**T.C.**
**PAMUKKALE ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**MAKİNA MÜHENDİSLİĞİ ANABİLİM DALI**

# LİNEER AKTÜATÖRLÜ PARALEL MODÜLLERDEN OLUŞAN SERİ ROBOTUN KONTROLÜ

**YÜKSEK LİSANS TEZİ**

**YALÇIN BULUT**

**DENİZLİ, ARALIK - 2015**

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MAKİNA MÜHENDİSLİĞİ ANABİLİM DALI


# LİNEER AKTÜATÖRLÜ PARALEL MODÜLLERDEN OLUŞAN SERİ ROBOTUN KONTROLÜ

YÜKSEK LİSANS TEZİ

YALÇIN BULUT

DENİZLİ, ARALIK - 2015

## KABUL VE ONAY SAYFASI

**Yalçın BULUT** tarafından hazırlanan "**LİNEER AKTÜATÖRLÜ PARALEL MODÜLLERDEN OLUŞAN SERİ ROBOTUN KONTROLÜ**" adlı tez çalışmasının savunma sınavı 25.12.2015 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / ~~oy çokluğu~~ ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Makina Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

| Jüri Üyeleri | İmza |
|---|---|
| Danışman<br>Prof. Dr. Erdinç Şahin ÇONKUR | ..... |
| Üye<br>Doç. Dr. Murat SARI | ..... |
| Üye<br>Yrd. Doç. Dr. Yasin YILMAZ | ..... |

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 06.01.2016 tarih ve ..01/14.... sayılı kararıyla onaylanmıştır.

.....

Prof. Dr. Orhan KARABULUT

Fen Bilimleri Enstitüsü Müdürü

**Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.**

**Yalçın BULUT**

# ÖZET


**LİNEER AKTÜATÖRLÜ PARALEL MODÜLLERDEN OLUŞAN SERİ ROBOTUN KONTROLÜ**

**YÜKSEK LİSANS TEZİ**

**YALÇIN BULUT**

**PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**MAKİNA MÜHENDİSLİĞİ ANABİLİM DALI**

**(TEZ DANIŞMANI:PROF. DR. ERDİNÇ ŞAHİN ÇONKUR)**

**DENİZLİ, ARALIK - 2015**

Seri robot manipülatörleri, ardışık iki uzuv arasına yerleştirilen aktüatörler ve aktüatöre bağlı redüktörler vasıtasıyla oluşturulur. Bu aktüatör ve redüktörlerin, mafsallara binen yükü taşıyabilecek kadar kapasiteli olabilmesi için ağır ve büyük boyutta olması gerekliliği dezavantaj olarak görülmektedir. Bu tezde, paralel manipülatörlerin seri olarak bağlanmasıyla oluşturulmuş hibrid bir yılansı robot tasarımı yapılarak bu dezavantajın önüne geçilmesi planlanmıştır. 3-RPS paralel manipülatörün ileri kinematiğinin çözülebilmesi için denklem sistemi elde edilmiş ve bu denklem sistemi Newton Metodu kullanılarak sayısal olarak çözülmüştür. C Sharp XNA ortamında 10 modüllü yılansı robotun üç boyutlu simülasyon programı yapılmıştır. 3-RPS paralel manipülatörün Maple kullanılarak çalışma alanı analizi yapıldıktan sonra bu manipülatörün ters kinematiğinin çözülebilmesi için elde edilen denklem sistemi, Aralıklı Newton Metodu ile sayısal olarak çözülmüştür. Yılansı robotun tahriki için seçilen lineer aktüatörün robotun kendi ağırlığı altında dayanıp dayanamayacağının analiz edilebilmesi için, Catia'da modellenen yılansı robot Ansys'e aktarılarak ilk modülün lineer aktüatörlerinin en uç noktalarındaki statik reaksiyon kuvvet ve momentleri simüle edilmiştir. Elde edilen sonuçlara göre, 10 modüllü yılansı robotun piyasada mevcut olan standart lineer aktüatörler ile mekanik olarak uygulanabilir olmadığı görülmüştür.



**ANAHTAR KELİMELER:** Yılansı robotlar, 3-RPS paralel manipülatör, hibrid manipülatörler, gereğinden çok serbestlik dereceli robotlar.

# ABSTRACT

**CONTROL OF SERIAL ROBOT CONSISTING OF PARALLEL MODULES WITH LINEAR ACTUATORS**
**MSC THESIS**
**YALÇIN BULUT**
**PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE**
**MECHANICAL ENGINEERING**
**(SUPERVISOR:PROF. DR. ERDINC SAHIN CONKUR)**
**DENİZLİ, DECEMBER 2015**

Serial robot manipulators consist of actuators and reduction gears placed between links in tandem. It is a disadvantage that those actuators and reduction gears should be heavy and big enough to withstand the loads applied to joints. On this thesis, a hybrid snake robot consisting of paralel manipulators serially connected in tandem will be designed so that it eliminates aforementioned disadvantage. System of equations has been obtained for forward kinematics of 3-RPS parallel manipulator to be solved and this system of equations has been solved numerically using Newton's Method. 3D simulation software of snake robot consisting of 10 modules has been coded in C Sharp XNA. After analyzing the workspace of 3-RPS parallel manipulator using Maple, system of equations obtained for inverse kinematics of this manipulator to be solved has been solved numerically using Interval Newton Method. For analyzing to see whether linear actuator selected for actuating the snake robot will withstand or not under the self weight of robot, snake robot modeled in Catia has been imported in Ansys so that the static reaction forces and moments at farthermost sections of linear actuators of first module have been simulated. According to the results obtained, snake robot with 10 modules is not mechanically applicable with standard linear actuators available on market.

**KEYWORDS:** Snake robots, 3-RPS parallel manipulator, hybrid manipulators, redundant robots.

# İÇİNDEKİLER

**Sayfa**

# ŞEKİL LİSTESİ

**Sayfa**

# TABLO LİSTESİ

**Sayfa**

# SEMBOL LİSTESİ

**α :** 3-RPS Paralel Manipülatörün Lineer Aktüatörlerinden Biri ile Sabit Platform Arasındaki Açı

**β :** 3-RPS Paralel Manipülatörün Lineer Aktüatörlerinden Biri ile Sabit Platform Arasındaki Açı

**θ :** 3-RPS Paralel Manipülatörün Lineer Aktüatörlerinden Biri ile Sabit Platform Arasındaki Açı

**ϕ :** Sabit Platform Vektörü ile Hareketli Platform Vektörü Arasındaki Platform Açısı

# ÖNSÖZ

Yüksek lisans eğitimim boyunca bilgi birikimini benden esirgemeyen, tecrübelerini aktarırken gösterdiği olağanüstü çabalarından dolayı çok değerli Hocam, Sayın Prof. Dr. Erdinç Şahin ÇONKUR'a en içten teşekkürlerimi sunarım.

Yine tecrübelerinden ve bilgi birikimlerinden faydalandığım çok değerli Hocalarım, Sayın Doç. Dr. Zekeriya GİRGİN'e, Sayın Doç. Dr. Gürkan ALTAN'a ve Sayın Prof. Dr. Numan Behlül BEKTAŞ'a teşekkürlerimi sunarım.

Beni bu günlere getirirken maddi ve manevi desteklerini esirgemeyen aileme sonsuz teşekkürlerimi sunarım.

# 1. GİRİŞ

Yılanlar, dar geçitlere veya pürüzlü yüzeylere sahip olan zorlu arazi şartlarında bile yüksek hareket kabiliyetine sahip canlılardır (Transeth ve diğ. 2009). Bu hareket yetenekleri sayesinde yılanlar, sürünmekten tepeye çıkabilmeye, kaygan zeminlerde ilerlemekten gövdesini ağacın etrafına dolayarak tırmanabilmeye kadar çok çeşitli görevleri yerine getirebilirler (Matsuno ve Suenaga 2003). Bu hareket kabiliyeti robotlara aktarılarak, yılan gibi görünen ve yılan gibi hareket edebilen robotlar ortaya çıkarılmıştır (Transeth ve diğ. 2009). Yılansı robot olarak adlandırılan bu robotlar, sahip oldukları çok sayıdaki uzuv ve mafsalların yardımıyla belirli bir görevi yerine getirebilmek için şekilden şekle girebilirler (Yamakita ve diğ. 2003). Bu görev; tünel içi yangınlarda insanların yerine yangına müdahalede bulunmak olabilir (Liljeback ve diğ. 2006). Ayrıca, bu robotlarla boru hatlarının iç muayenesi yapılabilir (Fjerdingen ve diğ. 2009). Nükleer tesislerin tehlikeli bölgelerinde çalışmak, yıkık binaların insanların giremeyeceği kadar dar veya tehlikeli bölgelerinde arama-kurtarma faaliyetlerinde bulunmak olabilir. Yılansı robotlar ayrıca; bir ucu sabit bir yüzeye monteli, gövdenin geri kalanı havada hareket edecek şekilde robot manipülatörler olarak da kullanılabilirler (Transeth ve diğ. 2009).

## 1.1 Bir Robot Manipülatörün Yapısı

Birbirine seri olarak bağlanmış uzuvlardan oluşan yapılar, robot manipülatörler olarak adlandırılabilir (Stone 1987). Bu yapılar, robotik kol olarak da betimlenebilirler (Almurib ve diğ. 2012). Robotik manipülatörler, ardışık iki uzuv arasındaki mafsala monte edilen motor-redüktör sistemleriyle veya hidrolik aktüatörlerle tahrik edilebilirler (Kim ve diğ. 2014). Robotik kolların en ucunda, uç elemanı olarak adlandırılan bir alet bulunmaktadır. Bu uç elemanı yardımıyla; yükleme-boşaltma, montaj, kaynak, püskürtmeli boyama gibi işler yerine getirilebilir (Appleton ve Williams 1987). Manipülatörler; seri manipülatörler, paralel manipülatörler ve paralel manipülatörlerin seri olarak birbirine bağlanmasıyla oluşturulan hibrid manipülatörler olmak üzere üç gruba ayrılabilir (Alvarado 2005).

### 1.1.1 Seri Manipülatörler

Bir seri manipülatör, sabit bir uzuv ve bu uzva mafsal veya mafsallar yardımıyla seri olarak bağlı bir veya birden fazla uzuvdan oluşan; bir uç elemanına sahip olan; uzuvların hareketinin aktüatörlerle sağlandığı bir yapıdır (Tsai 1999).

#### 1.1.1.1 Seri Manipülatörlerin Avantajları

Çalışma alanı, seri manipülatörlerde daha geniştir (Daniali 1995).

#### 1.1.1.2 Seri Manipülatörlerin Dezavantajları

Seri manipülatörlerin yapısı gereği her bir uzuvun, kendisinden sonraki uzuv veya uzuvların ağırlıkları ve uç elemanı tarafında taşınan yükün ağırlığını taşımak zorunda olması sebebiyle bu uzuvlar yüksek eğilme momentine maruz kalır (Merlet 2006). Eğer uzuvlar arasında aktüatör olarak motorlar kullanılıyorsa, manipülatörün sabit yüzeye monteli kısmına yakınlaştıkça her bir motor bir öncekinden daha büyük olmalıdır. Çünkü bu motorların; ek olarak, uç elamanının bulunduğu kısma yakınlaştıkça, buralarda bulunan motorların da ağırlığını taşıması gerekmektedir. Dolayısıyla; motorların, manipülatörün sabit yüzeye monteli kısmına yaklaştıkça büyümesi, ağırlığı ve maliyeti daha da artıracaktır (Lenarcic ve diğ. 2013). Ağır yüklerin hareket ettirilmesinde seri manipülatörlerin kullanılması uygun değildir. Gerektiğinde bu yapı güçlendirilebilir; ama bu sefer de manipülatör ağırlaşmış olur ve ataleti artar (Merlet 2006). Bu manipülatörler, paralel manipülatörlere göre daha düşük hızlarda çalışırlar (Daniali 1995). Seri manipülatörlerde konumlama hassasiyeti ağırlık sebebiyle uzuvlarda meydana gelen eğilmelere bağlıdır. Ayrıca, seri manipülatörün mafsallarında motorlarla birlikte redüktör de kullanıldığı için; motor ve redüktör arasındaki boşluklar bu hassasiyeti olumsuz yönde etkileyecektir (Merlet 2006).

#### 1.1.1.3 Seri Manipülatörlerin Kullanım Alanları

Seri manipülatörler, endüstriyel robot olarak sıklıkla kullanılırlar (Daniali 1995).

### 1.1.2 Paralel Manipülatörler

Bir paralel manipülatör; hareketli platformun, sabit bir platforma birbirinden bağımsız en az iki uzuv ile bağlanarak, hareketin aktüatörlerle sağlandığı bir mekanizmadır (Liu ve diğ. 2003).

#### 1.1.2.1 Paralel Manipülatörlerin Avantajları

Paralel manipülatörler yapısı gereği iki platform arasında birden çok uzuva sahip olduğu için, bu uzuvlar taşınması gereken yük miktarını paylaşarak, manipülatöre yüksek ağırlıklı yük taşıyabilme kabiliyeti kazandırır. Ayrıca bu manipülatörlerin ataleti düşüktür (Liu ve diğ. 2003). Çünkü, hareketli ve sabit platform arasındaki uzuvlar, manipülatörün pozisyonuna göre yükü belirli oranlarda paylaştıkları için daha küçük boyutlarda uzuvlar ve daha az güçlü aktüatörler kullanılabilir (Merlet 2006). Bu manipülatörler, seri manipülatörlere göre daha yüksek hızlarda çalışabilirler (Daniali 1995). Çünkü bir mekanizmayı hızlandırmanın bir yolu, o mekanizmanın ağırlığını azaltmaktan geçmektedir (Namiki ve diğ. 2003). Ayrıca, manipülatöre etkiyen yük hareketli ve sabit platform arasındaki uzuvlar tarafından paylaşıldığı için, ağırlık sebebiyle uzuvlarda daha az eğilme meydana gelecektir. Bu yüzden paralel manipülatörlerin konumlama hassasiyetlerinin iyi olduğu söylenebilir (Merlet 2006).

#### 1.1.2.2 Paralel Manipülatörlerin Dezavantajları

Paralel manipülatörlerin çalışma alanı, seri manipülatörlere göre daha dardır (Daniali 1995). Bunun sebebi, manipülatörün yapısı gereği mafsalların hareket

aralığının dar olması ve hareket esnasında manipülatörün uzuvları arasındaki olası çarpışmalardır (Kim ve diğ. 2001).

#### 1.1.2.3 Paralel Manipülatörlerin Kullanım Alanları

Paralel manipülatörler; elektrik üretmek amacıyla güneşi takip eden sistemlerin mekanizması olarak kullanılabilir (Shyam ve Ghosal 2014). Tıbbi alanda kalp masajı yapmak için de kullanılabilir (Li ve Xu 2007). Ayrıca bu manipülatörler uçuş simülatörlerinde, tekerlek test makinelerinde, yüksek hızlı takım tezgahlarında veya ağır yüklerin yüksek ivmeli bir şekilde hareket ettirilmek istendiği herhangi bir işte kullanılabilirler (Parasuraman ve Liang 2010).

### 1.1.3 Hibrid Manipülatörler

Hibrid manipülatörler; paralel manipülatörlerin birbirine seri olarak bağlanmasıyla oluşmuş manipülatörlerdir. Bu sayede seri manipülatörlerin daha geniş çalışma alanı gibi avantajlarından ve paralel manipülatörlerin daha rijit yapısı gibi avantajlarından aynı anda faydalanılmış olunur (Alvarado 2005).

## 1.2 Tezin Amacı

Robot manipülatörlerin bazılarında; ardışık iki uzuv arasına yerleştirilen aktüatör ve aktüatöre bağlı redüktör; tasarım basitliği ve kontrol kolaylığı sağladığı halde, bu aktüatörün ve redüktörün, mafsallara binen yükü taşıyabilecek kadar kapasiteli olabilmesi için ağır olması gerekliliği, dezavantaj olarak görülmektedir (Kim ve diğ. 2014). Bu yüzden bu tezde, paralel manipülatörlerin seri olarak bağlanmasıyla oluşturulmuş hibrid bir yılansı robot tasarımı yapılarak bu dezavantajların önüne geçilmesi planlanmaktadır.

# 2. YILANSI ROBOTLAR

## 2.1 Giriş

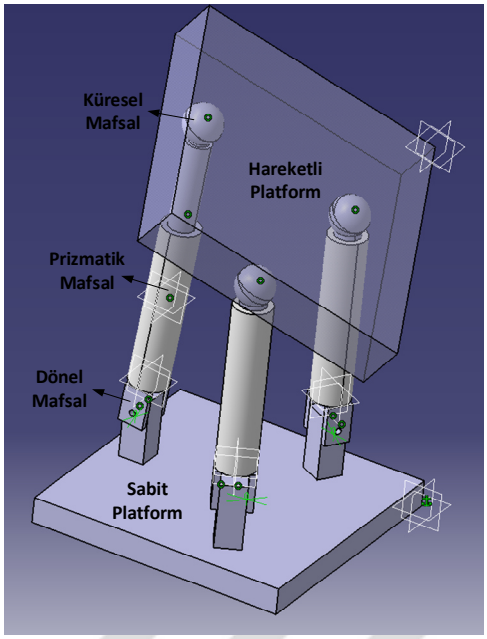
Bu bölümde yılansı robot kavramının anlaşılabilmesi için; mekanizmaların serbestlik derecesi, gereğinden çok serbestlik dereceli manipülatörler ve gereğinden çok fazla (hiper) serbestlik dereceli manipülatörler hakkında bilgi verilecektir. Ayrıca, tezde kullanılacak 3-RPS paralel manipülatörün yapısı tanıtılacak ve bu manipülatörün serbestlik derecesinin hesaplanma yöntemi belirtilecektir. Son olarak, yılansı robotlar hakkında literatürde yapılmış çalışmalara değinilecektir.

## 2.2 Mekanizmaların Serbestlik Derecesi

Bir mekanizmanın serbestlik derecesi; o mekanizmanın, uzaydaki noktasal ve açısal konumunu belirleyebilmek için gerekli olan bağımsız parametrelerin sayısıdır (Liu ve diğ. 2003).

## 2.3 3-RPS (Revolute-Prismatic-Spherical) Paralel Manipülatör

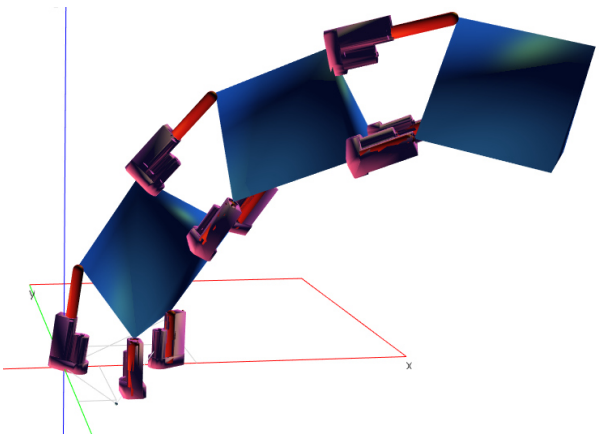
3-RPS paralel manipülatörler; sabit ve hareketli birer platforma sahiptirler. Bu iki platform arasında, her biri bağımsız birer aktüatör olarak kullanılacak prizmatik mafsallara sahip üç adet uzuv bulunur. Bu uzuvların her biri sabit platforma dönel mafsal ile, hareketli platforma ise küresel mafsal ile bağlanmıştır. Sabit platformdan başlayarak sırasıyla dönel (revolute), prizmatik (prismatic) ve küresel (spherical) mafsallarla hareketli platforma üç adet uzuv bağlandığı için bu manipülatörlere ingilizce revolute, prismatic ve spherical kelimelerinin baş harfleri verilerek kısaca 3-RPS paralel manipülatörler denir (Gallardo ve diğ. 2011). Şekil 2.1'de 3-RPS paralel manipülatör görülmektedir. Bu tezde 3-RPS manipülatörler birbirine seri olarak bağlanarak yılansı bir robot oluşturulacaktır.




**Şekil 2.1:** 3-RPS paralel manipülatör.

### 2.3.1 3-RPS Paralel Manipülatörlerin Birbirine Seri Olarak Bağlanması

Bir modüler robot, birbirine eş birden çok bölümden oluşur. Her bölüm, bir diğer bölümden bağımsız olarak mafsallarını hareket ettirebilir (Gallardo ve diğ. 2011). Birden çok 3-RPS paralel manipülatörün birbirine seri olarak bağlanarak oluşturduğu yapı, robotik kol mekanizması olarak kullanılabilir (Lu ve Leinonen 2005). Şekil 2.2'de, birden fazla 3-RPS paralel manipülatörlerin seri olarak birbirine bağlanmasıyla elde edilmiş modüler bir robot mekanizması görülmektedir.



**Şekil 2.2:** 3-RPS paralel manipülatörlerin seri olarak bağlanmasından oluşan modüler robot.

### 2.3.2 3-RPS Paralel Manipülatörün Serbestlik Derecesinin Hesaplanması

Uzaysal paralel manipülatörlerin serbestlik derecelerini hesaplamak için Kutz-bach denkleminden faydalanılabilir.

Kutz-bach Denklemi:

$$F=\lambda * \left(n-j-1\right) + \sum_i f_i \tag{2.1}$$

Bu denklemde; F simgesi, uzaysal paralel manipülatörün serbestlik derecesini; n simgesi, paralel manipülatördeki toplam uzuv sayısını; j simgesi, paralel manipülatördeki toplam mafsal sayısını; $f_i$, "i" nolu mafsalın serbestlik derecesini ifade etmektedir. Ayrıca λ simgesinin değeri uzaysal mekanizmalarda 6'ya eşittir.

Mekanizmada toplam 8 adet uzuv bulunmaktadır. Bunlar 1 adet sabit alt platform, 1 adet hareketli platform, 3 adet aktüatör dış uzvu ve 3 adet aktüatör iç uzvudur. Ayrıca, 3 tane küresel mafsal, 3 tane prizmatik mafsal ve 3 tane dönel mafsal olmak üzere 9 tane mafsal vardır (Rao ve Rao 2013). $f_1$'i dönel mafsal olarak kabul edersek dönel mafsalların her birinin serbestlik derecesi 1'dir. $f_2$'yi prizmatik mafsal olarak kabul edersek prizmatik mafsalların her birinin serbestlik derecesi 1'dir. $f_3$'ü küresel mafsal olarak kabul edersek küresel mafsalların her birinin serbestlik derecesi 3'tür (Lu ve Leinonen 2005). Bu değerleri Kutz-bach denkleminde yerine koyarsak 3-RPS paralel manipülatörün serbestlik derecesi (2. 2) eşitliğindeki gibi bulunur (Rao ve Rao 2013).

$$F = 6*\left(8-9-1\right) + 3*(1+1+3) = -12 + 15 = 3 \tag{2.2}$$

Bu sayede birbirinden bağımsız şekilde ayrı ayrı hareket ettirilebilen üç adet prizmatik mafsal sebebiyle manipülatör üç serbestlik derecesine sahip olur (Gallardo ve diğ. 2009).

Eğer, iki tane 3-RPS manipülatörün seri olarak birbirine bağlandığı mekanizmanın serbestlik derecesi hesaplanmak istenirse, mekanizmada toplamda 15 adet uzuv, 18 adet mafsal bulunmaktadır. Bu değerler yine Kutz-bach denkleminde yerine konursa, iki adet 3-RPS paralel manipülatörün birbirine seri olarak bağlandığı

mekanizmanın serbestlik derecesi; (2. 3) eşitliğindeki gibi bulunur (Lu ve Leinonen 2005).

$$F = 6*(15-18-1) + 6*(1+1+3) = -24 + 30 = 6 \quad (2.3)$$

## 2.4 Gereğinden Çok Serbestlik Dereceli Manipülatörler

Bir manipülatörün bir cismi, uzaydaki tüm eksenlerde noktasal ve açısal konumlandırabilmesi için, altı tane serbestlik derecesine sahip olması gerekmektedir. Dolayısıyla yedi veya daha fazla serbestlik derecesine sahip olan manipülatörler, gereğinden çok serbestlik dereceli manipülatörler olarak adlandırılmaktadır (Chirikjian ve Burdick 1994). Bu manipülatörlerin ters kinematikleri çözüldüğünde, sonsuz tane çözümün olduğu görülür (Çonkur 1997).

## 2.5 Gereğinden Çok Fazla (Hiper) Serbestlik Dereceli Manipülatörler

Eğer bir manipülatör çok fazla serbestlik derecesine sahipse; bu manipülatör, gereğinden çok fazla serbestlik dereceli manipülatör olarak adlandırılır (Chirikjian ve Burdick 1994). Bu manipülatörler bu özelliğiyle doğadaki yılanlar gibi hareket edebilir (Gallardo ve diğ. 2011). Bu kadar fazla serbestlik derecesi bu manipülatörlere engellerden kaçınma kabiliyeti kazandırır (Gallardo ve diğ. 2009).

## 2.6 3-RPS Yılansı Robotlar Hakkında Literatür Özeti

Zorlu çevre koşullarında robotların daha fazla serbestlik derecesine ihtiyacı vardır. Bu ihtiyacı karşılayabilmek için, biyolojik yılan hareketlerinden ilham alınarak tasarlanan yılansı robotlar kullanılabilir (Liljebäck ve diğ. 2012).

İki ya da daha fazla paralel manipülatör, seri olarak birbirine bağlanabilir. Bu yolla, gereğinden çok serbestlik dereceli modüler robotik bir sistem oluşturulabilir. Burdan yola çıkarak, 3-RPS paralel bir manipülatörün ileri kinematiğinin çözülebilmesi için oluşturulan denklemlerin Sylvester dyalitic eliminasyon yöntemiyle çözüm yolu gösterilmiştir. Önerilen metod kullanılarak; toplamda 18

serbestlik dereceli, birbirine seri olarak bağlanmış 6 adet 3-RPS manipülatörün ileri kinematiği analiz edilmiştir (Gallardo-Alvarado ve diğ. 2008).

İnsan cerrahisinde tıbbi sonda olarak kullanılmak üzere, doğadan ilham alınarak yılansı bir robot tasarlanmıştır. Üç adet 3-RPS paralel manipülatörün seri bir şekilde birbirine bağlanarak oluşturulduğu yapı, yılansı şekilde hareket edebilmektedir. Bu yapının toplam serbestlik derecesi dokuzdur. Mekanizmanın, uç elemanının noktasal ve açısal konumunu bulabilmek amacıyla homojen transformasyon yöntemiyle ileri kinematiğin çözülebileceği söylenmiştir. Sistemin ters kinematiğinin hesabı sonucu sonsuz sayıda çözümün çıkacağından bahsedildikten sonra bu sorunun backbone curve hesaplama metoduyla giderilebileceği belirtilerek ters kinematiğin hesaplanabilmesi için izlenecek yol adım adım anlatılmıştır. Robotun dinamik analiziyle birlikte ileri ve ters kinematiğinin hesaplanmasının manuel olarak mümkün olmadığından bahsedilmiş, lineer aktüatörler kullanılarak tasarlanan robot Matlab/Simulink ortamında modellenmiştir (Mintenbeck ve Estana 2010).

3-RPS paralel manipülatörlerde, tahrik edilen aktüatörler ile bu tahrik sonucu hareket eden hareketli platform arasında bir bağlantı kurabilmek ve mekanizmanın noktasal ve açısal konumunu hesaplayabilmek, problem olarak görülmektedir. Bu bağlamda öncelikle, iki adet uzaysal 3-RPS paralel manipülatörün seri olarak birbirine bağlanmasından oluşan ve adına uzaysal 2(3-RPS) manipülatör denilen mekanizmanın geometrisi açıklandıktan sonra bu mekanizmanın serbestlik derecesi Kutzbach Grubler denklemi yardımıyla hesaplanmıştır. Ardından, n tane 3-RPS paralel manipülatörün seri olarak birbirine bağlandığı mekanizmanın noktasal ve açısal konumunu hesaplamak için gerekli genel bir denklem türetilmiştir. Ancak analitik yaklaşımın kolay olmayışı ve bilgisayar yazılımında derlenmesi zor olacağından iki tane 3-RPS paralel manipülatörün seri olarak birbirine bağlanmasından oluşan mekanizmanın simülasyonu SolidWorks yazılımında gerçekleştirilmiştir. SolidWorks'te mekanizmanın simülasyonunun nasıl oluşturulacağı adım adım anlatılmıştır. Sonuç olarak yapılan çalışmanın, kol mekanizması tasarlayabilmek adına yararlı olduğu görülmüştür (Lu ve Leinonen 2005).

Birden fazla paralel manipülatörün seri bir biçimde birbirine bağlanarak oluşturulan yapı, yılansı hareketlere sahip olacağı için engellere çarpmadan ilerleyebilir. Bu manipülatörü engellere çarptırmadan hareket ettirebilmek için manipülatörün ters kinematiğinin çözülmesi amaçlanmıştır. Ters kinematik, Suthakorn metodu kullanarak çözülmüştür. Engellere çarpmadan geçmek için önerilen algoritma, yirmi tane 3-RPS paralel manipülatörlü sistem üzerinde denenmiştir. Elde edilen sonuçlar, Genetik Algoritma Metodu kullanılarak bulunan sonuçlarla karşılaştırılmış olup, bunun sonucunda önerilen metodun Genetik Algoritma metoduna göre daha hızlı ve daha doğru sonuçlar verdiği görülmüştür (Motahari ve diğ. 2012).

Bir başka çalışmada, 3-RPS paralel manipülatörün geometrik yapısı açıklanıp Grübler-Kutzbach denklemiyle mekanizmanın serbestlik derecesi hesaplandıktan sonra ileri kinematik denklemleri türetilmiştir. Nonlineer denklem sistemlerinin çözümünde kullanılan Newton-Kantorovich sayısal yöntemi, altı denklemden oluşan ileri kinematik denklem sisteminin çözümünde kullanılmıştır. CAD modeli oluşturulan 3-RPS manipülatörün MATLAB/Simulink ortamında simülasyonu yapılarak elde edilen sonuçlar Newton-Kantorovich sayısal yöntemi ile elde edilen sonuçlarla karşılaştırılmış; sonuçların birbirine çok yakın çıktığı görülmüştür. Ayrıca, Lukanin yöntemi ve ileri kinematik denklemleri yardımıyla, 3-RPS manipülatörün çalışma alanı analizi yapılmıştır (Rad ve diğ. 2010).

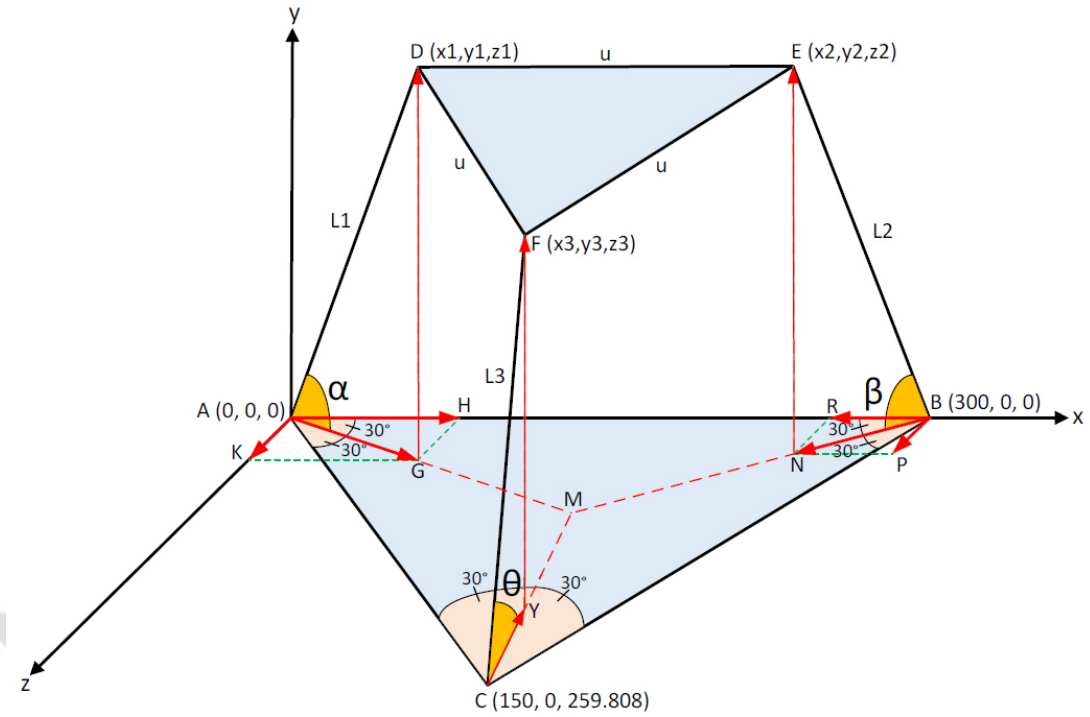
# 3. 3-RPS MANİPÜLATÖRLERDEN OLUŞAN YILANSI ROBOTUN İLERİ KİNEMATİĞİ

## 3.1 Giriş

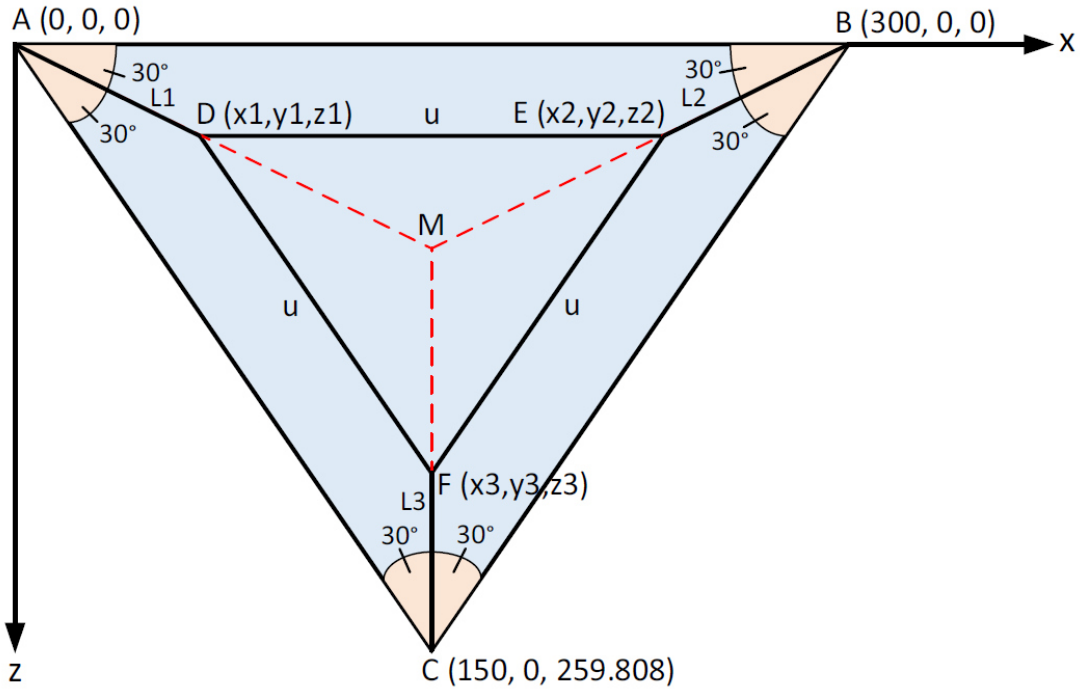
Bu bölümde öncelikle 3-RPS manipülatörün geometrisi incelenecektir. Ardından 3-RPS manipülatörü tahrik eden üç adet birbirinden bağımsız lineer aktüatörün değişen strok uzunluklarına göre hareketli platformun ağırlık merkezinin referans koordinat sistemine göre uzaydaki konumunu bulabilmek için gerekli olan denklem sistemi oluşturulacaktır. Bulunan denklem sistemi sayısal olarak çözüldükten sonra çözüm yöntemi C Sharp'a aktarılıp yılansı robotun her bir modülünün lineer aktüatör strok uzunluğu kontrol edilerek üç boyutlu simülasyonu yapılacaktır.

## 3.2 3-RPS Paralel Manipülatörün Geometrik Olarak İncelenmesi

3-RPS paralel manipülatörü geometrik olarak incelemek için Şekil 3.1 ve Şekil 3.2'den yararlanılacaktır. Bu manipülatörün sabit ve hareketli olmak üzere iki adet platformu bulunmaktadır. Bu platformlardan sabit olanı A, B ve C noktalarının birleşerek oluşturduğu eşkenar üçgen olan platformdur ve bu platform xz-düzleminde yer almaktadır. Hareketli platform ise yine bir eşkenar üçgendir ve bu üçgen D, E ve F noktalarının birleştirilmesiyle oluşturulur. Bu iki platformu her iki üçgenin köşe noktalarından birbirine bağlayan lineer aktüatörleri; L1 uzunluğundaki [AD] doğru parçası, L2 uzunluğundaki [BE] doğru parçası ve L3 uzunluğundaki [CF] doğru parçası temsil etmektedir. Bu doğru parçaları ile sabit platform arasında sırasıyla $\alpha$, $\beta$ ve $\theta$ açıları bulunmaktadır. Bu doğru parçalarının sabit platform üzerindeki izdüşümlerinin bu platform üzerindeki uzantıları, sabit platformun çevrel çemberinin merkezi olan M noktasından geçmektedir. Sabit platformun her bir kenarının 300 mm olduğu kabul edilmiştir. Hareketli platformun ise her bir kenarı "u" mm uzunluğundadır.




**Şekil 3.1:** 3-RPS manipülatörün geometrik yapısı.




**Şekil 3.2:** 3-RPS manipülatörün üst görünüşü.

## 3.3 İleri Kinematiğin Çözümü İçin Denklem Sisteminin Oluşturulması

3-RPS manipülatörü tahrik eden üç adet birbirinden bağımsız lineer aktüatörün bilinen ve değişen strok uzunluklarına göre hareketli platformun ağırlık merkezinin referans koordinat sistemine göre uzaydaki konumunu bulabilmek için öncelikle hareketli platformun köşe noktalarındaki D(x1,y1,z1), E(x2,y2,z2) ve F(x3,y3,z3) (bkz. Şekil 3.1) koordinatlarının bulunması gereklidir. Bu koordinatların bulunabilmesi için ise; hareketli ve sabit platformları her iki üçgenin köşe noktalarından birbirine bağlayan lineer aktüatörleri temsil eden ve uzunlukları bilinen L1 uzunluğundaki [AD] doğru parçası, L2 uzunluğundaki [BE] doğru parçası ve L3 uzunluğundaki [CF] doğru ile sabit platform arasında bulunan; sırasıyla α, β ve θ açılarının değerlerinin bulunması gerekmektedir. Bu üç bilinmeyen açıyı bulabilmek için bu açıları içeren en az üç denklemden oluşan bir denklem sistemi elde edilmelidir. Bu denklem sistemini elde edebilmek için hareketli platformun geometrik sınırlamalarından yararlanılabilir. Bu sınırlamalar; hareketli platformun DF, EF ve DE uzunluklarının (bkz. Şekil 3.1) platformun hareketinden bağımsız olarak daima sabit kalmasından kaynaklanır. Her bir uzunluğu bulabilmek için uzayda iki nokta arasındaki uzaklığı veren formül kullanılarak;

D ve F noktaları arasındaki uzaklık,

$$(x3 - x1)^2 + (y3 - y1)^2 + (z3 - z1)^2 = u^2 \tag{3.1}$$

E ve F noktaları arasındaki uzaklık,

$$(x3 - x2)^2 + (y3 - y2)^2 + (z3 - z2)^2 = u^2 \tag{3.2}$$

D ve E noktaları arasındaki uzaklık,

$$(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2 = u^2 \tag{3.3}$$

olarak bulunabilir. Eşitlikler (3.1), (3.2) ve (3.3)'ten de görülebileceği gibi x1, x2, x3, y1, y2, y3, z1, z2, z3 olmak üzere toplam dokuz tane bilinmeyen vardır. Bu bilinmeyenler; α, β ve θ açıları cinsinden yazılabilir. Bu sayede üç bilinmeyenli üç adet denklem elde edilmiş olur. D noktasının sabit platform üzerindeki izdüşümü olan $\overrightarrow{AG}$ vektörünün değeri L1*cos (α°) 'dır. Bu değerin cos(30°) ile çarpımı D noktasının x koordinatı olan x1'i; sin(30°) ile çarpımı ise D noktasının z koordinatı olan z1'i verir. D noktasının y koordinatı olan y1'in değeri, L1*sin(α°) ile

bulunabilir. E noktasının sabit platform üzerindeki izdüşümü olan $\overrightarrow{BN}$ vektörünün değeri L2*cos(β°)'dir. Bu değerin cos(30°) ile çarpılıp 300'den çıkartılmasıyla E noktasının x koordinatı olan x2, sin(30°) ile çarpımıyla E noktasının z koordinatı olan z2 elde edilir. E noktasının y koordinatı olan y2'nin değeri ise L2*sin(β°)'dir. F noktasının x koordinatı olan x3'ün değeri 150'dir. Çünkü F noktasının sabit platform üzerindeki izdüşümünün uzantısının x eksenini kesim noktası daima 150'dir. F noktasının y koordinatı olan y3'ün değeri L3*sin(θ°)'dir. F noktasının sabit platform üzerindeki izdüşümü olan $\overrightarrow{CY}$ vektörünün değeri L3*cos(θ°)'dir. Bu değer, C noktasının z koordinatının değeri olan 259.808'den çıkarılırsa; F noktasının z koordinatı olan z3 elde edilir.

$$x1 = L1*\cos(\alpha°)*\cos(30°) \tag{3.4}$$

$$y1 = L1*\sin(\alpha°) \tag{3.5}$$

$$z1 = L1*\cos(\alpha°)*\sin(30°) \tag{3.6}$$

$$x2 = 300 - L2*\cos(\beta°)*\cos(30°) \tag{3.7}$$

$$y2 = L2*\sin(\beta°) \tag{3.8}$$

$$z2 = L2*\cos(\beta°)*\sin(30°) \tag{3.9}$$

$$x3 = 150 \tag{3.10}$$

$$y3 = L3*\sin(\theta°) \tag{3.11}$$

$$z3 = 259.808 - L3*\cos(\theta°) \tag{3.12}$$

Hareketli platformun kenar uzunluğu 300 mm olarak kabul edilip, x1, y1, z1, x2, y2, z2, x3, y3, z3 koordinatlarının α, β ve θ açıları cinsinden bulunan değerleri; (3.1), (3.2) ve (3.3) numaralı eşitliklerde yerine koyulursa;

$$[\ 150 - L1*\cos(\alpha°)*\cos(30°)\ ]^2 + [\ L3*\sin(\theta°) - L1*\sin(\alpha°)\ ]^2 + [\ 259.808 - L3*\cos(\theta°) - L1*\cos(\alpha°)*\sin(30°)\ ]^2 = 300^2 \tag{3.13}$$

$$[\ 150 - (\ 300 - L2*\cos(\beta°)*\cos(30°)\ )\ ]^2 + [\ L3*\sin(\theta°) - L2*\sin(\beta°)\ ]^2 + [\ 259.808 - L3*\cos(\theta°) - L2*\cos(\beta°)*\sin(30°)\ ]^2 = 300^2 \tag{3.14}$$

$$[\ 300 - L2*\cos(\beta°)*\cos(30°) - L1*\cos(\alpha°)*\cos(30°)\ ]^2 + [\ L2*\sin(\beta°) - L1*\sin(\alpha°)\ ]^2 + [L2*\cos(\beta°)*\sin(30°) - L1*\cos(\alpha°)*\sin(30°)\ ]^2 = 300^2 \tag{3.15}$$

(3.13), (3.14) ve (3.15) eşitliklerinde görülen üç denklemden oluşan bir denklem sistemi elde edilir. Sisteme, uzunlukları bilinen L1, L2 ve L3 lineer aktüatör boyları

girildiğinde; bilinmeyen α, β ve θ açılarının bulunabilmesi için denklem sisteminin çözülmesi gereklidir. Analitik çözüm yolu bulunamadığı için bu denklem sistemi sayısal olarak çözülecektir. Öncelikle (3.13), (3.14) ve (3.15) numaralı eşitliklerin sağ tarafındaki $300^2$, denklemlerin sol tarafına geçirilip, bu denklemlere (3.16), (3.17) ve (3.18) numaralı denklemlerde görüldüğü üzere birer isim verilir.

$$\textbf{denklem1} = [\ 150 - L1*\cos(\alpha°)*\cos(30°)\ ]^2 + [\ L3*\sin(\theta°) - L1*\sin(\alpha°)\ ]^2 + [259.808 - L3*\cos(\theta°) - L1*\cos(\alpha°)*\sin(30°)\ ]^2 - 300^2 \quad (3.16)$$

$$\textbf{denklem2} = [\ 150 - (\ 300 - L2*\cos(\beta°)*\cos(30°)\ )\ ]^2 + [\ L3*\sin(\theta°) - L2*\sin(\beta°)\ ]^2 + [\ 259.808 - L3*\cos(\theta°) - L2*\cos(\beta°)*\sin(30°)\ ]^2 - 300^2 \quad (3.17)$$

$$\textbf{denklem3} = [\ 300 - L2*\cos(\beta°)*\cos(30°) - L1*\cos(\alpha°)*\cos(30°)\ ]^2 + [\ L2*\sin(\beta°) - L1*\sin(\alpha°)\ ]^2 + [L2*\cos(\beta°)*\sin(30°) - L1*\cos(\alpha°)*\sin(30°)\ ]^2 - 300^2 \quad (3.18)$$

Denklem sistemlerini sayısal olarak çözebilmek için, (3.19) numaralı denklemde gösterilen Newton'un Metodu kullanılabilir.

$$\begin{bmatrix} \alpha_{(i+1)} \\ \beta_{(i+1)} \\ \theta_{(i+1)} \end{bmatrix} = \begin{bmatrix} \alpha_{(i)} \\ \beta_{(i)} \\ \theta_{(i)} \end{bmatrix} - (J)^{-1} \cdot \begin{bmatrix} denklem1_{(i)} \\ denklem2_{(i)} \\ denklem3_{(i)} \end{bmatrix} \quad (3.19)$$

(3.19) numaralı denklemde (J) ile gösterilen, Jacobian Matrisi'dir. Jacobian Matrisi, (3.20) numaralı denklemde gösterilmiştir.

$$(J) = \begin{pmatrix} \frac{d}{d\alpha} denklem1 & \frac{d}{d\beta} denklem1 & \frac{d}{d\theta} denklem1 \\ \frac{d}{d\alpha} denklem2 & \frac{d}{d\beta} denklem2 & \frac{d}{d\theta} denklem2 \\ \frac{d}{d\alpha} denklem3 & \frac{d}{d\beta} denklem3 & \frac{d}{d\theta} denklem3 \end{pmatrix} \quad (3.20)$$

(3.19) numaralı denklemin sağ tarafında (i) alt indisi ile gösterilen $\alpha_{(i)}$, $\beta_{(i)}$ ve $\theta_{(i)}$ açıları, çözümle ilgili yapılan ilk tahminlerdir. Bu tahminlere bağlı olarak (3.19) numaralı denklemin sol tarafında (i+1) alt indi ile gösterilen $\alpha_{(i+1)}$, $\beta_{(i+1)}$ ve $\theta_{(i+1)}$ açıları elde edilir. Bu açılar yapılan ilk tahminlere göre elde edilen yeni değerlerdir. Bu yeni değerler yeniden (3.19) numaralı denklemin sağ tarafına koyarak aynı

işlemler, (3.19) numaralı denklemin sol tarafındaki sonuçlar sabit değerlere yakınsayana kadar tekrar edilir. Bu sabit değerler, denklem sisteminin çözümü olan α, β ve θ açılarını verir.

Newton'un, denklem sistemlerinin çözümünde kullanılan yöntemi öncelikle Maple'da kodlanmış; ardından, bulunan sembolik sonuçlar C Sharp diline Maple yardımıyla çevrilerek C Sharp ortamında metod test edilmiştir.

### 3.3.1 Newton Metodu'nun Maple Kullanılarak Kodlanması

Aşağıda Maple dilinde yazılan kodlar kullanılarak Newton Metodu gerçekleştirilebilir.

```
> restart:
> with(LinearAlgebra):
> with(CodeGeneration):
>
> denklem1:=(150-L1*cos(α)*cos(Pi/6))^2+(L3*sin(θ)-L1*sin(α))^2+(259.808-
L3*cos(θ)-  L1*cos(α)*sin(Pi/6))^2-90000:
> denklem2:=(150-300+L2*cos(β)*cos(Pi/6))^2+(L3*sin(θ)-L2*sin(β))^2+(259.808-
L3*cos(θ)-L2*cos(β)*sin(Pi/6))^2-90000:
> denklem3:=(300-L2*cos(β)*cos(Pi/6)-L1*cos(α)*cos(Pi/6))^2+(L2*sin(β)-
L1*sin(α))^2+(L2*cos(β)*sin(Pi/6)-L1*cos(α)*sin(Pi/6))^2-90000:
>
> matris11:=diff(denklem1, α):
> matris12:=diff(denklem1, β):
> matris13:=diff(denklem1, θ):
> matris21:=diff(denklem2, α):
> matris22:=diff(denklem2, β):
> matris23:=diff(denklem2, θ):
> matris31:=diff(denklem3, α):
> matris32:=diff(denklem3, β):
> matris33:=diff(denklem3, θ):
```

```
> m11:=evalf(matris11):
> m12:=evalf(matris12):
> m13:=evalf(matris13):
> m21:=evalf(matris21):
> m22:=evalf(matris22):
> m23:=evalf(matris23):
> m31:=evalf(matris31):
> m32:=evalf(matris32):
> m33:=evalf(matris33):
> jacobianMatrisi:=Matrix(3,3,[m11,m12,m13,m21,m22,m23,m31,m32,m33]):
>
> sayisalDenklem1:=evalf(denklem1):
> sayisalDenklem2:=evalf(denklem2):
> sayisalDenklem3:=evalf(denklem3):
>
>denklemMatrisi:=Matrix(3,1,[sayisalDenklem1,sayisalDenklem2,sayisalDenklem3]
):
> matrisCarpim:=MatrixInverse(jacobianMatrisi). denklemMatrisi:
> forCsharp:=Matrix(3,1,[ α, β, θ])-matrisCarpim:
>
> CSharp(forCsharp[1,1],resultname="BirinciSatirBirinciSutun"):
> CSharp(forCsharp[2,1],resultname="IkıncıSatirBirinciSutun"):
> CSharp(forCsharp[3,1],resultname="UcuncuSatirBirinciSutun"):
```

Yukarıda görülen kodlarda with(LinearAlgebra) kodu, Maple'da matris hesaplamaları yapmak; with(CodeGeneration) kodu ise Maple ortamında yazılan kodu C sharp diline devirmek için kullanılır. Ardından çözülmesi istenen denklem1, denklem2 ve denklem3 yazılır. Sonrasında diff( ) komutu yardımıyla parantez içine hangi denklemin hangi değişkene göre türev alınacağı yazılır. Bu kod sonucu bulunan değerler Jacobian Matrisini oluşturmak üzere kullanılacağı için, denklem1, denklem2 ve denklem3'ün ayrı ayrı α, β ve θ açılarına göre türevi alınır. İstenirse evalf( ) komutu kullanılarak, denklemlerin içerisindeki kesirli ifadeler ondalıklı hale getirilebilir. Bulunan değerler ile Jacobian Matrisi oluşturduktan sonra denklem1, denklem2 ve denklem3 kullanılarak denklemMatrisi oluşturulur. Jacobian Matrisinin

tersi alınıp denklemMatrisi ile çarpılarak elde edilen değer; α, β ve θ açılarının ilk tahmin değerlerinin girilebilmesi için oluşturulan matristen çıkarılır. Elde edilen sonuçlar; ilk tahmin açı değerlerine göre yeni açı değerlerini sembolik olarak verir. Bu sonuçların birinci satır birinci sütununda bulunan yeni α, ikinci satır birinci sütununda bulunan yeni β ve üçüncü satır birinci sütununda bulunan yeni θ değerlerini veren sembolik ifadeleri, ayrı ayrı C Sharp koduna dönüştürülür.

### 3.3.2 Newton Metodu'nun C Sharp ve Maple Kullanılarak Test Edilmesi

Bu bölümde Newton Metodu, C Sharp ve Maple kullanılarak test edilecektir. Aşağıda, C Sharp dilinde yazılan kodlar görülmektedir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
namespace YalcinBulutYuksekLisansTezi
{
   public partial class Form1 : Form
   {
      public Stopwatch sw = new Stopwatch();
      double α = 1.483;
      double β = 1. 483;
      double θ= 1. 483;
      int L1 = 250;
      int L2 = 200;
      int L3 = 150;
```

```
        public Form1()
        {
            InitializeComponent();
        }
        private void newtonMethodTestButton_Click(object sender, EventArgs e)
        {
            sw.Start();
            for (int i = 0; i < 4; i++)
            {
                α = α + (-0.5196152424e3 * Math.Sin(β) + 0.1000000000e-8 *
Math.Sin(β) * L2 *…
                β = β - (-0.5196152424e3 * Math.Sin(α) + 0.1000000001e1 *
Math.Sin(α) * L2 *…
                θ = θ + (-0.5196152424e3 * Math.Sin(α) + 0.1000000001e1 *
Math.Sin(α) * L2 *…
                label1.Text = (α * (180 / Math.PI)).ToString();
                label2.Text = (β* (180 / Math.PI)).ToString();
                label3.Text = (θ* (180 / Math.PI)).ToString();
            }
            string cozumSuresi = sw.ElapsedMilliseconds.ToString();
            label4.Text = cozumSuresi;
            sw.Reset();
        }
    }
}
```

Yukarıdaki C Sharp kodunda öncelikle α, β ve θ açılarına ilk tahmin değeri olarak 1.483 radyan (84.97°) verildi. Lineer aktüatörün kapalı haldeki boyu 150, maksimum stroklu durumdaki boyu ise 300 kabul edildi. Bu aralığa göre L1, L2 ve L3 uzunlukları girildikten sonra bir butona basıldığında bu uzunluklara göre α, β ve θ açılarının hesap yapılabilmesi için newtonMethodTestButton'u oluşturuldu. Bu butonun içine bu hesabın ne kadar sürede yapıldığını görebilmek amacıyla Stopwatch kullanabilmek için ilgili kodlar yazıldı. For döngüsü içerisine yeni α, β ve θ açılarını veren sembolik denklemlerin uzunluğundan dolayı, kod işleyişi hakkında

fikir verilmesi amacıyla yalnızca giriş kısımları yazıldı. Dört iterasyon sonrası virgülden sonra on iki basamaklı değerlere yakınsayan α, β ve θ açı değerleri, radyandan açıya dönüştürülerek label'lara yazdırıldı. Her iterasyon sonucu α, β ve θ açıları ve toplam hesap süresi Tablo 3.1'de gösterilmiştir. Tablo 3.1'den de görüldüğü üzere dördüncü iterasyonla birlikte α, β ve θ açıları, virgülden sonra on iki basamaklı değerlere yakınsamıştır. Bulunan sonuçların (3.16), (3.17) ve (3.18) numaralı sırasıyla denklem1, denklem2 ve denklem3'te yerine koyulmasıyla, çözüm sayısal yolla elde edildiği için bu üç denklemin ayrı ayrı sıfıra çok yakın çıkması beklenir. Bu yolla sonuçların doğru olup olmadığı test edilebilir. Bu test, Maple yardımıyla, C Sharpta iterasyon sonucunda elde edilen α, β ve θ açıları, lineer aktüatör uzunlukları ve denklem1, denklem2 ve denklem3 kodlanarak yapılabilir. İlgili Maple kodları aşağıdadır.

**Tablo 3.1:** C sharpta iterasyon sonrası elde edilen değerler.

| **İterasyon Sayısı** | **α°** | **β°** | **θ°** | **Toplam Hesap Süresi (milisaniye)** |
|---|---|---|---|---|
| 1 | 87.913649607217 | 91.766369026294 | 86.197459576316 | 2 |
| 2 | 87.732567741430 | 91.423773002692 | 86.187370998886 | |
| 3 | 87.731995029286 | 91.423003083976 | 86.187370066377 | |
| 4 | 87.731995024907 | 91.423003080113 | 86.187370066377 | |
| 5 | 87.731995024907 | 91.423003080113 | 86.187370066377 | |

```
> restart:
> with(LinearAlgebra):
> L1:=250:
> L2:=200:
> L3:=150:
> α:=87.731995024907*2*Pi/(360):
> β:=91.423003080113*2*Pi/(360):
> θ:=86.187370066377*2*Pi/(360):
> denklem1:=(150-L1*cos(α)*cos(Pi/6))^2+(L3*sin(θ)-L1*sin(α))^2+(259.808-
L3*cos(θ)-  L1*cos(α)*sin(Pi/6))^2-90000:
```

```
> denklem2:=(150-300+L2*cos(β)*cos(Pi/6))^2+(L3*sin(θ)-L2*sin(β))^2+(259.808-
L3*cos(θ)-L2*cos(β)*sin(Pi/6))^2-90000:
> denklem3:=(300-L2*cos(β)*cos(Pi/6)-L1*cos(α)*cos(Pi/6))^2+(L2*sin(β)-
L1*sin(α))^2+(L2*cos(β)*sin(Pi/6)-L1*cos(α)*sin(Pi/6))^2-90000:
> evalf(denklem1):
> evalf(denklem2):
> evalf(denklem3):
```

Kodlar çalıştırıldığında;

$$\text{denklem1} = -0.00013 \tag{3.21}$$

$$\text{denklem2} = -0.00002 \tag{3.22}$$

$$\text{denklem3} = -0.00007 \tag{3.23}$$

olarak bulunur. Sonuçlar sıfıra çok yakın çıktığı için, C Sharpta iterasyon sonucu elde edilen α, β ve θ açılarının doğru olduğu söylenebilir.

Ayrıca, sonuçların doğruluğunu test etmek için Maple'ın kendi bünyesinde bulunan fsolve( ) komutu da kullanılabilir. Bu komut yardımıyla denklem sisteminin çözümünü gösteren Maple kodları aşağıdadır.

```
> restart:
> L1:=250:
> L2:=200:
> L3:=150:
> denklem1:= proc(α, β, θ)  (150-L1*cos(α)*cos(Pi/6))^2+(L3*sin(θ)-
L1*sin(α))^2+(259.808-L3*cos(θ)-  L1*cos(α)*sin(Pi/6))^2-90000  end proc:
> denklem2:= proc(α, β, θ)  (150-300+L2*cos(β)*cos(Pi/6))^2+(L3*sin(θ)-
L2*sin(β))^2+(259.808-L3*cos(θ)-L2*cos(β)*sin(Pi/6))^2-90000  end proc:
> denklem3:= proc(α, β, θ)  (300-L2*cos(β)*cos(Pi/6)-
L1*cos(α)*cos(Pi/6))^2+(L2*sin(β)-L1*sin(α))^2+(L2*cos(β)*sin(Pi/6)-
L1*cos(α)*sin(Pi/6))^2-90000  end proc:
> c:=fsolve([denklem1,denklem2,denklem3],[0..Pi,0..Pi,0..Pi]):
> α:=evalf(c[1]*360/(2*Pi)):
> β:=evalf(c[2]*360/(2*Pi)):
> θ:=evalf(c[3]*360/(2*Pi)):
```

Kodlar çalıştırıldığında;

$$\alpha = 87.73199502 \tag{3.24}$$

$$\beta = 91.42300307 \tag{3.25}$$

$$\theta = 86.18737003 \tag{3.26}$$

olarak bulunur. Bu sonuçlar; C Sharp'ta iterasyon sonucu elde edilmiş sonuçlarla (bkz. Tablo 3.1) kıyaslandığında, α 'nın virgülden sonra sekiz basamağının, β ve θ'nın virgülden sonra yedi basamağının aynı olduğu görülür. C Sharp'ta, for döngüsü içinde Newton'un denklem sistemleri için çözüm metodu kullanılırken, Şekil 3.3'te görüleceği üzere döngü içinde bulunan α açısının yeni değeri; β açısının yeni değerini hesaplarken, eski β ve θ açı değerleriyle birlikte kullanılmaktadır. Aynı şekilde β açısının yeni değeri; θ açısının yeni değerini hesaplarken, yeni α ve eski θ açı değerleriyle birlikte kullanılmaktadır. Bu yöntem yerine; yeni α, β ve θ açıları döngü içinde hesaplanırken, denklemlerin sağ tarafında bir önceki iterasyonda bulunan eski α, β ve θ açı değerleri kullanılabilir. Bu yöntemle her iterasyon sonucu bulunan α, β ve θ açı değerleri Tablo 3.2'de gösterilmiştir.

```
for (int i = 0; i < 5; i++)
{
    α = α + (-0.5196152424e3 * Math.Sin(β) + 0.1000000000e-8 * Math.Sin(β) * L2 * Math.Cos(β) +

    β = β - (-0.5196152424e3 * Math.Sin(α) + 0.1000000001e1 * Math.Sin(α) * L2 * Math.Cos(β) +

    θ = θ + (-0.5196152424e3 * Math.Sin(α) + 0.1000000001e1 * Math.Sin(α) * L2 * Math.Cos(β) +

    label1.Text = (α * (180 / Math.PI)).ToString();
    label2.Text = (β * (180 / Math.PI)).ToString();
    label3.Text = (θ * (180 / Math.PI)).ToString();
}
```

**Şekil 3.3:** C sharp denklem sistemi for döngüsü.

**Tablo 3.2:** C sharpta iterasyon sonrası elde edilen değerler.

| İterasyon Sayısı | α° | β° | θ° |
|---|---|---|---|
| 1 | 87.913649607217 | 91.859786784679 | 85.996479060310 |
| 2 | 87.732823454959 | 91.424473612444 | 86.186284071334 |
| 3 | 87.731995037924 | 91.423003098085 | 86.187370053717 |
| 4 | 87.731995024907 | 91.423003080113 | 86.187370066377 |
| 5 | 87.731995024907 | 91.423003080113 | 86.187370066377 |

Görüldüğü üzere (bkz. Tablo 3.2), dördüncü iterasyonla birlikte α, β ve θ açıları, virgülden sonra on iki basamaklı değerlere yakınsamıştır. Dördüncü ve beşinci iterasyondaki değerlerin daha önceki değerlerle (bkz. Tablo 3.1) aynı çıktığı görülmüştür. İterasyon sonucu elde edilen α, β ve θ açıları, Maple kullanılarak; x1, x2, y1, y2, y3, z1, z2, z3 değişkenlerinin α, β ve θ açılarına bağlı denklemlerinde yerine koyulup, sonuçlar üç boyutlu grafik halinde aşağıdaki kodlar yazılarak gösterilebilir.

```
> restart:
> L1:=250:
> L2:=200:
> L3:=150:
> α:=evalf(87.731995024907*2*Pi/360):
> β:=evalf(91.423003080113*2*Pi/360):
> θ:=evalf(86.187370066377*2*Pi/360):
> x1:=evalf(L1*cos(α)*cos(Pi/6)):
> y1:=L1*sin(α):
> z1:=L1*cos(α)*sin(Pi/6):
> x2:=evalf(300-L2*cos(β)*cos(Pi/6)):
> y2:=L2*sin(β):
> z2:=L2*cos(β)*sin(Pi/6):
> x3:=150:
> y3:=evalf(L3*sin(θ)):
> z3:=259.808-L3*cos(θ):
> with(plottools):
> with(plots):
>display(line([0,0,0],[x1,y1,z1]),line([300,0,0],[x2,y2,z2]),line([150,0,259.808],[x3,y
3,z3]),line([x3,y3,z3],[x1,y1,z1]),line([x3,y3,z3],[x2,y2,z2]),line([x2,y2,z2],[x1,y1,z
1]),line([0,0,0],[150,0,259.308]),line([300,0,0],[150,0,259.808]),line([0,0,0],[300,0,0
]),axes=normal,color=red,linestyle=solid):
```

Bu kodlar sonucu;

$$x1 = 8.567982967 \quad (3.27)$$

$$y1 = 249.8041624 \quad (3.28)$$

$$z1 = 4.946727271 \quad (3.29)$$

$$x2 = 304.3012946 \quad (3.30)$$

$$y2 = 199.9383200 \quad (3.31)$$

$$z2 = -2.483353600 \quad (3.32)$$

$$x3 = 150 \quad (3.33)$$

$$y3 = 149.6680252 \quad (3.34)$$

$$z3 = 249.8339227 \quad (3.35)$$

olarak bulunur. Bu koordinatlara göre 3-RPS paralel manipülatörün Maple yardımıyla elde edilen üç boyutlu grafiği Şekil 3.4'te gösterilmiştir.

Hareketli platform üçgeninin ağırlık merkezinin koordinatları xG, yG, zG ise;

$$xG = (x1+x2+x3)/3 \quad (3.36)$$

$$yG = (y1+y2+y3)/3 \quad (3.37)$$

$$zG = (z1+z2+z3)/3 \quad (3.38)$$

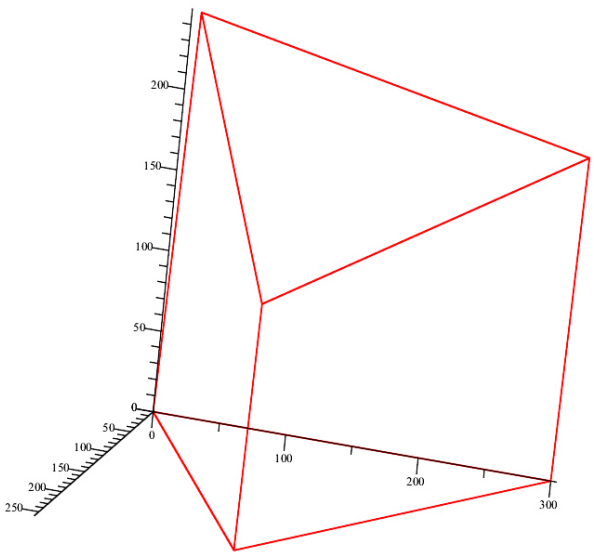
formülleri yardımıyla bulunabilir. Bu formüllere göre;

$$xG = 154.2897592 \quad (3.39)$$

$$yG = 199.8035025 \quad (3.40)$$

$$zG = 84.09909879 \quad (3.41)$$

olarak bulunur.




**Şekil 3.4:** Elde edilen koordinatlara göre 3-RPS paralel manipülatörün üç boyutlu grafiği.

#### 3.3.2.1 Newton Metodu İle İlk Tahmin Değerlerine Göre Farklı Çözüm Elde Edilmesi

İlk tahmin değerleri; α, β, θ açılarının her biri için 0.8 radyan ( 45.837°) olarak kabul edilip, C Sharp ortamında yazılan kodlar tekrar çalıştırıldığında Tablo 3.3'te görülen değerler elde edilmiştir.

Tablo 3.3'ten de görüldüğü üzere, sekizinci iterasyonla birlikte α, β, θ açıları, virgülden sonra on iki basamaklı değerlere yakınsamıştır. İterasyon sonucu elde edilen değerlerin denklem sisteminin çözümü olup olmadığını anlamak için; daha önce de yapıldığı gibi (3.16), (3.17) ve (3.18) numaralı denklemler olan sırasıyla, denklem1, denklem2 ve denklem3'ü ayrı ayrı sıfır yapıp yapmadığına bakılır. Bununla ilgili Maple'da yazılan kodlar yeniden çalıştırıldığında;

$$\text{denklem1} = -0.00010 \tag{3.42}$$

$$\text{denklem2} = -0.00002 \tag{3.43}$$

$$\text{denklem3} = -0.00005 \tag{3.44}$$

**Tablo 3.3:** C sharpta iterasyon sonrası elde edilen değerler.

| **İterasyon Sayısı** | **α°** | **β°** | **θ°** | **Toplam Hesap Süresi (milisaniye)** |
|---|---|---|---|---|
| 1 | 247.517766343261 | 137.271722964718 | 72.467667381466 | 8 |
| 2 | 378.769559711413 | 110.727630324930 | 112.923962343186 | |
| 3 | 267.814094509174 | 71.6270401984286 | 118.546301860368 | |
| 4 | 344.656637577306 | 75.9070544340835 | 105.978208959698 | |
| 5 | 339.491101222750 | 75.4806301755010 | 105.985201762914 | |
| 6 | 339.678844051366 | 75.4799697440597 | 105.985059127908 | |
| 7 | 339.679083796769 | 75.4799697421774 | 105.985059127757 | |
| 8 | 339.679083797160 | 75.4799697421774 | 105.985059127757 | |
| 9 | 339.679083797160 | 75.4799697421774 | 105.985059127757 | |

olarak bulunur. Sonuçlar sıfıra çok yakın çıktığı için; C Sharp'ta iterasyon sonucu elde edilen α, β ve θ açılarının, denklem sisteminin diğer bir çözümü olduğu söylenebilir. İterasyon sonucu elde edilen α, β ve θ açıları; Maple kullanılarak, daha önce de yapıldığı gibi x1, x2, y1, y2, y3, z1, z2, z3 değişkenlerinin α, β ve θ açılarına bağlı denklemlerinde yerine koyulup, sonuçlar üç boyutlu grafik halinde gösterilebilir. Bu durumda;

$$x1 = 203.0314765 \quad (3.45)$$

$$y1 = -86.81950262 \quad (3.46)$$

$$z1 = 117.2202776 \quad (3.47)$$

$$x2 = 256.5742912 \quad (3.48)$$

$$y2 = 193.6120100 \quad (3.49)$$

$$z2 = 25.07184466 \quad (3.50)$$

$$x3 = 150 \quad (3.51)$$

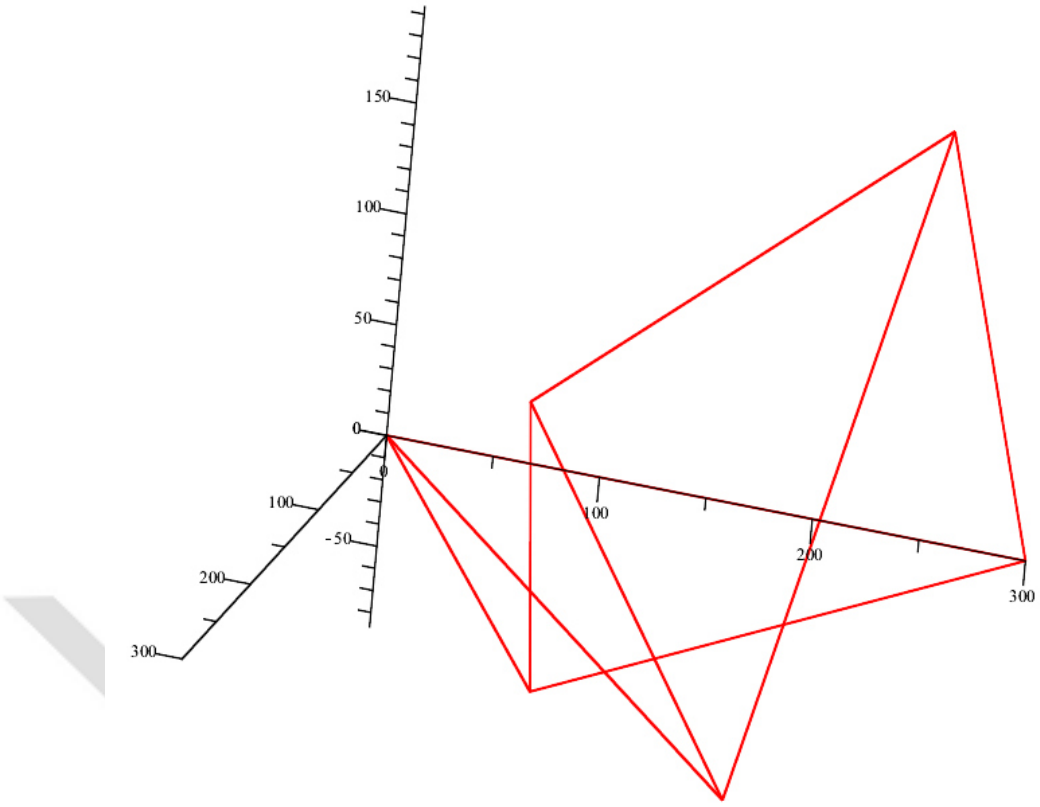
$$y3 = 144.2000311 \quad (3.52)$$

$$z3 = 301.1160020 \quad (3.53)$$

olarak bulunur. Bu koordinatlara göre 3-RPS paralel manipülatörün Maple yardımıyla elde edilen üç boyutlu grafiği Şekil 3.5'te gösterilmiştir.

Gerçek hayatta; lineer aktüatörler, fiziki sınırlamalardan dolayı Şekil 3.5'te görülen konfigürizasyona gelemeyeceğinden α, β ve θ açılarının yeni değerleri kullanılmak için uygun değildir.

Bu bulgulara göre; denklem sisteminin birden fazla çözümü olduğu durumlarda; Newton Metodu ile α, β, θ açılarının ilk tahmin değerlerine göre farklı sonuçlar bulunduğu gözlenmiştir. İstenilen sonuçları elde edebilmek için ilk tahmin değerlerinin doksan dereceye yakın seçilmesinin uygun olduğu anlaşılmıştır.




**Şekil 3.5:** Elde edilen koordinatlara göre 3-RPS paralel manipülatörün üç boyutlu grafiği.

#### 3.3.2.2 İlk Tahmin Değerleri ve Lineer Aktüatör Boyuna Göre Farklı Çözüm Elde Edilmesi

Daha önce belirtildiği üzere, 3-RPS paralel manipülatörün hareketli ve sabit platformları birer eşkenar üçgen olup, her birinin kenarı 300 mm'dir. Lineer aktüatörlerin maksimum stroklu boylarının 300 mm'yi geçtiği bazı durumlarda; α, β, θ açılarının ilk tahmin değerleri doksan dereceye yakın seçilse bile, iterasyon sonucu elde edilen α, β, θ açılarının doğru olduğu halde uygun konfigürizasyona sahip olmadığı görülmüştür. Örneğin, ilk tahmin değerleri α, β, θ açılarının her biri için 1.483 radyan ( 84.97°) olarak ve lineer aktüatör boyları L1 = 2000 mm, L2 = 2000 mm, L3 = 2000 mm kabul edilip, C Sharp ortamında yazılan kodlar tekrar çalıştırıldığında Tablo 3.4'te görülen değerler elde edilmiştir. Tablo 3.4'ten de görüldüğü üzere, sekizinci iterasyonla birlikte α, β, θ açıları, virgülden sonra on iki basamaklı değerlere yakınsamıştır. İterasyon sonucu elde edilen değerlerin denklem sisteminin çözümü olup olmadığını anlamak için; daha önce de yapıldığı gibi (3.16), (3.17) ve (3.18) numaralı denklemler olan sırasıyla, denklem1, denklem2 ve

denklem3'ü ayrı ayrı sıfır yapıp yapmadığına bakılır. Bununla ilgili Maple'da yazılan kodlar yeniden çalıştırıldığında;

$$\text{denklem1} = 0.00088 \tag{3.54}$$

$$\text{denklem2} = 0.00088 \tag{3.55}$$

$$\text{denklem3} = 0.00042 \tag{3.56}$$

olarak bulunur.

**Tablo 3.4:** C sharpta iterasyon sonrası elde edilen değerler.

| **İterasyon Sayısı** | **α°** | **β°** | **θ°** | **Toplam Hesap Süresi (milisaniye)** |
|---|---|---|---|---|
| 1 | -114.467135562456 | -114.537917742087 | -193.438110421679 | |
| 2 | -99.270103614496 | -86.990676814032 | -108.138643901668 | |
| 3 | -93.128412540093 | -89.053530853876 | -96.450189784657 | |
| 4 | -90.720523872095 | -89.828939865442 | -91.531022035156 | |
| 5 | -90.062070014856 | -89.991441693301 | -90.132392792568 | 6 |
| 6 | -90.000544163705 | -89.999964930082 | -90.001148212890 | |
| 7 | -90.000000041794 | -89.999999998715 | -89.999989236562 | |
| 8 | -90.000000000000 | -90.000000000000 | -89.999989146326 | |
| 9 | -90.000000000000 | -90.000000000000 | -89.999989146326 | |

Sonuçlar sıfıra çok yakın çıktığı için; C Sharpta iterasyon sonucu elde edilen α, β ve θ açılarının, denklem sisteminin diğer bir çözümü olduğu söylenebilir. İterasyon sonucu elde edilen α, β ve θ açıları; Maple kullanılarak, daha önce de yapıldığı gibi x1, x2, y1, y2, y3, z1, z2, z3 değişkenlerinin α, β ve θ açılarına bağlı denklemlerinde yerine koyulup, sonuçlar üç boyutlu grafik halinde gösterilebilir.

$$x1 = -3.552494764 * 10^{-7} \tag{3.57}$$

$$y1 = -2000 \tag{3.58}$$

$$z1 = -2.051033808 * 10^{-7} \tag{3.59}$$

$$x2 = 300.0000004 \quad (3.60)$$

$$y2 = -2000 \quad (3.61)$$

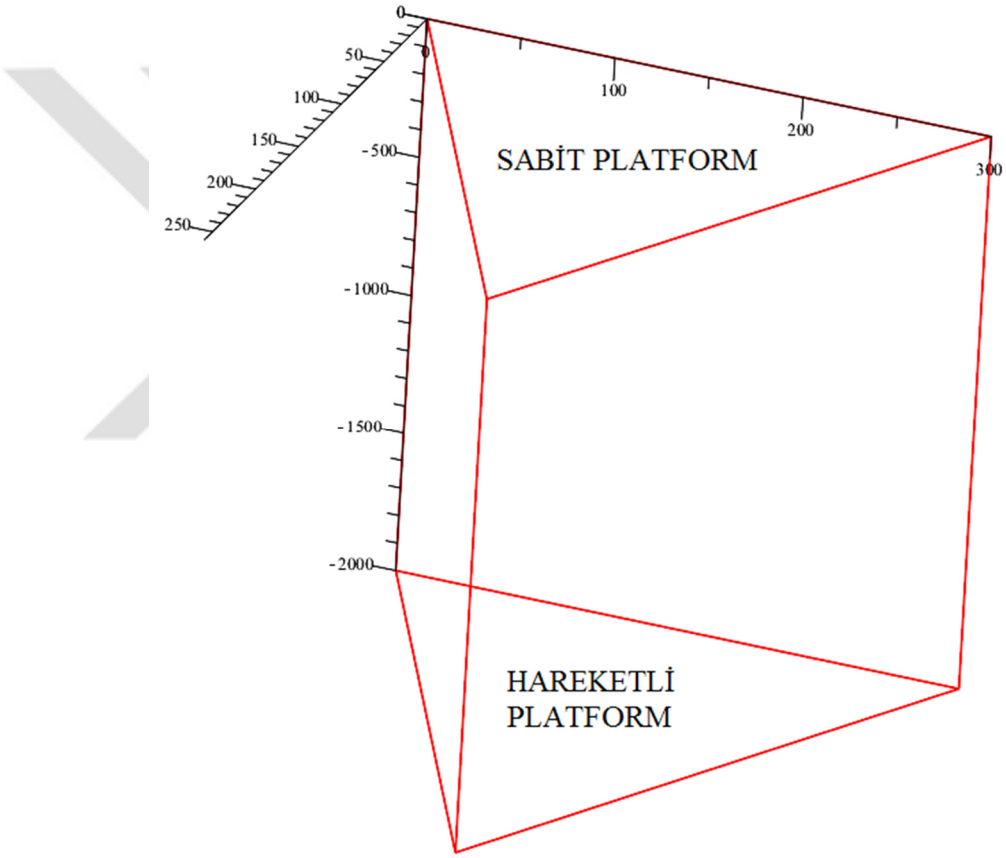
$$z2 = -2.051033808*10^{-7} \quad (3.62)$$

$$x3 = 150 \quad (3.63)$$

$$y3 = -2000 \quad (3.64)$$

$$z3 = 259.8076224 \quad (3.65)$$

olarak bulunur. Bu koordinatlara göre 3-RPS paralel manipülatörün Maple yardımıyla elde edilen üç boyutlu grafiği Şekil 3.6'da gösterilmiştir.




**Şekil 3.6:** Elde edilen koordinatlara göre 3-RPS paralel manipülatörün üç boyutlu grafiği.

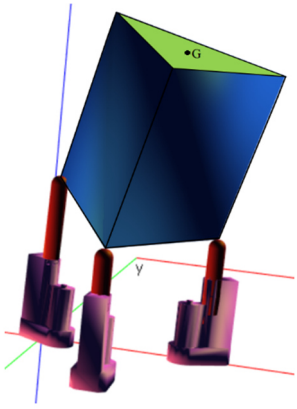
Gerçek hayatta; fiziki sınırlamalardan dolayı hareketli platformun y koordinatları, pozitiften negatife hareket edemeyeceği için Şekil 3.6'da görülen konfigürizasyona gelemeyeceğinden α, β ve θ açılarının yeni değerleri kullanılmak için uygun değildir. Dolayısıyla, lineer aktüatörlerin maksimum stroklu boyları; hareketli ve sabit platformların kenar uzunlukları olan 300 mm'yi geçmeyecek şekilde tercih edilmiştir. Bu tezde, lineer aktüatörlerin kapalı haldeki boylarının 150 mm,

maksimum stroklu boylarının ise 300 mm olduğu kabul edilmiştir. Lineer aktüatörlerin çalışma aralığı; 150 mm <= Lineer Aktüatörlerin Çalışma Aralığı <=300 mm olacaktır. Bu aralıklarda uygun olmayan konfigürizasyonlara sebep olabilecek α, β ve θ açılarının varlığı; tezin ilerleyen bölümlerinde bahsedilecek olan üç boyutlu simülasyonun çalıştırılması esnasında tespit edilememiştir. Yılansı robot oluşturulurken; 3-RPS manipülatörlerin seri olarak birbirine bağlanma aşamasında, ardışık iki 3-RPS manipülatör arasına, robotun boyunu uzatabilmek amacıyla ara platformlar eklenecektir.

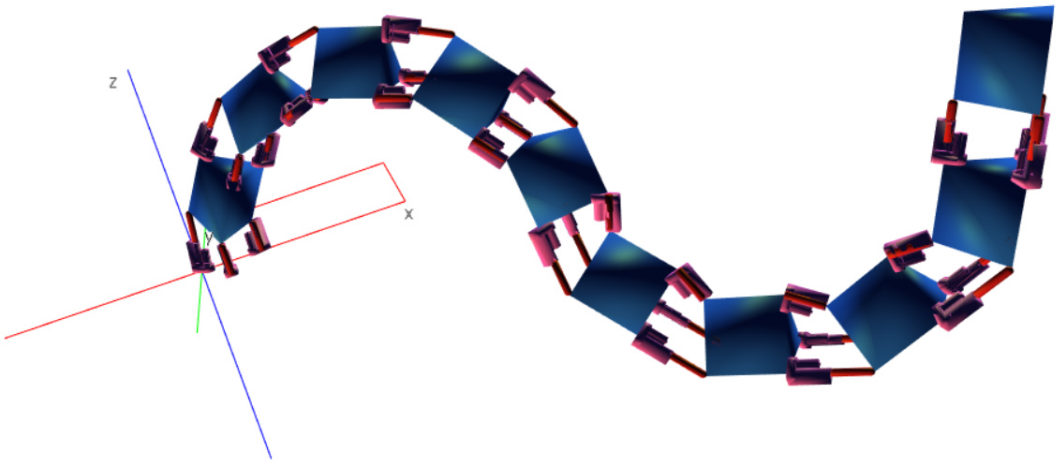
## 3.4 Modüllerin Seri Olarak Bağlanmasının İleri Kinematiğinin İncelenmesi

### 3.4.1 Giriş

Bu bölümde öncelikle Şekil 3.7'den de görüleceği üzere, 3-RPS paralel manipülatöre seri olarak 300 mm yüksekliğinde bir platform bağlanacaktır. Bu platformun tavan yüzeyinin ağırlık merkezi olan "G" noktasının koordinatları, 3-RPS manipülatörün lineer aktüatörlerinin hareketlerine bağlı olarak değişeceği için, "G" noktasının koordinatlarının lineer aktüatörlerin strok uzunluklarına göre değişimi matematiksel olarak ifade edilecektir. Ardından Şekil (3.8)'de görüldüğü gibi; C Sharp XNA ortamında ardışık iki 3-RPS manipülatör arasına ve en uca 300 mm'lik ara platformlar bağlanarak oluşturulan on modüllü yılansı robotun nasıl kodlandığı anlatılacaktır.




**Şekil 3.7:** Hareketli platformun 3-RPS manipülatöre seri olarak bağlanması.




**Şekil 3.8:** On tane 3-RPS modülden oluşan yılansı robot.

### 3.4.2 G Noktasının Koordinatlarının Bulunması

Bu bölümde, G noktasının (bkz. Şekil 3.7) koordinatlarının lineer aktüatörlerin strok uzunluklarına göre değişimi matematiksel olarak ifade edilecektir. Bunun için daha önce L1=250 mm, L2=200 mm ve L3=150 mm lineer aktüatör strok uzunluklarına göre hesaplanan x1, y1, z1, x2, y2, z2, x3, y3, z3 koordinatlarının değerleri (3.66) - (3.74) eşitlikleri arasında verilmiştir.

$$x1 = 8.567982967 \tag{3.66}$$

$$y1 = 249.8041624 \tag{3.67}$$

$$z1 = 4.946727271 \tag{3.68}$$

$$x2 = 304.3012946 \tag{3.69}$$

$$y2 = 199.9383200 \tag{3.70}$$

$$z2 = -2.483353600 \tag{3.71}$$

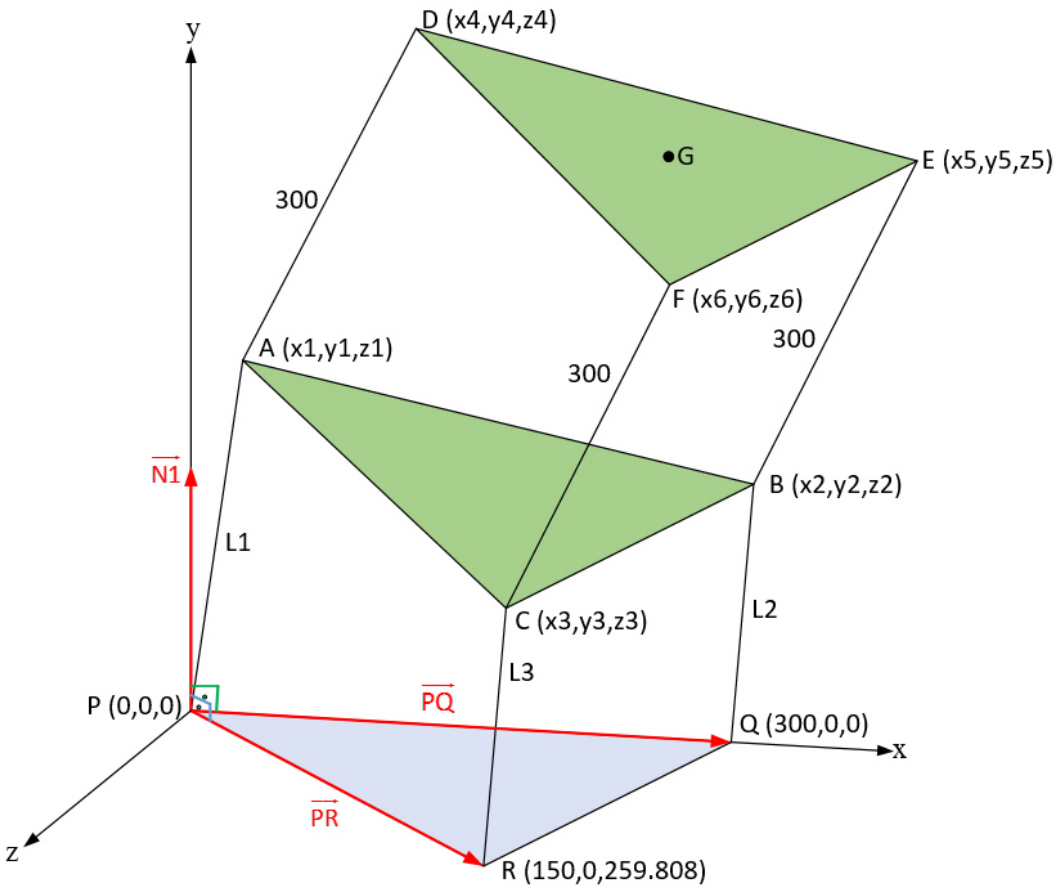
$$x3 = 150 \tag{3.72}$$

$$y3 = 149.6680252 \tag{3.73}$$

$$z3 = 249.8339227 \tag{3.74}$$

Hareketli platformun, 3-RPS manipülatörün lineer aktüatörlerinin hareketine bağlı olarak konumu değişeceği için, bu konum değişikliğinin Şekil 3.9'da görülen D, E ve F noktalarına yansıtılmasını sağlayacak matrislere ihtiyaç vardır. Bu matrislerin bulunabilmesi için öncelikle Şekil 3.9'da x-z düzlemindeki P, R ve Q noktalarının

oluşturduğu 3-RPS manipülatörün sabit platformunun üzerinde görülen $\overrightarrow{PR}$ ve $\overrightarrow{PQ}$ vektörlerinin ifade edilmesi gereklidir.




**Şekil 3.9:** Sabit platform ile ilgili vektörlerin bulunması.

$$\overrightarrow{PR} = (150 - 0)\vec{i} + (0 - 0)\vec{j} + (259.808 - 0)\vec{k} = 150\vec{i} + 259.808\vec{k} \quad (3.75)$$

$$\overrightarrow{PQ} = (300 - 0)\vec{i} + (0 - 0)\vec{j} + (0 - 0)\vec{k} = 300\vec{i} \quad (3.76)$$

Bu vektörler, (3.77) ve (3.78) eşitliklerindeki gibi de ifade edilebilirler.

$$\overrightarrow{PR} = \begin{pmatrix} 150 \\ 0 \\ 259.808 \end{pmatrix} \quad (3.77)$$

$$\overrightarrow{PQ} = \begin{pmatrix} 300 \\ 0 \\ 0 \end{pmatrix} \tag{3.78}$$

Bu iki vektör birbirleriyle vektörel olarak çarpılırsa, bu iki vektörün bulunduğu düzleme dik olan $\overrightarrow{N1}$ vektörü elde edilir (bkz. Şekil 3.9).

$$\overrightarrow{N1} = \overrightarrow{PR} \text{ x } \overrightarrow{PQ} \tag{3.79}$$

$$\overrightarrow{N1} = \overrightarrow{PR} \times \overrightarrow{PQ} = \begin{pmatrix} 150 \\ 0 \\ 259.808 \end{pmatrix} \times \begin{pmatrix} 300 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 77942.4 \\ 0 \end{pmatrix} \tag{3.80}$$

$$\overrightarrow{N1} = (0)\vec{i} + (77942.4)\vec{j} + (0)\vec{k} = (77942.4)\vec{j} \tag{3.81}$$

olarak bulunur. $\overrightarrow{N1}$ vektörünün yönü ise, vektörel çarpım kuralı gereği pozitif y ekseni yönündedir ve bu vektör $\overrightarrow{PR}$ ve $\overrightarrow{PQ}$ vektörlerinin bulunduğu düzleme diktir.
Daha sonra, Şekil 3.10'da A, B ve C noktalarının oluşturduğu, hareketli platformun tabanı üzerinde görülen $\overrightarrow{AB}$ ve $\overrightarrow{AC}$ vektörlerinin ifade edilmesi gereklidir.

$$\overrightarrow{PA} = (x1)\vec{i} + (y1)\vec{j} + (z1)\vec{k} \tag{3.82}$$

$$\overrightarrow{PB} = (x2)\vec{i} + (y2)\vec{j} + (z2)\vec{k} \tag{3.83}$$

$$\overrightarrow{PC} = (x3)\vec{i} + (y3)\vec{j} + (z3)\vec{k} \tag{3.84}$$

$$\overrightarrow{AC} = \overrightarrow{PC} - \overrightarrow{PA} = (x3 - x1)\vec{i} + (y3 - y1)\vec{j} + (z3 - z1)\vec{k} \tag{3.85}$$

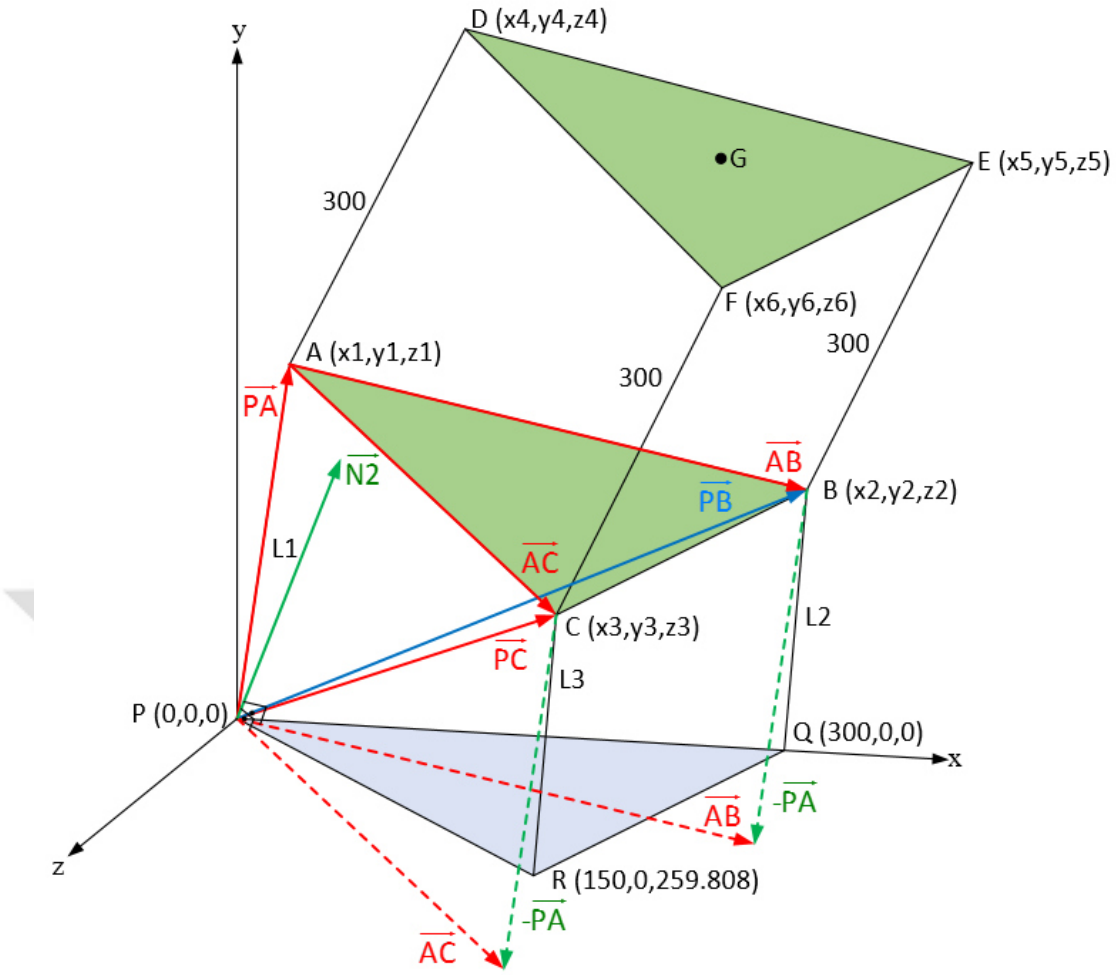
$$\overrightarrow{AB} = \overrightarrow{PB} - \overrightarrow{PA} = (x2 - x1)\vec{i} + (y2 - y1)\vec{j} + (z2 - z1)\vec{k} \tag{3.86}$$

$$\overrightarrow{AC} = (150 - 8.5679)\vec{i} + (149.6680 - 249.8041)\vec{j} + (249.8339 - 4.9467)\vec{k} \tag{3.87}$$

$$\overrightarrow{AC} = (141.4321)\vec{i} + (-100.1361)\vec{j} + (244.8872)\vec{k} \tag{3.88}$$

$$\overrightarrow{AB} = (304.3012 - 8.5679)\vec{i} + (199.9383 - 249.8041)\vec{j} + (-2.4833 - 4.9467)\vec{k} \tag{3.89}$$

$$\overrightarrow{AB} = (295.7333)\vec{i} + (-49.8658)\vec{j} + (-7.43)\vec{k} \tag{3.90}$$




**Şekil 3.10:** Hareketli platform ile ilgili vektörlerin bulunması.

Bu vektörler aşağıdaki gibi matris formatıyla da ifade edilebilirler.

$$\overrightarrow{AC} = \begin{pmatrix} 141.4321 \\ -100.1361 \\ 244.8872 \end{pmatrix} \tag{3.91}$$

$$\overrightarrow{AB} = \begin{pmatrix} 295.7333 \\ -49.8658 \\ -7.43 \end{pmatrix} \tag{3.92}$$

Bu iki vektör birbirleriyle vektörel olarak çarpılırsa, bu iki vektörün bulunduğu düzleme dik olan Şekil (3.10)'da görülen $\overrightarrow{N2}$ vektörü elde edilir.

$$\overrightarrow{N2}=\overrightarrow{AC} \times \overrightarrow{AB} \tag{3.93}$$

$$\overrightarrow{N2} = \overrightarrow{AC} \times \overrightarrow{AB} = \begin{pmatrix} 141.4321 \\ -100.1361 \\ 244.8872 \end{pmatrix} \times \begin{pmatrix} 295.7333 \\ -49.8658 \\ -7.43 \end{pmatrix} = \begin{pmatrix} 12955.5074 \\ 73472.1403 \\ 22560.9545 \end{pmatrix} \quad (3.94)$$

$$\overrightarrow{N2} = (12955.5074)\vec{i} + (73472.1403)\vec{j} + (22560.9545)\vec{k} \quad (3.95)$$

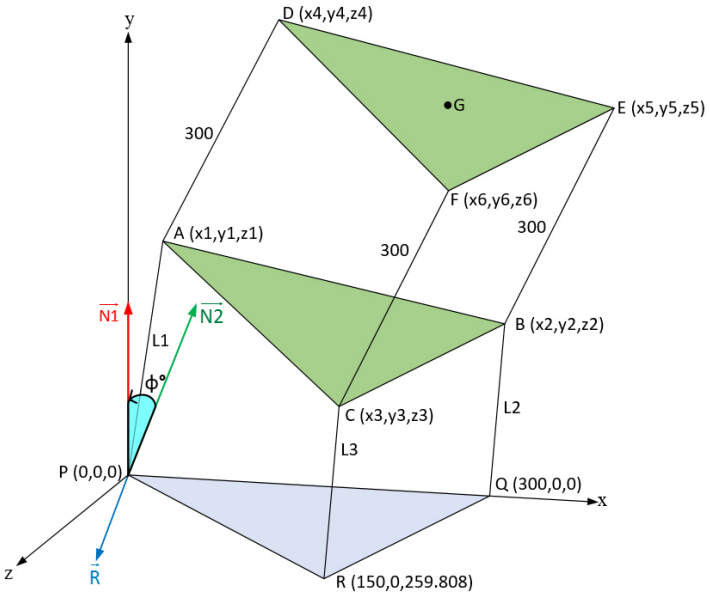
Hareketli platformun hangi eksen etrafında hareket ettiğinin belirlenmesi gerekmektedir. Bu eksen $\overrightarrow{N2}$ ve $\overrightarrow{N1}$ vektörlerinin vektörel olarak çarpılmasıyla elde edilir ve (3.96) eşitliğindeki gibi $\vec{R}$ vektörüyle gösterilebilir. $\overrightarrow{N2}$ ve $\overrightarrow{N1}$ vektörlerinin Şekil 3.11'de gösterildiği gibi arasındaki $\phi$ açısı, vektörel çarpımın sırasından dolayı ok yönünde pozitif kabul edilmiştir.

$$\vec{R} = \overrightarrow{N2} \times \overrightarrow{N1} \quad (3.96)$$

$$\vec{R} = \begin{pmatrix} 12955.5074 \\ 73472.1403 \\ 22560.9545 \end{pmatrix} \times \begin{pmatrix} 0 \\ 77942.4 \\ 0 \end{pmatrix} = \begin{pmatrix} -1758454940.0208 \\ 0 \\ 1009783339.9738 \end{pmatrix} \quad (3.97)$$

$$\vec{R} = (-1758454940.0208)\vec{i} + (1009783339.9738)\vec{k} \quad (3.98)$$

$\vec{R}$ vektörünün, rotasyon matrisinde kullanılabilmesi amacıyla birim vektör haline dönüştürülmesi için öncelikle $\vec{R}$ vektörünün büyüklüğü olan $|\overrightarrow{R}|$'nin hesaplanması gerekmektedir.




**Şekil 3.11:** Hareketli platformun dönme ekseninin bulunması.

$$\left|\vec{R}\right| = \sqrt{(-1758454940.0208)^2 + (1009783339.9738)^2} \tag{3.99}$$

$$\left|\vec{R}\right| = 2027763834.8122 \tag{3.100}$$

Ardından $\vec{U}$ birim vektörü (3.102) eşitliğindeki gibi bulunabilir.

$$\vec{U} = \frac{\vec{R}}{\left|\vec{R}\right|} \tag{3.101}$$

$$\vec{U} = (-0.8672)\vec{i} + (0.4980)\vec{k} \tag{3.102}$$

$\vec{U}$ birim vektörünün x eksenindeki bileşeni Ux, z eksenindeki bileşeni ise Uz'dir.

$$Ux = -0.8672 \tag{3.103}$$

$$Uy = 0 \tag{3.104}$$

$$Uz = 0.4980 \tag{3.105}$$

$\overrightarrow{N2}$ ve $\overrightarrow{N1}$ vektörlerinin arasındaki açı $\phi$, (3.106) eşitliğindeki formülden yola çıkarak hesaplanabilir.

$$\cos(\phi) = \frac{\vec{N2}\cdot\vec{N1}}{\left|\vec{N2}\right| \cdot \left|\vec{N1}\right|} \tag{3.106}$$

$$\vec{N2}\cdot\vec{N1} = \begin{pmatrix} 12955.5074 \\ 73472.1403 \\ 22560.9545 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 77942.4 \\ 0 \end{pmatrix} = 5726594948.1187 \tag{3.107}$$

$$\left|\overrightarrow{N2}\right| = \sqrt{(12955.5074)^2 + (73472.1403)^2 + (22560.9545)^2} \tag{3.108}$$

$$\left|\overrightarrow{N2}\right| = 77942.2686 \tag{3.109}$$

$$\left|\overrightarrow{N1}\right| = \sqrt{(77942.4)^2} = 77942.4 \tag{3.110}$$

$$\cos(\phi) = 0.9426 \tag{3.111}$$

$$\phi = 19.5092° \tag{3.112}$$

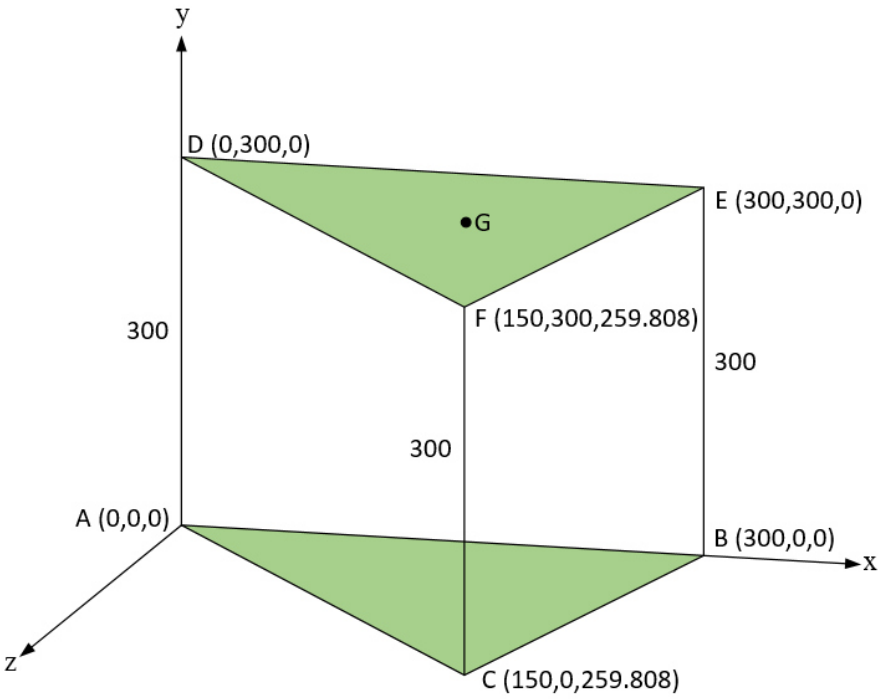
$\vec{U}$ birim vektörü ile gösterilen eksen etrafında $\phi$ açısı kadar dönmeyi ifade eden rotasyon matrisi Rot, (3.113) eşitliği yardımıyla hesaplanabilir.

$$\text{Rot} = \begin{bmatrix} \cos(\phi) + Ux^2 \cdot (1 - \cos(\phi)) & Ux \cdot Uy \cdot (1 - \cos(\phi)) - Uz \cdot \sin(\phi) & Ux \cdot Uz \cdot (1 - \cos(\phi)) + Uy \cdot \sin(\phi) \\ Uy \cdot Ux \cdot (1 - \cos(\phi)) + Uz \cdot \sin(\phi) & \cos(\phi) + Uy^2 \cdot (1 - \cos(\phi)) & Uy \cdot Uz \cdot (1 - \cos(\phi)) - Ux \cdot \sin(\phi) \\ Uz \cdot Ux \cdot (1 - \cos(\phi)) - Uy \cdot \sin(\phi) & Uz \cdot Uy \cdot (1 - \cos(\phi)) + Ux \cdot \sin(\phi) & \cos(\phi) + Uz^2 \cdot (1 - \cos(\phi)) \end{bmatrix} \quad (3.113)$$

$\phi$ açısını (3.113) eşitliğinde yerine koyarken negatif değeri kullanılmalıdır. Çünkü $\vec{R}$ vektörü ile tanımlanan dönme ekseni bulunurken $\phi$ açısı saat yönünün tersine pozitif kabul edilmiştir. Fakat lineer aktüatörlerin mevcut konumuna gelebilmesi için hareketli platformun tabanına göre tanımlanan $\overrightarrow{N2}$ vektörü, $\vec{R}$ vektörü ile gösterilen dönme ekseni etrafında saat yönünde dönecektir. Bu durumda daha önce bulunan Ux, Uy, Uz ve $\phi$ değerleri, (3.113) eşitliğinde yerine konursa aşağıdaki sonuç elde edilir.

$$\text{Rot} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \quad (3.114)$$

Sadece hareketli platformun koordinat sistemi üzerinde olduğu varsayılırsa ve hareketli platformun tabanının xz-düzleminde olduğu kabul edilirse, hareketli platform Şekil 3.12'deki koordinatlarda bulunur.




**Şekil 3.12:** Hareketli platformun referans koordinat sistemine göre koordinatları.

Rotasyon matrisi; A, B, C, D, E, F noktalarının (bkz. Şekil 3.12) koordinatları ile teker teker skaler olarak çarpılırsa, lineer aktuatörlerin hareketinden kaynaklanan dönme etkisi hareketli platforma aktarılmış olur.

A noktasının rotasyon matrisi ile çarpımı sonucu yeni koordinatları :

$$A_{yeni} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (3.115)$$

B noktasının rotasyon matrisi ile çarpımı sonucu yeni koordinatları :

$$B_{yeni} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \cdot \begin{pmatrix} 300 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 295.74 \\ -49.89 \\ -7.44 \end{pmatrix} \quad (3.116)$$

C noktasının rotasyon matrisi ile çarpımı sonucu yeni koordinatları :

$$C_{yeni} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \cdot \begin{pmatrix} 150 \\ 0 \\ 259.808 \end{pmatrix} = \begin{pmatrix} 141.4268 \\ -100.1854 \\ 244.8643 \end{pmatrix} \quad (3.117)$$

D noktasının rotasyon matrisi ile çarpımı sonucu yeni koordinatları :

$$D_{yeni} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 300 \\ 0 \end{pmatrix} = \begin{pmatrix} 49.89 \\ 282.78 \\ 86.88 \end{pmatrix} \quad (3.118)$$

E noktasının rotasyon matrisi ile çarpımı sonucu yeni koordinatları :

$$E_{yeni} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \cdot \begin{pmatrix} 300 \\ 300 \\ 0 \end{pmatrix} = \begin{pmatrix} 345.63 \\ 232.89 \\ 79.44 \end{pmatrix} \quad (3.119)$$

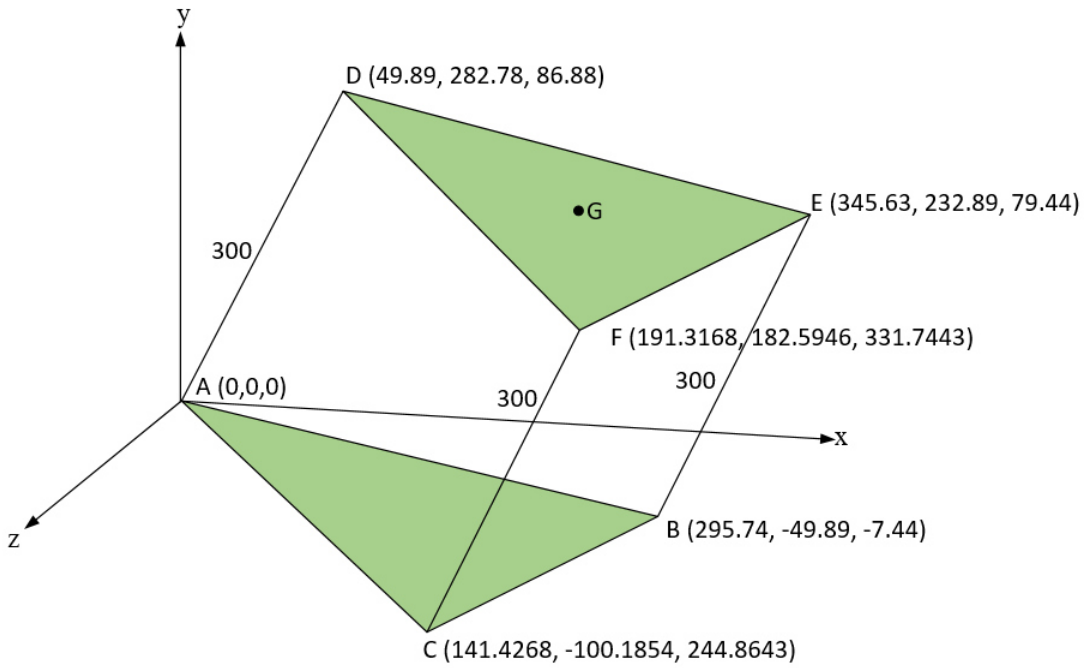
F noktasının rotasyon matrisi ile çarpımı sonucu yeni koordinatları :

$$F_{yeni} = \begin{pmatrix} 0.9858 & 0.1663 & -0.0248 \\ -0.1663 & 0.9426 & -0.2896 \\ -0.0248 & 0.2896 & 0.9568 \end{pmatrix} \cdot \begin{pmatrix} 150 \\ 300 \\ 259.808 \end{pmatrix} = \begin{pmatrix} 191.3168 \\ 182.5946 \\ 331.7443 \end{pmatrix} \quad (3.120)$$

Yeni koordinatlara göre hareketli platformun konfigürizasyonu Şekil 3.13'te görülmektedir. Bu yeni noktaların her birine ayrı ayrı birinci lineer aktüatörün uç noktasının x1, y1, z1 koordinatları (bkz. Şekil 3.11) eklenerek; hareketli platform, lineer aktüatörlerin hareketine bağlı olarak referans koordinat sistemine göre uzaydaki gerçek konumuna ötelenmiş olur.

A noktasının öteleme sonucu gerçek koordinatları :

$$A_{gerçek} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} = \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} \quad (3.121)$$




**Şekil 3.13:** Hareketli platformun referans koordinat sistemine göre yeni koordinatları.

B noktasının öteleme sonucu gerçek koordinatları :

$$B_{gerçek} = \begin{pmatrix} 295.74 \\ -49.89 \\ -7.44 \end{pmatrix} + \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} = \begin{pmatrix} 304.3079 \\ 199.9141 \\ -2.4933 \end{pmatrix} \quad (3.122)$$

C noktasının öteleme sonucu gerçek koordinatları :

$$C_{gerçek} = \begin{pmatrix} 141.4268 \\ -100.1854 \\ 244.8643 \end{pmatrix} + \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} = \begin{pmatrix} 149.9947 \\ 149.6187 \\ 249.811 \end{pmatrix} \quad (3.123)$$

D noktasının öteleme sonucu gerçek koordinatları :

$$D_{gerçek} = \begin{pmatrix} 49.89 \\ 282.78 \\ 86.88 \end{pmatrix} + \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} = \begin{pmatrix} 58.4579 \\ 532.5841 \\ 91.8267 \end{pmatrix} \quad (3.124)$$

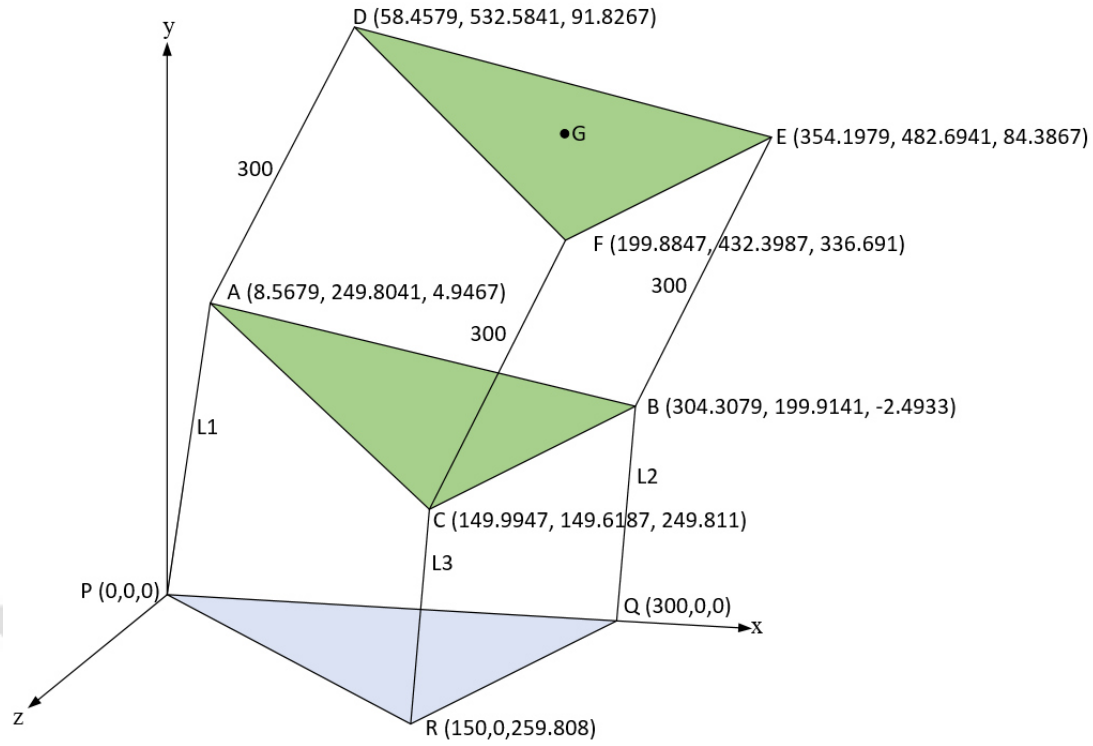
E noktasının öteleme sonucu gerçek koordinatları :

$$E_{gerçek} = \begin{pmatrix} 345.63 \\ 232.89 \\ 79.44 \end{pmatrix} + \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} = \begin{pmatrix} 354.1979 \\ 482.6941 \\ 84.3867 \end{pmatrix} \quad (3.125)$$

F noktasının öteleme sonucu gerçek koordinatları :

$$F_{gerçek} = \begin{pmatrix} 191.3168 \\ 182.5946 \\ 331.7443 \end{pmatrix} + \begin{pmatrix} 8.5679 \\ 249.8041 \\ 4.9467 \end{pmatrix} = \begin{pmatrix} 199.8847 \\ 432.3987 \\ 336.691 \end{pmatrix} \quad (3.126)$$

Gerçek koordinatlara göre hareketli platformun konfigürizasyonu Şekil 3.14'te görülmektedir.




**Şekil 3.14:** Hareketli platformun referans koordinat sistemine göre gerçek koordinatları.

A noktası, birinci lineer aktüatörün uç noktasının x1, y1, z1 koordinatlarına ötelendiği için; B ve C noktalarının; öteleme sonucu elde edilen koordinatlarının, daha önce aynı noktalar için Newton Metodu ile elde edilen koordinatlar ile eşit olması gerekmektedir. Bu değerler Tablo 3.5'te görülmektedir.

Tablo 3.5'ten de görüldüğü gibi sonuçlar birbirine çok yakındır. Dolayısıyla yöntem diğer modüllerin birbirine eklenmesinde kullanılabilir.

Hareketli platformun G noktasının koordinatları aşağıdaki gibi bulunur.

$$xG = \frac{(x4 + x5 + x6)}{3} = \frac{(58.4579 + 354.1979 + 199.8847)}{3} \tag{3.127}$$

$$xG = 204.1802 \tag{3.128}$$

$$yG = \frac{(y4 + y5 + y6)}{3} = \frac{(532.5841 + 482.6941 + 432.3987)}{3} \tag{3.129}$$

$$yG = 482.559 \tag{3.130}$$

**Tablo 3.5:** Newton metodu ve öteleme sonucu bulunan koordinatların karşılaştırılması.

| | **Newton Metodu Sonucu** | **Öteleme Sonucu** |
|---|---|---|
| **B Noktası** | | |
| x2 | 304.3012946 | 304.3079 |
| y2 | 199.9383200 | 199.9141 |
| z2 | -2.483353600 | -2.4933 |
| **C Noktası** | | |
| x3 | 150 | 149.9947 |
| y3 | 149.6680252 | 149.6187 |
| z3 | 249.8339227 | 249.811 |

$$zG = \frac{(z4 + z5 + z6)}{3} = \frac{(91.8267 + 84.3867 + 336.691)}{3} \tag{3.131}$$

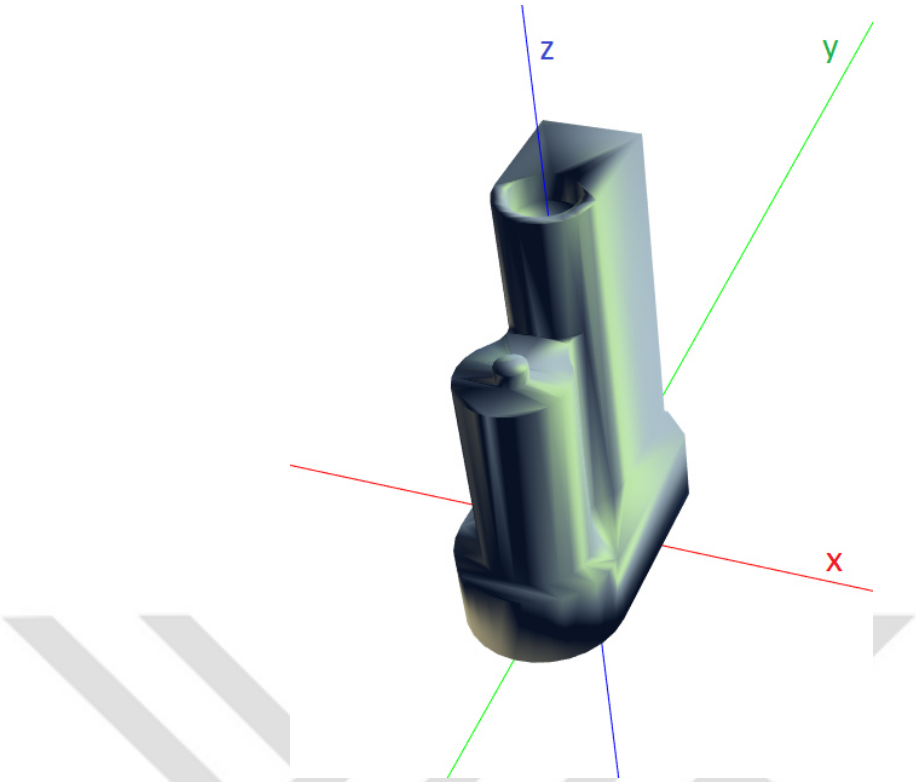
$$zG = 170.9681 \tag{3.132}$$

Diğer modüller birbirine eklenerek on modüllü yılansı bir robot C Sharp XNA ortamında oluşturulacaktır.

### 3.4.3 C Sharp XNA Ortamında On Modüllü Yılansı Robotun Kodlanması

Yılansı Robotun, 3D CAD ortamında modellenen lineer aktüatör iç ve dış ünitesi ile hareketli ara platform, FBX dosya uzantısıyla kaydedilip XNA'e aktarılır. L1 uzunluğundaki lineer aktüatörün dış ünitesi XNA'e aktarıldığında Şekil 3.15'teki gibi görünmektedir.

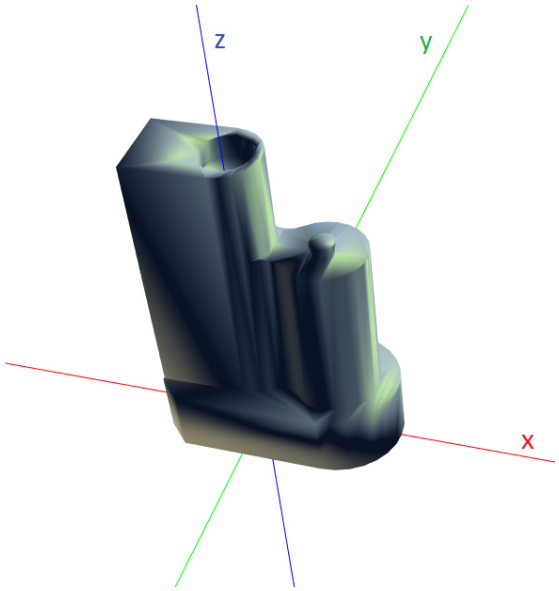
Anlatım sadeliği açısından bundan sonra L1 uzunluğundaki lineer aktüatöre birinci aktüatör, L2 uzunluğundaki lineer aktüatöre ikinci aktüatör ve L3 uzunluğundaki lineer aktüatöre üçüncü aktüatör denilecektir. Birinci aktüatörü manipüle edebilmek için, actuatorOut1 adında bir matris tanımlanır. Bu aktüatörün dönme ekseninin; geometrik olarak daha önce tanımlanan dönel mafsalın dönme ekseni ile çakışacak şekilde konumlandırılabilmesi için, öncelikle aktüatör z ekseni etrafında "Matrix.CreateRotationZ" kodu ile doksan derece döndürülür. Matrisin içine doksan derece, radyana dönüştürülüp yazılır.




**Şekil 3.15:** XNA ortamında L1 uzunluğundaki lineer aktüatörün dış ünitesi.

**actuatorOut1[0]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571));

actuatorOut1 matrisinin içindeki sıfır indisi, birinci modülle ilgili bilgiyi içinde saklayacaktır. Yazılan bu kod sonrası birinci aktüatörün konumu Şekil 3.16'da görülmektedir.

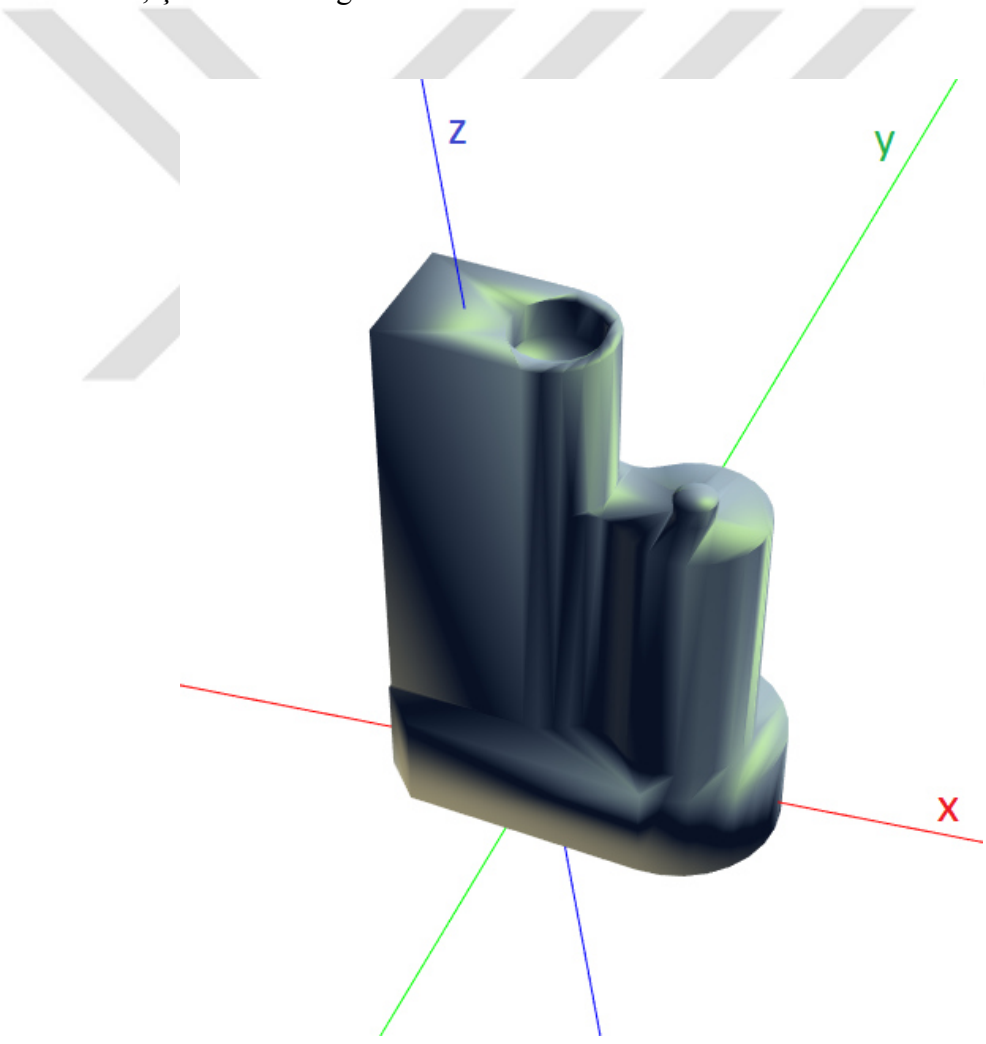



**Şekil 3.16:** Birinci aktüatörün dış ünitesinin z Ekseni etrafında 90° döndürülmesi.

Ardından, aktüatörün y ekseni etrafında dönmesi için önceki koda aşağıdaki ekleme yapılır.

**actuatorOut1[0]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle0[0]);

Bu kodda görülen angle0[0] matrisi, birinci modüldeki birinci aktüatörün ne kadar döneceği bilgisini saklayacaktır. Yazılımda her modül için üç adet buton vasıtasıyla sadece birinci, ikinci ve üçüncü lineer aktüatörün uzunluğu daha önce tanımlanan sınırlar içinde değiştirilecektir. Bu değişime bağlı olarak aktüatörler, tanımlı dönme eksenleri etrafında dönecektir. Birinci aktüatörün yazılan kod sonrası y eksenindeki dönmesi, Şekil 3.17'de görülmektedir.

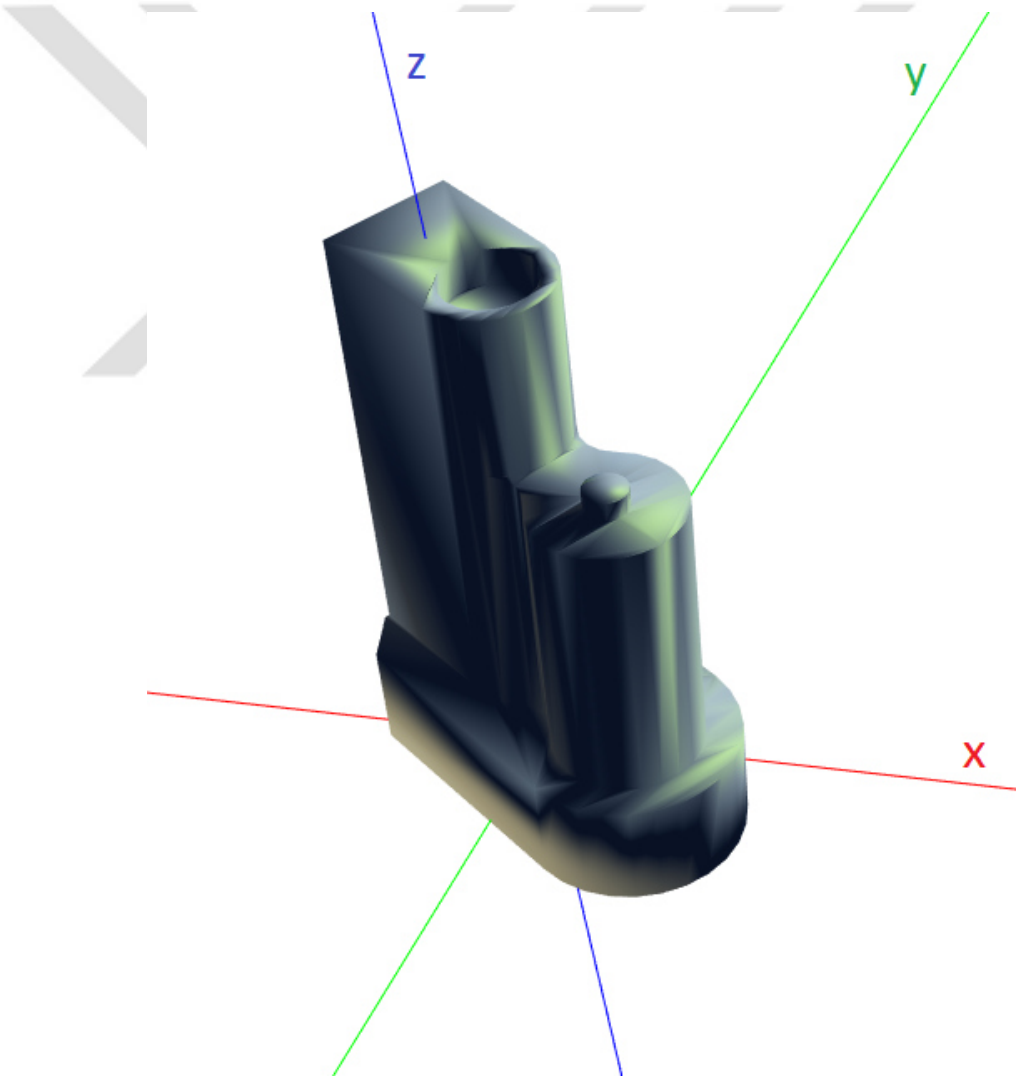



**Şekil 3.17:** Birinci aktüatörün dış ünitesinin y ekseni etrafında döndürülmesi.

Ardından, aktüatör z ekseninde otuz derece döndürülerek, aktüatörün dönme ekseninin geometrik olarak daha önce tanımlanan dönel mafsalın dönme ekseni ile çakışacak şekilde konumlandırılması, aşağıdaki kod ile sağlanmış olur. Şekil 3.18'de aktüatörün son hali görülmektedir.

**actuatorOut1[0]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle0[0]) *Matrix.CreateRotationZ(Convert.ToSingle(-0.524));

Benzer işlemler ikinci aktüatör için de yapıldıktan sonra, buna ek olarak bir de önceden tanımlı konumuna kod yazılarak Şekil 3.19'da görüldüğü gibi ötelenmelidir.




**Şekil 3.18:** Birinci aktüatörün dış ünitesinin z ekseni etrafında 30° döndürülmesi.

*İkinci aktüatör için yazılması gereken kod*

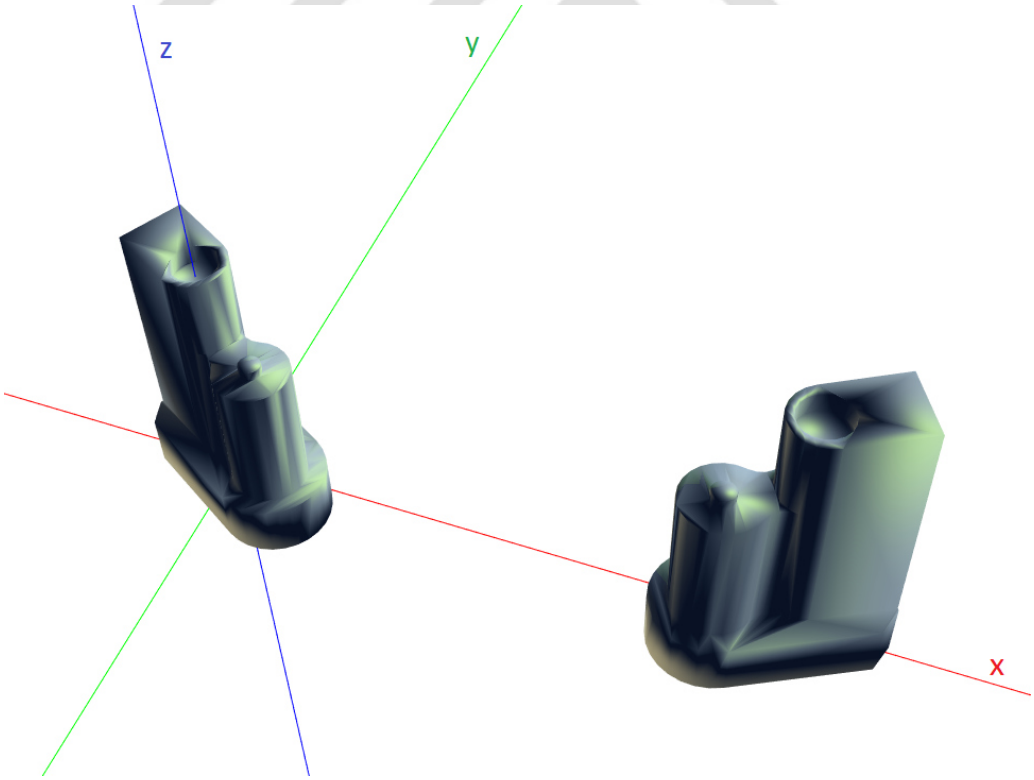
**actuatorOut2[0]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle1[0]) * Matrix.CreateRotationZ(Convert.ToSingle(-2.618)) * Matrix.CreateTranslation(300, 0, 0);

Birinci ve ikinci aktüatöre benzer döndürme ve öteleme işlemleri üçüncü aktüatör için de yapılmalıdır.

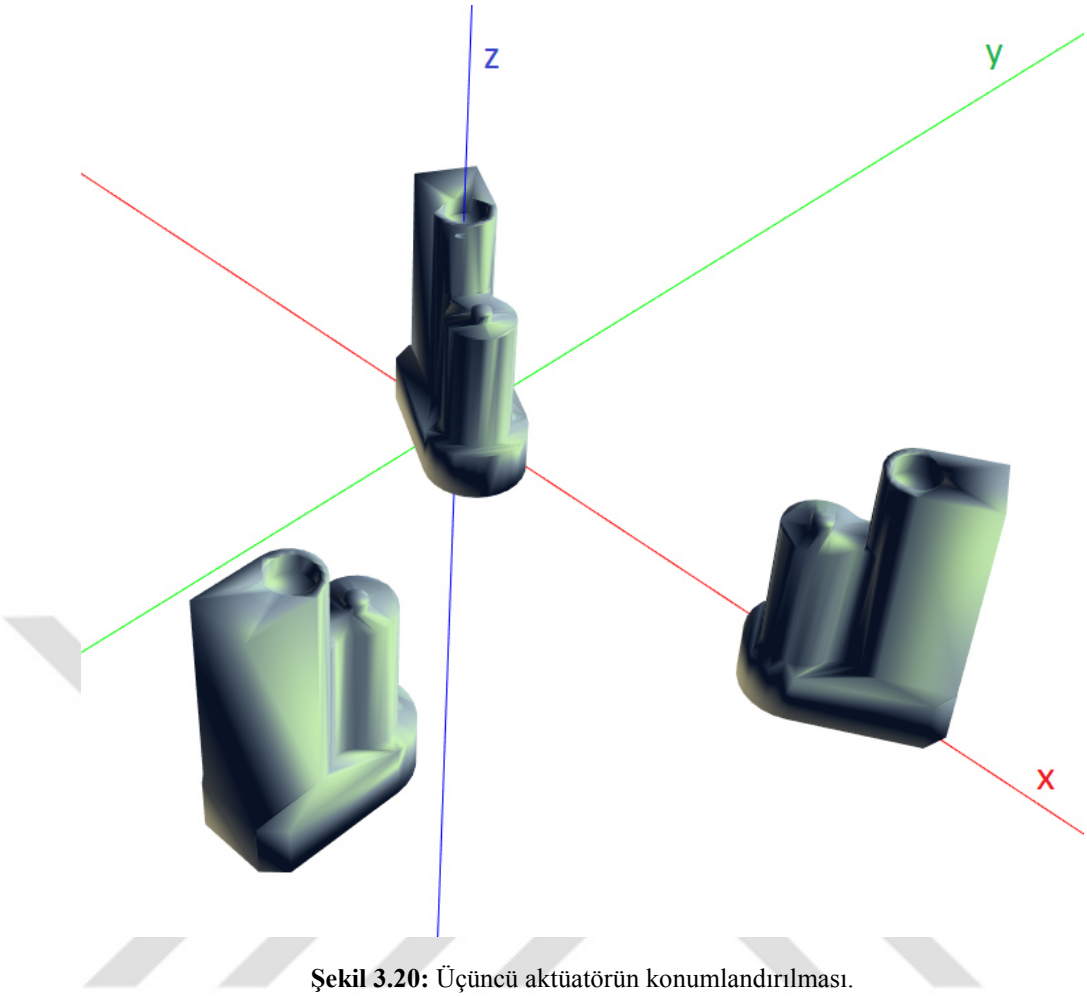
*Üçüncü aktüatör için yazılması gereken kod*

**actuatorOut3[0]** = Matrix.CreateRotationZ(Convert.ToSingle(3.142)) * Matrix.CreateRotationX(angle2[0]) * Matrix.CreateTranslation(150, -259.808f, 0);

Üçüncü aktüatör, Şekil 3.20'de görülmektedir.




**Şekil 3.19:** İkinci aktüatörün konumlandırılması.




**Şekil 3.20:** Üçüncü aktüatörün konumlandırılması.

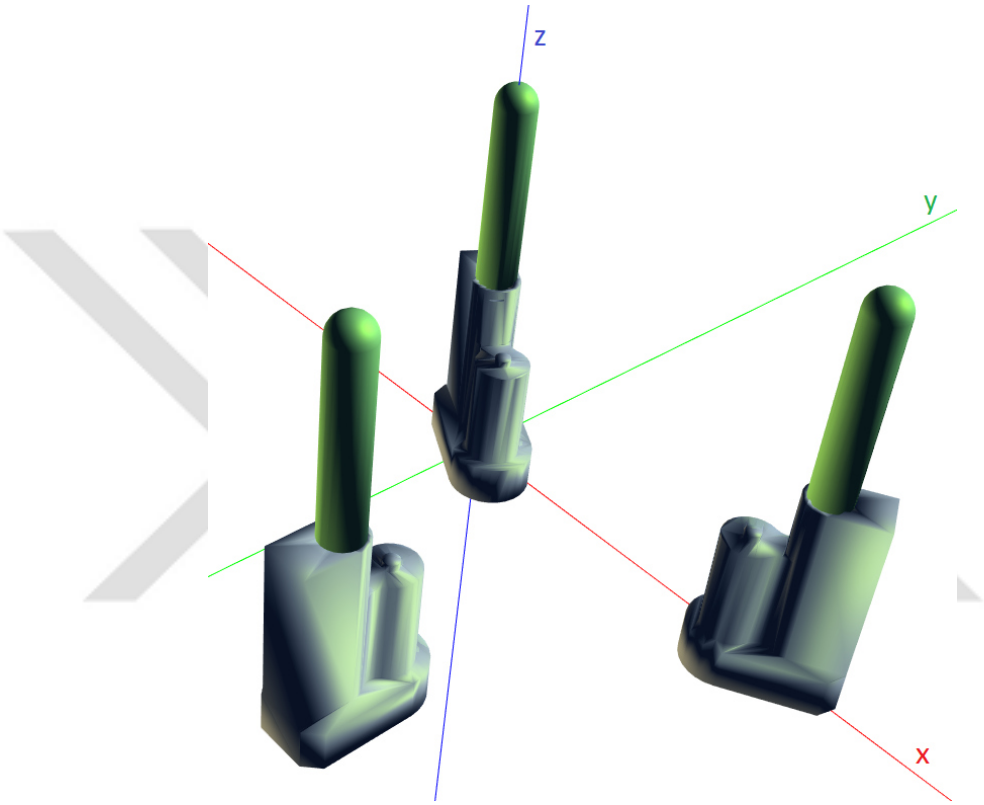
Lineer aktüatörlerin dış üniteleri konumlandırıldıktan sonra, iç ünitelerinin de yerleştirilmesi gereklidir.

***Aktüatörlerin iç ünitesi için yazılması gereken kodlar***

**actuatorBirinciModulInside1** = Matrix.CreateTranslation(0, 0, length[0]) * actuatorOut1[0];
**actuatorBirinciModulInside2** = Matrix.CreateTranslation(0, 0, length[1]) * actuatorOut2[0];
**actuatorBirinciModulInside3** = Matrix.CreateTranslation(0, 0, length[2]) * actuatorOut3[0];

length[0], length[1] ve length[2] kodları sırasıyla birinci, ikinci ve üçüncü aktüatörün iç ünitesinin uzunluğunu temsil etmektedir. Uzayıp kısalacak olan iç üniteler olduğundan dolayı bunların konumu butonlar vasıtasıyla referans koordinat

sisteminin z ekseninde orijinden başlayarak ötelenir. Öteleme işleminin hemen ardından; iç ünitelerin, dış ünitelerle birlikte hareket etmesini sağlamak için her bir iç ünite daha önce tanımlanan sırasıyla actuatorOut1[0], actuatorOut2[0] ve actuatorOut3[0] matrisleriyle çarpılır. Bu sayede dış ünitelerin konumlarındaki değişiklik eş zamanlı olarak iç ünitelere de aktarılmış olur. İç ünitelerin, maksimum uzunluklarına getirilmiş haldeki durumları Şekil 3.21'de görülmektedir.




**Şekil 3.21:** Aktüatörlerin iç ünitelerinin konumlandırılması.

Lineer aktüatörlerin iç ve dış üniteleri konumlandırıldığına göre sıra, hareketli platformun konumlandırılmasına gelmiştir.

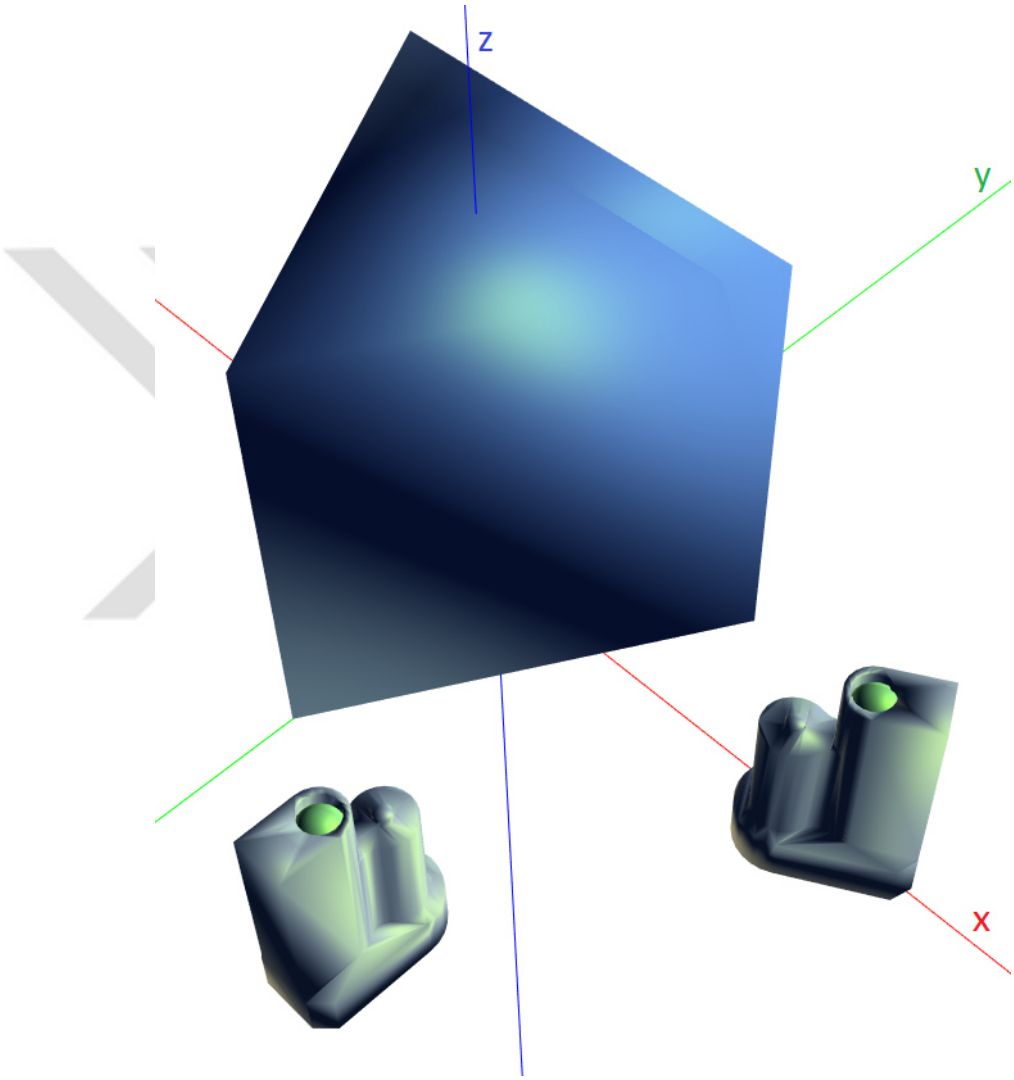
***Hareketli platformun konumlandırılması için gerekli kodlar***

**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));

**mx[0]** = Matrix.CreateTranslation(Convert.ToSingle((c.X1 + c.X2 + c.X3) / 3), -Convert.ToSingle((c.Z1 + c.Z2 + c.Z3) / 3), Convert.ToSingle((c.Y1 + c.Y2 + c.Y3) / 3));

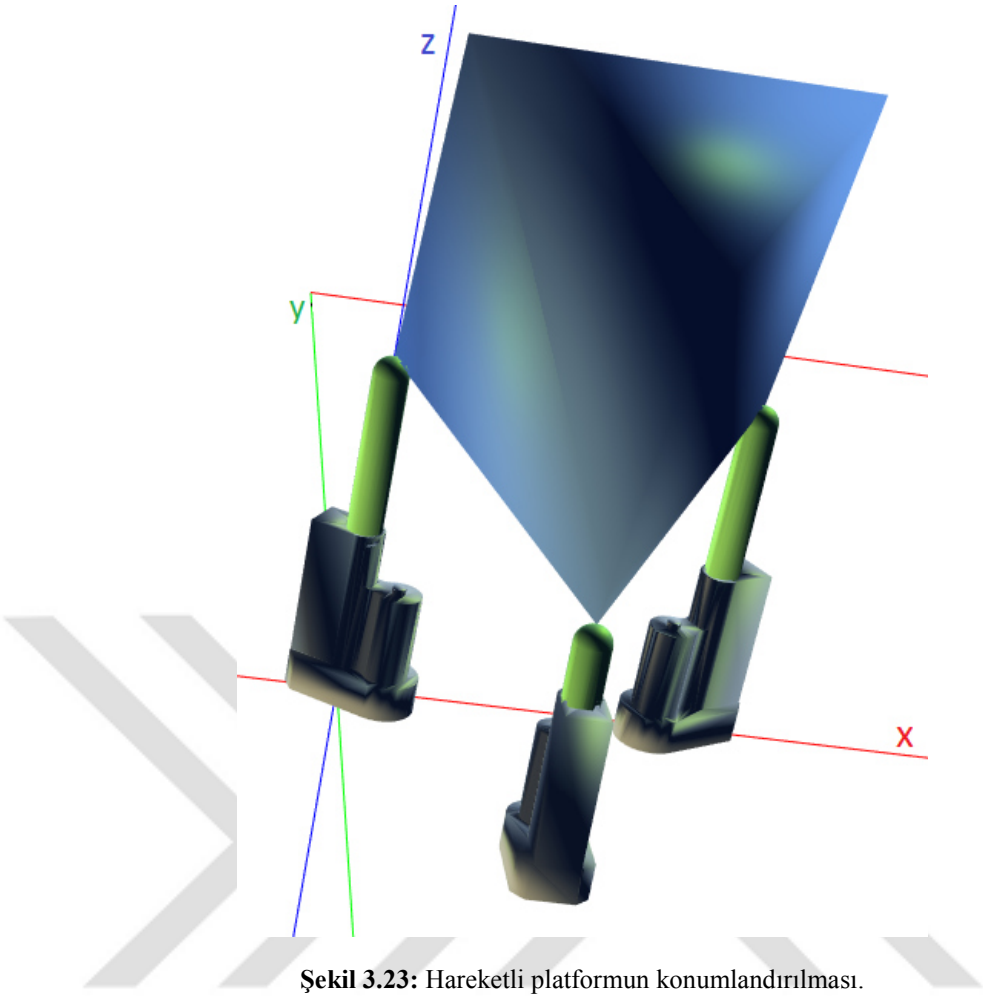
**hareketliPlatformBirinciModul** = mx[1] * mx[0];

mx[1] matrisi tanımlanırken kullanılan Matrix.CreateFromAxisAngle( ) fonksiyonu, Şekil 3.22'de görüldüğü gibi başlangıçta tabanının ağırlık merkezi, referans koordinat sisteminin orijininde bulunan hareketli platformu, belirtilen "uTransformed" dönme ekseni etrafında "-rotAxisAngle" açısı kadar döndürür.




**Şekil 3.22:** Hareketli platformun orijine konumlandırılması.

Daha sonra hareketli platform, lineer aktüatörlerin iç ünitelerinin uç noktalarının koordinatlarının oluşturduğu eşkenar üçgenin ağırlık merkezine mx[0] matrisi ile Şekil 3.23'de görüldüğü gibi ötelenmelidir.



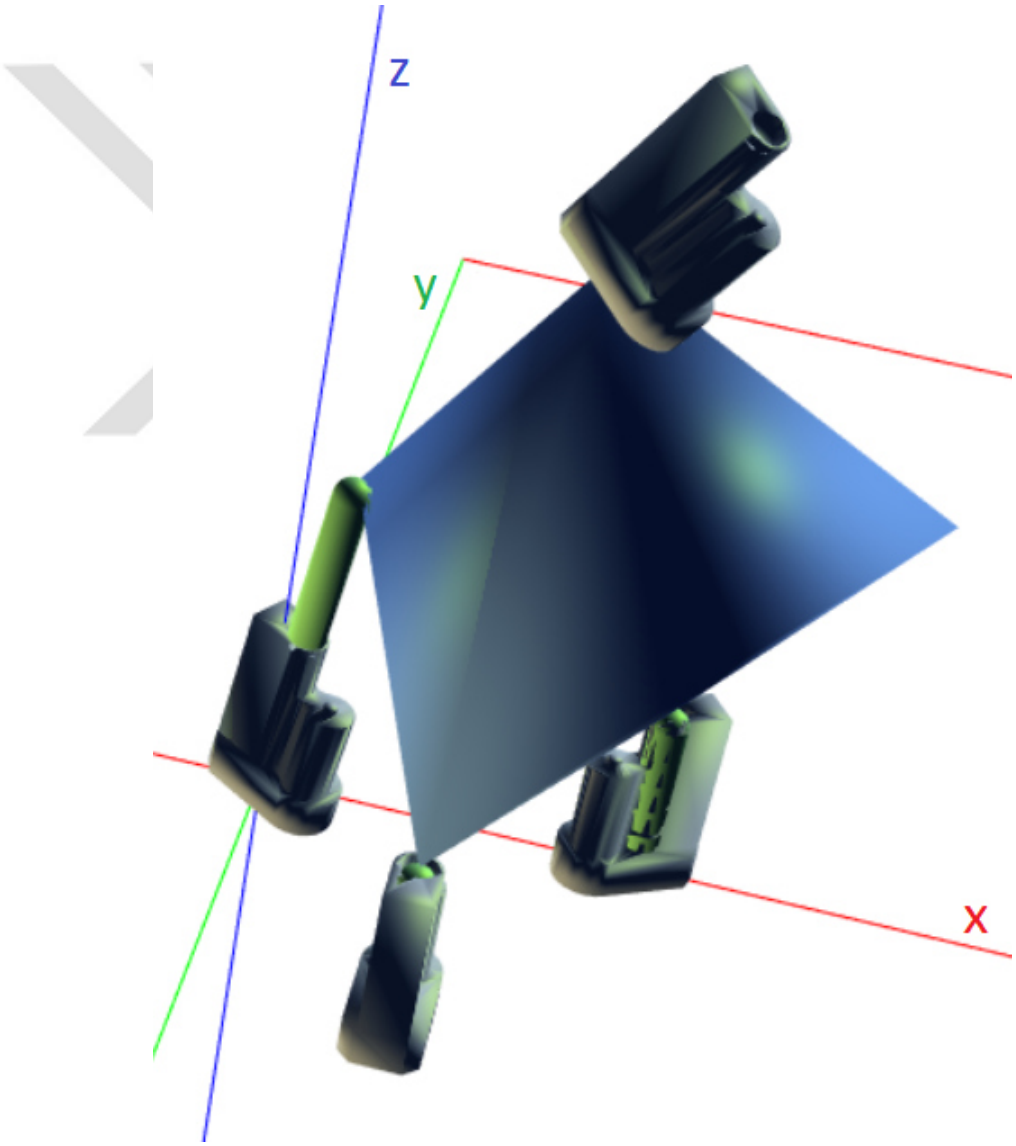

**Şekil 3.23:** Hareketli platformun konumlandırılması.

***İkinci modülün birinci aktüatörü için yazılması gereken kodlar***

**actuatorOut1[1]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle0[1]) * Matrix.CreateRotationZ(Convert.ToSingle(-0.524));
**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**MZ[0]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));
**ikinciModulBirinciAktuator** = actuatorOut1[1] * mx[1] * MZ[0];

İkinci modülün birinci aktüatörünü konumlandırabilmek için öncelikle actuatorOut1[1] kodu ile birlikte aktüatörün gerekli döndürme işlemleri birinci modülün birinci aktüatörü konumundaymış gibi yapılır. İkinci modülün birinci aktüatörü, birinci modülün hareketli platformunun tavanına yerleştirileceği için, bu

platformun dönme işleminden etkilenir. Bu etki, ikinci modülün birinci aktüatörüne daha önce hesaplanan mx[1] matrisi ile yansıtılır. Döndürme işlemleri tamamlanan ikinci modülün birinci aktüatörü, birinci modülün hareketli platformunun tavanına MZ[0] matrisi ile ötelenir. MZ[0] matrisi içerisinde; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içerir. Şekil 3.24'te ikinci modülün birinci aktüatörünün konumlandırılması görülmektedir.

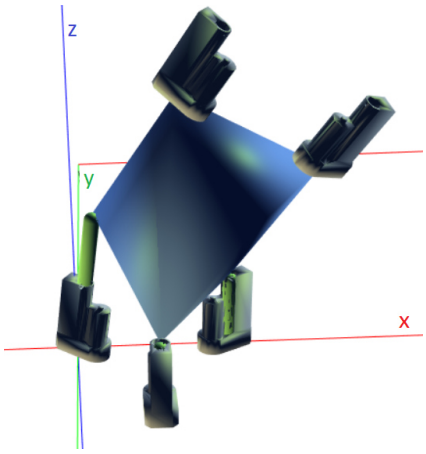



**Şekil 3.24:** İkinci modülün birinci aktüatörünün konumlandırılması.

*İkinci modülün ikinci aktüatörü için yazılması gereken kodlar*

**actuatorOut2[1]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle1[1]) * Matrix.CreateRotationZ(Convert.ToSingle(-2.618));
**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**mx[29]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform2X), -Convert.ToSingle(c.Z1 + c.platform2Z), Convert.ToSingle(c.Y1 + c.platform2Y));
**ikinciModulIkinciAktuator** = actuatorOut2[1] * mx[1] * mx[29];

İkinci modülün ikinci aktüatörünü konumlandırabilmek için öncelikle actuatorOut2[1] kodu ile birlikte aktüatörün gerekli döndürme işlemleri birinci modülün birinci aktüatörü konumundaymış gibi yapılır. İkinci modülün ikinci aktüatörü, birinci modülün hareketli platformunun tavanına yerleştirileceği için, bu platformun dönme işleminden etkilenir. Bu etki, ikinci modülün ikinci aktüatörüne daha önce hesaplanan mx[1] matrisi ile yansıtılır. Döndürme işlemleri tamamlanan ikinci modülün ikinci aktüatörü, birinci modülün hareketli platformunun tavanına mx[29] matrisi ile ötelenir. mx[29] matrisi içerisinde; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan E noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform2X, platform2Y ve platform2Z koordinatlarının bilgisini içerir. Şekil 3.25'te ikinci modülün ikinci aktüatörünün konumlandırılması görülmektedir.




**Şekil 3.25:** İkinci modülün ikinci aktüatörünün konumlandırılması.

*İkinci modülün üçüncü aktüatörü için yazılması gereken kodlar*

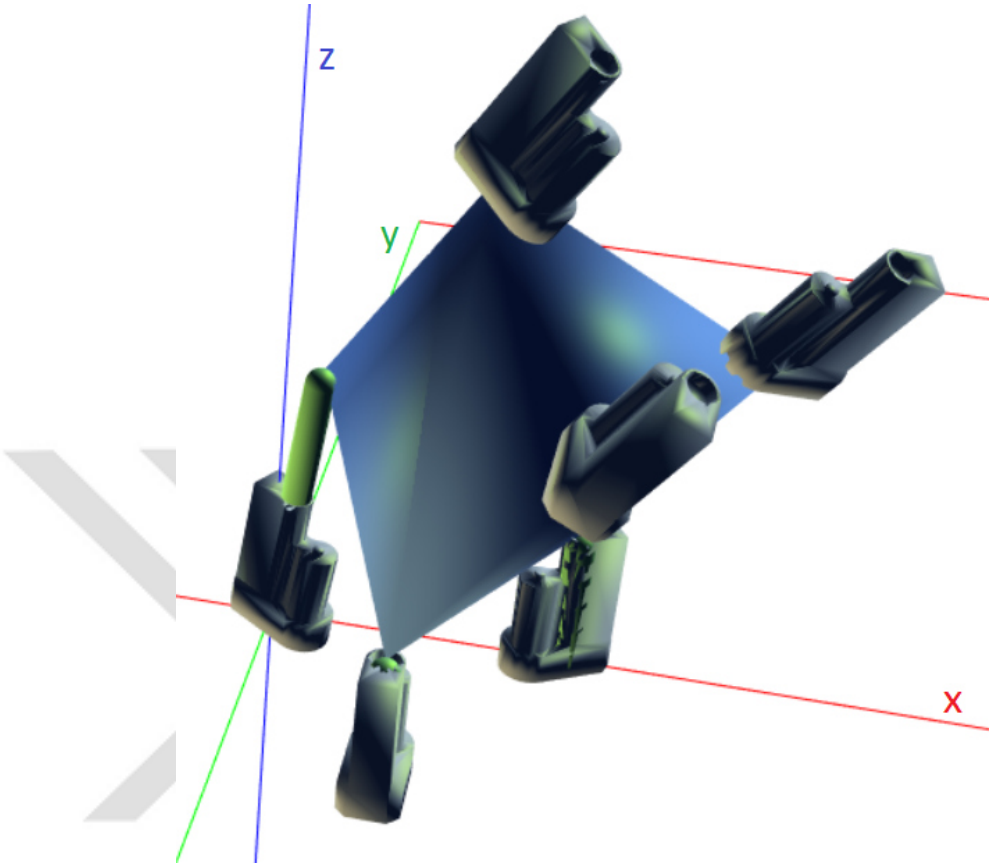
**actuatorOut3[1]** = Matrix.CreateRotationZ(Convert.ToSingle(3.142)) * Matrix.CreateRotationX(angle2[1]);
**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**mx[30]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform3X), -Convert.ToSingle(c.Z1 + c.platform3Z), Convert.ToSingle(c.Y1 + c.platform3Y));
**ikinciModulUcuncuAktuator** = actuatorOut3[1] * mx[1] * mx[30];

İkinci modülün üçüncü aktüatörünü konumlandırabilmek için öncelikle actuatorOut3[1] kodu ile birlikte aktüatörün gerekli döndürme işlemleri birinci modülün birinci aktüatörü konumundaymış gibi yapılır. İkinci modülün üçüncü aktüatörü, birinci modülün hareketli platformunun tavanına yerleştirileceği için, bu platformun dönme işleminden etkilenir. Bu etki, ikinci modülün üçüncü aktüatörüne daha önce hesaplanan mx[1] matrisi ile yansıtılır. Döndürme işlemleri tamamlanan ikinci modülün üçüncü aktüatörü, birinci modülün hareketli platformunun tavanına mx[30] matrisi ile ötelenir. mx[30] matrisi içerisinde; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan F noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform3X, platform3Y ve platform3Z koordinatlarının bilgisini içerir. Şekil 3.26'da ikinci modülün üçüncü aktüatörünün konumlandırılması görülmektedir.

*İkinci Modülün Aktüatörlerinin iç ünitesi için yazılması gereken kodlar*

**actuatorİkinciModulInside1** = Matrix.CreateTranslation(0, 0, length[3]) * ikinciModulBirinciAktuator;
**actuatorİkinciModulInside2** = Matrix.CreateTranslation(0, 0, length[4]) * ikinciModulIkinciAktuator;
**actuatorİkinciModulInside3** = Matrix.CreateTranslation(0, 0, length[5]) * ikinciModulUcuncuAktuator;
length[3], length[4] ve length[5] kodları ikinci modülün sırasıyla birinci, ikinci ve üçüncü aktüatörünün iç ünitesinin uzunluğunu temsil etmektedir. Uzayıp kısalacak

olan iç üniteler olduğundan dolayı bunların konumu butonlar vasıtasıyla referans koordinat sisteminin z ekseninde orijinden başlayarak ötelenir.




**Şekil 3.26:** İkinci modülün üçüncü aktüatörünün konumlandırılması.

Öteleme işleminin hemen ardından; iç ünitelerin, dış ünitelerle birlikte hareket etmesini sağlamak için her bir iç ünite daha önce tanımlanan sırasıyla ikinciModulBirinciAktuator, ikinciModulIkinciAktuator ve ikinciModulUcuncuAktuator matrisleriyle çarpılır. Bu sayede dış ünitelerin konumlarındaki değişiklik eş zamanlı olarak iç ünitelere de aktarılmış olur. İkinci modülün aktüatörlerinin iç üniteleri Şekil 3.27'de görülmektedir.

***İkinci Modülün Hareketli platformunun konumlandırılması için gerekli kodlar***

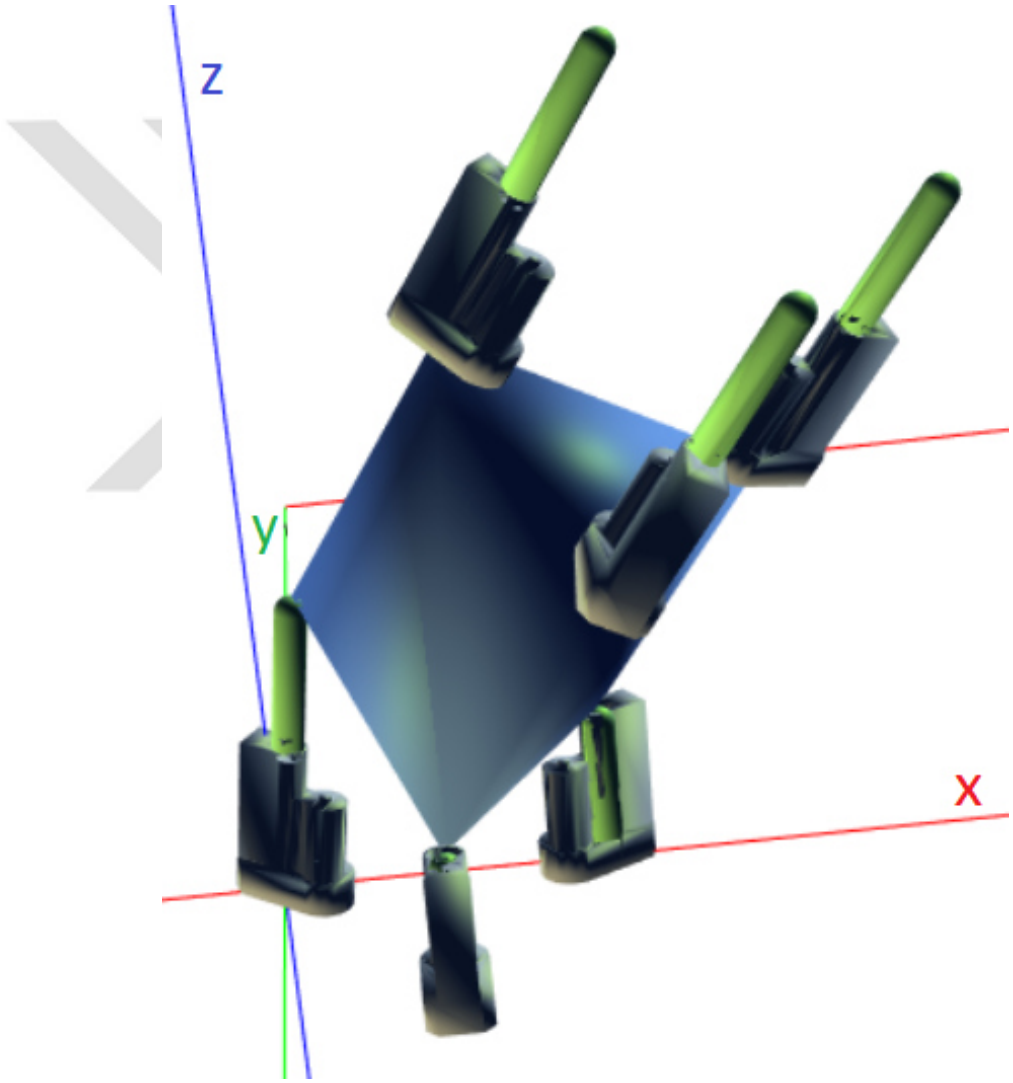
**mx[3]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));

**mx[2]** = Matrix.CreateTranslation(Convert.ToSingle((c.X1 + c.X2 + c.X3) / 3), -Convert.ToSingle((c.Z1 + c.Z2 + c.Z3) / 3), Convert.ToSingle((c.Y1 + c.Y2 + c.Y3) / 3));

**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));

**MZ[0]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));

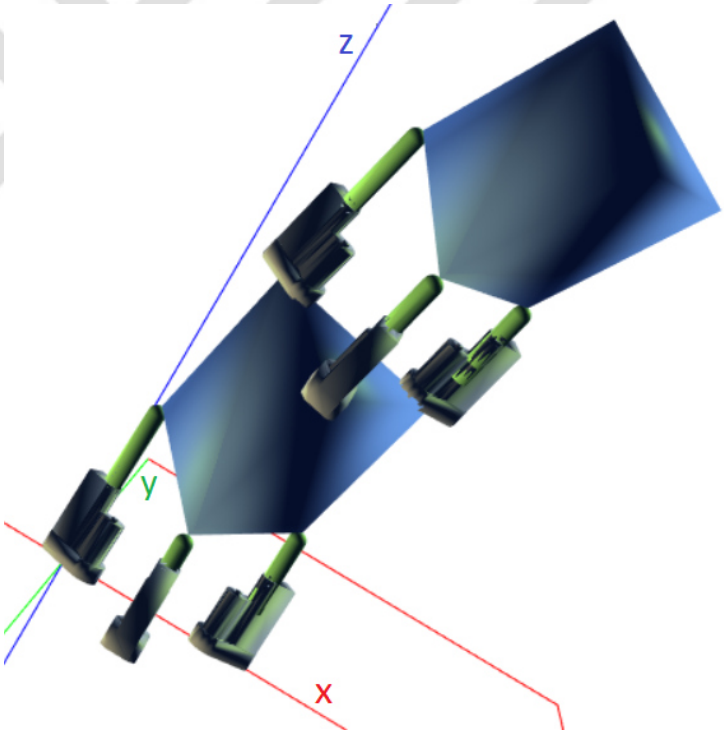
**hareketliPlatformIkinciModul** = mx[3] * mx[2] * mx[1] * MZ[0];




**Şekil 3.27:** İkinci modülün aktüatörlerinin iç ünitelerinin konumlandırılması.

mx[3] matrisi tanımlanırken kullanılan Matrix.CreateFromAxisAngle( ) fonksiyonu, başlangıçta tabanının ağırlık merkezi referans koordinat sisteminin orijininde bulunan ikinci modülün hareketli platformunu, belirtilen "uTransformed" dönme

ekseni etrafında "-rotAxisAngle" açısı kadar döndürür. Daha sonra hareketli platform, ikinci modülün lineer aktüatörlerin iç ünitelerinin uç noktalarının koordinatlarının oluşturduğu eşkenar üçgenin ağırlık merkezine mx[2] matrisi ile ötelenir. Ardından ikinci modülün hareketli platformu, birinci modülün hareketli platformunun dönmesinden etkileneceği için, birinci modülün hareketli platformunun dönüş bilgisini taşıyan mx[1] matrisi ile çarpılır. Bu işlemden sonra ikinci modülün hareketli platformu, gerçek konumuna gelebilmesi için; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içeren MZ[0] matrisi ile çarpılarak ötelenmiş olur. İkinci modülün hareketli platformu Şekil 3.28'de görülmektedir.



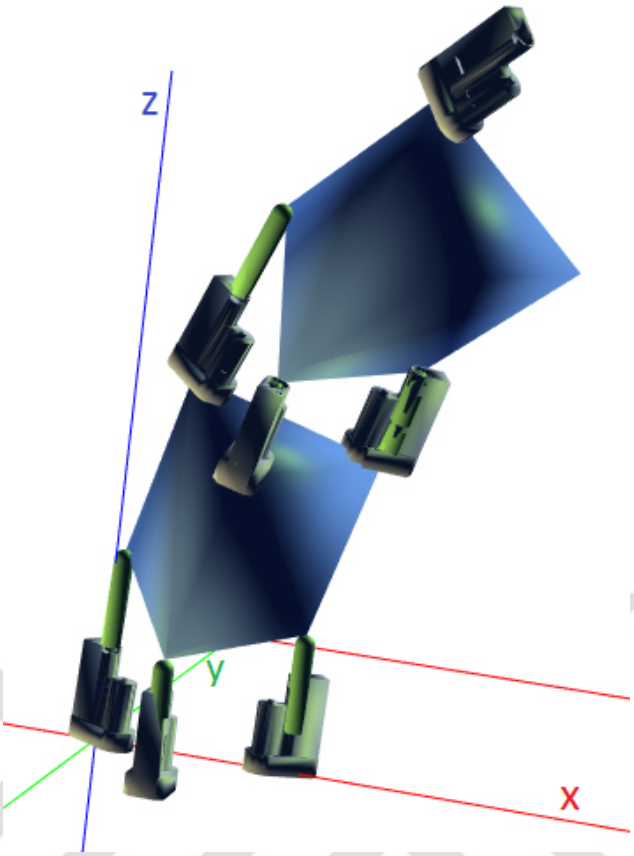

**Şekil 3.28:** İkinci modülün hareketli platformunun konumlandırılması.

***Üçüncü modülün birinci aktüatörü için yazılması gereken kodlar***

**actuatorOut1[2]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle0[2]) * Matrix.CreateRotationZ(Convert.ToSingle(-0.524));

**mx[3]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**MZ[1]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));
**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**MZ[0]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));
**ucuncuModulBirinciAktuator** = actuatorOut1[2] * mx[3] * MZ[1] * mx[1] * MZ[0];

Üçüncü modülün birinci aktüatörünü konumlandırabilmek için öncelikle actuatorOut1[2] kodu ile birlikte aktüatörün gerekli döndürme işlemleri birinci modülün birinci aktüatörü konumundaymış gibi yapılır. Üçüncü modülün birinci aktüatörü, ikinci modülün hareketli platformunun tavanına yerleştirileceği için, bu platformun dönme işleminden etkilenir. Bu etki, üçüncü modülün birinci aktüatörüne daha önce hesaplanan mx[3] matrisi ile yansıtılır. Döndürme işlemleri tamamlanan üçüncü modülün birinci aktüatörü, ikinci modülün hareketli platformunun tavanına MZ[1] matrisi ile ötelenir. MZ[1] matrisi içerisinde; ikinci modülün birinci modülmüş gibi düşünülüp hesaplanan x1, y1 ve z1 koordinatlarını ve ikinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içerir. İkinci modül ve üçüncü modülün birinci aktüatörü, birinci modülün hareketli platformunun dönmesinden etkileneceği için bu etki mx[1] matrisi ile yansıtılır. Son olarak ikinci modül ve üçüncü modülün birinci aktüatörü, birinci modülün hareketli platformunun tavanına MZ[0] matrisi ile ötelenir. MZ[0] matrisi içerisinde; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içerir. Şekil 3.29'da üçüncü modülün birinci aktüatörünün konumlandırılması görülmektedir.




**Şekil 3.29:** Üçüncü modülün birinci aktüatörünün konumlandırılması.

***Üçüncü modülün ikinci aktüatörü için yazılması gereken kodlar***

**actuatorOut2[2]** = Matrix.CreateRotationZ(Convert.ToSingle(1.571)) * Matrix.CreateRotationY(angle1[2]) * Matrix.CreateRotationZ(Convert.ToSingle(-2.618));

**mx[3]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));

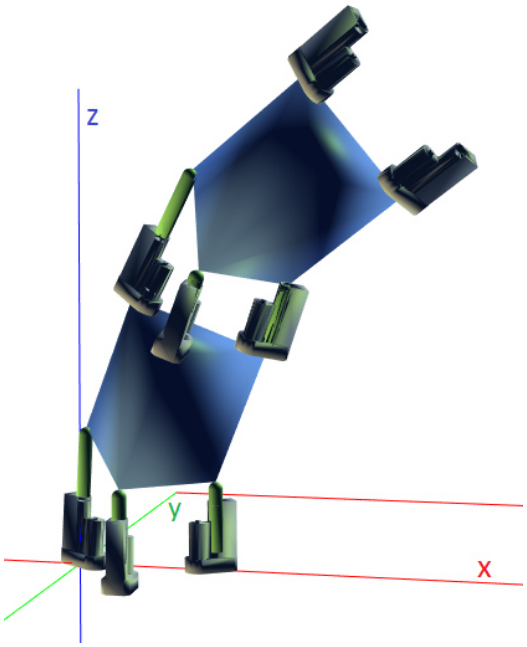
**mx[31]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform2X), -Convert.ToSingle(c.Z1 + c.platform2Z), Convert.ToSingle(c.Y1 + c.platform2Y));

**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));

**MZ[0]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));

**ucuncuModulIkinciAktuator** = actuatorOut2[2] * mx[3] * mx[31] * mx[1] * MZ[0];

Üçüncü modülün ikinci aktüatörünü konumlandırabilmek için öncelikle actuatorOut2[2] kodu ile birlikte aktüatörün gerekli döndürme işlemleri birinci modülün birinci aktüatörü konumundaymış gibi yapılır. Üçüncü modülün ikinci aktüatörü, ikinci modülün hareketli platformunun tavanına yerleştirileceği için, bu platformun dönme işleminden etkilenir. Bu etki, üçüncü modülün birinci aktüatörüne daha önce hesaplanan mx[3] matrisi ile yansıtılır. Döndürme işlemleri tamamlanan üçüncü modülün ikinci aktüatörü, ikinci modülün hareketli platformunun tavanına mx[31] matrisi ile ötelenir. mx[31] matrisi içerisinde; ikinci modülün birinci modülmüş gibi düşünülüp hesaplanan x1, y1 ve z1 koordinatlarını ve ikinci modülün hareketli platformunun tavanında bulunan E noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform2X, platform2Y ve platform2Z koordinatlarının bilgisini içerir. İkinci modül ve üçüncü modülün ikinci aktüatörü, birinci modülün hareketli platformunun dönmesinden etkileneceği için bu etki mx[1] matrisi ile yansıtılır. Son olarak ikinci modül ve üçüncü modülün ikinci aktüatörü, birinci modülün hareketli platformunun tavanına MZ[0] matrisi ile ötelenir. MZ[0] matrisi içerisinde; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içerir. Şekil 3.30'da üçüncü modülün ikinci aktüatörünün konumlandırılması görülmektedir.




**Şekil 3.30:** Üçüncü modülün ikinci aktüatörünün konumlandırılması.

*Üçüncü modülün üçüncü aktüatörü için yazılması gereken kodlar*

**actuatorOut3[2]** = Matrix.CreateRotationZ(Convert.ToSingle(3.142)) * Matrix.CreateRotationX(angle2[2]);
**mx[3]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**mx[32]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform3X), -Convert.ToSingle(c.Z1 + c.platform3Z), Convert.ToSingle(c.Y1 + c.platform3Y));
**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**MZ[0]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));
**ucuncuModulUcuncuAktuator** = actuatorOut3[2] * mx[3] * mx[32] * mx[1] * MZ[0];

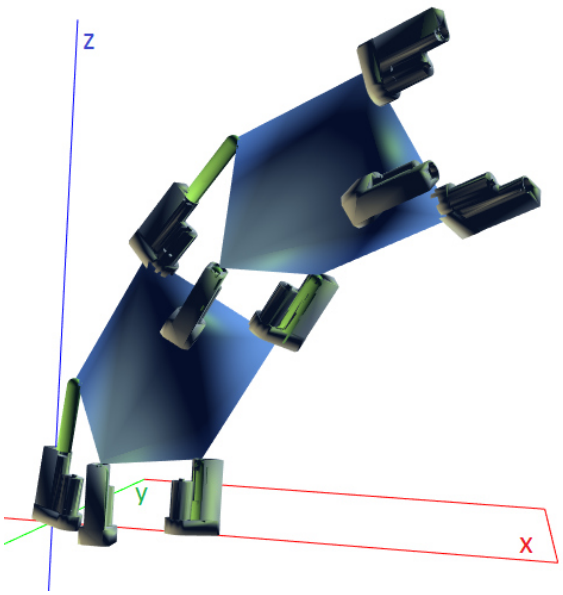
Üçüncü modülün üçüncü aktüatörünü konumlandırabilmek için öncelikle actuatorOut3[2] kodu ile birlikte aktüatörün gerekli döndürme işlemleri birinci modülün birinci aktüatörü konumundaymış gibi yapılır. Üçüncü modülün üçüncü aktüatörü, ikinci modülün hareketli platformunun tavanına yerleştirileceği için, bu platformun dönme işleminden etkilenir. Bu etki, üçüncü modülün üçüncü aktüatörüne daha önce hesaplanan mx[3] matrisi ile yansıtılır. Döndürme işlemleri tamamlanan üçüncü modülün üçüncü aktüatörü, ikinci modülün hareketli platformunun tavanına mx[32] matrisi ile ötelenir. mx[32] matrisi içerisinde; ikinci modülün birinci modülmüş gibi düşünülüp hesaplanan x1, y1 ve z1 koordinatlarını ve ikinci modülün hareketli platformunun tavanında bulunan F noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform3X, platform3Y ve platform3Z koordinatlarının bilgisini içerir. İkinci modül ve üçüncü modülün üçüncü aktüatörü, birinci modülün hareketli platformunun dönmesinden etkileneceği için bu etki mx[1] matrisi ile yansıtılır. Son olarak ikinci modül ve üçüncü modülün üçüncü aktüatörü, birinci modülün hareketli platformunun tavanına MZ[0] matrisi ile ötelenir. MZ[0] matrisi içerisinde; birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini

içerir. Şekil 3.31'de üçüncü modülün üçüncü aktüatörünün konumlandırılması görülmektedir.

***Üçüncü Modülün Aktüatörlerinin iç ünitesi için yazılması gereken kodlar***

**actuatorUcuncuModulInside1** = Matrix.CreateTranslation(0, 0, length[6]) * ucuncuModulBirinciAktuator;
**actuatorUcuncuModulInside2** = = Matrix.CreateTranslation(0, 0, length[7]) * ucuncuModulIkinciAktuator;
**actuatorUcuncuModulInside3** = = Matrix.CreateTranslation(0, 0, length[8]) * ucuncuModulUcuncuAktuator;

length[6], length[7] ve length[8] kodları üçüncü modülün sırasıyla birinci, ikinci ve üçüncü aktüatörünün iç ünitesinin uzunluğunu temsil etmektedir. Uzayıp kısalacak olan iç üniteler olduğundan dolayı bunların konumu butonlar vasıtasıyla referans koordinat sisteminin z ekseninde orijinden başlayarak ötelenir. Öteleme işleminin hemen ardından; iç ünitelerin, dış ünitelerle birlikte hareket etmesini sağlamak için her bir iç ünite daha önce tanımlanan sırasıyla ucuncuModulBirinciAktuator, ucuncuModulIkinciAktuator ve ucuncuModulUcuncuAktuator matrisleriyle çarpılır. Bu sayede dış ünitelerin konumlarındaki değişiklik eş zamanlı olarak iç ünitelere de aktarılmış olur.

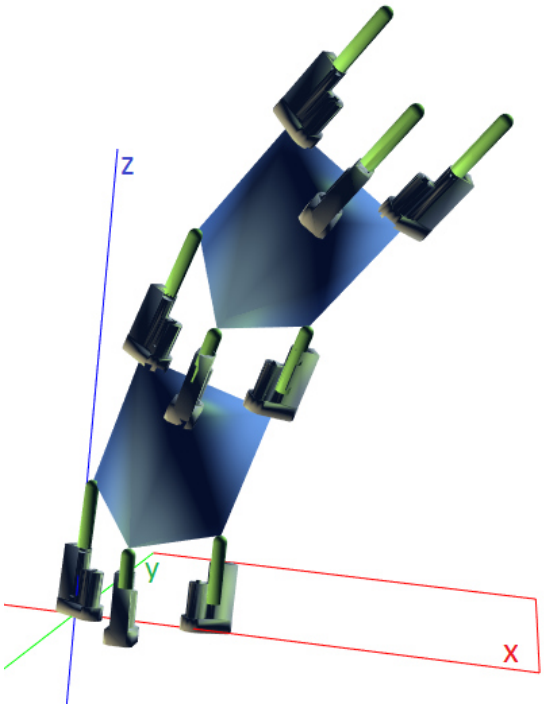



**Şekil 3.31:** Üçüncü modülün üçüncü aktüatörünün konumlandırılması.

Üçüncü modülün aktüatörlerinin iç üniteleri Şekil 3.32'de görülmektedir.

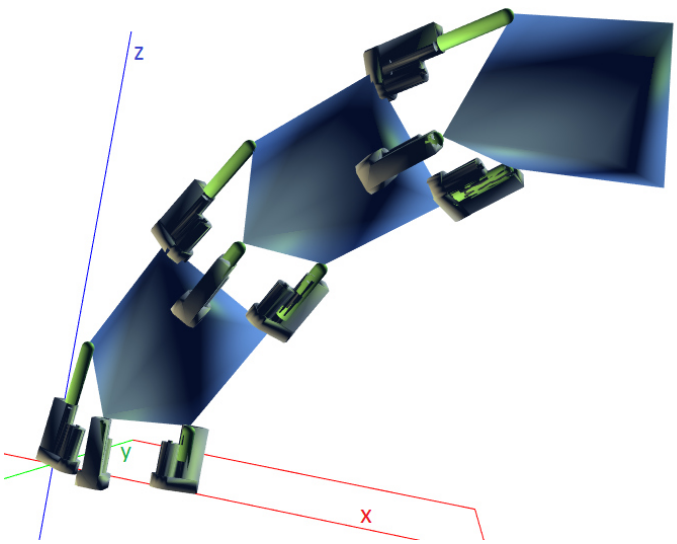
***Üçüncü Modülün Hareketli platformunun konumlandırılması için gerekli kodlar***

**mx[5]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**mx[4]** = Matrix.CreateTranslation(Convert.ToSingle((c.X1 + c.X2 + c.X3) / 3), -Convert.ToSingle((c.Z1 + c.Z2 + c.Z3) / 3), Convert.ToSingle((c.Y1 + c.Y2 + c.Y3) / 3));
**mx[3]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**MZ[1]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));
**mx[1]** = Matrix.CreateFromAxisAngle(uTransformed, Convert.ToSingle(-rotAxisAngle));
**MZ[0]** = Matrix.CreateTranslation(Convert.ToSingle(c.X1 + c.platform1X), -Convert.ToSingle(c.Z1 + c.platform1Z), Convert.ToSingle(c.Y1 + c.platform1Y));
**hareketliPlatformUcuncuModul** = mx[5] * mx[4] * mx[3] * MZ[1] * mx[1] * MZ[0];



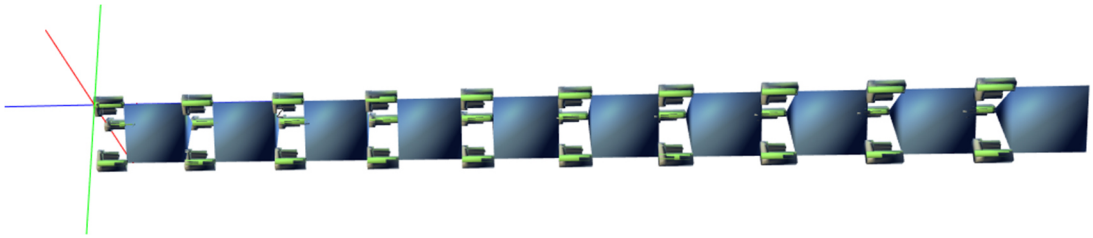

**Şekil 3.32:** Üçüncü modülün aktüatörlerinin iç ünitelerinin konumlandırılması.

mx[5] matrisi tanımlanırken kullanılan Matrix.CreateFromAxisAngle( ) fonksiyonu, başlangıçta tabanının ağırlık merkezi referans koordinat sisteminin orijininde bulunan üçüncü modülün hareketli platformunu, belirtilen "uTransformed" dönme ekseni etrafında "-rotAxisAngle" açısı kadar döndürür. Daha sonra hareketli platform, üçüncü modülün lineer aktüatörlerin iç ünitelerinin uç noktalarının koordinatlarının oluşturduğu eşkenar üçgenin ağırlık merkezine mx[4] matrisi ile ötelenir. Ardından üçüncü modülün hareketli platformu, ikinci modülün hareketli platformunun dönmesinden de etkileneceği için, ikinci modülün hareketli platformunun dönüş bilgisini taşıyan mx[3] matrisi ile çarpılır. Bu işlemden sonra üçüncü modülün hareketli platformu, ikinci modülün orijindeymiş gibi kabul edilerek hesaplanan x1, y1 ve z1 koordinatlarını ve ikinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içeren MZ[1] matrisi ile çarpılarak ötelenir. Üçüncü modülün hareketli platformu, birinci modülün hareketli platformunun dönmesinden de etkileneceği için, birinci modülün hareketli platformunun dönüş bilgisini taşıyan mx[1] matrisi ile çarpılır. Son olarak, üçüncü modülün hareketli platformu, birinci modülün x1, y1 ve z1 koordinatlarını ve birinci modülün hareketli platformunun tavanında bulunan D noktasının A noktası orijin kabul edilerek rotasyon matrisi ile hesaplanan koordinatları olan platform1X, platform1Y ve platform1Z koordinatlarının bilgisini içeren MZ[0] matrisi ile çarpılarak ötelenir. Üçüncü modülün hareketli platformu Şekil 3.33'te görülmektedir.

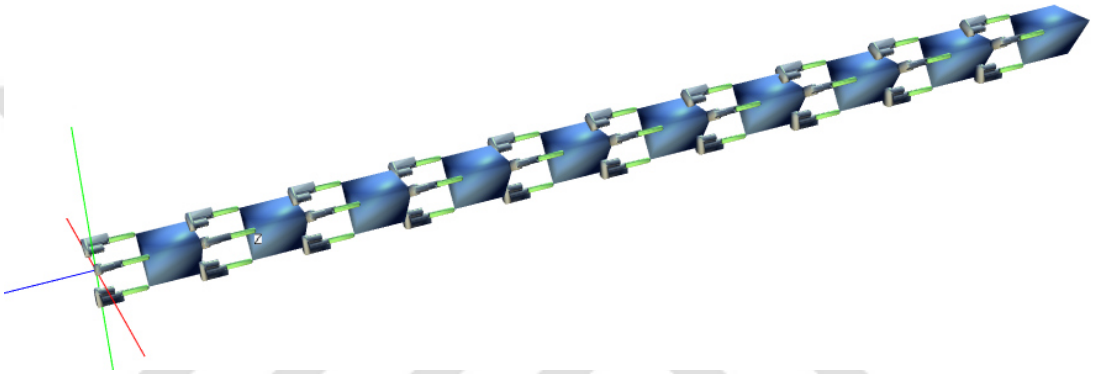



**Şekil 3.33:** Üçüncü modülün hareketli platformunun konumlandırılması.

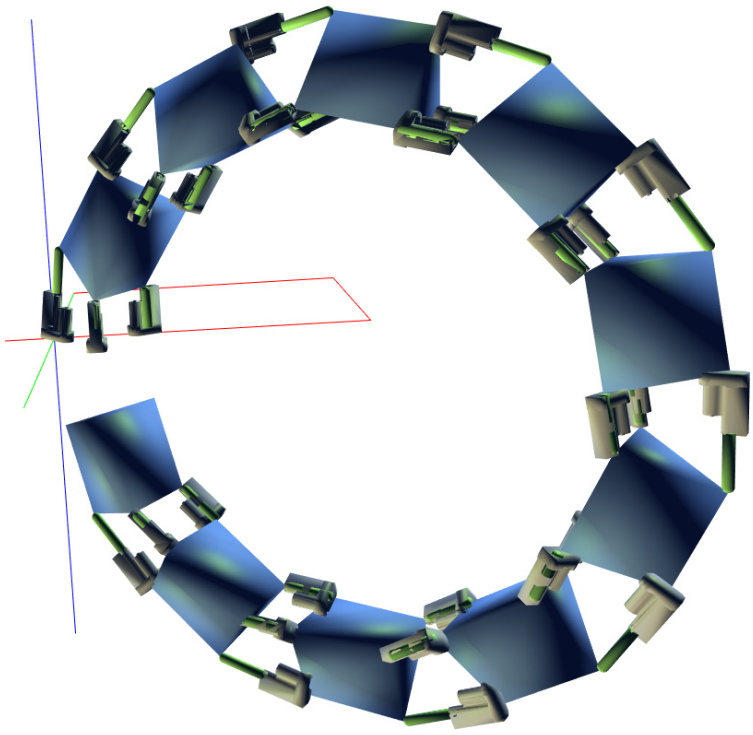
On modüllü yılansı robotun diğer modülleri aynı mantıkla konumlandırılabilir. Yılansı robotun on modülünün de konumlandırılmış haldeki durumları Şekil 3.34a, Şekil 3.34b ve Şekil 3.34c'de görülmektedir.



**Şekil 3.34a:** On modüllü yılansı robotun birinci ekran görüntüsü.

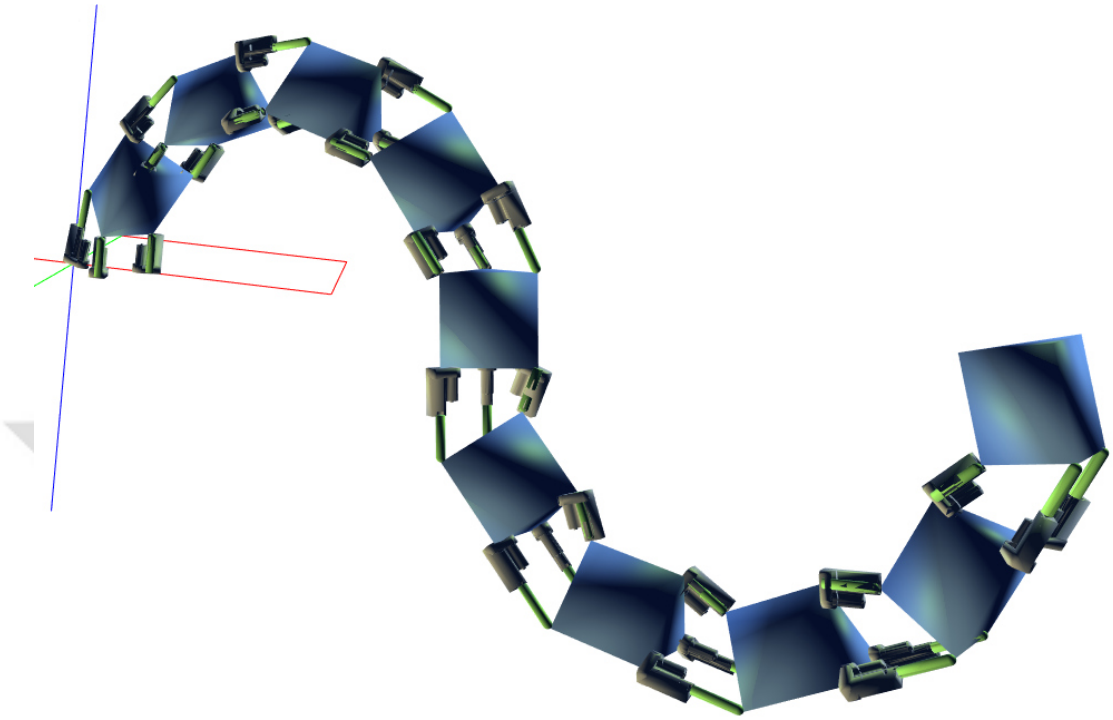


**Şekil 3.34b:** On modüllü yılansı robotun ikinci ekran görüntüsü.

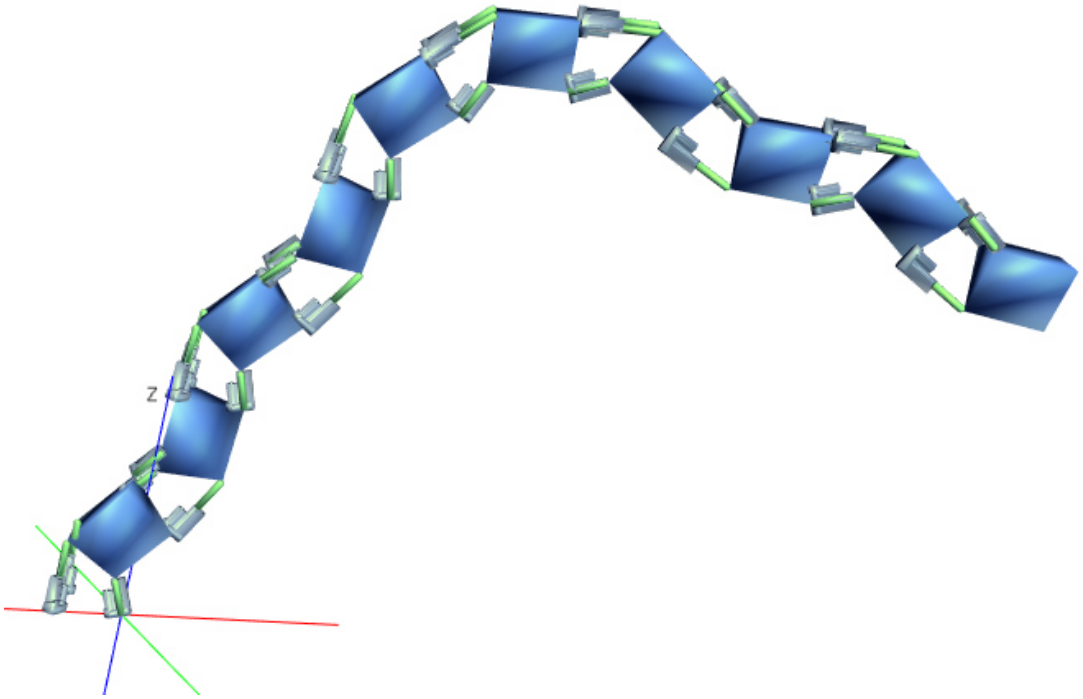


**Şekil 3.34c:** On modüllü yılansı robotun üçüncü ekran görüntüsü.

Yılansı robotun on modülünün de konumlandırılmış haldeki diğer durumları Şekil 3.35a ve Şekil 3.35b'de görülmektedir.

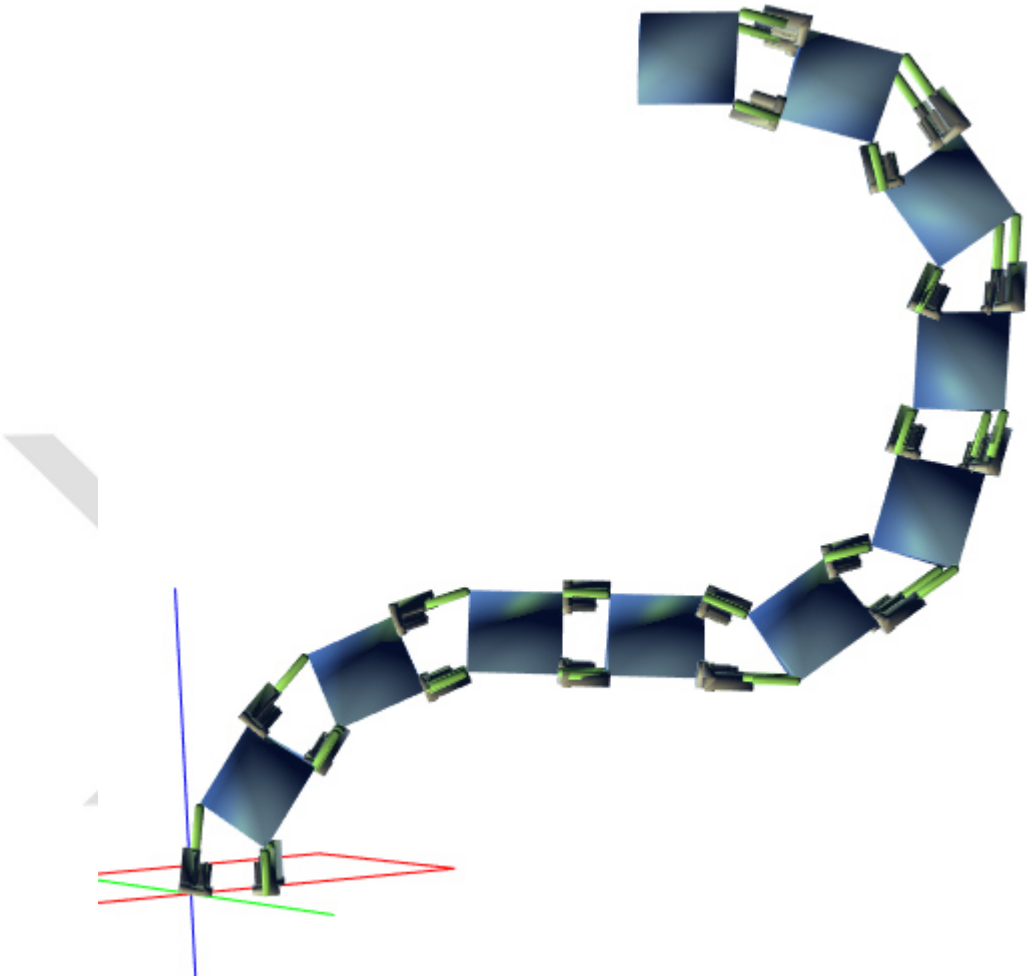


**Şekil 3.35a:** On modüllü yılansı robotun dördüncü ekran görüntüsü.

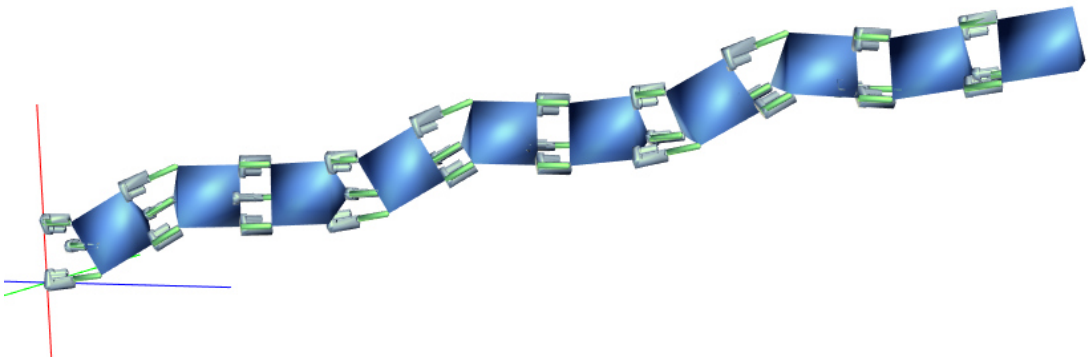



**Şekil 3.35b:** On modüllü yılansı robotun beşinci ekran görüntüsü.

Yılansı robotun on modülünün de konumlandırılmış haldeki diğer durumları Şekil 3.36a ve Şekil 3.36b'de görülmektedir.



**Şekil 3.36a:** On modüllü yılansı robotun altıncı ekran görüntüsü.



**Şekil 3.36b:** On modüllü yılansı robotun yedinci ekran görüntüsü.

# 4. 3-RPS PARALEL MANİPÜLATÖRÜN ÇALIŞMA ALANI ANALİZİ

## 4.1 Giriş

Bu bölümde 3-RPS paralel manipülatörün lineer aktüatörlerinin en uç noktalarının birleştirilmesiyle oluşan eşkenar üçgenin ağırlık merkezinin uzayda erişebileceği noktaların oluşturduğu üç boyutlu şekil çıkarılmaya çalışılacaktır. Yöntemin ne olduğu açıklandıktan sonra Maple yardımıyla çalışma alanının üç boyutlu şekli oluşturulacaktır.

## 4.2 Çalışma Alanının Oluşturulması

3-RPS paralel manipülatörün ileri kinematiğinin bulunabilmesi amacıyla denklem1, denklem2, ve denklem3'ten oluşan bir denklem sistemi oluşturulmuştu. Bu denklem sistemine lineer aktüatör uzunlukları girilerek, hareketli platformun ağırlık merkezinin koordinatları bulunmuştu. Şimdi ise, çalışma alanını bulabilmek için, her bir aktüatörün uzunluğu 150 mm'den başlayarak onar onar 300 mm'ye kadar artırılacaktır. Örneğin L1 ve L2 uzunluğu 150 mm'de sabit bırakılarak, L3 uzunluğu onar mm artırılıp her adımda hareketli platformun ağırlık merkezinin koordinatları hesaplanır ve saklanır. Daha sonra L1 uzunluğu 150 mm de sabit bırakılıp, L2 uzunluğu 10 mm artırılıp 160 mm yapıldıktan sonra L3 uzunluğu 150 mm'den başlayarak yeniden 300 mm'ye onar onar artırılırken her adımda hareketli platformun ağırlık merkezinin koordinatları hesaplanır ve saklanır. Bu mantıkla L1, L2 ve L3 uzunluklarının her biri 300 mm'ye gelene kadar devam edilir. Bu işlemlerdeki amaç 150 mm ve 300 mm dahil olmak üzere, bu iki değer arasında 10 mm aralıktaki tüm L1, L2 ve L3 uzunluklarının kombinasyonlarının hesaba katılmasıdır. Bu sayede 3-RPS manipülatörün çalışma alanının üç boyutlu şekli yaklaşık olarak bulunmuş olur. Şeklin daha hassas elde edilmesi için 10 mm yerine daha küçük aralıklarda artış yapılabilir. Fakat artış miktarının küçülmesi işlem

sayısını artıracaktır. Bu işlemler Maple kullanılarak bir for döngüsü içinde yapılabilir. Bunun için öncelikle, aşağıdaki Maple kodu bir kez çalıştırılmalıdır.

```
> restart:
> with(plots):
> with(ArrayTools):
> with(CurveFitting):
> L1:=300:
> L2:=300:
> #for i from 0 by 10 to 150 do :
#artisL1:=10+i:
#L1:=140+artisL1:

#for j from 0 by 10 to 150 do :
#artisL2:=10+j:
#L2:=140+artisL2:

for k from 0 by 10 to 150 do :
artisL3:=10+k:
L3:=140+artisL3:

denklem1:=proc(α,β,θ)(150-L1*cos(α)*cos(Pi/6))^2+(L3*sin(θ)-
L1*sin(α))^2+(259.808-L3*cos(θ)-L1*cos(α)*sin(Pi/6))^2-90000 end proc:
denklem2:=proc(α, β, θ)(150-300+L2*cos(β)*cos(Pi/6))^2+(L3*sin(θ)-
L2*sin(β))^2+(259.808-L3*cos(θ)-L2*cos(β)*sin(Pi/6))^2-90000 end proc:
denklem3:=proc(α, β, θ)(300-L2*cos(β)*cos(Pi/6)-
L1*cos(α)*cos(Pi/6))^2+(L2*sin(β)-L1*sin(α))^2+(L2*cos(β)*sin(Pi/6)-
L1*cos(α)*sin(Pi/6))^2-90000 end proc:
c:=fsolve([denklem 1, denklem 2, denklem 3],[Pi/6..Pi,Pi/6..Pi,Pi/6..Pi]):
α:=evalf(c[1]*360/(2*Pi)):
β:=evalf(c[2]*360/(2*Pi)):
θ:=evalf(c[3]*360/(2*Pi)):
x1:=evalf(L1*cos(c[1])*cos(Pi/6)):
y1:=L1*sin(c[1]):
z1:=L1*cos(c[1])*sin(Pi/6):
```

```
x2:=evalf(300-L2*cos(c[2])*cos(Pi/6)):
y2:=L2*sin(c[2]):
z2:=L2*cos(c[2])*sin(Pi/6):
x3:=150:
y3:=evalf(L3*sin(c[3])):
z3:=259.808-L3*cos(c[3]):
xort:=(x1+x2+x3)/3:
yort:=(y1+y2+y3)/3:
zort:=(z1+z2+z3)/3:
increase:=1;
xv:=Concatenate(2, Vector[row]([xort]));
yw:=Concatenate(2, Vector[row]([yort]));
zy:=Concatenate(2, Vector[row]([zort]));
xv:=Concatenate(2, Vector[row]([xort]), xv);
yw:=Concatenate(2, Vector[row]([yort]), yw):
zy:=Concatenate(2, Vector[row]([zort]), zy):
increase:=increase+1;
yMatrix:=Matrix(1,increase,zy):
points:={seq([xv[i], yw[i], yMatrix[1, i]],i = 1 .. increase)}:

end do;
#end do;
#end do;
```

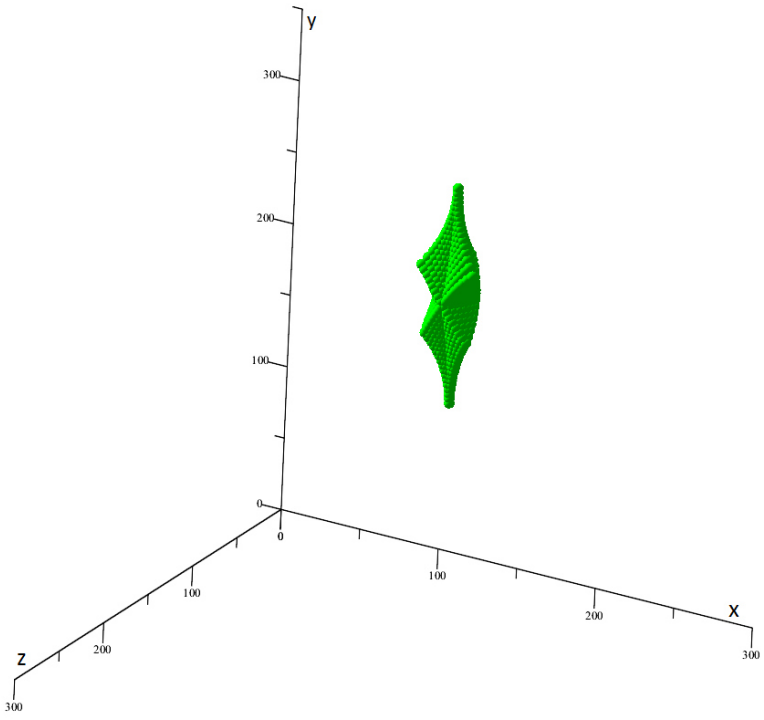
Kodlar bir kez çalıştırıldıktan sonra, “ # ” işareti ile çalışması engellenen satırların engeli kaldırılıp, aşağıdaki kodda görüleceği üzere bazı satırların çalışması “ # ” işareti ile engellenerek ve son satıra üç boyutlu grafiğin çizilmesi için ilgili kod yazılarak yeniden çalıştırılmalıdır.

```
> #restart:
> with(plots):
> with(ArrayTools):
> with(CurveFitting):
> #L1:=300:
> #L2:=300:
```

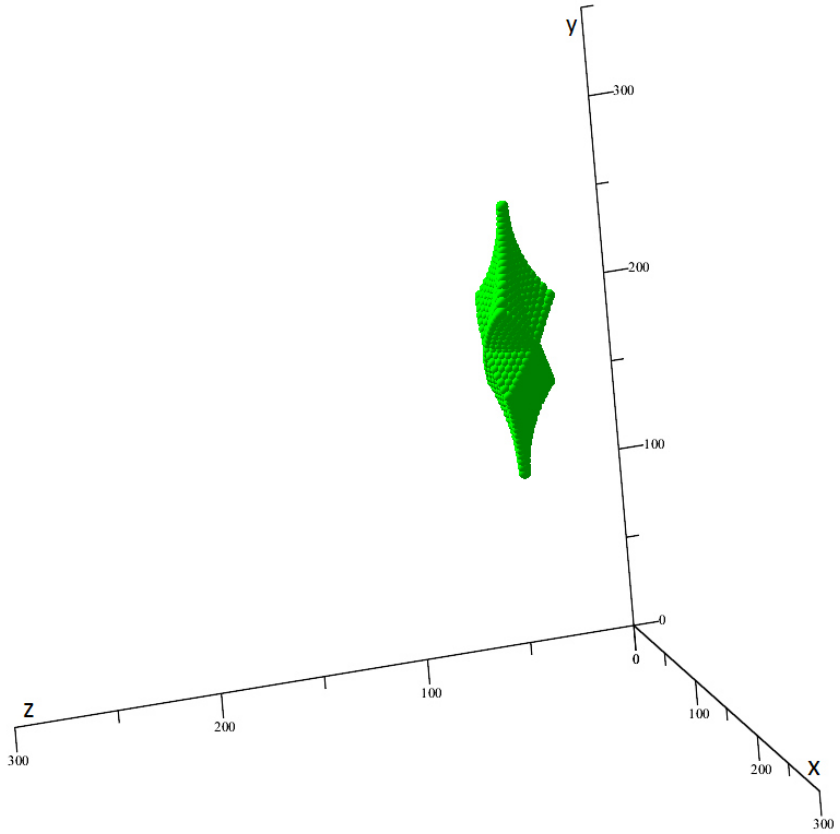
```
> for i from 0 by 10 to 150 do :
artisL1:=10+i:
L1:=140+artisL1:

for j from 0 by 10 to 150 do :
artisL2:=10+j:
L2:=140+artisL2:

for k from 0 by 10 to 150 do :
artisL3:=10+k:
L3:=140+artisL3:
denklem1:=proc(α, β, θ)(150-L1*cos(α)*cos(Pi/6))^2+(L3*sin(θ)-
L1*sin(α))^2+(259.808-L3*cos(θ)-L1*cos(α)*sin(Pi/6))^2-90000 end proc:
denklem2:=proc(α, β, θ)(150-300+L2*cos(β)*cos(Pi/6))^2+(L3*sin(θ)-
L2*sin(β))^2+(259.808-L3*cos(θ)-L2*cos(β)*sin(Pi/6))^2-90000 end proc:
denklem3:=proc(α, β, θ)(300-L2*cos(β)*cos(Pi/6)-
L1*cos(α)*cos(Pi/6))^2+(L2*sin(β)-L1*sin(α))^2+(L2*cos(β)*sin(Pi/6)-
L1*cos(α)*sin(Pi/6))^2-90000 end proc:
c:=fsolve([y1,y2,y3],[Pi/6..Pi,Pi/6..Pi,Pi/6..Pi]):
α:=evalf(c[1]*360/(2*Pi)):
β:=evalf(c[2]*360/(2*Pi)):
θ:=evalf(c[3]*360/(2*Pi)):
x1:=evalf(L1*cos(c[1])*cos(Pi/6)):
y1:=L1*sin(c[1]):
z1:=L1*cos(c[1])*sin(Pi/6):
x2:=evalf(300-L2*cos(c[2])*cos(Pi/6)):
y2:=L2*sin(c[2]):
z2:=L2*cos(c[2])*sin(Pi/6):
x3:=150:
y3:=evalf(L3*sin(c[3])):
z3:=259.808-L3*cos(c[3]):
xort:=(x1+x2+x3)/3:
yort:=(y1+y2+y3)/3:
zort:=(z1+z2+z3)/3:
```

```
#increase:=1;
#xv:=Concatenate(2, Vector[row]([xort]));
#yw:=Concatenate(2, Vector[row]([yort]));
#zy:=Concatenate(2, Vector[row]([zort]));
xv:=Concatenate(2, Vector[row]([xort]), xv);
yw:=Concatenate(2, Vector[row]([yort]), yw):
zy:=Concatenate(2, Vector[row]([zort]), zy):
increase:=increase+1;
yMatrix:=Matrix(1,increase,zy):
points:={seq([xv[i], yw[i], yMatrix[1, i]],i = 1 .. increase)}:
end do;
end do;
end do;

> pointplot3d(points,axes = normal, symbol = solidsphere,color=green,view = [0 ..
300, 0 .. 350, 0..300]);
```

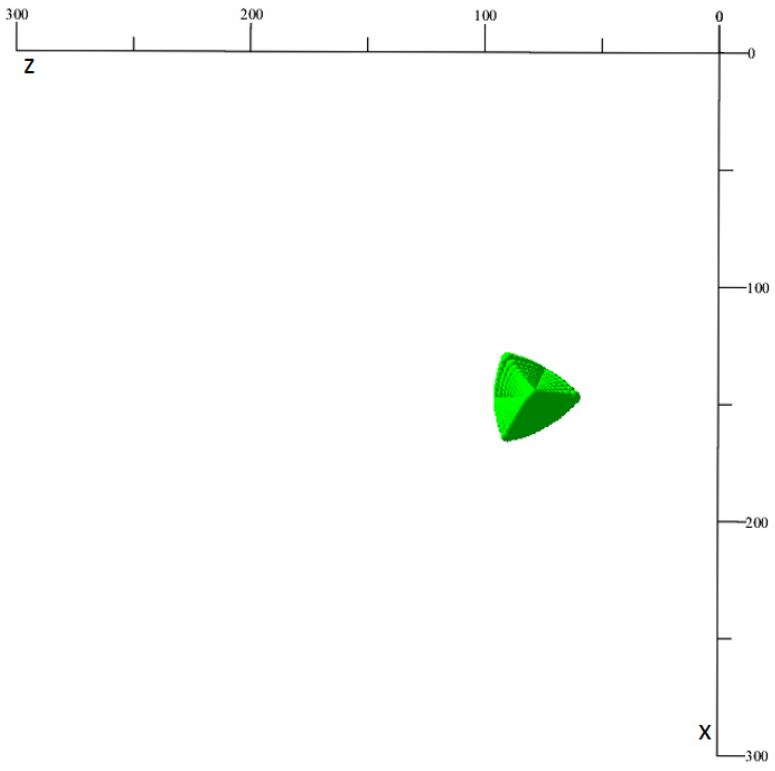
Kodun çalışması sonunda çalışma alanın elde edilen şekli, farklı açılardan Şekil 4.1a, Şekil 4.1b ve Şekil 4.1c'de görülmektedir.




**Şekil 4.1a:** 3-RPS paralel manipülatörün çalışma alanının birinci ekran görüntüsü.




**Şekil 4.1b:** 3-RPS paralel manipülatörün çalışma alanının ikinci ekran görüntüsü.

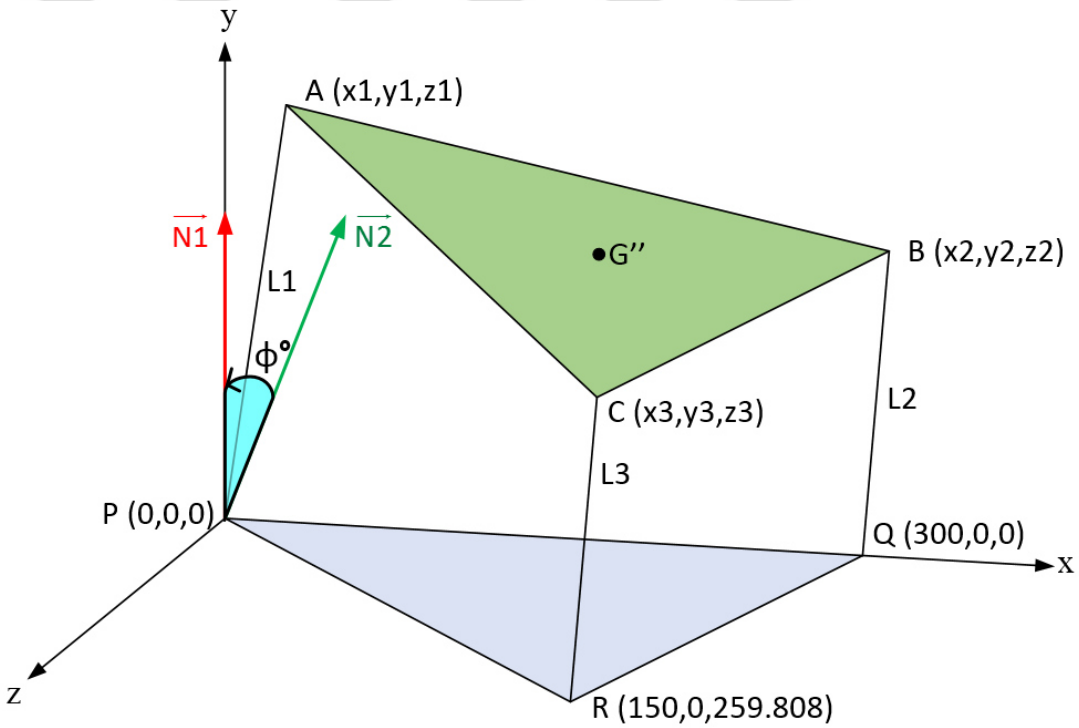



**Şekil 4.1c:** 3-RPS paralel manipülatörün çalışma alanının üçüncü ekran görüntüsü.

# 5. 3-RPS PARALEL MANİPÜLATÖRÜN TERS KİNEMATİĞİ

## 5.1 Giriş

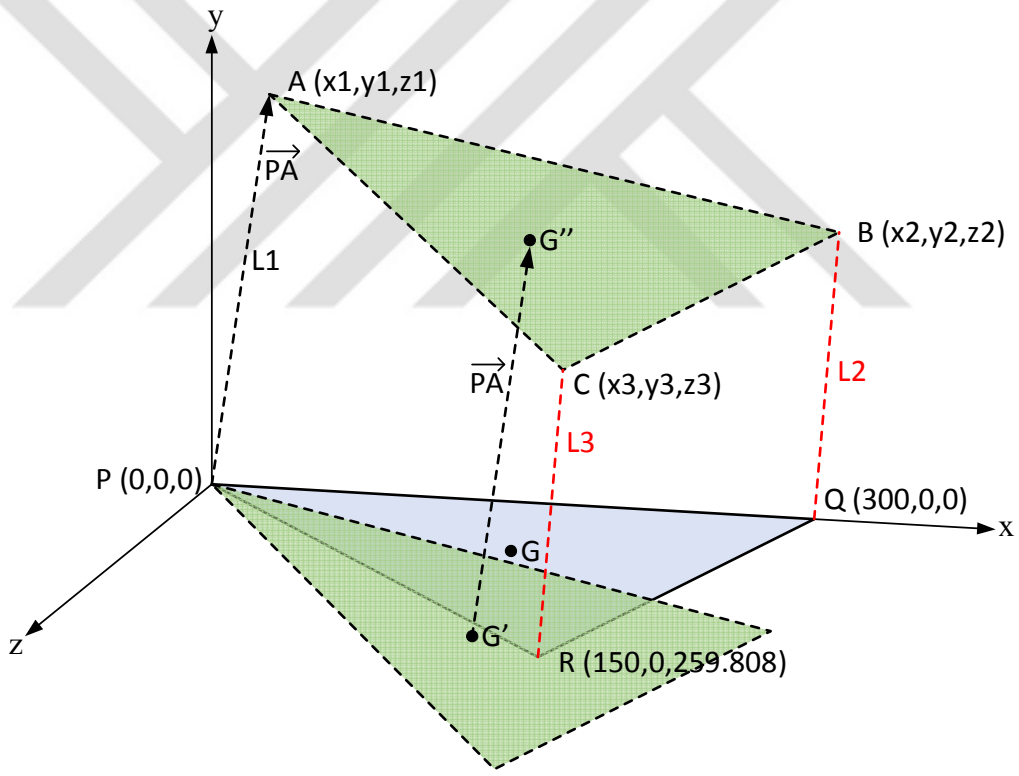
Bu bölümde 3-RPS paralel manipülatörün lineer aktüatörlerinin en uç noktalarının birleştirilmesiyle oluşan eşkenar üçgenin Şekil 5.1'de görüleceği üzere ağırlık merkezi G'' noktasının koordinatı ve platform açısı ϕ verilerek, bu şartları sağlayan lineer aktüatör uzunlukları L1, L2 ve L3 bulunmaya çalışılacaktır.




**Şekil 5.1:** Ters kinematik için gerekli ön bilgiler.

## 5.2 Ters Kinematiğin Çözümü İçin Denklem Sisteminin Oluşturulması

Ters kinematiğin çözümü için denklem sistemi oluşturulacaktır. Bunun için, Şekil 5.2'de görüleceği üzere P, Q ve R noktalarının 3-RPS manipülatörün tabanını oluşturduğu eşkenar üçgenin ağırlık merkezi G noktası, daha önce bu tezin üçüncü bölümünde açıklanan rotasyon matrisi ile çarpılıp üçgenin bu çarpılmaya göre elde edilen yeni ağılık merkezi G' noktası, $\overrightarrow{PA}$ vektörü kadar ötelenirse; L1, L2 ve L3 lineer aktüatör uzunluklarına bağlı olarak, 3-RPS paralel manipülatörün lineer aktüatörlerinin en uç noktalarının birleştirilmesiyle oluşan eşkenar üçgenin ağırlık merkezinin gerçek konumu olan G'' noktasına ulaşılmış olunur.




**Şekil 5.2:** Ters kinematik denklem sistemi için ağırlık merkezinin döndürülüp ötelenmesi.

Öncelikle bu tezin üçüncü bölümünde açıklanan bazı bilgiler yeniden kullanılarak rotasyon matrisi bulunacaktır.

$$\overrightarrow{N1} = \overrightarrow{PR} \times \overrightarrow{PQ} = \begin{pmatrix} 150 \\ 0 \\ 259.808 \end{pmatrix} \times \begin{pmatrix} 300 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 77942.4 \\ 0 \end{pmatrix} \tag{5.1}$$

$$\overrightarrow{AC} = \begin{pmatrix} x3 - x1 \\ y3 - y1 \\ z3 - z1 \end{pmatrix} \tag{5.2}$$

$$\overrightarrow{AB} = \begin{pmatrix} x2 - x1 \\ y2 - y1 \\ z2 - z1 \end{pmatrix} \tag{5.3}$$

$$\overrightarrow{N2} = \overrightarrow{AC} \text{ x } \overrightarrow{AB} \tag{5.4}$$

$$\overrightarrow{N2} = \overrightarrow{AC} \times \overrightarrow{AB} = \begin{pmatrix} x3 - x1 \\ y3 - y1 \\ z3 - z1 \end{pmatrix} \times \begin{pmatrix} x2 - x1 \\ y2 - y1 \\ z2 - z1 \end{pmatrix} \tag{5.5}$$

$$\overrightarrow{N2} = \overrightarrow{AC} \times \overrightarrow{AB} = \begin{bmatrix} (y1 - y3) \cdot (z1 - z2) - (y1 - y2) \cdot (z1 - z3) \\ (x1 - x2) \cdot (z1 - z3) - (x1 - x3) \cdot (z1 - z2) \\ (x1 - x3) \cdot (y1 - y2) - (x1 - x2) \cdot (y1 - y3) \end{bmatrix} \tag{5.6}$$

$$\vec{R} = \overrightarrow{N2} \text{ x } \overrightarrow{N1} \tag{5.7}$$

$$\vec{R} = \begin{bmatrix} (y1 - y3) \cdot (z1 - z2) - (y1 - y2) \cdot (z1 - z3) \\ (x1 - x2) \cdot (z1 - z3) - (x1 - x3) \cdot (z1 - z2) \\ (x1 - x3) \cdot (y1 - y2) - (x1 - x2) \cdot (y1 - y3) \end{bmatrix} \times \begin{pmatrix} 0 \\ 77942.4 \\ 0 \end{pmatrix} \tag{5.8}$$

$$\vec{R} = \begin{bmatrix} 77942.4 \cdot (x1 - x2) \cdot (y1 - y3) + -77942.4 \cdot (x1 - x3) \cdot (y1 - y2) \\ 0 \\ -77942.4 \cdot (y1 - y2) \cdot (z1 - z3) + 77942.4 \cdot (y1 - y3) \cdot (z1 - z2) \end{bmatrix} \tag{5.9}$$

$$Rx = 77942.4(x1 - x2) \cdot (y1 - y3) + -77942.4(x1 - x3) \cdot (y1 - y2) \tag{5.10}$$

$$Ry = 0 \tag{5.11}$$

$$Rz = -77942.4(y1 - y2) \cdot (z1 - z3) + 77942.4(y1 - y3) \cdot (z1 - z2) \tag{5.12}$$

$$\left|\vec{R}\right| = \sqrt{Rx^2 + Ry^2 + Rz^2} \tag{5.13}$$

$$\vec{U} = \frac{\vec{R}}{\left|\vec{R}\right|} \tag{5.14}$$

$$Ux = \frac{Rx}{\left|\vec{R}\right|} \tag{5.15}$$

$$Uy = \frac{Ry}{\left|\vec{R}\right|} \tag{5.16}$$

$$Uz = \frac{Rz}{\left|\vec{R}\right|} \tag{5.17}$$

Bulunan bu değerler yine (5.18) eşitliğindeki rotasyon matrisinde yerine koyulmalıdır.

$$Rot = \begin{bmatrix} \cos(\phi) + Ux^2 \cdot (1 - \cos(\phi)) & Ux \cdot Uy \cdot (1 - \cos(\phi)) - Uz \cdot \sin(\phi) & Ux \cdot Uz \cdot (1 - \cos(\phi)) + Uy \cdot \sin(\phi) \\ Uy \cdot Ux \cdot (1 - \cos(\phi)) + Uz \cdot \sin(\phi) & \cos(\phi) + Uy^2 \cdot (1 - \cos(\phi)) & Uy \cdot Uz \cdot (1 - \cos(\phi)) - Ux \cdot \sin(\phi) \\ Uz \cdot Ux \cdot (1 - \cos(\phi)) - Uy \cdot \sin(\phi) & Uz \cdot Uy \cdot (1 - \cos(\phi)) + Ux \cdot \sin(\phi) & \cos(\phi) + Uz^2 \cdot (1 - \cos(\phi)) \end{bmatrix} \tag{5.18}$$

Ardından, 3-RPS manipülatörün tabanını oluşturan eşkenar üçgenin ağırlık merkezi G noktasının koordinatları bulunmalıdır.

$$xG = \frac{0 + 300 + 150}{3} = 150 \tag{5.19}$$

$$yG = \frac{0 + 0 + 0}{3} = 0 \tag{5.20}$$

$$zG = \frac{0 + 0 + 259.808}{3} = 86.6027 \tag{5.21}$$

G'' noktasının (bkz. Şekil 5.2) koordinatları ise (5.22) eşitliği ile bulunur. G'' noktasının x koordinatına moveX, y koordinatına moveY ve z koordinatına moveZ denilecektir.

$$\begin{pmatrix} moveX \\ moveY \\ moveZ \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & x1 \\ 0 & 1 & 0 & y1 \\ 0 & 0 & 1 & z1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot Rot \cdot \begin{pmatrix} 150 \\ 0 \\ 86.6027 \\ 1 \end{pmatrix} \tag{5.22}$$

Daha önce üçüncü bölümde de bahsedildiği üzere, x3 koordinatının (bkz. Şekil 5.1) uzunluğu daima sabit olup 150 mm'dir. Bilinmeyen olarak geriye x1, x2, y1, y2, y3,

z1, z2, z3 olmak üzere toplam sekiz tane değişken kalır. Sekiz bilinmeyen olduğuna göre bu bilinmeyenlerin bulunabilmesi için bu değişkenleri içeren en az sekiz ayrı denklem elde edilmelidir. Üç denklem; G'' noktasının move,X moveY ve moveZ bileşenlerinden gelir. Ters kinematik için başlangıçtaki G'' noktasının koordinatları xOrt, yOrt ve zOrt bilindiğine göre buradan da (5.23), (5.24) ve (5.25) eşitliklerinde görüldüğü gibi üç denklem elde edilebilir.

$$xOrt = \frac{x1 + x2 + x3}{3} \tag{5.23}$$

$$yOrt = \frac{y1 + y2 + y3}{3} \tag{5.24}$$

$$zOrt = \frac{z1 + z2 + z3}{3} \tag{5.25}$$

Ayrıca yine üçüncü bölümde geometrik sınırlamalardan çıkarılan denklemlerden de (5.26), (5.27) ve (5.28) eşitliklerinde görüldüğü gibi üç adet denklem elde edilir.

$$(x3 - x1)^2 + (y3 - y1)^2 + (z3 - z1)^2 = 300^2 \tag{5.26}$$

$$(x3 - x2)^2 + (y3 - y2)^2 + (z3 - z2)^2 = 300^2 \tag{5.27}$$

$$(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2 = 300^2 \tag{5.28}$$

Yine bu tezin üçüncü bölümünde kullanılan iki vektör arasındaki açıyı bulma formülünden de, (5.29) eşitliğinde görüldüğü gibi bir denklem elde edilebilir.

$$\cos(\phi) = \frac{\overrightarrow{N2} \cdot \overrightarrow{N1}}{\left|\overrightarrow{N2}\right| \cdot \left|\overrightarrow{N1}\right|} \tag{5.29}$$

Şu ana kadar toplamda on denklem elde edilmiş oldu. xOrt, yOrt ve zOrt'un hesap edildiği üç denklemden (5.30), (5.31) ve (5.32) eşitlikleri elde edilebilir. Kalan yedi denklemde x1, y1 ve z1 bileşenleri yerine bu eşitliklerin sağ tarafı konulabilir.

$$x1 = 3*xOrt - x2 - 150 \tag{5.30}$$

$$y1 = 3*yOrt - y2 - y3 \tag{5.31}$$

$$z1 = 3*zOrt - z2 - z3 \tag{5.32}$$

Dolayısıyla denklemlerde bilinmeyen olarak x2, y2, y3, z2 ve z3 olmak üzere beş bilinmeyen kalmış olur. Bu beş bilinmeyen bulunduktan sonra (5.30), (5.31) ve (5.32) eşitliklerinde yerine konularak x1, y1 ve z1 bulunabilir. Beş bilinmeyenin bulunabilmesi için kalan yedi denklemden öncelikle kullanılmak istenen beş denklem seçilmelidir. Bu çalışmada G'' noktasının moveX bileşenli denklemi, üç adet geometrik sınırlama denklemi ve iki vektör arasındaki açıyı hesaplarken kullanılan denklemin kullanılması tercih edilmiştir. Aşağıda Maple kullanılarak, geometrik sınırlama denklemleri hariç kullanılacak diğer denklemler elde edilmiştir.

```
> restart:
> with(LinearAlgebra):
> with(CodeGeneration):
>
> x1:=-x2-150+xOrt*3:
> y1:=-y2-y3+yOrt*3:
> z1:=-z2-z3+zOrt*3:
> x3:=150:
>
> n2AC:=Matrix(3,1,[x3-x1,y3-y1,z3-z1]):
> n2AB:=Matrix(3,1,[x2-x1,y2-y1,z2-z1]):
> n2:=CrossProduct(n2AC,n2AB):
> n1:=Matrix(3,1,[0,77940,0]):
> r:=CrossProduct(n2,n1):
> rx:=-(77940*(x3+x2+150-3*xOrt))*(2*y2+y3-3*yOrt)+(77940*(2*y3+y2-
3*yOrt))*(2*x2+150-3*xOrt): # r vektörünün x bileşeni bu satıra manuel olarak
kopyalanmıştır.
> ry:=0:
> rz:=(77940*(2*y3+y2-3*yOrt))*(2*z2+z3-3*zOrt)-(77940*(2*z3+z2-
3*zOrt))*(2*y2+y3-3*yOrt): # r vektörünün z bileşeni bu satıra manuel olarak
kopyalanmıştır.
> rLength:=(rx^2+ry^2+rz^2)^(1/2):
> ux:=rx/rLength:
> uy:=ry/rLength:
> uz:=rz/rLength:
```

```
> n2Length:=(((y3-y1)*(z2-z1)-(z3-z1)*(y2-y1))^2+(-(x3-x1)*(z2-z1)+(z3-z1)*(x2-
x1))^2+((x3-x1)*(y2-y1)-(y3-y1)*(x2-x1))^2)^(1/2):
> n1Length:=((77940)^2)^(1/2):
> PhiPay:=(-(x3-x1)*(z2-z1)+(z3-z1)*(x2-x1))*77940:
> movePhi:=evalf(PhiPay/(n1Length*n2Length)):
>
> R11:=cos(ϕ)+(ux^2)*(1-cos(ϕ)):
> R12:=ux*uy*(1-cos(ϕ))-uz*sin(ϕ):
> R13:=ux*uz*(1-cos(ϕ))+uy*sin(ϕ):
> R21:=uy*ux*(1-cos(ϕ))+uz*sin(ϕ):
> R22:=cos(ϕ)+(uy^2)*(1-cos(ϕ)):
> R23:=uy*uz*(1-cos(ϕ))-ux*sin(ϕ):
> R31:=uz*ux*(1-cos(ϕ))-uy*sin(ϕ):
> R32:=uz*uy*(1-cos(ϕ))+ux*sin(ϕ):
> R33:=cos(ϕ)+(uz^2)*(1-cos(ϕ)):
> R:=evalf(Matrix(3,3,[R11,R12,R13,R21,R22,R23,R31,R32,R33])):
> baseOrt:=Matrix(3,1,[150,0,86.603]):
> orjinRotate:=evalf(R.baseOrt):
> translationMatrix:=Matrix(4,4,[1,0,0,x1,0,1,0,y1,0,0,1,z1,0,0,0,1]):
> rotationX:=150.*cos(ϕ)+150.*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-
3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2*(1.-1.*cos(ϕ))/((-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)+(86.603*(-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt)))*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))*(1.-1.*cos(ϕ))/((-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2): # orjinRotate
matrisinin x bileşeni bu satıra manuel olarak kopyalanmıştır.
```

> rotationY:=(150.*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt)))*sin(ϕ)/sqrt((-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)-(86.603*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt)))*sin(ϕ)/sqrt((-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2): # orjinRotate matrisinin y bileşeni bu satıra manuel olarak kopyalanmıştır.

>

> rotationZ:=(150.*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt)))*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))*(1.-1.*cos(ϕ))/((-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)+86.603*cos(ϕ)+86.603*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2*(1.-1.*cos(ϕ))/((-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2): # orjinRotate matrisinin z bileşeni bu satıra manuel olarak kopyalanmıştır.

>

> rotationFOUR:=Matrix(4,1,[rotationX,rotationY,rotationZ,1]):

> move:=translationMatrix.rotationFOUR:

>

> **moveX:=**150.*cos(ϕ)+150.*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2*(1.-1.*cos(ϕ))/((-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-

3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)+(86.603*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt)))*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))*(1.-1.*cos(ϕ))/((-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)-150.-x2+3*xOrt: # move matrisinin x bileşeni bu satıra manuel olarak kopyalanmıştır.

> **movePhi:=** (0.1283038234e-4*(-(77940.*(300.+x2-3.*xOrt))*(2.*z2+z3-3.*zOrt)+(77940.*(2.*z3+z2-3.*zOrt))*(2.*x2+150.-3.*xOrt)))/sqrt(((2.*y3+y2-3.*yOrt)*(2.*z2+z3-3.*zOrt)-(1.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2+(-(1.*(300.+x2-3.*xOrt))*(2.*z2+z3-3.*zOrt)+(2.*z3+z2-3.*zOrt)*(2.*x2+150.-3.*xOrt))^2+((300.+x2-3.*xOrt)*(2.*y2+y3-3.*yOrt)-(1.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2)**:** # movePhi bu satıra yukardaki satırından manuel olarak kopyalanmıştır.

### 5.3 Denklem Sisteminin Aralıklı Newton Metodu İle Maple Kullanılarak Çözülmesi

Tek bir denklemin çözümünü bulmak için "Aralıklı Newton Metodu" kullanılabilir (Kananen 2012). İleri kinematiğin çözümü için kullanılan Newton Metodu ile bu denklem sistemi çözülmeye çalışılmış; fakat her bir bilinmeyen için belirli bir aralıkta verilmeyen ilk tahmin değerleri ile çözümlere yakınsanamadığı görülmüştür. Bu yüzden bu tezde; Aralıklı Newton Metodu, beş denklemden oluşan denklem sisteminin çözümü için (5.33) eşitliğindeki gibi kullanılacaktır.

$$\begin{bmatrix} x2(ilk,son)_{(i+1)} \\ y2(ilk,son)_{(i+1)} \\ y3(ilk,son)_{(i+1)} \\ z2(ilk,son)_{(i+1)} \\ z3(ilk,son)_{(i+1)} \end{bmatrix} = \begin{bmatrix} x2\left(\frac{ilk+son}{2}\right)_{(i)} \\ y2\left(\frac{ilk+son}{2}\right)_{(i)} \\ y3\left(\frac{ilk+son}{2}\right)_{(i)} \\ z2\left(\frac{ilk+son}{2}\right)_{(i)} \\ z3\left(\frac{ilk+son}{2}\right)_{(i)} \end{bmatrix} - \left[J_{(ilk,son)}\right]^{-1} \cdot \begin{bmatrix} denklem1\left(\frac{ilk+son}{2}\right)_{(i)} \\ denklem2\left(\frac{ilk+son}{2}\right)_{(i)} \\ denklem3\left(\frac{ilk+son}{2}\right)_{(i)} \\ denklem4\left(\frac{ilk+son}{2}\right)_{(i)} \\ denklem5\left(\frac{ilk+son}{2}\right)_{(i)} \end{bmatrix} \quad (5.33)$$

Denklemin çözümü olabilmesi için, 3-RPS paralel manipülatörün lineer aktüatörlerinin en uç noktalarının birleştirilmesiyle oluşan eşkenar üçgenin ağırlık merkezi G'' noktasının koordinatının dördüncü bölümde bahsedilen çalışma alanı içerisinde olması ve platform açısı $\phi$'nin de G'' noktasını sağlayan açı olması gerekmektedir. Bu yüzden Aralıklı Newton Metodu'nu test edebilmek için öncelikle çalışma alanı içerisinde bir nokta ve bu noktanın platform açısı seçilmelidir.

L1 =250, L2 =200 ve L3 =200 seçilirse (5.34)-(5.46) eşitlikleri arasındaki koordinatlar ve platform açısı elde edilir.

$$x1 = 4.197999824 \quad (5.34)$$

$$y1 = 249.9530004 \quad (5.35)$$

$$z1 = 2.423716328 \quad (5.36)$$

$$x2 = 300 \quad (5.37)$$

$$y2 = 200 \quad (5.38)$$

$$z2 = -2.051033808*10^{-8} \quad (5.39)$$

$$x3 = 150 \quad (5.40)$$

$$y3 = 200 \quad (5.41)$$

$$z3 = 259.8076210 \quad (5.42)$$

Bu bileşenlere göre Aralıklı Newton Metodu'nda kullanılacak değerler;

$$xG'' = 151.3993333 \quad (5.43)$$

$$yG'' = 216.6510002 \quad (5.44)$$

$$zG'' = 87.41044577 \quad (5.45)$$

$$\phi = -0.1934739547 \text{ radyan} \quad (5.46)$$

Birinci tahmin değerlerinin aralıkları;

x2 [299 , 301]
y2 [199 , 201]
y3 [199 , 201]
z2 [-0.5 , 0.5]
z3 [258 , 260]

Birinci tahmin değerlerinin aralıklarının ilk tahmin değerleri kullanılarak Aralıklı Newton Metodu, Maple yardımıyla şu şekilde kodlanabilir:

```
> restart:
> with(LinearAlgebra):
> with(CodeGeneration):
> with(VectorCalculus):
>
> x1:=-x2-150+xOrt*3:
> y1:=-y2-y3+yOrt*3:
> z1:=-z2-z3+zOrt*3:
> phiValue:=evalf(cos(ϕ)):
>
> moveX:= 150.*cos(ϕ)+150.*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-
3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2*(1.-1.*cos(ϕ))/((-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)+(86.603*(-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt)))*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))*(1.-1.*cos(ϕ))/((-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)-150.-x2+3*xOrt:
>
> movePhi:=(0.1283038234e-4*(-(77940.*(300.+x2-3.*xOrt))*(2.*z2+z3-
3.*zOrt)+(77940.*(2.*z3+z2-3.*zOrt))*(2.*x2+150.-3.*xOrt)))/sqrt(((2.*y3+y2-
3.*yOrt)*(2.*z2+z3-3.*zOrt)-(1.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2+(-
(1.*(300.+x2-3.*xOrt))*(2.*z2+z3-3.*zOrt)+(2.*z3+z2-3.*zOrt)*(2.*x2+150.-
3.*xOrt))^2+((300.+x2-3.*xOrt)*(2.*y2+y3-3.*yOrt)-(1.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2):
>
> d1:=moveX-xOrt:
> d2:=movePhi-phiValue:
> d3:=(150-x1)^2+(y3-y1)^2+(z3-z1)^2-90000:
> d4:=(x2-x1)^2+(y2-y1)^2+(z2-z1)^2-90000:
```

```
> d5:=(150-x2)^2+(y3-y2)^2+(z3-z2)^2-90000:
>
> denklemMatrisi:=Matrix(5,1,[d1,d2,d3,d4,d5]):
>
> jacobianMatrisi:=Jacobian([d1,d2,d3,d4,d5],[x2,y2,y3,z2,z3]):
>
> x2ilk:=299:
> x2son:=301:
> x2ave:=evalf((x2ilk+x2son)/2):
>
> y2ilk:=199:
> y2son:=201:
> y2ave:=evalf((y2ilk+y2son)/2):
>
> y3ilk:=199:
> y3son:=201:
> y3ave:=evalf((y3ilk+y3son)/2):
>
> z2ilk:=-0.5:
> z2son:=0.5:
> z2ave:=evalf((z2ilk+z2son)/2):
>
> z3ilk:=258:
> z3son:=260:
> z3ave:=evalf((z3ilk+z3son)/2):
>
> xOrt:=151.3993333:
> yOrt:=216.6510002:
> zOrt:=87.41044577:
> ϕ:=-0.1934739547:
>
> x2:=x2ave:
> y2:=y2ave:
> y3:=y3ave:
```

```
> z2:=z2ave:
> z3:=z3ave:
>
> denklemMatrisi;
>
> denklemMatrisiSon:=Matrix(5,1,[0.0021815,-0.867192e-4,-828.86568,4.56707,-
419]); #denklemMatrisi'nin değerleri manuel olarak bu satıra kopyalanmıştır.
> x2:=x2ilk:
> y2:=y2ilk:
> y3:=y3ilk:
> z2:=z2ilk:
> z3:=z3ilk:
>
> jacobianMatrisiSon:=evalf(jacobianMatrisi):
>
> ilkTahminMatrisi:=Matrix(5,1,[x2ave,y2ave,y3ave,z2ave,z3ave]):
> sonTahminMatrisi:=ilkTahminMatrisi-
MatrixInverse(jacobianMatrisiSon).denklemMatrisiSon;
```

Bu kod çalıştırıldığında son satırda x2, y2, y3, z2, z3 bileşenlerinin yeni aralıklı değerlerinin ilki (5.47)-(5.51) eşitliklerinde görülmektedir.

$$x2 = 299.997625206566 \quad (5.47)$$

$$y2 = 199.947668310266 \quad (5.48)$$

$$y3 = 200.039301657186 \quad (5.49)$$

$$z2 = 0.00651999331684 \quad (5.50)$$

$$z3 = 259.818333704981 \quad (5.51)$$

Birinci tahmin değerlerinin aralıklarının son tahmin değerleri kullanılarak Aralıklı Newton Metodu, Maple yardımıyla aşağıdaki gibi yeniden kodlanır.

```
> restart:
> with(LinearAlgebra):
> with(CodeGeneration):
> with(VectorCalculus):
```

```
> x1:=-x2-150+xOrt*3:
> y1:=-y2-y3+yOrt*3:
> z1:=-z2-z3+zOrt*3:
>
> phiValue:=evalf(cos(ϕ)):
> moveX:= 150.*cos(ϕ)+150.*(-(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-
3.*yOrt)+(77940.*(2.*y3+y2-3.*yOrt))*(2.*x2+150.-3.*xOrt))^2*(1.-1.*cos(ϕ))/((-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)+(86.603*(-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt)))*((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))*(1.-1.*cos(ϕ))/((-
(77940.*(300.+x2-3.*xOrt))*(2.*y2+y3-3.*yOrt)+(77940.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2+((77940.*(2.*y3+y2-3.*yOrt))*(2.*z2+z3-
3.*zOrt)-(77940.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2)-150.-x2+3*xOrt:
>
> movePhi:=(0.1283038234e-4*(-(77940.*(300.+x2-3.*xOrt))*(2.*z2+z3-
3.*zOrt)+(77940.*(2.*z3+z2-3.*zOrt))*(2.*x2+150.-3.*xOrt)))/sqrt(((2.*y3+y2-
3.*yOrt)*(2.*z2+z3-3.*zOrt)-(1.*(2.*z3+z2-3.*zOrt))*(2.*y2+y3-3.*yOrt))^2+(-
(1.*(300.+x2-3.*xOrt))*(2.*z2+z3-3.*zOrt)+(2.*z3+z2-3.*zOrt)*(2.*x2+150.-
3.*xOrt))^2+((300.+x2-3.*xOrt)*(2.*y2+y3-3.*yOrt)-(1.*(2.*y3+y2-
3.*yOrt))*(2.*x2+150.-3.*xOrt))^2):
>
> d1:=moveX-xOrt:
> d2:=movePhi-phiValue:
> d3:=(150-x1)^2+(y3-y1)^2+(z3-z1)^2-90000:
> d4:=(x2-x1)^2+(y2-y1)^2+(z2-z1)^2-90000:
> d5:=(150-x2)^2+(y3-y2)^2+(z3-z2)^2-90000:
>
> denklemMatrisi:=Matrix(5,1,[d1,d2,d3,d4,d5]):
>
> jacobianMatrisi:=Jacobian([d1,d2,d3,d4,d5],[x2,y2,y3,z2,z3]):
```

```
> x2ilk:=299:
> x2son:=301:
> x2ave:=evalf((x2ilk+x2son)/2):
>
> y2ilk:=199:
> y2son:=201:
> y2ave:=evalf((y2ilk+y2son)/2):
>
> y3ilk:=199:
> y3son:=201:
> y3ave:=evalf((y3ilk+y3son)/2):
>
> z2ilk:=-0.5:
> z2son:=0.5:
> z2ave:=evalf((z2ilk+z2son)/2):
>
> z3ilk:=258:
> z3son:=260:
> z3ave:=evalf((z3ilk+z3son)/2):
> xOrt:=151.3993333:
> yOrt:=216.6510002:
> zOrt:=87.41044577:
> ϕ:=-0.1934739547:
>
> x2:=x2ave:
> y2:=y2ave:
> y3:=y3ave:
> z2:=z2ave:
> z3:=z3ave:
>
> denklemMatrisi;
>
> denklemMatrisiSon:=Matrix(5,1,[0.0021815,-0.867192e-4,-828.86568,4.56707,-
419]); #denklemMatrisi'nin değerleri manuel olarak bu satıra kopyalanmıştır.
```

```
> x2:=x2son:
> y2:=y2son:
> y3:=y3son:
> z2:=z2son:
> z3:=z3son:
> jacobianMatrisiSon:=evalf(jacobianMatrisi):
>
> ilkTahminMatrisi:=Matrix(5,1,[x2ave,y2ave,y3ave,z2ave,z3ave]):
> sonTahminMatrisi:=ilkTahminMatrisi-
MatrixInverse(jacobianMatrisiSon).denklemMatrisiSon;
```

Bu kod çalıştırıldığında son satırda x2, y2, y3, z2, z3 bileşenlerinin yeni aralıklı değerlerinin son değerleri (5.52)-(5.56) eşitliklerinde görülmektedir.

$$x2 = 300.002487883803 \quad (5.52)$$

$$y2 = 200.047802601198 \quad (5.53)$$

$$y3 = 199.963794754162 \quad (5.54)$$

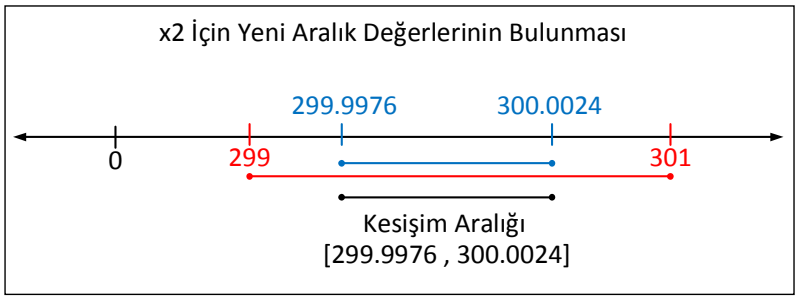
$$z2 = -0.0043260559884 \quad (5.55)$$

$$z3 = 259.801548045844 \quad (5.56)$$

Bu durumda kodlar sonucunda elde edilen değerler büyüklüklerine uygun olarak aralıklara yerleştirildiğinde aşağıdaki yeni aralıklar elde edilmiş olur.

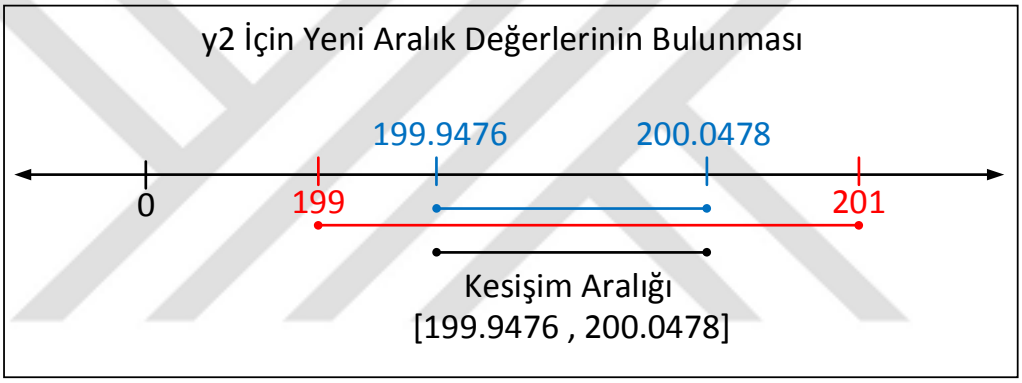
x2 [299.997625206566 , 300.002487883803]
y2 [199.947668310266 , 200.047802601198]
y3 [199.963794754162 , 200.039301657186]
z2 [- 0.0043260559884 , 0.00651999331684]
z3 [259.801548045844 , 259.818333704981]

Birinci tahmin değerleri aralıklarıyla, elde edilen yeni aralıklar sayı doğrusu üzerinde Şekil 5.3'te görüldüğü gibi gösterilebilir.

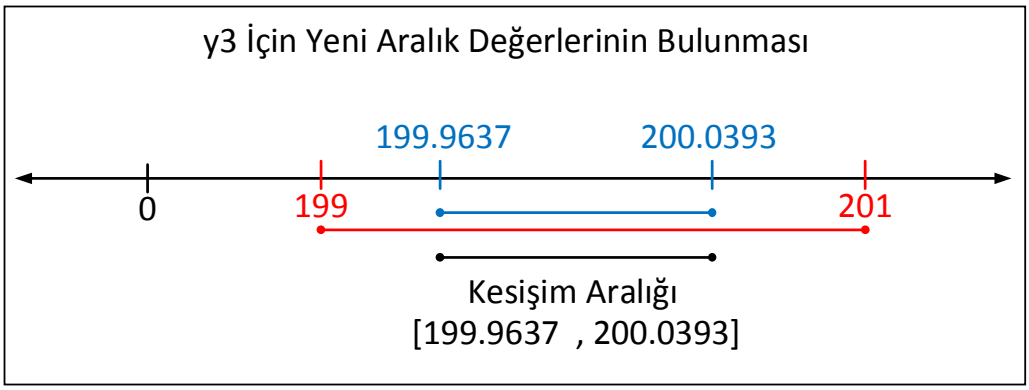



**Şekil 5.3:** x2 aralıklarının sayı doğrusu üzerinde gösterilmesi.

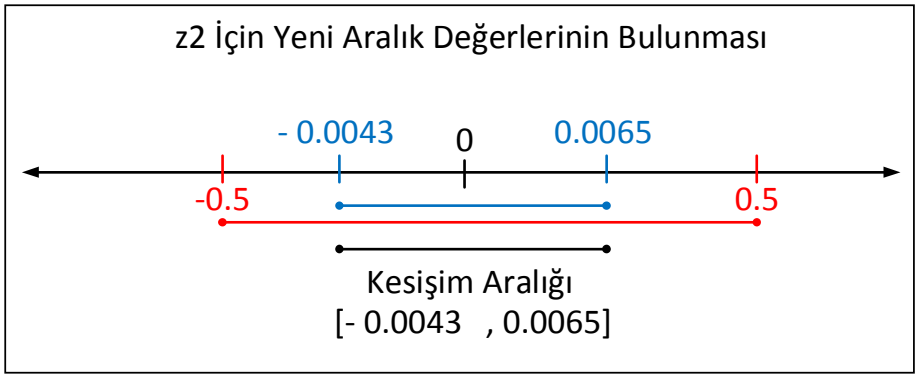
Aralıklı Newton Metodu'na göre kesişim aralığı, x2 için yeni aralık kabul edilir. Diğer bileşenlerin yeni aralıkları Şekil 5.4, Şekil 5.5, Şekil 5.6 ve Şekil 5.7'de görülmektedir.



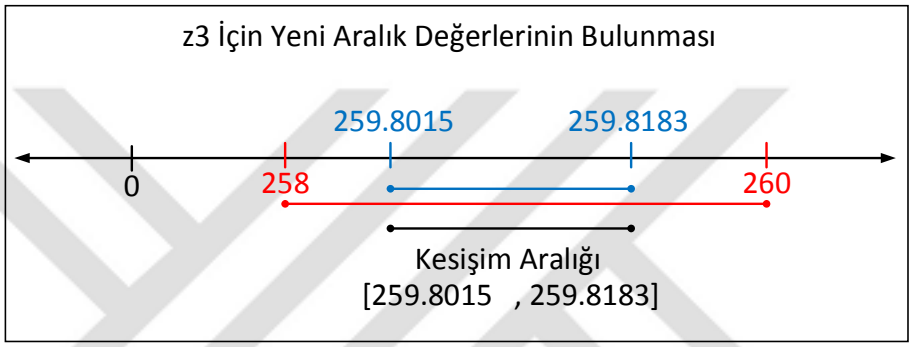

**Şekil 5.4:** y2 aralıklarının sayı doğrusu üzerinde gösterilmesi.




**Şekil 5.5:** y3 aralıklarının sayı doğrusu üzerinde gösterilmesi.




**Şekil 5.6:** z2 aralıklarının sayı doğrusu üzerinde gösterilmesi.




**Şekil 5.7:** z3 aralıklarının sayı doğrusu üzerinde gösterilmesi.

İkinci tahmin değerlerinin aralıkları;

x2 [299.997625206566 , 300.002487883803]
y2 [199.947668310266 , 200.047802601198]
y3 [199.963794754162 , 200.039301657186]
z2 [- 0.0043260559884 , 0.00651999331684]
z3 [259.801548045844 , 259.818333704981]

İkinci tahmin değerlerinin aralıkları ve bu aralıkların ilk ve son tahmin değerleri kullanılarak Aralıklı Newton Metodu, Maple yardımıyla yeniden çalıştırılırsa; x2, y2, y3, z2, z3 bileşenlerinin yeni aralıklı değerlerinin ilk ve son değerleri aşağıdaki gibi bulunmuş olur.

x2 = [299.999987132205 , 299.999990947898]
y2 = [199.999838798608 , 199.999895827956]
y3 = [200.000105320521 , 200.000161523997]

z2 = [0.00000543989693 , 0.00000751505389]
z3 = [259.807631706690 , 259.807636290286]

Bu aralıklar sayı doğrusu üzerinde ikinci tahmin değerlerinin aralıkları ile daha önce yapıldığı gibi kesiştirilirse üçüncü tahmin değerlerinin aralıkları elde edilir.
Üçüncü tahmin değerlerinin aralıkları;

x2 = [299.999987132205 , 299.999990947898]
y2 = [199.999838798608 , 199.999895827956]
y3 = [200.000105320521 , 200.000161523997]
z2 = [0.00000543989693 , 0.00000751505389]
z3 = [259.807631706690 , 259.807636290286]

Üçüncü tahmin değerlerinin aralıkları ve bu aralıkların ilk ve son tahmin değerleri kullanılarak Aralıklı Newton Metodu, Maple yardımıyla yeniden çalıştırılırsa; x2, y2, y3, z2, z3 bileşenlerinin yeni aralıklı değerlerinin ilk ve son değerleri aşağıdaki gibi bulunmuş olur.

x2 = [299.999988901052 , 299.999988901054]
y2 = [199.999865643453 , 199.999865643483]
y3 = [200.000135322355 , 200.000135322385]
z2 = [0.00000659304194 , 0.00000659304309]
z3 = [259.807634153444 , 259.807634153447]

Bu aralıklar sayı doğrusu üzerinde üçüncü tahmin değerlerinin aralıkları ile daha önce yapıldığı gibi kesiştirilirse dördüncü tahmin değerlerinin aralıkları elde edilir.
Dördüncü tahmin değerlerinin aralıkları;

x2 = [299.999988901052 , 299.999988901054]
y2 = [199.999865643453 , 199.999865643483]
y3 = [200.000135322355 , 200.000135322385]
z2 = [0.00000659304194 , 0.00000659304309]
z3 = [259.807634153444 , 259.807634153447]

Bu değerlerden de görüldüğü gibi aralıklar virgülden sonra on haneye kadar yakınsamıştır. Bu yüzden iterasyon sonlandırılabilir. Bu bileşenlerin son değerleri aralıklarının ortalaması alınarak yaklaşık olarak bulunabilir.

$$x2 = (299.999988901052 + 299.999988901054) / 2 = 299.99998890105 \quad (5.57)$$
$$y2 = (199.999865643453 + 199.999865643483) / 2 = 199.99986564346 \quad (5.58)$$
$$y3 = (200.000135322355 + 200.000135322385) / 2 = 200.00013532237 \quad (5.59)$$
$$z2 = (0.00000659304194 + 0.00000659304309) / 2 = 0.0000065930425 \quad (5.60)$$
$$z3 = (259.807634153444 + 259.807634153447) / 2 = 259.80763415344 \quad (5.61)$$

Bu değerler kullanılarak, daha önce bilinen değerler ve formüller yardımıyla x1, y1 ve z1 koordinatları, (5.66)-(5.68) eşitliklerinde görüldüğü gibi hesaplanabilir.

$$xG'' = xOrt = 151.3993333 \quad (5.62)$$
$$yG'' = yOrt = 216.6510002 \quad (5.63)$$
$$zG'' = zOrt = 87.41044577 \quad (5.64)$$
$$x3 = 150 \quad (5.65)$$
$$x1 = -x2 - 150 + xOrt*3 = 4.198010998947 \quad (5.66)$$
$$y1 = -y2 - y3 + yOrt*3 = 249.952999634162 \quad (5.67)$$
$$z1 = -z2 - z3 + zOrt*3 = 2.423696563511 \quad (5.68)$$

Bu değerlere göre lineer aktüatör uzunlukları L1, L2 ve L3 bulunabilir. A(x1, y1, z1) noktası ile P(0, 0, 0) noktası arasındaki uzunluk L1 aktüatörünün uzunluğunu, B(x2, y2, z2) noktası ile Q(300, 0, 0) noktası arasındaki uzunluk L2 aktüatörünün uzunluğunu ve C(x3, y3, z3) noktası ile R(150, 0, 259.808) noktası arasındaki uzunluk L3 aktüatörünün uzunluğunu vermektedir (bkz. Şekil 5.1). Aralıklı Newton Metodu ile bilinmeyen koordinatlar da bulunduğuna göre, iki nokta arasındaki uzaklık bulma formülünden L1, L2 ve L3 aktüatörlerinin uzunlukları hesaplanabilir.

$$L1 = \sqrt{(4.198010998947)^2 + (249.952999634162)^2 + (2.423696563511)^2} \quad (5.69)$$
$$L1 = 249.999999254989 \quad (5.70)$$
$$L2 = \sqrt{(299.999988901053 - 300)^2 + (199.999865643468)^2 + (0.00000659304251)^2} \quad (5.71)$$
$$L2 = 199.999865643468 \quad (5.72)$$
$$L3 = \sqrt{(150 - 150)^2 + (200.00013532237)^2 + (259.807634153446 - 259.808)^2} \quad (5.73)$$

L3 = 200.000135322705 (5.74)

Bu değerlerden de görüldüğü gibi iterasyon sonrası elde edilen değerler, Aralıklı Newton Metodu'nu test edebilmek için çalışma alanı içerisinde önceden verilen bir nokta bulunurken kullanılan lineer aktüatör uzunluklarına ve bu uzunluklara bağlı koordinatlara çok yakın çıkmaktadır. Dolayısıyla olması gereken gerçek değerler arasındaki hata payı kabul edilebilirdir.

## 5.4 Denklem Sisteminin Aralıklı Newton Metodu İle C Sharp Kullanılarak Çözülmesi

Denklem sisteminin C Sharp ortamında çözülebilmesi için; Maple kullanılarak Aralıklı Newton Metodu sembolik olarak ifade edilip elde edilen kodlar C Sharp diline Maple kullanılarak çevrilecektir. Çevrilen kodlar, bu işlemin ardından C Sharp'a aktarılabilir. Beş denklemden oluşan bir denklem sistemi bulunduğu için Jacobian Matrisi 5x5 bir matris olacaktır. Bu matrisin sembolik olarak Maple'da doğrudan tersinin alınması mevcut bilgisayarla mümkün olamamıştır. Bu yüzden öncelikle, 5x5 matrisin tersini alabilecek bir yöntem manuel olarak hazırlanıp Maple'a aktarılmalıdır.

### 5.4.1 Ters Matrisin Gauss-Jordan Yöntemi İle Sembolik Olarak Bulunması

Matrislerin tersinin alınması için Gauss-Jordan Yöntemi kullanılabilir. Bu yöntemde Tablo 5.1'de tersi alınacak olan 5x5 matris, birim matrise dönüştürülürken; dönüştürme kuralından Tablo 5.2'de bulunan 5x5 birim matris de etkilenir. Tablo 5.1'deki matris tamamen birim matrise dönüştürüldüğünde, Tablo 5.2'de yer alan matris ise, başlangıçta tersi alınması planlanan matrisin tersi alınmış haline dönüşmüş olur.

**Tablo 5.1:** Tersi alınacak 5x5 matrisin birinci adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | a11 | a12 | a13 | a14 | a15 |
| 2 | a21 | a22 | a23 | a24 | a25 |
| 3 | a31 | a32 | a33 | a34 | a35 |
| 4 | a41 | a42 | a43 | a44 | a45 |
| 5 | a51 | a52 | a53 | a54 | a55 |

**Tablo 5.2:** 5x5 matrisin tersi alınmış halinin görüleceği matrisin birinci adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1 |

**Tablo 5.3:** Tersi alınacak 5x5 matrisin ikinci adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | b11 | b12 | b13 | b14 | b15 |
| | a11/a11 | a12/a11 | a13/a11 | a14/a11 | a15/a11 |
| 2 | b21 | b22 | b23 | b24 | b25 |
| | a21-a21 | a22-a21*(a12/a11) | a23-a21*(a13/a11) | a24-a21*(a14/a11) | a25-a21*(a15/a11) |
| 3 | b31 | b32 | b33 | b34 | b35 |
| | a31-a31 | a32-a31*(a12/a11) | a33-a31*(a13/a11) | a34-a31*(a14/a11) | a35-a31*(a15/a11) |
| 4 | b41 | b42 | b43 | b44 | b45 |
| | a41-a41 | a42-a41*(a12/a11) | a43-a41*(a13/a11) | a44-a41*(a14/a11) | a45-a41*(a15/a11) |
| 5 | b51 | b52 | b53 | b54 | b55 |
| | a51-a51 | a52-a51*(a12/a11) | a53-a51*(a13/a11) | a54-a51*(a14/a11) | a55-a51*(a15/a11) |

**Tablo 5.4:** 5x5 matrisin tersi alınmış halinin görüleceği matrisin ikinci adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | t11 | 0 | 0 | 0 | 0 |
| | 1/a11 | | | | |
| 2 | t21 | 1 | 0 | 0 | 0 |
| | 0-a21*(1/a11) | | | | |
| 3 | t31 | 0 | 1 | 0 | 0 |
| | 0-a31*(1/a11) | | | | |
| 4 | t41 | 0 | 0 | 1 | 0 |
| | 0-a41*(1/a11) | | | | |
| 5 | t51 | 0 | 0 | 0 | 1 |
| | 0-a51*(1/a11) | | | | |

**Tablo 5.5:** Tersi alınacak 5x5 matrisin üçüncü adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | c11 | c12 | c13 | c14 | c15 |
| | 1 | b12-b12 | b13-b12*(b23/b22) | b14-b12*(b24/b22) | b15-b12*(b25/b22) |
| 2 | c21 | c22 | c23 | c24 | c25 |
| | 0 | b22/b22 | b23/b22 | b24/b22 | b25/b22 |
| 3 | c31 | c32 | c33 | c34 | c35 |
| | 0 | b32-b32 | b33-b32*(b23/b22) | b34-b32*(b24/b22) | b35-b32*(b25/b22) |
| 4 | c41 | c42 | c43 | c44 | c45 |
| | 0 | b42-b42 | b43-b42*(b23/b22) | b44-b42*(b24/b22) | b45-b42*(b25/b22) |
| 5 | c51 | c52 | c53 | c54 | c55 |
| | 0 | b52-b52 | b53-b52*(b23/b22) | b54-b52*(b24/b22) | b55-b52*(b25/b22) |

**Tablo 5.6:** 5x5 matrisin tersi alınmış halinin görüleceği matrisin üçüncü adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | w11 | w12 | 0 | 0 | 0 |
| | t11-b12*(t21/b22) | 0-b12*(1/b22) | | | |
| 2 | w21 | w22 | 0 | 0 | 0 |
| | t21/b22 | 1/b22 | | | |
| 3 | w31 | w32 | 1 | 0 | 0 |
| | t31-b32*(t21/b22) | 0-b32*(1/b22) | | | |
| 4 | w41 | w42 | 0 | 1 | 0 |
| | t41-b42*(t21/b22) | 0-b42*(1/b22) | | | |
| 5 | w51 | w52 | 0 | 0 | 1 |
| | t51-b52*(t21/b22) | 0-b52*(1/b22) | | | |

**Tablo 5.7:** Tersi alınacak 5x5 matrisin dördüncü adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | d11 | d12 | d13 | d14 | d15 |
| | 1 | 0 | c13-c13 | c14-c13*(c34/c33) | c15-c13*(c35/c33) |
| 2 | d21 | d22 | d23 | d24 | d25 |
| | 0 | 1 | c23-c23 | c24-c23*(c34/c33) | c25-c23*(c35/c33) |
| 3 | d31 | d32 | d33 | d34 | d35 |
| | 0 | 0 | c33/c33 | c34/c33 | c35/c33 |
| 4 | d41 | d42 | d43 | d44 | d45 |
| | 0 | 0 | c43-c43 | c44-c43*(c34/c33) | c45-c43*(c35/c33) |
| 5 | d51 | d52 | d53 | d54 | d55 |
| | 0 | 0 | c53-c53 | c54-c53*(c34/c33) | c55-c53*(c35/c33) |

**Tablo 5.8:** 5x5 matrisin tersi alınmış halinin görüleceği matrisin dördüncü adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | x11 | x12 | x13 | 0 | 0 |
| | w11-c13*(w31/c33) | w12-c13*(w32/c33) | 0-c13*(1/c33) | | |
| 2 | x21 | x22 | x23 | 0 | 0 |
| | w21-c23*(w31/c33) | w22-c23*(w32/c33) | 0-c23*(1/c33) | | |
| 3 | x31 | x32 | x33 | 0 | 0 |
| | w31/c33 | w32/c33 | 1/c33 | | |
| 4 | x41 | x42 | x43 | 1 | 0 |
| | w41-c43*(w31/c33) | w42-c43*(w32/c33) | 0-c43*(1/c33) | | |
| 5 | x51 | x52 | x53 | 0 | 1 |
| | w51-c53*(w31/c33) | w52-c53*(w32/c33) | 0-c53*(1/c33) | | |

**Tablo 5.9:** Tersi alınacak 5x5 matrisin beşinci adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | e11 | e12 | e13 | e14 | e15 |
| | 1 | 0 | 0 | d14-d14 | d15-d14*(d45/d44) |
| 2 | e21 | e22 | e23 | e24 | e25 |
| | 0 | 1 | 0 | d24-d24 | d25-d24*(d45/d44) |
| 3 | e31 | e32 | e33 | e34 | e35 |
| | 0 | 0 | 1 | d34-d34 | d35-d34*(d45/d44) |
| 4 | e41 | e42 | e43 | e44 | e45 |
| | 0 | 0 | 0 | d44/d44 | d45/d44 |
| 5 | e51 | e52 | e53 | e54 | e55 |
| | 0 | 0 | 0 | d54-d54 | d55-d54*(d45/d44) |

**Tablo 5.10:** 5x5 matrisin tersi alınmış halinin görüleceği matrisin beşinci adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | y11 | y12 | y13 | y14 | 0 |
| | x11-d14*(x41/d44) | x12-d14*(x42/d44) | x13-d14*(x43/d44) | 0-d14*(1/d44) | |
| 2 | y21 | y22 | y23 | y24 | 0 |
| | x21-d24*(x41/d44) | x22-d24*(x42/d44) | x23-d24*(x43/d44) | 0-d24*(1/d44) | |
| 3 | y31 | y32 | y33 | y34 | 0 |
| | x31-d34*(x41/d44) | x32-d34*(x42/d44) | x33-d34*(x43/d44) | 0-d34*(1/d44) | |
| 4 | y41 | y42 | y43 | y44 | 0 |
| | x41/d44 | x42/d44 | x43/d44 | 1/d44 | |
| 5 | y51 | y52 | y53 | y54 | 1 |
| | x51-d54*(x41/d44) | x52-d54*(x42/d44) | x53-d54*(x43/d44) | 0-d54*(1/d44) | |

**Tablo 5.11:** Tersi alınarak birim matrise dönüştürülmüş 5x5 matrisin altıncı adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | f11 | f12 | f13 | f14 | f15 |
| | 1 | 0 | 0 | 0 | e15-e15 |
| 2 | f21 | f22 | f23 | f24 | f25 |
| | 0 | 1 | 0 | 0 | e25-e25 |
| 3 | f31 | f32 | f33 | f34 | f35 |
| | 0 | 0 | 1 | 0 | e35-e35 |
| 4 | f41 | f42 | f43 | f44 | f45 |
| | 0 | 0 | 0 | 1 | e45-e45 |
| 5 | f51 | f52 | f53 | f54 | f55 |
| | 0 | 0 | 0 | 0 | e55/e55 |

**Tablo 5.12:** 5x5 matrisin tersi alınmış halinin görüldüğü matrisin altıncı adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | z11 | z12 | z13 | z14 | z15 |
| | y11-e15*(y51/e55) | y12-e15*(y52/e55) | y13-e15*(y53/e55) | y14-e15*(y54/e55) | 0-e15*(1/e55) |
| 2 | z21 | z22 | z23 | z24 | z25 |
| | y21-e25*(y51/e55) | y22-e25*(y52/e55) | y23-e25*(y53/e55) | y24-e25*(y54/e55) | 0-e25*(1/e55) |
| 3 | z31 | z32 | z33 | z34 | z35 |
| | y31-e35*(y51/e55) | y32-e35*(y52/e55) | y33-e35*(y53/e55) | y34-e35*(y54/e55) | 0-e35*(1/e55) |
| 4 | z41 | z42 | z43 | z44 | z45 |
| | y41-e45*(y51/e55) | y42-e45*(y52/e55) | y43-e45*(y53/e55) | y44-e45*(y54/e55) | 0-e45*(1/e55) |
| 5 | z51 | z52 | z53 | z54 | z55 |
| | y51/e55 | y52/e55 | y53/e55 | y54/e55 | 1/e55 |

Bu değerler (bkz. Tablo 5.11 ve Tablo 5.12) sırasıyla Tablo 5.13 ve Tablo 5.14'te tekrar gösterilmiştir.

**Tablo 5.13:** Tersi alınarak birim matrise dönüştürülmüş 5x5 matrisin altıncı adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1 |

**Tablo 5.14:** 5x5 matrisin tersi alınmış halinin görüldüğü matrisin altıncı adımı.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | z11 | z12 | z13 | z14 | z15 |
| 2 | z21 | z22 | z23 | z24 | z25 |
| 3 | z31 | z32 | z33 | z34 | z35 |
| 4 | z41 | z42 | z43 | z44 | z45 |
| 5 | z51 | z52 | z53 | z54 | z55 |

Hesaplanan sembolik değerler aşağıdaki kod ile Maple ortamına aktarılarak Newton Metodu sembolik olarak ifade edilir.

```
> restart:
> with(LinearAlgebra):
> with(CodeGeneration):
> b11:=1:
> b12:=a12/a11:
> b13:=a13/a11:
> b14:=a14/a11:
> b15:=a15/a11:
>
> b21:=0:
> b22:=a22-a21*(a12/a11):
> b23:=a23-a21*(a13/a11):
```

```
> b24:=a24-a21*(a14/a11):
> b25:=a25-a21*(a15/a11):
>
> b31:=0:
> b32:=a32-a31*(a12/a11):
> b33:=a33-a31*(a13/a11):
> b34:=a34-a31*(a14/a11):
> b35:=a35-a31*(a15/a11):
>
> b41:=0:
> b42:=a42-a41*(a12/a11):
> b43:=a43-a41*(a13/a11):
> b44:=a44-a41*(a14/a11):
> b45:=a45-a41*(a15/a11):
>
> b51:=0:
> b52:=a52-a51*(a12/a11):
> b53:=a53-a51*(a13/a11):
> b54:=a54-a51*(a14/a11):
> b55:=a55-a51*(a15/a11):
>
> t11:=1/a11:
> t21:=-a21*(1/a11):
> t31:=-a31*(1/a11):
> t41:=-a41*(1/a11):
> t51:=-a51*(1/a11):
>
> c12:=0:
> c13:=b13-b12*(b23/b22):
> c14:=b14-b12*(b24/b22):
> c15:=b15-b12*(b25/b22):
>
> c22:=1:
> c23:=b23/b22:
```

```
> c24:=b24/b22:
> c25:=b25/b22:
>
> c32:=0:
> c33:=b33-b32*(b23/b22):
> c34:=b34-b32*(b24/b22):
> c35:=b35-b32*(b25/b22):
>
> c42:=0:
> c43:=b43-b42*(b23/b22):
> c44:=b44-b42*(b24/b22):
> c45:=b45-b42*(b25/b22):
>
> c52:=0:
> c53:=b53-b52*(b23/b22):
> c54:=b54-b52*(b24/b22):
> c55:=b55-b52*(b25/b22):
>
> w11:=t11-b12*(t21/b22):
> w21:=t21/b22:
> w31:=t31-b32*(t21/b22):
> w41:=t41-b42*(t21/b22):
> w51:=t51-b52*(t21/b22):
>
> w12:=-b12*(1/b22):
> w22:=1/b22:
> w32:=-b32*(1/b22):
> w42:=-b42*(1/b22):
> w52:=-b52*(1/b22):
>
> d13:=0:
> d14:=c14-c13*(c34/c33):
> d15:=c15-c13*(c35/c33):
```

```
> d23:=0:
> d24:=c24-c23*(c34/c33):
> d25:=c25-c23*(c35/c33):
>
> d33:=1:
> d34:=c34/c33:
> d35:=c35/c33:
>
> d43:=0:
> d44:=c44-c43*(c34/c33):
> d45:=c45-c43*(c35/c33):
> d53:=0:
> d54:=c54-c53*(c34/c33):
> d55:=c55-c53*(c35/c33):
>
> d63:=0:
> d64:=c64-c63*(c34/c33):
> d65:=c65-c63*(c35/c33):
>
> x11:=w11-c13*(w31/c33):
> x21:=w21-c23*(w31/c33):
> x31:=w31/c33:
> x41:=w41-c43*(w31/c33):
> x51:=w51-c53*(w31/c33):

> x12:=w12-c13*(w32/c33):
> x22:=w22-c23*(w32/c33):
> x32:=w32/c33:
> x42:=w42-c43*(w32/c33):
> x52:=w52-c53*(w32/c33):
>
> x13:=-c13/c33:
> x23:=-c23/c33:
> x33:=1/c33:
```

```
> x43:=-c43/c33:
> x53:=-c53/c33:
>
> e14:=0:
> e15:=d15-d14*(d45/d44):
>
> e24:=0:
> e25:=d25-d24*(d45/d44):
>
> e34:=0:
> e35:=d35-d34*(d45/d44):
> e44:=1:
> e45:=d45/d44:
>
> e54:=0:
> e55:=d55-d54*(d45/d44):
>
> y11:=x11-d14*(x41/d44):
> y21:=x21-d24*(x41/d44):
> y31:=x31-d34*(x41/d44):
> y41:=x41/d44:
> y51:=x51-d54*(x41/d44):
>
> y12:=x12-d14*(x42/d44):
> y22:=x22-d24*(x42/d44):
> y32:=x32-d34*(x42/d44):
> y42:=x42/d44:
> y52:=x52-d54*(x42/d44):
>
> y13:=x13-d14*(x43/d44):
> y23:=x23-d24*(x43/d44):
> y33:=x33-d34*(x43/d44):
> y43:=x43/d44:
> y53:=x53-d54*(x43/d44):
```

```
> y14:=-d14/d44:
> y24:=-d24/d44:
> y34:=-d34/d44:
> y44:=1/d44:
> y54:=-d54/d44:
>
> f15:=0:
> f25:=0:
> f35:=0:
> f45:=0:
> f55:=1:
>
> z11:=y11-e15*(y51/e55):
> z21:=y21-e25*(y51/e55):
> z31:=y31-e35*(y51/e55):
> z41:=y41-e45*(y51/e55):
> z51:=y51/e55:
>
> z12:=y12-e15*(y52/e55):
> z22:=y22-e25*(y52/e55):
> z32:=y32-e35*(y52/e55):
> z42:=y42-e45*(y52/e55):
> z52:=y52/e55:
>
> z13:=y13-e15*(y53/e55):
> z23:=y23-e25*(y53/e55):
> z33:=y33-e35*(y53/e55):
> z43:=y43-e45*(y53/e55):
> z53:=y53/e55:
>
> z14:=y14-e15*(y54/e55):
> z24:=y24-e25*(y54/e55):
> z34:=y34-e35*(y54/e55):
> z44:=y44-e45*(y54/e55):
```

```
> z54:=y54/e55:
>
> z15:=-e15/e55:
> z25:=-e25/e55:
> z35:=-e35/e55:
> z45:=-e45/e55:
> z55:=1/e55:
>
>jacobianMatrisininTersi:=evalf(Matrix(5,5,[z11,z12,z13,z14,z15,z21,z22,z23,z24,z
25,z31,z32,z33,z34,z35,z41,z42,z43,z44,z45,z51,z52,z53,z54,z55])):
>
denklemMatrisi:=Matrix(5,1,[denklem1,denklem2,denklem3,denklem4,denklem5]):
> ilkTahminOrtalama:=Matrix(5,1,[x2Ave,y2Ave,y3Ave,z2Ave,z3Ave]):
>
> sonTahmin:=ilkTahminOrtalama-jacobianMatrisininTersi.denklemMatrisi:
>
> CSharp(sonTahmin [1,1],resultname=" x2Son ");
> CSharp(sonTahmin [2,1],resultname=" y2Son ");
> CSharp(sonTahmin [3,1],resultname=" y3Son ");
> CSharp(sonTahmin [4,1],resultname=" z2Son ");
> CSharp(sonTahmin [5,1],resultname=" z3Son ");
```

Kod sonundaki değerler Newton Metodu'na göre değişkenlerin son değerlerini sembolik olarak C Sharp diline dönüştürülmüş olarak verirler. Bu ifadelerin Aralıklı Newton Metodu olarak kullanılabilmesi için; denklem1, denklem2, denklem3, denklem4, denklem5'teki x2, y2, y3, z2, z3 değişkenlerinin Maple'da x2Ave, y2Ave, y3Ave, z2Ave, z3Ave olarak kodlanıp, C Sharp'ta kodu çalıştırırken bu kodlara, her bir değişkenin ilk ve son aralıklarının ortalaması girilmelidir. Bu mantıkla ilk tahmin değerlerinin matrisine de Maple kodunda görülebileceği üzere x2, y2, y3, z2, z3 değişkenleri; x2Ave, y2Ave, y3Ave, z2Ave, z3Ave olarak kodlanmıştır. Ayrıca; Maple'da denklem1, denklem2, denklem3, denklem4, denklem5 C Sharp diline çevrildikten sonra C Sharp'a aktarılmalıdır. Jacobian matrisinin oluşturulabilmesi için Maple'da denklem1, denklem2, denklem3, denklem4, denklem5'in x2, y2, y3, z2, z3 değişkenlerine göre türevlerinin sembolik ifadeleri oluşturulup C Sharp diline

çevrildikten sonra C Sharp ortamında a11, a12, a13, a14, a15, a21, a22, a23, a24, a25, a31, a32, a33, a34, a35, a41, a42, a43, a44, a45, a51, a52, a53, a54 ve a55 bileşenlerine (bkz. Tablo 5.1) atanmalıdır. Bu işlemlerden sonra daha önce sayı doğrusu üzerinde gösterilen yeni aralıkları belirleme algoritması C Sharp'ta kodlanarak program çalıştırılmaya hazır hale getirilir.

Tablo 5.15'te Aralıklı Newton Metoduyla çözüm için C Sharp'a aktarılan kod kullanılarak yazılan algoritmayı test etmek amacıyla kullanılacak lineer aktüatör uzunluklarına bağlı olarak elde edilen ağırlık merkezi koordinatları ve platform açıları görülmektedir. Bu koordinat ve platform açılarına bağlı olarak Aralıklı Newton Metodu ile lineer aktüatör uzunlukları elde edilmeye çalışılacaktır.

**Tablo 5.15:** Referans aktüatör uzunlukları, koordinatlar ve platform açıları.

| L1 | L2 | L3 | xOrt (xG") | yOrt (yG") | zOrt (zG") | ϕ (radyan) |
|---|---|---|---|---|---|---|
| 200 | 200 | 250 | 150 | 216.6509976 | 84.98672996 | - 0.1934739266 |
| 200 | 250 | 200 | 148.6006667 | 216.6510002 | 87.41044577 | - 0.1934739589 |
| 200 | 250 | 250 | 151.4053497 | 233.3135803 | 87.41391941 | - 0.1938907379 |
| 250 | 200 | 200 | 151.3993333 | 216.6510002 | 87.41044577 | - 0.1934739547 |
| 250 | 200 | 250 | 148.5946503 | 233.3135803 | 87.41391941 | - 0.1938907379 |
| 250 | 250 | 200 | 150 | 233.3135773 | 84.97978165 | - 0.1938907743 |

Birinci tahmin değerlerinin aralıkları olarak aşağıdaki değerler seçilmiştir. Bu değerlerde x2, z2, z3 daima sabit aralıkta; y2 ve y3, referans ağırlık merkezi koordinatlarını bulurken kullanılan y2 ve y3 değerlerinin -1 ve +1 fazlası olarak seçilmiştir.

x2 [299 , 301]

y2 [$y2_{referans}$-1 , $y2_{referans}$+1]

y3 [$y3_{referans}$-1 , $y3_{referans}$+1]

z2 [-0.5 , 0.5]

z3 [258 , 260]

Bu değerlere göre C Sharp'a aktarılan kod kullanılarak yazılan algoritma çalıştırıldığında Tablo 5.16'daki değerler elde edilmiştir.

**Tablo 5.16:** Aralıklı Newton Metodu ile bulunan lineer aktüatör uzunlukları.

| Referans Lineer Aktüatör Uzunlukları | | | Aralıklı Newton Metodu İle Bulunan Lineer Aktüatör Uzunlukları | | |
|---|---|---|---|---|---|
| L1 | L2 | L3 | L1a | L2a | L3a |
| 200 | 200 | 250 | 200.000003290 | 200.000003290 | 249.999983852 |
| 200 | 250 | 200 | 199.999946904 | 249.998754334 | 200.000054798 |
| 200 | 250 | 250 | 200.000003022 | 250.000137361 | 249.999868729 |
| 250 | 200 | 200 | 250.000000930 | 199.999947373 | 200.000054329 |
| 250 | 200 | 250 | 250.000137862 | 199.998065098 | 249.999868228 |
| 250 | 250 | 200 | 250.000006135 | 250.000006135 | 199.999981782 |

Tablo 5.16'dan görüldüğü üzere Aralıklı Newton Metodu ile bulunan lineer aktüatör uzunlukları, referans lineer aktüatör uzunluklarına çok yakın çıkmıştır. Tablo 5.16'nın ilk sırasındaki referans lineer aktüatör uzunlukları olan L1 = 200, L2 = 200, L3 = 250 ve Aralıklı Newton Metodu ile bulunan lineer aktüatör uzunlukları L1a = 200.000003290, L2a = 200.000003290 ve L3a = 249.999983852 uzunlukları için hesaplanan bağıl hatalar (5.75), (5.76) ve (5.77) eşitliklerinde görülmektedir.

$$\text{Bağıl Hata} = [|(L1a - L1)| / L1]*100 = \% \ 0.00000164 \quad (5.75)$$

$$\text{Bağıl Hata} = [|(L2a - L2)| / L2]*100 = \% \ 0.00000164 \quad (5.76)$$

$$\text{Bağıl Hata} = [|(L3a - L3)| / L3]*100 = \% \ 0.00000645 \quad (5.77)$$

Hata payları kabul edilebilirdir. Her bir çözüm için; kod ilk çalıştırıldığında 56 milisaniyede, devam eden çözümlerde 3 milisaniyede çözüme ulaşılmıştır.

Birinci tahmin değerlerinin aralıkları her zaman sonuca çok yakın seçilemeyebilir. Örneğin, y2 ve y3 değerlerinin birinci tahmin değerlerinin aralıklarının artırıldığı; bu duruma ek olarak aşağıdaki birinci tahmin değerlerinin aralıklarının, farklı ağırlık merkezleri için daima sabit tutulduğu varsayılsın.

x2 [299 , 301]
y2 [200 , 300]
y3 [200 , 300]
z2 [-0.5 , 0.5]
z3 [258 , 260]

Bu değerlere göre C Sharp'a aktarılan kod kullanılarak yazılan algoritma çalıştırıldığında Tablo 5.17'deki değerler elde edilmiştir. Aralıklı Newton Metoduna

göre elde edilen değerlerin bulunmasında Tablo 5.16'daki referans lineer aktüatör uzunlukları aynı sıra ile kullanılmıştır. Tablo 5.17'de gösterilen değerler, önceki değerlerle (bkz. Tablo 5.15) karşılaştırıldığında; ikinci ve üçüncü sıradaki değerlerin birbirine çok yakın olduğu görülmektedir. Birinci, beşinci ve altıncı sıradaki değerlerin aktüatör uzunlukları farklı olmasına rağmen ağırlık merkezi koordinatlarının ve platform açılarının çok yakın olmasa da yakın olduğu söylenebilir. Dördüncü sıradaki değerlerin ise aktüatör uzunlukları farklı olmasına rağmen ağırlık merkezinin koordinatları ve platform açıları birbirine çok yakındır. Bu durumdan, aynı ağırlık merkezi koordinatları ve platform açısı için farklı lineer aktüatör uzunluklarının mevcut olabileceği sonucu çıkarılabilir. Sonuç olarak; her seferde en hassas çözümlere ihtiyaç duyuluyorsa, birinci tahmin değerlerinin aralıklarının çözüme mümkün olduğunca yakın seçilmesinin en uygun olduğu görülmüştür.

**Tablo 5.17:** Aralıklı Newton Metodu ile bulunan lineer aktüatör uzunlukları.

| Aralıklı Newton Metoduna İle Bulunan Lineer Aktüatör Uzunlukları (Referans Lineer Aktüatör Uzunluklarına Göre) | | | Aralıklı Newton Yöntemi İle Bulunan Lineer Aktüatör Uzunluklarına Göre Elde Edilen Değerler | | | |
|---|---|---|---|---|---|---|
| L1b | L2b | L3b | xOrt (xG") | yOrt (yG") | zOrt (zG") | ϕ (radyan) |
| *187.83842832* | *245.51094968* | *216.657020401* | *150.0000545* | *216.6504372* | ***88.21825498*** | *-0.1934681742* |
| 199.99982805 | 249.99887329 | 200.000054665 | 148.6007333 | 216.6505868 | 87.41042201 | -0.1934697798 |
| 200.00000302 | 250.00013736 | 249.999868729 | 151.4053422 | 233.3135834 | 87.41393253 | -0.1938907379 |
| *183.40968771* | *233.34055024* | *233.266856833* | *151.3975555* | *216.6509997* | *87.41418737* | *-0.1934941613* |
| *200.79619623* | *247.57830783* | *251.641597904* | ***151.4430238*** | *233.3207120* | *87.17861474* | ***-0.1897114504*** |
| *200.34779959* | *248.90528416* | *250.729783377* | ***151.4245845*** | *233.3087091* | ***87.30801821*** | ***-0.1918993877*** |

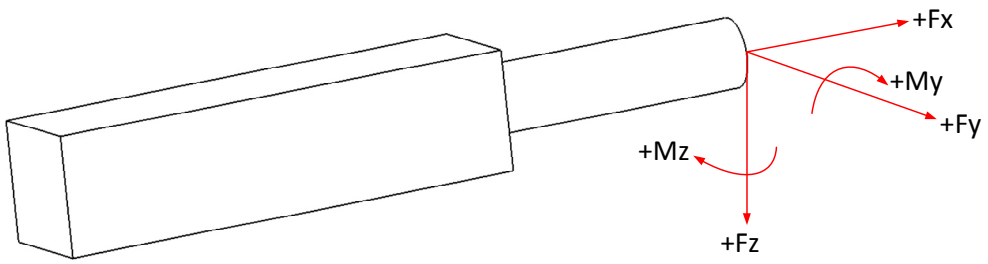
# 6. 10 MODÜLLÜ YILANSI ROBOTUN MEKANİK UYGULANABİLİRLİK ANALİZİ

## 6.1 Giriş

Bu bölümde; on modüllü yılansı robotun tüm lineer aktüatörlerinin maksimum stroklu yatay konumlarında, kendi ağırlığı altında, birinci modülün lineer aktüatörlerinin uç noktalarında meydana getirdiği reaksiyon kuvvetlerinin ve reaksiyon momentlerinin statik analizi Ansys kullanılarak yapılacaktır. Elde edilen sonuçlar; aktüatörleri üreten firmanın, aktüatörlerin taşıyabileceği maksimum yükler hakkında sağladığı teknik verilere göre değerlendirilecektir. Bu değerlendirme sonucunda, aktüatörlerin bu şartlar altında dayanıp dayanamayacağı anlaşılacaktır.

## 6.2 Reaksiyon Kuvvetleri ve Momentlerinin Ansys Kullanılarak Analiz Edilmesi

Statik analizlerin Ansys'te yapılabilmesi için öncelikle Catia Kullanılarak on modüllü yılansı robotun tasarımı yapılıp stp uzantılı dosya olarak kaydedilmiştir. Bu dosya Ansys'te import edilerek reaksiyon kuvveti ve reaksiyon momenti analizi için gerekli işlemler yapılıp çözülmüştür. Elde edilen sonuçlar, lineer aktüatörleri imal eden Endo Endüstriyel Donanım ve Otomasyon Sistemleri Tic. Ltd. Şti. tarafından sağlanan Şekil 6.1 ve Tablo 6.1'deki verilere göre yorumlanacaktır.

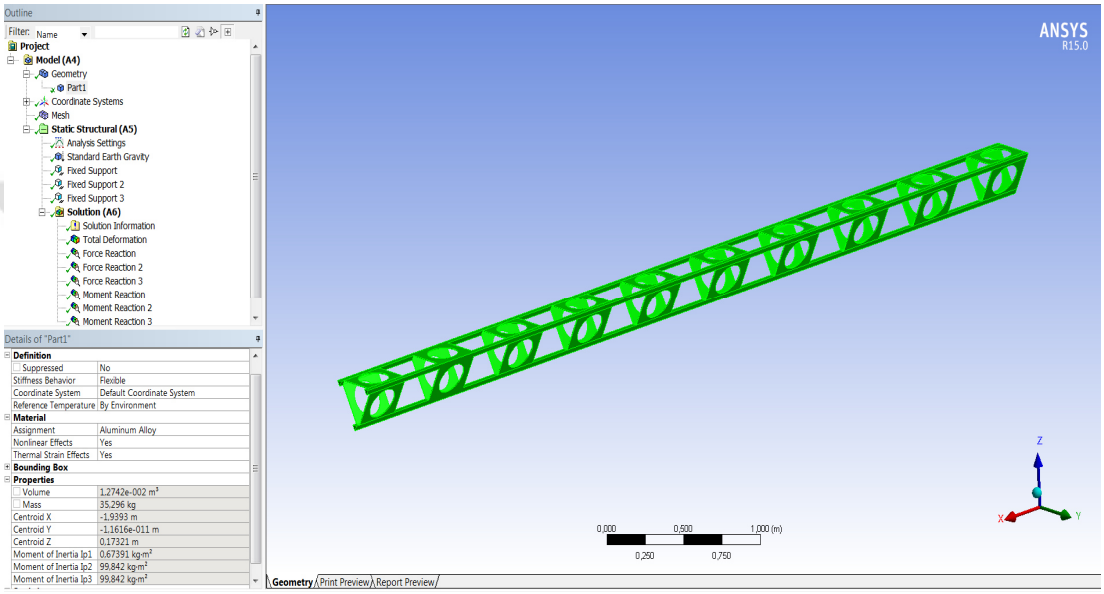



**Şekil 6.1:** Yük eksenlerinin tanımı.

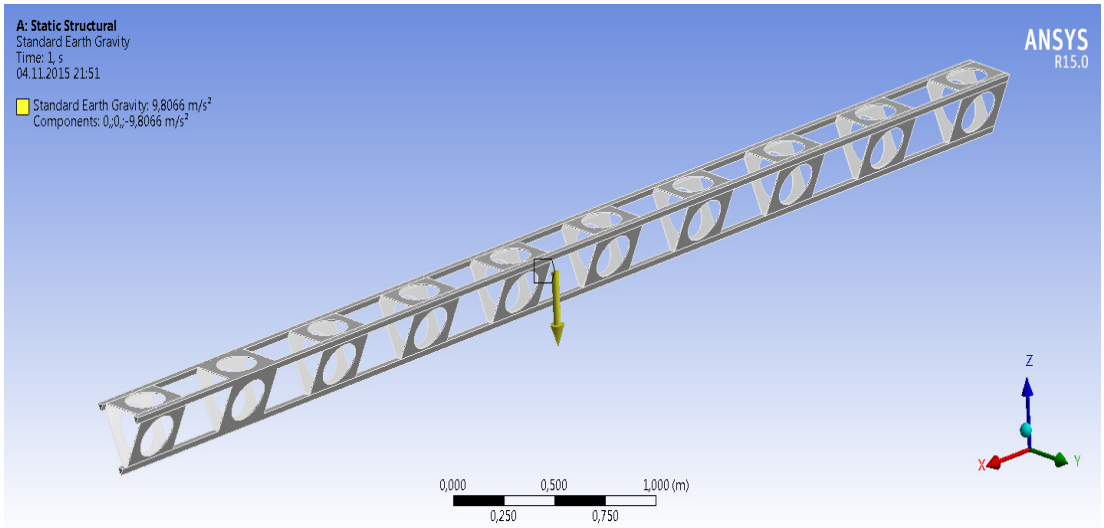
**Tablo 6.1:** Lineer aktüatör teknik özellikleri.

| | Dinamik Yük (Fx) | Dinamik Yük (Fy) | Dinamik Yük (Fz) | Eğilme Momenti (My,Mz) | Toplam Ağırlık (Servo Motor Dahil) |
|---|---|---|---|---|---|
| EMS 25 | 1470 N | 40 N | 40 N | 15 Nm | ~ 1.1 kg |

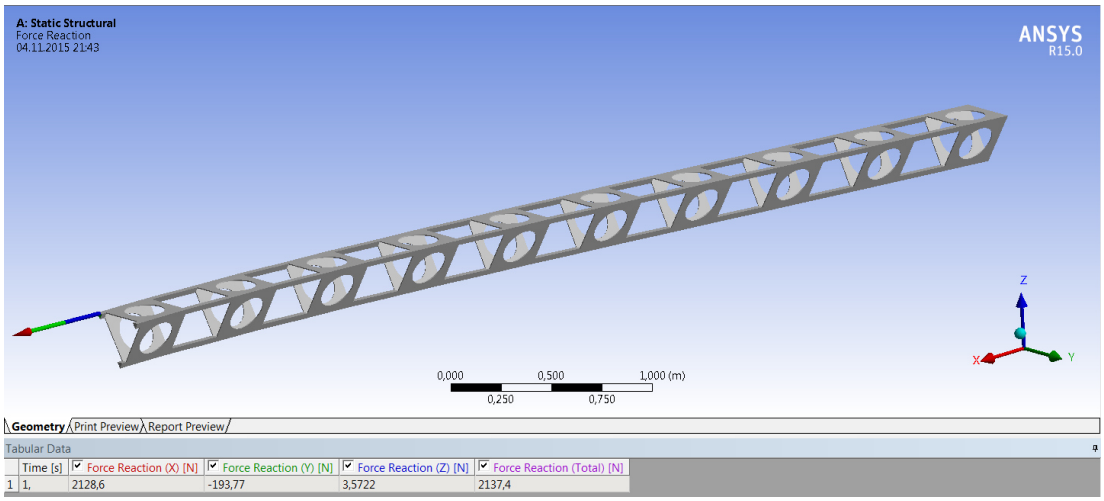
Ansys'te elde edilen, malzeme seçimi ve toplam ağırlığın gösterilmesi ile yılansı robotun kendi ağırlığı altındaki durumu, Şekil 6.2 ve Şekil 6.3'te görülmektedir.



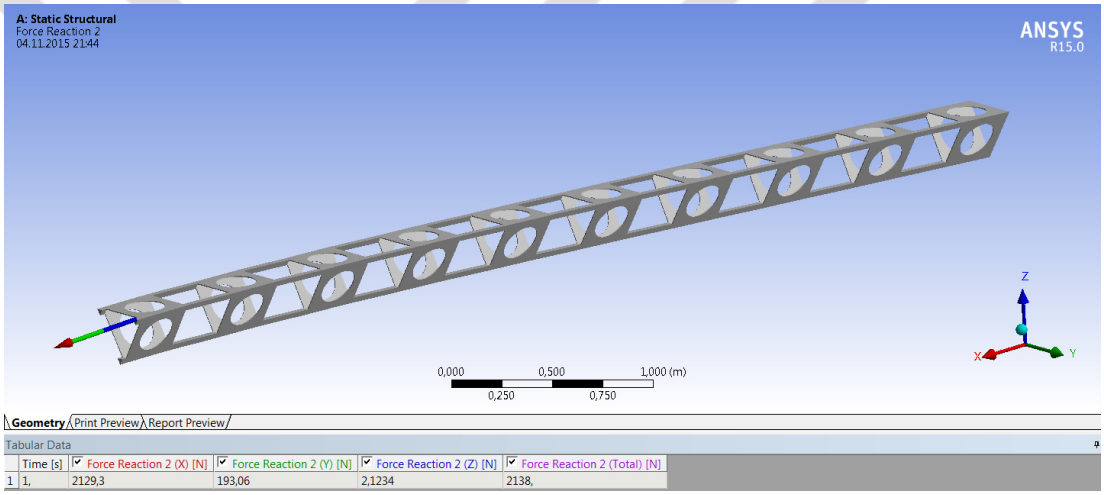

**Şekil 6.2:** Malzeme seçimi ve toplam ağırlığın gösterilmesi.



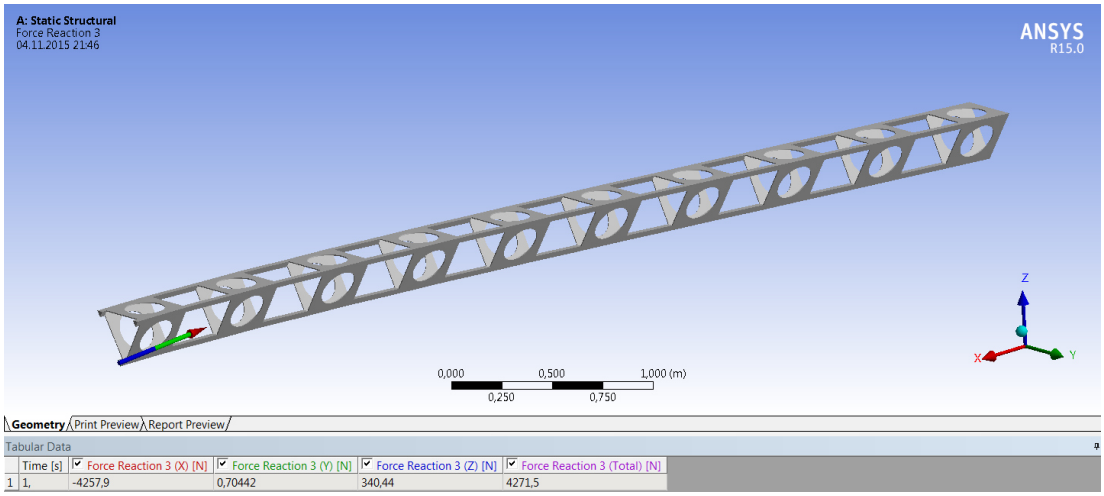

**Şekil 6.3:** Yılansı robotun kendi ağırlığı altındaki durumu.



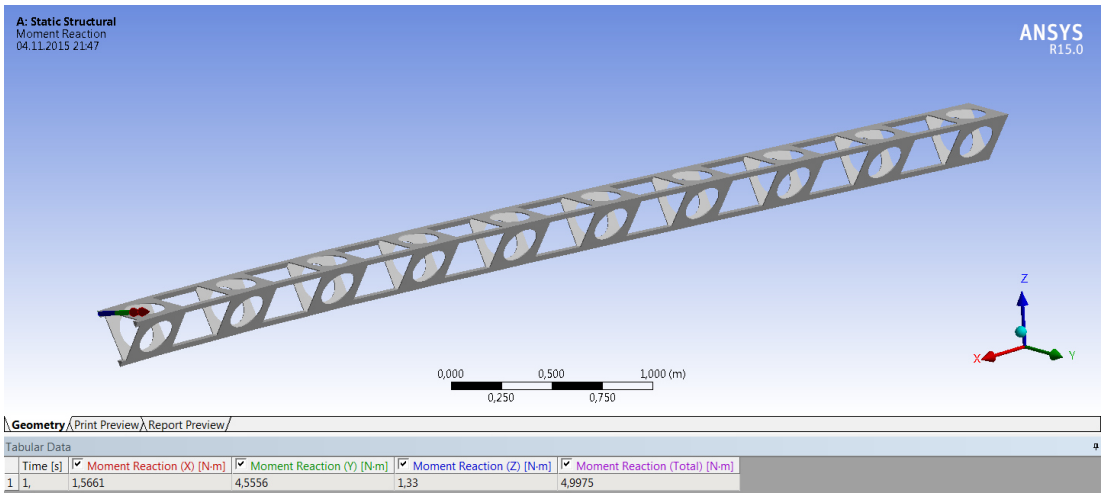

**Şekil 6.4:** Birinci reaksiyon kuvvetinin bileşenleri.



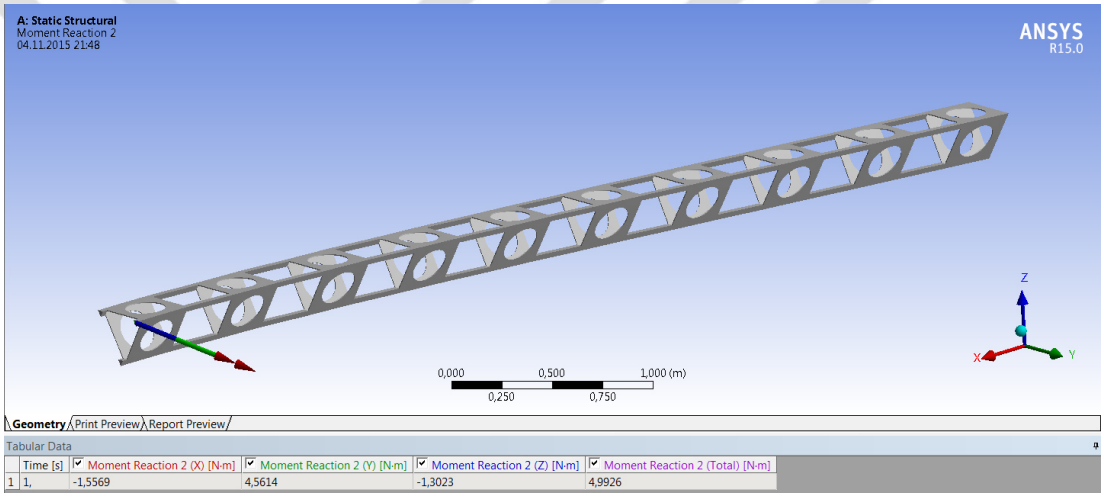

**Şekil 6.5:** İkinci reaksiyon kuvvetinin bileşenleri.



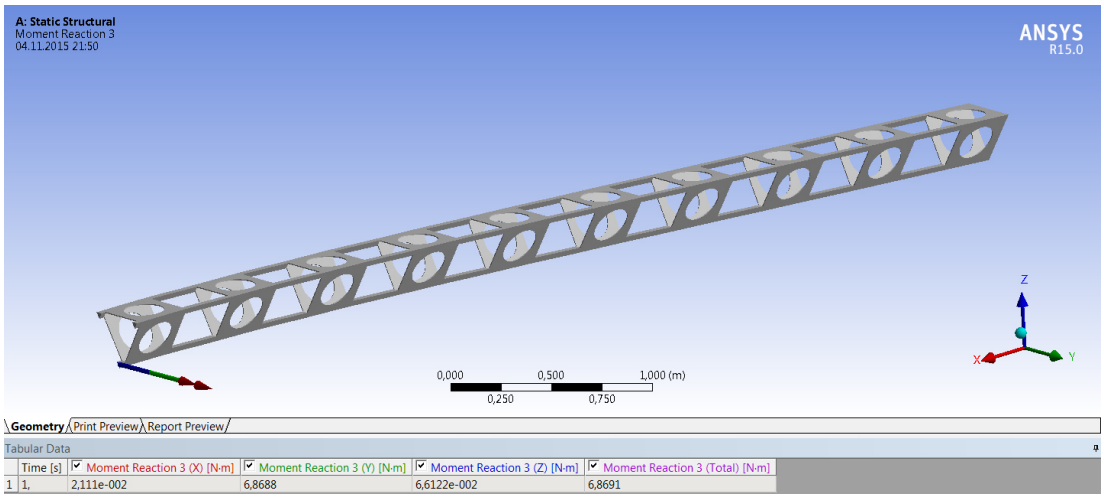

**Şekil 6.6:** Üçüncü reaksiyon kuvvetinin bileşenleri.




**Şekil 6.7:** Birinci reaksiyon momentinin bileşenleri.




**Şekil 6.8:** İkinci reaksiyon momentinin bileşenleri.




**Şekil 6.9:** Üçüncü reaksiyon momentinin bileşenleri.

Ansys'te elde edilen kuvvet reaksiyonları statik sonuçlardır. Yılansı robotun hareketinden dolayı dinamik reaksiyon kuvvetleri daha fazla olacaktır. EMS 25'teki Fx, Fy ve Fz doğrultularındaki dinamik yükler ihtiyacı karşılayamamaktadır. Statik sonuçların bile bazı bileşenleri, EMS 25'in dayanabileceği maksimum değerlerden çok fazladır.

Ansys'te elde edilen moment reaksiyonları da statik sonuçlardır. Yılansı robotun hareketinden dolayı dinamik moment reaksiyonları daha fazla olacaktır. EMS 25'teki dinamik eğilme momentleri ihtiyacı karşılayabilir. Çünkü robot yüksek hızlarda kullanılmayacaktır.

Sonuç olarak; Fx, Fy ve Fz doğrultularında ihtiyaç karşılanamadığı için bu lineer aktüatör için on modüllü yılansı robot uygulanabilir değildir. Lineer aktüatör imal eden başka şirketlerdeki lineer aktüatörlerin yılansı robota boyut olarak uygun olanlarının analizi aynı mantıkla yapılmış; fakat sonuçta bu aktüatörlerin de uygulanabilir olmadığı görülmüştür.

# 7. YÖNTEM

Denklem sistemlerini sayısal olarak çözebilmek için, Newton'un Metodu kullanılabilir. 3-RPS manipülatörün ileri kinematiğinin çözülebilmesi için denklem sistemi elde edilmiştir. Bu denklem sistemi Newton Metodu kullanılarak sayısal olarak çözülmüştür. Bu sayede C Sharp XNA ortamında on modüllü yılansı robotun üç boyutlu simülasyon programı yapılmıştır.

Tek bir denklemin çözümünü bulmak için "Aralıklı Newton Metodu" kullanılabilir (Kananen 2012). 3-RPS manipülatörün ters kinematiğinin çözülebilmesi için denklem sistemi oluşturulmuştur. Aralıklı Newton Metodu, denklem sisteminin çözümüne uygulanarak sonuca ulaşılmıştır.

Son olarak, yılansı robot için seçilen lineer aktüatörün, robotun kendi ağırlığı altında dayanıp dayanamayacağının analiz edilebilmesi için Catia'da modellenen yılansı robot Ansys'e aktarılarak ilk modülün lineer aktüatörlerinin en uç noktalarındaki statik reaksiyon kuvvet ve momentleri simüle edilmiştir.

Tüm bilgisayar analizleri, Intel Core I7, 2.3 GHz İşlemci kullanılarak yapılmıştır.

# 8. SONUÇ VE ÖNERİLER

3-RPS manipülatörün ileri kinematiğinin çözülebilmesi için oluşturulan denklem sisteminin Newton Metodu ile sayısal olarak çözülmesi sonucunda, eğer ilk tahmin değerleri çözüme yakın seçilirse, çok hızlı ve hassas bir şekilde sonuca yakınsandığı görülmüştür.

3-RPS manipülatörün ters kinematiğinin çözülebilmesi için oluşturulan denklem sisteminin Aralıklı Newton Yöntemi ile sayısal olarak çözülmesi sonucunda, eğer ilk tahmin aralıkları çözüme yakın seçilirse, çok hızlı ve hassas bir şekilde sonuca yakınsandığı görülmüştür. İlk tahmin aralıkları çözüme yakın seçilmediği taktirde daha önceki kadar hassas sonuçlar elde edilememiştir.

Ters kinematiğin çözümü sonucunda elde edilen lineer aktüatör uzunlukları belirlenen limitler (minimum 150 mm, maksimum 300 mm) dışında çıkarsa, bu değerlere göre daha mantıklı yeni tahmin aralıkları belirlenerek ters kinematik yeniden çözülebilir. Lineer aktüatör uzunlukları, belirlenen limitler arasına girene kadar bu yöntem devam ettirilebilir.

Son olarak, yılansı robot için seçilen lineer aktüatörün, robotun kendi ağırlığı altında dayanıp dayanamayacağının analiz edilebilmesi için Catia'da modellenen yılansı robot Ansys'e aktarılarak ilk modülün lineer aktüatörlerinin en uç noktalarındaki statik reaksiyon kuvvet ve momentleri simüle edilmiştir. Elde edilen sonuçlara göre, piyasada bulunan standart lineer aktüatörler kullanıldığında on modüllü yılansı robotun mekanik olarak uygulanabilir olmadığı görülmüştür. Fakat özel lineer aktüatörler tasarlanırsa robotun mekanik olarak uygulanmaması için bir sebep yoktur.

# 9. KAYNAKLAR


Almurib, H. A., Al-Qrimli, H. F. and Kumar, N., "A review of application industrial robotic design", *In ICT and Knowledge Engineering*, 105-112, (2012).

Alvarado, J. G., "Kinematics of a hybrid manipulator by means of screw theory", *Multibody System Dynamics*, 14(3-4), 345-366, (2005).

Appleton, E. and Williams, D. J., *Industrial robot applications*, U.S.A, Canada, Latin America: John Wiley and Sons, (1987).

Chirikjian, G. S. and Burdick, J. W., "A hyper-redundant manipulator", *IEEE Robotics and Automation Magazine*, 1(4), 22-29, (1994).

Çonkur, E. S., "Real time path planning and obstacle avoidance algorithms for redundant manipulators", Doktora Tezi, *University of Bristol*, Bristol, (1997).

Daniali, H. R. M., "Contributions to the kinematic synthesis of parallel manipulators", Doktora Tezi, *McGill University*, Montreal, (1995).

Fjerdingen, S., Liljebäck, P. and Transeth, A., "A snake-like robot for internal inspection of complex pipe structures (PIKo)", *In Intelligent Robots and Systems IROS 2009*, 5665-5671, (2009).

Gallardo-Alvarado, J., Aguilar-Nájera, C. R., Casique-Rosas, L., Pérez-González, L. and Rico-Martínez, J. M., "Solving the kinematics and dynamics of a modular spatial hyper-redundant manipulator by means of screw theory", *Multibody System Dynamics*, 20(4), 307-325, (2008).

Gallardo, J., Lesso, R., Rico, J. M. and Alici, G., "The kinematics of modular spatial hyper-redundant manipulators formed from RPS-type limbs", *Robotics and Autonomous Systems*, 59(1), 12-21, (2011).

Gallardo, J., Orozco, H., Rico, J. M. and González-Galván, E. J., "A new spatial hyper-redundant manipulator", *Robotics and Computer-Integrated Manufacturing*, 25(4), 703-708, (2009).

Kananen, J., "Interval Newton Method [online]", (12October2015), http://www.slideshare.net/JohannesKananen/interval-newton-method, (2012).

Kim, J., Park, F. C., Ryu, S. J., Kim, J., Hwang, J. C., Park, C. and Iurascu, C. C., "Design and analysis of a redundantly actuated parallel mechanism for rapid machining", *Robotics and Automation*, 17(4), 423-434, (2001).

Kim, S. H., Shin, Y. J., Kim, K. S. and Kim, S., "Design and control of robot manipulator with a distributed actuation mechanism", *Mechatronics*, 24(8), 1223-1230, (2014).

Lenarcic, J., Bajd, T. and Stanišić, M. M., *Robot mechanisms*, 60, Dordrecht: Springer Science and Business Media, (2013).

Li, Y. and Xu, Q., "Design and development of a medical parallel robot for cardiopulmonary resuscitation", *Mechatronics*, 12(3), 265-273, (2007).

Liljebäck, P., Pettersen, K. Y., Stavdahl, Ø. and Gravdahl, J. T., "A review on modelling, implementation, and control of snake robots", *Robotics and Autonomous Systems*, 60(1), 29-40, (2012).

Liljeback, P., Stavdahl, O. And Beitnes, A., "SnakeFighter-development of a water hydraulic fire fighting snake robot", *In Control, Automation, Robotics and Vision ICARCV'06*, 1-6, (2006).

Liu, X., Wang, J., Li, T. and Duan, G., "Parallel mechanisms with two or three degrees of freedom", *Tsinghua Science and Technology*, 8(1), 105-112, (2003).

Lu, Y. and Leinonen, T., "Solution and simulation of position-orientation for multi-spatial 3-RPS parallel mechanisms in series connection", *Multibody System Dynamics*, 14(1), 47-60, (2005).

Matsuno, F. and Suenaga, K., "Control of redundant 3D snake robot based on kinematic model", *In Robotics and Automation Proceedings ICRA'03*, 2, 2061-2066, (2003).

Merlet, J. P., *Parallel robots*, 128, Dordrecht: Springer Science and Business Media, (2006).

Mintenbeck, J. and Estana, R., "Design, modelling and control of a hyper-redundant 3-RPS parallel mechanism", *In Robotics and Biomimetics (ROBIO) 2010*, Tianjin, 591-596, (2010).

Motahari, A., Zohoor, H. and Korayem, M. H., "A new obstacle avoidance method for discretely actuated hyper-redundant manipulators", *Scientia Iranica*,19(4), 1081-1091, (2012).

Namiki, A., Imai, Y., Ishikawa, M. and Kaneko, M., "Development of a high-speed multifingered hand system and its application to catching", *In Intelligent Robots and Systems IROS 2003 Proceedings*, 3, Las Vegas, 2666-2671, (2003).

Parasuraman, S. and Liang, P. C. J., "Development of RPS parallel manipulators", *In Computer and Network Technology ICCNT 2010*, 600-605, (2010).

Rad, C. R., Stan, S. D., Balan, R. and Lapusan, C., "Forward kinematics and workspace analysis of a 3-RPS medical parallel robot", *In 2010 IEEE International Conference on Automation, Quality and Testing, Robotics*, 1-6, (2010).

Rao, P. S. and Rao, N. M., "Position analysis of spatial 3-RPS parallel manipulator", *Int. J. Mech. Eng. & Rob. Res.*, 2(2), 80-90, (2013).

Shyam, R. B. A. and Ghosal, A., "A Parallel Mechanism for Tracking the Sun", *Proceedings of 2014 IFToMM Asian Conference on Mechanism and Machine Science*, Tianjin, (2014).

Stone, H. W., *Kinematic modeling, identification, and control of robotic manipulators*, 29, Norwell, Lancaster, Dordrecht: Kluwer Academic Publishers, (1987).

Transeth, A. A., Pettersen, K. Y. and Liljebäck, P., "A survey on snake robot modeling and locomotion", *Robotica*, 27(07), 999-1015, (2009).

Tsai, L. W., *Robot analysis: the mechanics of serial and parallel manipulators*, Canada: John Wiley & Sons, (1999).

Yamakita, M., Hashimoto, M. and Yamada, T., "Control of locomotion and head configuration of 3D snake robot (SMA)", *In Robotics and Automation Proceedings ICRA'03*, 2, 2055-2060, (2003).

# 10. ÖZGEÇMİŞ


| | |
|---|---|
| Adı Soyadı | : Yalçın BULUT |
| Doğum Yeri ve Tarihi | : Adana / 05.01.1987 |
| Lisans Üniversite | : Çukurova Üniversitesi |
| Elektronik posta | : ybulut@pau.edu.tr |
| İletişim Adresi | : Pamukkale Üniversitesi Makine Mühendisliği Bölümü |