasıl hesaplanacağı konusunda fikir vermesi bakımından program arayüzü dinamik olarak şekillerle desteklenmiş ve bu şekilde kullncılar bilgilendirilmektedir. Sistem türü seçildikten sonra Ziegler-Nichols yönteminde kullanılacak metod türünün seçilmesi gerekmektedir. İlgili metod seçildiğinde hemen aşağısında yer alan Şekil'de bu yöntemin hangi formülasyonu kullanıdığı dinamik olarak kullnıcıya sunulmatadır. Daha sonra kullanıcıya seçtiği sistem ile ilgili ve Ziegler-Nichols yönteminin kullanacağı katsayıları girmelidir. Bunun için programın "Sistem T.F. Değerleri" bölümü kullanılmalıdır. En son olarak kullanıcı tasarlanacak denetleyici tipini belirlemelidir. Kontrolör tipi olarak oransal, oransal ve integral veya PID seçeneklerin birini programın "Kontrolör Tipinin Seçilmesi" bölümünde yer alan listeden seçmelidir. Daha sonra "Kontrolör için Gerekli Katsayıları Hesapla" butonu kullanılarak Ziegler-Nichols yöntemine göre seçilen denetleyici türü için gerekşi, katsayılar hesaplanmaktadır. Şekil 6.10'daki arayüzde verilen değerler kullanılarak PID bir kontrolör tasarımı için hesaplanan değerler örnek olması amacıyla Şekil 6.11'de gösterilmiştir.



Şekil 6.11 Uygulama 3 Kullanım Ekranı

6.3.5 Uygulamanın Algoritması

Bu uygulamada Ziegler-Nichols yöntemi kullanılarak gerekli kontrolör katsayılarının nasıl hesaplanmadığını gösteren algoritma için Algoritma 6.4'e bakılabilir.



Algoritma 6.4 Uygulama 3 Algoritması

6.4 Uygulama-4'ün Tanıtılması

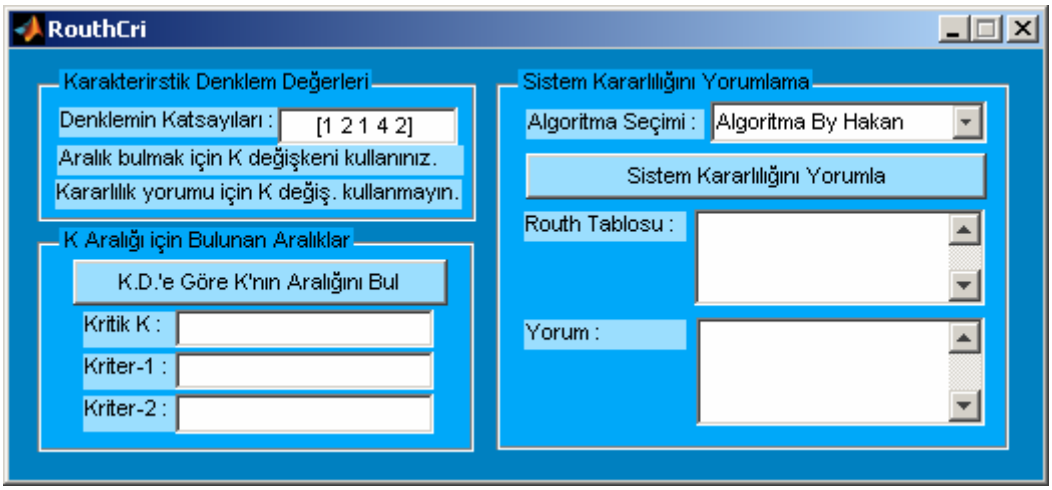
6.4.1 Uygulamanın Adı : Routh Kriteri Uygulaması

6.4.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerinin mutlak kararlılığı hakkında bilgi vermekte kullanılan yöntemlerden biri olan Routh krtiterinin kullanılarak kullanıcıların ekstra işlem yapmadan ve zaman harcamadan girilen bir sistemin kararlığı hakkında bilgi sahibi olmalarını ve girilen sistme ait Routh tablosunun öğrenilmesi sağlanmak istenmiştir. Ayrıca, yöntemin nasıl programlanacağı konusunda da kullanıcılara bilgi vermek amaçlanmıştır.

6.4.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.12'de görülmektedir.



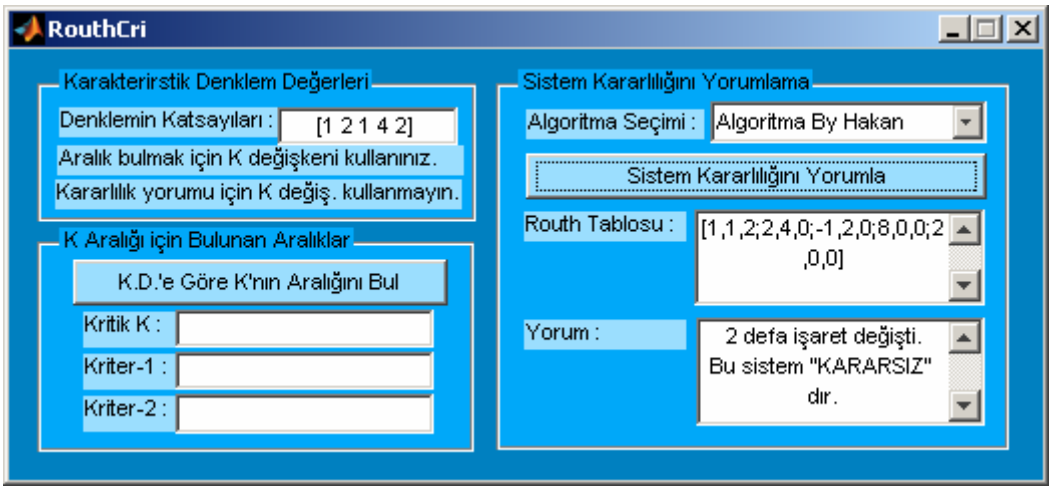
Şekil 6.12 Uygulama 4 Arayüzü

6.4.4 Uygulamanın Kullanılışı

Uygulamayı kullanabilmek için öncelikle kullnıcıların Routh kriteri uygulanacak sistemin karakteristik denkleminea ait katsayıların programa girilmesi gereklidir. Burada dikkat edilmesi gereken yapılacak işleme göre katsayıların programa girilmesinin farklı yazıma sahip olduğudur. Örneğin kullnıcılar uygulama ile kararlılığı yorumlamak isterlerse " [1 4 5 6] " formatında karakteristik denkleme ait katsayıları programa girebilirler. Ancak, programın kullanılmasındaki amaç Routh kriteri ile kritik bir K değerinin bulunması ise bu takdirde [1 2 1 4 2 K] , [1 2 1 4 2 K+2], [1 2 1 4 2 K-5], [1 2 1 4 2 5+K] veya [1 2 1 4 2 8*K] gibi örnek yazımlardan birini kullanmalıdırlar. Program temel olarak 2 farklı işlevi yerine getirmektedir. Bu işlevlerin program kullanılarak nasıl yerine getirileceği ile ilgili gerekli bilgiler aşağıda verilmiştir.

6.4.4.1 Sistemin Mutlak Kararlılığının Yorumlanması

Bu işlem için kullanıcılar öncelikle programın Routh kriterini uygularken programın kullanacağı algoritmalardan birini programın "Sistem Kararlılığını Yorumlama" kısmında yer alan "Algoritma Seçimi" kısmı kullanılarak açılan listeden seçmeleri gerekmektedir. Bu programda iki farklı algoritma kullanılmıştır. Düşük mertebeli sistemler (5. ve 6. mertebeye kadar) için "Algoritma By Kenan" seçeneği kullanılabilirken yüksek mertebeler ve düşük mertebeler için (10. mertebeye kadar) "Algoritma By Hakan" seçeneği kullanılabilir. Daha sonra kullanıcılar programda yer alan "Sistem Kararlılığınık Yorumla" butonunu kullanarak girilen sistemin Routh tablosunu "Rout Tablosu" bölümünden ve bu tablo ile ilgili yorum için ise programın "Yorum" bölümünden gerekli bilgileri öğrenebilirler. Örnek verilen sistem için sistem kararlılığı ile ilgili bilgi edinildiğinde Şekil 6.13'te programın sahip olduğu ekran görüntüsü gösterilmiştir.

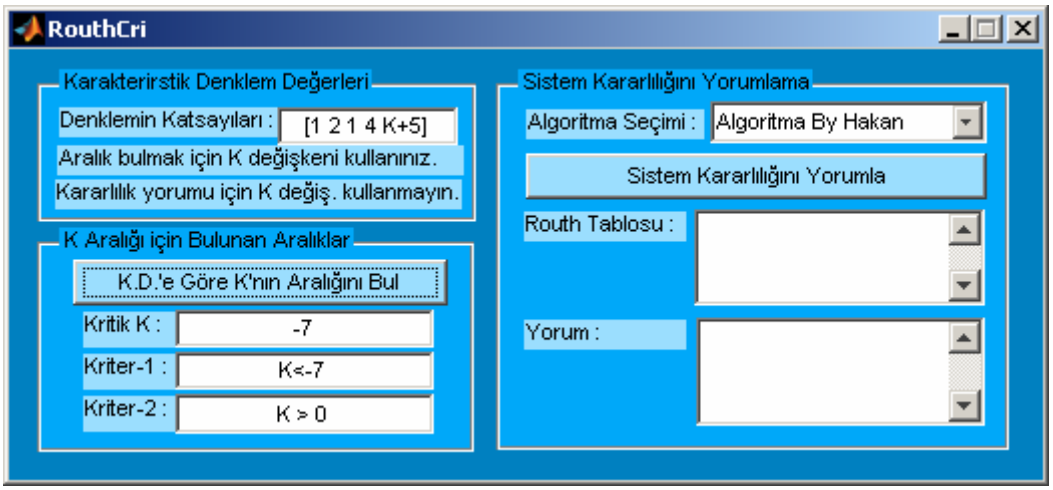


Şekil 6.13 Uygulama 4 Kullanım Ekranı 1

Örnek sistem için Routh tablosunda da görüldüğü üzere tablonun ilk sütununda işaret değişimi olduğu için bu sistem "KARARSIZ" olarak yorumlanmıştır.

6.4.4.2 Karakteristik Denklemde Yer Alan K Değeri için Aralığın Tespit Edilmesi

Bu işlem için kullanıcıların sistemin kullanacağı K değerini de içeren katsayıları programa girmiş olması gerekir. Bu durum ile ilgili ayrıntılı bilgiye 6.4.4 konu başlığından ulaşılabilir. Kritik K değerinin bulunması için kullanıcılar programda "Karakteristik Denkleme Göre K Aralığını Bul" butonunu kullanmalıdırlar. Örnek olarak "[1 2 1 4 K+5] " katsayılarını içeren bir karakteristik denklem için bulunan K aralığının hesaplandığı program ekran görüntü Şekil 6.14'te gösterilmiştir.

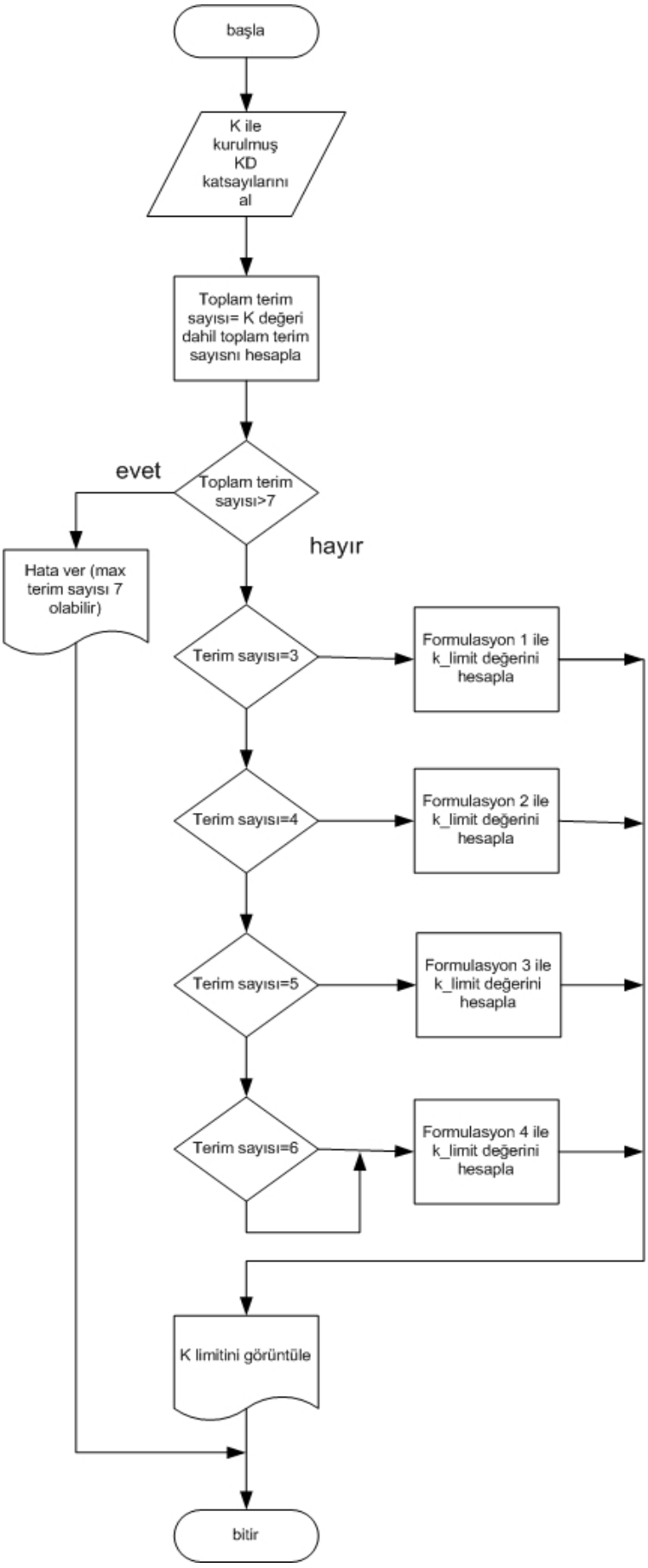


Şekil 6.14 Uygulama 4 Kullanım Ekranı 2

Programda örnek girilen karakteristik denklem katsayıları için "K<-7" ve "K>0" kriterleri bulunmuştur. Ayrıca, program arayüzünde hesaplanan kritik K değeri -7 olarak gösterilmektedir.

6.4.5 Uygulamanın Algoritması

Bu uygulamada kullnılan algoritmalardan "Algoritma By Kenan" bu projeyi hazırlayan kişi tarafından tasarlanmıştır. Diğer seçenek olan "Algoritma By Hakan" seçeneğine ait algoritma Hakan AYDIN tarafından tasarlanmış olup sadece "Algoritma By Kenan" yönteminin nasıl uygulandığı ile ilgili algoritma, Algoritma 6.5'ten öğrenilebilir.



Algoritma 6.5 Uygulama 4 Algoritması

6.5 Uygulama-5'in Tanıtılması

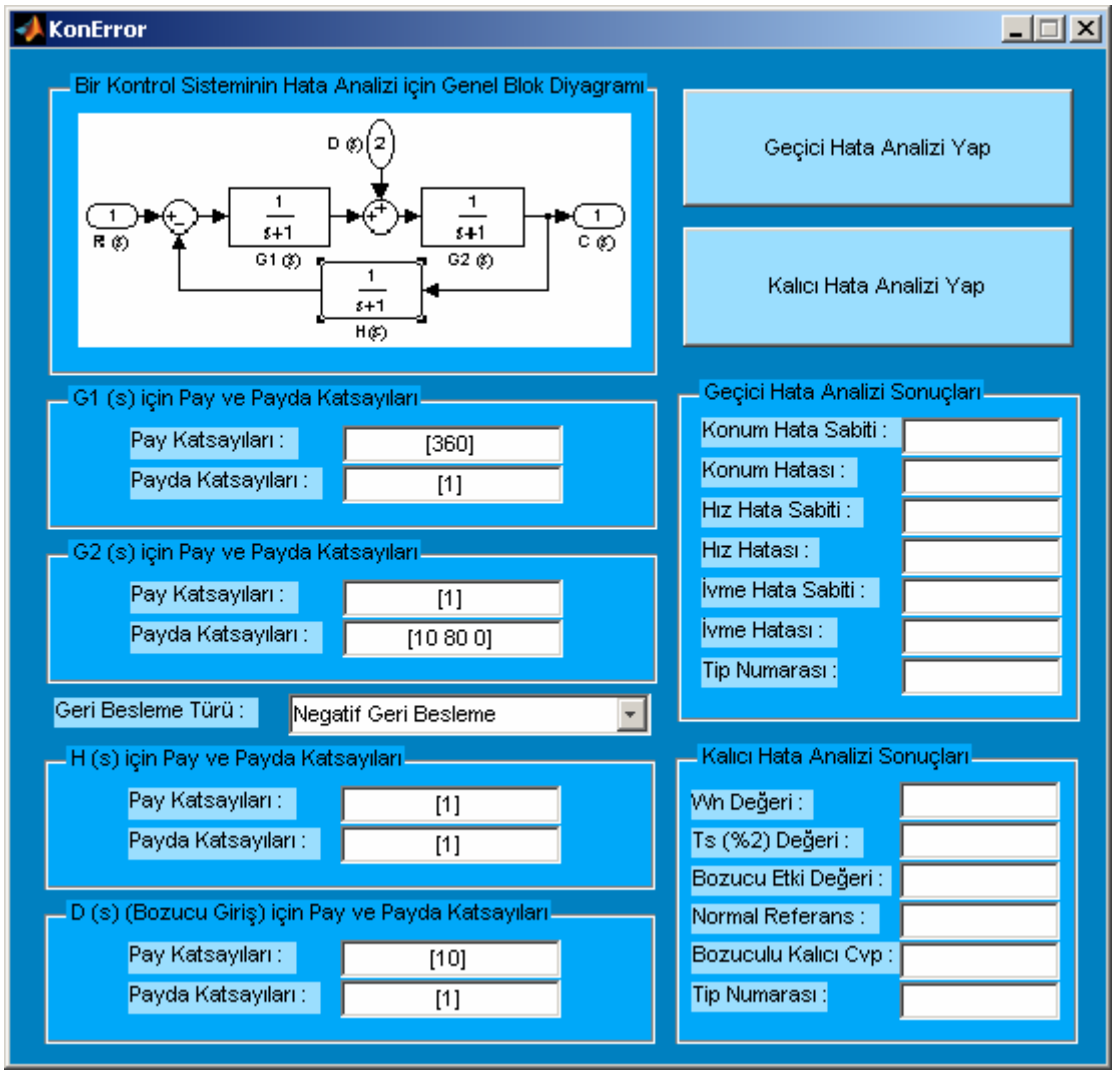
6.5.1 Uygulamanın Adı : Kontrol Sistemleri için Geçici ve Kalıcı Hata Analizi Programı

6.5.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerinde geçici ve kalıcı rejime ait hataların genel sistem mantığı göz önünde bulundurularak kullanıcılar tarafından çok kısa sürede ve karmaşık işlemleri yapma gereği bırakmadan hesaplanması, konum, hız, ivme gibi hata değerleri hakkında bilgilendirilmesi ve bu programda gerekli hesaplamanın nasıl programlandığı konusunda kullanıcılara bilgi vermek amacını taşımaktadır.

6.5.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.15'te görülmektedir.



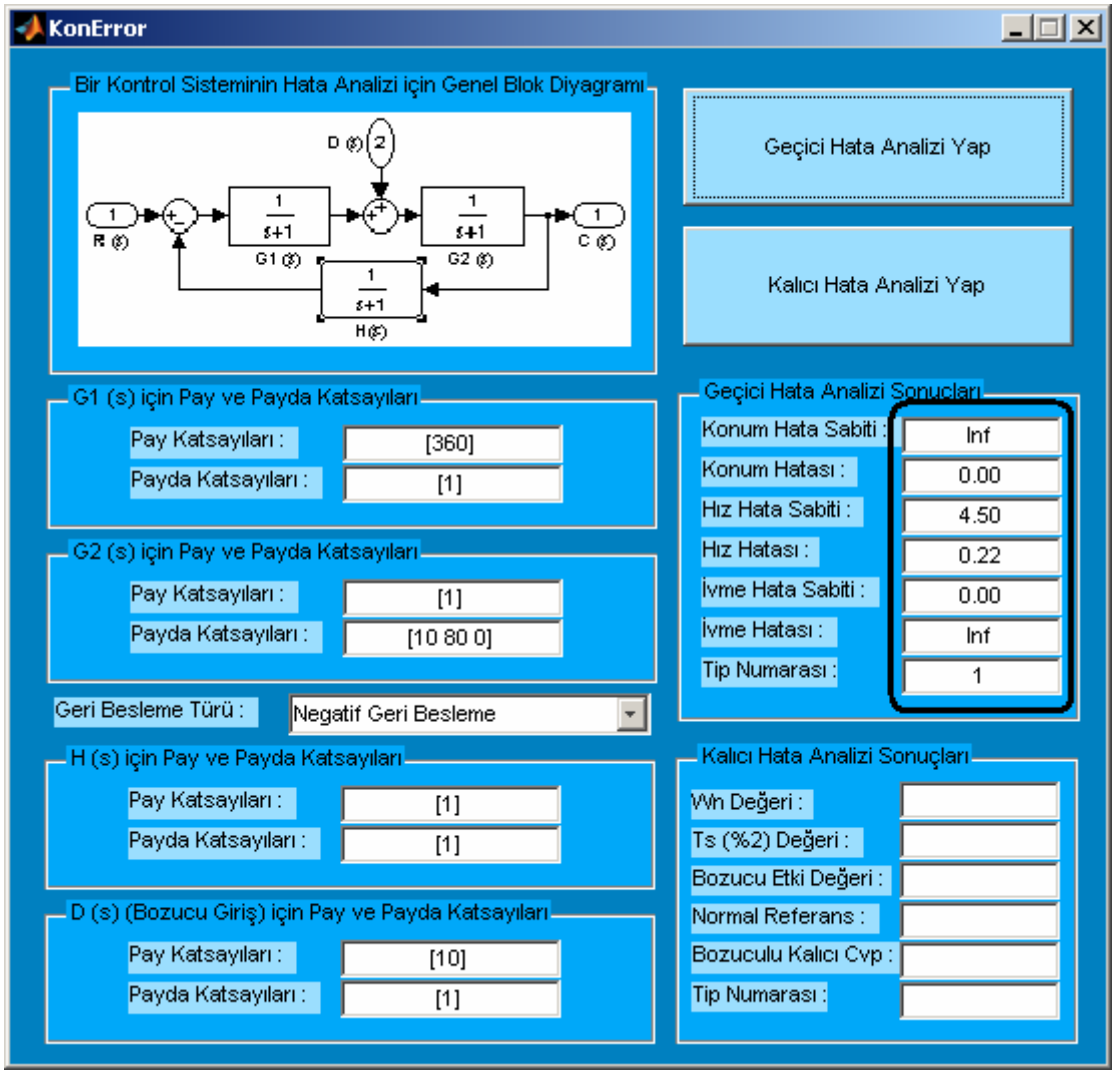
Şekil 6.15 Uygulama 5 Arayüzü

6.5.4 Uygulamanın Kullanılışı

Bu programda Şekil 6.15'te de arayüzde de görüldüğü üzere G1(s) ve G2(s) olmak üzere 2 adet ileri isistem ve bu iki sistem arasına konulmuş 1 adet bozucu giriş etkisi ile toplam ileri sistem için 1 adet H(s) adında geri besleme bloğu olması göz önünde bulunduurlmuştur. Dolayısıyla kullanıcıların program ile gerekli hata analiz sonuçlarını hesaplaması için bu bloklara ait değerlerin girilmesi gerekir. Değerler pay ve payda şeklinde ve matris yapısı formatında (örneğin " [10 80 0] " gibi.) girilmelidir. Ayrıca, kullanıcılar geri beslemenin pozitif veya negatif olması programda seçebilirler. Bozucu giriş için eğer tek bir katsayı değeri girilecek ise payda için katsayı olarak " [1] " girlmesi yeterlidir. Eğer, bozucu için 2 veya daha fazla terim girilirse bu durum s değişikeni için bir katsayı olarak işlem görecektir. Bu program temel olarak iki farklı işlevi yerine getirmektedir. Bu ilevler ile ilgili olarak programın nasıl kullanılacağı hakkında ayrıntılı bilgiler aşağıda verilmiştir.

6.5.4.1 Geçici Hata Analizi Yapılması

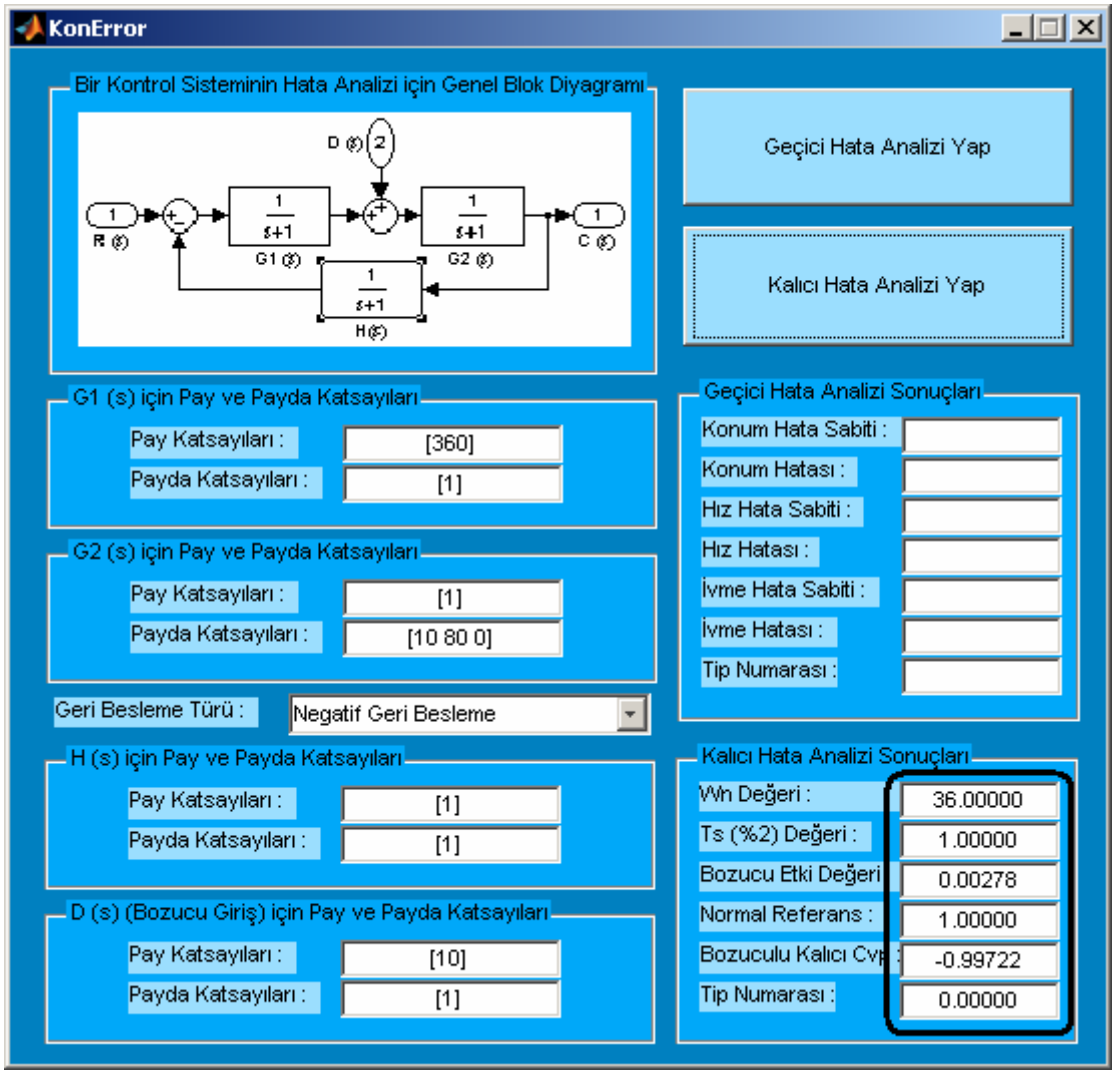
Program için gerekli tüm değerler girildikten sonra sistemin geçici rejimi hakkında bilgi sahibi olmak için programda yer alan "Geçici Hata Analizi Yap" butonunun tıklanması gerekmektedir. Bunun ardından hesaplanan değerler programın "Geçici Hata Analizi Sonuçları" bölümünde kullanıcıya gösterilir. Örnek olarak varsayılan program değerleri ile geçici hata analizi yaptığımızda program ekran görüntüsü Şekil 6.16'da gösterilmiştir.



Şekil 6.16 Uygulama 5 Kullanım Ekranı 1

6.5.4.2 Kalıcı Hata Analizi Yapılması

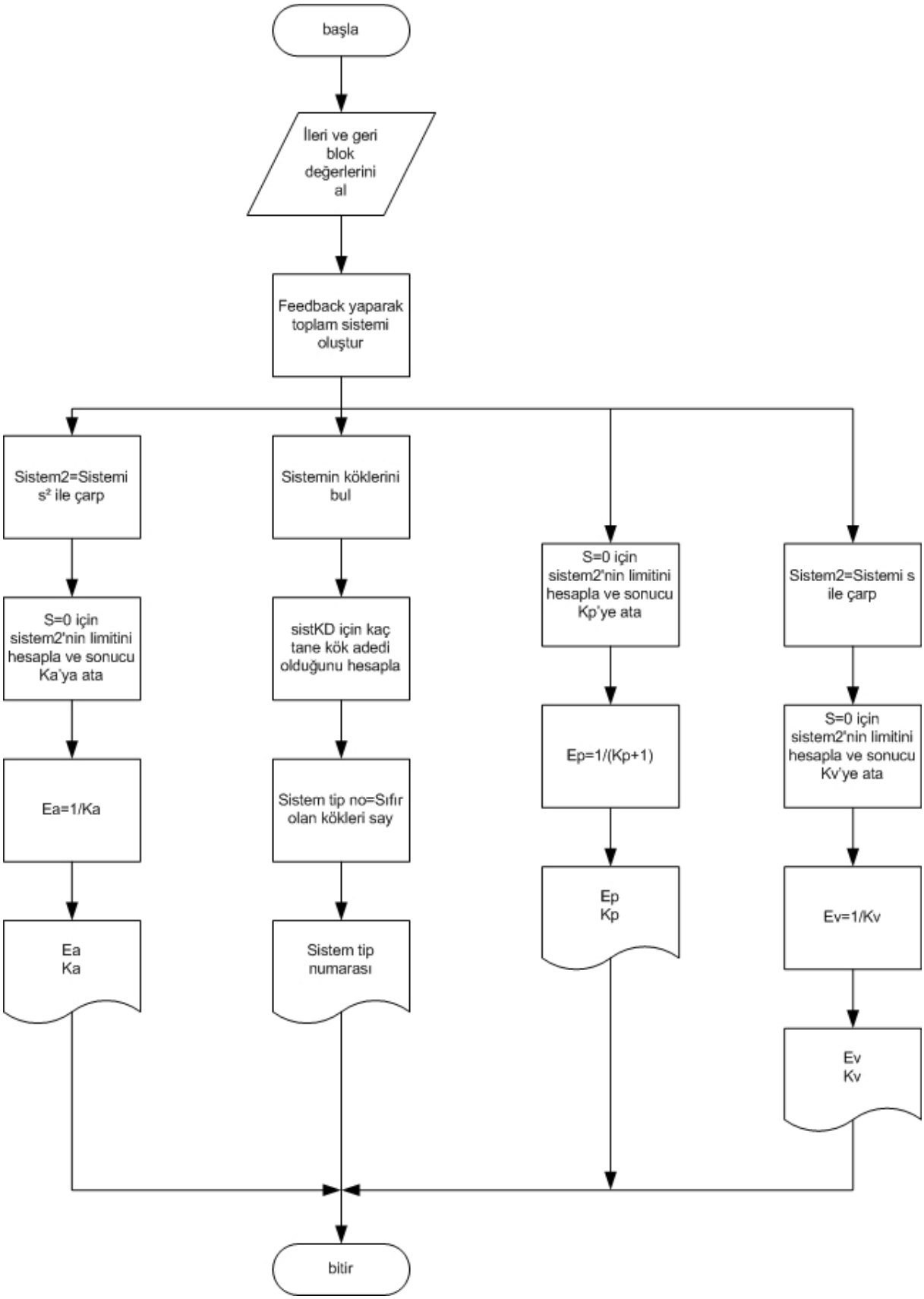
Program için gerekli tüm değerler girildikten sonra sistemin kalıcı rejimi hakkında bilgi sahibi olmak için programda yer alan "Kalıcı Hata Analizi Yap" butonunun tıklanması gerekmektedir. Bunun ardından hesaplanan değerler programın "Kalıcı Hata Analizi Sonuçları" bölümünde kullanıcıya gösterilir. Örnek olarak varsayılan program değerleri ile geçici hata analizi yaptığımızda program ekran görüntüsü Şekil 6.17'de gösterilmiştir.



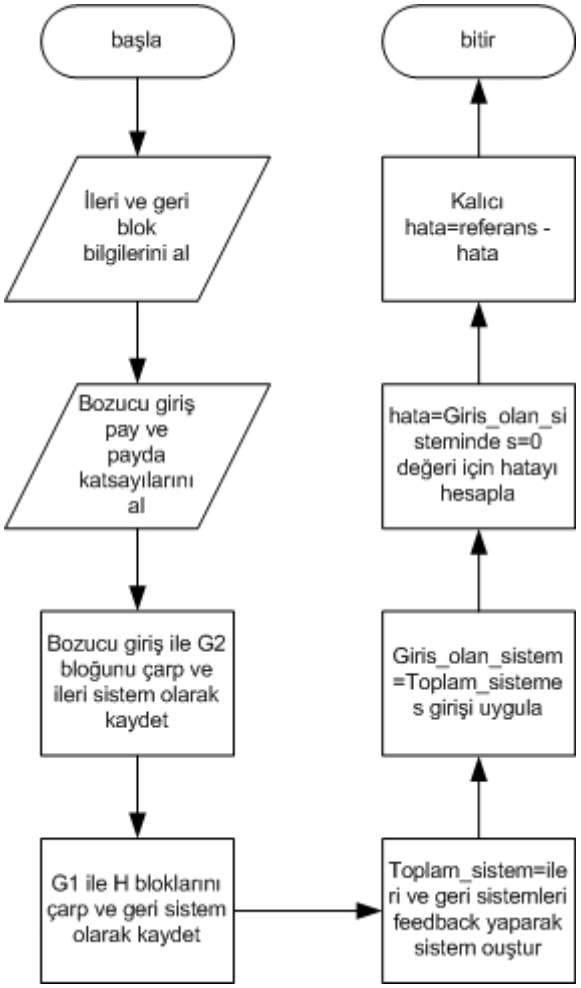
Şekil 6.17 Uygulama 5 Kullanım Ekranı 2

6.5.5 Uygulamanın Algoritması

Bu uygulamada bir kontrol sistemin geçici hata analizi ile ilgili partametrelerin nasıl yapıldığını gösteren algoritma ile ilgili bilgiye Algoritma 6.6'dan erişilebilir. Ayrıca, programın kontrol sistemlerin kalıcı rejimi ile ilgili değerlerin elde edilmesi için kullanılan yöntem Algoritma 6.7'de gösterilmiştir.



Algoritma 6.6 Uygulama 5 Algoritması 1



Algoritma 6.7 Uygulama 5 Algoritması 2

6.6 Uygulama-6'nın Tanıtılması

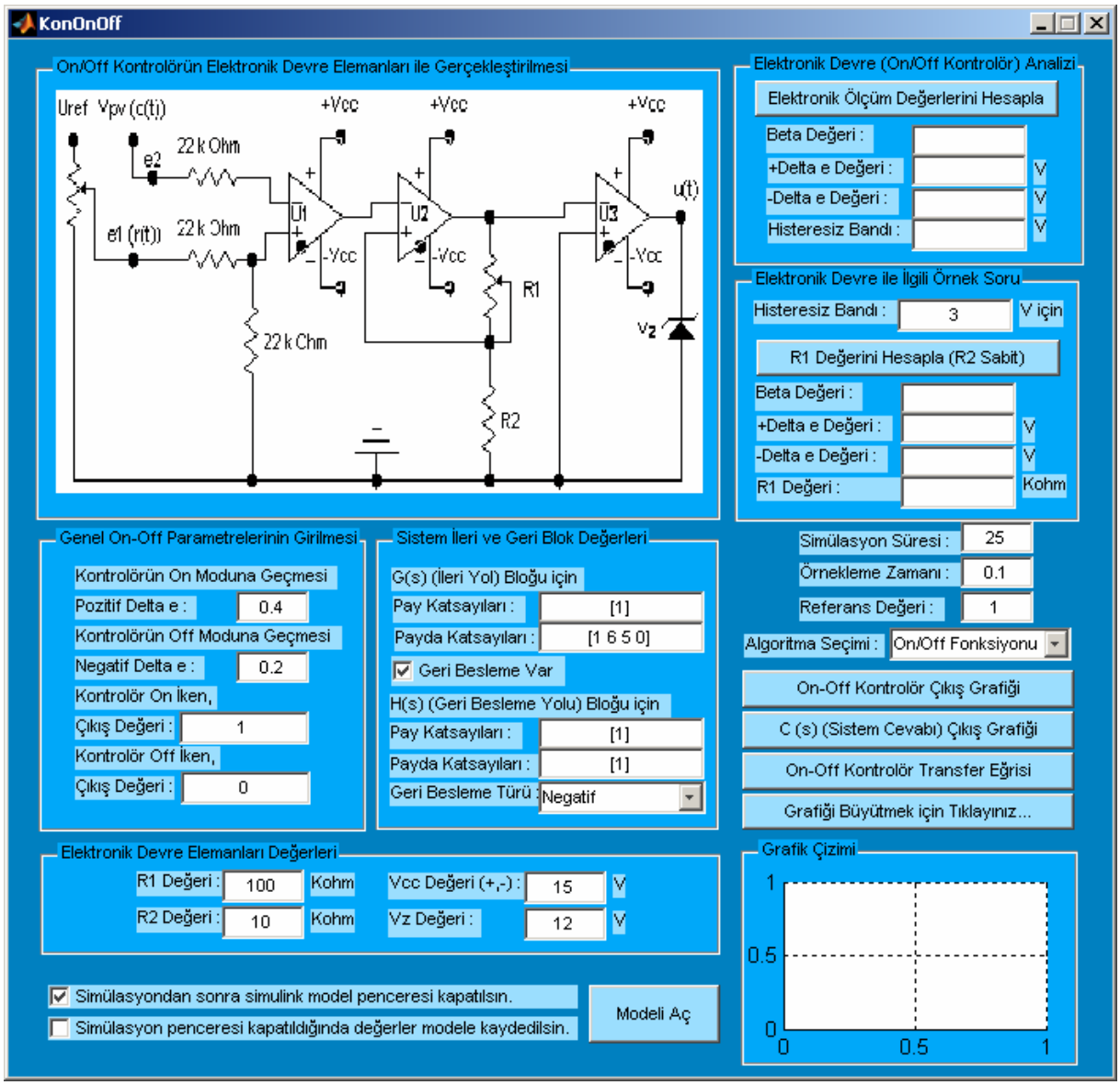
6.6.1 Uygulamanın Adı : On / Off Kontrolör Uygulaması

6.6.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerinde sıklıkla kullanılan on/off denetleyici ile ilgili hesaplamaların gerek elektronik devrelerle tasarım, grekse simülasyon ortamında on/off denetleyicinin çalışması sonucu bir sistemin cevabının nasıl olacağı konusunda kullanıcıları bilgilendirme amacını taşımaktadır. Ayrıca, 6.6 bölümünün sonunda kullanıcılar on/off kontrolör için gerekli algoritma konusunda da bilgilendirilmektedir.

6.6.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.18'de görülmektedir.



Şekil 6.18 Uygulama 6 Arayüzü

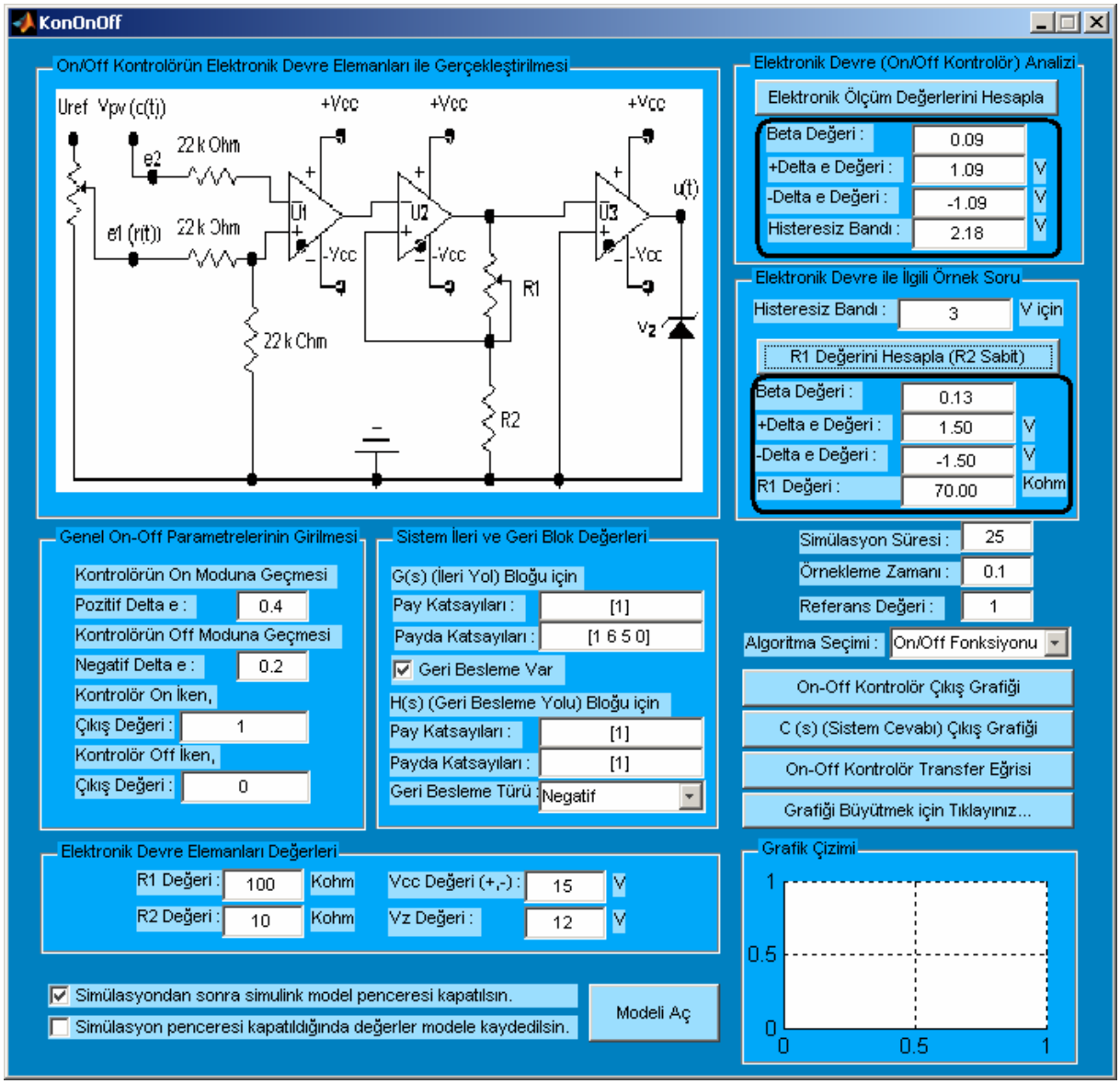
6.6.4 Uygulamanın Kullanılışı

Bu uygulama temel olarak üç farklı işlevi yerine getirmektedir. Bunlardan biri elektronik devre olarak tasarlanan bir on/off denetleyici ile ilgili analiz, diğer analiz olarak bir on/off denetleyicinin simülasyon ortamında çalıştırılarak sistem cevabının görülmesi ve en son olarak da bir on/off denetleyiciye ait on/off transfer eğrisinin kullanıcıya sunulmasıdır. Program arka planda bir simulink modelini kullanarak işlem yapmakta ve simulink modelinin çalışması sonucu üretilen değerleri kullanmaktadır. Bir kullanıcı isterse simulink modelini "Modeli Aç" butonunu kullanarak açabilir. Ayrıca, "Simülasyondan sonra simulink model penceresini kapatılsın." Seçeneği işaretlenmezse kontrolörün simülasyon çalışmasından sonra simulink penceresi kapatılmayacak ve açık kalmaya devam edecektir. Bunun yanında "Simülasyon penceresi kapatıldığında değerler modele kaydedilsin." Seçeneği eğer model kapatılsın seçneği (bir üstteki seçenek) işaretlenirse, oluşturulan sistem değerleri ve girilen parametreler simulink penceresi kapatılmadan önce model dosyasına kaydedilir ve daha sonra

simulink penceresi kapatılır. Bu işlevlerin program ile nasıl gerçekleştirildiği konusunda ayrıntılı bilgiler aşağıda verilmiştir.

6.6.4.1 On/Off Elektronik Devre ile Analiz

On/off denetleyici elektronik devresi ile ilgili değerlerin hesaplanabilmesi için öncelikle elektronik devresine ait değerlerin programın "Elektronik Devre Elemanları Değerleri" bölümünden programa girilmesi gereklidir. Bu değerler girildikten sonra programın "Elektronik Devre Analizi" bölümünden "Elektronik Ölçüm Değerlerini Hesapla" butonuna basıldığında gerekli değerler hesaplanacaktır. Bu değerlerin hesaplanmış hali Şekil 6.19'da gösterilmiştir. Ayrıca, on/off elektronik devresinde R2 sabit kalmak üzere histeresiz bandının

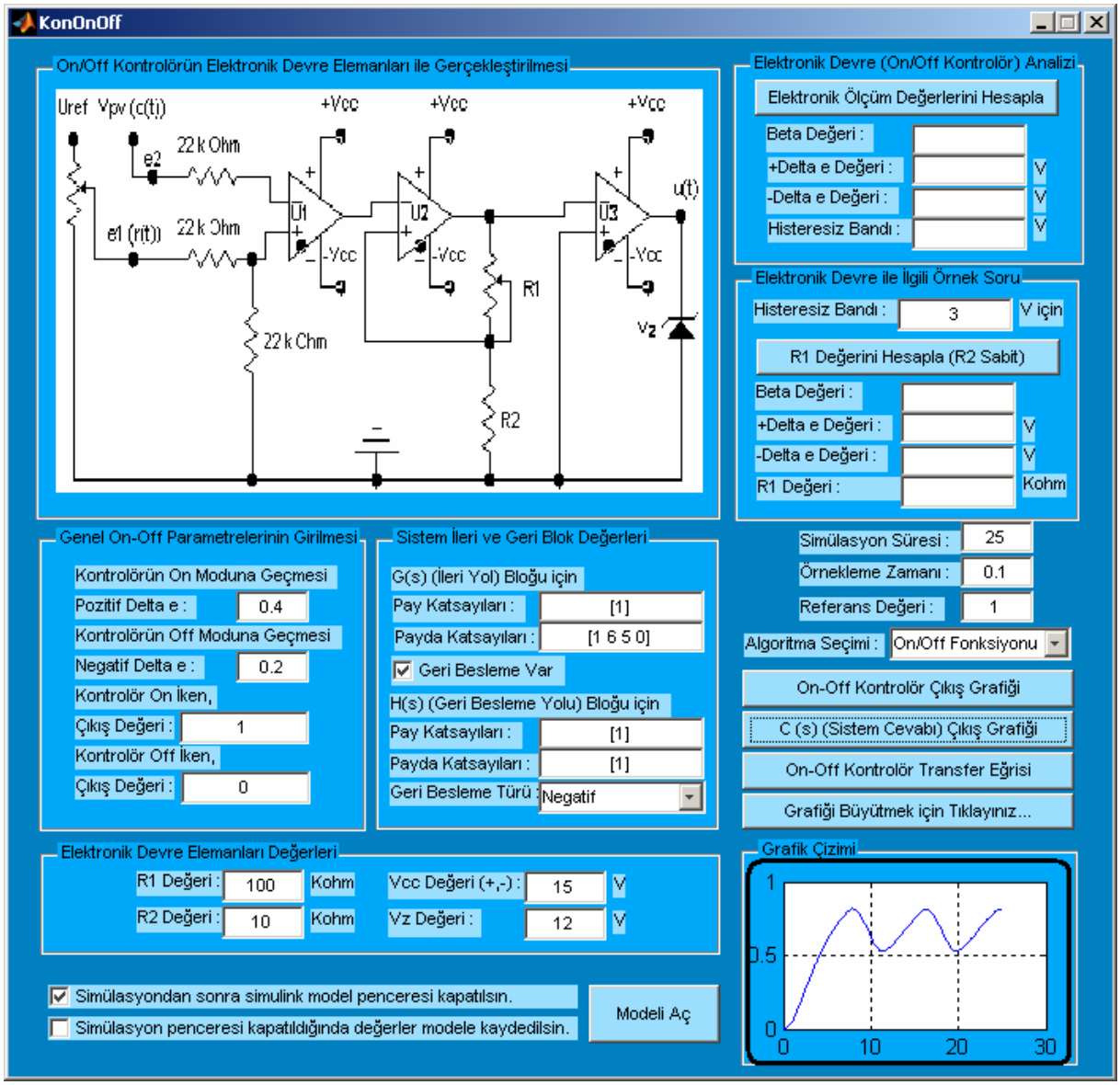


Şekil 6.19 Uygulama 6 Kullanım Ekranı 1

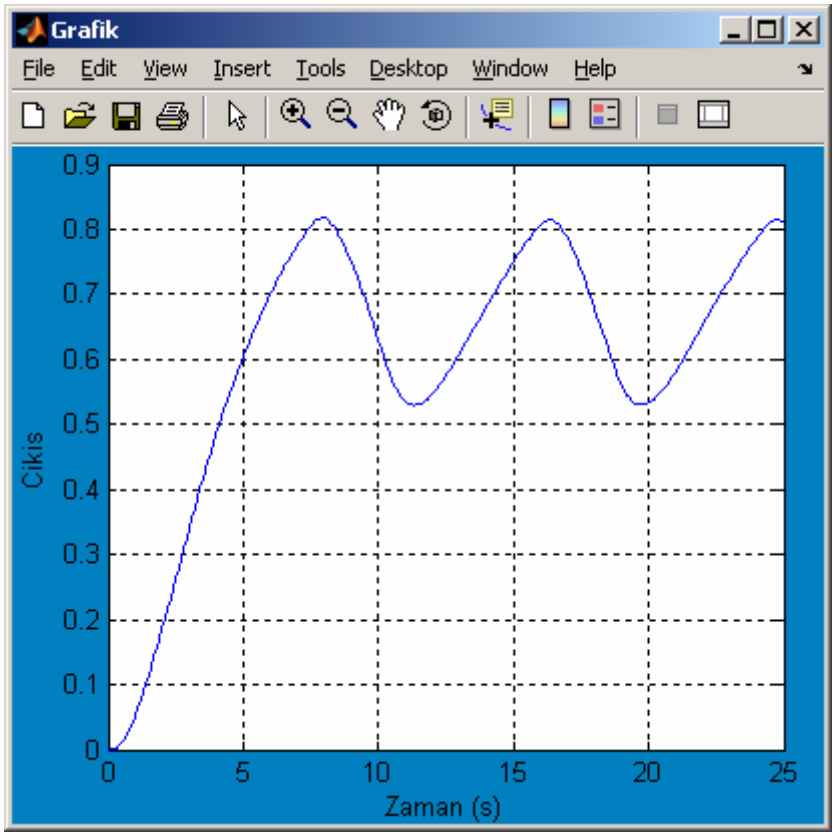
herhangi bir değeri programda yer alan “R1 Değerini Hesapla (R2 Sabit)” butonu kullanılarak hesaplanabilir. Bu durum için de Şekil 6.19’a bakılabilir.

6.6.4.2 On/Off Kontrolörün Simülasyon Ortamında Analizi

Bu analiz için kullanıcının öncelikle programın “Genel On-Off Parametrelerinin Girilmesi” bölümünden gerekli değerleri girmesi gerekir. Bu değerler on/off denetleyicinin on modunda iken çıkışını ne olacağını gösteren “Kontrolör On Çıkış Değeri” ile aktif olması için girişteki hata bilgisinin olması gereken değerini gösteren “Pozitif Delta e” değerleri ile on/off denetleyici off modunda iken çıkışını ne olacağını gösteren “Kontrolör Off Çıkış Değeri” ile denetleyicinin kapalı konumunda olması için girişteki hata bilgisinin olması gereken değerini gösteren “Negatif Delta e” değerleridir. Ayrıca, kullanıcı kontrol edilecek sistem için ileri ve geri blok pay ve payda değerlerini de programın “Sistem İleri ve Geri Blok Değerleri” bölümünden girmesi gerekmektedir. Bu değerler girildikten sonra kullanıcı simülayonun yapılacağı toplam süreyi belirlemeli, örneklem zamanını seçmeli ve referans değerini girmelidir. Program on/off denetleyici için iki farklı algoritma kullanmaktadır. İkisi arasında çok fazla bir farklılık olmamakla birlikte “On/Off Fonksiyonu” seçeneği “Simulink Model” seçeneğinden daha hassas işlem yapabilmektedir. En son olarak kullanıcı denetleyicinin çıkışı görmek için “On-Off Kontrolör Çıkış Grafiği” butonunu ya da kullanıcı sistemin çıkışını görmek için “C(s) Çıkış Grafiği” butonunu kullanabilir. Üretilen grafik programın sağ alt tarafında yer alan çizim alanında gösterilir. Bu durum Şekil 6.20’de gösterilmiştir. Ayrıca, kullanıcı isterse bu grafiği büyütebilir. Bun için “Grafiği Büyütmek İçin Tıklayınız…” butonu kullanılmalıdır. Bu durumda kullanıcı ayrıca açılan bir pencerede grafik çizimi görebilir ve istediği gibi boyutlandırabilir ya da pencerede yer alan araç çubuklarını kullanmak suretiyle özelliklerini değiştirebilir. Bu pencereyi görmek için Şekil 6.21’e bakılabilir.



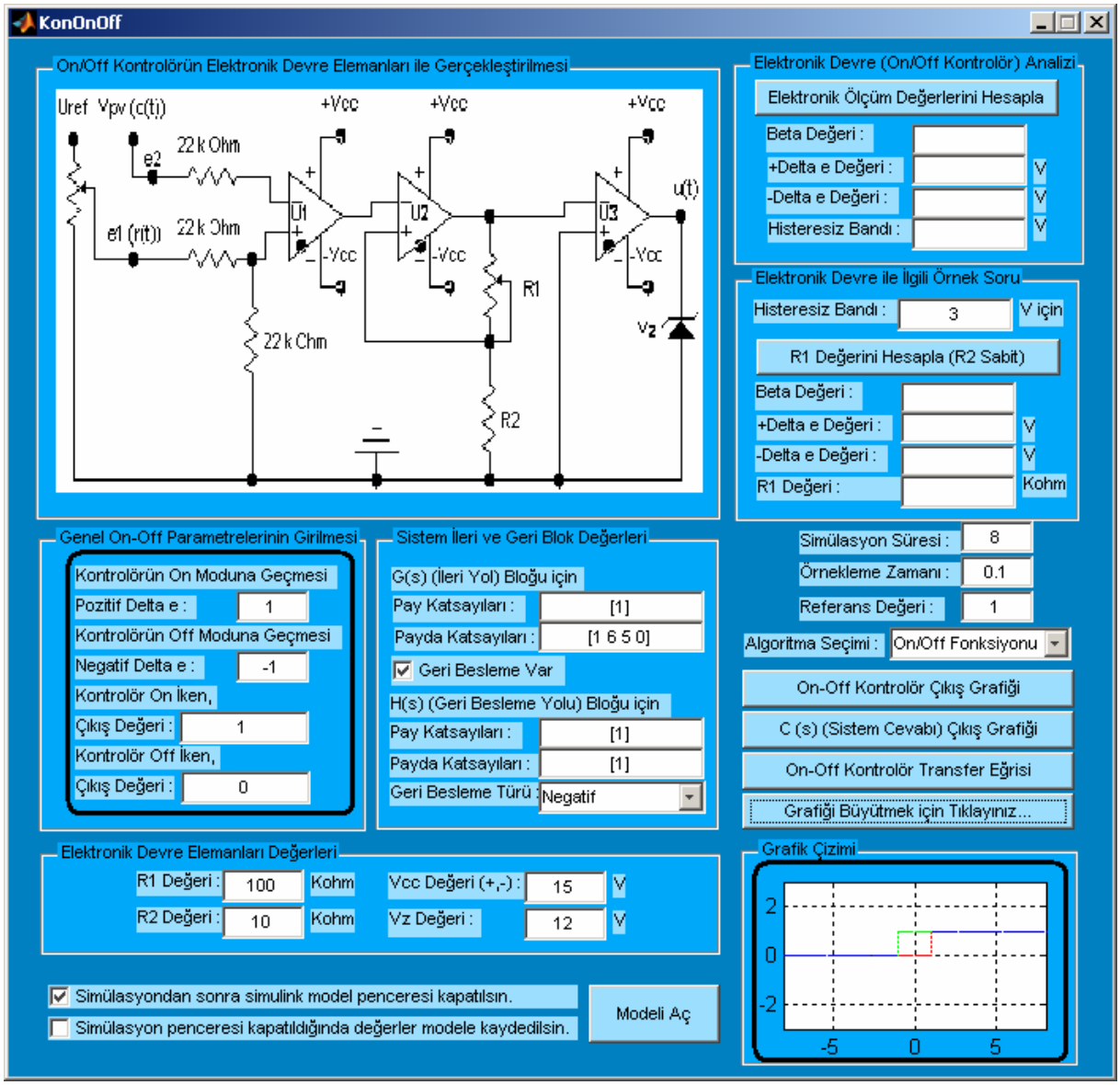
Şekil 6.20 Uygulama 6 Kullanım Ekranı 2



Şekil 6.21 Uygulama 6 Kullanım Ekranı 3

6.6.4.3 On/Off Kontrolörün Transfer Eğrisinin Çizilmesi

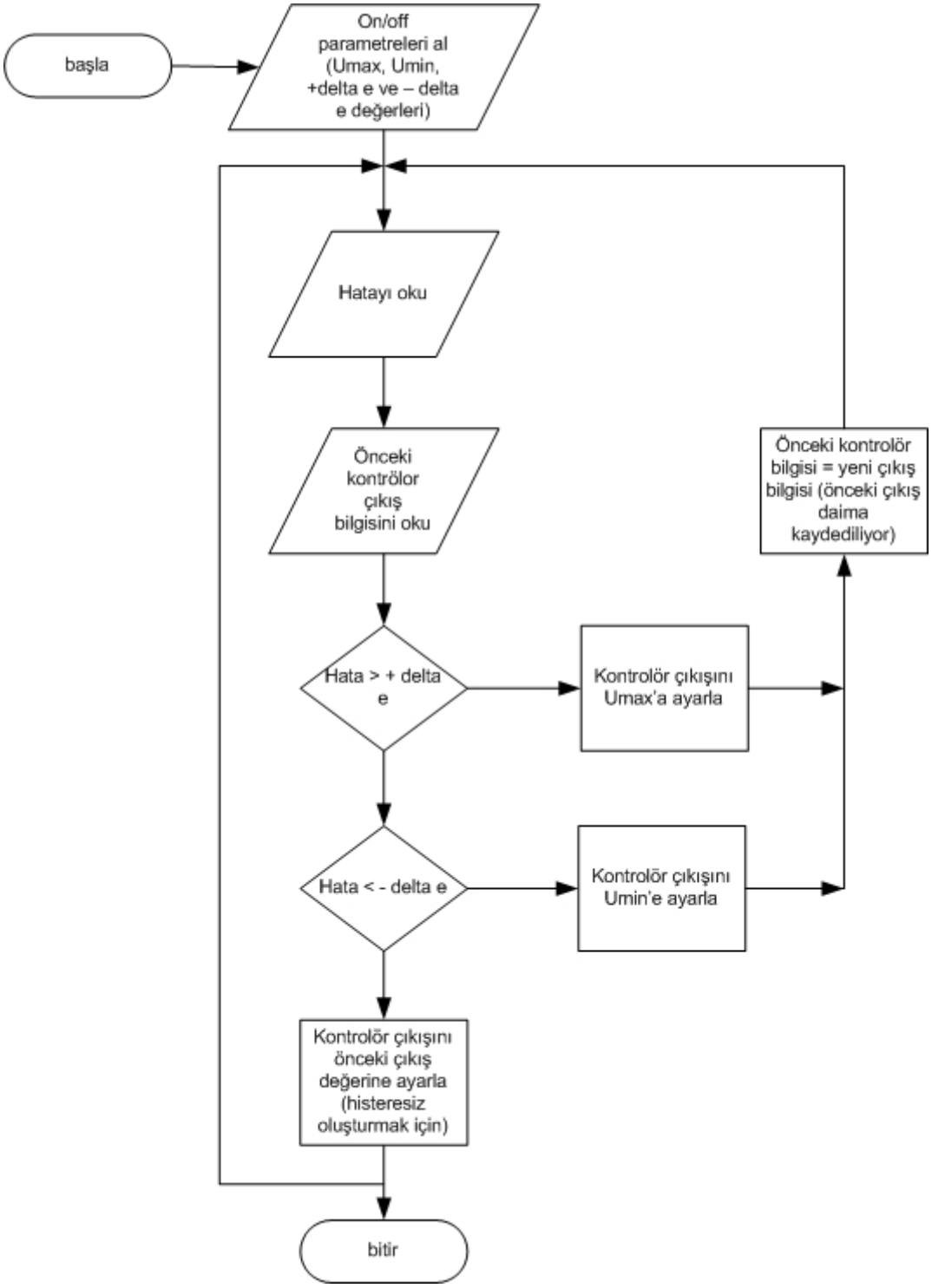
Girilen On/Off kontrolör parametrelerine göre on/off transfer eğrisi program tarafından çizilebilir. Bunun için programın "On/Off Kontrolör Transfer Eğrisi" butonu kullanılmalıdır. Örnek olarak çizilen bir transfer eğrisi Şekil 6.22'de gösterilmiştir.



Şekil 6.22 Uygulama 6 Kullanım Ekranı 4

6.6.5 Uygulamanın Algoritması

Bu uygulamada kullanılan on/off kontrolör algoritması, Algoritma 6.8'de gösterilmiştir.



Algoritma 6.8 Uygulama 6 Algoritması

6.7 Uygulama-7'nin Tanıtılması

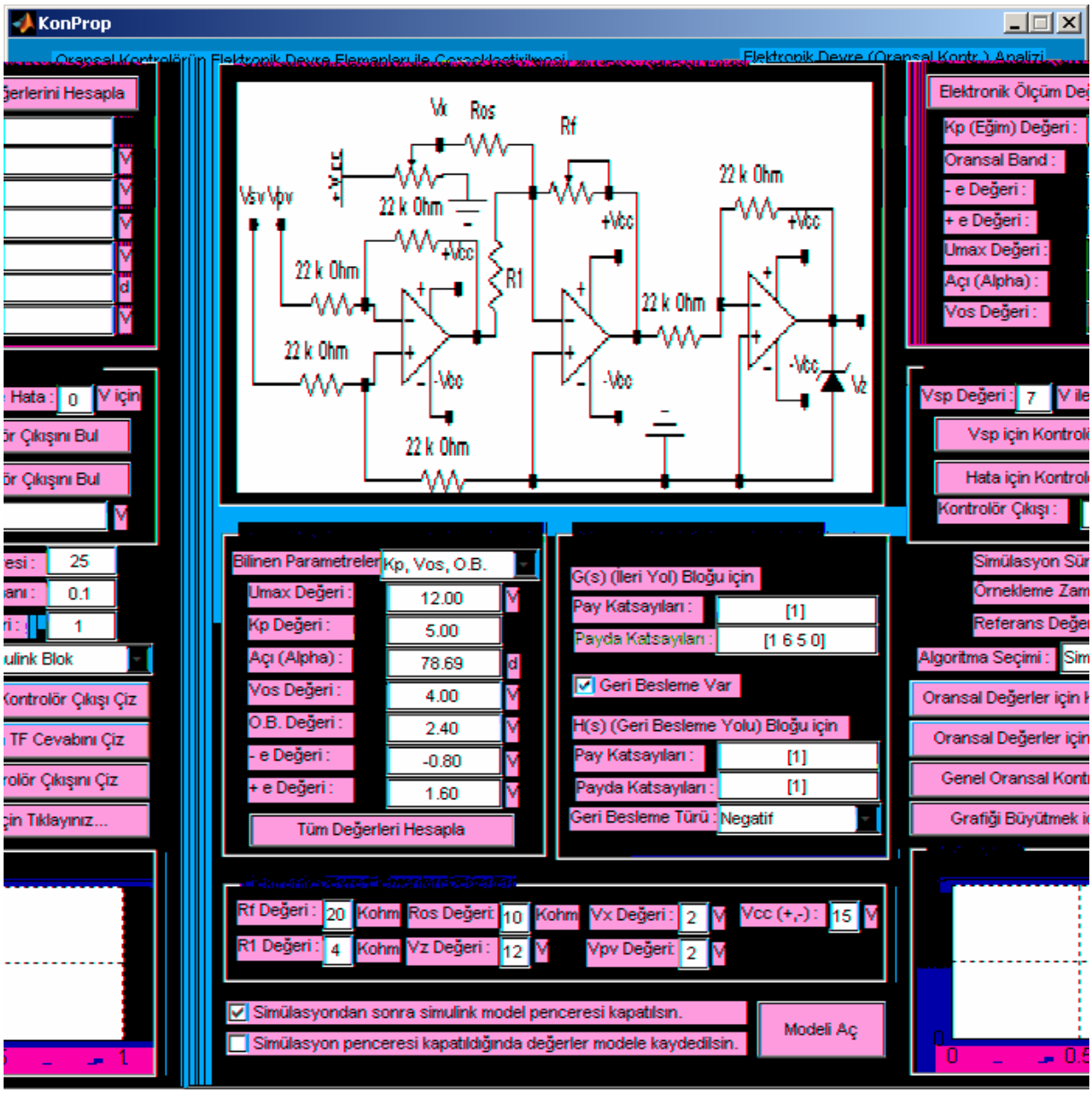
6.7.1 Uygulamanın Adı : Oransal Kontrolör Uygulaması

6.7.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerinde sıklıkla kullanılan oransal denetleyici ile ilgili hesaplamaların gerek elektronik devrelerle tasarım, grekse simülasyon ortamında oransal denetleyicinin çalışması sonucu bir sistemin cevabının nasıl olacağı konusunda kullanıcıları bilgilendirme amacını taşımaktadır. Ayrıca, 6.7 bölümünün sonunda kullanıcılar oransal kontrolör için gerekli algoritma konusunda da bilgilendirilmektedir.

6.7.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.18'de görülmektedir.



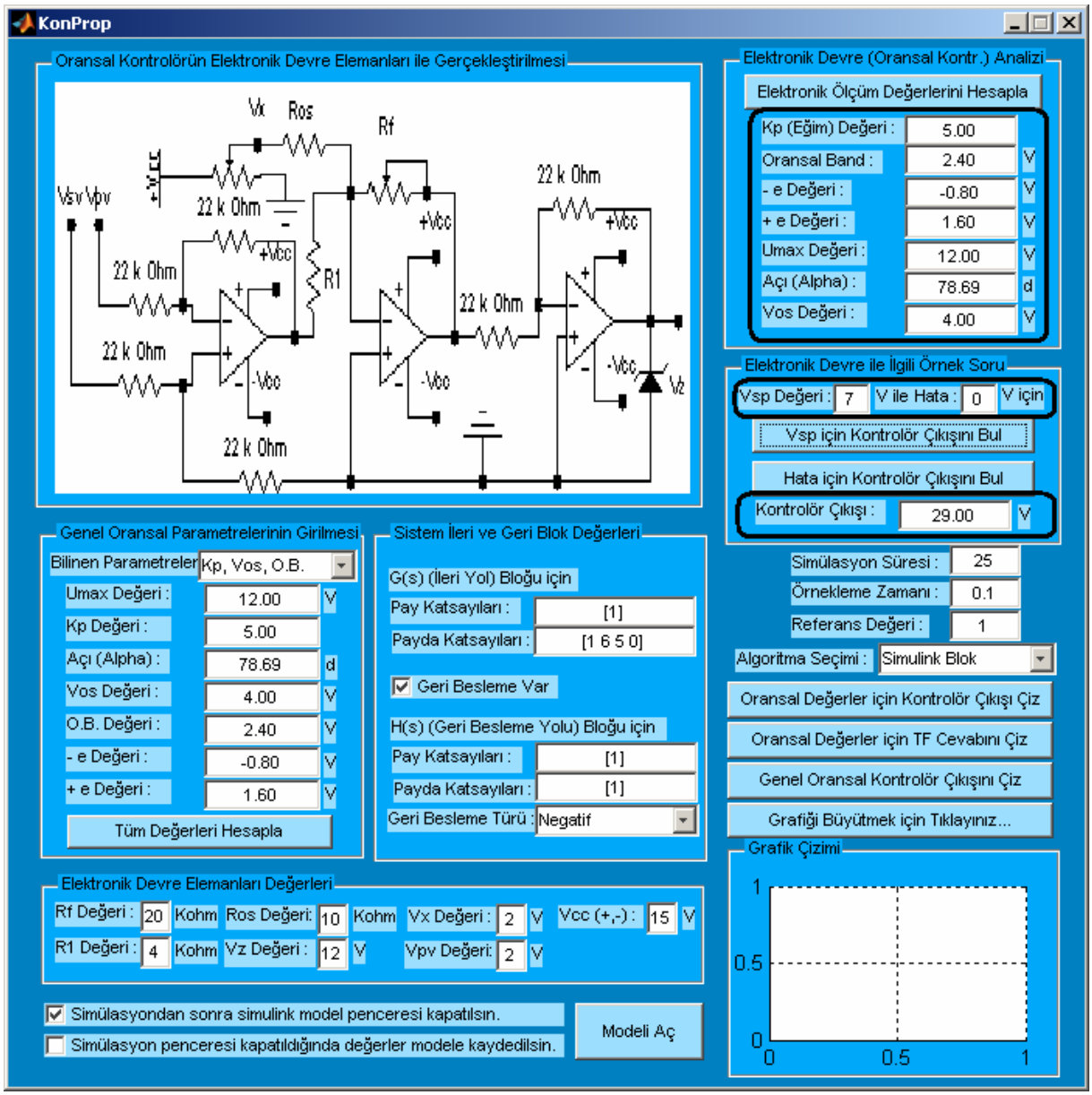
Şekil 6.23 Uygulama 7 Arayüzü

6.7.4 Uygulamanın Kullanılışı

Program arka planda bir simulink modelini kullanarak işlem yapmakta ve simulink modelinin çalışması sonucu üretilen değerleri kullanmaktadır. Bir kullanıcı isterse simulink modelini "Modeli Aç" butonunu kullanarak açabilir. Ayrıca, "Simülasyondan sonra simulink model penceresini kapatılsın." Seçeneği işaretlenmezse kontrolörün simülasyon çalışmasından sonra simulink penceresi kapatılmayacak ve açık kalmaya devam edecektir. Bunun yanında "Simülasyon penceresi kapatıldığında değerler modele kaydedilsin." Seçeneği eğer model kapatılsın seçneği (bir üstteki seçenek) işaretlenirse, oluşturulan sistem değerleri ve girilen parametreler simulink penceresi kapatılmadan önce model dosyasına kaydedilir ve daha sonra simulink penceresi kapatılır. Bu uygulama temel olarak üç farklı işlevi yerine getirmektedir. Bunlardan biri elektronik devre olarak tasarlanan bir oransal denetleyici ile ilgili analiz, diğer analiz olarak bir oransal denetleyicinin simülasyon ortamında çalıştırılarak sistem cevabının görülmesi ve en son olarak da bir oransal denetleyiciye ait oransal transfer eğrisinin kullanıcıya sunulmasıdır. Bu işlevlerin program ile nasıl gerçekleştirildiği konusunda ayrıntılı bilgiler aşağıda verilmiştir.

6.7.4.1 Oransal Elektronik Devre ile Analiz

Oransal denetleyici elektronik devresi ile ilgili değerlerin hesaplanabilmesi için öncelikle elektronik devresine ait değerlerin programın "Elektronik Devre Elemanları Değerleri" bölümünden programa girilmesi gereklidir. Bu değerler girildikten sonra programın "Elektronik Devre Analizi" bölümünden "Elektronik Ölçüm Değerlerini Hesapla" butonuna basıldığında gerekli değerler hesaplanacaktır. Bu değerlerin hesaplanmış hali Şekil 6.24'te gösterilmiştir. Ayrıca, oransal elektronik devresinde Vsp ve hata değerlerine göre oransal kontrolör çıkışı bulunabilir. Bu durum için de Şekil 6.24'e bakılabilir.

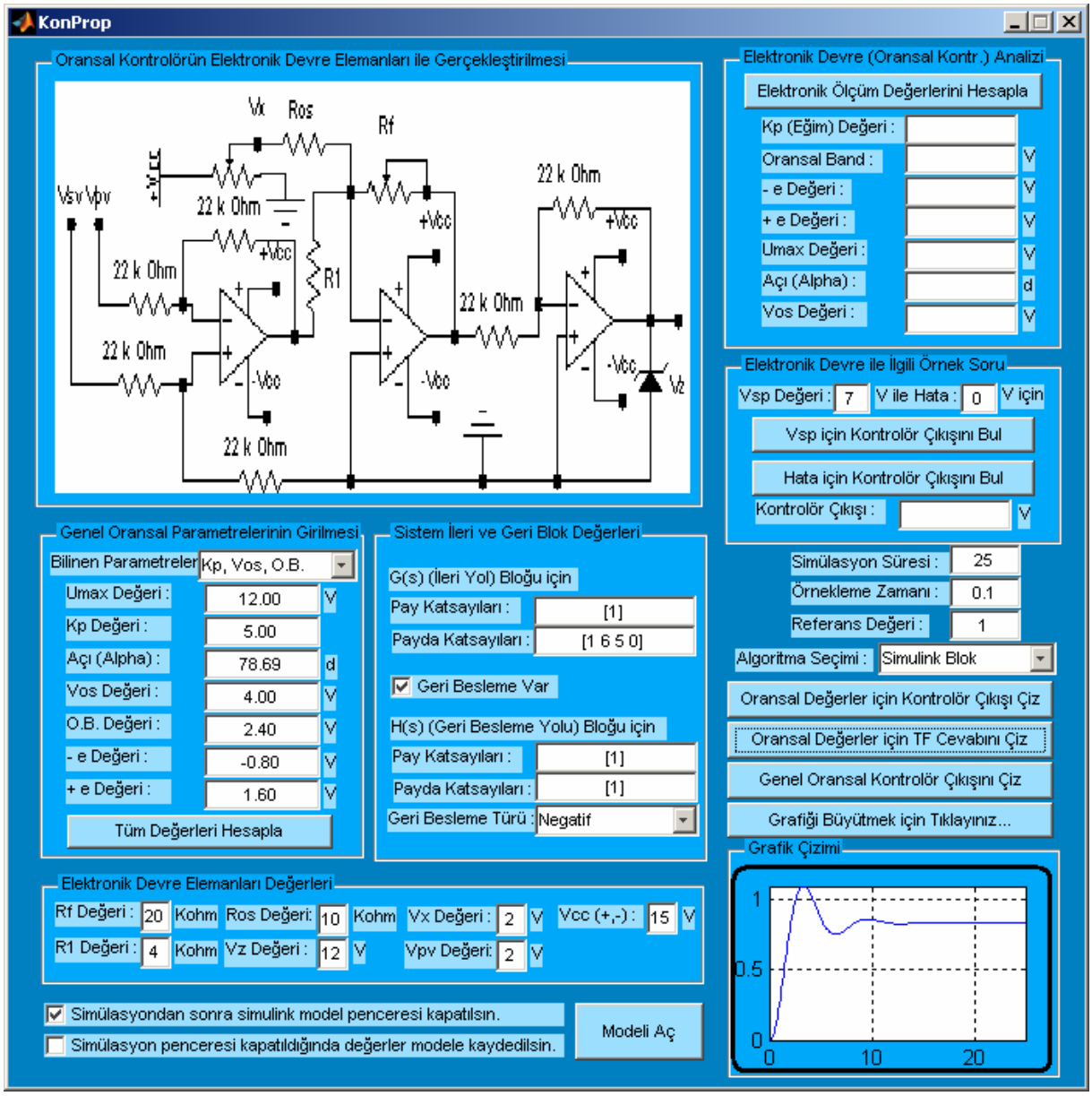


Şekil 6.24 Uygulama 7 Kullanım Ekranı 1

6.7.4.2 Oransal Kontrolörün Simülasyon Ortamında Analizi

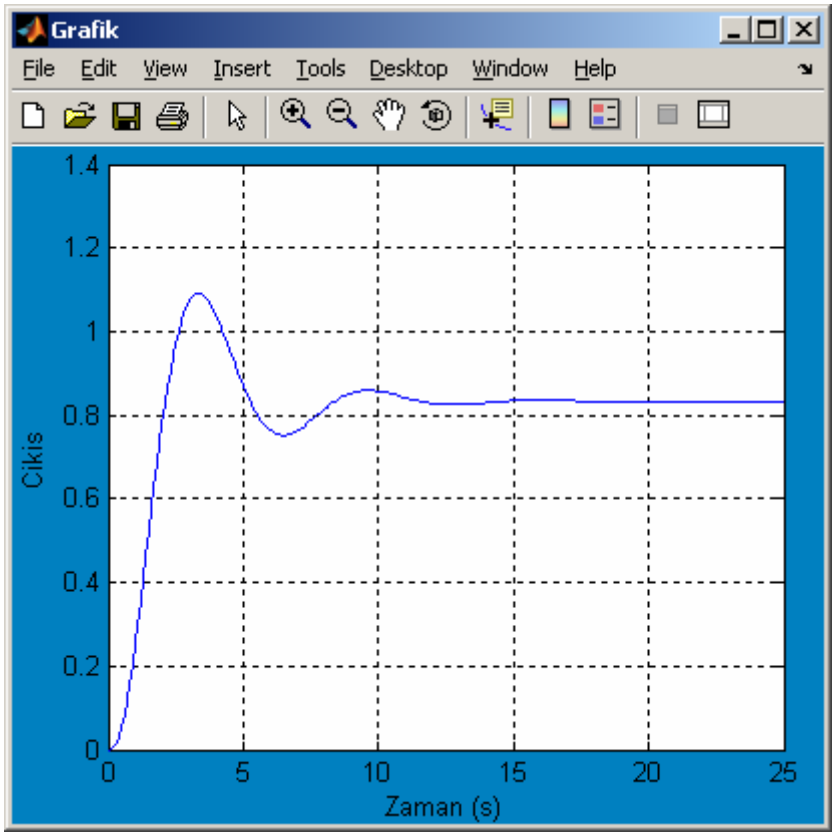
Bu analiz için kullanıcının öncelikle programın "Genel Oransal Parametrelerinin Girilmesi" bölümünden gerekli değerleri girmesi gerekir. Bu değerler oransal denetleyicinin hangi davranışta ve nasıl cevap vereceği ile ilgili Umax, pozitif ve negatif delta e, oransal band, kazanç (Kp), kontrolörün transfer eğrisi eğimi gibi parametrelerdir. Ayrıca, kullanıcı kontrol edilecek sistem için ileri ve geri blok pay ve payda değerlerini de programın "Sistem İleri ve Geri Blok Değerleri" bölümünden girmesi gerekmektedir. Bu değerler girildikten sonra kullnıcı simülayonun yapılacağı toplam süreyi belirlemeli, örneklem zamanını seçmeli ve referans değerini girmelidir. Program oransal denetleyici için iki farklı algoritma kullanmaktadır. Genellikle "Simulink Model" seçeneği "Oransal Fonksiyonu" seçeneğinden daha iyi sonuçlar vermektedir. Ancak, Umax değeri sadece "Oransal Fonksiyonu"

seçeneğinde dikkate alınmaktadır . En son olarak kullanıcı denetleyicinin çıkışı görmek için "Oransal Değerler için Kontrolör Çıkışını Çiz" butonunu ya da kullanıcı sistemin çıkışını



Şekil 6.25 Uygulama 7 Kullanım Ekranı 2

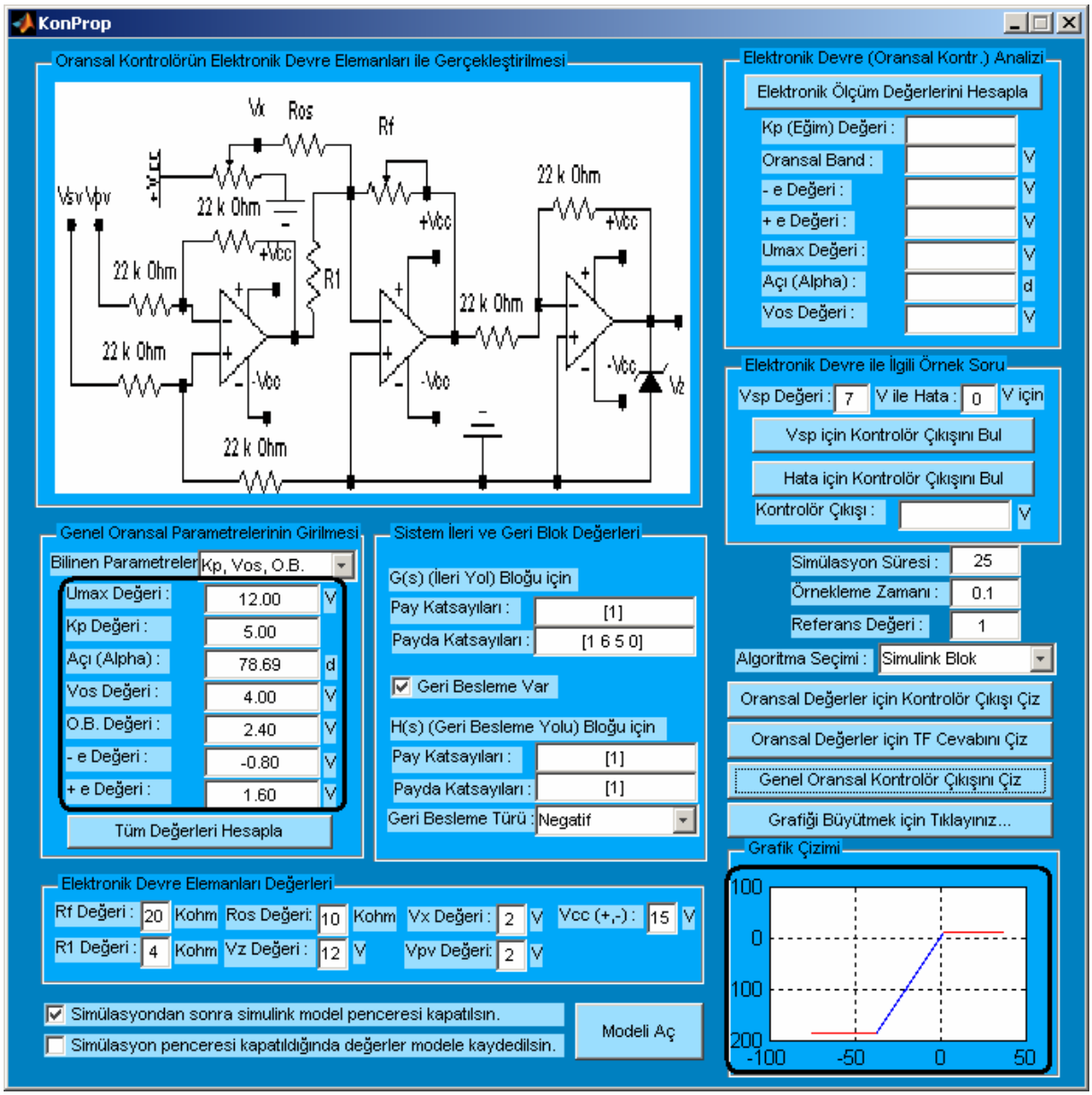
görmek için "Oransal Değerler için Sistem Cevabını Çiz" butonunu kullanabilir. Üretilen grafik programın sağ alt tarafında yer alan çizim alanında gösterilir. Bu durum Şekil 6.25'te gösterilmiştir. Ayrıca, kullanıcı isterse bu grafiği büyütebilir. Bun için "Grafiği Büyütmek İçin Tıklayınız…" butonu kullanılmalıdır. Bu durumda kullanıcı ayrıca açılan bir pencerede grafik çizimi görebilir ve istediği gibi boyutlandıranilir ya da pencerede yer alan araç çubuklarını kullanmak suretiyle özelliklerini değiştirebilir. Bu pencereyi görmek için Şekil 6.26'ya bakılabilir.



Şekil 6.26 Uygulama 7 Kullanım Ekranı 3

6.7.4.3 Oransal Kontrolörün Transfer Eğrisinin Çizilmesi

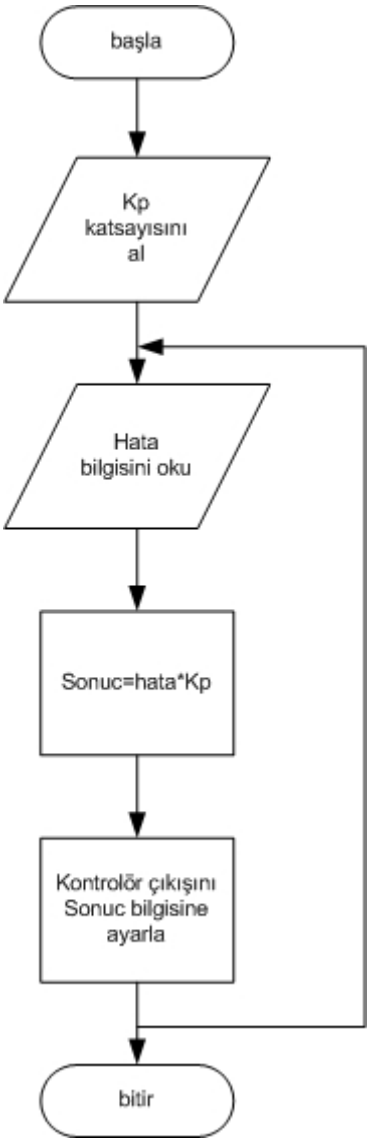
Girilen oransal kontrolör parametrelerine göre oransal transfer eğrisi program tarafından çizilebilir. Bunun için programın "Oransal Kontrolör Çıkışını Çiz" butonu kullanılmalıdır. Örnek olarak çizilen bir transfer eğrisi Şekil 6.27'de gösterilmiştir.



Şekil 6.27 Uygulama 7 Kullanım Ekranı 4

6.7.5 Uygulamanın Algoritması

Bu uygulamada kullanılan oransal kontrolör algoritması, Algoritma 6.9'da gösterilmiştir.



Algoritma 6.9 Uygulama 7 Algoritması

6.8 Uygulama-8'nin Tanıtılması

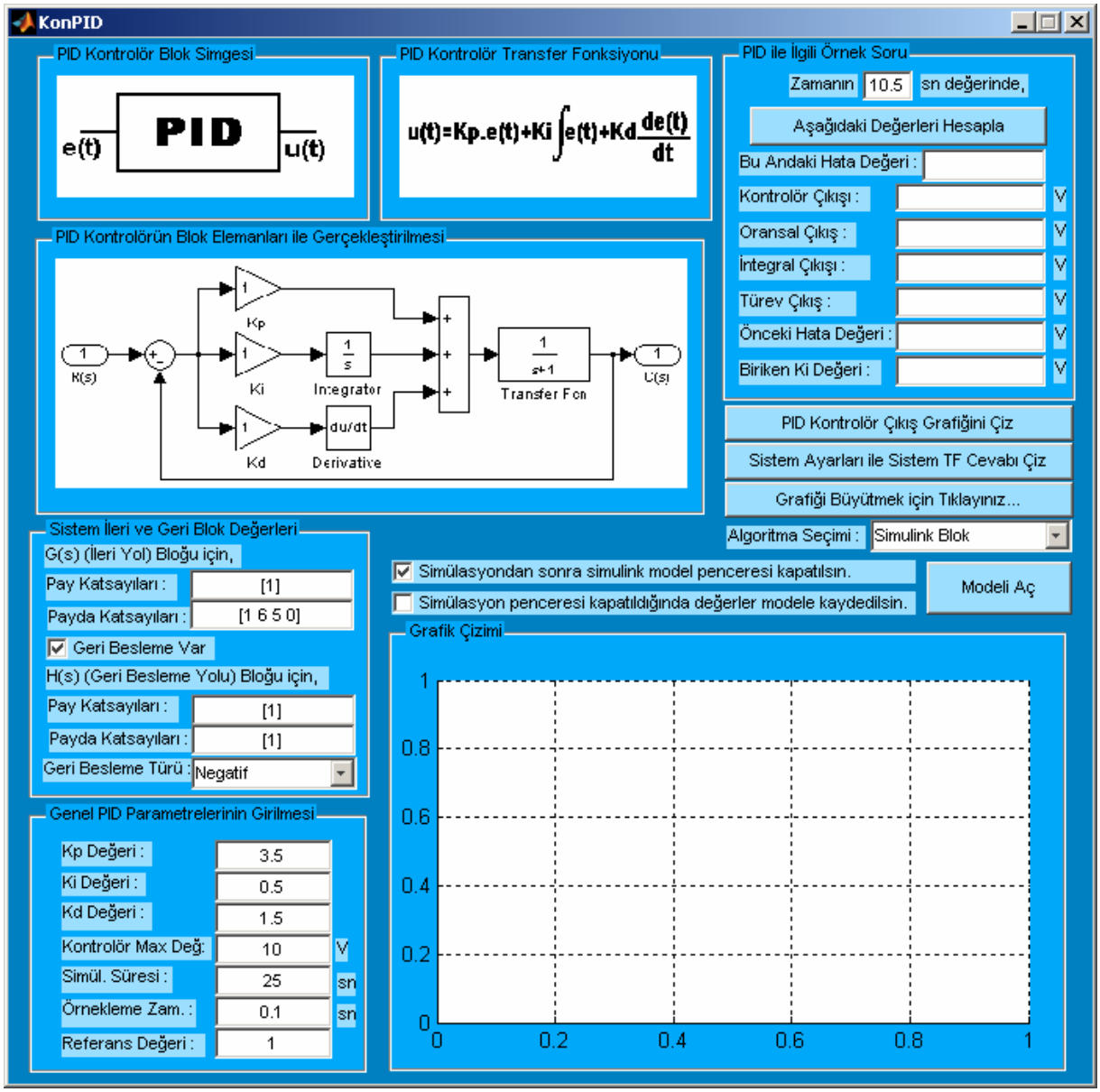
6.8.1 Uygulamanın Adı : PID Kontrolör Uygulaması

6.8.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerinde sıklıkla kullanılan PID denetleyici ile ilgili hesaplamaların gerek elektronik devrelerle tasarım, grekse simülasyon ortamında PID denetleyicinin çalışması sonucu bir sistemin cevabının nasıl olacağı konusunda kullanıcıları bilgilendirme amacını taşımaktadır. Ayrıca, 6.8 bölümünün sonunda kullanıcılar PID kontrolör için gerekli algoritma konusunda da bilgilendirilmektedir.

6.8.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.28'de görülmektedir.



Şekil 6.28 Uygulama 8 Arayüzü

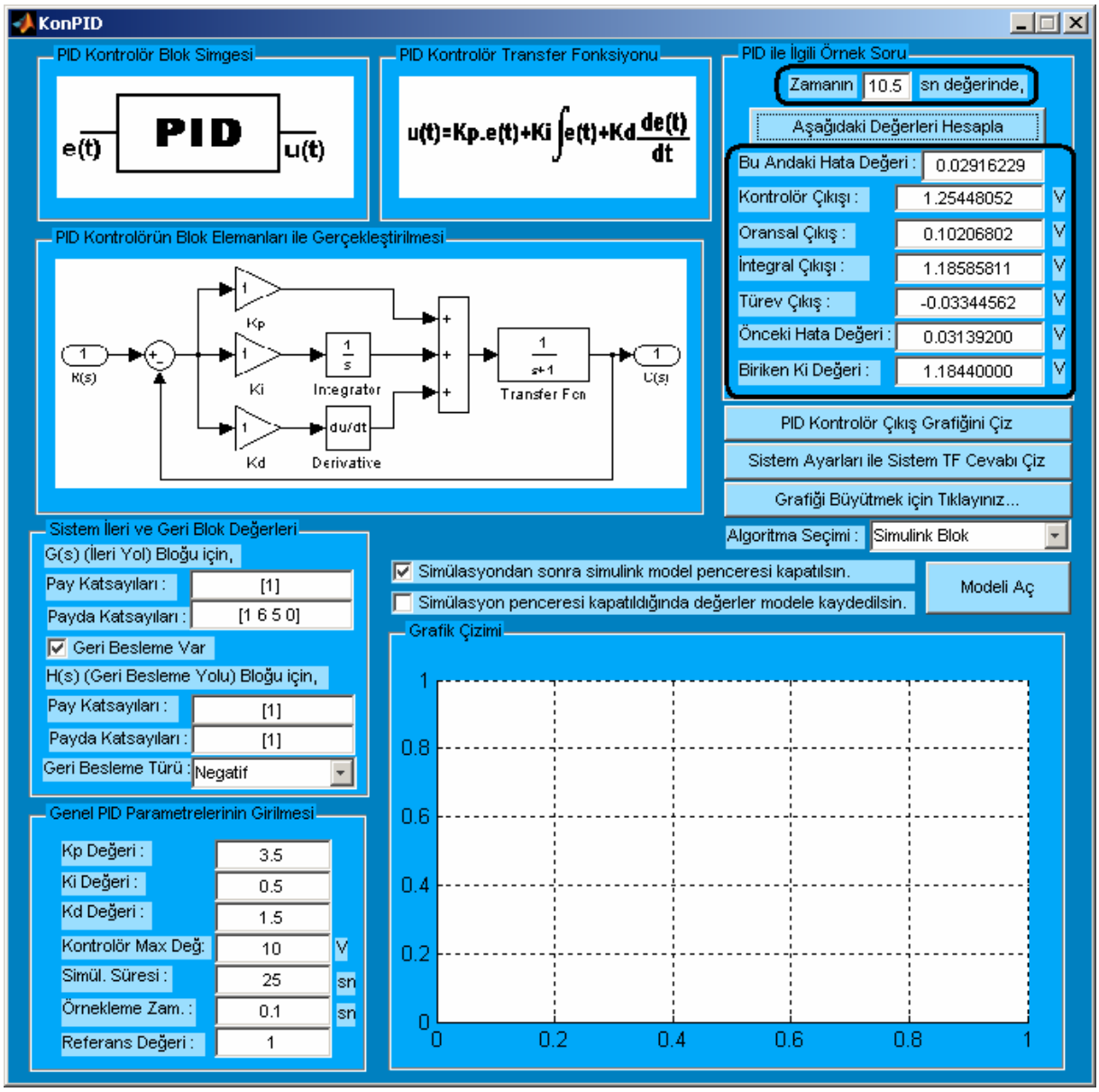
6.8.4 Uygulamanın Kullanılışı

Program arka planda bir simulink modelini kullanarak işlem yapmakta ve simulink modelinin çalışması sonucu üretilen değerleri kullanmaktadır. Bir kullanıcı isterse simulink modelini "Modeli Aç" butonunu kullanarak açabilir. Ayrıca, "Simülasyondan sonra simulink model penceresini kapatılsın." Seçeneği işaretlenmezse kontrolörün simülasyon çalışmasından sonra simulink penceresi kapatılmayacak ve açık kalmaya devam edecektir. Bunun yanında "Simülasyon penceresi kapatıldığında değerler modele kaydedilsin." Seçeneği eğer model kapatılsın seçneği (bir üstteki seçenek) işaretlenirse, oluşturulan sistem değerleri

ve girilen parametreler simulink penceresi kapatılmadan önce model dosyasına kaydedilir ve daha sonra simulink penceresi kapatılır. Bu uygulama temel olarak iki farklı işlevi yerine getirmektedir. Bunlardan biri PID kontrolörlü bir sistemde istenilen bir zaman değerinde PID kontrolörün çıkışının oransal, integral ve türev katları ve tüm değerleri ile integral geçmişi ve türev katı için önemli olan önceki hata değeri gibi paramtrelerin bulunması ve diğeri bir PID denetleyicinin simülasyon ortamında çalıştırılarak sistem cevabının ve kontrolör çıkışını kullanıcıya sunulmasıdır. Bu işlevlerin program ile nasıl gerçekleştirildiği konusunda ayrıntılı bilgiler aşağıda verilmiştir.

6.8.4.1 PID Kontrolör İçin Anlık Değerleri Hesapla

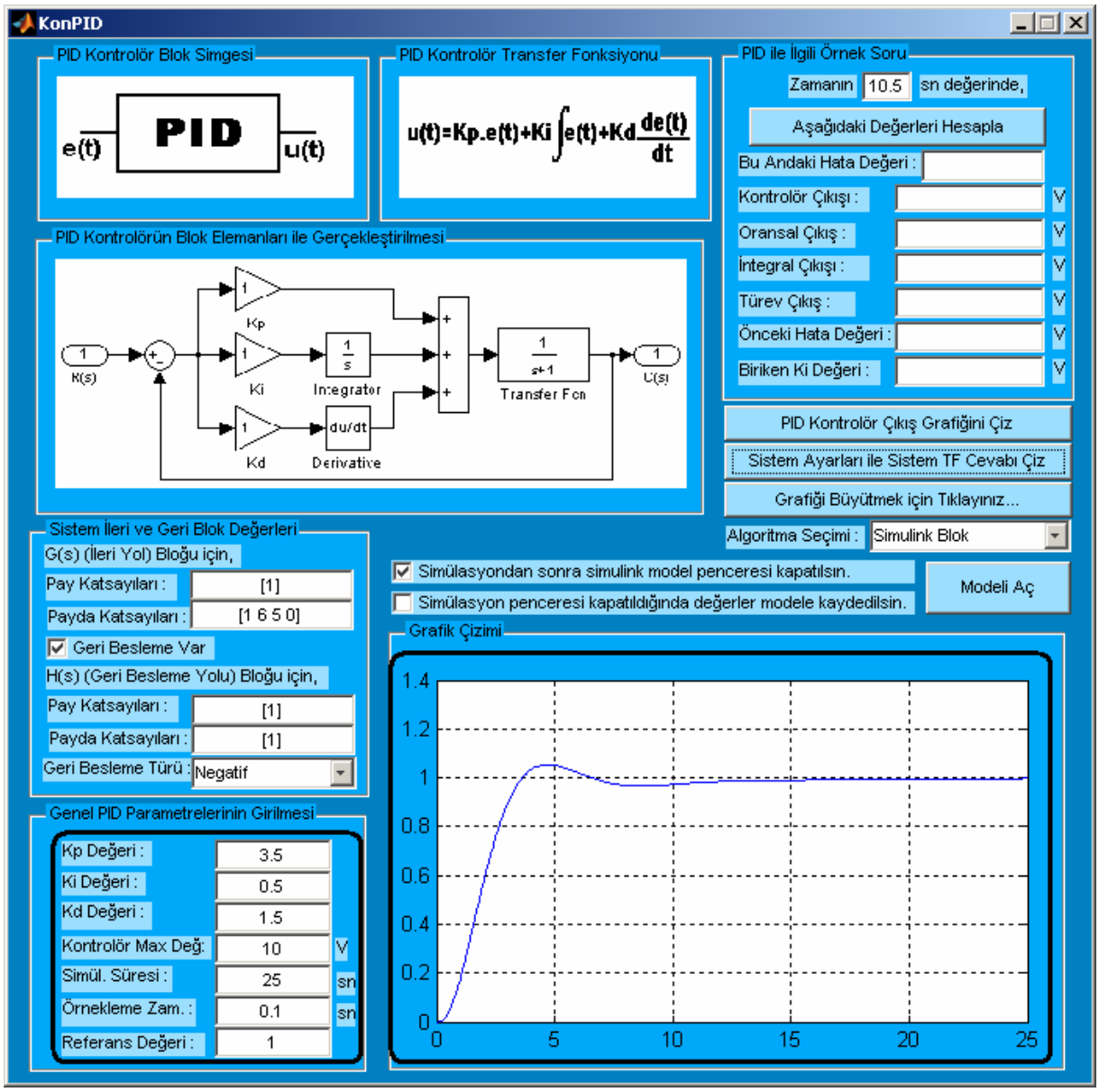
Programda herhangi bir sistem verilen PID katsayılarına göre kontrol edilmek istendiğinde herhangi bir zaman değeri için ileri yol bloğuna ve sisteme etki eden PID kontrolörün o andaki hata değeri, örneklemeye bağlı olarak önceki hata değeri (ki bu değer PID deetleyicinin türev katı tarafından kontrol edilmektedir), integral katının geçmişteki değerler toplamı ile ayrı ayrı oransal, integral ve türev katı ile toplam çıkış değerleri öğrenilebilir. Bu işlem için programın "PID ile İlgili Örnek Soru" bölümünde yer alan "Aşağıdaki Değerleri Hesapla" butonu kullanılmalıdır. Örnek bir sistem için 10.5 saniye zaman değerinde PID kontrolör ile ilgili değerler hesaplanması Şekil 6.29'da görülmektedir.



Şekil 6.29 Uygulama 8 Kullanım Ekranı 1

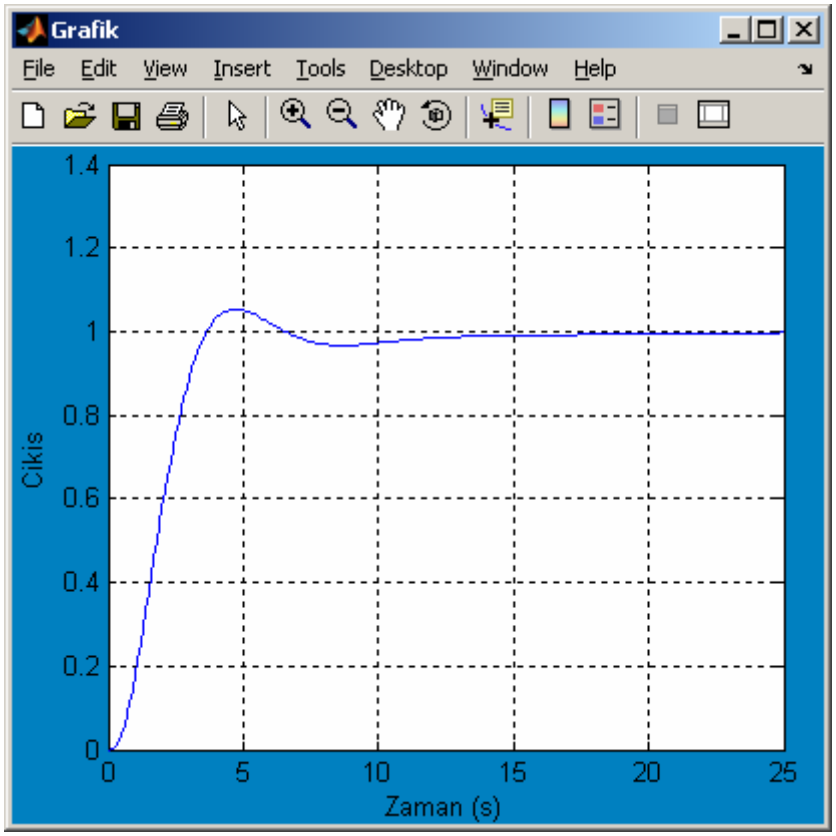
6.8.4.2 PID Kontrolörün Cevabının Çizdirilmesi

Bu analiz için kullanıcının öncelikle programın "Genel PID Parametrelerinin Girilmesi" bölümünden gerekli değerleri girmesi gerekir. Bu değerler PID denetleyicinin hangi davranışta ve nasıl cevap vereceği ile ilgili UmKp, Ki ve Kd gibi parametrelerdir. Benzer şekilde kullanıcı aynı yerden simülayonun yapılacağı toplam süreyi belirlemeli, örneklem zamanını seçmeli ve referans değerini girmelidir. Ayrıca, kullanıcı kontrol edilecek sistem için ileri ve geri blok pay ve payda değerlerini de programın "Sistem İleri ve Geri Blok Değerleri" bölümünden girmesi gerekmektedir. Program PID denetleyici için iki farklı algoritma kullanmaktadır. Genellikle "Simulink Model" seçeneği "PID Fonksiyonu" seçeneğinden daha iyi sonuçlar vermektedir. Ancak, Kontrolör için maksimum değer sadece "PID Fonksiyonu" seçeneğinde dikkate alınmaktadır . En son olarak kullanıcı denetleyicinin çıkışı görmek için "PID Kontrolör Çıkış Grafiği" butonunu ya da kullanıcı sistemin çıkışını



Şekil 6.30 Uygulama 8 Kullanım Ekranı 2

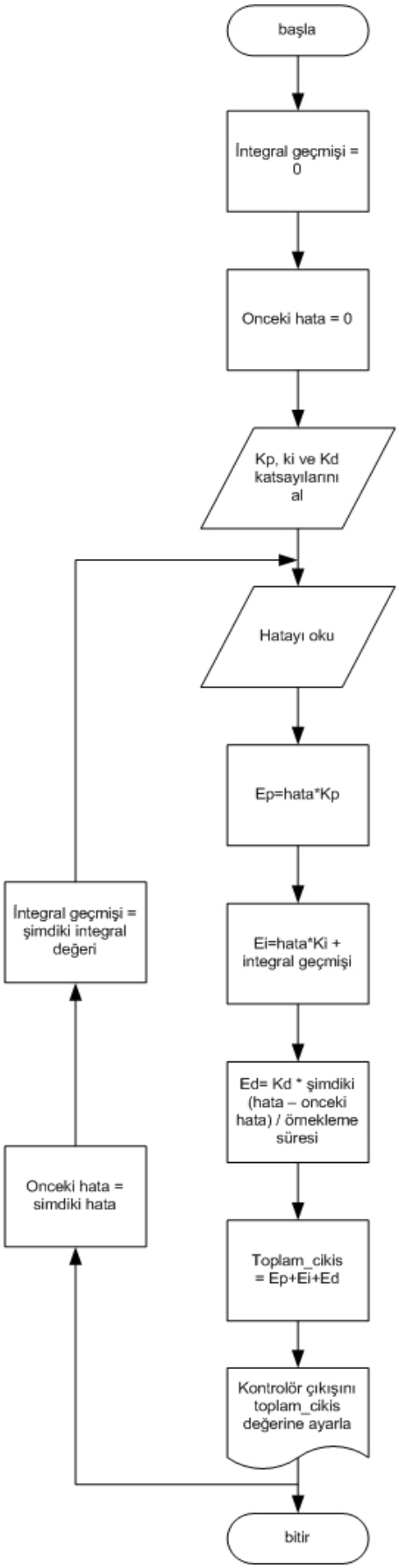
görmek için "Sistem Ayarları ile Sistem Transfer Fonksiyonu Cevabını Çiz" butonunu kullanabilir. Üretilen grafik programın sağ alt tarafında yer alan çizim alanında gösterilir. Bu durum Şekil 6.30'da gösterilmiştir. Ayrıca, kullanıcı isterse bu grafiği büyütebilir. Bun için "Grafiği Büyütmek İçin Tıklayınız…" butonu kullanılmalıdır. Bu durumda kullanıcı ayrıca açılan bir pencerede grafik çizimi görebilir ve istediği gibi boyutlandıranilir ya da pencerede yer alan araç çubuklarını kullanmak suretiyle özelliklerini değiştirebilir. Bu pencereyi görmek için Şekil 6.31'de bakılabilir.



Şekil 6.31 Uygulama 8 Kullanım Ekranı 3

6.8.5 Uygulamanın Algoritması

Bu uygulamada kullanılan PID kontrolör algoritması, Algoritma 6.10'da gösterilmiştir.



Algoritma 6.10 Uygulama 8 Algoritması

6.9 Uygulama-9'un Tanıtılması

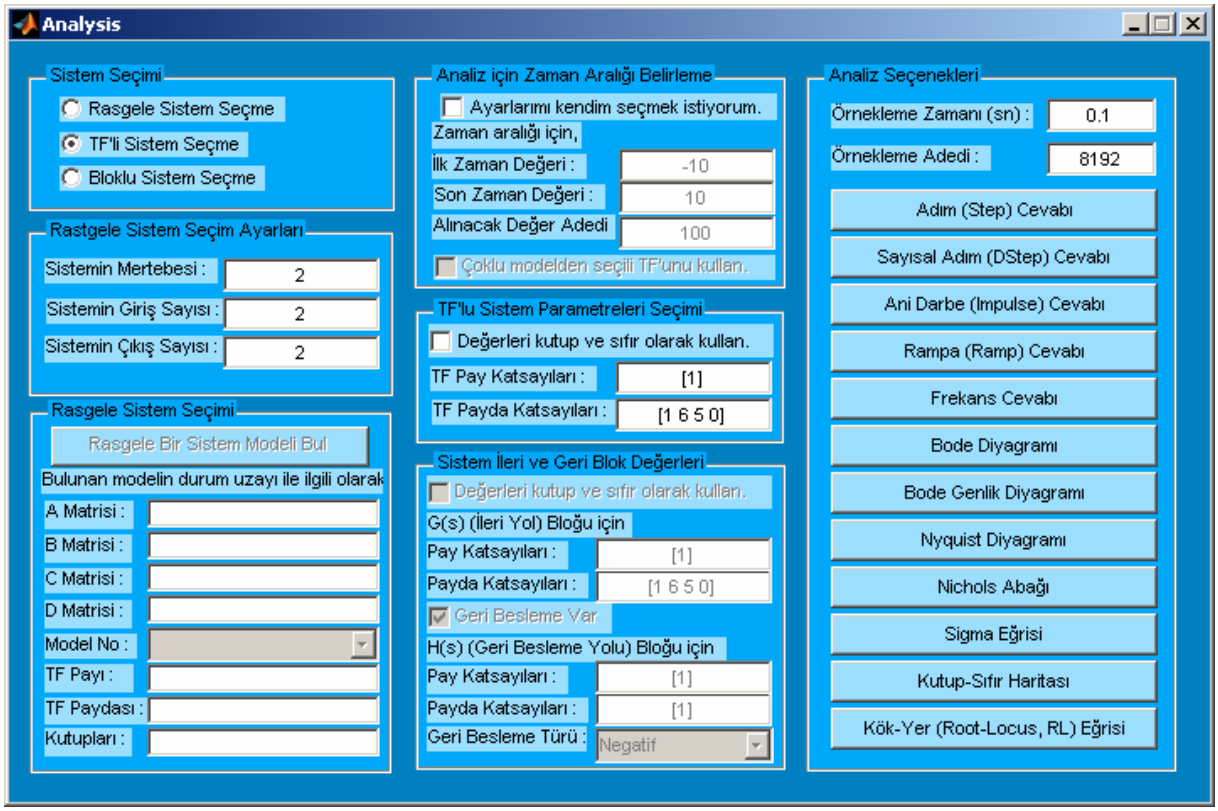
6.9.1 Uygulamanın Adı : Kontrol Sistemleri için Analiz Programı

6.9.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama kontrol sistemlerinin kararlılık, cevap ya da pek çok farklı yönlerden incelenmesi amacıyla kullnıcılara zengin analiz tercihleri sunularak özellikle pek çok faydalı aracı bir arada sunan bir araç olması ile kullanıcılara çalışmalarında kolaylık sağlaması ve zaman kazandırması durumları göz önünde bulundurularak tasarlanmıştır.

6.9.3 Uygulamanın Arayüzü

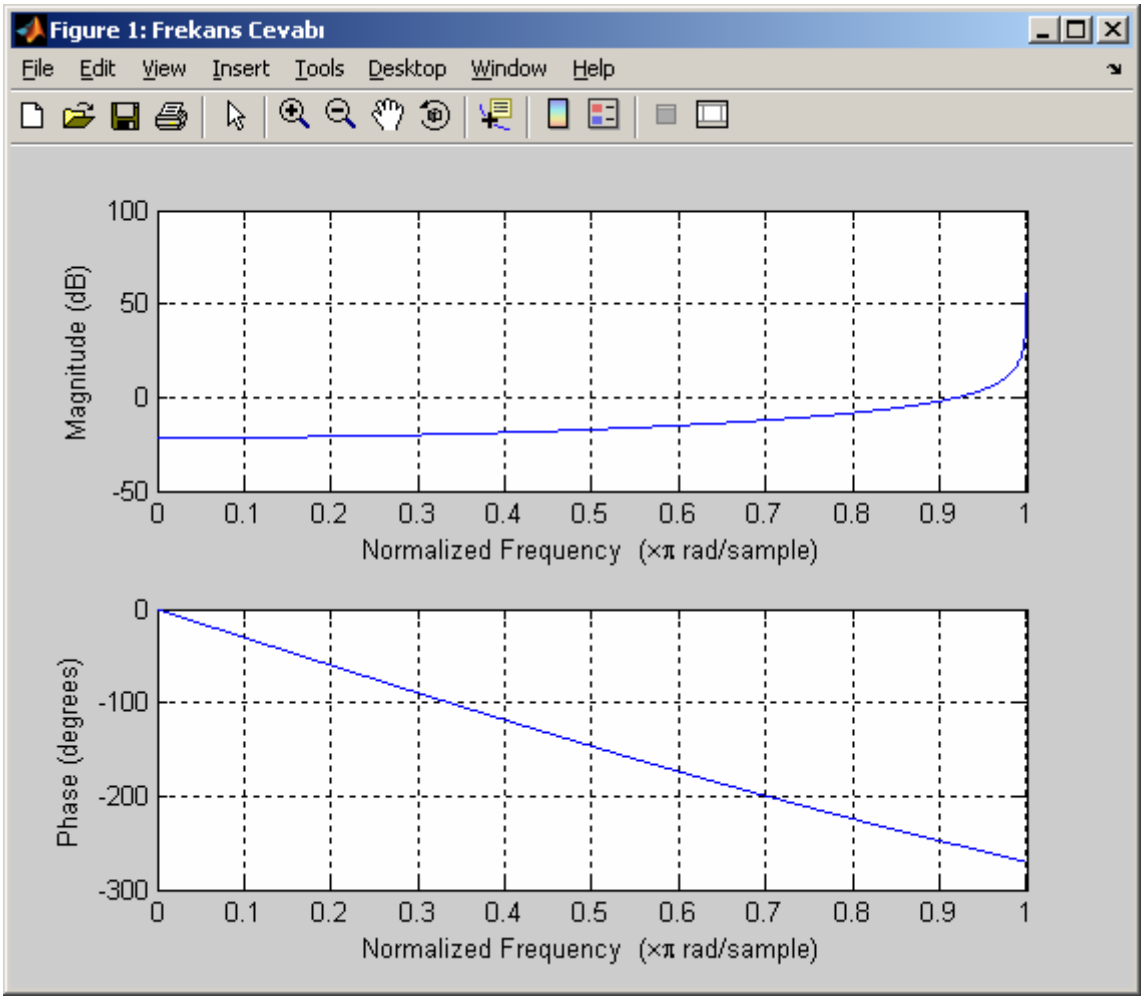
Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.32'de görülmektedir.



Şekil 6.32 Uygulama 9 Arayüzü

6.9.4 Uygulamanın Kullanılışı

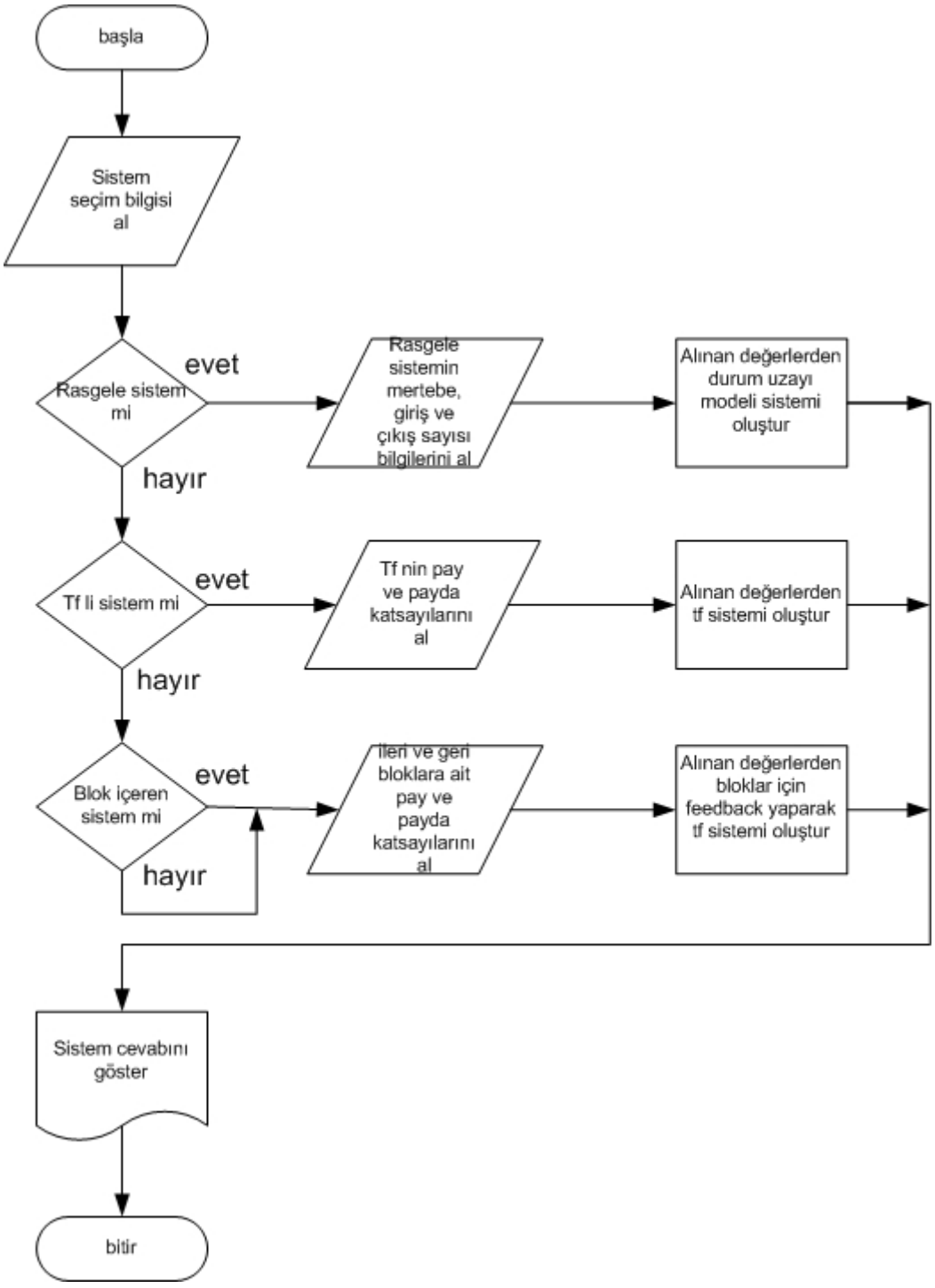
Uygulama genel anlamı ile girilen kontrol sistemi değerlerine göre çeşitli amaçlara hizmet eden sistem cevabı ve analiz amaçlı eğrilerin çizdirilmesini sağlamaktadır. Sistem girilmesi için programda üç farklı seçenek vardır. Bunlar: Rasgele sistem, TF'lu sistem ve Bloklu sistem. Her bir seçim için sırayla "Rastgele Sistem Seçim Ayarları" ile "Rastgele Sistem Seçimi" alanları, "TF'lu Sistem Parametreleri Seçimi" alanı ve "Sistem İleri ve Geri Blok Değerleri" alanı aktif olmaktadır. Herhangi bir anda bu alanlardan sadece biri aktif olmakta ve o aktif alan bilgileri kullanılarak ilgili analiz tercihi çalıştırılmaktadır. Kullanıcı rasgle bir sistem kullanmak isterse "Rasgele Bir Sistem Modeli Bul" butonunu kullanarak "Rastgele Sistem Seçim Ayarları" bölümünden belirlediği giriş ve çıkış sayısında rasgele bir durum uzayı modeli oluşturabilir. Bunların yanında eğer kullancıı birden fazla durum uzayından oluşan TF yerine tek bir seçili TF için cevap istiyor veya analiz yapmak gerekiyorsa bu takdirde programın "Analiz için Zaman Aralığı Belirleme" bölümünden "Çoklu modelden seçili TF'nu kullan." Seçeneğini işaretlemesi ve "rasgele Sistem Seçimi" bölümünden Model No listesi kullanılarak bir TF'nu seçilmelidir. Kullanıcı eğer direk bir sistemin tansfer fonksiyonunu kullanarak işlem yapmak isterse "TF'lu Sistem Parametreleri Seçimi" alanından sisteminin TF'nuna ait pay ve payda katsayılarını girmelidir. İstenirse bu katsayılar "Değerleri sıfır ve kutup olarak kullan." Seçeneği işaretlenerek direk pay kısmına sıfır değerleri ve payda kısmına kutup değerleri yazılarak da analize katılabilir. Benzer şekilde eğer kullancıı bloklar halinde sistemin cevabına bakmak ya da sistemi analiz etmek istiyorsa bu durumda programın "Sistem İleri ve Blok Değerleri" bölümüne kendi sistem değerlerini girmelidir. Cevaplar ve bazı analizler için örneklem adedi gibi değerlere gereksinim oluğu için bu değerlerin kullanıcı tarafından belirlenmesi gerekmektedir. Ayrıca, kullanıcı bir cevp grafiği ve analizi istediği bir aralık için yapmak istiyorsa bu takdirde "Analiz için Zaman Aralığı Belirleme" bölümüden dikkate alınacak zaman aralığının sınır değerleri ve kullanılacak değer adedi gibi parametreler programa girilmelidir. Son olarak kullanıcı program arayüzünün sağ tarafında yer alan butonlardan amacına uygun olanı kullanarak yeni açılan ikinci bir pencereden cevabı veya analiziyle ilgili çizime ulaşabilir. Örnek olarak varsayılan program değerleri kullanılarak ele alınan bir sistemin frekans cevabı analizi yapılmıştır. Bu duurm için Şekil 6.33'ye bakılabilir.



Şekil 6.33 Uygulama 9 Kullanım Ekranı

6.9.5 Uygulamanın Algoritması

Bu uygulamada çok çeşitli analiz yöntemleri kullanılmış olup, bu yöntemlerin nasıl çağrıldığı özellikle Matlab komutları dikkate alınarak aktarılmaya çalışılmıştır. Gerekli bilgilere Algoritma 6.11'den erişilebilir.



Algoritma 6.11 Uygulama 9 Algoritması

6.10 Uygulama-10'nun Tanıtılması

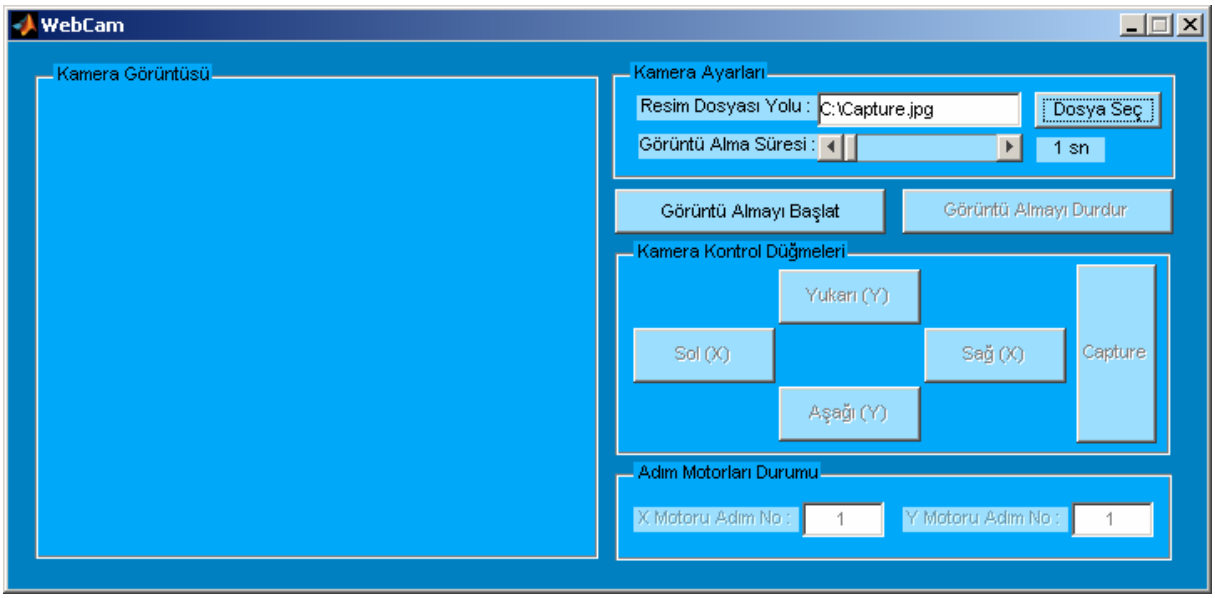
6.10.1 Uygulamanın Adı : X ve Y Eksenli WebCam Cihazının Paralel Port ile Kontrolü

6.10.2 Uygulamanın Hazırlanma Amacı :

Bu uygulama Matlab ile port tabanlı kontrol sağlamak ve bu işlemin nasıl gerçekleştiği konusunda kullanıcıları bilgilendirmek amacıyla tasarlanmıştır. Programaın algoritması ile ilgili bilgilere de bu bölümün sonunda yer verilmiştir.

6.10.3 Uygulamanın Arayüzü

Uygulamanın grafiksel arayüzü (GUI alanı) Şekil 6.34'te görülmektedir.



Şekil 6.34 Uygulama 10 Arayüzü

6.10.4 Uygulamanın Kullandığı Donanım Arabirimi

Bu proje çalışmasında pek çok donanım arabirimi bir arada kullanılmıştır. Aşağıda her bir arabirim ile ilgili detaylı bilgiler verilmektedir.

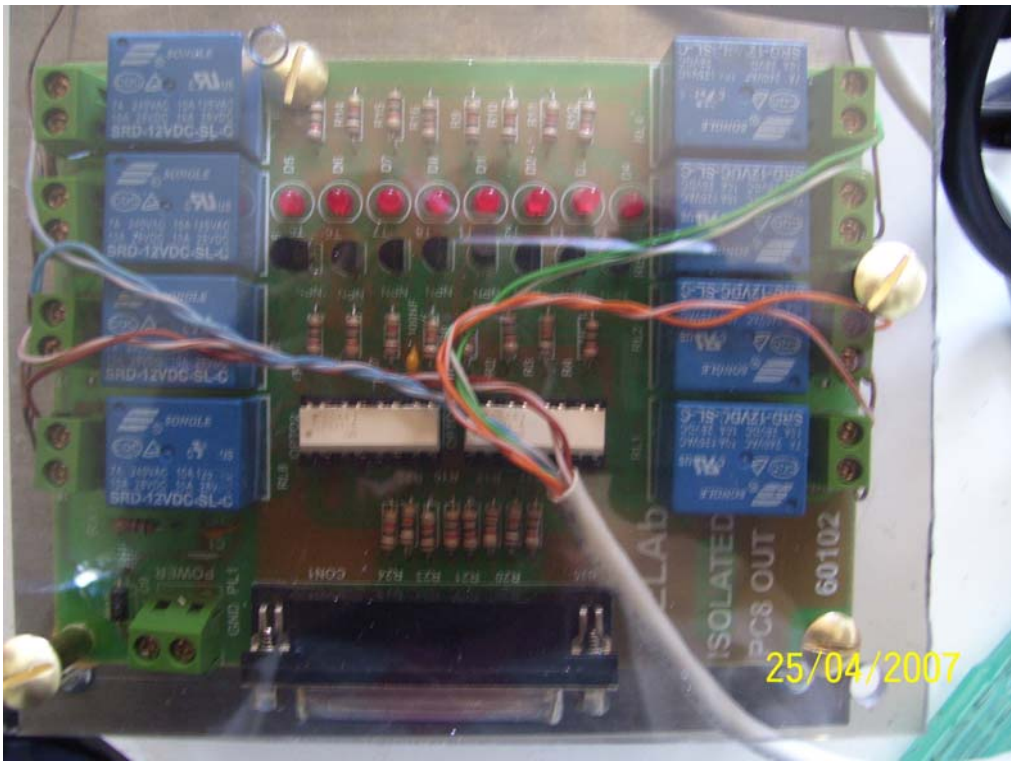
6.10.4.1 Besleme Devresi:

Projede sensörlere uygulanan gerilimlerin ve sürücü devresini besleyecek gerilimlerin gerekliliği için üç trafodan oluşan ve üç ayrı doğrultma devresi destekli elektronik devreler baskı devre tekniği kullanılarak ortaya çıkarılmıştır.

6.10.4.2 Sürücü Devresi:

Projede kullanılan bu devre hazır KIT özelliği taşımaktadır. Bu devre direk paralel port data pinleri ile beyzinden sürülen transistörlerin 8 adet röleyi anahtarlaması prensibine göre çalışmaktadır. Devremizin resmi aşağıda verilmiştir. Herhangi bir data pini HIGH seviyeye çekildiği durum aynı zamanda devremizin üzerinde yer alan LED'ler vasıtasıyla da görülebilir. Böylelikle herhangi bir anda hangi eksende hangi adımın işletildiği öğrenilebilir. Her bir motor için toplam dörder adım ve her adım da bir pin tarafından kontrol edilmektedir.

www.delab.com adresinden PC OUT 8 KIT devresi ile ilgili ayrıntılı bilgi elde edilebilir. Projede kullanılan sürücü devresi Resim 6.1'de gözükmektedir.



Resim 6.1 Step Motor Sürücü Devresi

6.10.4.3 Platform:

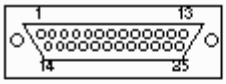
Projede yukarıda bahsedilen elemanların dağınık olmaması ve göze hoş gelmesi için bir arada metal bir plaka üzerinde montajı yapılmıştır. Ayrıca, dış darbelere karşı sürücü devre üzeri plastik şeffaf bir malzeme ile korunmuştur.

6.10.4.4 Paralel Port:

Paralel port, 25 pinlik D şeklindeki konnektördür. Seri porta göre hızlı olmasına rağmen aynı stabiliteyi sağlayamaz. Bu bağlantı noktasına aynı zamanda LPT (LinePrinTer) de denmektedir. Bu portun bir pini bir seferde 8 bit veri gönderebilir. DB25 isimli portu kullanır. DB25 ismindeki 25 rakamı kablo girişindeki pin sayısını ifade etmektedir. Bu arabirim Şekil 6.35'te gösterilmiştir.

Şekil 6.35 Paralel Portlarda Kullanılan DB-25 Konnektörü

Projede paralel port adresi olarak 378 hex adresi kullanılmıştır. (Delphi için bu değer $378 olarak geçer.) Paralel porttan x ve y ekseni adım motorlarını sürmek için bu porta ait data pin çıkış uçları kullanılmaktadır. Bu pinler ilk pin 1 nosu ile isimlendirilmek üzere 2 ile 9 nolu pinler dahil ve bu pinler arasında kalan pinlerdir. Bu pinlerden sürücü devresi vasıtasıyla ilgili eksene ait adım motorunun ilgili adımına ait röle enerjilendirilmekte, dolayısıyla da kontakları konum değiştiren röle vasıtasıyla o adıma ait adım motoru uçlarına gerilim uygulanmakta ve adım motoru bir adım atmaktadır.



View is looking at
Connector side of
DB-25 Male Connector.

Pin	Description	
1	Strobe	PC Output
2	Data 0	PC Output
3	Data 1	PC Output
4	Data 2	PC Output
5	Data 3	PC Output
6	Data 4	PC Output
7	Data 5	PC Output
8	Data 6	PC Output
9	Data 7	PC Output
10	ACK	PC Input
11	Busy	PC Input
12	Paper Empty	PC Input
13	Select	PC Input
14	Auto Feed	PC Output
15	Error	PC Input
16	Initialize Printer	PC Output
17	Select Input	PC Output

Pin Assignments

Note: 8 Data Outputs
4 Misc Other Outputs

5 Data Inputs

Note: Pins 18-25 are
Ground

Şekil 6.36 Paralel Port Pinleri

Şekil 6.36'da da görüldüğü üzere paralel port pinlerinden 5 tanesi (10, 11, 12, 13, 15 pinleri) input amacıyla (bilgi girişi için) kullanılmaktadır. Projede bilgi için porttan bilgi girişi yapılmamıştır. Ancak, bilgi girişinin LPT'den sağlanması gerekenler durumlar için burada bilgi verilecektir. Portun giriş olarak okunması için paralel port status bellek alanı kullanılmalıdır. Bu bellek alanının adresi taban paralel port adresinin 1 fazlasıdır. Yani, bu projede örnek olarak 379 hex adresi kullanılmıştır. Status portun ilgili pininin okunması için öncelikle 379 hex adresinden alınan değer örnek olarak 10 nolu pine bağlı bir sensör için 20 hex bilgisi ile AND işlemine tabi tutulmalı, elde edilen sepet bilgi eğer ki sıfırdan farklı ise bu pin HIGH (lojik 1 veya analog +5 volt) seviyesinde demektir. Diğer giriş pinlerine bağlı sensör ve arabirimlerin durumu da benzer şekilde ama ilgili hex değeri ile AND işlemine tabi tutularak elde edilebilir.

6.10.4.5 Adım Motorları:

Adım motorların, endüstriyel ve elektronik uygulamalarda kullanımı oldukça fazladır. Adım motorlar, girişlerine uygulanan lojik sinyalleri dönme hareketine çevirirler. İstediğiniz yönde ve derecede döndürebileceğiniz adım motorlar, hassas hareketleri sayesinde, bir çok cihazda konum kontrolü amacıyla kullanılmaktradır.

Step motorlar yapıları gereği, çok sayıdaki uçlarına düzenli olarak gelen palslerle hareket ederler. Her bir adım için belirli uçlara gerilim verilir. Ve step motor bir adım hareket eder ve öylece kalakalır. Sonra yeni palsler farklı bir sırayla tekrar verilir. Bir adım daha gider. Daha sonra yeni bir pals dizisi gelir ve bir adım daha gider. Bu pals dizilerini belirleyen tek şey, motorun içerisindeki, sargıların şekli ve sıralamasıdır. Şekil 6.37'de örnek bir adım motorunun şekli görülmektedir. Projede her bir eksen için birer tane olmak üzere toplam 2 adet adım motoru kullanılmıştır.

Şekil 6.37 Bir Step Motor

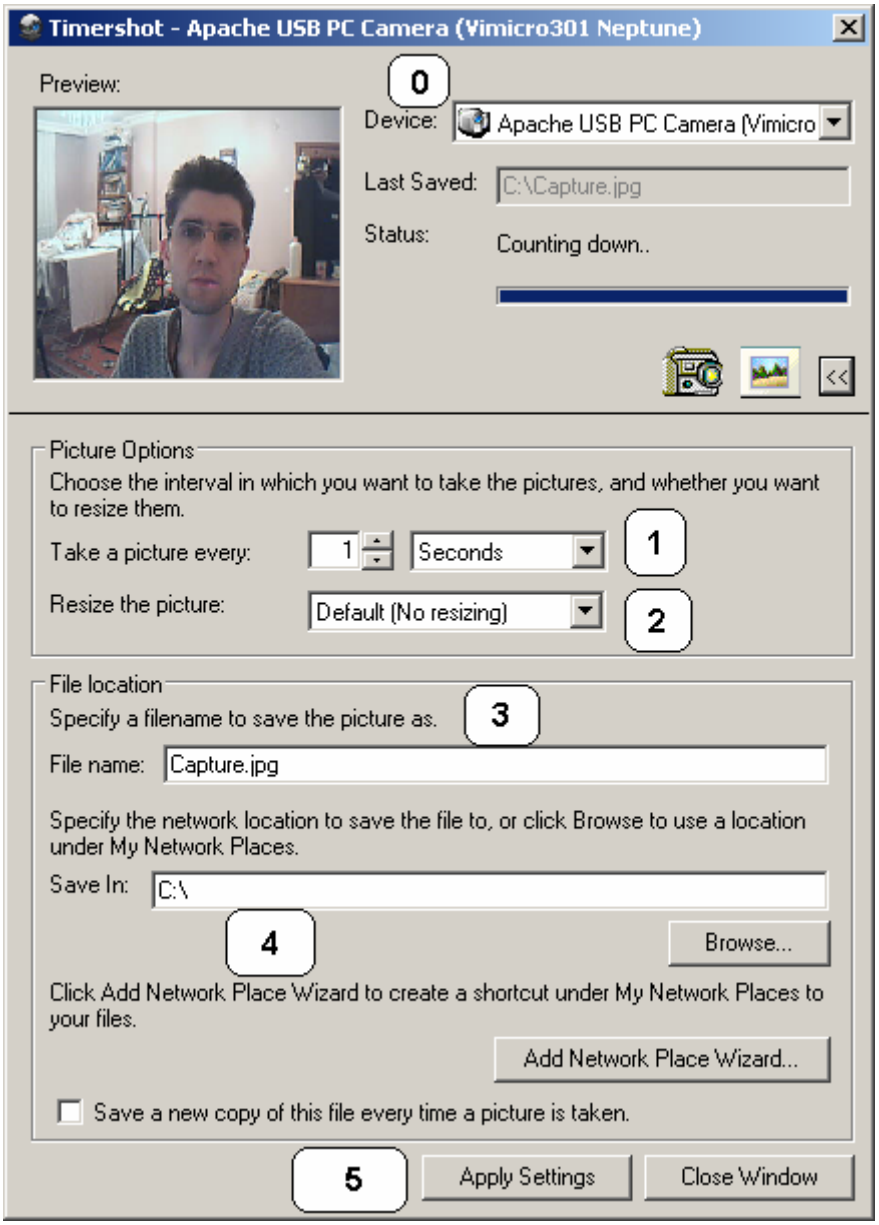
6.10.4.6 Proje Donanım Parçalarının Tümünün Birleştirilmiş Görüntüsü

Projenin donanım elemanlarının tamamen montaj yapılmış hali Resim 6.2'de gösterilmiştir.

Resim 6.2 X ve Y Eksenli WebCam Cihazı (Paralel Port Kontrollü) Görüntüsü

6.10.5 Uygulamanın Kullanılışı

Bu uygulamanın çalışabilmesi için bir görüntü cihazından sürekli olarak görüntülü alan ve her alınan görüntüye harddiske bir resim dosyası olarak kaydeden bir yazılıma ihtiyaç vardır. Bu amaçla projede Microsoft tarfından geliştirilmiş olan "Powertoys for Windows XP" program grubu bünyesinde yer alan "TimerShot" isimli program kullnılmıştır. Bu yazılım tez cdsine de ayrıca eklenmiştir. Bu yazılımın basitçe kullanımına burada yer verilecektir.

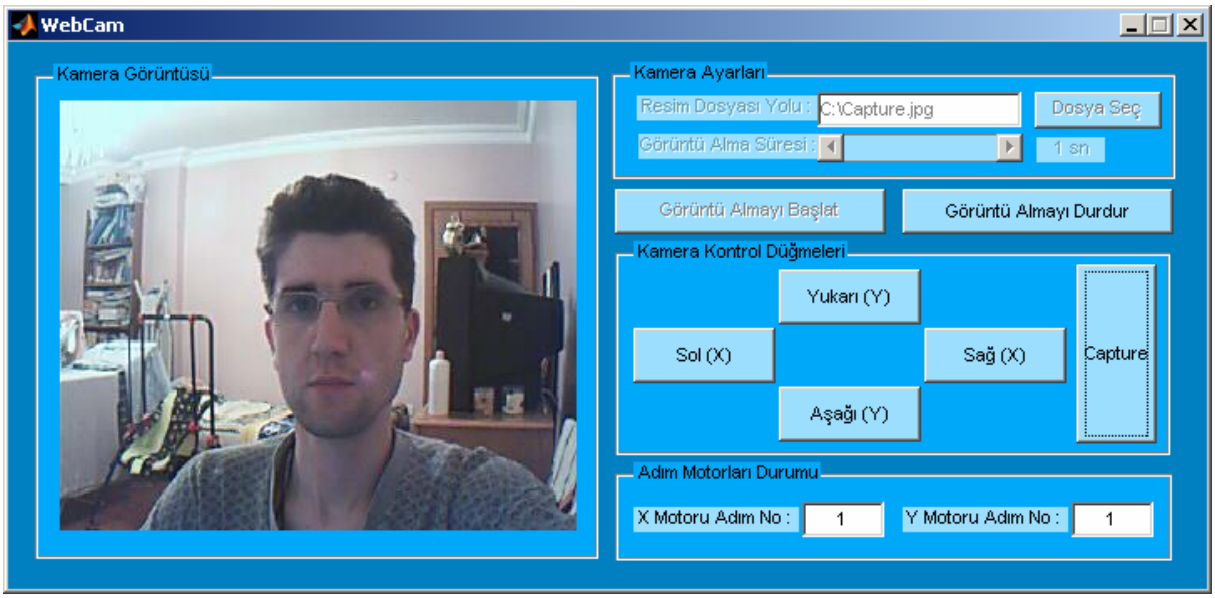


Şekil 6.38 TimerShot Programının Ayarlarının Yapılması

TimerShot programının ekran görüntüsü Şekil 6.38'de gösterilmiştir. Öncelikle programın 0 nolu bölümü kullanılarak bir görüntü aygıtı seçilir. Daha sonra 1 nolu bölüm kullanılarak bu aygıttan programın kaç saniye zaman aralıklarıyla fotoğraf kaydedeceği belirlenir. Örnekte 1 saniye olarak belirlenmiştir. Bundan sonra 2 nolu bölüm ile çekilen fotoğrafın

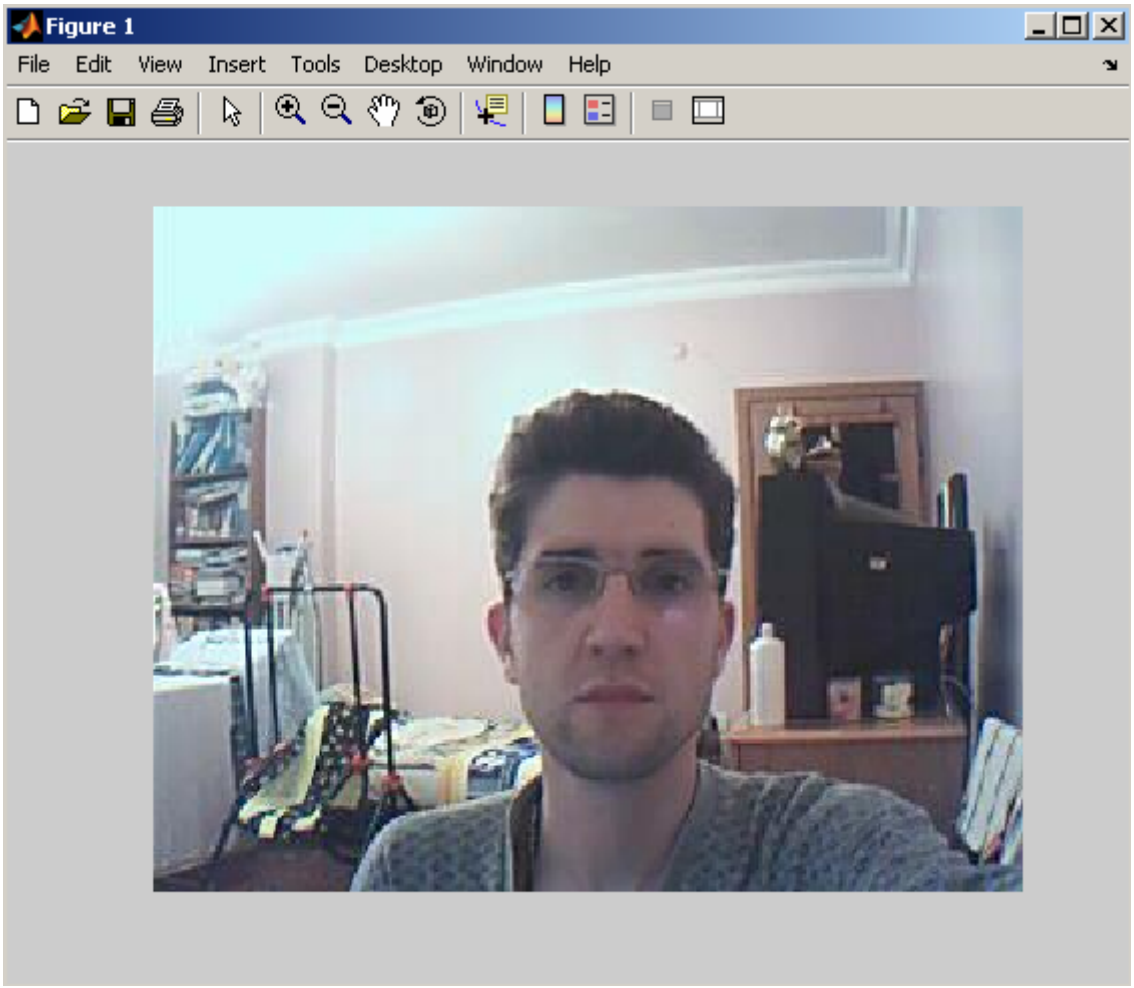
çözünürlüğünen ne olacağı belirlenebilir. Örnekte varsayılan olarak bırakılmıştır. Bu adımdan sonra 3 ve 4 nolu alanlar kullanılarak çekilen fotoğrafın kaydedileceği dosya ismi ve kayıt klasörü bilgileri girilir. En son olarak bu penceredeki tüm yapılan ayarlar "5" numaralı alan kullanılarak, yani "Apply Settings" butonuna basılarak kaydedilir. "close Window" butonu ile de pencere kapatılabilir.

Burada itibaren Matlab ile hazırlnan GUI uygulamsı hakkında bilgiler verilecektir. . Öncelikle programda kullanıcı bilgisayarındaki hangi dosyayı göstereceğini seçer (örnek olarak burada "C:\Capture.jpg" dosyası kullanılmıştır.). Bu işlem için "Resim Dosyası Yolu" alanından "Dosya Seç" butonu yardımıyla dosya seçilir. Daha sonra görüntü dosyasının ne kadar zaman aralıklarıyla kullanıcıya gösterileceği belirlenir. Bunun için "Görüntü Alma Süresi" alanındaki slider kullanılarak zaman aralığı seçilir. En son olarak da "Görüntü Almayı Başlat"



Şekil 6.39 Uygulama 10 Kullanım Ekranı 1

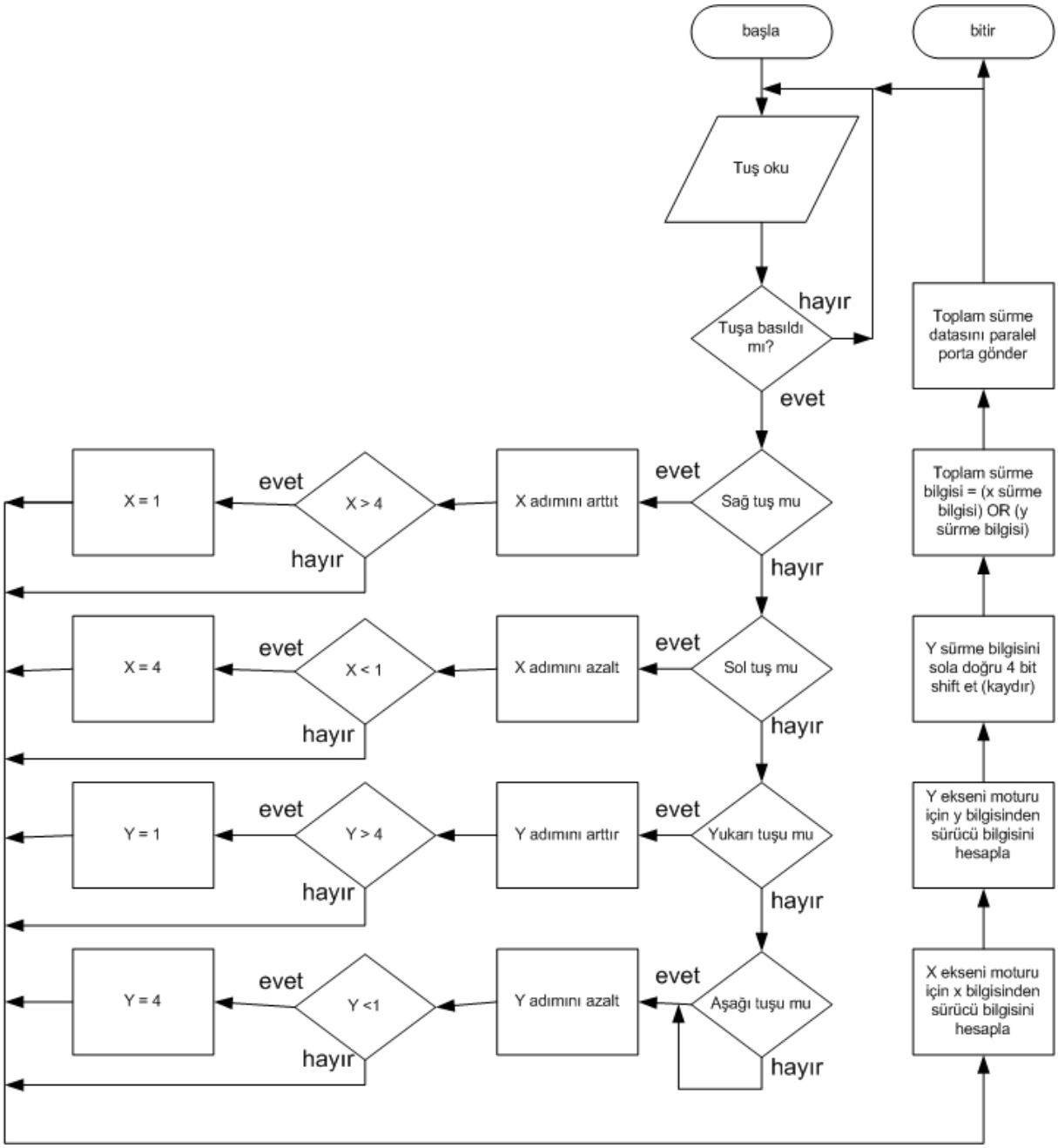
butonu kullanılarak belirtilen zaman aralıklarıyla program resim dosyasından görüntüyü sürekli olarak sunacaktır. Bu işlemi sonlandırmak için "Görüntü Almayı Durdur" butonu kullanılmalıdır. Programı bu aşamadan itibaren çalışmaya başlaması ile elde edilen ekran görüntüsü Şekil 6.39'da gösterilmiştir. Ayrıca, program ayrı bir pencerede de görüntü cihazından ekran görüntüsünü almaktadır. Bu pencere de Şekil 6.40'dan görülebilir. Bu pencere sürekli çekim içindir. Program ile tümleşik ekran görüntüsü penceresi, ancak "Capture" butonu kullnılırsa görüntüyü anlık yakalamak için kullnılır. Programda "Sol" ve "Sağ" düğmeleri yardımıyla X ekseni adım motoru sürülmekte ve "Yukarı" ve "Aşağı" butonları kullanılarak Y ekseni adım motoru sürülmektedir. Her motora ait adım numarası programın "Adım Motorları Durumu" alanından takip edilebilir. Her adım motoru toplam 4 adım sonunda başa dönmektedir.



Şekil 6.40 Uygulama 10 Kullanım Ekranı 2

6.10.6 Uygulamanın Algoritması

Bu uygulamada paralel port kullanılmış ve X ve Y iki ekseni iki farklı adım motoru yardımıyla kontrol edilmiştir. Motorların sürülmesi ve portun kullanımı hakkında bilgi almak için Algoritma 6.12'ye bakılabilir.



Algoritma 6.12 Uygulama 10 Algoritması

BÖLÜM-VII

MATLAB İLE PORT KULLANIMININ GERÇEKLEŞTİRİLMESİ

Matlab ile bilgisayarın dış dünyaya açılan uçları olan portların kullanımı mümkündür. Gerek pararlel, gerekse seri port kullanımı çok kolay bir şekilde gerek Simulink gerekse komut satırı ortamları kullanılarak yapılabilir. Aşağıda paralel ve seri port kullanımın bu ortamlarda nasıl yapılacağı ayrıntılı olarak açıklanmıştır.

7.1 Komut Tabanlı Olarak Portların Yönetilmesi

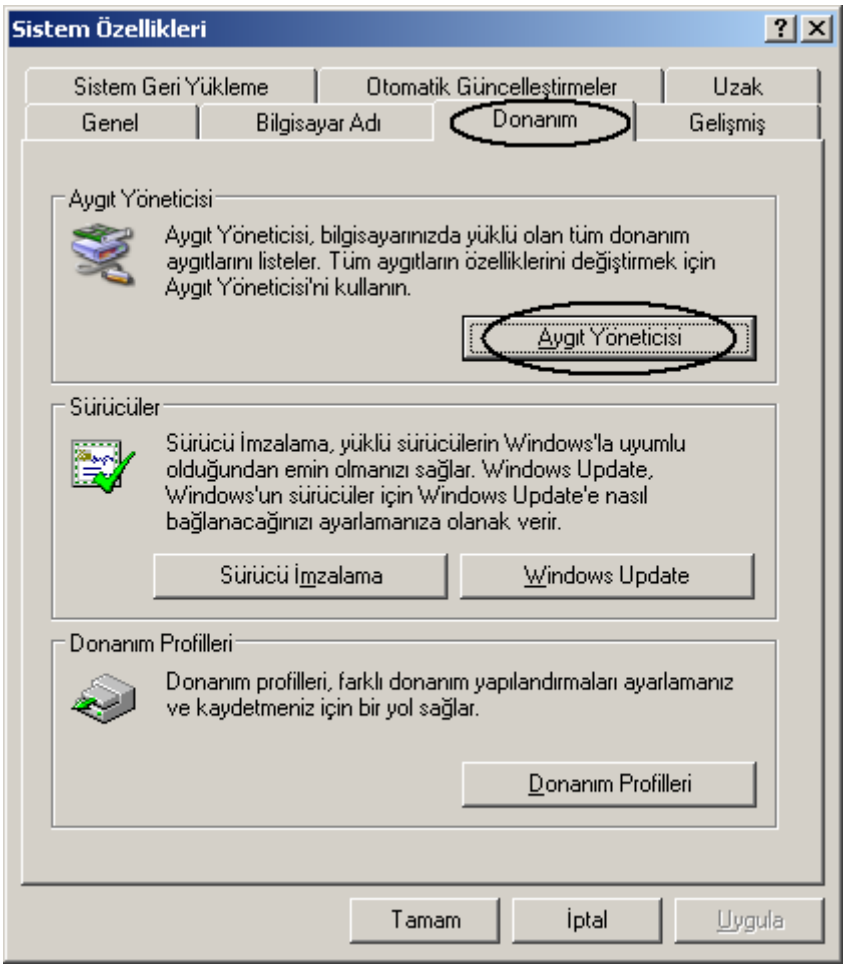
Seri ve paralel portların kullanımı farklı yöntemlerle olduğu için her iki port kullanımı ayrı başlıklar altında incelenmiştir.

7.1.1 Paralel Portun Komutlar Kullanılarak Yönetilmesi

Asagidaki islemi yapmadan önce winio aygitinin isletim sistemi icin register edilmesi gerekmektedir.
Bu islem icin MATLAB komut satirinda asagidaki komutlar icra edilmeli ve PC yeniden baslatilmalidir.

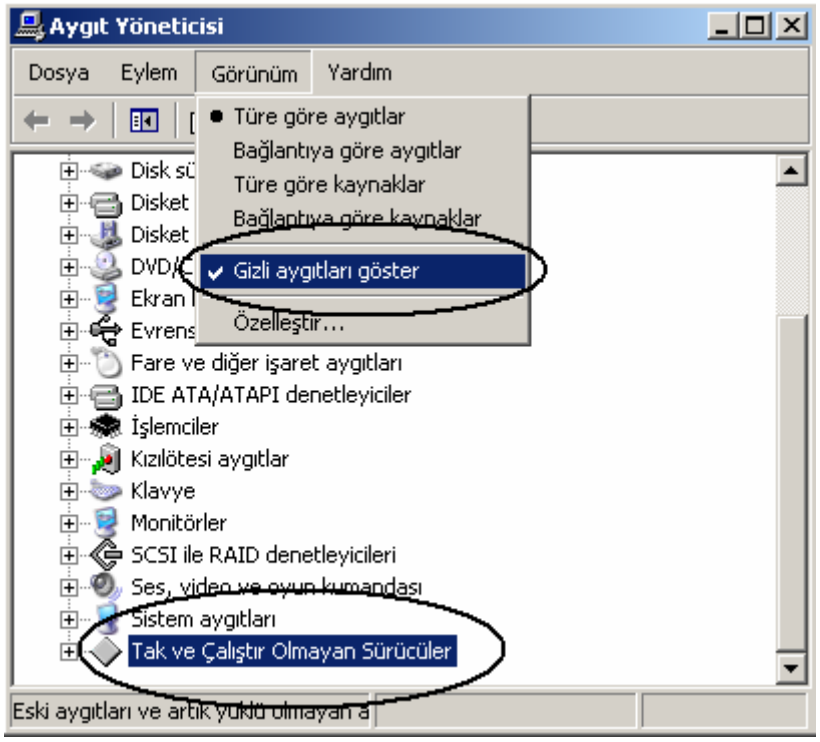
```
daqregister('c:\matlab\toolbox\daq\daq\private\mwparallel.dll'); daqregister('c:/matlab/toolbox/daq/daq/private/winio.dll');
```

Paralel portun kullanılması için öncelikle MATLAB'in paralel portlara erişebilmesi için Windows'ta "WINIO" aygıt sürücüsünün yüklenmesi ve çalışması gerekir. (Bu işlem Windows NT tabanını kullanan Windows 2000, Windows XP ve üst sürümlerinde kullanıcı yetkisi Administrator olanlar dışındaki Normal ve Power kullanıcıların Matlab ile donanıma erişebilmesi için gereklidir.) bu işlemin yapılabilmesi için öncelikleWindows'ta Masaüstü veya Başlat menüsü kullanılarak "Bilgisayarım" ikonu üzerinde farenin sağ tuşu ile tıklanıp açılan menüden "Özellikler" komutu verilerek açılan sistem özellikleri penceresinin

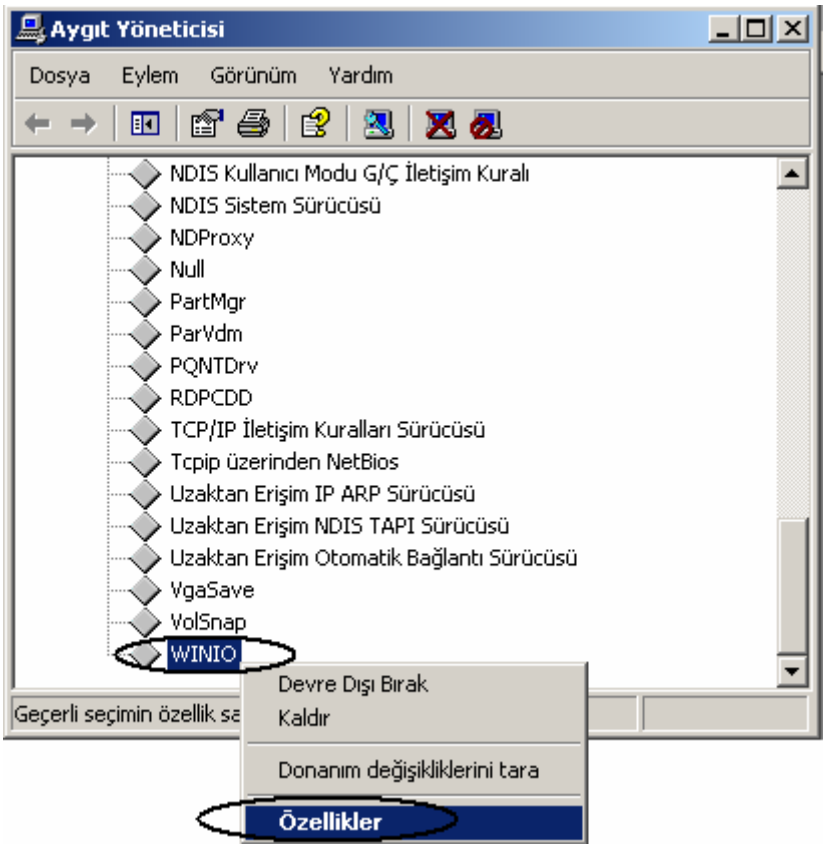


Şekil 7.1 Windows'ta Sistem Özellikleri Penceresinden Aygıt Yöneticisi'nin Çalıştırılması

"Donanım" sekmesinde yer alan "Aygıt Yöneticisi" butonu tıklanır. Bu durum Şekil 7.1'de gösterilmiştir. Daha sonra açılan "Aygıt Yöneticisi" penceresinde "Görünüm" menüsünden" "Gizli Aygıtları Göster" komutu Şekil 7.2'de de gösterildiği üzere çalıştırılır. Aynı pencerede "Aygıt Yöneticisi"nde yer alan aygıt grupları ağaç listesinden "tak ve Çalıştır Olmayan Sürücüler" grubunun altında yer alan "WINIO" aygıt sürücüsü üzerinde farenin sağ tuşu ile tıklanır ve açılan menüden Özellikler komutu verilir. Bu durum Şekil 7.3'te gösterilmiştir.

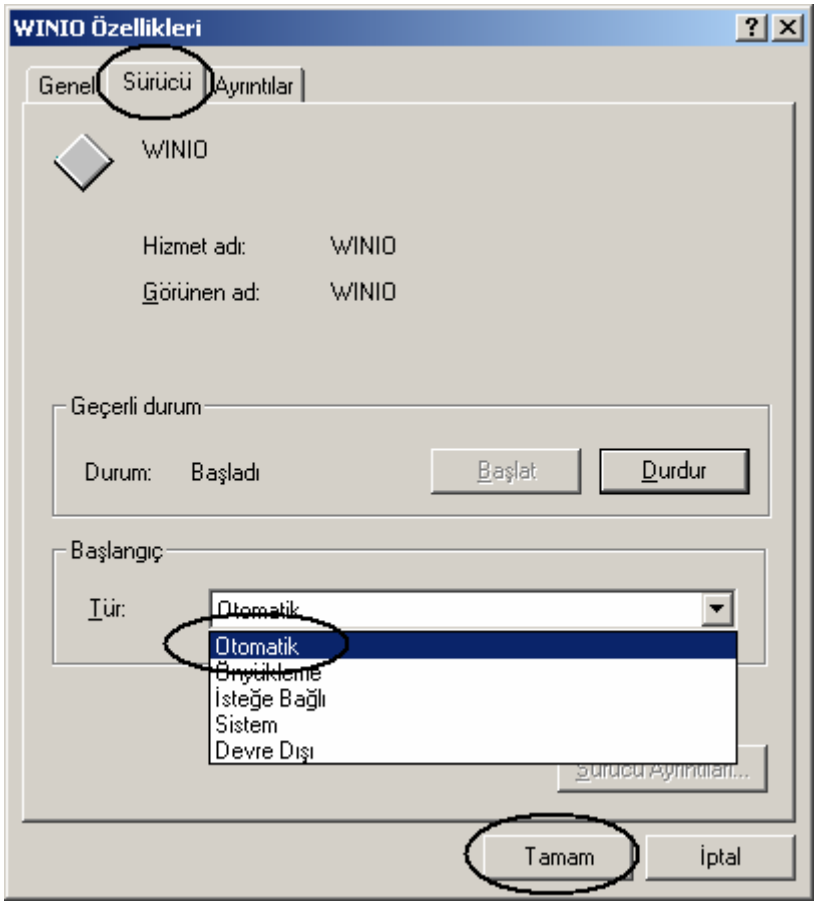


Şekil 7.2 Aygıt Yöneticisi Penceresinde Gizli Aygıtların Gösterilmesi



Şekil 7.3 WINIO Aygıt Sürücüsüne Ait Özellikler Penceresinin Açılması

Bir sonraki adımda açılan “WINIO” aygıt sürücüsüne “Özellikler” penceresinin “Sürücü” sekmesinde yer alan “Başlangıç Türü” seçeneği Şekil 7.4 penceresinde de görüldüğü gibi “Otomatik” olarak ayarlanır. Daha sonra bu pencere “Tamam” butonu tıklanılarak kapatılır. Ayarların aktif olması için bilgisayarın yeniden başlatılması gerekmektedir. Böylelikle



Şekil 7.4 WINIO Aygıtına Ait Ayarların Yapılması

“Administrator” yetkisine sahip olmayan kullanıcıların Matlab kullanarak bilgisayar portlarına erişimi aktif hale getirilmiş olmaktadır.

Matlab içinden paralel portun kullanılması için öncelikle bu portun bir nesne olarak digitalio komutu ile tanımlanması ve bu nesnenin bir değişkene atanması gerekir. Daha sonra portun data bitlerinin kullanıma açılmasını sağlamak üzere ilgili bitleri çıkış amaçlı kullanmak üzere bu nesneye adline komutu yardımıyla kanallar eklenmesi gerekir. En son olarak da bu nesne kullanılarak putvalue komutu ile porta bilgi gönderilebilir veya getvalue komutu ile yine bu nesne üzerinden porttan veri alınması gerçekleştirilebilir. Port ile ilgili veri alma işlemi bittikten sonra mutlaka öncelikle delete komutu ile oluşturulan nense silinmeli ve daha sonra da clear değişken ismi komutu ile bu nesneye ait bilgileri tutatn değişken bellekten temizlenmelidir.

Paralel portun kullanımına örnek komut satırları aşağıda sunulmuştur.

```
paralel_port = digitalio ( 'parallel' , 'LPT1' ) ;          % portun tanımlanması
addline(paralel_port , 0:7 , 'out') ;                       %  portun bitlerinin kull. açılması
dec_data = 88;
bin_data=dec2binvec(dec_data , 8)                           % dec verinin bin formata çevril.
putvalue (paralel_port , dec_data);                         % porta dec veri gönderilmesi
putvalue (paralel_port , bin_data);                         % porta bin veri gönderilmesi
putvalue ( paralel_port , logical ( [ 0 1 0 0 0 1 0 0 ] ) ); % porta direk bin veri gönderilm.
gelen_veri = getvalue(paralel_port);                        % paralel porttan verinin okunması
delete (paralel_port);          % port nesnesinin silinmesi
clear paralel_port;             % port nesnesini tutan değişkenin bellekten temizlenmesi
```

7.1.2 Seri Portun Komutlar Kullanılarak Yönetilmesi

Seri portun kullanılması paralel porta göre çok daha kolay olmaktadır. İlk olarak serial komutu kullanılarak port sınıfını kuracak bir nesne oluşturulur ve daha sonra bu nesne "fopen" komutuna parametre gönderilerek port ile bağlantı kurulur ve seri port bir çıkış aygıtı olmak üzere açılır. Daha sonra "fprintf" komutu ile porta bilgi gönderilir ya da "fget" komutu kullanılarak porttan bilgi okunabilir. Port ile ilgili tüm işlemler tamamlandıktan sonra "fclose" komutu kullanılarak port ile bağlantı koparılmalı ve en son olarak delete komutu ile port ile bağlantıyı tutan değişken bellekten temizlenmelidir.

Seri portun kullanılması ile ilgili komutlar şu şekildedir:

```
seri_port = serial ( 'COM1' , 'BAUD' , 9600 ) ;     % poru kullanacak nesnein oluşturulm.
fopen ( seri_port ) ;                               % port ile bağlantının kurulması
fprintf (seri_port , 'gönderilen integer=%d - kesirli sayı=%3.2f – karakter = %c – string = %s'
, sayi_integer, sayi_float, harf, text_ifadesi);    % portta çeşitli tiplerde verilerin gönderil.
gelen_veri = fget (seri_port);                      % porttan sadece verinin okunması
[ gelen_veri , toplam_veri_uzunluk , hata_mesaji ] = fget (seri_port);% porttan ayrınt.veriokm
fclose(seri_port);                                  % port ile bağlantının koparılması
delete ( seri_port );                               % port nesnesinin silinmesi
clear seri_port;                                    % port nesnesini tutan değişkenin
                                                    % bellekten temizlenmesi
```

7.2 Simulink Ortamı Kullanılarak Portların Yönetilmesi

Simulink ortamında Real Time modunda çalışılarak bilgisayar portları ile bağlantı kurulabilir. Bunun için oluşturulacak bir simulink modelinde,

- Paralel port kullanımı için “Simulink Library Browser” penceresi kullanılarak “Real-Time Windows Target” ağacı altından yapılacak tasarım amacına uygun block eklenmelidir (örneğin dijital veri alınacak ise digital input bloğu konulmalıdır.).
- Seri port kullanımı için “Simulink Library Browser” penceresi kullanılarak “Instrument Control Toolbox” ağacı altından yapılacak tasarım amacına uygun block eklenmelidir. (örneğin porta veri gönderilmek isteniyorsa To Instrument, porttan veri okunmak isteniyorsa Query Instrument blokları tasarıma konulmalıdır.).

Bu konuda daha ayrıntılı bilgi için 3.4 konu başlığına bakınız.

BÖLÜM-VIII
A. TARTIŞMA VE SONUÇ

Bu çalışmanın temel amacı Matlab ile iligli genel bilgilendirmede bulunmak ve Matlab ile nasıl GUI uygulamalrının hazırlanacağı husunda bilgiler vermektir. Çalışmada Matlab programının nasıl kullanılacağı ile ilgili çok çeşitli ve zengin açıklamalar getirilmiştir. Bunun yanında Matlab programı ile GUI uygulamalarının nasıl tasarlanacağı anlatılmıştır. Ayrıca, Kontrol bilimi ile alakalı problemlerin çözümüne yönelik 10 adet GUI uygulaması anlatılmış ve algoritmaları verilmiştir.

Bu tez çalışmasında gerçekleştirilenler dışında yapılmak istenilen bazı hedefler de bulunmaktadır. Bunlar ile ilgili bilgiler aşağıda listelenmiştir.

- Matlab Server programının kullanımı hakkında bilgilendirmede bulunmak,
- Matlab server programı ile tümleşik GUI uygulamaların hazırlanması,
- GUI uygulamalarında gösterilen kontrolör tekniklerine Bulanık Mantık ve Yapay Zeka gibi ileri düzey konuların eklenmesi,
- GUI uygulamalarında özellikle timer nesnesinin kontrolünün iyi derece yapılabilmesi,
- Real Time uygulamaları içeren GUI arayüzlerinin zenginleştirilmesidir.

B. EKLER

Projede kullanılan pdf dosyaları ile hazırlanan GUI uygulamaları M dosyaları ve simulink modelleri tez kitabının en arka sayfasının iç yüzüne eklenmiştir.

C. KAYNAKLAR

C.1 Kitaplar :

1) Hakan AYDIN, "Matlab İle Kontrol Sistemlerinin İncelenmesi", Bitirme Tezi, Marmara Üniversitesi teknik Eğitim Fakültesi, Göztepe, 2003

2) Uğur ARİFOĞLU ve Cemalettin KUBAT ; "MATLAB ve Mühendislik Uygulamaları", Alfa Basım Yayım Dağıtım, İstanbul ,Türkiye, Ekim 2003

3) Yrd. Doç Dr. Mehmet UZUNOĞLU, Ali KIZIL ve Ömer Çağlar ONAR; "Kolay Anlatımı ile İleri Düzeyde MATLAB 6.0 - 6.5", Türkmen Kitabevi, İstanbul ,Türkiye, 2002

4) Anna Kristine Wâhlin; "Matlab Course", Department of Geophysics, University of Oslo, January 2003 (pdf dosyası)

5) Gürsel ŞEFKAT ve İbrahim YÜKSEL, "Matlab Gui Tabanlı Elektromıknatıs Devre Tasarımı ve Analizi", 2003 (pdf dosyası)

6) David F. Griths, "An Introduction to Matlab v.2.3", Department of Mathematics, The University Dundee, September 2005 (pdf dosyası)

7) MathWorks, "MATLAB, The Language of Technical Computing", September 2006 (pdf dosyası)

8) Patrick March and O. Thomas Holland; "Graphics and GUIs with MATLAB®, 3rd Edition", 2002 (pdf dosyası)

9) MATLAB Help Dosyaları

10) MATLAB Demo Uygulamaları

C.2 Elektronik Yayınlar :

1) ftp://ftp.inf.ethz.ch/pub/SolvingProblems/
2) http://hans.munthe-kaas.no/H.Z.Munthe-Kaas.html
3) http://hans.munthe-kaas.no/Research.html
4) http://netec.wustl.edu/
5) http://physics.gac.edu/~huber/envision/flow/welcome.htm
6) http://physics.gac.edu/~huber/envision/matgui/animate.htm
7) http://physics.gac.edu/~huber/envision/matgui/matgui.htm
8) http://texas.math.ttu.edu/~gilliam/ttu/ttu.htm
9) http://web.mit.edu/afs/athena.mit.edu/software/matlab/www/home.html
10) http://wings.buffalo.edu/computing/sc/mat.html
11) http://www.cise.ufl.edu/~davis/sparse/
12) http://www.cise.ufl.edu/research/sparse/
13) http://www.csee.wvu.edu/~trapp/wvumatlab.htm
14) http://www.cyclismo.org/tutorial/matlab/
15) http://www.damtp.cam.ac.uk/user/na/FoCM/Links.html
16) http://www.engin.umich.edu/group/ctm/
17) http://www.elektronikhobi.com/
18) http://www.focm.net/
19) http://www.indiana.edu/~statmath/smdoc/
20) http://www.math.ttu.edu/~gilliam/m5399-matlab.html
21) http://www.math.ufl.edu/help/matlab-tutorial/
22) http://www.math.utah.edu/lab/ms/matlab/matlab.html
23) http://www.maths.dundee.ac.uk/software/
24) http://www.mathworks.com/
25) http://www.mathworks.com/company/newsletters/
26) http://www.mathworks.com/matlabcentral/
27) http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do?objectType=category&objectId=13
28) http://www.mathworks.com/products/demos/index.shtml
29) http://www.mathworks.com/products/matlab/
30) http://www.mathworks.com/support/product/product.html?product=ML
31) http://www.personal.engin.umich.edu/~tilbury/tutorials/matlab.html
32) http://www.rit.edu/~pnveme/Matlab_Course/DEFAULT.HTM
33) http://www.umassd.edu/specialprograms/atlast/welcome.html
34) http://www.utexas.edu/its/rc/
35) http://www.delab.com/

D. ÖZGEÇMİŞ

KENAN SAVAŞ

1983 yılında İstanbul'da doğdum. İlkokul öğrenimime 1990 yılında Cevizli İlkokulu'nda başladıktan sonra, dördüncü seneden itibaren Zeki Gezer İlkokulu'nda devam ettim. İki yıl burada okuduktan sonra ortaokul öğrenimimi Servet Çambol Ortaokulu'nda tamamladım. 1997 yılında Darıca Lafarge Aslan Çimento Endüstri Meslek Lisesi'nin Elektronik Bölümünde ortaöğretim eğitim hayatıma başladım. Birinci sınıfı burada okuduktan sonra Darıca Lafarge Aslan Çimento Teknik Lisesi'nin Bilgisayar Bölümü'ne geçiş yaptım. Ancak, teknik lise dönemimim 3. sınıfından ayrılarak Darıca Aslan Çimento E.M.L. Bilgisayar Bölümünden mezun oldum. 2000 senesinde girdiğim üniversite sınavı sonucu Marmara Üniversitesi Bilgisayar Programcılığı Önlisans Programı'nı kazandım. 2002 senesinde bu programdan bölüm birinciliği derecesi ile mezun oldum. Daha sonra 2003 yılında girdiğim Dikey Geçiş Sınavı sonucu Marmara Üniversitesi Bilgisayar ve Kontrol Öğretmenliği Bölümü'nü kazandım. Halen aynı bölümde son sınıf öğrencisi olarak öğrenimime devam etmekteyim