GRAFİKSEL KULLANICI ARAYÜZÜ YARDIMIYLA GÖRSEL ROBOTİK ARAÇ KUTUSU TASARIMI

SİBEL KAPLAN

Mersin Üniversitesi
Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği
Ana Bilim Dalı

YÜKSEK LİSANS TEZİ

Tez Danışmanı
Yrd.Doç. Hüseyin CANBOLAT

MERSİN
Ocak - 2010

Bu tezin gerek bilimsel içerik, gerekse elde edilen sonuçlar açısından tüm gerekleri sağladığı kanaatine ulaşan ve aşağıda imzaları bulunan biz jüri üyeleri, sunulan tezi oy birliği ile Yüksek Lisans Tezi olarak kabul ediyoruz.

Tez Danışmanı
Yrd.Doç.Dr. Hüseyin CANBOLAT

Jüri Üyesi
Yrd.Doç.Dr. Ali YILDIZ

Jüri Üyesi
Yrd.Doç.Dr. Zeki YETGİN

Bu tezin Fen Bilimleri Enstitüsü yazım kurallarına uygun olarak yazıldığı Enstitü Yönetim Kurulu'nun ..31.../.03.../.2010.tarih ve .2010/08/....192..... sayılı kararıyla onaylanmıştır.

Prof.Dr. Mahir TURHAN
Enstitü Müdürü

ÖZ

Bu çalışmada, komut tabanlı geliştirilmiş ücretsiz dağıtılan *Robotics Toolbox.Version 7 April-2002* araç kutusu içerisinde yer alan tüm komutların, MATLAB editörü olmadan görsel ekranlar aracılığı işletilmesi amaçlanmıştır. Hazırlanan araç kutusu içerisinde Homojen Dönüşümler, Yörünge Türetimi, Kinematikler, Dinamikler adlı konu başlıkları yer almaktadır. Bu konu başlıklarına ait Robotics Toolbox komutları görselleştirilmiştir.

Robotics Toolbox, komutlarının gerektirdiği parametrelerin programa rahatça verilebilmesi amacıyla MATLAB'ın grafiksel kullanıcı arayüzü (GUI) kullanılmıştır. GUI ile geliştirilen ara yüz kullanıcıya gerekli parametrelerin her biri için bir alan ayırdığından kullanıcının komuttaki parametre sayısı ve hangi parametreler olduğu gibi bilgileri önceden ayarlamadan daha hızlı komut işleme ve sonuç almasına yardımcı olmaktadır.

Anahtar kelimeler: Robotics Toolbox, GUI ile Programlama, Görsel Arayüz

ABSTRACT

In this work, the commands of the Robotics Toolbox.Version 7 April 2002 are executed through the visual interface windows out of the standard MATLAB editor. The toolbox contains the Homogeneous Transforms, Trajectory Generation, Kinematics, Dynamics main menu items. The commands under these main menus are transferred to the prepared visual toolbox.

The MATLAB Graphical User Interface (GUI) is utilized to enter the parameters and the arguments of the *Robotics Toolbox* commands. The developed interfaces supply an area for each of the parameters and arguments of the commands. Therefore, the user is not required to know the exact number of parameters and arguments for a given command. The execution of the commands are faster thanks to this feature of the developed visual interface

Keywords: Robotics Toolbox, Programming with GUI, Visual Interface

TEŞEKKÜR

Çalışmamda değerli yorum ve önerileri ile katkıda bulunan tez danışmanım sayın Yrd.Doç Hüseyin Canbolat'a, Matlab GUI ile programlama konusunda yardımcı olan arkadaşlarıma çok teşekkür ederim.

Aynı zamanda tez çalışmalarım süresince bana maddi ve manevi desteklerini esirgemeyen aileme, eşime teşekkür eder ve bu tezimi biricik kızıma armağan ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

ÖZ ..... **i**
ABSTRACT ..... **ii**
TEŞEKKÜR ..... **iii**
İÇİNDEKİLER DİZİNİ ..... **iv**
ŞEKİLLER DİZİNİ ..... **vii**
ÇİZELGELER DİZİNİ ..... **ix**

1. GİRİŞ **1**

2. KAYNAK ARAŞTIRMASI **3**

2.1. MATLAB PROGRAMLAMA DİLİ 4

2.1.1. MATLAB'ın Kullanım Amacı ve Alanı 4
2.1.2. MATLAB'ın Kullanım Yerleri 5
2.1.3. MATLAB'ın Avantajlar 6

2.2. MATLAB GRAFİKSEL KULLANICI ARAYÜZÜ (GUI) 7

2.2.1. MATLAB'da Kullanıcı Arayüzü Nasıl Çalışır? 7
2.2.2. Matlab'te GUI Oluşturma Yöntemleri 7
2.2.3. MATLAB GUIDE Aracı ile GUI Tasarımı Oluşturma 8
2.2.4. GUI Nesnelerinin Açıklanması 10
2.2.4.1. Push Buton 11
2.2.4.2. Toggle Buton 11
2.2.4.3. Radio Buton 12
2.2.4.4. Check Box 12
2.2.4.5. Edit Text 12
2.2.4.6. Static Text 12
2.2.4.7. Slider 12
2.2.4.8. List Box 12
2.2.4.9. Pop-Up Menu 12
2.2.4.10. Axes 13
2.2.4.11 Panel 13
2.2.4.12 Button Group 13
2.2.4.13 ActiveX Component 13
2.2.5. GUI Uygulamalarında Callback Türleri 13
2.2.6. GUI Uygulamalarında Callbackler Arasında Ortak Veri Geçişini Sağlayan Yollar 16
2.2.6.1. Handles Yapı Değişkeni Kullanılarak Global Kullanımı 16
2.2.6.2. Global Değişken Tanımlama Deyimi 17
2.2.6.3. GUI Alanında Visible ve Enable Özellikleri Off Yapılmış Nesne Kullanılması 18
2.2.6.4. Load Ve Save Deyimlerinin Kullanılması 18
2.2.6.5. Nesnelerin Userdata Özelliğini Kullanmak 19
2.2.6.6. Uygulama Datası Yöntemi 20

2.2.7. GUI Uygulamalarında Kullanılan Standart Handle Değişkenleri20
2.2.8. MATLAB GUI Uygulamalarında Etkileşim Kutuları Yönetimi .21

2.3. ROBOTIC TOOLBOX (Robotik Araç Kutusu).....22

3. MATERYAL VE METOD.....29

3.1. GÖRSEL ROBOTİK ARAÇ KUTUSU TASARIMI.....29

3.2. DÖNÜŞÜMLER (*TRANSFORMATIONS*).....31

3.2.1. Dönme Matrisi 31
3.2.2. Öteleme Vektörü 32
3.2.3. Homojen Dönüşüm 33
3.2.4. EUL2TR (Euler Açı Dönüşümü) 34
3.2.5. OA2TR (Yönlendirme ve Yaklaşım Vektör Dönüşümü) 36
3.2.6. ROTX, ROTY, ROTZ 39
3.2.7. RPY2TR (Roll/Pitch/Yaw Açı Dönüşümleri).....40
3.2.8. TR2EUL (Euler Açılara Dönüşüm) 42
3.2.9. TR2ROT (Rotasyon Altmatrisine Dönüşüm) 43
3.2.10. TR2RPY(Roll/Pitch/Yaw Açı Dönüşümleri).....43
3.2.11. TRANSL(Dönüşümsel Matris) 45
3.2.12. TRNORM(Dönüşüm Matrisi Normalizasyonu) 47

3.3. KİNEMATİKLER (KINEMATICS).....48

3.3.1. Düz Kinematikler.....48
3.3.2. Ters Kinematik.....49
3.3.3. Hız Kinematiği.....49
3.3.4. Kullanıcı Tanımlı Robot 49
3.3.5. DIFF2TR(Diferansiyel Hareket Vektör Dönüşümü) 61
3.3.6. FKINE(Düz/İleri Kinematik Hesaplama) 62
3.3.7. IKINE(Ters Kinematik Hesaplama) 65
3.3.8. IKINE560(Puma560 Robotu için Ters Kinematik Hesaplama)...65
3.3.9. JACOB0(Referans Koordinat Sisteminde Jacobian Hesabı) 66
3.3.10. JACOBN (Uç Birim Koordinat Sisteminde Jacobian Manipülator Hesabı) 68
3.3.11. TR2DIFF(Dönüşüm Mat. Diferansiyel Hareket Vektörü).....68
3.3.12. TR2JAC(Homojen Dönüşüm Matrisinden Jacobian Matrisi).....69

3.4. YÖRÜNGE TÜRETİMİ (TRAJECTORY GENERATION).....70

3.4.1. CTRAJ(Kartezyen Yörünge) 70
3.4.2. JTRAJ (Eklem Uzay Yörüngesi) 74
3.4.3. TRINTERP(Homojen Dönüşüm Enterpolasyonu) 74

3.5. DİNAMİKLER (DYNAMICS) 76

3.5.1. ACCEL(Düz/İleri Dinamik Hesabı) 77
3.5.2. CINERTIA (Kartezyen Manipülator Eylemsizlik Matrisi Hesabı) 77
3.5.3. CORIOLIS (Merkezkaç/Coriolus Tork Hesabı) 78
3.5.4. FRICTION (Eklem Sürtünmesi) 78
3.5.5. FTRANS(Kuvvet/Moment(Tork) Dönüşümü) 78
3.5.6. GRAVLOAD(Yerçekimi Etkisi Hesabı) 78
3.5.7. INERTIA (Manipülator Eylemsizlik Matrisini) 79
3.5.8. ITORQUE (Eylemsizlik Tork Hesabı) 79
3.5.9. NOFRICTION(Bir Robotta Sürtünme Etkisi Sıfırlanması) 79
3.5.10. RNE(Ters Dinamik) 80

4. BULGULAR VE TARTIŞMA 81

5. SONUÇ VE ÖNERİLER 82

5.1. SONUÇLAR 82

5.2. ÖNERİLER 82

KAYNAKLAR 84

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1 GUIDE Ekran Görüntüsü....9
Şekil 2.2 GUIDE LAYOUT Editor (GUIDE Çalışma Alanı) Penceresi....9
Şekil 2.3 GUIDE Ekran Görüntüsü-Tasarı....10
Şekil 2.4 Matlab GUI Tasarım Nesneleri....11
Şekil 2.5 Twolink MATLAB 3D Modeli....24
Şekil 2.6 Puma 560 MATLAB 3D Modeli....25
Şekil 2.7 Stanford Arm MATLAB 3D Modeli....25
Şekil 3.1. Programın Çalışır Ekran Görüntüsü....30
Şekil 3.2. Z_0 Ekseni Etrafında Dönme....31
Şekil 3.3. Ötelenmiş Çerçeve....32
Şekil 3.4. Euler Açıları Gösterilimi....34
Şekil 3.5 EUL2TR İçin Hazırlanan GUI Görüntüsü....35
Şekil 3.6 OA2TR İçin Hazırlanan GUI Görüntüsü....37
Şekil 3.7 ROTX İçin Hazırlanan GUI Görüntüsü....39
Şekil 3.8. Roll / Pitch /Yaw Açıları Gösterilimi....40
Şekil 3.9. RPY2TR İçin Hazırlanan GUI Görüntüsü....41
Şekil 3.10 TR2EUL İçin Hazırlanan GUI Görüntüsü....42
Şekil 3.11 TR2ROT İçin Hazırlanan GUI Görüntüsü....43
Şekil 3.12 TR2RPY İçin Hazırlanan GUI Görüntüsü....44
Şekil 3.13 TRANSL İçin Hazırlanan GUI Görüntüsü....46
Şekil 3.14 TRNORM İçin Hazırlanan GUI Görüntüsü....47
Şekil 3.15. *n*+1 Uzva Sahip Kinematik Zincir Oluşturan Manipülatör....48
Şekil 3.16. Kullanıcı Tanımlı Robot İçin GUI Görüntüsü....51
Şekil 3.17 Robot Simülasyonu....51
Şekil 3.18 DIFF2TR İçin Hazırlanan GUI Görüntüsü....61
Şekil 3.19 FKINE İçin Hazırlanan GUI Görüntüsü....62
Şekil 3.20 Kullanıcı Tanımlı Robot Simülasyonu....63
Şekil 3.21 IKINE İçin Hazırlanan GUI Görüntüsü....65
Şekil 3.22. IKINE560 İçin Hazırlanan GUI Görüntüsü....66
Şekil 3.23 JACOB0 İçin Hazırlanan GUI Görüntüsü....67
Şekil 3.24 JACOB0 İçin Robot Simülasyonu....67
Şekil 3.25 TR2DIFF İçin Hazırlanan GUI Görüntüsü....69
Şekil 3.26 Kartezyen Yörünge Türetimi Ekran Görüntüsü....71
Şekil 3.27 Uç Noktanın Kartezyen Koordinatlarda İzlediği Yol....72
Şekil 3.28 TRINTERP İçin Hazırlanan GUI Görüntüsü....75

ÇİZELGELER DİZİNİ

Sayfa

Çizelge 2.1 GUI Uygulamalarında Callback Türler 14
Çizelge 2.2. Matlab ile Tasarlanan GUI Uygulamalarında Kullanılabilecek Etkileşim Kutusu Türleri.....22
Çizelge 2.3. Robotics Toolbox İçin Komutlar 26
Çizelge 3.1. Bir Eklem İçin Alınan Parametreler 50
Çizelge 3.2. Robot Dinamik Denklemindeki Terimlerin Anlamları.....77

1. GİRİŞ

Robotics Toolbox, dünya çapında birçok robotik dersinde kullanılmaktadır ve *Robotics Toolbox*'ı kullanan birçok çalışma bulunmaktadır [1,2]. Yazılımın yaygın kullanılan MATLAB ile uyumlu olması amaçlanmıştır [1,3]. Bu çalışmadaki amaç *Robotics Toolbox*'ı mevcut kullanımından çıkarıp kullanıcıların, görsel bir arayüz ile kullanmalarını sağlamaktır. Mühendislik araştırmalarında çok sık kullandıkları bir program olan Matlab'da, bu konuda görsel bir araç kutusu henüz geliştirilmemiştir. Ancak, Mikrodalga alanında görsel bir araç kutusu daha önceden tasarlanmıştır [4]. Bu çalışmada, Matlab ortamında görsel olarak geliştirdiğimiz "Görsel Robotik Araç Kutusu" tasarımı tanıtılmaktadır.

Robot manipülatörlerinin simülasyonu ve programlanması için farklı yaklaşımlara dayalı paketler geliştirilmiştir [5,6,7]. Robot sistemleri özellikle otomotiv sanayinde yaygın biçimde kullanılmaktadır. Bu gibi yerlerde robotlar arası koordinasyon ve işbirliğinin sağlanması etkin bir veri alışverişine bağlıdır. Bunun için bilgisayar iletişimi önemli bir faktördür [6]. Bu sistemlerin tasarımında teorik performansın elde edilebilmesi için bilgisayar ortamında sistemi tanımlayarak simülasyonlarının yapılması önemli yer tutmaktadır. Fakat her araştırıcı kendi sistemini kendisi bilgisayar ortamında hazırlayarak gerekli testlerden elde edilecek sonuçları görmekte ve fiziksel prototiplerde deneyler yapılmaktadır. Genel olarak kullanılan robot manipülatörleri benzer özelliklere sahiptir. Üstelik simülasyonlar genellikle MATLAB paket programı kullanılarak gerçekleştirilmektedir [8]. Bunların defalarca bilgisayar ortamında yeniden tanımlanması zaman ve emek kaybına neden olmaktadır. Robot simülasyon yazılımı bu gibi ön hazırlıkları önemli ölçüde kısaltacaktır.

Matlab Grafiksel Kullanıcı Arayüzü, diğer bir söylemi ile Matlab GUI, Matlab programcısı tarafından hazırlanan grafik tabanlı uygulamaların, son kullanıcıya fare ve klavye arabirimi ile interaktif olarak hitap etmesini sağlayan bir platformdur [9]. Matlab GUI uygulamalarının gerekliliğinin temel sebeplerinin başında günümüzde

hazırlanan uygulamaların grafik tabanlı oluşu ve bu uygulamaların son kullanıcı tarafından kullanım kolaylığına sahip olması gelmektedir [9].

Matlab GUI' nin çalışması belirli üç temel özellik içerir. Bunlar GUI Yüzeyi, GUI Objeleri ve İşlevlerdir. GUI yüzeyi programda kullandığımız bütün objelerin bulunduğu kısımdır. GUI yüzeyine elemanların yerleştirildiği ve görsel temanın sağlandığı kısımda denilebilir. GUI objeleri programı oluştururken kullandığımız buton, slider, axes gibi her birinin kendine ait bir işlevi olan ve bu işlevlere göre programcının oluşturduğu program yapısında çalışan elemanlardır. En önemli kısım olan işlevler kısmı *(Callback)* bir nesnenin ne yapması gerektiği belirtilen kısımdır. Gerçekte eğer m-function seklinde yazılan grafik tabanlı programlarda nesnelerin *Callback* (işlev)' nin belirtilmesi gerekir [8,9].

Görsel ortamda, program nesne tabanlı olduğundan, bu görsel araç kutusunun kullanımı çok fonksiyonel hale gelmiştir. Kullanıcının Matlab ve Matlab GUI programlama dilini bile bilmesine gerek kalmadan işlemlerini, menüler ve hazırlanan figur dosyalarıyla yapabilmektedir. Kullanıcının sadece bu parametrelere uygun değerleri yazıp, gerekli düğmelere basması yeterlidir.

Çalışmada Matlab GUI'nin nasıl kullanılacağı ile ilgili çok çeşitli ve zengin açıklamalar getirilmiştir. Robotik Toolbox için GUI uygulamaları anlatılmış ve algoritmaları verilmiştir.

2. KAYNAK ARAŞTIRMASI

Yapılan bu tez çalışmasında, komut tabanlı geliştirilmiş ve ücretsiz dağıtılan *Robotics Toolbox.Version 7 April-2002* araç kutusu ve MATLAB programının nesneye yönelik uygulaması olan GUI (Grafiksel Kullanıcı Arayüzü) kullanılmıştır. Geliştirilen görsel robotik araç kutusunun (GRAK) grafiksel kullanıcı ara yüzünün hazırlanmasında, bir ana ekran kullanılmıştır. Tüm işlemler bu ekran aracılığı ile yürütülecek şekilde işlemler yapılmıştır.

Robotics Toolbox kinematikler, dinamikler ve yörünge türetme gibi robotikte geçerli birçok işlevi oluşturmada yararlı bir modelleme aracıdır. Robotics Toolbox gerçek robotlarla yapılan deneyleri çözümlemek kadar önemli olan 3D benzetimi yerine getirir. Peter I. Corke(CSIRO Manufacturing Science and Technology-Australia) tarafından 1996 yılında ilk versiyonu çıkarılmıştır. Şu anda kullanılan versiyon Release6'dır ve yazılım tarihi 2001'dir. Yazılım http://www.mathworks.com/ adresinden ücretsiz olarak edinilebilir. Robotics Toolbox seri uzuvlu manipülatörler için yazılmıştır. Matlab'ın genel özelliği olarak yazılıma müdahale edilebilmektedir. Böylece kullanıcı belirli konular için kendi benimsediği algoritmaları kullanarak gerekli hesaplamaları yapabilir.

Bu çalışmaya başlamadan önce robotik ve GUI üzerine yapılan çalışmaları incelediğimizde, Robotik Toolbox'un komutlarının GUI üzerinde görselleştirilmesi üzerine herhangi bir çalışmaya rastlanmamıştır. Ancak başka konularda GUI üzerinde çalışmalar bulunmaktadır. 2004 yılında Young Jin Choi, ve ark. tarafından yapılan bir çalışmada "Tomoğrafik" görüntüleri işlemede kullanıcı arabirimi kullanılmıştır. 2005 yılında M.Sait Vural tarafından GUI ile "Mikrodalga Araç Kutusu" tasarlanmıştır.

Bütün bu çalışmalar incelendikten sonra MATLAB'ın 6.5 versiyonu ile birlikte gelen GUIDE'ın yaygın olarak kullanılmadığı, Robotik Toolbox'da ise genel işlemlerin tümü için değilse sadece spesifik konular için kullanıldığı görülmüştür.

Ayrıca yapılan dosyaların derlenmediği, sadece MATLAB içerisindeki bir klasörde tutulduğu görülmüştür.

2.1. MATLAB PROGRAMLAMA DİLİ

MATLAB; (MATrix LABoratory); ilk defa 1985'de C.B Moler tarafından matematik ve özellikle de matris esaslı matematik ortamında kullanılmak üzere geliştirilmiş etkileşimli bir paket programlama dilidir. İlk sürümleri FORTRAN diliyle yazılmış olmakla beraber son sürümleri (2009 yılı itibariyle MATLAB 9) C dilinde hazırlanmıştır. MATLAB mühendislik alanında; sayısal hesaplama, veri çözümleri ve grafik işlemlerinde kullanılabilecek genel amaçlı bir program olmakla beraber özel amaçlı modüler paketlere de sahiptir. Robotics Toolbox, Control Toolbox, Signal Toolbox gibi paket programlar (bilgisayar destekli denetim sistemi tasarımı) paketler olup bunlar denetim sistemlerinin tasarımında çok etkili araçlardır. Ayrıca WINDOWS ortamında çalışan SIMULINK, etkileşimli benzetim programlarının hazırlanması ve çalıştırılmasında büyük kolaylıklar sağlamaktadır.

MATLAB, çok yönlü bir teknik hesaplama ortamı olarak matematiksel işlemler, veri analizi ve işlemesi, görselleştirme ve kuvvetli bir programlama dili yapısı gibi gereksinimlerin bir birleşimidir.

2.1.1. MATLAB'ın Kullanım Amacı ve Alanı

MATLAB tüm mühendislik alanında, sayısal hesaplamalar, veri çözümlenmesi ve grafik işlemlerinde kolaylıkla kullanılabilen bir program dilidir. FORTRAN ve C dili gibi yüksek seviyeli programa dili ile yapılabilen hesaplamaların pek çoğunu MATLAB ile yapmak mümkündür. Ayrıca bunu yanında diğer programlama dillerine göre, MATLAB'ta daha az sayıda komutla çözüm üretmek mümkündür. Gerçekte MATLAB, M-dosyaları (M-Files) olarak biline pek çok sayıda fonksiyon dosyalarından, alt programlardan ibarettir. Hazırlanması düşünülen bir program içinde M-dosyalarını kullanmak suretiyle komut sayısını çok kısa tutmak mümkündür.

Bütün dünyada teknoloji geliştiren tüm bilim adamları, şirketler ve araştırmacılar, çalışmalarını hızlandırmak, analiz ve metot geliştirme zamanlarını en aza indirgemek ve piyasaya daha gelişmiş ürünler sunmak için MATLAB kullanmaktadırlar.

MATLAB'ın açık mimari yapısı ve kullanım kolaylığı; yapay sinir ağlarından güç sistemleri analizine, görüntü işlemeden finanssal modelleme araçlarına, kontrol sistemlerinden veri tabanı uygulamalarına, havacılık sistemlerinden otomotiv uygulamaları konularına ve çok daha fazla sayıdaki farklı alanlara yönelik özel ürünleri, kullanıcılara problemlerini en hızlı ve kolay yoldan çözebilme, algoritma hazırlama ve kişisel araçlar ve fonksiyonlar geliştirebilme imkanı tanır.

MATLAB, projelerdeki analiz ve geliştirme sürelerini azaltmakta, proje maliyetlerini indirmekte ve hemen her matematik-mühendislik-finans probleminde etkin çözümler sunabilmektedir. MATLAB hemen her konuya ilişkin çözümler sunabilen ürünlerini aynı platformda buluşturmaktadır. Böylelikle, mühendislerin aynı platformu kullanarak çalıştıkları projelerin farklı proseslerini zaman kaybına uğramadan işlemelerine, platform değiştirmeksizin etkin bir şekilde geliştirebilmelerine ve hayal gücünün sınırlarında diğer ürünler ile etkileşim içinde çalışma imkan tanır. Kullanıcılar, çok yönlü ve görsel MATLAB grafiksel kullanıcı arayüzünü, yazı yazma kolaylığındaki dil yapısını ve matematiksel-grafiksel hazır fonksiyon özelliklerini, C, Fortran veya diğer dil ve uygulamalara tercih etmektedir. MATLAB kullanılarak yapılan uygulamaların, C ve C++ dillerine dönüştürülebilir olması ve C, C++, Java veya Fortran'da hazırlanmış rutinlerin MATLAB'da kullanılabilir olması birçok araştırmacıyı MATLAB kullanmaya yöneltmiştir.

2.1.2. MATLAB'ın Kullanım Yerleri:

- Denklem takımlarının çözümü, doğrusal ve doğrusal olmayan diferansiyel denklerinin çözümü, integral hesabı gibi sayısal hesaplamalar
- Veri çözümleme işlemleri

- İstatiksel hesaplamalar ve çözümlemeler
- Grafik çizimi ve çözümlemeler
- Bilgisayar destekli denetim sitemi tasarımı
- Devre analizinde düğüm kol denklemlerinin çözümü

2.1.3. MATLAB'ın Avantajları

Kullanım Kolaylığı: Matlab'ın sahip olduğu kolay kodlama algoritması ile yeni programlar çok hızlı bir şekilde oluşturulabilmektedir. MATLAB'ın geliştirme araçları kullanımı daha kolaylaştırmaktadır.

İşletim Sistemi Uyumluluğu: MATLAB birçok işletim sisteminde hatasız bir şekilde çalışmaktadır.(Windows Sürümleri, Unix tabanlı işletim sistemleri, Macintosh İşletim Sistemi)

Hazır Fonksiyonlar: Matlab, bir çok teknik probleme karşılık veren ve önceden yazılmış geniş bir fonksiyon kütüphanesiyle birlikte sunulmaktadır.

Donanım Kısıtlamasız Görüntüleme: Diğer programlama dillerine göre, Matlab birçok önemli görüntüleme ve çizim komutu içermektedir. Figürler ve görüntüler, işletim sisteminin desteklediği herhangi bir grafik adaptöründen görüntülenebilmektedir.

Grafiksel Kullanıcı Arayüzü Geliştirme: Grafiksel Kullanıcı Arayüzü geliştirme eklentisi, Matlab diliyle deneyimsiz kullanıcılara teknik veri işleme ve görüntüleme imkanı sağlar.

Matlab Derleyicisi(Compiler): Matlab derleyicisi, Matlab ile geliştirilen uygulamaları Matlab'ın yüklü olmadığı sistemlerde de çalıştırılabilir hale getirmektedir.

2.2. MATLAB GRAFİKSEL KULLANICI ARAYÜZÜ (GUI)

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) dır. Bu uygulama sayesinde kullanıcıya fare ve klavye aracılığı ile interaktif bir uygulama sunmuş oluruz. Böylece kullanıcı herhangi bir komut yada fonksiyon parametresi bilmeden uygulamayı çalıştırması sağlanacaktır.

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.

2.2.1. MATLAB'da Kullanıcı Arayüzü Nasıl Çalışır?

Her bir nesne (veya component) 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.

2.2.2. Matlab'ta GUI Oluşturma Yöntemleri

MATLAB GUI tasarımları iki ayrı yöntem kullanılarak yapılabilir. Bunlar,

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

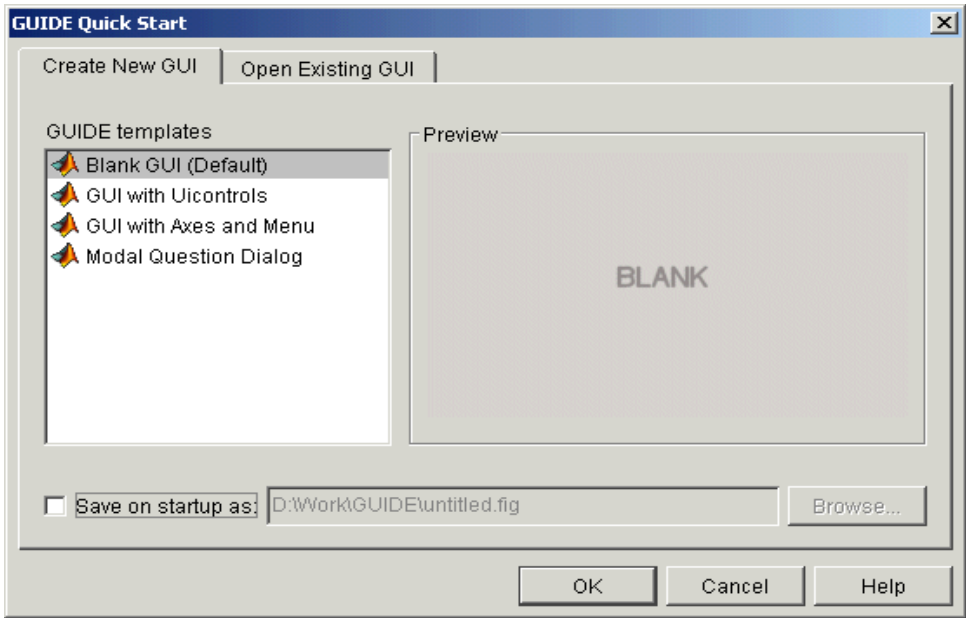
Özellikle GUI tasarımında hızlı arayüzler dizayn etmek ve bu işe ilk başlayan programcılar için MATLAB GUIDE aracının kullanılması büyük bir kolaylık sağlar. Bu aracın kullanılması ile GUI arabirimi kolaylıkla ve yorulmadan sürükle bırak ve açılan pencerelerde özelliklerin değiştirilmesine dayanan bir 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 tamamı 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.

2.2.3. MATLAB GUIDE Aracı ile GUI Tasarımı Oluşturma

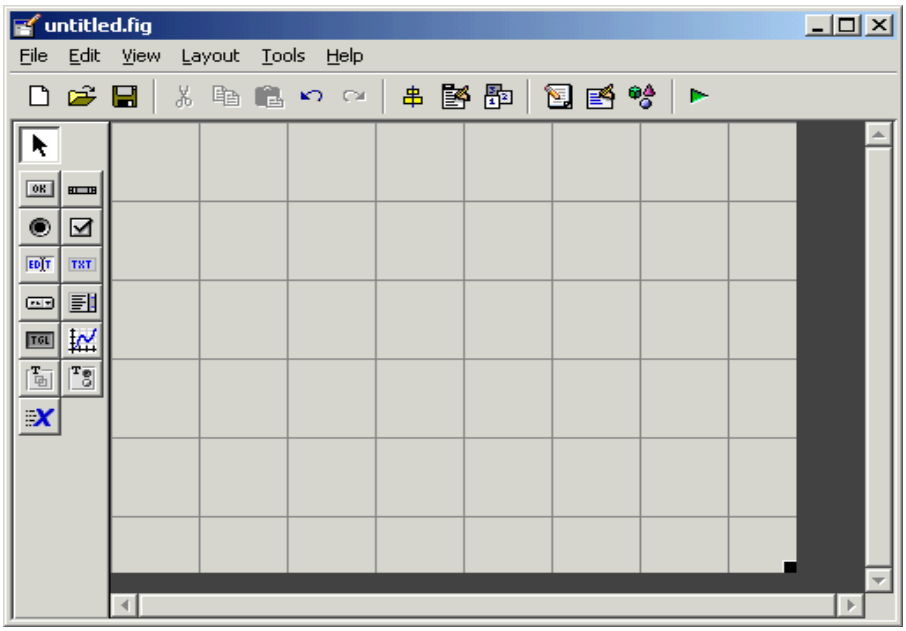
GUIDE Matlab'ın GUI tasarımcılarına sunduğu içerisinde çeşitli araçlar içeren ve kolaylık sağlayan bir grafiksel GUI geliştirme ortamıdır. GUIDE kullanılarak tıkla ve sürükle-bırak tekniği ile GUI arayüzüne nesneler (örneğin butonlar, text kutuları, liste kutuları, grafikler v.s.) kolaylıkla eklenebilir. 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 2.1 deki gibi bir pencere gelir.



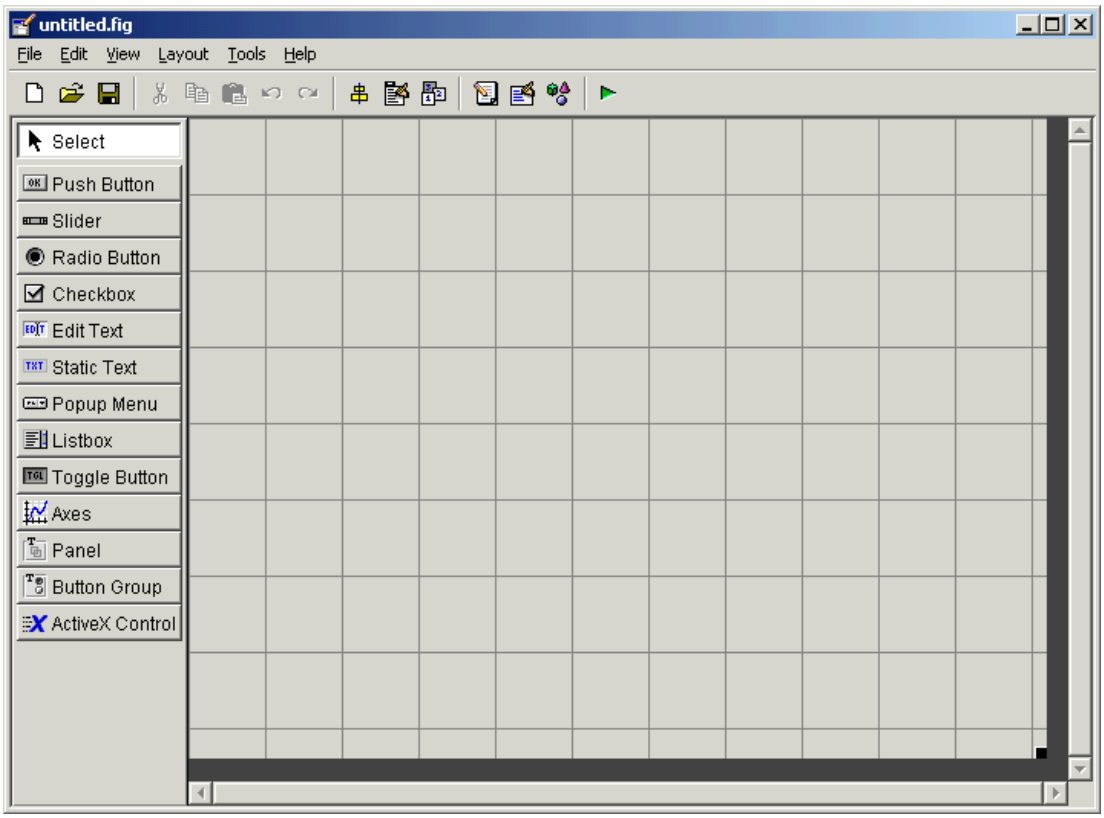
Şekil 2.1 GUIDE Ekran Görüntüsü

Bu pencereden eğer yeni bir GUI tasarımı yapacak isek Blank GUI seçeneğini seçeriz. Şayet önceden yapılmış bir tasarımı açmak istiyor isek Open Existing GUI sekmesinden sonra istenilen dosyayı seçeriz. Burada yeni bir tasarım oluşturulacağını kabul edelim. Bundan sonra OK düğmesi tıklanılarak Şekil 2.2'teki GUIDE LAYOUT Editor (GUIDE Çalışma Alanı) penceresine ulaşırız.



Şekil 2.2 GUIDE LAYOUT Editor (GUIDE Çalışma Alanı) Penceresi

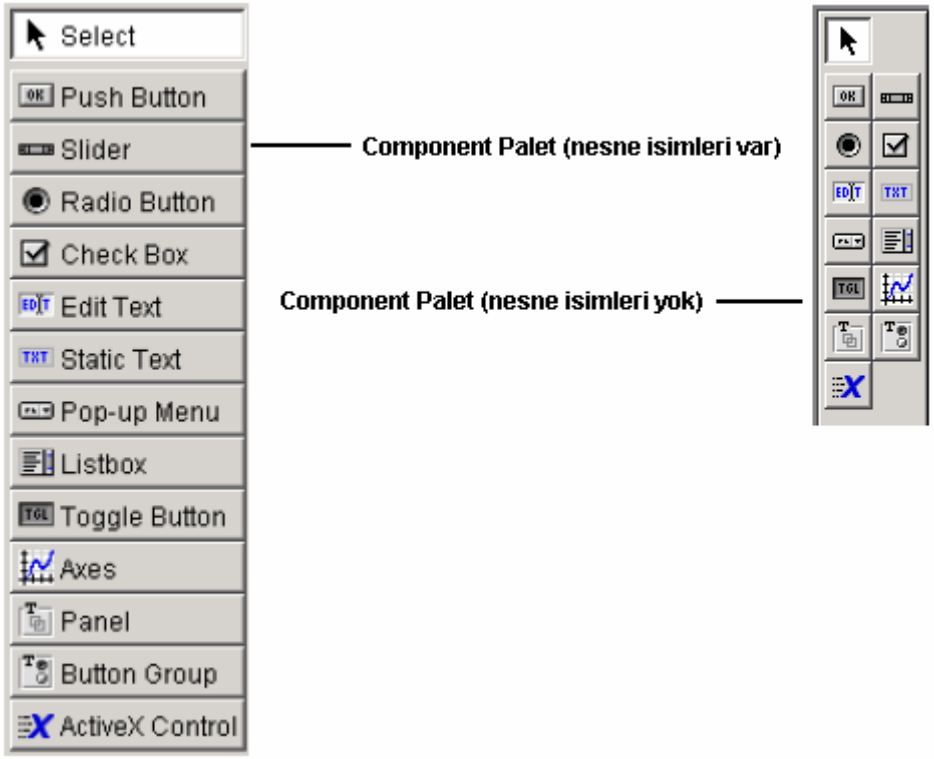
Bu adımdan sonra File/Prefences/Guide yolunu kullanılarak gelen pencereden "Show names in component palette" seçeneğini tıklayıp OK düğmesine basalım. Karşımıza Şekil 2.3'teki gibi bir pencere gelecektir.



Şekil 2.3 GUIDE Ekran Görüntüsü-Tasarım

2.2.4. 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 2.4 Matlab GUI Tasarım Nesneleri

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

2.2.4.1. Push Button:

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

2.2.4.2. 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

2.2.4.3.  Radio Buton:

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

2.2.4.4.  Check Box:

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

2.2.4.5.  Edit Text:

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

2.2.4.6.  Static Text:

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

2.2.4.7. Slider:

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

2.2.4.8.  List Box:

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

2.2.4.9.  Pop-Up Menu:

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

2.2.4.10.  Axes:

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

2.2.4.11.  Panel:

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

2.2.4.12.  Button Group:

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

2.2.4.13.  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.

2.2.5. GUI Uygulamalarında Callback Türleri

Callbackler oluşan herhangi bir olaya bağlı olarak her nesne için ayrı ve olayın türüne göre icra edilen alt program parçalarıdır. Aşağıda sunulan çizelgede kullanılan callbacklerin işlevi ve hangi nesneler ile birlikte kullanıldığı belirtilmiştir.

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.

Çizelge 2.1. GUI Uygulamalarında Callback Türleri

<table>
<tr><th>Callback Türü</th><th>Tetiklendiği Olay</th><th colspan="2">Kullanıldığı Nesneler</th></tr>
<tr><td rowspan="3">ButtonDownFcn</td><td rowspan="3">Fare göstergeci Figure veya bir nesnenin kenarlarından 5 piksel içerde olduğunda fare butonu tıklandığında oluşur. UI control için Enable özelliğinin true olmalıdır</td><td>axes</td><td>figure</td></tr>
<tr><td>button group</td><td>panel</td></tr>
<tr><td>User interface conrols</td><td></td></tr>
</table>

Çizelge 2.1. (devam)

Callback Türü	Tetiklendiği Olay	Kullanıldığı Nesneler	
Callback	Nesnenin temel olayıdır. Örneğin bir push button tıklandığında ya da bir menü öğesi seçildiğinde oluşur.	context menu	menu
		user interface conrols	
CloseRequestFcn	Figure kapanmadan önce çalıştırılır.	figure	
CreateFcn	Bir nesnenin create edilmesi anında oluşur. Nesne oluşturulduğunda initializing için kullanılabilir. Bu olay nesne create edilince ancak nesne GUI alanında gözükmeden önce icra edilir.	axes	fgure
		button group	context menu
		menu	panel
		user interface conrols	
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 conrols	
KeyPressFcn	Figure veya bir nesne focus (aktif) olduğunda veya klavyeden herhangi bir tuşa basıldığında oluşur.	figure	
		user interface controls	
ResizeFcn	Panel, button group veya figure nesnelerinin boyutları kullanıcı tarafından değiştirildiğinde oluşur. Ayrıca, bu durumun gerçekleşmesi için figure ait "Resize" özelliği "on" olmalıdır.	button group	figure
		panel	

Çizelge 2.1. (devam)

Callback Türü	Tetiklendiği Olay	Kullanıldığı Nesneler	
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 hareket 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 isimlendirilebilir.			

2.2.6. GUI Uygulamalarında Callbackler Arasında Ortak Veri Geçişini Sağlayan Yollar

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

2.2.6.1. Handles Yapı Değişkeni Kullanılarak Global Kullanımı

GUI uygulamalarında en sık kullanılan yöntemdir. Bu yöntemde 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 local bir "sistem_cikis_sayisi" isminde değişken oluşturulmuş ve içeriği 4 olarak atanmıştır. Ancak, local bu değişkenin değeri ve kendisinin varlığı callback fonksiyonunun icrası tamamlandıktan sonra kaybolacaktır. Tekrar aynı callbacke 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 callback çağrıldığında bu yapı içerisinden kullanılmasına imkân verilmektedir.

Örneğin başka bir callback içerisinde 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 içerisinde icra edilmesi düşünülen komut satırları
End
```

2.2.6.2. 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 deyimi ile tanımlanan değişken bellekte bir yere sahip ve değeri var ise bu değerini 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.

2.2.6.3. 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.

2.2.6.4. 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.

2.2.6.5. 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 tutabildiği gibi sayısal tip verileri de direk olarak gönderme ve kullanma imkânı programcıya sunulur. Yani, aralarda 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.

2.2.6.6. Uygulama Datası Yöntemi

En gelişmiş, ancak kullanımı biraz zahmetli olan yöntemdir. Bu yöntem ile herhangi bir GUI nesnesinin var olan özelliklerine sanki handles yapısına yeni bir değişken ekler gibi yeni bir değişken eklenebilir. Eklenen nesne genellikle figure olduğu için yöntem ismini buradan almaktadır. Örnek olarak toplam_boyut alanının bir herhangi bir nesne callback'inde icrası sonucu bahsedilen değişken 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 içerisinde application data saklanılan nesne edit4 isimli bir nesne olsun. Buna göre ilgili komutlar şu şekilde yazılmalıdır:

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

2.2.7. GUI Uygulamalarında Kullanılan Standart Handle Değişkenleri

GUI uygulamalarında birden fazla nesne veya figure alanı ile çalışıldığı düşünülürse bazen yanlış veya istenmeyen nesnelere kontrollerin kaydığı görülebilir. Bu gibi durumlardan sakınmak için aktif axes veya grafik nesnesi gibi bazı belirli nesneler için özel olarak handle numarasını tutmak amacıyla tanımlanmış değişkenler Matlab 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 numarasını 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 figurin içerdiği 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 numarası da elde edilebilir.

2.2.8. 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 kullanıcıların 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 bilgilendirilebilir. 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 karşılaşmamış olur. Ayrıca, etkileşim kutularının kullanımı ile programcının da GUI arayüzünde herhangi bir tasarım

değişikliğine gitmesine fırsat verilmez, dolayısıyla daha etkin GUI tabanlı uygulama geliştirme imkânı sağlanmış olmaktadır.

Matlab ile hazırlanan GUI uygulamalarında kullanılabilecek etkileşim kutuları aşağıdaki çizelgede gösterilmiştir.

Çizelge 2.2. Matlab ile Tasarlanan GUI Uygulamalarında Kullanılabilecek Etkileşim Kutusu Türleri

Etkileşim Kutusu Türü	**Açıklama**
Errordlg	Hata Penceresi
Helpdlg	Yardım Penceresi
İnputdlg	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

2.3. ROBOTIC TOOLBOX (Robotik Araç Kutusu)

Robotik fiziksel aktivite ve karar verme gibi uygulamalarla bir görevi yürüterek insanların yerini alabilecek makinelerle ilgili çalışmaları içerir. Robotik,

geleneksel mühendislik sınırlarını kesiştiren yeni bir modern teknoloji alanıdır. Robotların karmaşıklığını ve uygulama alanlarını anlamak elektrik-elektronik mühendisliği, makine mühendisliği, endüstri mühendisliği, bilgisayar mühendisliği, matematik alanlarında geniş bir bilgi ağı gerektirmektedir.

Robot, bir dizi verilen görev çerçevesinde çeşitli programlanmış hareketler ile materyalleri, parçaları, aletleri veya özel donanımları hareket ettirmek için tasarlanmış programlanabilir çok işlevli manipülatördür. Bu tanım Robot Institute of America-RIA tarafından verilmiştir.

Robot dört ana kısımdan meydana gelir :

1. Bir mekanik yapı ya da eklemlerle birbirine bağlanmış sıralı rijid cisimlerden(uzuvlardan) oluşan manipülatör; manipülatör, serbestliği sağlayan bir koldan(arm), el becerisi sağlayan bir bilekten(wrist) ve robotun yapması gereken görevi tamamlayan sonlandırıcıdan(end effector) oluşmaktadır.
2. Eklemlerin hareketlenmesiyle manipülatörün hareketini sağlayan hareketlendiriciler(actuators-motors)
3. Manipülatörün veya çevrenin durumunu gözleyen algılayıcılar(sensors)
4. Manipülatör hareketini kontrol eden ve yöneten bir kontrol sistem,

Robotics Toolbox kinematikler, dinamikler ve yörünge türetme gibi robotikte geçerli birçok işlevi oluşturmada yararlı bir modelleme aracıdır. Robotics Toolbox gerçek robotlarla yapılan deneyleri çözümlemek kadar önemli olan 3D benzetimi yerine getirir. Peter I. Corke(CSIRO Manufacturing Science and Technology-Australia) tarafından 1996 yılında ilk versiyonu çıkarılmıştır. Şu anda kullanılan versiyon Release6'dır ve yazılım tarihi 2001'dir. Yazılım http://www.mathworks.com/ adresinden ücretsiz olarak edinilebilir.

Robotics Toolbox seri uzuvlu manipülatörler için yazılmıştır. Matlab'ın genel özelliği olarak yazılıma müdahale edilebilmektedir. Böylece kullanıcı belirli konular için kendi benimsediği algoritmaları kullanarak gerekli hesaplamaları yapabilir.

Robotics Toolbox'ta robot modeli tanımlama iki şekilde olabilir :

1. *Kendi modelini oluşturma :*

Robotics Toolbox'ta robot modeli tanımlamak için her uzuv için DH parametrelerini tanımlamak ve uzuvlarla robotu oluşturmak yeterlidir.

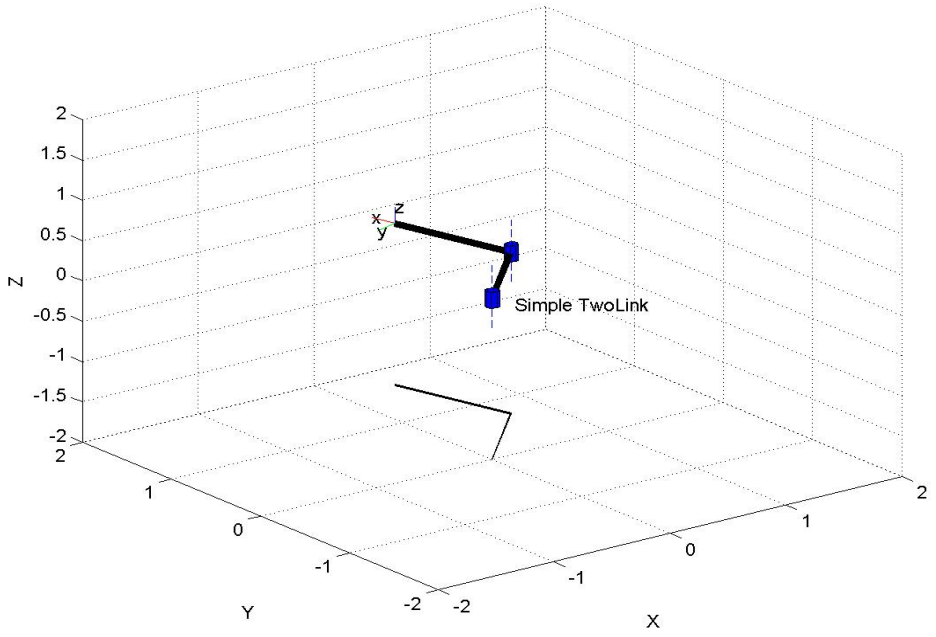
```
L1= link([alpha1, a1, theta1, d1, sigma1])
L2= link([alpha2, a2, theta2, d2, sigma2])
r=robot({L1 L2 . . . })
```

Burada link komutu ile uzuv nesnesi tanımlanır. alpha1, a1, theta1, d1 değerleri ise uzvun DH parametreleridir. En sondaki sigma1 parametresi ise 0 değeri için uzvun döner, 1 değeri için uzvun kayar olduğunu belirtmektedir. Oluşturulan bu uzuv nesneleri robot komutu ile birleştirilerek robot tanımı tamamlanmış olur. Uzuv nesnesinin diğer kinematik ve dinamik parametreleri de istenildiği gibi tanımlanabilir.

2. *Hazır robot modellerini kullanma :*

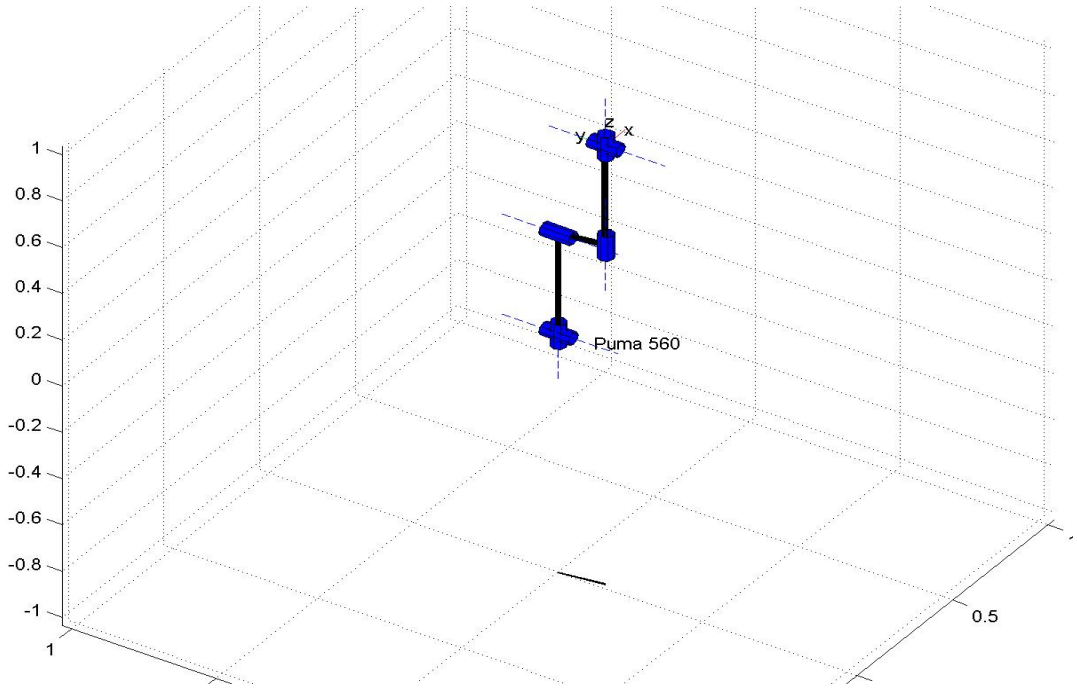
Robotics Toolbox'ta tanımlanmış üç tane hazır robot modeli vardır :

a) Basit iki uzuvlu(twolink)



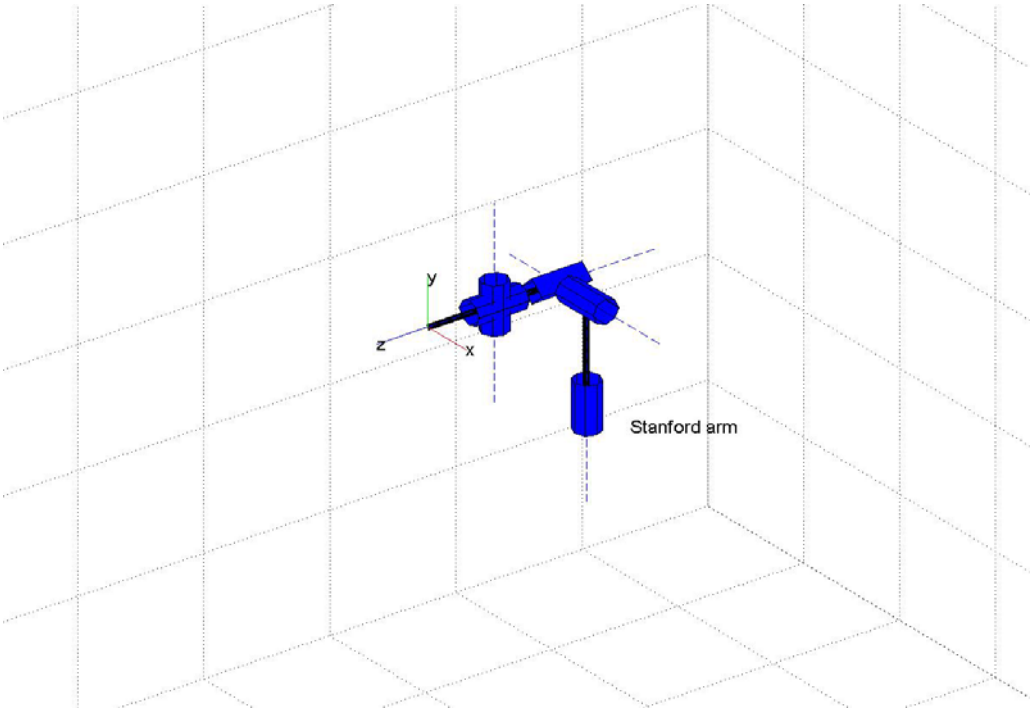
Şekil 2.5 Twolink MATLAB 3D modeli

b) Puma 560(p560)



Şekil 2.6 Puma 560 MATLAB 3D modeli

c) Stanford Arm(stanf)



Şekil 2.7 Stanford Arm MATLAB 3D Modeli

Bu modeller için verilen M dosyalarında tüm kinematik ve dinamik parametreler tanımlanmıştır. Robotics Toolbox'ta kullanılan komutların listesi Çizelge 2.3.'de verilmiştir.

Çizelge 2.3. Robotics Toolbox İçin Komutlar

Homojen Dönüşümler	
eul2tr	Euler açısından homojen dönüşüme geç
oa2tr	yönelim ve yaklaşım vektörü homojen dönüşüme geç
rot2tr	homojen dönüşümden 3 x 3 dönme altmatrisini al
rotx	X-ekseni etrafında dönme için homojen dönüşüme geç
roty	Y-ekseni etrafında dönme için homojen dönüşüme geç
rotz	Z-ekseni etrafında dönme için homojen dönüşüme geç
rpy2tr	roll/pitch/yaw açılarından homojen dönüşüme geç
tr2eul	homojen dönüşümden Euler açılarına geç
tr2rot	homojen dönüşümden dönme altmatrisine geç
tr2rpy	homojen dönüşümden roll/pitch/yaw açılarına geç
transl	homojen dönüşümden öteleme altmatrisini al
trnorm	homojen dönüşümün normalini al

Kuaternionlar	
/	kuaternionu bir kuaterniona veya skalere böl
*	kuternionu bir kuaternionla veya skalerle çarp
inv	kuaternionun tersini al
norm	kuaternionun normalini al
plot	3D dönme olarak bir kuaternionu göster
q2tr	kuaterniondan homojen matrise geç
qinterp	kuaternionları interpole et
unit	kuaternionu birimleştir

Yörünge Türetimi	
Ctraj	kartezyen yörünge türet
jtraj	eklem uzayı yörüngesi türet
trinterp	homojen dönüşümleri interpole et

Çizelge 2.3. (devam)

Kinematikler	
diff2tr	diferansiyel hareket vektöründen homojen dönüşüme geç
fkine	düz kinematikleri hesapla
ikine	ters kinematikleri hesapla
ikine560	Puma 560 için ters kinematikleri hesapla
ikinetl2	Twolink için ters kinematikleri hesapla
jacob0	temel koordinat çerçevesinde Jakobyeni hesapla
jacobn	sonlandırıcı koordinat çerçevesinde Jakobyeni hesapla
tr2diff	homojen dönüşümden diferansiyel hareket vektörüne geç
tr2jac	homojen dönüşümden Jakobyene geç

Dinamikler	
Accel	düz dinamikleri hesapla
cinertia	kartezyen manipülatör eylemsizlik matrisini hesapla
coriolis	coriolis/merkezkaç momentlerini hesapla
friction	sürtünme ekle
ftrans	kuvvet/moment dönüştür
gravload	yerçekimi yükünü hesapla
inertia	manipülatör eylemsizlik matrisini hesapla
itorque	eylemsizlik momentini hesapla
nofriction	robot nesnesinden sürtünmeyi çıkart
rne	ters dinamikleri hesapla

Robot Modelleri	
Link	bir uzuv nesnesi oluştur
puma560	Puma 560 verisi
puma560akb	Puma 560 verisi (Yenilenmiş DH)
robot	bir robot nesnesi oluştur.
Stanford	Stanford Arm verisi
Twolink	basit iki uzuvlu

Çizelge 2.3. (devam)

Grafikler	
drivebot	Puma 560 için grafiksel robot sürücüsü
drivebot2	Stanford Arm için grafiksel robot sürücüsü
drivebot3	Twolink için grafiksel robot sürücüsü

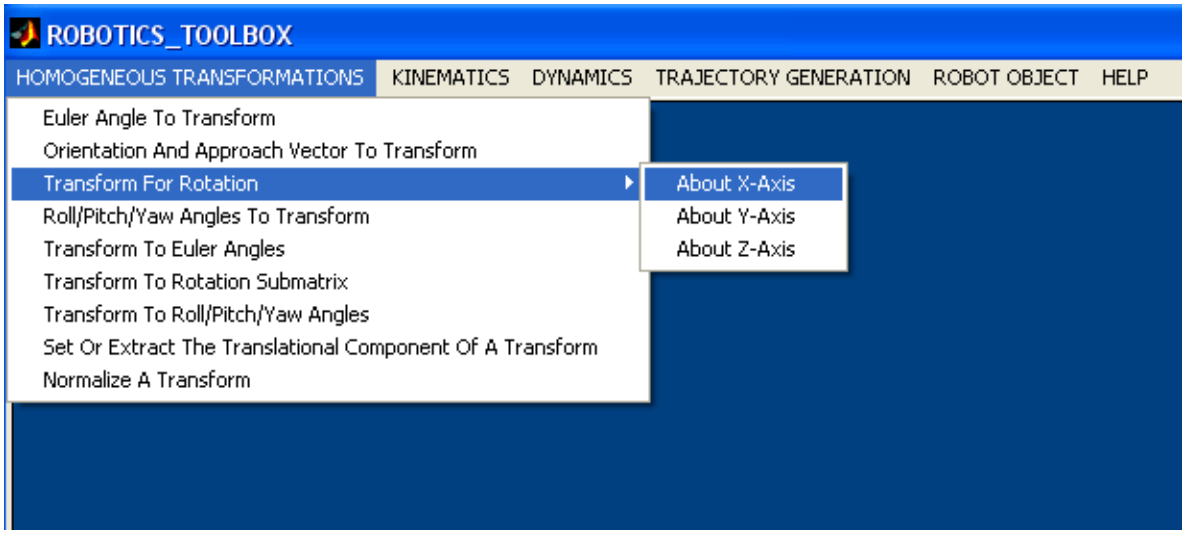
Diğer	
manipltyyoshi	Yoshikawa hareket edebilirlik ölçütünü hesapla
manipltyasada	Asada hareket edebilirlik ölçütünü hesapla
rtdemos	gösterileri başlat

3. MATERYAL VE METOD

Tez daha çok programsal yapı üzerine kurulduğundan temel olarak GUI(Grafiksel Kullanıcı Arayüzü)'den bahsedilecektir. Bu tez çalışmasında var olan Robotik Toolbox komutları tek tek alınarak her bir komut için m-file ve figür dosyaları oluşturulmuştur. Böylelikle hesaplamalar grafiksel bir alana taşınarak gerçekleştirilmiştir. Kullanıcıların Matlab Programlama Dilini bilmeksizin ekrandaki ilgili kutuları doldurarak hesaplamalarını kolaylıkla yapması sağlanmıştır.

3.1. GÖRSEL ROBOTİK ARAÇ KUTUSU TASARIMI

Geliştirilen görsel robotik araç kutusunun (GRAK) grafiksel kullanıcı ara yüzünün hazırlanmasında, bir ana ekran kullanılmıştır. GRAK ana ekranında Robotik Toolbox içerisinde kullanılan komutlar mönüler halinde listelenmiştir. Görsel robotik araç kutusunun geliştirilmesinde Matlab GUI baz alınmıştır. GUI içerisinde var olan tüm görsel öğeler kullanılarak, Robotikte kullanılan denklemleri çözüp sonuçlar üreten birer figür ve Matlab *.m file dosyaları hazırlanmış ve bunlar ana ekranda oluşturulan mönü aracılığı ile çağrılmıştır. Bunun ardından, robotikte kullanılan bazı olaylar için animasyonları hazırlanmış, hazırlanan animasyonlar programa eklenmiştir. Ayrıca son aşamada araç kutusunun nasıl kullanılacağını içeren bir yardım mönüsü hazırlanmış ve programa bir menü elemanı olarak eklenmiştir. Şekil 3.1.'de program çalıştırıldığında karşılaşılan ilk ekran görülmektedir.



Şekil 3.1. Programın Çalışır Ekran Görüntüsü

Programın en belirgin özelliği kullanıcıların işlemlerini yapmak için Matlab editöründen herhangi bir işlem yapmaması, komut ya da parametre girmemesidir. Her bir işlem için hazırlanan görsel ekran aracılığı ile işlemlerini kolaylıkla gerçekleştirebilir. Bu programda kullanıcıların Matlab Programlama Dilini de bilmesi gerekmemektedir. Çünkü kullanılacak olan *.m file dosyaları görsel mönüler ve görsel ekran öğeleri şeklinde, içlerindeki gereken parametre sorgularıyla beraber kullanıcıya sunulmaktadır. Kullanıcının sadece bu parametreleri uygun değerlerle doldurup, gerekli düğmelere basması yeterlidir.

Programda Robotik Toolbox içerisinde işlem gören tüm komutlar için gerekli figür ve *.m file dosyaları oluşturulmuştur. Aşağıdaki listede programda kullanılan mönü elemanları listelenmiştir. Ayrıca bu mönülerin işlevlerine göre alt elemanları da programda mevcuttur.

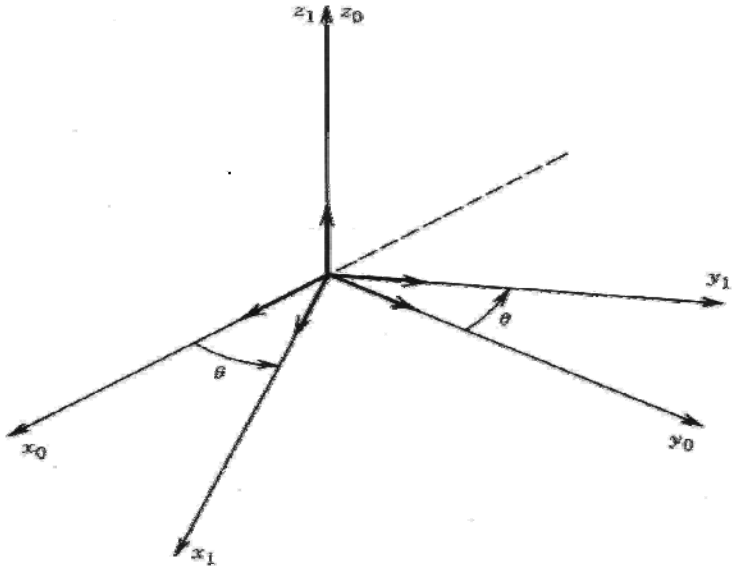
- Dönüşümler (*Transformations)*
- Yörünge Türetimi *(Trajectory generation)*
- Kinematikler *(Kinematics)*
- Dinamikler*(Dynamics)*

3.2. DÖNÜŞÜMLER (*TRANSFORMATIONS*)

Manipülatör eklemlerle birbirine bağlanmış rijid cisimlerin açık uçlu kinematik zinciri olarak kabul edilir. Zincirin bir ucu yere bağlı iken diğer ucu sonlandırıcıya bağlıdır. Sonuçta bu yapının hareketi her bir uzvun diğerine göre hareketlerinin toplamından oluşturulur. Bunun için önce bir rijid cismin uzaydaki konumunu ve yönelimini belirtmek için dönme(rotation) matrisi ve öteleme(translation) vektörü oluştururuz. Bu matris ve vektörü birleştirmek için daha sonra homojen dönüşüm gösterimini kullanırız.

3.2.1. Dönme Matrisi

Şekil 3.2.'de gösterildiği üzere $\{i_0,j_0,k_0\}$ $ox_0y_0z_0$ koordinat çerçevesi için , $\{i_1,j_1,k_1\}$ $ox_1y_1z_1$ koordinat çerçevesi için birim vektörler olsun. 1. koordinat çerçevesi 0. koordinat çerçevesinde z_0 ekseni etrafında θ açısı kadar döndürülerek elde edilmiştir. Bu iki koordinat çerçevesi arasındaki dönüşüm Eşitlik 1 ile bulunur. Buradaki R_0^1 matrisi 1. koordinat çerçevesinden 0. koordinat çerçevesine dönme matrisini göstermektedir. 1. koordinat çerçevesi 0. koordinat çerçevesinden belli bir dönme ile elde edildiğinden bu matris dönme matrisi adını alır.



Şekil 3.2. Z_0 Ekseni Etrafında Dönme

Burada

$$i_0.i_1 = \cos\theta$$
$$j_0.i_1 = -\sin\theta$$
$$j_0.j_1 = \cos\theta$$
$$i_1.j_0 = \sin\theta$$
$$k_0.k_1 = 1$$

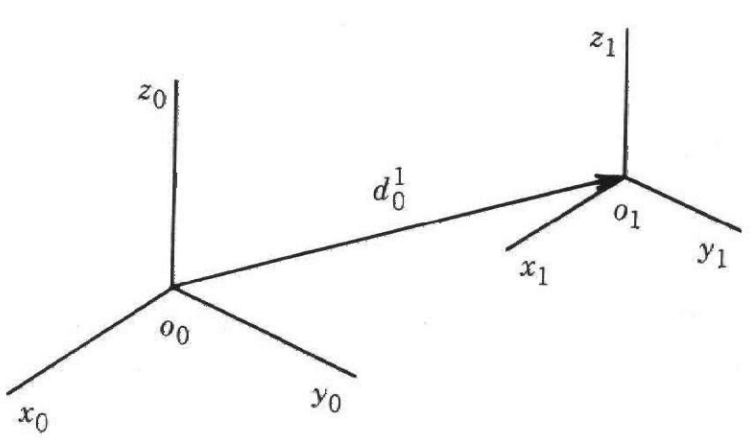
ve diğer tüm nokta çarpımlar sıfır olmak üzere,

$$R_0^1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

dönme matrisi elde edilir.

3.2.2. Öteleme Vektörü

Şekil 3.3.'de gösterildiği üzere oxyz'den $|d|$ kadar bir öteleme ile elde edilmiş oxyz koordinat çerçevesi düşünelim. Bu iki koordinat çerçevesi arasındaki dönüşüm öteleme vektörü ile tanımlanır.



Şekil 3.3. Ötelenmiş Çerçeve

$$d_0^1 = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \qquad R_0^1 = \begin{bmatrix} i_0.i_1 & j_0.i_1 & k_0.i_1 \\ i_0.j_1 & j_0.j_1 & k_0.j_1 \\ i_0.k_1 & j_0.k_1 & k_0.k_1 \end{bmatrix}$$

3.2.3. Homojen Dönüşüm

Üç boyutlu uzayda koordinat çerçeveleri arasındaki dönüşüm, dönme matrisleri ve öteleme vektörleri yardımıyla yapılır. Her ikisinin birlikte gösterilimi için yani koordinat çerçeveleri arasında hem dönmenin, hem de ötelemenin var olduğu durumda homojen dönüşüm matrisleri kullanılır. Homojen dönüşüm matrisleri aşağıdaki gibi elde edilir.

$$T_0^1 = \left[\begin{array}{ccc|c} & R_0^1 & & d_0^1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4x4}$$

Burada elde edilen matrisi 1. koordinat çerçevesinden 0. koordinat çerçevesine homojen dönüşüm matrisini göstermektedir. T_0^1 matrisini 4x4 boyutundan kare matris olduğuna dikkat edilmelidir. Bu homojen matris oluşturulurken matrisin tersinin alınabilmesi için yapılmıştır. matrisindeki $T_{4,4}$ elemanı olan 1 tüm elemanların bire bir ölçeklendiğini göstermektedir.

Bu elde ettiğimiz genel kalıba bağlı olarak temel homojen dönüşüm matrisleri şu şekildedir:

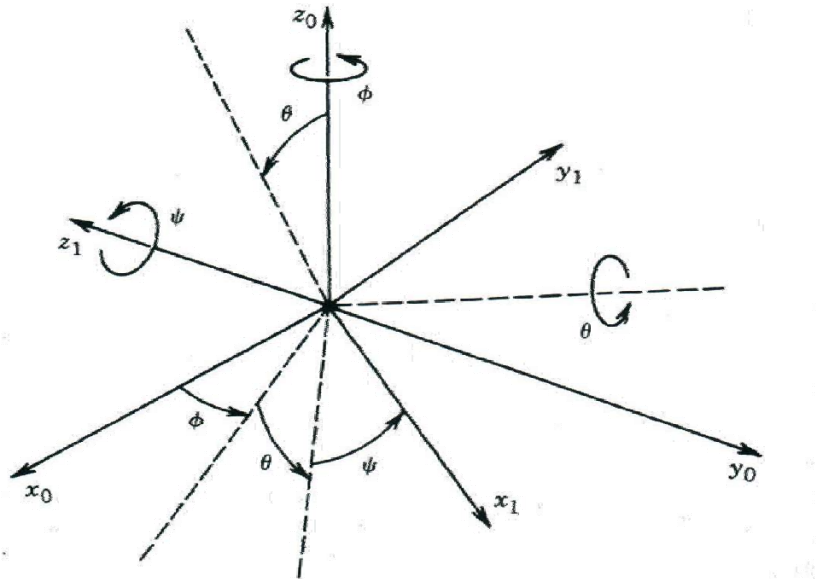
$$Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & -s\alpha & 0 \\ 0 & s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Rot_{y,\phi} = \begin{bmatrix} c\phi & 0 & s\phi & 0 \\ 0 & 1 & 0 & 0 \\ -s\phi & 0 & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot_{z,\theta} = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$d_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad d_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad d_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.4. EUL2TR (Euler Açı Dönüşümü)

Euler açıları, sırasıyla z ekseni etrafında Φ açısı kadar, y ekseni etrafında θ açısı kadar, tekrar z ekseni etrafında Ψ açısı kadar dönmelere karşılık gelmektedir. Euler açıları gösterilimi Şekil 3.4. verilmiştir.



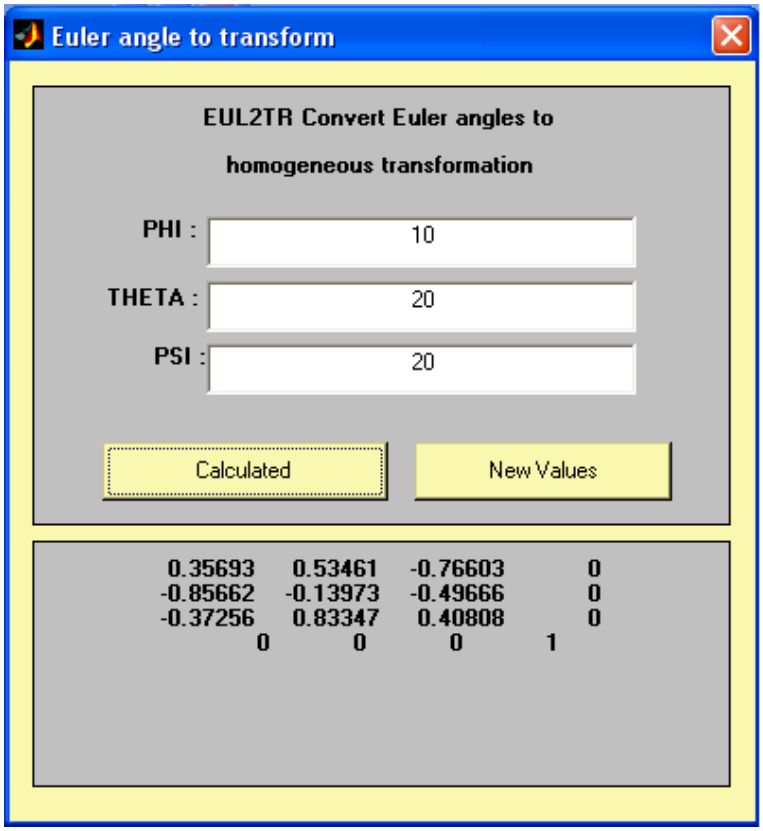
Şekil 3.4. Euler Açıları Gösterilimi

EUL2TR, Eular açıları homojen dönüşüme(transformation) çevirir.

TR = EUL2TR([PHI THETA PSI])

TR = EUL2TR(PHI, THETA, PSI)

Bu fonksiyon, belirli Eular açılar için bir homojen dönüşüm matrisi döndürür. Bu sırasıyla Z, Y, Z eksenleri etrafında döndürmenin karşılığıdır



Şekil 3.5. EUL2TR İçin Hazırlanan GUI Görüntüsü

```
function Calculated_Callback(hObject, eventdata, handles)
try
    str1=findobj(gcbf,'Tag','edit1');
    str2=findobj(gcbf,'Tag','edit2');
    str3=findobj(gcbf,'Tag','edit3');
    str4=findobj(gcbf,'Tag','sonuc');
    x=str2num(get(str1,'String'));
    y=str2num(get(str2,'String'));
    z=str2num(get(str3,'String'));
    set(str4,'String','');
    s=num2str(EUL2TR([x,y,z]));
    set(str4,'String',s);
catch
    errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end
```

```
% --- Executes on button press in NewValues.
function NewValues_Callback(hObject, eventdata, handles)
   str1=findobj(gcbf,'Tag','edit1');
   str2=findobj(gcbf,'Tag','edit2');
   str3=findobj(gcbf,'Tag','edit3');
   str4=findobj(gcbf,'Tag','sonuc');
   set(str1,'String','');
   set(str2,'String','');
   set(str3,'String','');
   set(str4,'String','');
```

3.2.5. OA2TR (Yönlendirme ve Yaklaşım Vektör Dönüşümü)

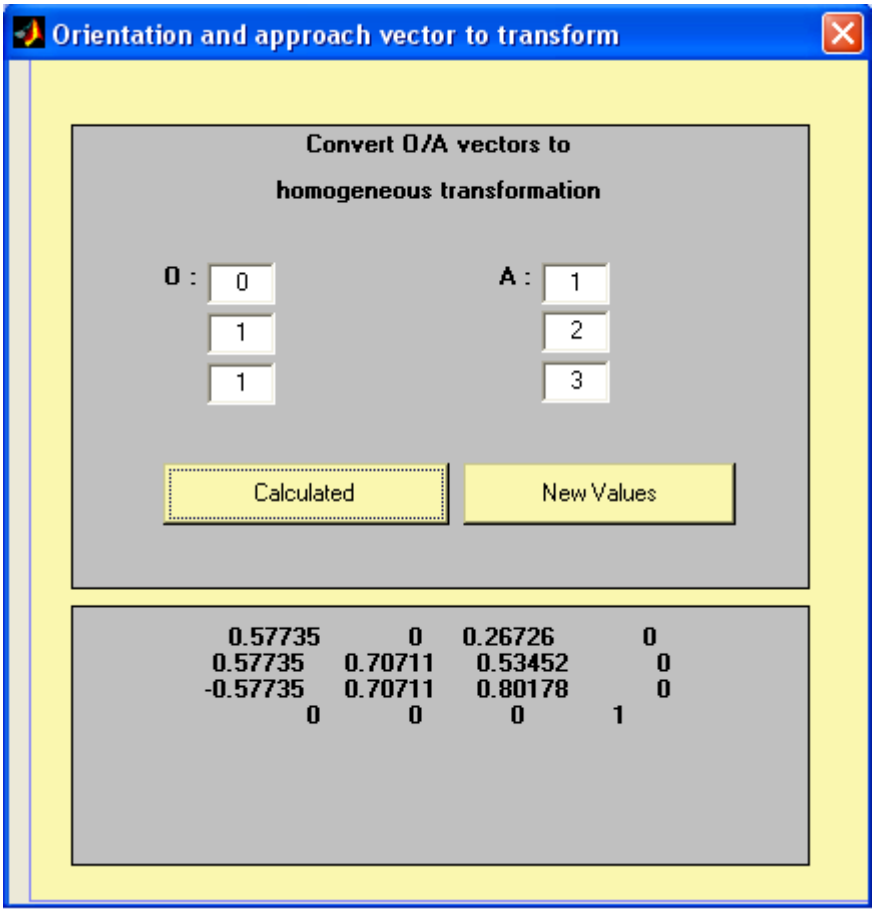
OA2TR, O/A vektörlerini homojen dönüşüm matrisine çevirir.

TR = OA2TR(O, A)

Belirtilen yönlendirme(orientation) ve yaklaşım(approach) vektörlerini, homojen dönüşüm matrisine çevirir. Dönüşüm altmatrisi 3 vektörden oluşmaktadır. Bunlar şöyledir;

R = [N O A] and N = O x A.

Bu altmatris, O ve A vektörleri paralel olmadığı sürece orthonormal olmayı garanti eder.



Şekil 3.6 OA2TR İçin Hazırlanan GUI Görüntüsü

```
% --- Executes on button press in Calculated.
function Calculated_Callback(hObject, eventdata, handles)
try
    a1=findobj(gcbf,'Tag','A1');
    a2=findobj(gcbf,'Tag','A2');
    a3=findobj(gcbf,'Tag','A3');
    n1=findobj(gcbf,'Tag','N1');
    n2=findobj(gcbf,'Tag','N2');
    n3=findobj(gcbf,'Tag','N3');
    alinan_a1=str2num(get(a1,'String'));
    alinan_a2=str2num(get(a2,'String'));
    alinan_a3=str2num(get(a3,'String'));
    alinan_n1=str2num(get(n1,'String'));
    alinan_n2=str2num(get(n2,'String'));
```

```
    alinan_n3=str2num(get(n3,'String'));
    A=[alinan_a1;alinan_a2;alinan_a3]
    N=[alinan_n1;alinan_n2;alinan_n3]
    str4=findobj(gcbf,'Tag','sonuc');
    set(str4,'String','');
    s=OA2TR(N,A);
    s=num2str(s)
    set(str4,'String',s);
catch
    errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end

% --- Executes on button press in NewValues.
function NewValues_Callback(hObject, eventdata, handles)
a1=findobj(gcbf,'Tag','A1');
a2=findobj(gcbf,'Tag','A2');
a3=findobj(gcbf,'Tag','A3');
n1=findobj(gcbf,'Tag','N1');
n2=findobj(gcbf,'Tag','N2');
n3=findobj(gcbf,'Tag','N3');
str4=findobj(gcbf,'Tag','sonuc');
set(a1,'String','');
set(a2,'String','');
set(a3,'String','');
set(n1,'String','');
set(n2,'String','');
set(n3,'String','');
set(str4,'String','');
```

3.2.6. ROTX, ROTY, ROTZ

ROTX, X ekseni etrafında dönüşüm

TR = ROTX(theta)

X ekseni etrafında, theta açısı kadar döndürmeyi sağlayan bir homojen dönüşüm matrisi verir.

ROTY, Y ekseni etrafında dönüşüm

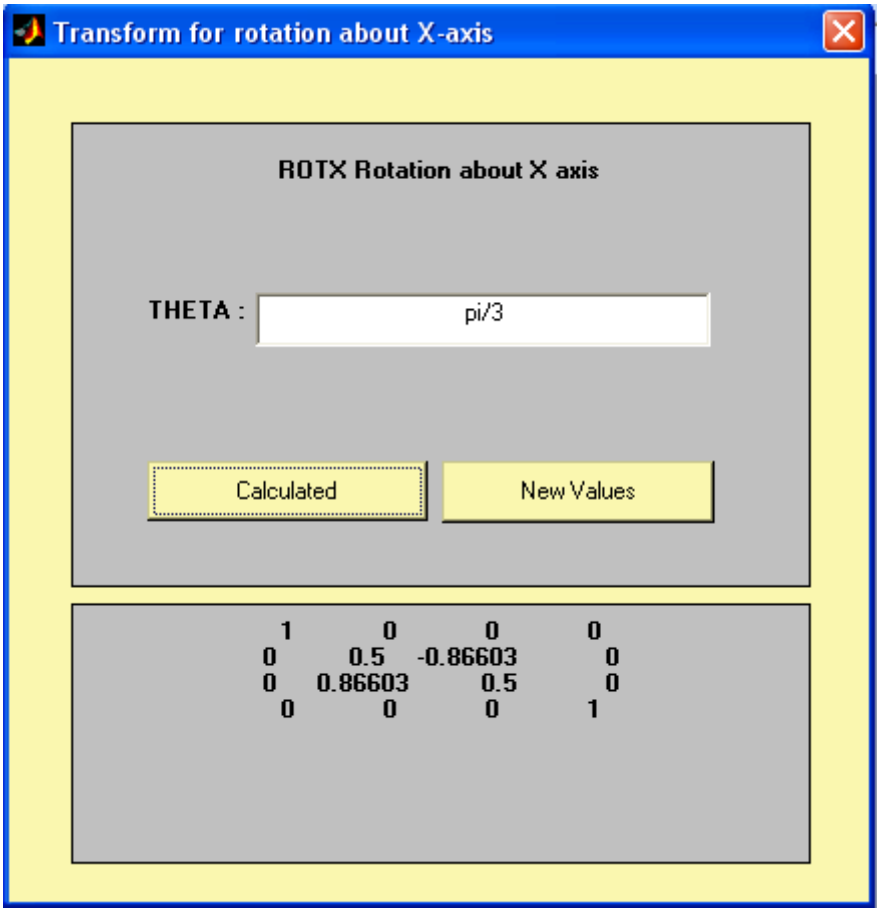
TR = ROTY(theta)

Y ekseni etrafında, theta açısı kadar döndürmeyi sağlayan bir homojen dönüşüm matrisi verir.

ROTZ, Z ekseni etrafında dönüşüm

TR = ROTZ(theta)

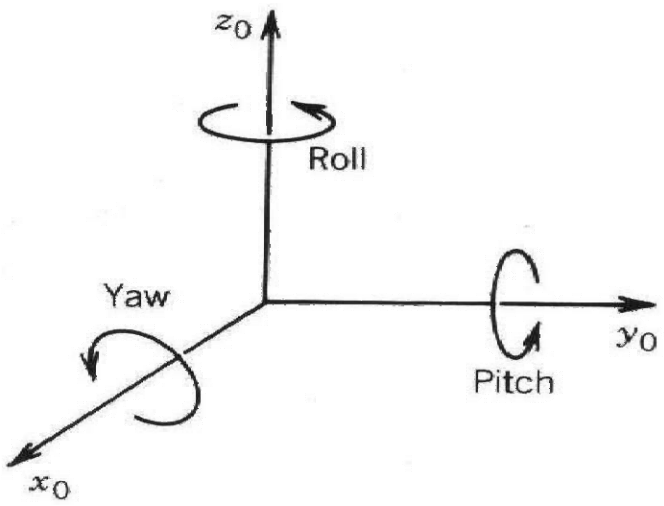
Z ekseni etrafında, theta açısı kadar döndürmeyi sağlayan bir homojen dönüşüm matrisi verir.



Şekil 3.7 ROTX İçin Hazırlanan GUI Görüntüsü

3.2.7. RPY2TR (Roll/Pitch/Yaw Açı Dönüşümleri)

Roll / Pitch / Yaw açıları sırasıyla z ekseni etrafında Φ açısı kadar , y ekseni etrafında θ açısı kadar ve x ekseni etrafında Ψ açısı kadar dönmelere karşılık gelmektedir. Roll / Pitch / Yaw açıları gösterilimi Şekil 3.8.'de verilmiştir.



Şekil 3.8. Roll / Pitch /Yaw Açıları Gösterilimi

Burada dönme matrisi şu şekilde elde edilir:

$$R_0^1 = R_{z,\phi} R_{y,\theta} R_{x,\psi}$$

$$= \begin{bmatrix} c\phi & -s\phi & 0 \\ s\phi & c\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\psi & -s\psi \\ 0 & s\psi & c\psi \end{bmatrix}$$

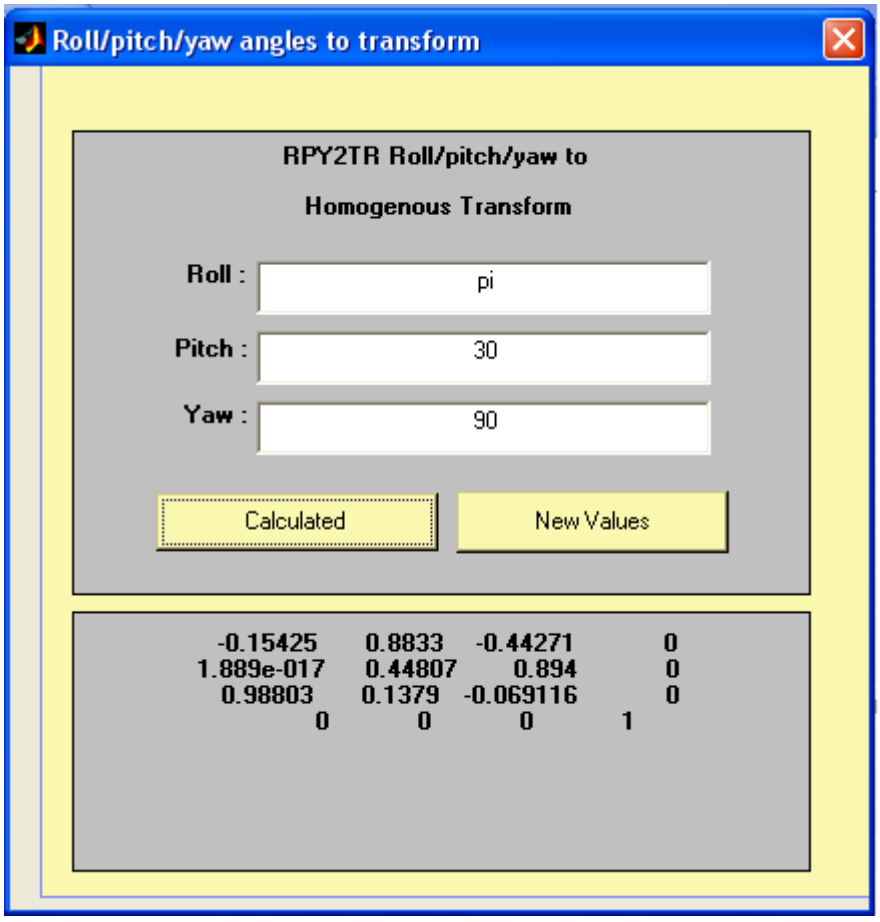
$$= \begin{bmatrix} c\phi c\theta & -s\phi c\psi + c\phi s\theta s\psi & s\phi s\psi + c\phi s\theta c\psi \\ s\phi c\theta & c\phi c\psi + s\phi s\theta s\psi & -c\phi s\psi + s\phi s\theta c\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix}$$

RPY2TR, Roll/Pitch/Yaw Açıları Homojen Dönüşümü

TR = RPY2TR([R P Y])

TR = RPY2TR(R,P,Y)

Belirtilen Roll/Pitch/Yaw açıları için bir homojen dönüşüm matrisi verir. Bu, sırasıyla Z,Y, X eksenleri etrafında döndürmenin karşılığıdır.



Şekil 3.9. RPY2TR İçin Hazırlanan GUI Görüntüsü

```
function Calculated_Callback(hObject, eventdata, handles)
try
   str1=findobj(gcbf,'Tag','edit1');
   str2=findobj(gcbf,'Tag','edit2');
   str3=findobj(gcbf,'Tag','edit3');
   str4=findobj(gcbf,'Tag','sonuc');
```

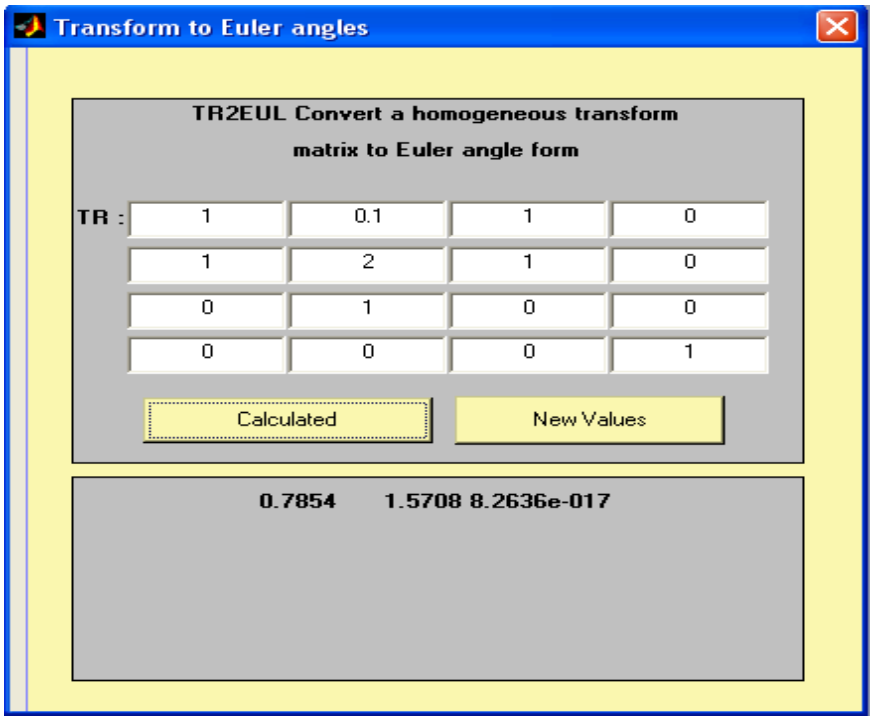
```
    x=str2num(get(str1,'String'));
    y=str2num(get(str2,'String'));
    z=str2num(get(str3,'String'));
    set(str4,'String','');
    s=num2str(RPY2TR([x,y,z]));
    set(str4,'String',s);
catch
    errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end
```

3.2.8. TR2EUL (Euler Açılara Dönüşüm)

TR2EUL, Bir homojen dönüşüm matrisini Eular açıya çevirir.

[PHI THETA PSI] = TR2EUL(TR)

Bu fonksiyon, parametre olarak, TR homojen dönüşüm matrisini alıp, bu matrisin döndürme parçasına uygun bir Eular açının vektörünü verir. Bu 3 açı, sırasıyla Z, Y ve Z eksenleri etrafında döndürmenin karşılığıdır.



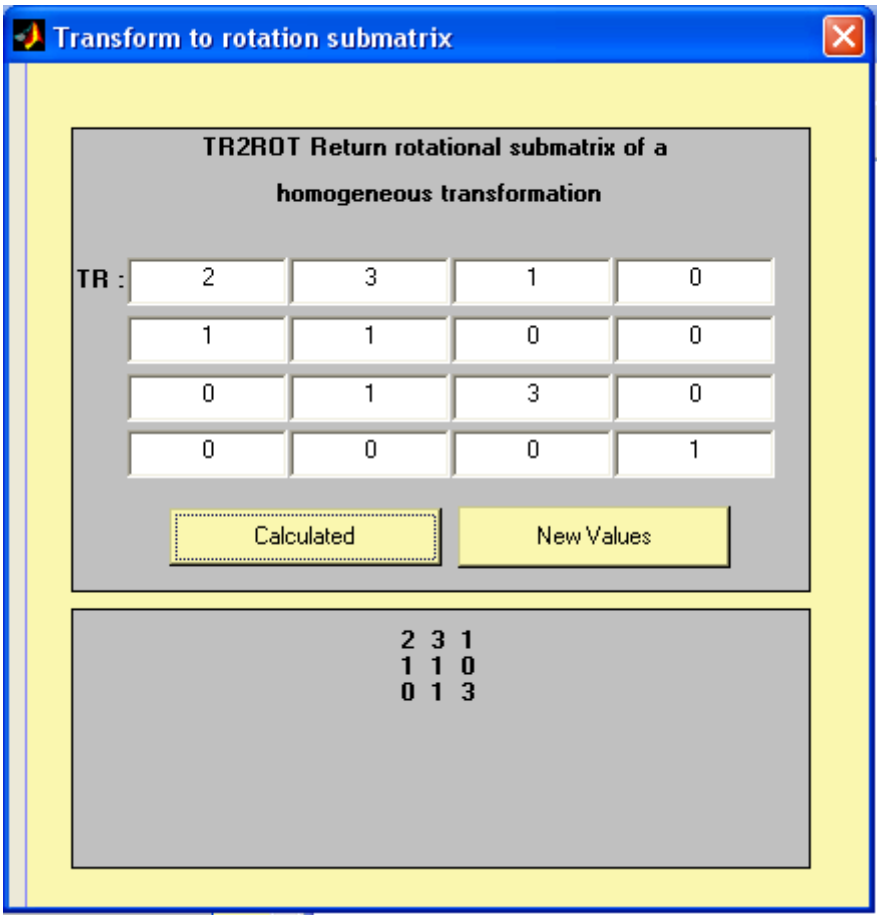
Şekil 3.10 TR2EUL İçin Hazırlanan GUI Görüntüsü

3.2.9. TR2ROT (Rotasyon Altmatrisine Dönüşüm)

TR2ROT, Bir homojen dönüşüm matrisinin, rotasyon altmatrisini verir.

R = TR2ROT(T)

Geriye T(homojen dönüşüm)matrisinden, 3X3'lük R orthonormal rotasyon matrisini verir.



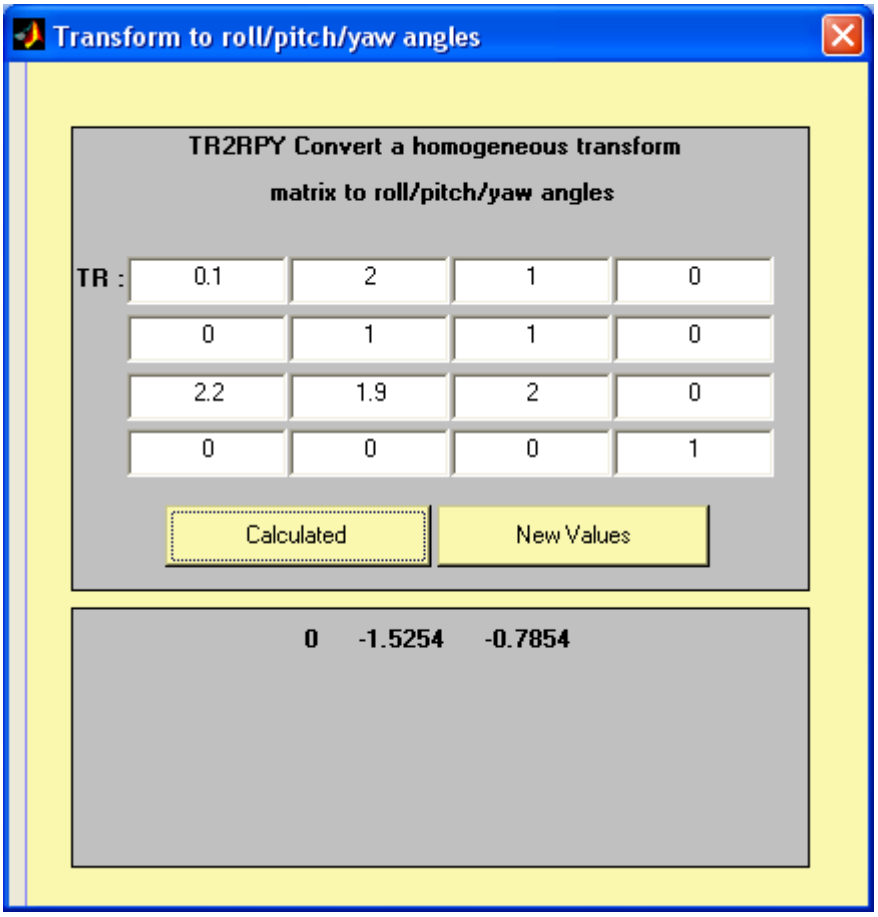
Şekil 3.11 TR2ROT İçin Hazırlanan GUI Görüntüsü

3.2.10. TR2RPY(Roll/Pitch/Yaw Açı Dönüşümleri)

TR2RPY, homojen dönüşüm matrisini, Roll/Pitch/Yaw açılarına çevirir.

[R P Y] = TR2RPY(TR)

Bu fonksiyon, parametre olarak, TR homojen dönüşüm matrisini alıp, bu matrisin döndürme parçasına uygun Roll/Pitch/Yaw açılarının vektörünü verir.



Şekil 3.12 TR2RPY İçin Hazırlanan GUI Görüntüsü

```
function Calculated_Callback(hObject, eventdata, handles)
try
    alinan_n1=str2num(get(findobj(gcbf,'Tag','N1'),'String'));
    alinan_n2=str2num(get(findobj(gcbf,'Tag','N2'),'String'));
    alinan_n3=str2num(get(findobj(gcbf,'Tag','N3'),'String'));
    alinan_n4=str2num(get(findobj(gcbf,'Tag','N4'),'String'));
    alinan_n5=str2num(get(findobj(gcbf,'Tag','N5'),'String'));
    alinan_n6=str2num(get(findobj(gcbf,'Tag','N6'),'String'));
    alinan_n7=str2num(get(findobj(gcbf,'Tag','N7'),'String'));
    alinan_n8=str2num(get(findobj(gcbf,'Tag','N8'),'String'));
    alinan_n9=str2num(get(findobj(gcbf,'Tag','N9'),'String'));
    alinan_n10=str2num(get(findobj(gcbf,'Tag','N10'),'String'));
    alinan_n11=str2num(get(findobj(gcbf,'Tag','N11'),'String'));
    alinan_n12=str2num(get(findobj(gcbf,'Tag','N12'),'String'));
```

```
    alinan_n13=str2num(get(findobj(gcbf,'Tag','N13'),'String'));
    alinan_n14=str2num(get(findobj(gcbf,'Tag','N14'),'String'));
    alinan_n15=str2num(get(findobj(gcbf,'Tag','N15'),'String'));
    alinan_n16=str2num(get(findobj(gcbf,'Tag','N16'),'String'));

    TR=[alinan_n1, alinan_n5, alinan_n9, alinan_n13 ;
    alinan_n2, alinan_n6, alinan_n10, alinan_n14 ;
    alinan_n3, alinan_n7, alinan_n11, alinan_n15;
    alinan_n4, alinan_n8, alinan_n12, alinan_n16];
    str4=findobj(gcbf,'Tag','sonuc');
    set(str4,'String','');
    s=TR2RPY(TR)
    s=num2str(s)
    set(str4,'String',s);
  catch
    errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
  end
```

3.2.11. TRANSL(Dönüşümsel Matris)

Verilen bir doğrusal bileşeni kullanarak dönüşüm matrisini oluşturur veya dönüşüm matrisinden doğrusal bileşeni çeker.

TR = TRANSL(X, Y, Z)

TR = TRANSL([X Y Z])

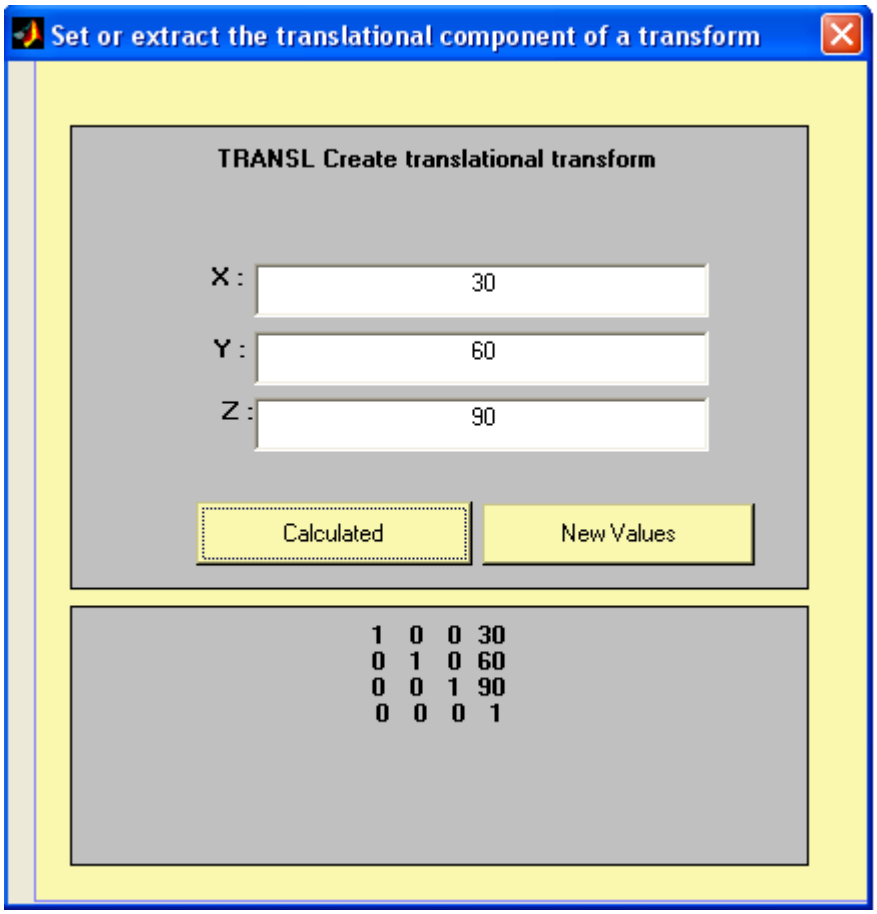
Bu fonksiyon, X,Y ve Z 'nin dönüşümünü temsil eden bir homojen dönüşüm matrisi döndürür.

[X Y Z]' = TRANSL(T)

Bu fonksiyon, homojen dönüşüm matrisinin(T) dönüşümsel bölümünü 3 elemanlı sütun vektörü olarak döndürür.

[X Y Z] = TRANSL(TG)

Bu fonksiyon, Kartezyen yörünge matrisi TG'den X,Y,Z elemanlarının bir matrisini çıkartır.



Şekil 3.13 TRANSL İçin Hazırlanan GUI Görüntüsü

```
function Calculated_Callback(hObject, eventdata, handles)
try
    str1=findobj(gcbf,'Tag','edit1');
    str2=findobj(gcbf,'Tag','edit2');
    str3=findobj(gcbf,'Tag','edit3');
    str4=findobj(gcbf,'Tag','sonuc');
    x=str2num(get(str1,'String'));
    y=str2num(get(str2,'String'));
    z=str2num(get(str3,'String'));
    set(str4,'String','');
    s=TRANSL([x,y,z]);
    s=num2str(s);
    set(str4,'String',s);
catch
```

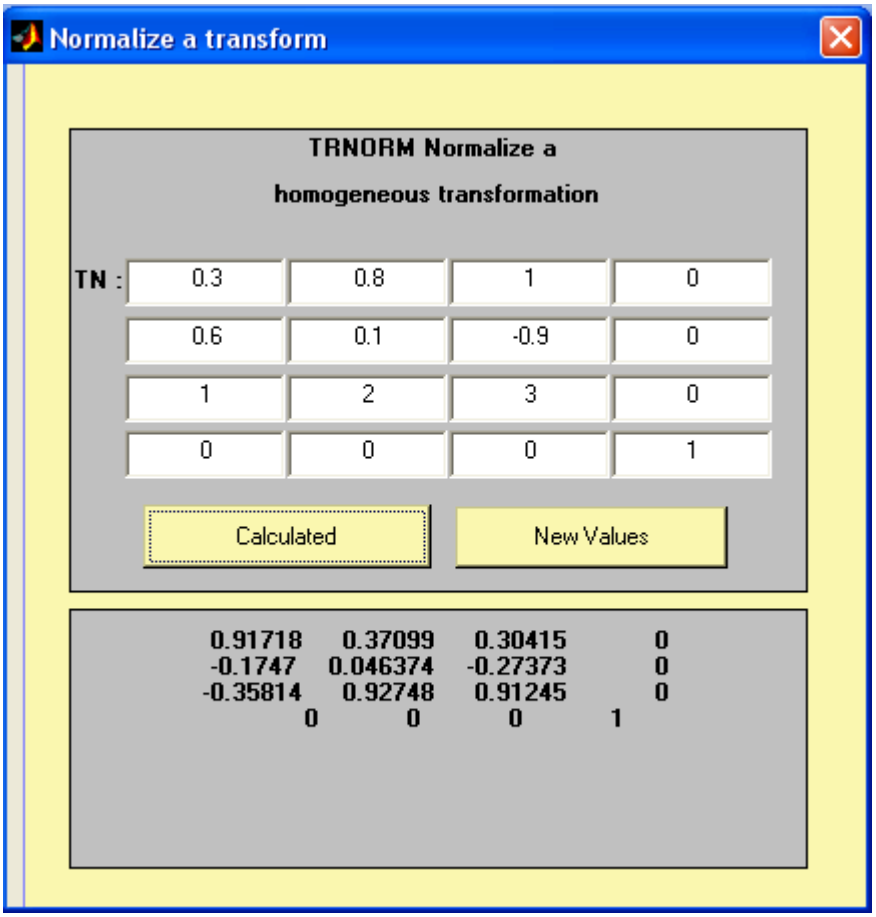
```
    errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end
```

3.2.12. TRNORM(Dönüşüm Matrisi Normalizasyonu)

TRNORM, Homojen dönüşüm matrisini normalize eder.

TN = TRNORM(T)

Bu fonksiyon normalize edilmiş homojen dönüşüm matrisini verir. Bu rotasyon matrisi uygun bir orthogonal matristir. O ve V vektörleri normalize edilmiştir ve normal vektör O x A'dan oluşmaktadır.



Şekil 3.14 TRNORM İçin Hazırlanan GUI Görüntüsü

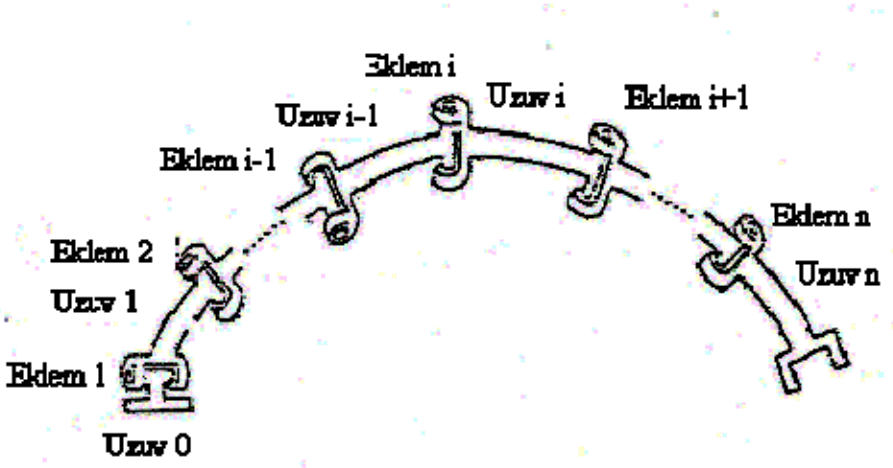
3.3. KİNEMATİKLER (KINEMATICS)

3.3.1. Düz Kinematikler

Robotikte düz kinematikler manipülatörün verilen eklem değişken değerleri için sonlandırıcının konumunu ve yönelimini bulmak olarak tanımlanabilir. Eklem değişkenleri, eklemin döner olması durumunda uzuvlar arasındaki açı, eklemin kayar olması durumunda uzuv uzanma miktarıdır. Bu değişkenler için gösterilimler aşağıda verilmiştir:

$$q_i = \begin{Bmatrix} \theta_i & döner\ eklem\ için \\ d_i & kayar\ eklem\ için \end{Bmatrix}$$

Manipülatörlerin eklemlerle birleştirilmiş bir dizi uzuv olduğundan yola çıkarak Şekil 3.15.'de gösterildiği üzere n+1 uzva sahip bir manipülatörümüz olduğunu ve her bir uzva bir koordinat çerçevesi atandığını düşünelim.



Şekil 3.15. n+1 Uzva Sahip Kinematik Zincir Oluşturan Manipülatör

Şimdi $T_{i\text{-}1}^{\ i}$'in çerçeve i'den çerçeve i-1'e homojen dönüşüm matrisi olduğunu düşünelim. Burada $T_{i\text{-}1}^{\ i}$ matrisinin sabit olmadığına, manipülatörün hareketiyle değiştiğine dikkat edilmelidir.

3.3.2. Ters Kinematik

Düz kinematiklerin tersi biçimde robotikte ters kinematikler verilen sonlandırıcı konum ve yönelimi için gerekli eklem değişken değerlerini bulmak olarak tanımlanabilir. Ters kinematik problemlerinde düz kinematiklerin tersi biçimde homojen dönüşüm matrisleriyle oluşturulan doğrusal olmayan denklemlerin çözümlenmesi istenir.

3.3.3. Hız Kinematiği

Matematiksel olarak düz kinematikler, kartezyen konum ve yönelim uzayı ile eklem konumları arasında bir işlev tanımlar. Hız kinematiklerini (sonlandırıcının doğrusal ve açısal hızları) bu işlevin Jakobyen'ini belirleyerek elde edebiliriz.

Jakobyen matris değerli bir işlevdir ve skaler bir işlevin türevinin vektörel hali olarak düşünülebilir. Bu Jakobyen veya Jakobyen matrisi robotikte aşağıdaki alanlarda büyük önem taşır:

1. Düzgün yörünge(smooth trajectory) türetilmesi
2. Tekil(Singüler) konfigürasyonların bulunması
3. Hareket denklemlerinin türetilmesi
4. Sonlandırıcı kuvvet ve momentlerinin diğer manipülatör eklemlerine taşınması

n eklemli bir manipülatör için Jakobyen, eklem hızlarının *n*-vektörü ile sonlandırıcının 6 vektörden oluşan doğrusal ve açısal hızları arasındaki ilişkiyi verir. Buna göre *n* eklemli bir manipülatör için Jakobyen 6 x *n* boyutunda bir matristir.

3.3.4. Kullanıcı Tanımlı Robot

Matlab'da Robotic Toolbox içerisinde hazır robotlar mevcuttur. Ancak Bu robotlar dışında kullanıcılarda kendi verdikleri eklem parametrelerine göre robotlar oluşturabilirler. Matlab'da kullanıcıların robot tanımlaması için birçok robot

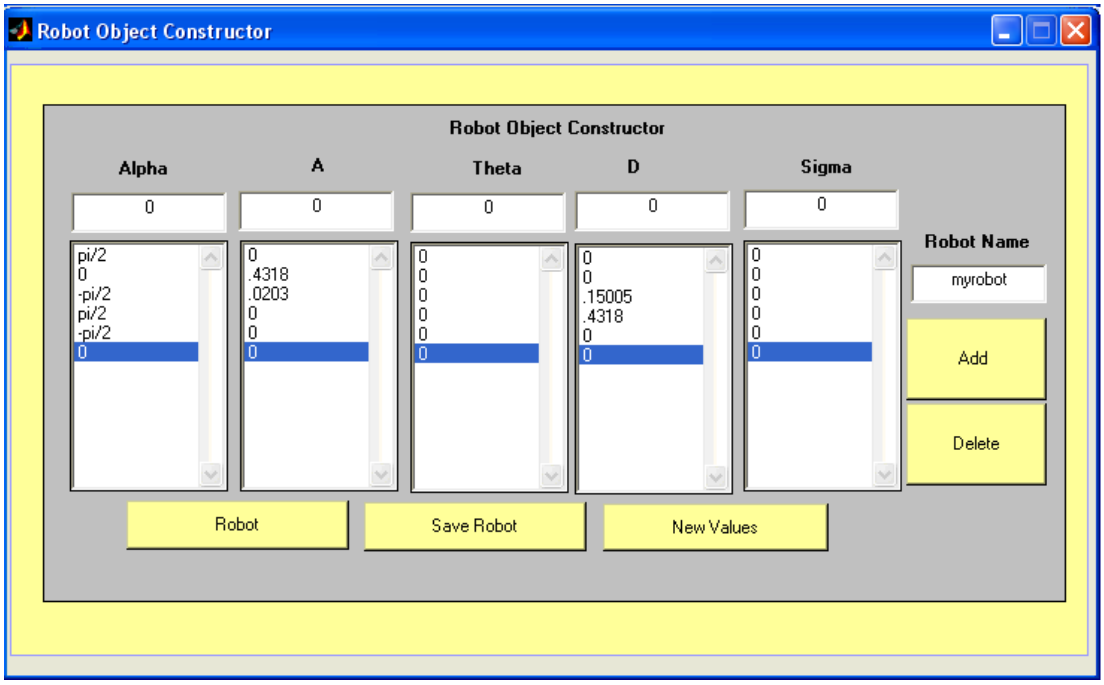
komutunu bilmeleri gerekli, bunun yerine hazırladığımız ekran görüntüsünde ilgi metin kutularını doldurup kaydetmeleri yeterlidir. Ayrıca bu hazırladıkları robotları istedikleri Robotic Toolbox komutlarında, sadece ismini vererek kullanabilirler.

Çizelge 3.1.'de her bir eklem için kullanılan parametreler ve neye karşılık geldikleri belirtilmiştir.

Çizelge 3.1. Bir Eklem İçin Alınan Parametreler

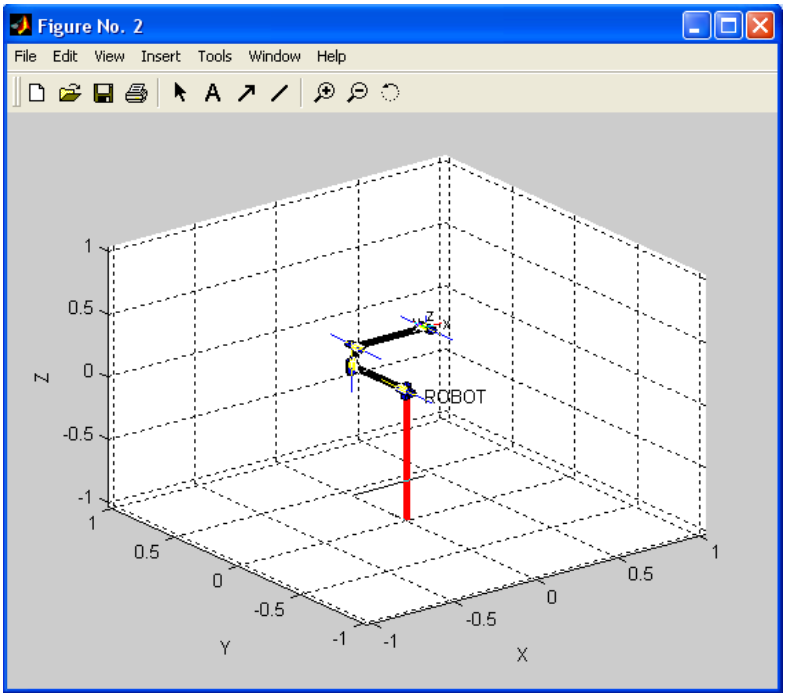
Robot Parametresi	Sembol	Tanımı
alpha	α	Eklem burulma(twist) açısı
A	A	Eklem uzunluğu
theta	θ	Eklem dönme açısı
D	D	İki eklem arası uzaklık
sigma	σ	Mafsal Tipi (Dönel İçin 0, prizmatik için 0'dan farklı)

Aşağıda kullanıcı tanımlı robot için guide ile hazırlanmış ekran görüntüsü bulunmaktadır. Yukarıdaki çizelgede gösterilen değerlere hazırlanmış ve robot kaydedilmiştir.



Şekil 3.16. Kullanıcı Tanımlı Robot İçin GUI Görüntüsü

Yukarıdaki örnekte 6 eklemli bir robot oluşturulmuştur. Myrobot ismiyle kaydedilmiştir. Bu robotun simülasyonu da Şekil 3.17.' de görülmektedir.



Şekil 3.17 Robot Simülasyonu

```
function Add_Callback(hObject, eventdata, handles)
try
    str1=findobj(gcbf,'Tag','edit1');
    str2=findobj(gcbf,'Tag','edit2');
    str3=findobj(gcbf,'Tag','edit3');
    str4=findobj(gcbf,'Tag','edit4');
    str5=findobj(gcbf,'Tag','edit5');

    l1=findobj(gcbf,'Tag','listbox1');
    l2=findobj(gcbf,'Tag','listbox2');
    l3=findobj(gcbf,'Tag','listbox3');
    l4=findobj(gcbf,'Tag','listbox4');
    l5=findobj(gcbf,'Tag','listbox5');

    prev_str = get(l1, 'String');
    prev_str{end + 1} = get(str1,'String');
    set(l1, 'String', prev_str, 'Value', length(prev_str));

    prev_str2 = get(l2, 'String');
    prev_str2{end + 1} = get(str2,'String');
    set(l2, 'String', prev_str2, 'Value', length(prev_str2));

    prev_str3 = get(l3, 'String');
    prev_str3{end + 1} = get(str3,'String');
    set(l3, 'String', prev_str3, 'Value', length(prev_str3));

    prev_str4 = get(l4, 'String');
    prev_str4{end + 1} = get(str4,'String');
    set(l4, 'String', prev_str4, 'Value', length(prev_str4));

    prev_str5 = get(l5, 'String');
```

```
   prev_str5{end + 1} = get(str5,'String');
   set(l5, 'String', prev_str5, 'Value', length(prev_str5));
catch
      errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end

function Delete_Callback(hObject, eventdata, handles)
try
   l1=findobj(gcbf,'Tag','listbox1');
   l2=findobj(gcbf,'Tag','listbox2');
   l3=findobj(gcbf,'Tag','listbox3');
   l4=findobj(gcbf,'Tag','listbox4');
   l5=findobj(gcbf,'Tag','listbox5');

   selected = get(l1,'Value');
   prev_str = get(l1, 'String');
   len = length(prev_str);
   if len > 0
      index = 1:len;
      prev_str = prev_str(find(index ~= selected),1);
      set(l1, 'String', prev_str, 'Value', min(selected, length(prev_str)));
   end

   selected = get(l2,'Value');
   prev_str = get(l2, 'String');
   len = length(prev_str);
   if len > 0
     index = 1:len;
     prev_str = prev_str(find(index ~= selected),1);
     set(l2, 'String', prev_str, 'Value', min(selected, length(prev_str)));
   end
```

```
    selected = get(l3,'Value');
    prev_str = get(l3, 'String');
    len = length(prev_str);
    if len > 0
      index = 1:len;
      prev_str = prev_str(find(index ~= selected),1);
      set(l3, 'String', prev_str, 'Value', min(selected, length(prev_str)));
    end

    selected = get(l4,'Value');
    prev_str = get(l4, 'String');
    len = length(prev_str);
    if len > 0
      index = 1:len;
      prev_str = prev_str(find(index ~= selected),1);
      set(l4, 'String', prev_str, 'Value', min(selected, length(prev_str)));
    end

    selected = get(l5,'Value');
    prev_str = get(l5, 'String');
    len = length(prev_str);
    if len > 0
      index = 1:len;
      prev_str = prev_str(find(index ~= selected),1);
      set(l5, 'String', prev_str, 'Value', min(selected, length(prev_str)));
    end
catch
       errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end

function listbox1_Callback(hObject, eventdata, handles)
l1=findobj(gcbf,'Tag','listbox1');
```

```
l2=findobj(gcbf,'Tag','listbox2');
l3=findobj(gcbf,'Tag','listbox3');
l4=findobj(gcbf,'Tag','listbox4');
l5=findobj(gcbf,'Tag','listbox5');

a = get(l1,'Value');
set(l2,'Value',a) ;
set(l3,'Value',a) ;
set(l4,'Value',a) ;
set(l5,'Value',a) ;

function listbox2_Callback(hObject, eventdata, handles)
l1=findobj(gcbf,'Tag','listbox1');
l2=findobj(gcbf,'Tag','listbox2');
l3=findobj(gcbf,'Tag','listbox3');
l4=findobj(gcbf,'Tag','listbox4');
l5=findobj(gcbf,'Tag','listbox5');

a = get(l2,'Value');
set(l1,'Value',a) ;
set(l3,'Value',a) ;
set(l4,'Value',a) ;
set(l5,'Value',a) ;

function listbox3_Callback(hObject, eventdata, handles)
l1=findobj(gcbf,'Tag','listbox1');
l2=findobj(gcbf,'Tag','listbox2');
l3=findobj(gcbf,'Tag','listbox3');
l4=findobj(gcbf,'Tag','listbox4');
l5=findobj(gcbf,'Tag','listbox5');

a = get(l3,'Value');
```

```
set(l2,'Value',a) ;
set(l1,'Value',a) ;
set(l4,'Value',a) ;
set(l5,'Value',a) ;

function listbox4_Callback(hObject, eventdata, handles)
l1=findobj(gcbf,'Tag','listbox1');
l2=findobj(gcbf,'Tag','listbox2');
l3=findobj(gcbf,'Tag','listbox3');
l4=findobj(gcbf,'Tag','listbox4');
l5=findobj(gcbf,'Tag','listbox5');

a = get(l4,'Value');
set(l2,'Value',a) ;
set(l3,'Value',a) ;
set(l1,'Value',a) ;
set(l5,'Value',a) ;

function listbox5_Callback(hObject, eventdata, handles)
l1=findobj(gcbf,'Tag','listbox1');
l2=findobj(gcbf,'Tag','listbox2');
l3=findobj(gcbf,'Tag','listbox3');
l4=findobj(gcbf,'Tag','listbox4');
l5=findobj(gcbf,'Tag','listbox5');

a = get(l5,'Value');
set(l1,'Value',a) ;
set(l2,'Value',a) ;
set(l3,'Value',a) ;
set(l4,'Value',a) ;
```

```
function Robot_Callback(hObject, eventdata, handles)
try
    l1=findobj(gcbf,'Tag','listbox1');
    l2=findobj(gcbf,'Tag','listbox2');
    l3=findobj(gcbf,'Tag','listbox3');
    l4=findobj(gcbf,'Tag','listbox4');
    l5=findobj(gcbf,'Tag','listbox5');

    str1 = get(l1, 'String');
    str2 = get(l2, 'String');
    str3 = get(l3, 'String');
    str4 = get(l4, 'String');
    strsigma = get(l5, 'String');

    len = length(str1);
    sayac=len
    if len > 0
       for index = 1:len
          sonuc1=str1{sayac};
          sonuc2=str2{sayac};
          sonuc3=str3{sayac};
          sonuc4=str4{sayac};
          sonuc5=strsigma{sayac};

          sonuc1=str2num(sonuc1);
          sonuc2=str2num(sonuc2);
          sonuc3=str2num(sonuc3);
          sonuc4=str2num(sonuc4);
          sonuc5=str2num(sonuc5);

          L{index}=link([sonuc1 sonuc2 sonuc3 sonuc4 sonuc5]);
```

```
            aci(1,index)=sonuc3;
            sayac=sayac-1 ;
        end
    end
    r=robot(L);
    r.name='ROBOT';
    drivebot(r,aci);
catch
        errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end


function NewValues_Callback(hObject, eventdata, handles)
try
    str1=findobj(gcbf,'Tag','edit1');
    str2=findobj(gcbf,'Tag','edit2');
    str3=findobj(gcbf,'Tag','edit3');
    str4=findobj(gcbf,'Tag','edit4');
    strsigma=findobj(gcbf,'Tag','edit5');
    str5=findobj(gcbf,'Tag','RobotName');

    liste1=findobj(gcbf,'Tag','listbox1');
    liste2=findobj(gcbf,'Tag','listbox2');
    liste3=findobj(gcbf,'Tag','listbox3');
    liste4=findobj(gcbf,'Tag','listbox4');
    liste5=findobj(gcbf,'Tag','listbox5');

    set(str1,'String','');
    set(str2,'String','');
    set(str3,'String','');
    set(str4,'String','');
    set(strsigma,'String','');
```

```
    set(str5,'String','');

    set(liste1,'String','');
    set(liste2,'String','');
    set(liste3,'String','');
    set(liste4,'String','');
    set(liste5,'String','');
catch
        errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end

function SaveRobot_Callback(hObject, eventdata, handles)
try
    l1=findobj(gcbf,'Tag','listbox1');
    l2=findobj(gcbf,'Tag','listbox2');
    l3=findobj(gcbf,'Tag','listbox3');
    l4=findobj(gcbf,'Tag','listbox4');
    l5=findobj(gcbf,'Tag','listbox5');

    str1 = get(l1, 'String');
    str2 = get(l2, 'String');
    str3 = get(l3, 'String');
    str4 = get(l4, 'String');
    strsigma = get(l5, 'String');

    robotname=findobj(gcbf,'Tag','RobotName')
    robotname=get(robotname,'String')
    robotname_file=strcat(robotname,'_tez.m');
    fid=fopen(robotname_file,'w');
    fprintf(fid,'clear L;\n');
    fprintf(fid,'clear aci;\n');
```

```
len = length(str1);
sayac=len
if len > 0
   for index = 1:len
      sonuc1=str1{sayac};
      sonuc2=str2{sayac};
      sonuc3=str3{sayac};
      sonuc4=str4{sayac};
      sonuc5=strsigma{sayac};

      sonuc1=str2num(sonuc1);
      sonuc2=str2num(sonuc2);
      sonuc3=str2num(sonuc3);
      sonuc4=str2num(sonuc4);
      sonuc5=str2num(sonuc5);

       fprintf(fid,'L{%d}=link([%g %g %g %g %g]);\n', index, sonuc1, sonuc2,
       sonuc3, sonuc4,sonuc5)
      L{index}=link([sonuc1 sonuc2 sonuc3 sonuc4 sonuc5]);
      if (sonuc5==0)
         aci(1,index)=sonuc3;
         fprintf(fid,'aci(1, %d)=%g;\n',index,sonuc3);
      else
         aci(1,index)=sonuc4;
         fprintf(fid,'aci(1, %d)=%g;\n',index,sonuc4);
      end
      sayac=sayac-1 ;
   end
end

fprintf(fid,'%s=robot(L);\n',robotname)
```

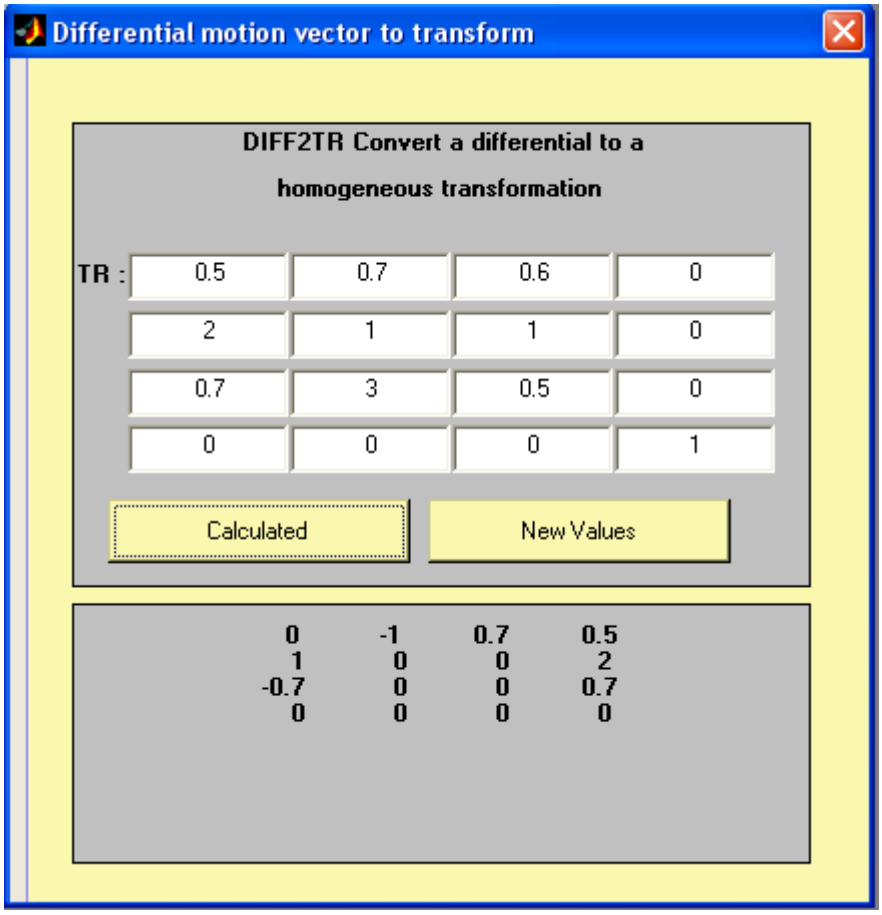
```
  fprintf(fid,'%s.name="%s";\n',robotname,robotname)
  fprintf(fid,'clear L;\n');
  fprintf(fid,'joint_count=%d',len)
  fclose(fid);
catch
      errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end
```

3.3.5. DIFF2TR(Diferansiyel Hareket Vektör Dönüşümü)

DIFF2TR, homojen dönüşüm matrisini diferansiyel matrise dönüştürme

TR = DIFF2TR(D)

Bir homojen dönüşüm matrisi döndürür. Bu olay diferansiyel çevirim ve rotasyon olarak gösterilmektedir. Bu matris, ters simetrik rotasyon altmatrisleri içerir.



Şekil 3.18 DIFF2TR İçin Hazırlanan GUI Görüntüsü

3.3.6. FKINE(Düz/İleri Kinematik Hesaplama)

FKINE, Seri bağlantılı manipülator için düz robot kinematiği

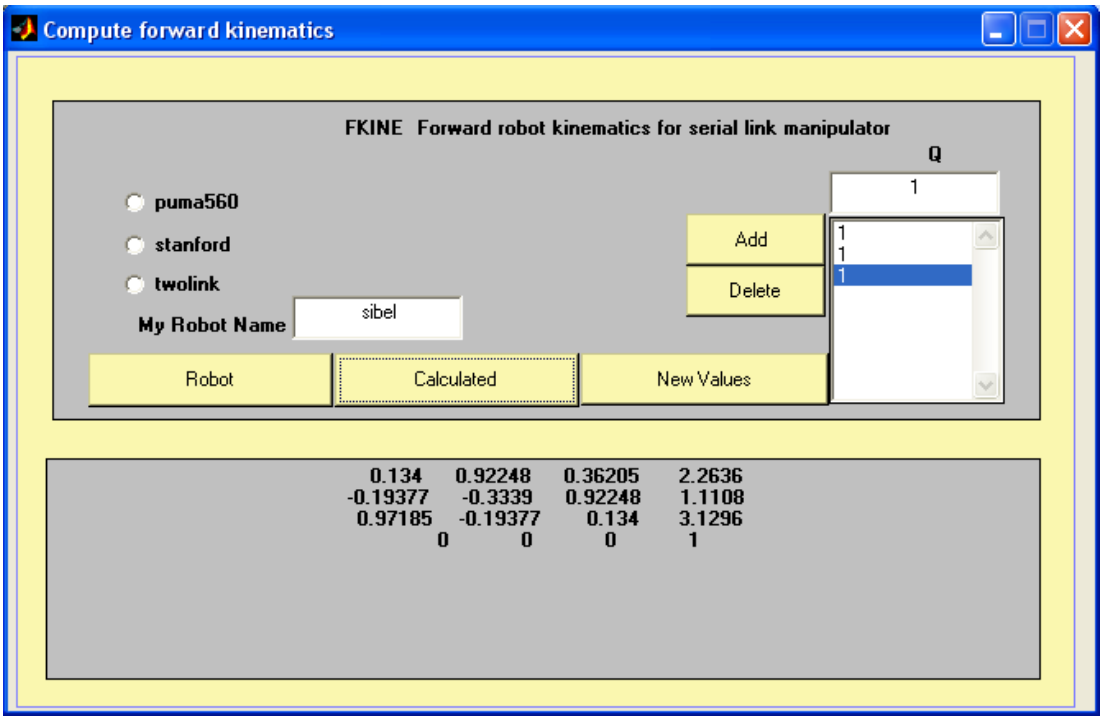
TR = FKINE(ROBOT, Q)

Q'nun içerdiği eklem değişkenlerine göre düz kinematikleri hesaplar. ROBOT parametresi ise bir robot nesnesidir.

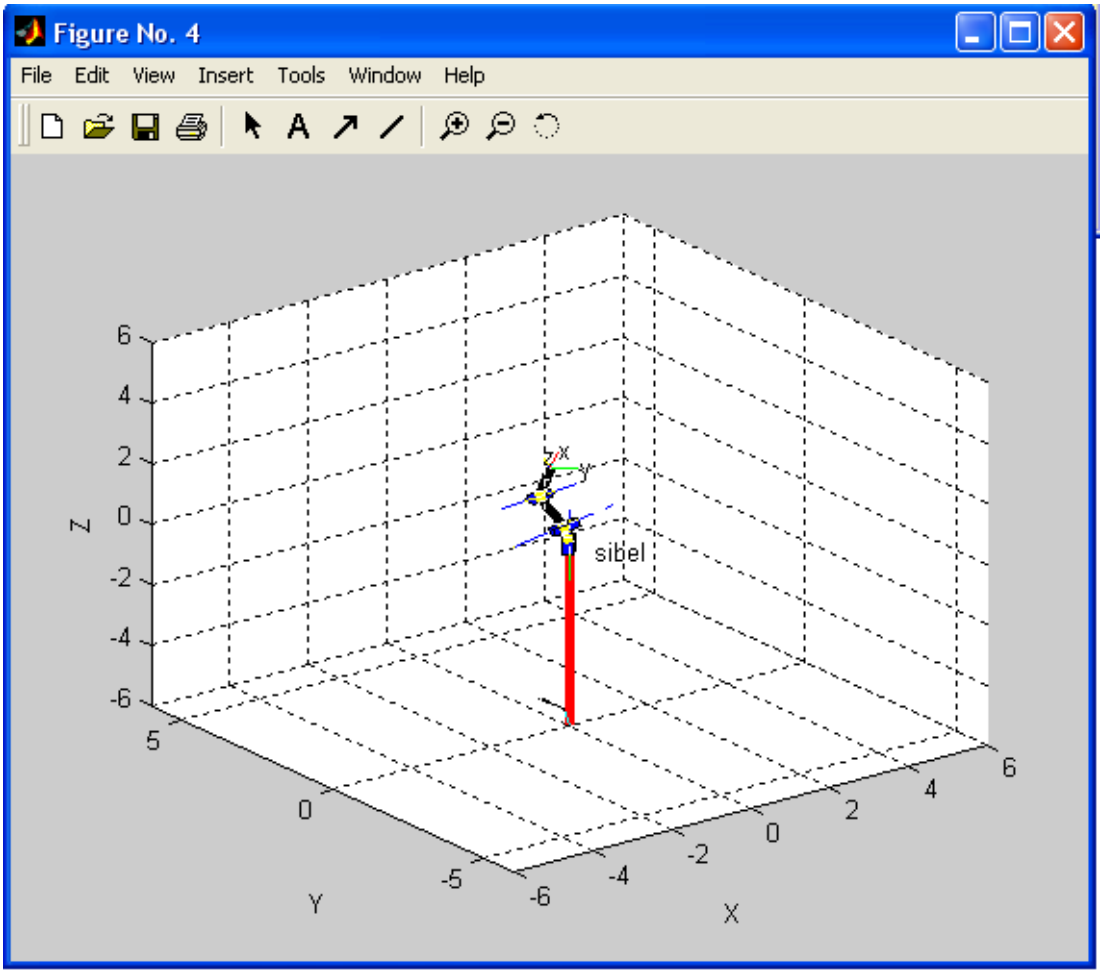
N eksenli manipülator için Q, N elemanlı bir vektör ya da robotun eklem koordinatlarının mxn bir matrisidir.

Eğer Q bir vektör ise, genelleştirilmiş eklem koordinatlarını yorumlar ve FKINE(ROBOT, Q) fonksiyonu geriye 4x4'lük bir homojen matris döndürür.
Eğer Q bir matrisi ise satırlar, bir yörünge boyunca sıralı noktalar için, genelleştirilmiş eklem koordinatları olarak yorumlanmaktadır. Q(i,j), i.yörünge noktası için j. eklem parametresidir. Bu durumda;

FKINE(ROBOT, Q) , son indisi m olan 3 boyutlu bir matrisi verir.



Şekil 3.19 FKINE İçin Hazırlanan GUI Görüntüsü



Şekil 3.20 Kullanıcı Tanımlı Robot Simülasyonu

```
function Calculated_Callback(hObject, eventdata, handles)
try
   l1=findobj(gcbf,'Tag','listbox1');
   str5=findobj(gcbf,'Tag','sonuc');
   str1 = get(l1, 'String');
   len = length(str1);
   sayac=len
   if len > 0
      for index = 1:len
         sonuc1=str1{sayac};
         sonuc1=str2num(sonuc1);
```

```
            q_d(1,index)=sonuc1;
            sayac=sayac-1 ;
         end
      end
      r1=findobj(gcbf,'Tag','puma560');
      r2=findobj(gcbf,'Tag','stanford');
      r3=findobj(gcbf,'Tag','twolink');
      r1=get(r1,'Value');
      r2=get(r2,'Value');
      r3=get(r3,'Value');
      if r1==1
         puma560;
         t=fkine(p560,q_d);
      elseif r2==1
         stanford;
         t=fkine(stanf,q_d);
      elseif r3==1
         twolink;
         t=fkine(t1,q_d);
      else
        robotname=findobj(gcbf,'Tag','RobotName');
        robotname=get(robotname,'String');
        strname=strcat(robotname,'_tez');
        eval(strname);
        q_d=num2str(q_d)
        q_d=strcat('[',q_d,']');
        strson=strcat('fkine(',robotname,',',q_d, ')');
        t=eval(strson);
      end
      t=num2str(t);set(str5,'String',t);
   catch
         errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
```

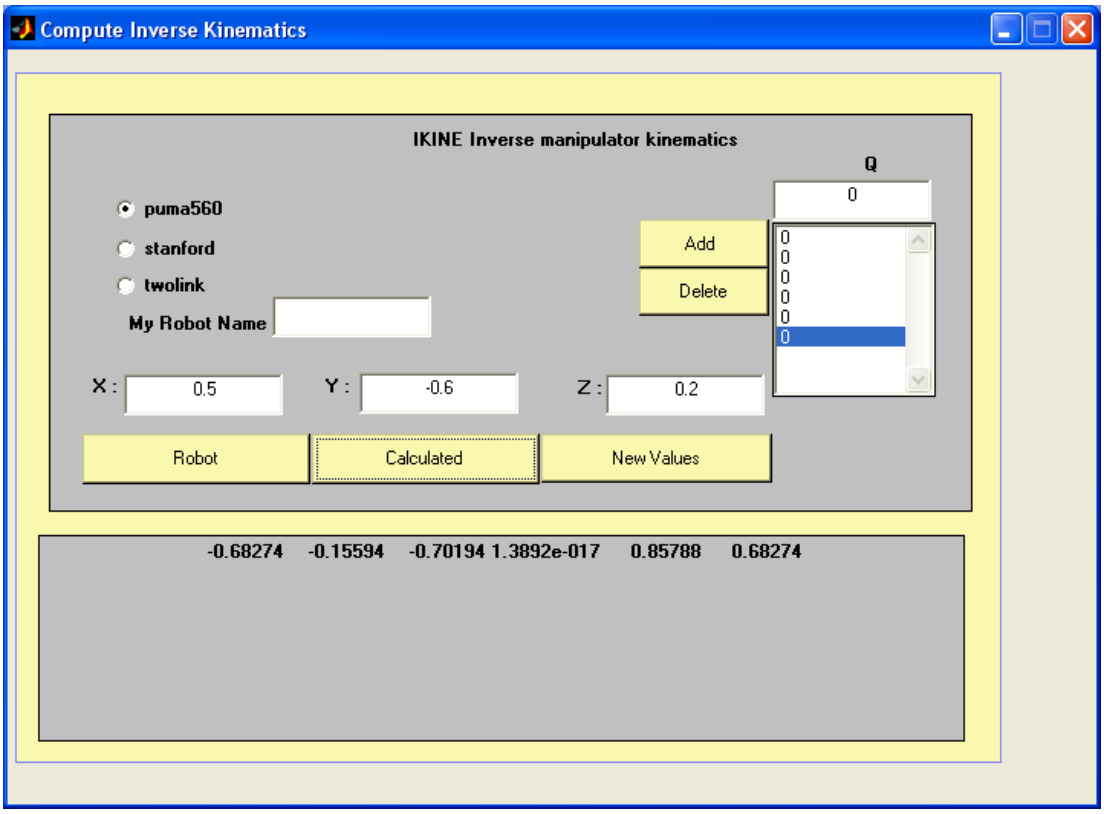
end

3.3.7. IKINE(Ters Kinematik Hesaplama)

IKINE, Ters Manipülatör Kinematiği

Q = IKINE(ROBOT, T)

Bu fonksiyon T uç birim dönüşümüne karşılık gelen eklem koordinatlarını döndürür. Unutulmamalıdır ki, Ters Kinematik çözümleri tek değildir ve başlangıç tahmini varsayılan değeri 0 olan Q'ya bağımlıdır.



Şekil 3.21 IKINE İçin Hazırlanan GUI Görüntüsü

3.3.8. IKINE560(Puma560 Robotu için Ters Kinematik Hesaplama)

IKINE560, Puma560 Robotu için Ters Kinematik

Q = IKINE560(ROBOT, T, CONFIG)

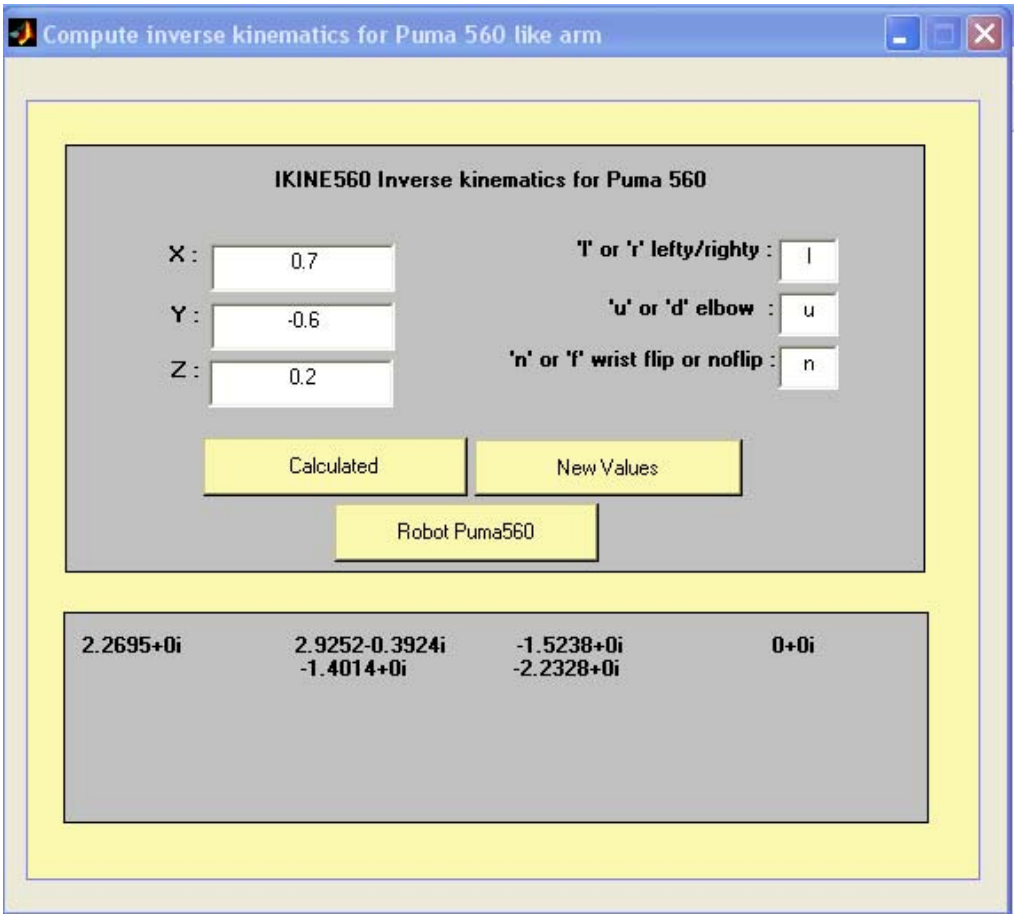
Puma560(küresel eklem) robotunun ters kinematiğini çözer. T ile verilen, robot uç biriminin pozisyonudur.İsteğe bağlı 3. parametre ise, bir ya da daha fazla string yapı içerir. Bu yapı kolun biçimini belirler.

'l' or 'r' lefty/righty (sağa/sola)

'u' or 'd' elbow (dirsek yukarı, dirsek aşağı)

'n' or 'f' wrist noflip or flip (oynamaz bilek/oynar bilek)

Default yapı 'lun' şeklindedir.



Şekil 3.22. IKINE560 İçin Hazırlanan GUI Görüntüsü

3.3.9. JACOB0(Referans Koordinat Sisteminde Jacobian Hesabı)

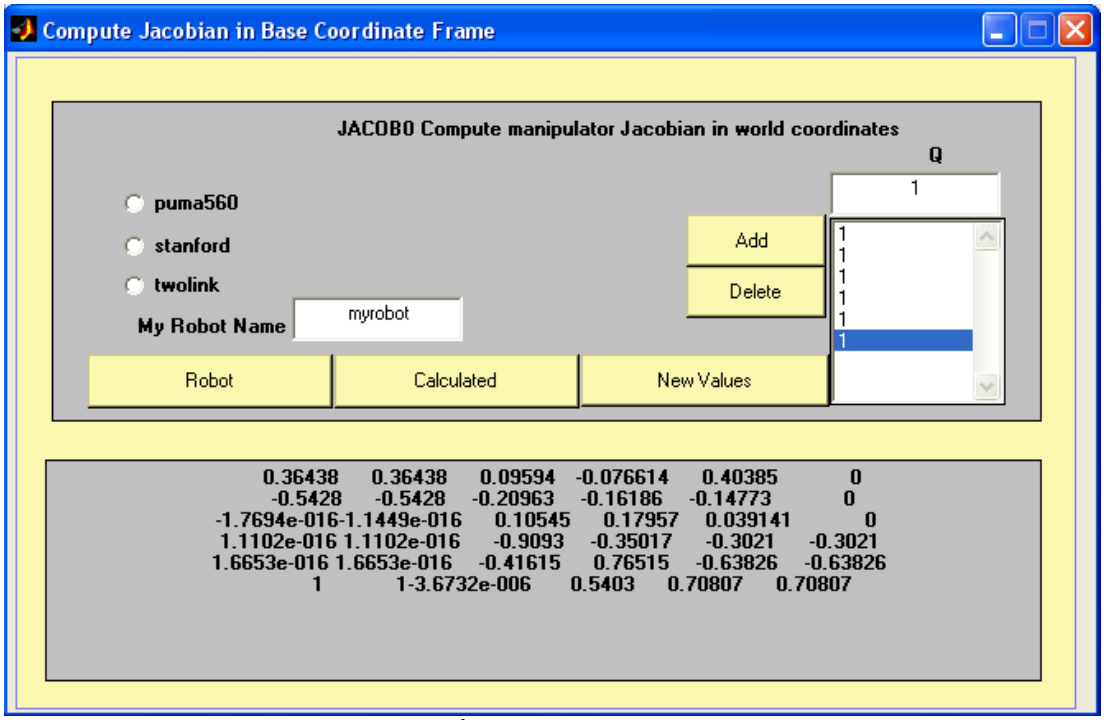
JACOB0 Referans (ilk ekleme göre) koordinat sisteminde jacobian manipülatör hesaplar.

J0 = JACOB0(ROBOT, Q)

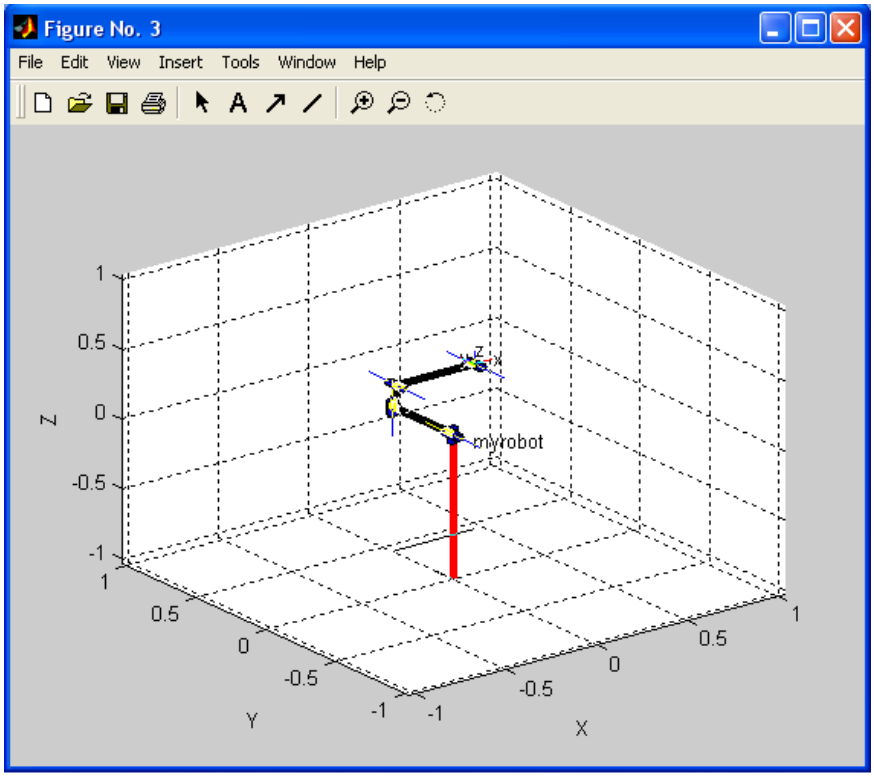
Q pozisyonunda ROBOT adlı manipülatorun Jacobian matrisini döndürür. Manipülator Jacobian matrisi eklem uzayındaki diferansiyel değişimleri uç birimin diferansiyel kartezyen hareketleriyle ilişkilendirir.

$$dX = J\, dQ$$

n eklemli manipülator için Jacobian 6xn lik bir matristir.



Şekil 3.23 JACOB0 İçin Hazırlanan GUI Görüntüsü



Şekil 3.24 JACOB0 İçin Robot Simülasyonu

3.3.10. JACOBN (Uç Birim Koordinat Sisteminde Jacobian Manipülator Hesabı)

JACOBN, Compute Uç birim(end-effector) koordinat sisteminde Jacobian manipülator hesaplar

JN = JACOBN(ROBOT, Q)

Q pozisyonunda ROBOT adlı manipülatorun Jacobian matrisini döndürür. Q pozisyonunda ROBOT adlı manipülatorun Jacobian matrisini döndürür. Manipülator Jacobian matrisi eklem uzayındaki diferansiyel değişimleri uç birimin diferansiyel kartezyen hareketleriyle ilişkilendirir.

$$dX = J\, dQ$$

n eklemli manipülator için Jacobian 6xn lik bir matristir..

3.3.11. TR2DIFF(Dönüşüm Matrisinden Diferansiyel Hareket Vektörü)

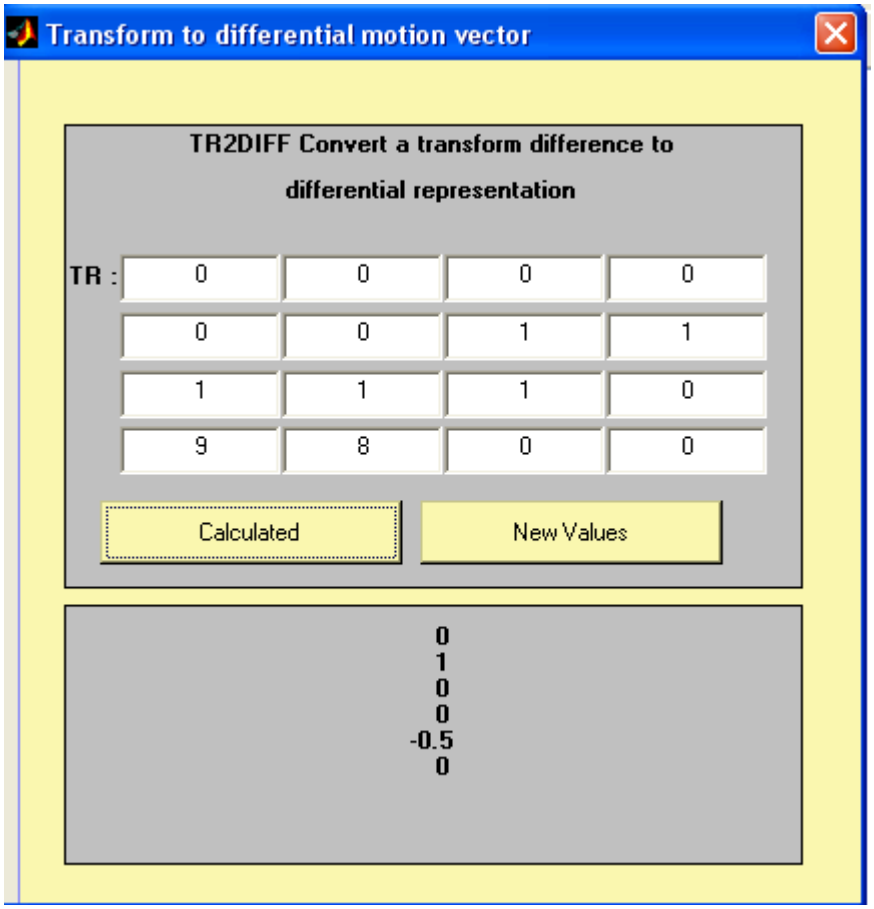
TR2DIFF dönüşüm farkını diferansiyel gösterime çevirir.

D = TR2DIFF(T)

D = TR2DIFF(T1, T2)

İlk yapı, sonsuz küçük bir hareketi temsil eden homojen dönüşüm matrisini 6 elemanlı bir diferansiyel gösterime çevirir. Böyle bir dönüşüm yaklaşık olarak ters simetrik olan bir rotasyon altmatrisi içerir.

İkinci durumda ise t1 den t2 ye hareket etmek için gereken 6 elemanlı diferansiyel vektörü verir.



Şekil 3.25 TR2DIFF İçin Hazırlanan GUI Görüntüsü

3.3.12. TR2JAC(Homojen Dönüşüm Matrisinden Jacobian Matrisi)

TR2JAC Homojen dönüşüm matrisinden jacobian matrisi elde edilebilir

J = TR2JAC(T)

T homojen dönüşüm matrisine bağlı olarak 6x6 bir jacobian matrisi verir.

3.4. YÖRÜNGE TÜRETİMİ (TRAJECTORY GENERATION)

Manipülatörün uç noktasının, başlangıç noktasından son durumuna kadar hareketi esnasında, yer değiştirme ve dönme yollarını belirleyen noktalar kümesi "Yörünge" olarak isimlendirilir. Robot hareketinde temel problem, robot ucunu, o anki başlangıç değerinden (Tbaşlangıç), istenilen bir son değere (Tson) taşımaktır. Bu harekette, robot kolunun hem yönü hem de pozisyonu değişmektedir. Yolun daha detaylı tanımlanması istendiğinde, uç nokta için, başlangıç ve sonuç noktaları arasında geçiş noktaları ya da ara noktalar tanımlanmalıdır. Bunlara ilave olarak, iki geçiş noktası arasındaki hareketin süresi de yol tanımlanırken verilebilir.

3.4.1. CTRAJ(Kartezyen Yörünge)

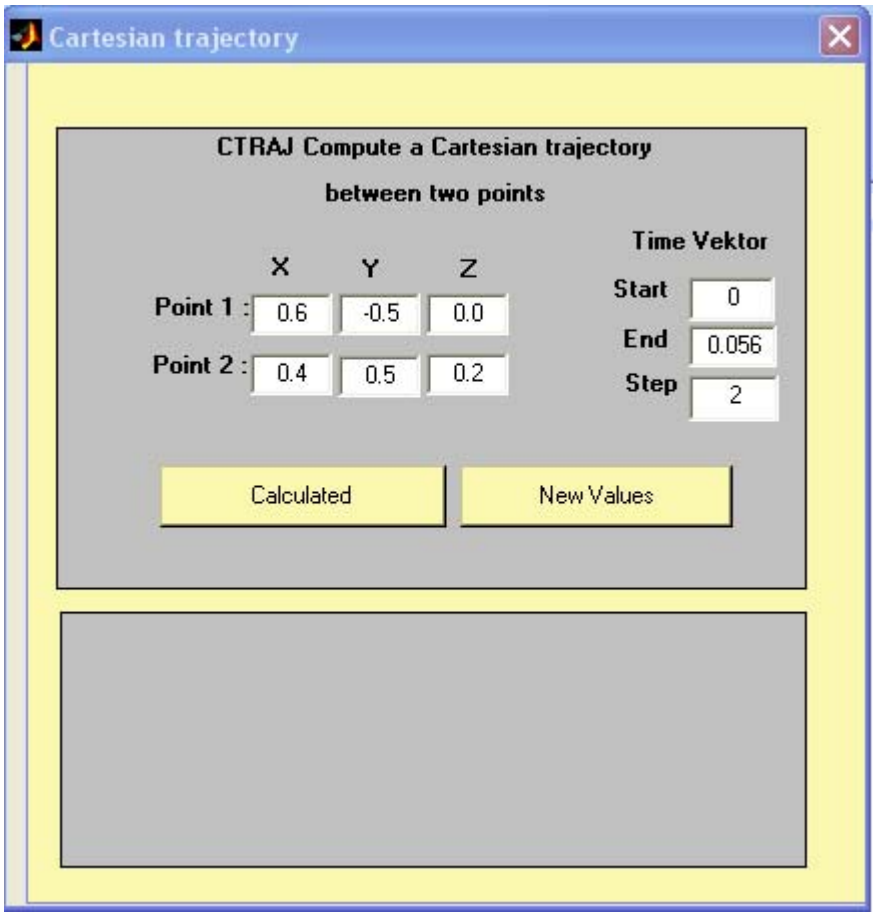
Kartezyen yörünge türetiminde esas olan robot ucunu, o anki başlangıç değerinden (Point1), istenilen son değere (Point2) taşımaktır. Programda görsel olarak ilk ve son nokta değerleri ve t'ye göre zaman vektörü girilecektir.

CTRAJ, iki nokta arasındaki kartezyen yörünge hesaplama

TC = CTRAJ(T0, T1, N)

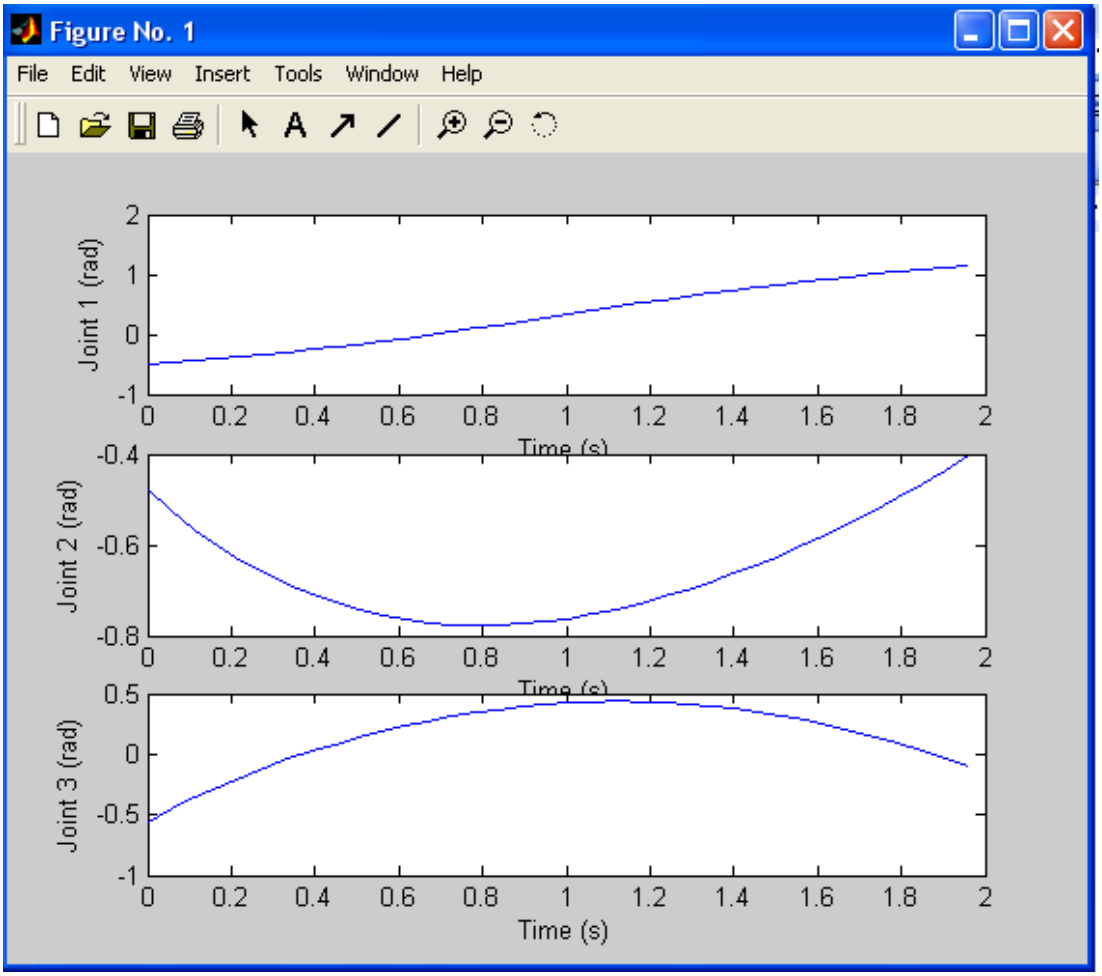
TC = CTRAJ(T0, T1, R)

Bu fonksiyon T0 - T1 noktasından, TC Kartezyen yörünge hesabını döndürür. Noktaların sayısı N ve verilen yol mesafe vektörünün uzunluğu R'dir.İlk durumda, T0 ve T1 arasındaki noktalar eşit aralıklıdır. İkinci durumda R yol mesafesini verir ve R'nin elemanları [0 1] arasında olmalıdır.



Şekil 3.26 Kartezyen Yörünge Türetimi Ekran Görüntüsü

Yukarıdaki ekran görüntüsüne gerekli değerler girilip *"Calculated"* düğmesine tıklandığında Şekil 3.26 'de görüldüğü gibi uç noktanın izlediği yolun kartezyen koordinatlardaki çizimi ekranda görülür.



Şekil 3.27 Uç Noktanın Kartezyen Koordinatlarda İzlediği Yol

Kartezyen Yörünge Türetimi için yazılmış olan *.m dosyası aşağıdaki gibidir:

```
function Calculated_Callback(hObject, eventdata, handles)
try
p1_x=findobj(gcbf,'Tag','Point1_X');
p1_y=findobj(gcbf,'Tag','Point1_Y');
p1_z=findobj(gcbf,'Tag','Point1_Z');
p2_x=findobj(gcbf,'Tag','Point2_X');
p2_y=findobj(gcbf,'Tag','Point2_Y');
p2_z=findobj(gcbf,'Tag','Point2_Z');
t1=findobj(gcbf,'Tag','T1');
t2=findobj(gcbf,'Tag','T2');
```

```
t3=findobj(gcbf,'Tag','T3');

p1_x=str2num(get(p1_x,'String'));
p1_y=str2num(get(p1_y,'String'));
p1_z=str2num(get(p1_z,'String'));
p2_x=str2num(get(p2_x,'String'));
p2_y=str2num(get(p2_y,'String'));
p2_z=str2num(get(p2_z,'String'));
alinan_t1=str2num(get(t1,'String'));
alinan_t2=str2num(get(t2,'String'));
alinan_t3=str2num(get(t3,'String'));
 P1 = transl(p1_x, p1_y, p1_z) % define the start point
 P2 = transl(p2_x, p2_y, p2_z) % and destination
 Tm=[alinan_t1:alinan_t2:alinan_t3] % create a time vector
 str4=findobj(gcbf,'Tag','sonuc');
 set(str4,'String','');
 TT=ctraj(P1, P2, length(Tm));
 s=ctraj(P1, P2, length(Tm));
 s=num2str(s)
 set(str4,'String',s);
  puma560;
  q = ikine(p560, TT);
  figure(1);
 subplot(3,1,1);
    plot(Tm,q(:,1));
    xlabel('Time (s)');
    ylabel('Joint 1 (rad)');
    subplot(3,1,2);
    plot(Tm,q(:,2));
    xlabel('Time (s)');
    ylabel('Joint 2 (rad)');
    subplot(3,1,3);
```

```
   plot(Tm,q(:,3));
   xlabel('Time (s)');
   ylabel('Joint 3 (rad)');
catch
      errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end
```

3.4.2. JTRAJ (Eklem Uzay Yörüngesi)

JTRAJ, İki Nokta arasındaki "joint space trajectory" hesaplar.

```
[Q QD QDD] = JTRAJ(Q0, Q1, N)
[Q QD QDD] = JTRAJ(Q0, Q1, N, QD0, QD1)
[Q QD QDD] = JTRAJ(Q0, Q1, T)
[Q QD QDD] = JTRAJ(Q0, Q1, T, QD0, QD1)
```

Bu fonksiyon, joint space trajectory Q'nun Q0 dan Q1'e değerini döndürür. Noktaların sayısı N veya verilen zaman vektörünün uzunluğu T'dir. Hız ve hızlanma için varsayılan sınır değerleri 0 olan , 7. dereceden bir polinom kullanılır. Hız sınırları istendiğinde QD0 ve QD1 olarak belirlenebilir.

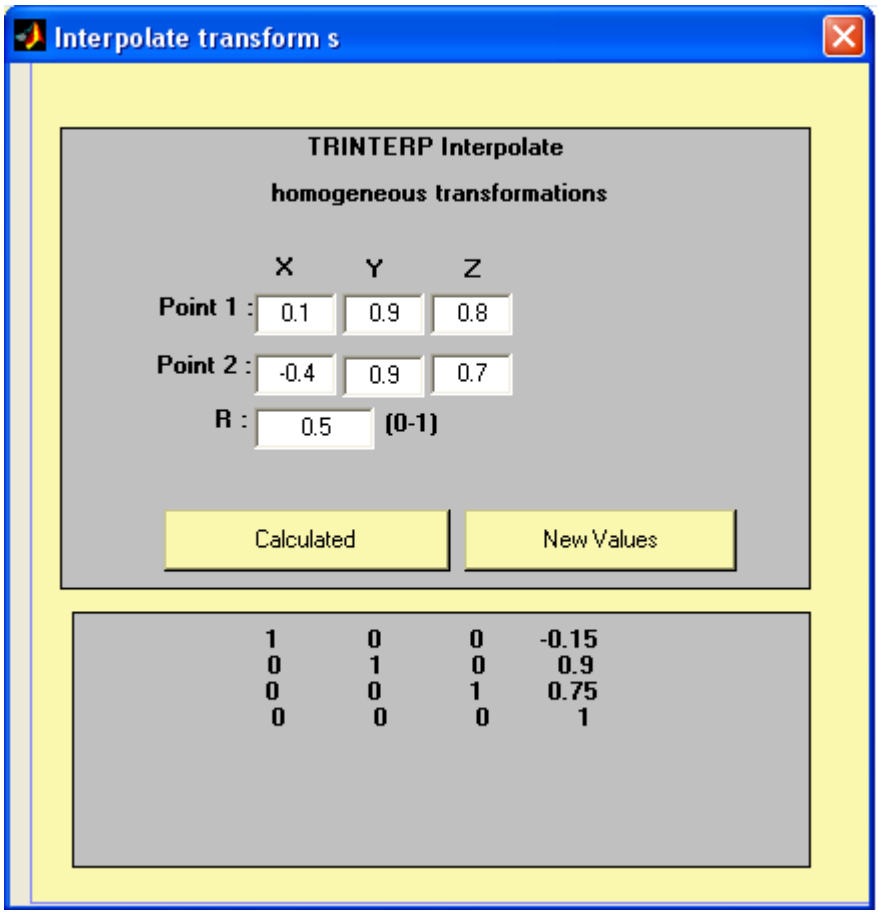
Bu fonksiyon, opsiyonel olarak hız ve hızlanma yörüngelerini QD ve QDD olarak döndürür. Her bir yörünge, mxn'lik bir matristir. Bu matrisin her bir satırı zaman, her bir sutünuda eklem parametresidir.

3.4.3. TRINTERP(Homojen Dönüşüm Enterpolasyonu)

TRINTERP Homojen Dönüşüm Enterpolasyonu

```
TR = TRINTERP(T0, T1, R)
```

T0 ve T1 arasındaki homojen dönüşüm enterpolasyonunu 0 ile 1 aralığında R değeri olarak döndürür.



Şekil 3.28 TRINTERP İçin Hazırlanan GUI Görüntüsü

```
function Calculated_Callback(hObject, eventdata, handles)
try
    alinan_a1=str2num(get(findobj(gcbf,'Tag','A1'),'String'));
    alinan_a2=str2num(get(findobj(gcbf,'Tag','A2'),'String'));
    alinan_a3=str2num(get(findobj(gcbf,'Tag','A3'),'String'));
    alinan_n1=str2num(get(findobj(gcbf,'Tag','N1'),'String'));
    alinan_n2=str2num(get(findobj(gcbf,'Tag','N2'),'String'));
    alinan_n3=str2num(get(findobj(gcbf,'Tag','N3'),'String'));
    alinan_t1=str2num(get(findobj(gcbf,'Tag','T1'),'String'));
    P1 = transl(alinan_n1, alinan_n2, alinan_n3) % ilk nokta
    P2 = transl(alinan_a1, alinan_a2, alinan_a3) % ikinci nokta
    str4=findobj(gcbf,'Tag','sonuc');
    set(str4,'String','');
```

```
    s=TRINTERP(P1, P2, alinan_t1);
    s=num2str(s)
    set(str4,'String',s);
catch
    errordlg('Yanlış yada eksik değer girildi.','Hata Penceresi','modal');
end
```

3.5. DİNAMİKLER (DYNAMICS)

Manipülatör dinamiği hareketlendiriciler veya dış kuvvetler tarafından uygulanan momentler sonucu hareket eden manipülatörün hareket denklemleri ile ilgilenir. Tersi biçimde hareket denklemleri oluşturulmuş bir manipülatör için uygulanması gereken hareketlendirici momentlerinin hesabı da manipülatör dinamiği yardımıyla yapılır. Bu çözümleme fiziksel bir sisteme gerek kalmadan bir manipülatörün yapısını, hareketlendiricileri, sürücüleri, kontrol yaklaşımlarını ve hareket planını tasarlamada önemli bir rol oynamaktadır.

Manipülatör dinamiğini çözümlemede ve hareket denklemlerini oluşturmada kullanılan iki önemli eşitlik takımı vardır:

1. Newton-Euler(NE) eşitlikleri : Newton-Euler(NE) eşitlikleri ise her bir uzuv için Newton Yasaları'na bağlı olarak hareket eşitliklerini türetir.

2. Lagrange eşitlikleri : Lagrange eşitlikleri basittir ve sistematik olarak elde edilebilir. Ayrıca sistemin mekanik yapısıyla ilgili tüm ayrıntılar(elastik deformasyon, eylemsizlik momenti vb.) bu eşitliklerde görülmektedir. Bu sebeple gerçek bir manipülatör tasarımı sırasında Lagrange eşitlikleri mutlaka çıkarılmalıdır.

Lagrange denklemleri mekanik sistemin kinetik ve potansiyel enerjileri arasındaki farktan yola çıkarak oluşturulur.

$$L = K - V$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i$$

L=Lagrange işlevi

K=Kinetik enerji

V=Potansiyel enerji

Lagrange denkleminde gerekli terimler yerine konulduğunda aşağıdaki genel hareket eşitliği ortaya çıkar.

$$\underline{Q} = M(\underline{q})\ddot{\underline{q}} + C(\underline{q},\dot{\underline{q}})\dot{\underline{q}} + F(\dot{\underline{q}}) + G(\underline{q})$$

Çizelge 3.2. Robot Dinamik Denklemindeki Terimlerin Anlamları

Terim	Anlamı
$\underline{q},\dot{\underline{q}},\ddot{\underline{q}}$	Eklemlerin açı, hız ve ivme vektörleri
M	Manipülatör eylemsizlik matrisi
C	Coriolis ve merkezkaç(centrifugal) torkları Coriolis torkları ile orantılı iken merkezkaç torkları ile orantılıdır.
F	Viskoz ve Coulomb sürtünmeleri
G	Yerçekimi kuvvetleri
$\underline{Q}$	Genelleştirilmiş kuvvetler(kuvvet veya tork)

3.5.1. ACCEL(Düz/İleri Dinamik Hesabı)

ACCEL, Manipülator için düz kinematikleri hesaplar

QDD = ACCEL(ROBOT, Q, QD, TORQUE)

QDD = ACCEL(ROBOT, [Q QD TORQUE])

Eyleyici torku, TORQUE vektörünün Q ve OD durumunda ROBOT'a uygulanmasıyla elde edilen eklem ivmeleri vektörü QDD'yi verir.

3.5.2. CINERTIA (Kartezyen Manipülator Eylemsizlik Matrisi Hesabı)

CINERTIA, Kartezyen(hareket alanı) manipülator eylemsizlik matrisini hesaplar.

M = CINERTIA(ROBOT, Q)

Bu fonksiyon nxn eylemsizlik matrisini verir. Bu matriste; Kartezyen Kuvvet/Tork ‘u ile Kartezyen hızlanma arasında ilişki kurar. ROBOT ise n eksenli robot nesnesidir ve Q, n elemanlı eklem durumunun vektörüdür.

3.5.3. CORIOLIS (Merkezkaç/Coriolus Tork Hesabı)

CORIOLIS, Manipülator Coriolis matrisini hesaplar.

C = CORIOLIS(ROBOT, Q, QD)

Q ve QD ile belirlenen eklem konumu ve hızlarına karşılık gelen Merkezkaç/Coriolus Torkunu hesaplar. Robot, robot nesnesidir.

3.5.4. FRICTION (Eklem Sürtünmesi)

FRICTION, Bir robot için sürtünme torklarını hesaplar.

TAU = FRICTION(ROBOT, QD)

QD eklem hızına sahip belirli bir robot için, eklem sürtünme torklarının vektörünü verir.

3.5.5. FTRANS(Kuvvet/Moment(Tork) Dönüşümü)

FTRANS, Güç/Moment Dönüşümü

FT = FTRANS(T, F)

F ile verilen kuvvet/tork vektörünü temel koordinat sistemine göre dönüşüm matrisi T olan koordinat sistemine aktarır. Fx, Fy, Fz Kuvvet vektörü, Mx, My, Mz Tork vektörüdür. F ve FT [Fx Fy Fz Mx My Mz] kuvvet ve tork vektörlerinden oluşan 6 elemanlı vektördür.

3.5.6. GRAVLOAD(Yerçekimi Etkisi Hesabı)

GRAVLOAD, Manipülator eklemlerindeki yer çekimi yükünü hesaplar.

TAUG = GRAVLOAD(ROBOT, Q)

TAUG = GRAVLOAD(ROBOT, Q, GRAV)

Q konfigürasyonunda ki Robot manipülatorunun eklem yerçekimi etkisini hesaplar. Eğer Q bir vektör ise, sonuç eklem torklarının bir satır vektörüdür. Eğer Q bir matris ise her satır bir eklem durum vektörü olarak değerlendirilir. Bu durumda sonuç her bir satırı eklem torklarına karşılık gelen bir matristir.

Yerçekimi vektörü, GRAV argümanıyla verilir. Verilmediği taktirde Robotun değerlerini varsayılan olarak alır.

3.5.7. INERTIA (Manipülator Eylemsizlik Matrisini)

INERTIA Manipülator Eylemsizlik matrisini hesaplar

INERTIA(ROBOT, Q)

Geriye nxn'lik simetrik eylemsizlik matrisini döndürür. Bu matris, eklem torklarıyla eklem ivmelerini ilişkilendirir.

3.5.8. ITORQUE (Eylemsizlik Tork Hesabı)

ITORQUE, Manipulator eylemsizlik torkunu hesaplar

TAUI = ITORQUE(ROBOT, Q, QDD)

Bu fonksiyon, belirli bir pozisyon ve hızlanmada n elenamlı eylemsizlik tork vektörünü verir. Bu da aşağıdaki formülle hesaplanır.

TAUI = INERTIA(Q)*QDD

ROBOT parametresi ile manipülatorun dinamik ve kinematiğini tanımlanır. Eğer Q ve ODD satır vektörü ise, sonuç eklemlerin torkunun bir satır vektörüdür. Eğer Q ve QDD matris ise, her satır bir eklem durum vektörü olarak değerlendirilir. Bu durumda sonuç her bir satırı eklem torklarına karşılık gelen bir matristir.

3.5.9. NOFRICTION(Bir Robotta Sürtünme Etkisi Sıfırlanması)

NOFRICTION, Robotu eklem sürtünmelerinin sıfırı olduğu robot haline getirir.

ROBOT = NOFRICTION(ROBOT)

Liner olmayan (Coulomb) sürtünme katsayıları sıfır olan robot döndürür.

ROBOT = NOFRICTION(ROBOT, 'all')

Bütün sürtünme katsayıları sıfır olan bir robot döndürür.

3.5.10. RNE(Ters Dinamik)

RNE, Yinemeli (recursive) Newton-Eular eşitliği aracılığı ile ters dinamiği hesaplar.

TAU = RNE(ROBOT, Q, QD, QDD)

TAU = RNE(ROBOT, [Q QD QDD])

Belirli bir eklem pozisyonu, hızı ve ivmesine ulaşmak için gerekli eklem torkunu verir.

4. BULGULAR VE TARTIŞMA

Bu çalışma MATLAB GUI ile Matlab içerisindeki ToolBox'ların görselleştirilmesi açısından birçok uygulama içermektedir. Bu açıdan yaptığımız çalışma bu tip görselleştirme çalışmalarına önemli kaynak sunacaktır.

Robotik Toolbox'ın bazı noktalarda eksikliği ortaya konmuştur. Robotik Toolbox ile yapılan çok az uygulamanın olduğu ve bu uygulamaların çoğunun GUI ile gerçekleştirilmemiş olduğu tespit edilmiştir.

Bu çalışmadaki amaç *Robotics Toolbox*'ı mevcut kullanımından çıkarıp kullanıcıların, görsel bir arayüz ile kullanmalarını sağlamaktı ve bu amaca ulaşılmıştır. Robotik toolbox içerisindeki hazır robotların dışında kullanıcı tanımlı yeni robotlar oluşturulması ve kaydedilmesi de sağlanmıştır. Daha önce incelenen hiçbir çalışmada kendi robotunu oluşturulup kaydedilmemiştir.

5. SONUÇ VE ÖNERİLER

5.1. SONUÇLAR

Bu çalışmada Matlab GUI ile hazırlanmış "Görsel Robotik Araç Kutusu Tasarlanmıştır". Böylelikle robot sistemleri için genel bir simülasyon ortamı hazırlanmıştır. Yazılım, belli başlı robotlar için gerekli matematiksel karakteristiklerini tanımakta ve yeni bir robot manipülatörü tanımlanmasına da olanak sağlamaktadır. Tanımlanan bu robotlar m-file dosyaları olarak kaydedilmekte ve istenildiğinde kullanılmaktadır. Görsel ekranlar aracılığı ile girilen parametrelere göre robotun hareketlerinin görsel bir animasyonu ve gerekli diğer analizleri yapılmaktadır. Kullanıcıların Matlab Programlama Dili, herhangi bir formül ya da komutlarını bilmeksizin işlemlerini yapmaları sağlanmıştır.

Hazırlanan araç kutusu için fonksiyonel öğeler kaldırılıp, verilen bir teoremin hesaplanmasında kullanılacak formül veya komutu bilmeye gerek kalmadan, sadece gerekli parametreleri girerek, görsel olarak robotik uygulamaları yapmak mümkün olmaktadır. Bir diğer nokta ise kullanıcının en az hata ile görsel bir platformda işlemlerini gerçekleştirebilmesidir.

Çalışmada Matlab GUI'nin nasıl kullanılacağı ile ilgili çok çeşitli ve zengin açıklamalar getirilmiştir. Robotik Toolbox için GUI uygulamaları anlatılmış ve algoritmaları verilmiştir.

5.2. ÖNERİLER

Bu tez çalışmasından sonra MATLAB'ın diğer araç kutularıda tasarlanabilir. Hatta nesnesel dillerden oluşan bir fonksiyonel program yapısı oluşturularak MATLAB'ın içerisine entegre edilirse, bir DLL kütüphanesi kullanıyormuşuz gibi diğer kullanıcılara GUI ile oluşturacakları kolay programlar sunulabilir.

GUI ile programlarda var olan mevcut nesneler yapılacak işlemleri tam anlamıyla karşılamamaktadır. Bu konuda yeni nesneler oluşturulmalıdır. Ayrıca Robotik Toolbox içerisindeki komutların yardım dosyalarının çok yetersiz olması, bu yardım dosyalarının içeriğinin zenginleştirilmesi.

KAYNAKLAR

[1] The MathWorks Inc., Matlab Users Guide, USA, http://www.mathworks.com.

[2] Corke P., "Robotics Toolbox For Matlab(Release 6)", April (2001)

[3] Mathworks Inc., "User Reference For Matlab Ver.6.5", (2003)

[4] Vural M.S., "Matlab Grafiksel Arabirimi Yardımıyla Görsel Mikrodalga Araç Kutusu Tasarımı", Yüksek Lisans Tezi, Mersin Üni. Fen Bil. Ens., (2005)

[5] Gourdeau R., "Object Oriented Programming for Robotic Manipulator Simulation", IEEE Robotics and Automation Magazine, 4(3):21-29 (1997)

[6] Halsall F., "Data Communications, Computer Networks and Open Systems", Third Edition, Addison-Wesley, (1992)

[7] Pires J.N., Sá da Costa J.M.G., "Object Oriented and Distributed Approach for Programming Robotic Manufacturing Cells", IFAC Journal on Robotics and Computer Integrated Manufacturing, (1999)

[8] Yüksel İ., "Matlab İle Mühendislik Sistemlerinin Analizi ve Çözümü", Türkmen Yayınevi, (2000)

[9] Uzunoğlu M., Kızıl A., "Onar Ö. Ç., Kolay Anlatımı İle İleri Düzeyde Matlab 6.0-6.5," Türkmen Yayınevi, (2002)

[10] K.S.Fu, R.C.Gonzalez, C.S.G.Lee, "Robotics Control, Sensing, Vision, and Intelligence", McGrav-Hill Book Company, (1987)

[11] Savaş K., Kontrol Sistemleri İçin Matlab'da GUI Uygulamaları, Tez, Marmara Üniversitesi, İstanbul 2007

[12] Yüksel T., "Robotikte Matlab Kullanımı", Yüksek Lisans Tezi, O.M.U Elektrik-Elektronik Müh., (2004)

[13] Kaplan S., Canbolat H., "Matlab GUI Yardımıyla Görsel bir Robotik Araç Kutusu Tasarımı", TOK09 Otomatik Kontrol Milli Toplantısı, İstanbul, 13-16 Ekim (2009)

ÖZGEÇMİŞ

1980 yılında Sivas - Zara'da doğdu. İlkokulu Yüzüncü Yıl İlkokulunda, orta okulu Zara Lise ortaokul kısmında ve liseyi de İzmir Aliağa Lisesi'nde tamamladı.

1998-2002 yılları arasında Mersin Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde Lisans eğitimini yapıp hem bölüm hem fakülte birincisi olarak mezun oldu.

2002 yılında Mersin Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Anabilim dalında Yüksek Lisans öğrenimine başladı.

2005 yılından beri Mersin Üniversitesi Bilgisayar Mühendisliği Bölümünde okutman olarak görev yapmakta olup evli ve bir çocuk annesidir.