İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İKİ AYAKLI YÜRÜYEN ROBOT İÇİN KONTROL SİSTEMİ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Numan Mert TAN

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

KASIM 2012

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İKİ AYAKLI YÜRÜYEN ROBOT İÇİN KONTROL SİSTEMİ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Numan Mert TAN
(518101031)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Yrd.Doç.Dr. Zeki Yağız BAYRAKTAROĞLU

KASIM 2012

İTÜ, Fen Bilimleri Enstitüsü'nün 518101031 numaralı Yüksek Lisans Öğrencisi **Numan Mert TAN** , ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"İKİ AYAKLI YÜRÜYEN ROBOT İÇİN KONTROL SİSTEMİ GELİŞTİRİLMESİ"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Yrd.Doç.Dr. Zeki Yağız BAYRAKTAROĞLU**
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Prof.Dr. Şeniz ERTUĞRUL**
İstanbul Teknik Üniversitesi

Prof.Dr. Metin GÖKAŞAN
İstanbul Teknik Üniversitesi

Teslim Tarihi : **1 Kasım 2012**
Savunma Tarihi : **30 Kasım 2012**

Aileme,

ÖNSÖZ

İki Ayaklı Yürüme Hareketi Modellemesi Kontrolü ve Prototip İmalatı başlıklı TÜBİTAK projesi'nin bir devamı niteliğinde olan bu çalışma, iki ayaklı robotun yörünge planlaması, mevcut iletişim protokolünün geliştirilmesi ve kuvvet/moment sensörleri yardımıyla gerçekleştirilen deneysel çalışmaları içermektedir.

Çalışmalarım sırasında her zaman yardımlarını sunan ve yol gösteren danışmanım sayın Yrd.Doç.Dr.Zeki Yağız Bayraktaroğlu'na teşekkür ederim. Ayrıca sürekli olarak yardımlarına başvurduğum Ar.Gör.Mesut Acar'a, Ar.Gör.Gökçe Burak Tağlıoğlu'na, Ar.Gör.Cihat Bora Yiğit'e, Ar.Gör.Ömer Faruk Argın'a ve beni her zaman destekleyen ve motivasyonumu üst düzeyde tutmamı sağlayan aileme de teşekkürlerimi sunarım.

Kasım 2012

Numan Mert TAN
Makina Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

ZMP : Zero Moment Point
PID : Proportional Integral Derivative
CAN : Controller Area Network
SDO : Service Data Object
PDO : Process Data Object
RxPDO : Receive PDO
TxPDO : Transmit PDO
RTR : Remote Transmission Request
TDF : Tek Destek Fazı
ÇDF : Çift Destek Fazı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

İKİ AYAKLI YÜRÜYEN ROBOT İÇİN KONTROL SİSTEMİ GELİŞTİRİLMESİ

ÖZET

İki Ayaklı Yürüme Hareketi Modellemesi, Kontrolü ve Prototip İmalatı başlıklı TÜBİTAK projesinde, bu çalışmanın öncülü olarak, robotun bilgisayar ortamında benzetimi yapıldıktan sonra, katı modeli oluşturulmuş, mekanik yapı tasarlanıp uzuvlar ve aktarma organları gibi elemanlar seçilmiş ve elektronik donanımın entegrasyonu ve programlanmasıyla prototip çalışması hayata geçirilmiştir.

Bu çalışma kapsamında robotun kontrol sisteminde geliştirmeler ve deneyler yapılmıştır. Bunlardan birincisi, iletişim protokolü CANopen üzerinde yapılan değişikliklerdir. Prototip geliştirme çalışmaları sırasında, CANopen protokolünün bir aracı olan SDO iletişimi kullanılmıştır. Deneysel çalışmalarda, uygulanan bu iletişim yöntemiyle, robotun referans hareket hızları ile uyumlu örnekleme periyodları elde edilememiştir. Dolayısıyla SDO iletişimi ile sağlanan kapalı çevrim kontrol sistemi, istenen hızlar için gerçek zamanlı hareket kontrolüne olanak vermemektedir. Bu nedenle, SDO yerine daha hızlı ve daha esnek veri alışverişine olanak sağlayan PDO iletişimine geçiş yapılmıştır.

Yapılan ikinci çalışma, eklem yörüngelerinin üretilmesiyle ilgilidir. Daha önce yapılan çalışmalarda, robotun hareketlerinin referans yörüngelerini oluşturmak amacıyla bir MATLAB programı yazılmış ve yörüngeler bir dosyaya kaydedilerek gömülü bilgisayara iletilmiştir. Bu tez kapsamında eklem referans yörüngeleri, C dilinde yazılan ters kinematik algoritmasıyla gömülü bilgisayar üzerinden doğrudan üretilebilir hale getirilmiştir.

Bir diğer çalışma, robota daha önce entegre edilmiş ancak kullanılmamış kuvvet/moment sensörlerinin devreye alınmasıdır. 3 eksende kuvvet ve moment ölçümü yapabilen kuvvet/moment sensörlerinin arayüzleri ve gömülü bilgisayar arasındaki iletişimi sağlamak için de CAN bus protokolü kullanılmıştır. Sensörlerin kontrol sistemine entegrasyonu için C dilinde yazılan kodlar Ekler bölümünde sunulmuştur. Kuvvet/moment sensörlerinin kullanılması özellikle kapalı çevrim dinamik yürüme kontrolü için sıfır moment noktası (ZMP)'nın hesaplanması ve kontrolü açısından oldukça önemlidir.

Yapılan deneylerde, eklem momentleri iki farklı yöntemle elde edilmiştir. Bileklerdeki sensörlerden alınan ayak/yer temas kuvvet ve momentleri, robotun Jakobiyen matrisi aracılığıyla eklem momentlerine çevrilmiştir. Eklem momentleri ayrıca ölçülen motor akımları üzerinden de hesaplanmıştır. Her iki yöntemle elde edilen eklem momentleri karşılaştırmalı olarak verilmiştir.

CONTROL SYSTEM DEVELOPMENT FOR BIPEDAL WALKING ROBOT

SUMMARY

One of the most important research topics in the field of humanoid and bipedal robotics is the study of human locomotion system and designing mechanisms that are able to fulfill human-like walking by means of generating and controlling task oriented walking trajectories.

As a precursor of this study, TUBITAK project titled 'Modeling and Control of Biped Walking and Prototype Manufacturing', computer simulations of 12 dof biped robot was carried out and a solid model was constructed using SolidWorks program. Initial simulation studies were followed by the design of mechanical structure, design and selection of robot links and transmission systems and a prototype was completed with the integration and programming of electronic hardware.

In the scope of this study, certain improvements were realized in the control system of the robot and experiments were carried out. One of these works are the changes applied on CAN bus communication protocol. Service Data Objects(SDO) is one of the protocols that can be used under CANopen communication and it allows the user to configure data inside CANopen object dictionary. During experimental studies such as making the robot follow discretized horizontal and vertical trajectories, it was observed that with the aforementioned communication method, sampling periods in compliance with the robot's reference joint velocities could not be obtained. In this case, closed loop control system established with SDO protocol cannot afford real time walking control. Therefore, SDO protocol was replaced with PDO protocol that allows for faster and more flexible data exchange.

PDO(Process Data Object) is a high priority CANopen protocol within data bus that is used for fast data transmission for real time applications. PDOs use producer/consumer model and are unconfirmed messages with high priority and no protocol overhead and thanks to this, process data can be transmitted from one node to any number of nodes swiftly.

While SDO enables data exchange between only one node and server simultaneously during communication, if configured, PDO allows one driver to initiate data transfer between limitless number of drivers. There are four Write(Receive)-PDOs and four Read(Transmit)-PDOs configurable inside drivers that are abbreviated as RxPDO and TxPDO respectively.

CANopen has 3 distinct message triggering modes. These are event or timer driven, remote transmission request and synchronous transmission(cyclic,acyclic). CANopen offers 2 different PDO transmission modes namely synchronous and asynchronous modes. Synchronous PDOs are transmitted within synchronization window after the synchronization object. On the other hand, asynchronous PDOs can be transmitted anytime during communication according to their priority as well as within synchronous window. Priority of synchronized PDOs are higher than those of asynchronized PDOs. In current experimental system configuration, both

synchronized and asynchronized PDOs are available. Asynchronous PDO transmission mode possesses Remote Transmissino Request (RTR) defined in the motor drivers

In 'Motion Planning' chapter, Jacobian matrix which is required to calculate torque values to be used with measurements obtained with force/torque sensors integrated on ankle joints, was found using the forward kinematics equations.

Jacobian matrix can be obtained with the geometric model available. Using the geometric model, with the assumption of a 6 degrees of freedom manipulator arm, Jacobian can be calculated by taking partial derivatives of geometric model functions. It can also be derived decomposing the coefficients of joint velocities in terms of robot's end effector velocity in Cartesian plane.

Within the context of this thesis, Jacobian was found with the method of recursive linear and angular velocity formulations. Jacobian's first three rows and six columns represent transition caused by linear velocities, and last three rows and six columns represent rotation caused by angular velocities.

The second major work was regarding the generation of joint trajectories. In the previous studies, reference joint trajectories for any movement were generated with a MATLAB program and resulting trajectory arrays were saved on a file and then passed to embedded computer while in this study, joint reference trajectories were generated with an inverse kinematics algorithm written in C language and applied to robot directly from the embedded computer.

Inverse kinematics solution of the bipedal robot is the calculation of joint angles during motion, with the available knowledge of torso and feet distances.

In trajectory planning section, generation of desired trajectories with the help of inverse kinematics and point to point polynomial fitting, was summarized and application to robot was presented schematically.

After determining of desired torso and feet distances, joint trajectories can be generated by entering motion parameters such as step height, step length and step period into the functions of inverse kinematics program written in C language. Trajectories are created in transversal, longitudinal and vertical axes with regards to third and fifth order polynomials. In addition to sample joint trajectories for stepping of left foot, revelant velocity and acceleration profiles were also included in the related section. Longitudinal motion takes place in X axis while transversal motion is carried out in Y axis and vertical motion in Z axis according to pre-determined reference frames.

The order of robot's motion is as follows: Firstly, transfer of torso takes place transversally in Y axis, in the opposite direction of the stepping foot, followed by the first step. Then, torso moves forward to the foot in the front with the transfer of the foot in back latest.

In order to maintain stability during motion, projection of robot's center of mass has to fall inside the support polygon formed between the feet. Thus, displacement of torso takes place in double support phase where both feet contact with the ground whereas displacement of the feet takes place in single support phase where only one of the feet is in contact with the ground. Therefore, torso moves in opposite direction of the first stepping foot, to secure the location of projection of center of mass inside the support polygon formed by only one foot.

Moreover, force/torque sensors previously mounted on the ankle joints were put into service. Force/torque sensors can measure forces and torques in 3 axis and communication between sensor interface and embedded computer was established by also using CAN bus protocol. Programming the protocol to utilize sensor measurement was explained in the related part of the thesis and codes written in C language for the integration of the sensors into the control system were presented in the attachment. Putting force/torque sensors into service is important in the way that they are key components on a biped walking mechanism while realizing closed loop dynamic walking control to locate and control the zero moment point.

In the conducted experiments, two distinct methods of determining the joint torques were presented. One of the methods is the usage of motor currents and transmission system equations. Second method is the utilization of foot/ground reaction forces and torques obtained from the sensors integrated on ankle joints and Jacobian matrix. Comparison of joint torques obtained with both methods were given graphically.

Torque behaviours of hip joints follow each other with small differences due to simpler mechanica structure of mentioned joints.

In the knee joints, torque curves are less similar than hip joint torque curves. Main reason for this is that the mechanical transmission for knees are more complex than hip mechanical structure and transmission kinematics rely on nonlinear equations.

Ankle transmission structure comprises universal joints, ball screw mechanism and spherical joints and is more complex than both the hip and knee joint structures. As a result, in order to express the calculations of ankle joint torques by the method of actuator currents, more complicated and nonlinear expressions were built. Also, the elastic behavior of spherical joints is not compatible with the assumption of rigid joint mechanism which plays an important role for torque behavior difference of aforementioned methods. Other neglected factors are friction and mechanical imperfectness.

1. GİRİŞ

İnsansı robotların üretimindeki temel amaç, insan kadar esnek hareket edebilen sistemler meydana getirip, üretim hattında insanlarla birlikte çalışabilecek, zorlu çalışma şartlarında insanın yerini alabilecek, yaşlı veya engelli insanların bakımı gibi sorumlulukları üstlenecek mekanizmalar meydana getirip insan hayatını kolaylaştırmaktır. İnsansı robotlar hakkında son yarım yüzyılda meydana gelen hem teorik hem de uygulamalı gelişmeler, bu tür robotların özellikle rutin işleri insanların yerine yapabilmelerine olanak verecek şekilde hızla ilerlemektedir. Henüz bu amaca ulaşılamadıysa da yapılan çalışmalar yakın gelecek için umut vericidir. Bu tür robotlarda en önemli araştırma konularından biri insan yürüme hareketinin incelenmesi ve bu hareketi yapabilecek mekanizmaların tasarlanıp, insansı yürüme hareketinin gerçekleştirilebilmesi için yürüme planlarının çıkarılıp kontrol edilmesidir. Tez kapsamında daha önce bilgisayar benzetimleri ve imalatı yapılmış olan robotun kinematik kullanılarak hareketinin planlanması, iletişim protokolüyle ilgili çalışmalar ve tez kapsamında devreye alınan kuvvet/moment sensörleri yardımıyla çeşitli deneysel çalışmalar yapılmış ve sonuçları sunulmuştur.

1.1 Literatür Araştırması

Literatürde bulunan iki ayaklı robotik konusundaki en önemli teorik çalışmalardan biri Vukobratovic [1] tarafından 1968 yılında önerilen sıfır moment noktası(ZMP) yöntemidir. Bu yöntem, iki ayaklı robotun yürürken dengede kalması için sıfır moment noktası denilen ve dikey atalet kuvvetleri ve ağırlık kuvvetleri toplamının sıfır olduğu noktanın bulunması problemidir. Teorik temellerin sağlamlaştırılmasından sonra robotik literatürüne önemli katkılar yapan çeşitli prototipler geliştirilmiştir. Bunlardan belki de en önemlisi Honda'nın 2000 yılında tanıttığı ve dinamik yürüme için ZMP kontrolü kullanan ASIMO (Şekil 1.1) adlı robottur. 2011 yılında piyasa çıkarılan modelinde, ASIMO 130cm uzunluğunda, 48 kg ağırlığındadır ve toplam 57 serbestlik derecesine sahiptir. 40 dakika-1 saat arası yürüyebilir ve 2.7 km/saat yürüme hızına ve 9 km/saat koşma hızına ulaşabilir.

Şekil 1.1 : Honda ASIMO [2].

Avrupa'da yapılan iki ayaklı robot çalışmalarından biri ise Münih Teknik Üniversitesi'nde yapılan JOHNNIE(Şekil 1.2) adlı robottur. Toplam 17 serbestlik derecesine sahip JOHNNIE her bir bacağında 6, omuzlarında 2 ve üst gövdesinde 1 serbestlik derecesine sahiptir. 1.80m boyunda ve 40 kg ağırlığındaki robot, sensör bazında artımlı kodlayıcılar ve optik algıyıcılar kullanmakta, iletişim protokolü olarak CAN bus ve işletim sistemi olarak gerçek zamanlı Linux kullanmaktadır.

Şekil 1.2 : JOHNNIE [3].

Literatürdeki bir diğer önemli çalışma ise, Japonya'da Kawada Industries firmasının AIST (National Institute of Advanced Science and Technology) ile birlikte geliştirdiği HRP-4 adlı prototiptir (Şekil 1.3). HRP-4, 151 cm boyunda ve 39 kg ağırlığında, bacaklarda 6, boyunda 2, göğüste 2, kollarda 7 ve ellerde 2 olmak üzere toplam 34 serbestlik derecesine sahiptir. HRP-4 tek eliyle maksimum 0.5 kg kaldırabilir.

Şekil 1.3 : HRP-4 [4].

Kontrol mimarisi olarak gerçek zamanlı Linux işletim sistemine sahip 1.6 GHz işlemcili Pentium bilgisayar kullanılmış ve iletişim olarak kablosuz LAN bağlantısı seçilmiştir. Ayrıca görüntü işleme için eller ve kafada kameralar, ses işleme içinse mikrofon bulundurmaktadır.

Dikkat çeken çalışmalardan bir diğeri Kore'de KAIST (Korea Advanced Institute of Science and Technology) tarafından geliştirilip, imal edilen HUBO isimli robottur (Şekil 1.4). Çizelge 1.1'de 2005'ten itibaren piyasaya sürülen HUBO serisi robotların özellikleri verilmiştir. HUBO serisi algılayıcı bazında 3 eksenli kuvvet/moment sensörleri, eğimölçer ve jiroskop kullanmaktadır. İşletim sistemi olarak ise gerçek zamanlı hale getirilmiş Windows XP kullanmaktadır.

Şekil 1.4 : HUBO [5].

Çizelge 1.1 : HUBO serisi robotlar.

	HUBO	Albert HUBO	HUBO2
Ağırlık	56kg.	57kg.	45kg.
Uzunluk	125cm.	137cm.	125cm.
Yürüme Hızı	1.25km/h	1.25km/h	1.5km/h
Çalışma Süresi	60dk.	60dk.	60dk.
Serbestlik Der.	41	66	40

1.2 İki Ayaklı Robotun Mekanik Yapısı

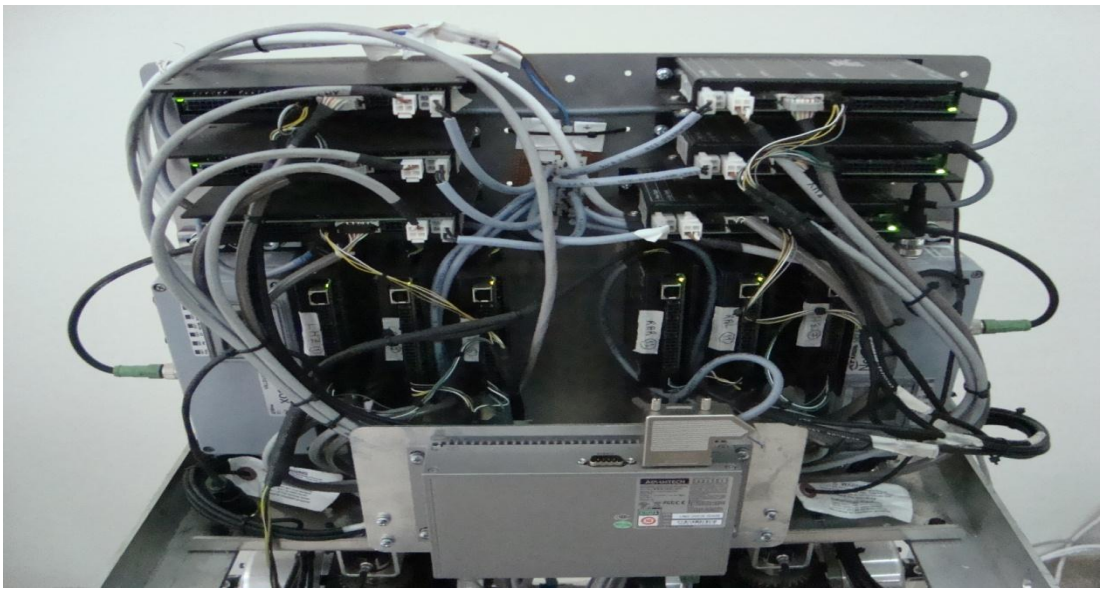
Bu bölümde robotun serbestlik dereceleri, eklemler üzerinde bulunan aktarma organları, yapıları ve bazı eklemlerde kullanılan mekanizmalardan kısaca bahsedilecektir.

142 cm boyunda ve 55 kg ağırlığındaki iki ayaklı robot her bir bacakta 6 olmak üzere toplam 12 serbestlik derecesine sahiptir. Bu serbestlik derecelerinden 3 tanesi kalçalarda 1 tanesi dizlerde ve 2 tanesi de bileklerdedir. Toplam 12 serbestlik derecesinde hareketi sağlamak için 12 adet DC motor kullanılmıştır. DC motorlar ve sürülen eklemler arasında kullanılan aktarma organları kalçalarda, farklı çevrim oranlarına sahip redüktörler, dizlerde ve bileklerde ise bilyalı vida mekanizmalarıdır. Bileklerde kullanılan vida mekanizması, bileklerin iki eksende de hareketini sağlayacak özel bir paralel mekanizmadır. Bu mekanizma sayesinde, motorların aynı yönde veya ters yönde dönmesine bağlı olarak 2 serbestlik derecesinde de hareket gerçekleştirilebilir. Çizelge 1.2'de robotta kullanılan aktarma organlarının listesi verilmiştir.

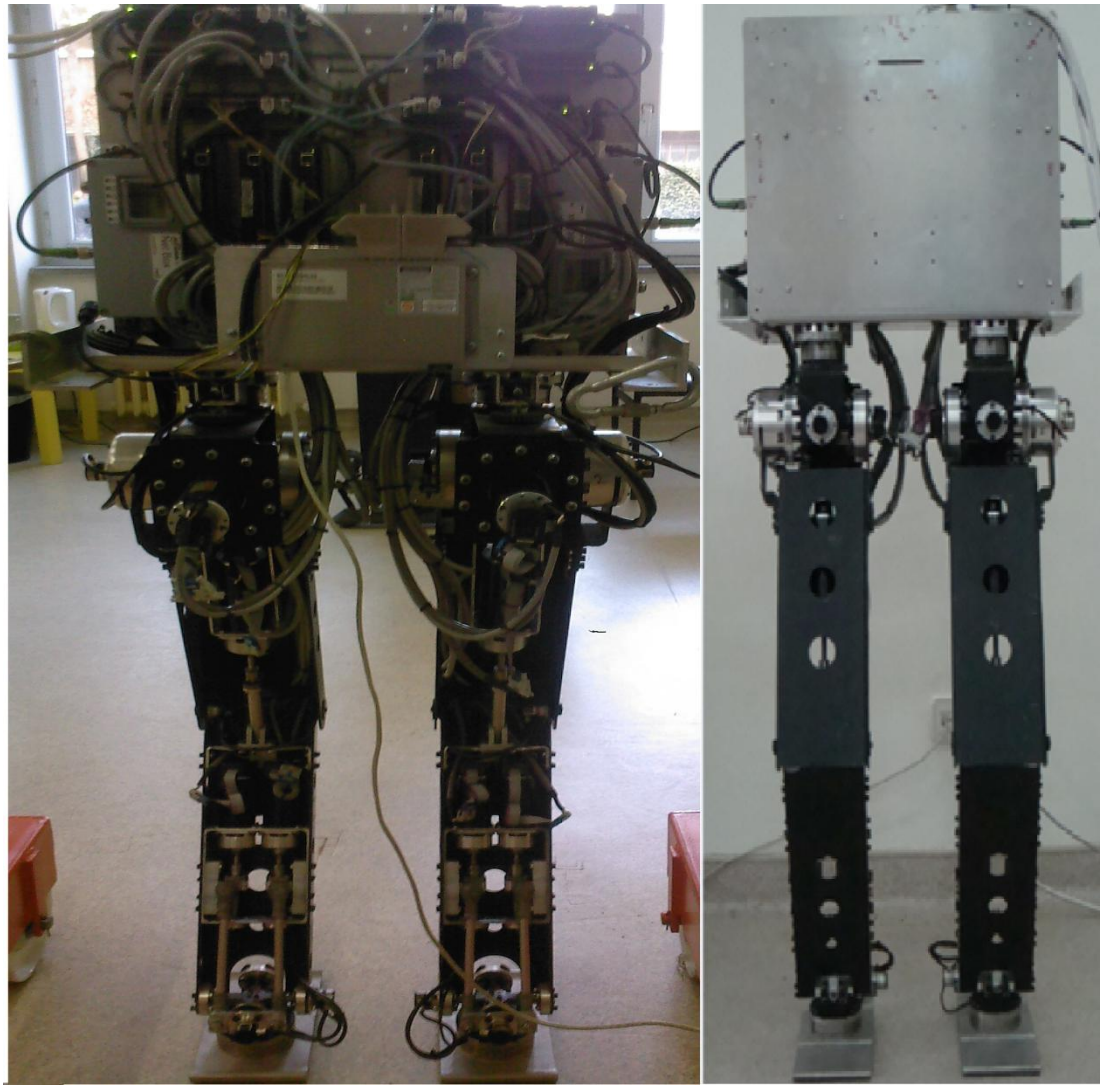
Çizelge 1.2 : Motorlar, eklemler, aktarma organları ve kısıtlar.

Motorlar	Eklemler	Aktarma organı	Kısıtlar
Sol Kalça Z	θ_1	Harmonik Sürücü (1:120)	$-20 < \theta_1 < 20$
Sol Kalça X	θ_2	Harmonik Sürücü (1:160)	$-60 < \theta_1 < 60$
Sol Kalça Y	θ_3	Harmonik Sürücü (1:160)	$-90 < \theta_1 < 90$
Sol Diz Y	θ_4	Rexroth Bilyalı Vida 1531-2	$5 < \theta_1 < 82$
Sol Bilek,Sol	θ_5	Rexroth Bilyalı Vida 1531-1	$-53 < \theta_1 < -1$
Sol Bilek,Sağ	θ_6	Rexroth Bilyalı Vida 1531-1	$-30 < \theta_1 < 30$
Sağ Kalça Z	θ_7	Harmonik Sürücü (1:120)	$-20 < \theta_1 < 20$
Sağ Kalça X	θ_8	Harmonik Sürücü (1:160)	$-60 < \theta_1 < 60$
Sağ Kalça Y	θ_9	Harmonik Sürücü (1:160)	$-90 < \theta_1 < 90$
Sağ Diz Y	θ_{10}	Rexroth Bilyalı Vida 1531-2	$3 < \theta_1 < 82$
Sağ Bilek,Sol	θ_{11}	Rexroth Bilyalı Vida 1531-1	$-52 < \theta_1 < -3$
Sağ Bilek, Sağ	θ_{12}	Rexroth Bilyalı Vida 1531-1	$-30 < \theta_1 < 30$

Diz ve bilek eklemlerindeki motorların sürülen eklemlerden daha yukarıda bulunmalarının sebebi lineer ters sarkaç modeline göre robotun ağırlık merkezinin mümkün olduğu kadar yukarıya çekilmesidir [6]. Ayrıca aynı nedenden dolayı elektronik donanımın birçoğu da gövdeye yerleştirilmiştir. Şekil 1.5'te gövde kısmının görünümü ve Şekil 1.6'da robotun önden ve arkadan görünümleri verilmiştir.

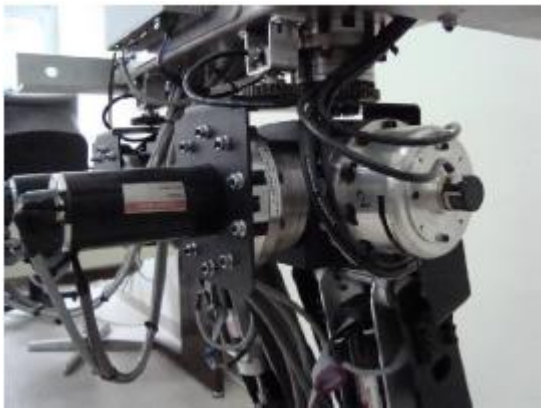


Şekil 1.5 : Robotun gövde kısmının görünümü.



Şekil 1.6 : Robotun önden ve arkadan görünümleri.

Şekil 1.7 ve Şekil 1.8 kalça ve diz eklemlerinin detay görünümlerini, Şekil 1.9 ise bilek ekleminin detay görünümünü göstermektedir.

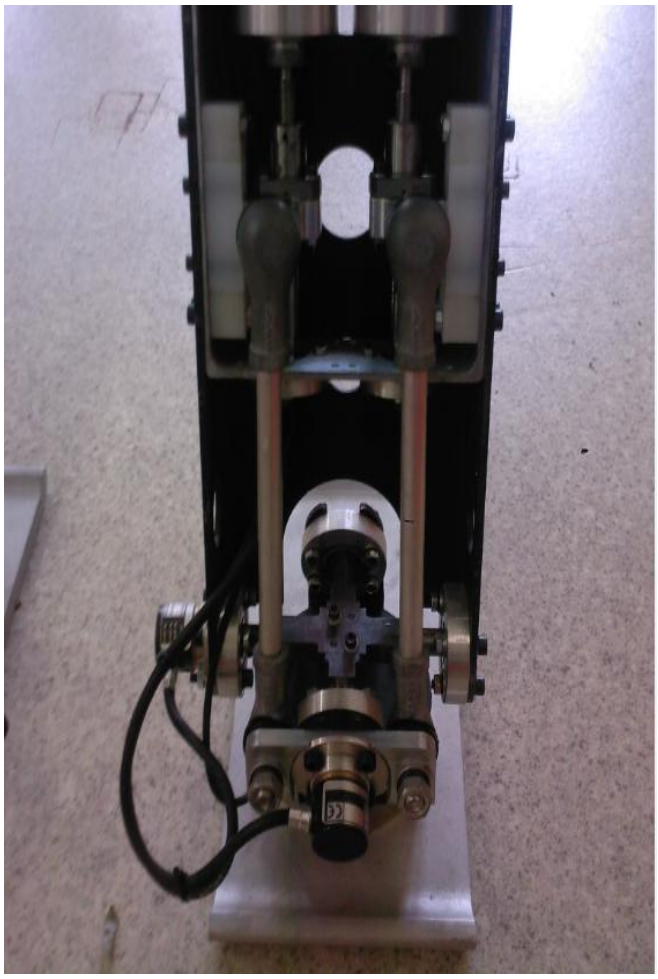


Şekil 1.7 : Kalça eklemi detay görünümü

Şekil 1.8 : Diz eklemi detay görünümü

1.3 İki Ayaklı Robotun Elektronik Donanımı

Seçilen hareket sistemine göre [7] robotun tüm eklemlerinde artırımlı enkoder (Şekil 1.10) ve mutlak enkoder (Şekil 1.10) bulunmaktadır. Motor sürücüsü ve barındırdığı PID kontrolörü, DC motor ve motor çıkışı üzerinde bulunan artırımlı enkoder sayesinde motor milinin istenilen konuma gitmesi sağlanır. Ancak artırımlı enkoder bulunduğu konumu sıfır pozisyonu kabul ederek puls saydığı için eklemler üzerinde ayrıca mutlak enkoder bulundurma zorunluluğu meydana gelmiştir. Mutlak enkoder sayesinde robot hangi konumda olursa o konum mutlak olarak ölçülebileceği için



Şekil 1.9 : Bilek eklemi detay görünümü

harekete başlamadan önce sıfır pozisyonuna getirme sorunu ortadan kalkmış olur. Artırımlı enkoder motor milinin bir turunda 2000 artımla sayar, mutlak enkoder ise 12 bit çözünürlüğe sahiptir.

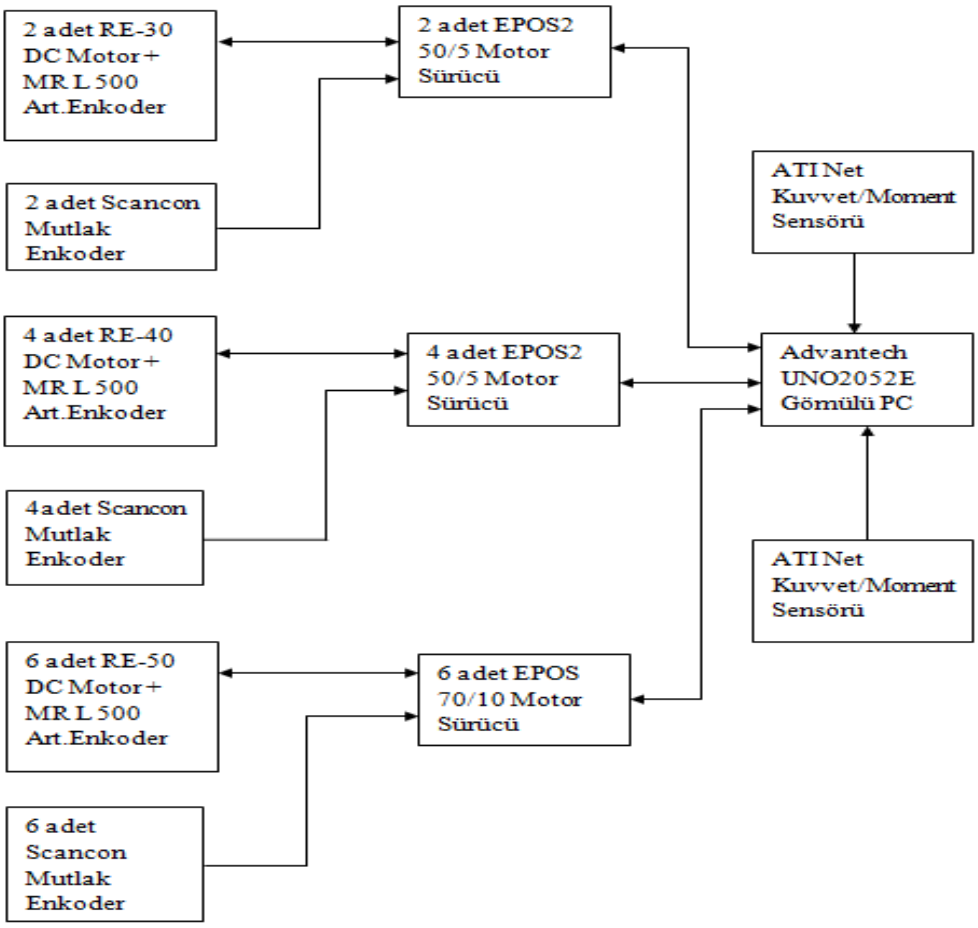
Şekil 1.10 : Artırımlı enkoder ve mutlak enkoder.

Sistemin bilgisayarla kontrolünü yapabilmek adına 12 adet Maxon DC motor için 12 adet Maxon EPOS2 sürücü (Şekil 1.11), ve 1 adet Linux işletim sistemine sahip Advantech UNO2052E (Şekil 1.11) marka gömülü bilgisayar seçilmiştir. Sistemin öngörülen örnekleme periyodunda hareket edebilmesi için sürücüler, enkoderler ve bilgisayar arasında CAN bus yoluyla iletişim metodu kullanılmıştır. Bir başka önemli donanım ise kuvvet/moment sensörleridir. Her bir bacak için bileklere monte edilen ATI Mini85 Kuvvet/Moment sensörleri (Şekil 1.12) 3 eksende kuvvet ve 3 eksende moment ölçümü yapabilir ve özellikle dinamik yürüme kontrolü sırasında ZMP metodunun uygulanabilmesi için gereklidir.

Şekil 1.11 : Gömülü bilgisayar ve motor sürücüsü.

Şekil 1.12 Kuvvet/moment sensörü ve arayüzü.

Şekil 1.13'te robotta kullanılan elektronik aksamın şematik hali ve birbirleriyle bağlantıları gösterilmiştir.



Şekil 1.13 : Elektronik donanım bağlantı şeması.

1.4 Sunuş Planı

İkinci bölümde robotun ileri kinematiği kullanılarak eklem momentlerinin nasıl elde edileceği anlatılacak, ters kinematik ile ilgili teorik bilgi verilecek ve yörünge planlaması altbölümünde ters kinematik kullanılarak eklem yörüngelerinin nasıl elde edildiği ve uygulandığı anlatılacaktır. Üçüncü bölümde daha önce kullanılmış ve şu anda kullanılmakta olan CAN bus iletişim protokollerinin karşılaştırılması ve şu anda kullanılan protokol hakkında detaylı bilgi bulunmaktadır. Bu bölümde ayrıca çevrime dahil edilen kuvvet/moment sensörlerinin kontrol sistemine nasıl entegre edildiği açıklanmıştır. Dördüncü bölümde deneysel olarak elde edilen minimum örnekleme periyodu açıklanmış ve iki farklı yöntemle eklem momentleri belirlenerek sonuçlar karşılaştırmalı olarak sunulmuştur.

2. HAREKET PLANLAMASI

2.1 İleri Kinematik Kullanılarak Jakobiyen Matrisinin Elde Edilmesi

Bu bölümde robotun ileri kinematik ifadeleri kullanılarak, deneysel çalışmalar bölümünde kuvvet/moment sensörlerinden yararlanarak elde edilecek olan eklem momentlerini bulabilmek için gerekli Jakobiyen matrisinin nasıl bulunduğu anlatılacaktır.

Jakobiyenin tanımı [8],

$$\delta x = J\delta\theta, \tag{2.1}$$

ve

$$dX = Jd\theta\text{'dır.} \tag{2.2}$$

Her iki tarafı birim zaman dt'ye bölersek,

$$dX/dt = J\, d\theta/dt \text{ ve} \tag{2.3}$$

$$\dot{X} = J\dot{\theta}\text{ 'dır.} \tag{2.4}$$

Jakobiyen matrisi geometrik modelden de yararlanılarak bulunabilir. Geometrik model,

$$X = f(\theta), \tag{2.5}$$

yazılarak, 6 serbestlik dereceli bir manipulatör olarak düşünülebilecek robot için f fonksiyonunun eklem değişkenlerine göre kısmi türevleri alınırsa, Jakobiyen matrisi (2.6)'daki gibi elde edilebilir. Jakobiyen matrisi, f fonksiyonlarının θ eklem açılarına göre kısmi türevleri alınarak türetilebileceği gibi robotun kartezyen düzlemdeki uç hızlarının $\dot{\theta}$ eklem hızlarına göre katsayılarına ayrıştırılarak ta bulunabilir. Jakobiyen matrisi tez kapsamında bahsedilen ikinci yöntem olan çizgisel ve açısal hızların elde edilmesiyle bulunmuştur.

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \theta_3} & \frac{\partial f_1}{\partial \theta_4} & \frac{\partial f_1}{\partial \theta_5} & \frac{\partial f_1}{\partial \theta_6} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \theta_3} & \frac{\partial f_2}{\partial \theta_4} & \frac{\partial f_2}{\partial \theta_5} & \frac{\partial f_2}{\partial \theta_6} \\ \frac{\partial f_3}{\partial \theta_1} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial \theta_3} & \frac{\partial f_3}{\partial \theta_4} & \frac{\partial f_3}{\partial \theta_5} & \frac{\partial f_3}{\partial \theta_6} \\ \frac{\partial f_4}{\partial \theta_1} & \frac{\partial f_4}{\partial \theta_2} & \frac{\partial f_4}{\partial \theta_3} & \frac{\partial f_4}{\partial \theta_4} & \frac{\partial f_4}{\partial \theta_5} & \frac{\partial f_4}{\partial \theta_6} \\ \frac{\partial f_5}{\partial \theta_1} & \frac{\partial f_5}{\partial \theta_2} & \frac{\partial f_5}{\partial \theta_3} & \frac{\partial f_5}{\partial \theta_4} & \frac{\partial f_5}{\partial \theta_5} & \frac{\partial f_5}{\partial \theta_6} \\ \frac{\partial f_6}{\partial \theta_1} & \frac{\partial f_6}{\partial \theta_2} & \frac{\partial f_6}{\partial \theta_3} & \frac{\partial f_6}{\partial \theta_4} & \frac{\partial f_6}{\partial \theta_5} & \frac{\partial f_6}{\partial \theta_6} \end{bmatrix} \quad \textbf{(2.6)}$$

şeklinde ifade edilebilir. Jakobiyen matrisinin ilk üç satır ve altı sütunu robotun uç konumunun çizgisel hızlardan kaynaklanan ötelemesini, son üç satır ve altı sütunu açısal hızlardan kaynaklanan dönmeyi ifade eder (2.7).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} (\text{öteleme})_{3x6} \\ (\text{dönme})_{3x6} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \quad \textbf{(2.7)}$$

Çizgisel ve açılar hızlar tekrarlanan formda aşağıdaki gibi yazılabilir [9].

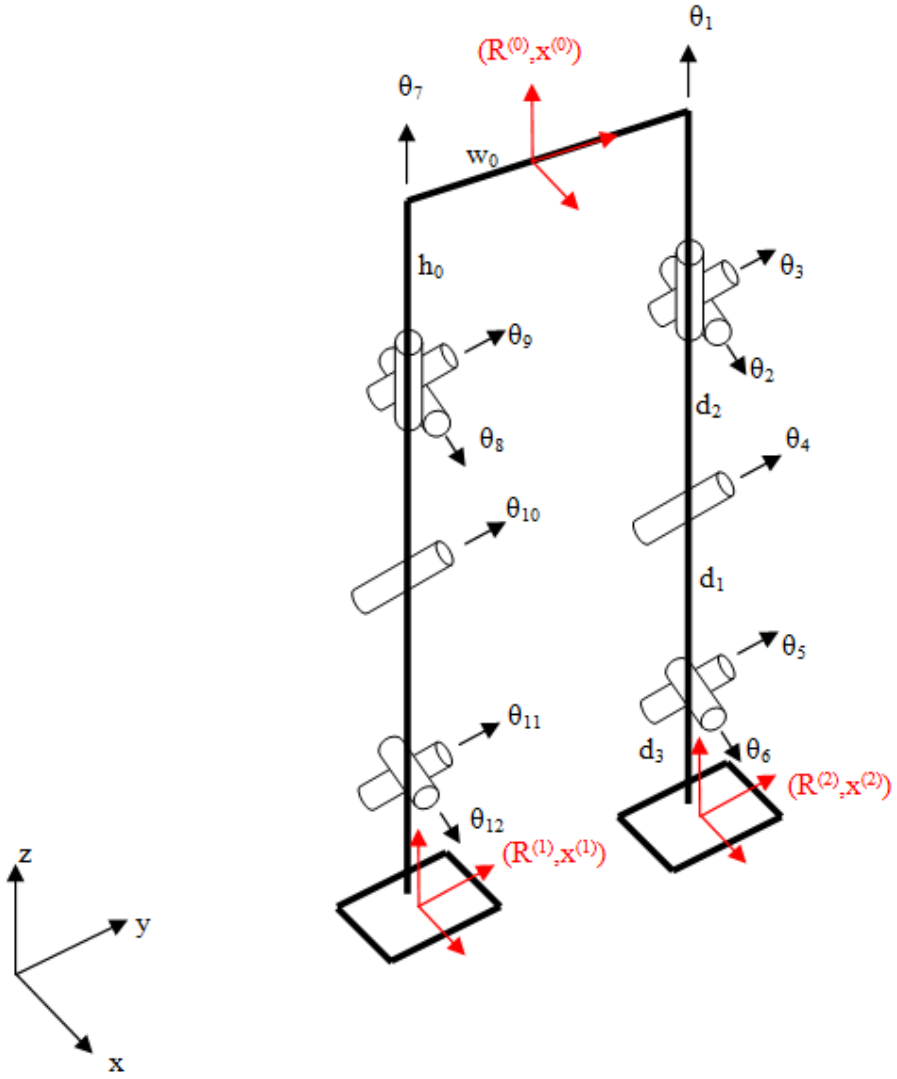
$${}^{i+1}_{i+1}\omega = {}^{i+1}_{i}T\,{}^{i}_{i}\omega + \dot{\theta}_{i+1}\hat{Z}_{i+1} \quad \textbf{(2.8)}$$

$${}^{i+1}_{i+1}v = {}^{i+1}_{i}T({}^{i}_{i}v + {}^{i}_{i}\omega \; x \; {}^{i}_{i+1}O) \quad \textbf{(2.9)}$$

Buna göre,

$$V^0 = \begin{bmatrix} v^0 \\ \omega^0 \end{bmatrix} = J(\theta)^0\dot{\theta} \quad \textbf{(2.10)}$$

şeklinde jakobiyen matrisi çizgisel ve açısal hızların matris-vektör çarpımı haline getirilmesiyle bulunur. (2.8)' deki $\hat{Z}_{i+1}$terimi dönmenin gerçekleştiği eksenlerin birim vektörleri, O, dönmenin gerçekleştiği eksenlere yerleştirilen eksen takımları ve T ise eksenler arası dönmeyi ifade eden rotasyon matrisleri olmak üzere açısal hız vektörleri sol bacak için Şekil 2.1'de gösterildiği gibi ;



Şekil 2.1 : Uzuv uzunlukları, eksen takımları ve serbestlik dereceleri.

$$\dot{\theta}_1\hat{Z}_1 = [0\ 0\ \ \dot{\theta}_1]^T$$

$$\dot{\theta}_2\hat{Z}_2 = [\dot{\theta}_2\ 0\ 0]^T$$

$$\dot{\theta}_3\hat{Z}_3 = [0\ \dot{\theta}_3\ 0]^T \qquad \textbf{(2.11)}$$

$$\dot{\theta}_4\hat{Z}_4 = [0\ \dot{\theta}_4\ 0]^T$$

$$\dot{\theta}_5\hat{Z}_5 = [0\ \dot{\theta}_5\ 0]^T$$

$$\dot{\theta}_6\hat{Z}_6 = [\dot{\theta}_6\ 0\ 0]^T$$

Şekil 2.1 oluşturulan kinematik parametreleri belirten çubuk adam modelidir.

şeklindedir. Sağ bacak için de $\dot{\theta}_6$ ve $\dot{\theta}_{12}$ arası açısal hız vektörleri aynı şekilde yazılabilir. X, Y ve Z eksenlerindeki rotasyon matrisleri (2.12)'de gösterildiği gibi

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix},$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \qquad \textbf{(2.12)}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

olmak üzere eklemler arası dönme matrisleri sol bacak için şöyle olacaktır :

$$\begin{aligned} {}^{0}_{1}T = R_z(\theta_1), {}^{1}_{0}T = R_z(\theta_1)^T \\ {}^{1}_{2}T = R_x(\theta_2), {}^{2}_{1}T = R_x(\theta_2)^T \\ {}^{2}_{3}T = R_y(\theta_3), {}^{3}_{2}T = R_y(\theta_3)^T \\ {}^{3}_{4}T = R_y(\theta_4), {}^{4}_{3}T = R_y(\theta_4)^T \\ {}^{4}_{5}T = R_y(\theta_5), {}^{5}_{4}T = R_y(\theta_5)^T \\ {}^{5}_{6}T = R_x(\theta_6), {}^{6}_{5}T = R_x(\theta_6)^T \end{aligned} \qquad \textbf{(2.13)}$$

Sağ bacak için de θ_7 ve θ_{12} arası aynı işlemler yapılarak dönme matrisleri elde edilir. Jakobiyen matrisini hesaplamak için hazır bir Maple programından yararlanılmıştır. Bu program EK A.6'da sunulmuştur.

2.1.1 Virtüel iş ve jakobiyen

Virtüel iş tanımına göre, sonsuz küçük uzunlukta çizgisel ve açısal yer değiştirmeler ele alınıp, iş denklemi yazılırsa [10] (2.14) elde edilir. (2.14)'teki F kartezyen düzlemde etki eden kuvvet, δx, kartezyen düzlemde sonsuz küçük uzunluktaki çizgisel yer değişimi, δθ ise sonsuz küçük uzunluktaki açısal yer değişimini ifade eder.

$$F\delta x = \tau\delta\theta \quad \textbf{(2.14)}$$

elde edilir. Denklemin sol tarafı kartezyen çalışma uzayındaki virtüel işi, sağ tarafı ise eklem yer değiştirmeleri sırasında yapılan virtüel işi ifade eder. Aynı ifade aşağıdaki şekilde de yazılabilir.

$$F^T\delta x = \tau^T\delta\theta \quad \textbf{(2.15)}$$

Jakobiyenin tanımından dolayı,

$$\delta x = J\delta\theta, \quad \textbf{(2.16)}$$

$$F^T J\delta\theta = \tau^T\delta\theta, \quad \textbf{(2.17)}$$

bu durum bütün $\boldsymbol{\delta\theta}$'lar için geçerlidir. Dolayısıyla,

$$F^T J = \tau^T \quad \textbf{(2.18)}$$

veya,

$$\tau = J^T F \quad \textbf{(2.19)}$$

yazılabilir.

2.2 Ters Kinematik Model

İki ayaklı robotun ters kinematik çözümü ayakların ve gövdenin gitmesi istenen konumlar bilinirken hareket esnasında robotun eklem açılarının alması gereken değerlerin hesaplanmasıdır. Şekil 2.1'de gösterilen modelde kullanılan geometrik parametreler h_0, gövde yüksekliğinin yarısı, w_0, gövde genişliğinin yarısı, d_1, üst bacağın uzunluğu, d_2, alt bacağın uzunluğu ve d_3, ayağın yüksekliğidir. ($\mathbf{R^{(0)}}$, $\mathbf{x^{(0)}}$) gövdenin rotasyon matrisi ve gövdenin merkezinin konumu, ($\mathbf{R^{(i)}}$, $\mathbf{x^{(i)}}$) ise sağ ve sol ayakların rotasyon matrisleri ve konumlarıdır. $\theta^{(i)}$ ise i numaralı bacağın eklem açılarını göstermektedir [11].

$\mathbf{x_h}^{(i)}$ kalça konumunu ve $\mathbf{x_a}^{(i)}$ i numaralı bacağın bilek konumunu göstersin. Bu durumda, $\mathbf{x_h}^{(i)}$ ve $\mathbf{x_a}^{(i)}$ için i numaralı kalça ve bilek konum vektörleri (2.20) ile hesaplanabilir. $w^{(i)}$ ilgili kalça genişliğinin yarısı, $-h_0$ ise gövde yüksekliğinin yarısdır. d_3 ise ayağın yüksekliğidir.

$$\mathbf{x_h^{(i)}} = \mathbf{x^{(0)}} + \mathbf{R^{(0)}} \begin{bmatrix} 0 \\ w^{(i)} \\ -h_0 \end{bmatrix},$$

$$\mathbf{x_a^{(i)}} = \mathbf{x^{(i)}} + \mathbf{R^{(i)}} \begin{bmatrix} 0 \\ 0 \\ d_3 \end{bmatrix}, \quad \textbf{(2.20)}$$

ve bilekten kalçaya olan konum vektörü,

$$\mathbf{x_h^{(i)}} - \mathbf{x_a^{(i)}} \quad \textbf{(2.21)}$$

şeklinde hesaplanır. Ayaklara yerleştirilen eksen takımlarına gore kalçanın konumunu belirten **uvw** vektörü (2.22) ile hesaplanabilir.

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \left(\mathbf{R^{(i)}}\right)^{-1} (\mathbf{x_h^{(i)}} - \mathbf{x_a^{(i)}}) \quad \textbf{(2.22)}$$

Bu bağıntıyla i numaralı bacağın eklem açıları bulunabilir.

$$\theta_6^{(i)} = \tan^{-1}(v/w) \quad \textbf{(2.23)}$$

$$\theta_4^{(i)} = \cos^{-1}((u^2 + w^2 + v^2 - d_1^2 - d_2^2)/2d_1d_2) \quad \textbf{(2.24)}$$

$$\theta_5^{(i)} = \sin^{-1}\left(-d_1 \sin\left(\theta_4^{(i)}\right)/\sqrt{u^2 + w^2 + v^2}\right) - \tan^{-1}\left(u/\sqrt{w^2 + v^2}\right) \quad \textbf{(2.25)}$$

$\theta_1^{(i)}$, $\theta_2^{(i)}$ ve $\theta_3^{(i)}$ ise (2.26) ve (2.27)'deki oryantasyon ilişkisi kullanılarak bulunabilir.

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix},$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \quad \textbf{(2.26)}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_z\left(\theta_3^{(i)}\right)\mathbf{R}_y\left(-\theta_2^{(i)}\right)\mathbf{R}_x\left(-\theta_1^{(i)}\right) =$$
$$\mathbf{R}_x\left(\theta_6^{(i)}\right)\mathbf{R}_y\left(\theta_5^{(i)}\right)\mathbf{R}_y\left(\theta_4^{(i)}\right)(\mathbf{R}^{(i)})^{-1}\mathbf{R}^{(0)} = \quad \textbf{(2.27)}$$
$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\theta_3^{(i)} = -\text{Atan2}(r_{21}, r_{11}), \quad \textbf{(2.28)}$$

$$\theta_2^{(i)} = -\text{Atan2}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}) \quad \textbf{(2.29)}$$

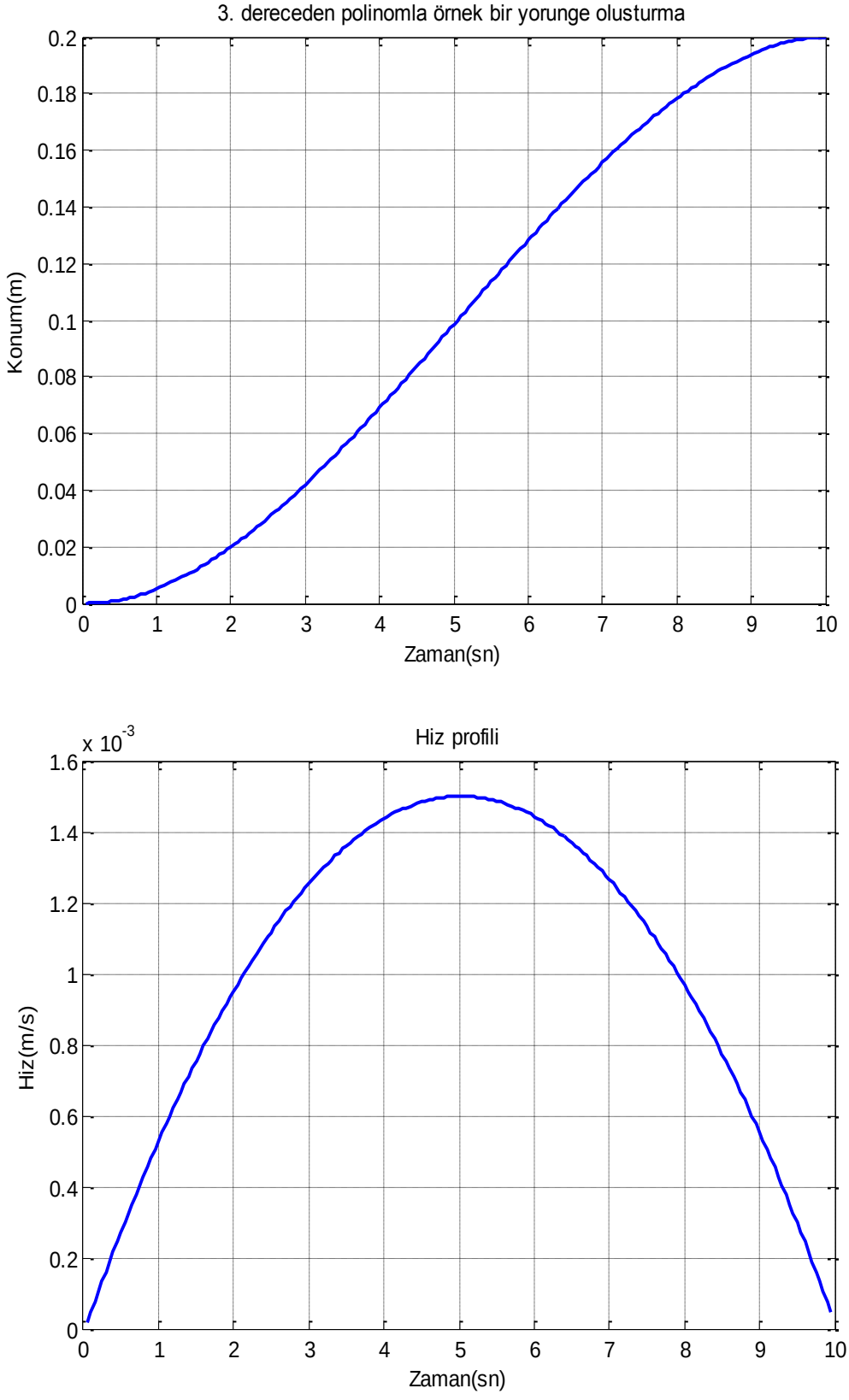
$$\theta_1^{(i)} = -\text{Atan2}(r_{32}, r_{33}) \quad \textbf{(2.30)}$$

2.3 Yörünge Planlama

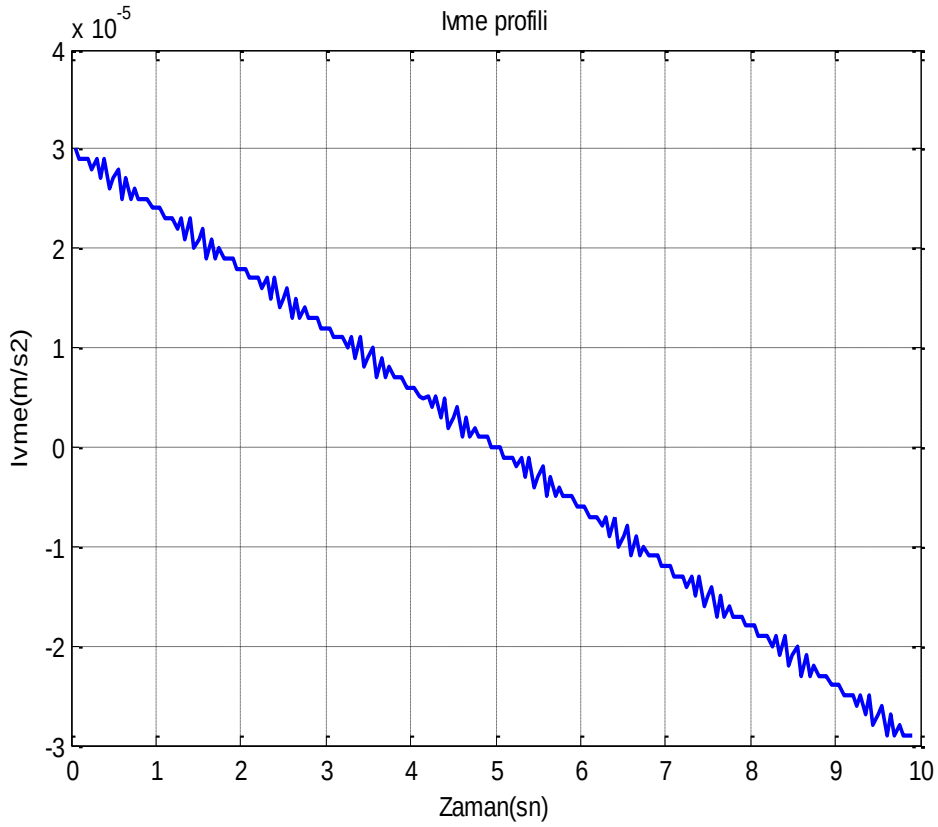
Bu bölümde, ters kinematik ve eklem seviyesinde noktadan noktaya uydurulan polinomlar yardımıyla istenilen yörüngenin oluşturulması ve robota uygulanması anlatılacaktır.

C dilinde yazılan ve EK A.4'te bulunan terskinematik.c programı içindeki 'Yörünge Planlamaları' başlığı altında robotun ayaklarının ve gövdesinin üç boyutlu düzlemde çizgisel ve rotasyonel olarak gitmesi istenen mesafe belirlendikten sonra, program içindeki torsoMove, stepLeft ve stepRight fonksiyonlarının parametreleri girilerek gerekli eklem yörüngeleri oluşturulur. torsoMove fonksiyonunun parametreleri gövdenin çizgisel ve açısal olarak gitmesi istenen üç çizgisel ve üç rotasyonel konumdan ve hareket süresinden oluşmaktadır. stepLeft ve stepRight fonksiyonunun parametreleri ise adım uzunluğu, adım yüksekliği ve hareket süresinden oluşmaktadır. Adım uzunluğu ve adım yüksekliği normal bir insanın yürüme sırasındaki adım uzunluğu ve adım yüksekliğinden yola çıkılarak belirlenir. Hareket süresi uygulanan her referans için 10 saniye olarak belirlenmiştir. Gövdenin referansları 3. dereceden polinomlar yardımıyla oluşturulur. Ayak referans yörüngeleri ise, ayağın kaldırılması ve ileri atılmasına göre 3. ve 5. dereceden polinomlar (Şekil 2.2) ve (Şekil 2.3) uydurularak oluşturulmaktadır. Programın çalışması sırasında fonksiyonlar içerisindeki 3. ve 5. derece polinomlar eş zamanlı

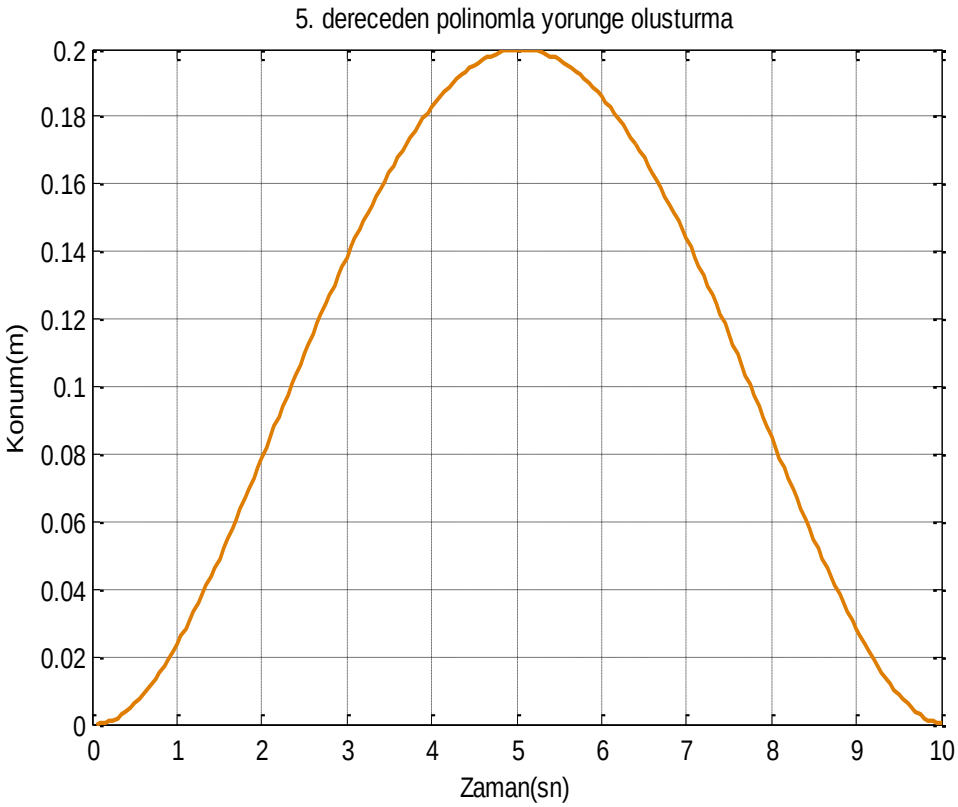
olarak çalışarak ayağın kaldırılması ve ileri doğru adım atılmasını sağlar. Şekil 2.2 ve Şekil 2.3 oluşturulan konum, hız ve ivme profillerini içermektedir.



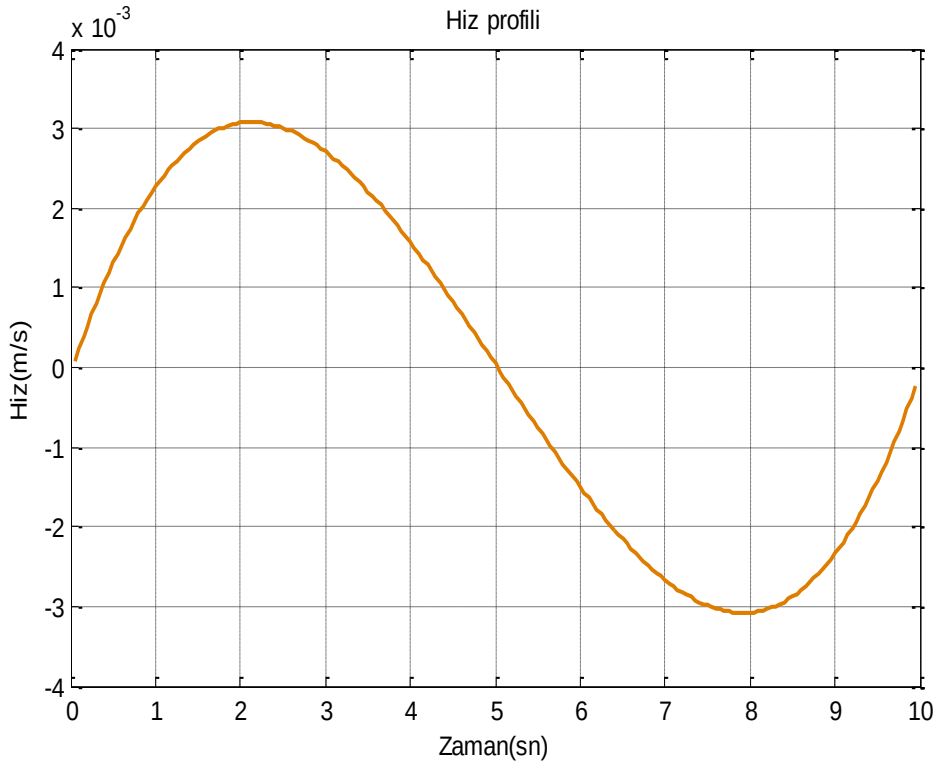
Şekil 2.2 : 3. dereceden polinomla oluşturulan konum, hız, ivme profilleri.

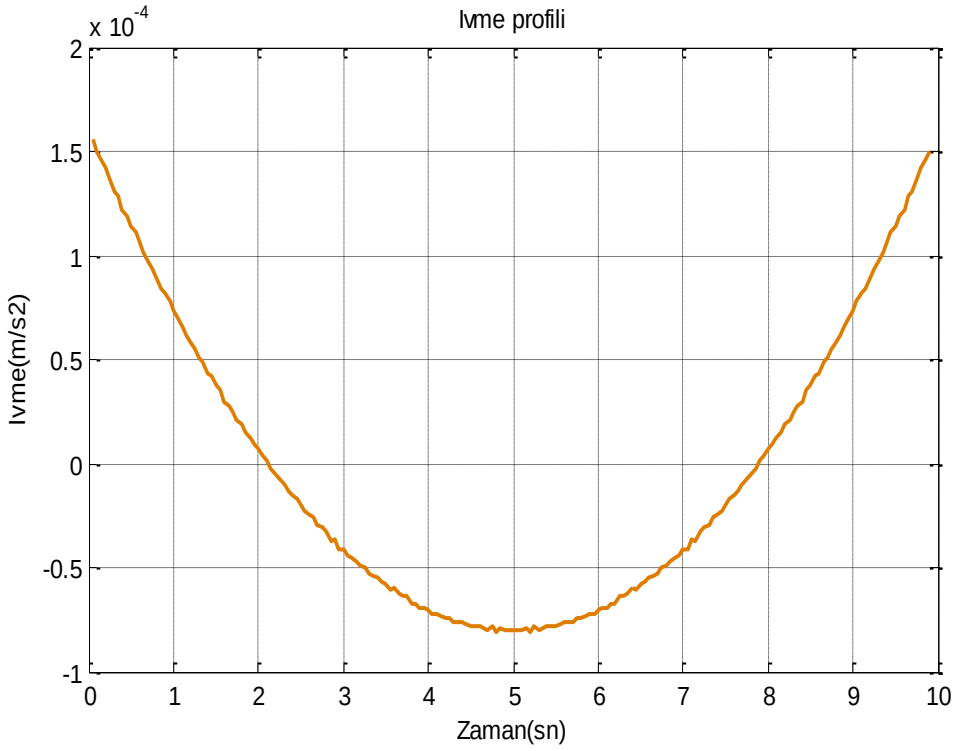


Şekil 2.2 (devam) : 3. dereceden polinomla oluşturulan konum, hız, ivme profilleri.



Şekil 2.3 : 5. dereceden polinomla oluşturulan konum, hız, ivme profilleri.





Şekil 2.3 (devam) : 5. dereceden polinomla oluşturulan konum, hız, ivme profilleri.

Program içindeki pol3 fonksiyonu ile ilk konum, son konum ve hareket süresi bilindiği takdirde her eklem için;

$$k(0) = k_0,$$

$$k(t_f) = k_f,$$

$$\dot{k}(0) = 0 \qquad \textbf{(2.31)}$$

$$\dot{k}(t_f) = 0$$

sınır koşulları uygulanabilir. Burada bahsi geçen k, eklem yörüngeleri, k_0 ve k_f ilk ve son konumdaki eklem yörüngeleri, $\dot{k}(0)$ ve $\dot{k}(t_f)$ eklemlerin ilk ve son konumlarındaki hızlarını belirtmektedir. Dört sınır koşulunu sağlayan

$$k(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \qquad \textbf{(2.32)}$$

kübik polinomu çözülürse a_0, a_1, a_2, a_3 katsayıları bulunur.

Program içindeki bir diğer polinom uydurma fonksiyonu pol5 ile de

$$k(0) = k_0,$$

$$k(t_m) = k_m,$$

$$k(t_f) = k_f, \qquad \textbf{(2.33)}$$

$$\dot{k}(0) = 0,$$

$$\dot{k}(t_m) = 0,$$

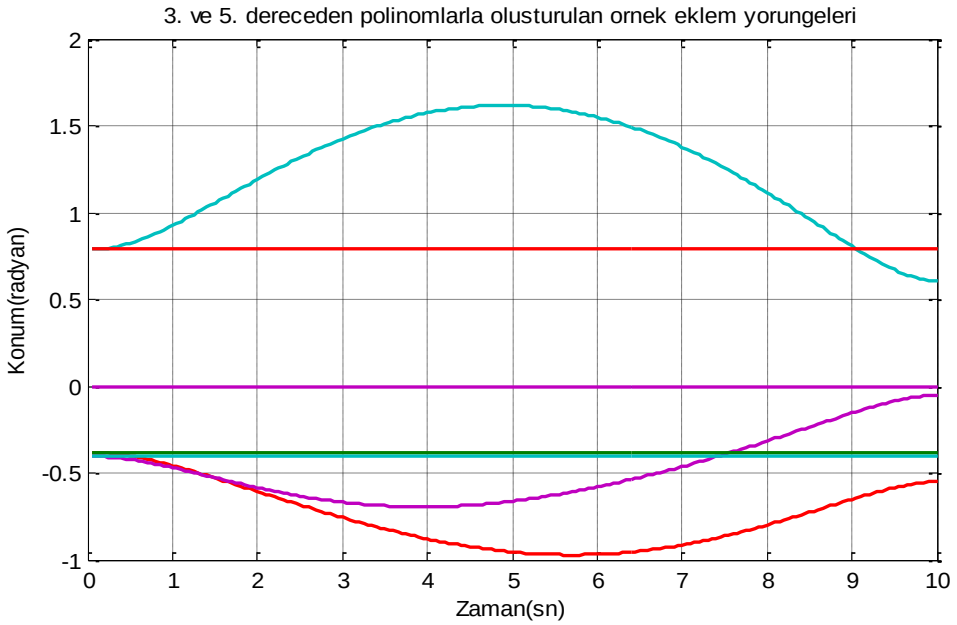
$$\dot{k}(t_f) = 0$$

sınır koşullarını sağlayan

$$k(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \qquad \textbf{(2.34)}$$

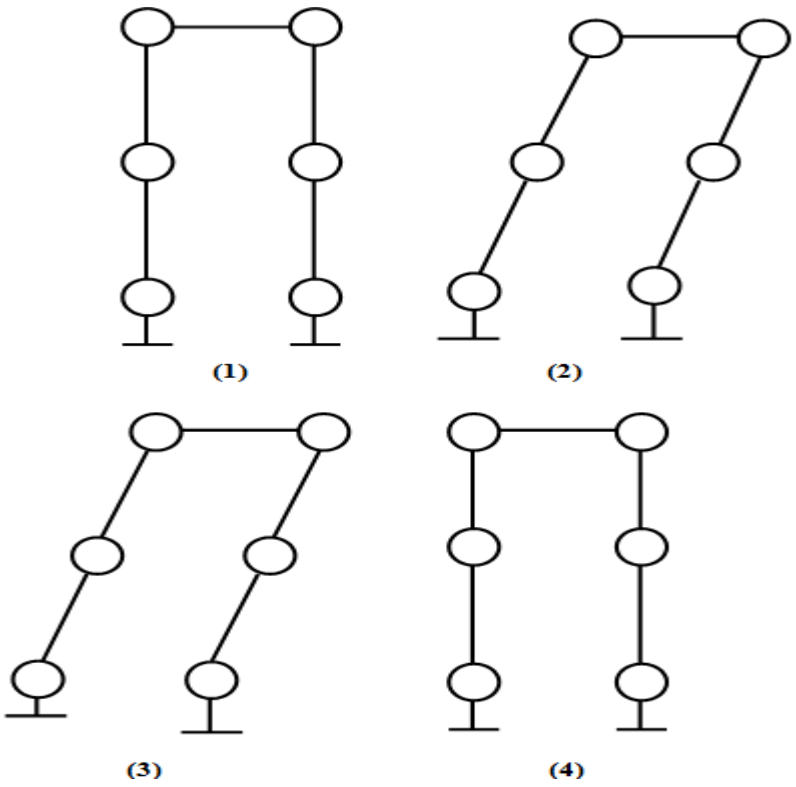
beşinci derece polinomu çözülürse $a_0, a_1, a_2, a_3, a_4, a_5$ katsayıları bulunabilir. Şekil 2.4'te 3. ve 5. derece polinomlar kullanılarak sol ayağın kaldırılması ve ileri atılması için oluşturulan örnek referans eklem yörüngeleri görülmektedir. İlgili gövde ve ayak yörüngeleri oluşturulmadan önce kartezyen koordinatlarda adım parametreleri girilir ve metre cinsinden gövdenin ve ayakların X, Y ve Z eksenlerindeki ötelemeleri seçilen örnekleme periyoduna göre ayrıklaştırılarak hesaplanır. Ayrıklaştırılan yörüngeler ters kinematik hesabıyla 12 eklem için hareket

süresince radyan cinsinden hesaplanır ve robota uygulanır. Hesaplama işlemleri robot hareket etmeden önce çevrim dışı yapılmaktadır.



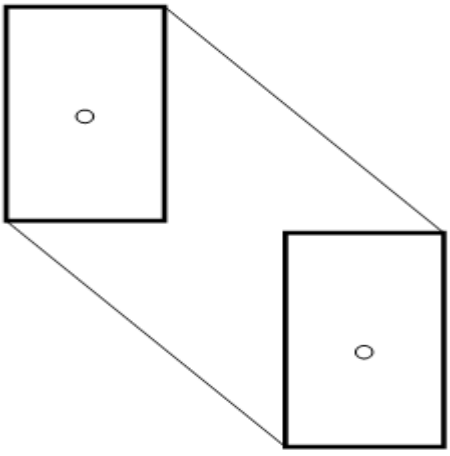
Şekil 2.4 : Örnek referans eklem yörüngeleri.

Şekil 2.5'te ise sol ayağın kaldırılıp ileri doğru adım atılması için yapılan hareketler sırasıyla gösterilmiştir.



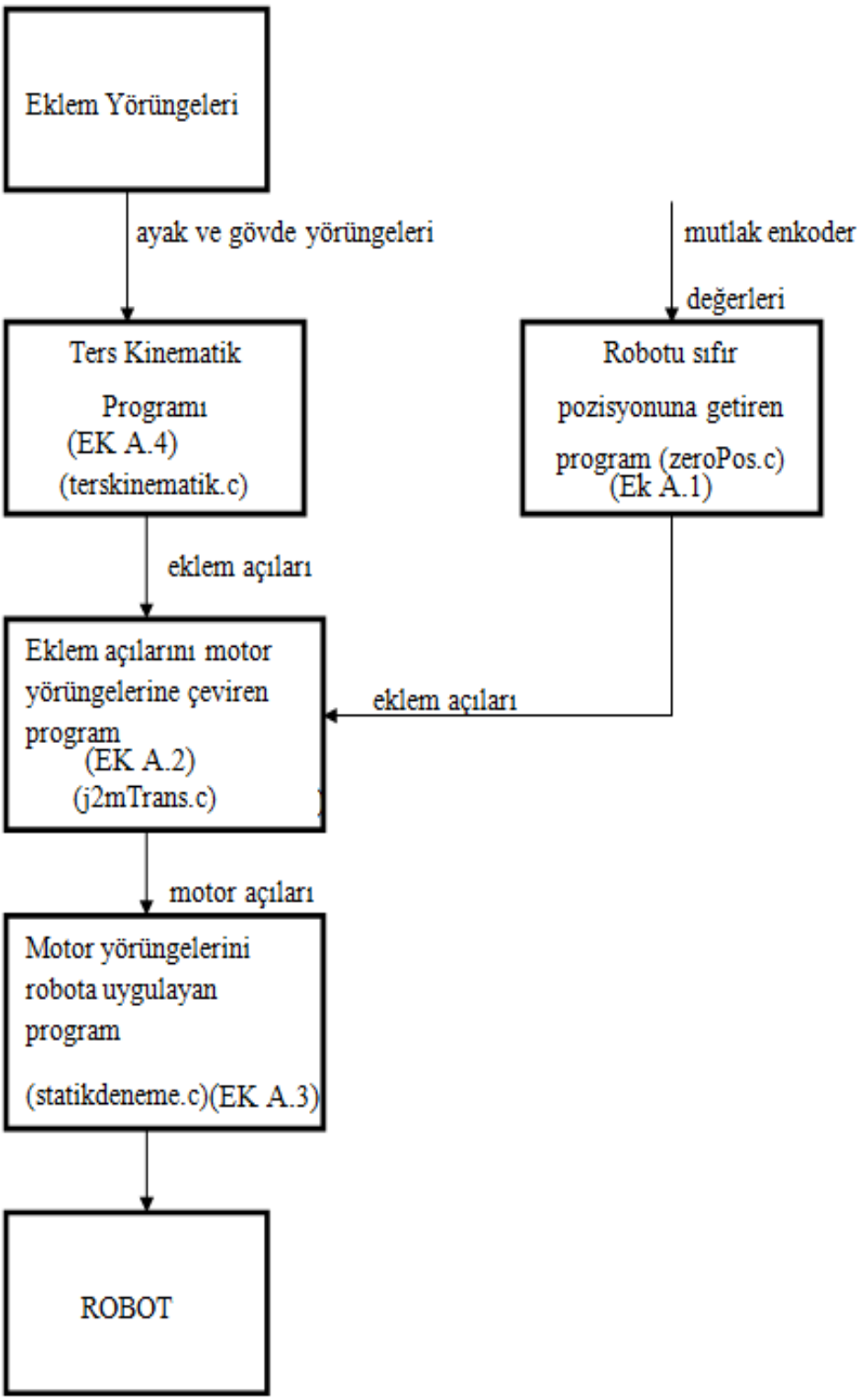
Şekil 2.5 : Sol ayağın adım atması.

Daha önce belirlenmiş eksen takımlarına göre ileri doğru hareket X ekseninde, yanal hareket Y ekseninde, yukarı doğru hareket ise Z ekseninde gerçekleşmektedir. Robotun hareketi, önce Y ekseninde gövdenin, atılacak adımın sağ veya sol olmasına göre ters yöne yanal hareket gerçekleştirmesi, gövdenin hareketinden sonra bir ayağın adımını atması, gövdenin konumunu gerideki ayaktan ilerideki ayağa değiştirmesi ve son olarak ta gerideki ayağın adımını atması şeklinde planlanmıştır. Hareket esnasında denge sağlanabilmesi için robotun ağırlık merkezinin izdüşümünün ayaklar arasında oluşan destek çokgeni (Şekil 2.6) içerisinde kalması gerekir. Bu yüzden gövdenin yer değişimi iki ayağın da yerle temasının olduğu ÇDF, ayakların yer değişimi ise yalnızca tek ayağın yerle temasının olduğu TDF'de gerçekleşir. Adım atmadan önce adım atacak ayağa göre gövdenin ters yöne hareketinin sebebi bahsedildiği gibi ağırlık merkezi izdüşümünün yere basan ayağa doğru kaydırılması, dolayısıyla izdüşümün destek çokgeni içerisinde kalmasının sağlanmasıdır. Gövde için torsoMove fonksiyonuna hangi eksende hareket edilecekse o eksen için gerekli uzunluk girilerek 3. dereceden polinom yardımıyla gövde yörüngesi üretilir. Ayaklar içinse stepRight ve stepLeft fonksiyonlarına adım uzunluğu girilerek X eksenindeki yörünge 3.dereceden ve adım yüksekliği girilerek Z eksenindeki yörünge 5.dereceden polinomlar yardımıyla üretilir ve bahsedilen fonksiyonların program içinde tekrarlamalı olarak çağırılmasıyla istenilen yürüme yörüngesi elde edilir ve bir dosyaya kaydedilir. Dosya içinde bulunan her örnekleme periyodundaki yörünge değerleri, CalcR fonksiyonu içerisinde ters kinematik işlem yardımıyla eklem yörüngelerine radyan cinsinden çevrilir.



Şekil 2.6 : Ayaklar tarafından oluşturulan destek çokgeni.

Şekil 2.7'de robotun sıfır pozisyonuna getirildikten sonra planlanan yörüngeyi gerçekleştirmesi için uygulanması gereken adımlar verilmiştir. Ters kinematik programı yazılarak robotun eklem açıları gömülü PC ile üretilebilir hale gelmiştir.



Şekil 2.7 : Hareket uygulama şeması.

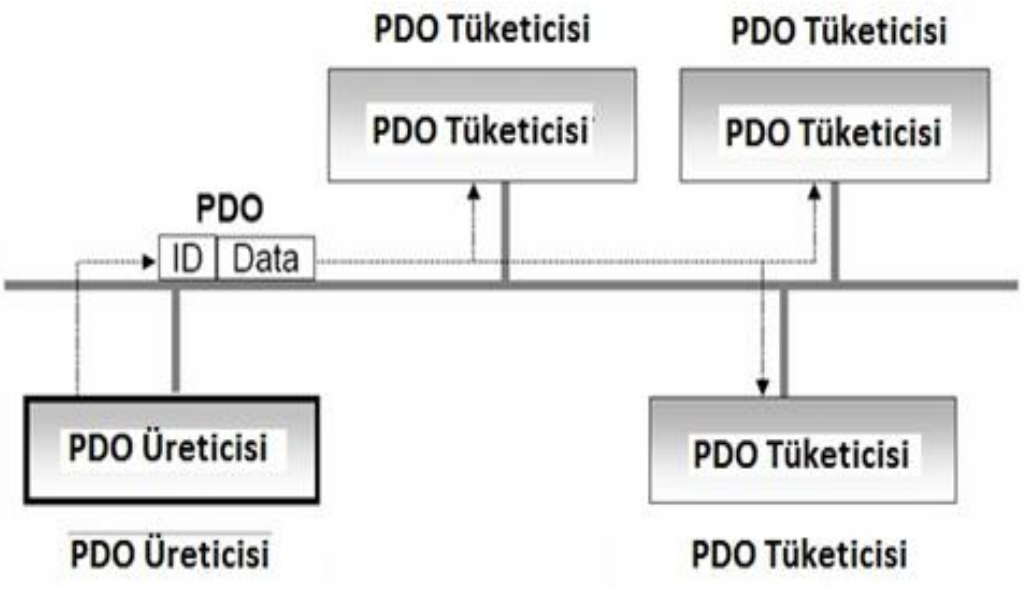
3. CAN bus İLETİŞİMİ

3.1 Kullanılan Can bus İletişim Nesneleri

İki ayaklı robot üzerinde daha önce yapılan deneysel çalışmalarda CANopen iletişim protokolü içerisinde bulunan SDO kullanılmıştır [12]. SDO, motor sürücüleri içerisinde bulunan nesne sözlüğüne erişmek ve nesne sözlüğü içerisinde değişiklik yapmanın yanı sıra sistemin hareketi sırasında veri gönderme ve okuma için de kullanılabilir. Ancak gönderilen ve alınan verilerin onaylanma sürecinden geçmesi ve Can bus üzerinden gönderilen mesaj formatının her durum için 8 byte büyüklüğünde sabit olması bu iletişim nesnesinin gerçek zamanlı sistemler için kullanılmasını imkansız kılar. İlk yapılan çalışmalarda robotu sıfır konumuna getirmek için uygulanan program süresince 10 sn sürmesi gereken hareket yaklaşık 50 sn içinde tamamlanmıştır. Bu nedenle SDO yerine daha hızlı veri alışverişine olanak sağlayan PDO nesnesi robota uygulanmıştır. Bundan sonraki bölümlerde PDO nesnesi ve PDO içerisinde bulunan iletişim modları tanıtılıp hareketin normal süre içerisinde gerçekleşmesi için elde edilen minimum örnekleme periyodundan deneysel çalışmalar bölümünde bahsedilecektir.

3.1.1 PDO

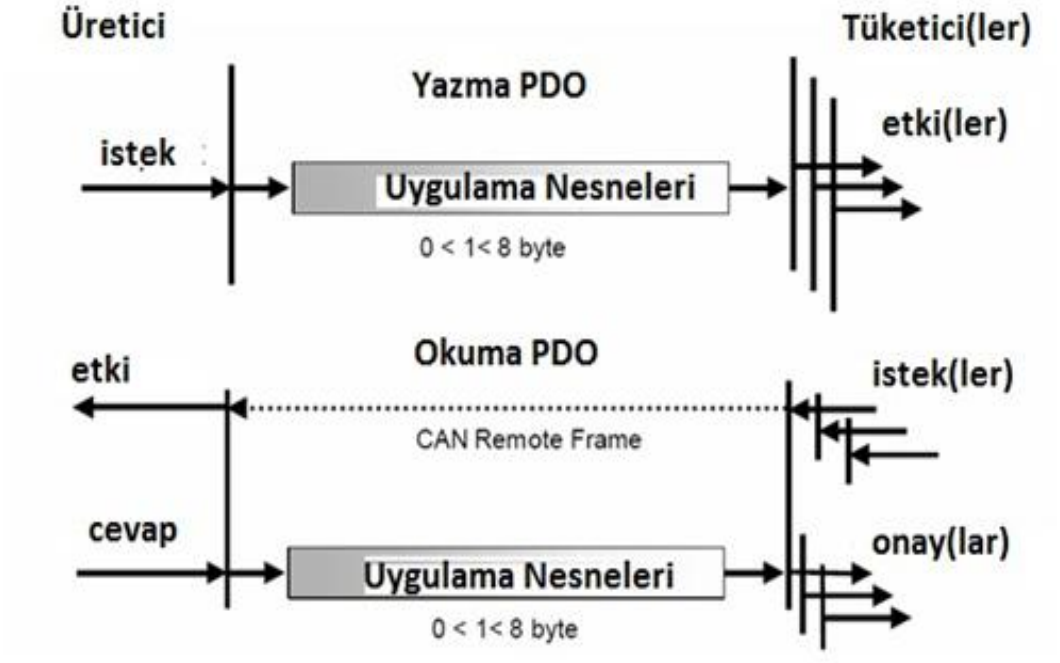
PDO, veriyolu içerisinde yüksek önceliğe sahip ve gerçek zamanlı uygulamalar için hızlı veri iletişimi sağlayan bir CANopen protokolüdür. PDO'lar üretici/tüketici modelini kullanan, onaylanmayan servislerdir, bu sayede bir sürücü üzerinden herhangi sayıda sürücüye veri aktarımı hızlı bir biçimde gerçekleşir (Şekil 3.1). SDO protokolü çerçevesinde haberleşme sırasında aynı anda sunucu ve bir sürücü arasında veri aktarımı mümkünken, sürücü ve bilgisayar arası iletişim için PDO protokolü kullanıldığında veri gönderimi bir sürücü tarafından başlatılıp, birden fazla sürücü tarafından veri kabulü yapılabilmektedir. Sürücü içerisinde konfigüre edilebilen 4 adet Yazma PDO(Receive-PDO) bulunur ve kısaca RxPDO olarak adlandırılır. Ayrıca 4 adet Okuma PDO(Transmit-PDO) bulunur ve kısaca TxPDO olarak adlandırılır.



Şekil 3.1 : PDO iletişim modeli[13].

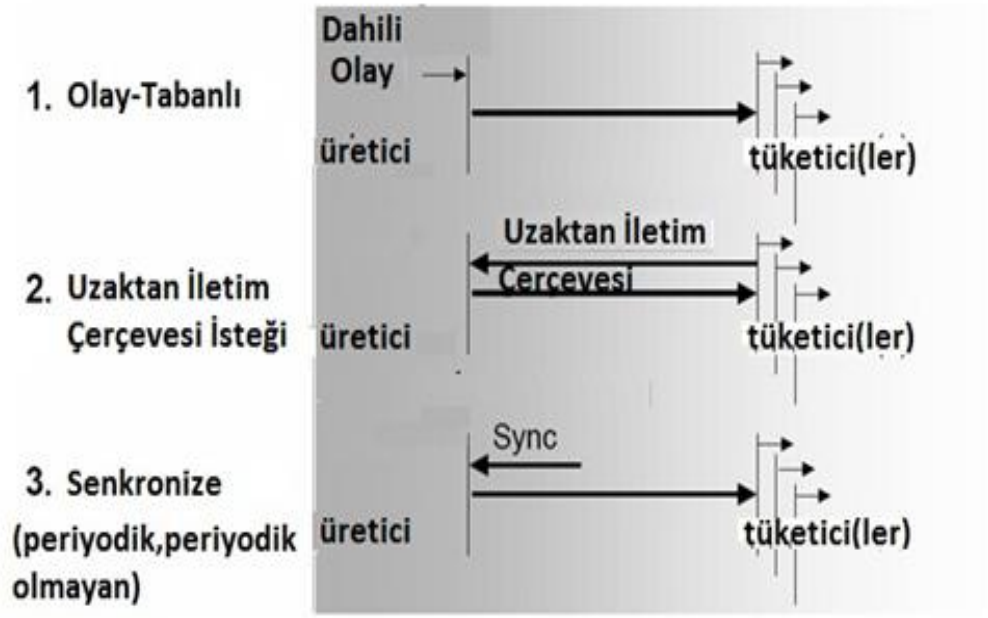
PDO'lar kullanıcının ihtiyacına göre maksimum 8 byte uzunluğunda proses verisi taşıyabilir. PDO üreten sürücü belli bir ID kullanarak bir veya daha fazla sayıda sürücünün Receive-PDO (RxPDO) ID'lerine karşılık gelen bir Transmit-PDO (TxPDO) üretir.

Yazma ve okuma olmak üzere iki adet PDO servisi vardır. Yazma-PDO tek bir CAN veri çerçevesine adreslenmiştir. Okuma-PDO ise bir CAN Remote Frame üzerinden gerçekleştirilen istek üzerine bir CAN veri çerçevesine adreslenir(Şekil 3.2).



Şekil 3.2 : PDO protokolü[14].

CANopen iletişim protokolünde 3 farklı mesaj tetikleme modu bulunmaktadır (Şekil 3.3).

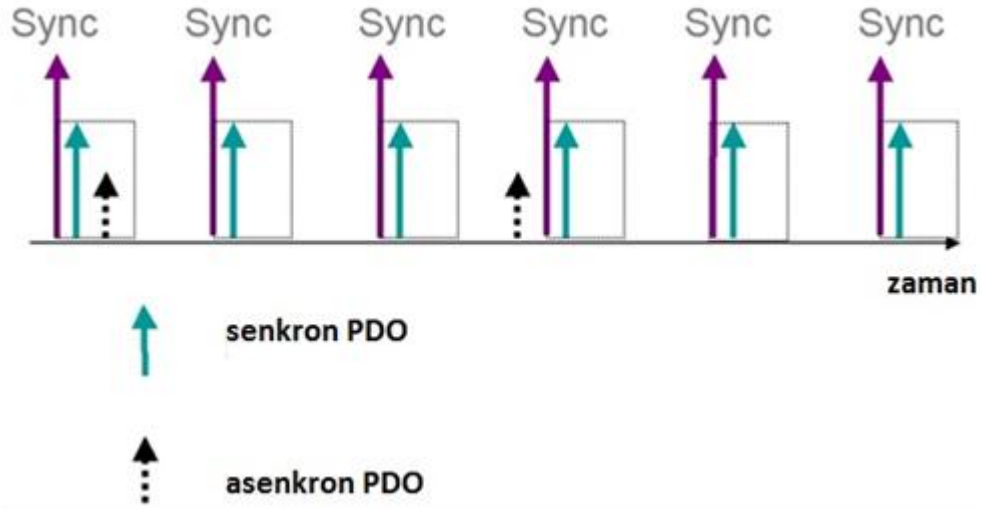


Şekil 3.3 : PDO iletişim modları [15].

Bunlardan birincisi, sürücü içerisinde tanımlanmış bir nesneye özgü olayın meydana gelmesiyle mesaj iletiminin tetiklenmesidir. Periyodik olarak veri ileten sürücüler herhangi bir olay meydana gelmese bile zamanlayıcı yoluyla ek olarak tetiklenir. İkinci iletim modu ise Remote Transmission Request (RTR) yani uzaktan iletim isteğidir. Bu modda asenkron PDO'ların iletimi başka bir cihaz (örn. Gömülü PC) tarafından gönderilen RTR ile başlatılabilir. Üçüncü iletim modu ise sürücü nesne sözlüğü içerisinde tanımlanan bir senkronizasyon nesnesi kullanılarak, bu nesne içerisinde belirtilmiş iletim periyodunun sonlanmasıyla senkron PDO'ların gönderiminin başlatılabildiği senkronize iletim modudur.

CANopen kullanıcıya iki farklı PDO iletim modu sunar. Bu iletim modları senkronize ve asenkronize modlarıdır(Şekil 3.4).Senkronize PDO'lar senkronizasyon nesnesi gönderildikten sonra senkronize penceresi içerisinde iletilirler. Asenkronize PDO'lar ise önceliklerine göre haberleşme sırasında her zaman gönderilebilecekleri için, senkronize pencere içerisinde de iletilebilirler. Senkronize PDO'ların önceliği asenkronize PDO'lardan daha yüksektir. Sürücüler üzerinde yapılan PDO konfigürasyonunda deneysel çalışmalar bölümünde bahsedileceği üzere hem senkron hem de asenkron PDO iletim tipleri kullanılmıştır. Asenkron PDO iletim tipi için

sürücü üzerinde tanımlanmış olan Uzaktan İletim İsteği (RTR) mesaj tetikleme modu mevcuttur.



Şekil 3.4 : PDO iletim tipleri [16].

3.2 Kuvvet/Moment Sensörlerinin Devreye Alınması

Sistem üzerindeki UNO2052E gömülü bilgisayar, iki farklı CAN portuna sahiptir ve bunlardan biri daha önce bahsedildiği gibi, motor sürücüleri ve bilgisayar arasındaki iletişimi sağlar. İkinci CAN portu ise, robotun özellikle dinamik yürüme hareketinin kontrolünde büyük öneme sahip olacak olan kuvvet/moment sensörleri ile bilgisayar arasındaki iletişim sağlar. Bu çalışmada, daha önce yapılan çalışmalara [17],[18] ek olarak, kuvvet/moment sensörleri devreye alınarak, arayüz ve bilgisayar arasındaki iletişim için gerekli kodlar C dilinde yazılmış ve uygulanmıştır.

3.2.1 Protokol açıklaması

Bir istek veri mesajı gönderilir ve ATI Net F/T arayüzü, mevcut kuvvet/moment değerlerinin çıkış arabelleğine kopyalanması ve bunu izleyen adımda çıkış arabelleğinin iletimi işlemlerini başlatır. İstek mesajı kimliğine bağlı olarak, Net F/T ya 4 mesaj içerisine paketlenmiş 32 bitlik veri ya da 2 mesaj içerisine paketlenmiş 16 bitlik veri gönderir [19]. Çizelge 3.1'de kısa veri formatının mesaj yapısı açıklanmıştır. Kısa veri isteği gönderildikten sonra kısa formatta kuvvet/moment değerlerinin kopyası gönderilir ve sırasıyla X ve Y ekseni kuvvet/moment değerleri ve son olarak ta Z ekseni kuvvet/moment değerleri, sistem durum kelimesi ve örnek sayısı gönderilir.

Çizelge 3.1 : Kısa veri isteği formatı[20].

Giden Mesaj	Gelen Cevap	CAN Kimliği	Veri Uzunluğu	1.-4.byte arası	5.-8.byte arası	Açıklama
Kısa veri iste		0	1 byte	0x02(byte)	N/A	Kısa formatta kuvvet/Moment değerlerinin kopyası
	Fx,Tx,Fy,Ty	5	8 byte	Fx(INT) Tx(INT)	Fy(INT) Ty(INT)	Kısa formatta X ve Y eks. kuvvet/momentleri
	Fz,Tz,durum ve örnek sayısı	6	8 byte	Fz(INT) Tz(INT)	Sistem durumu(INT) Örnek sayısı(INT)	Kısa formatta Z eks.kuvv./mom., durum kelimesi ve örnek sayısı

Net F/T arayüz birimi, her kuvvet/moment değerini CAN arayüzü üzerinden göndermeden önce belirli bir faktörle çarpar. Bu sayede kuvvet/moment değerleri tam çözünürlükle gönderilmiş olur. Dolayısıyla gerçek değerleri elde etmek için, programın içinde kuvvet/moment değerlerinin bu faktöre bölünmesi gerekir (3.1).

$$F = \frac{F_{data}}{32{,}77}$$

$$M = \frac{M_{data}}{1310{,}62} \qquad \textbf{(3.1)}$$

Yazılan programlardan biri olan ftzero.c sensor değerlerinin robot yere basmadan önce sıfırlanması için kullanılır. Bu durum robot havadayken sensörlerde oluşan sapma değerlerini gidermek için önemlidir. Çizelge 3.2'de sıfırlama komutunun yapısı verilmiştir.

Çizelge 3.2 : Sıfırlama komutunun yapısı[21].

Giden mesaj	Gelen cevap	CAN kimliği	Veri uzunluğu	1.-4.byte arası	5.-8.byte arası	Açıklama
Sıfırla		0	1 byte	0x04(byte)	N/A	Mevcut yükleme seviyesinde kuvv./mom. değerlerini sıfırlar.

Ayaklar yere bastıktan sonra ölçülen gerçek kuvvet/moment değerlerini okumak için EK A.3'te bulunan statikdeneme.c programı içindeki sendCanFT1 ve sendCanFT2 fonksiyonları kullanılır.

4. DENEYSEL ÇALIŞMALAR

Bu bölüm, iletişim yapısı ve tipleri açıklanmış olan PDO protokolüyle deneysel olarak elde edilebilmiş en küçük örnekleme periyodunun nasıl bulunduğu ve eklem momentlerinin, iki farklı yoldan, motorların çektiği akımlar ve kuvvet/moment sensörleri üzerinden alınan ölçümlerle deneysel olarak bulunarak karşılaştırılması konularını kapsamaktadır.

4.1 En Uygun Örnekleme Periyodunun Belirlenmesi

İki ayaklı robot üzerinde daha önce yapılan çalışmalarda teorik olarak CAN bus iletişimi için gerek ve yeterli örnekleme periyodu 10 ms olarak öngörülmüştü. Seçilen CAN bus hızının 1 MBit/s olduğu, bir CAN mesajı gönderip-almanın 11 bayt uzunluğa sahip olduğu ve motor sürücülerine konum bilgisi göndermek için toplam 176 bit uzunluklu bir CAN bus mesajının bir sürücü için 176 mikro saniye süreceği, bu veriler göz önüne alındığında 12 sürücü kartı için toplam mesaj transfer süresinin 2.1 mili saniye olacağı hesaplanmıştı. Ancak daha sonra yapılan deneysel çalışmalarda sistemin 10 ms örnekleme periyoduyla 10 sn içinde yapması gereken düşey ve yatay hareketlerin ancak 50 sn içinde yapıldığı gözlenmiştir ve bu nedenle sistem iletişiminde revizyona gidilmesi gerekliliği ortaya çıkmıştır. Bahsedilen revizyonu gerçekleştirmek adına motor sürücülerinin konfigürasyonu için kullanılan SDO protokolü yerine daha hızlı ve daha esnek bir iletişim imkanı sağlayan PDO protokolüne geçilmesine karar verilmiştir.

Minimum örnekleme zamanını bulmak için robota üç farklı senaryo uygulanmıştır. Bunlardan ilkinde, hem sürücüye veri yazdırmak, hem de sürücüden veri okumak için Asenkron PDO modu kullanılmıştır. Gözlemlenen deney sonuçlarına göre 10 ms örnekleme periyodu ile robot 10 sn'lik sıfır pozisyonuna getirme hareketini 32 sn içinde, 25 ms örnekleme periyodu ile aynı hareketi 10 sn içinde yapmayı başarabilmiştir. 25 ms ile hareket istenilen sürede yapılsa da, kuvvet/moment sensörlerinin daha sonra devreye alınarak işlem yükünü artıracağı öngörüldüğünden örnekleme periyodunun 25 ms'den daha aşağıda olması gerekmektedir. Bu nedenle,

ikinci senaryoda veri yazdırma için Asenkron PDO, veri okumak içinse Asenkron PDO'dan iletişim bazında önceliği olan Senkron PDO kullanılmıştır. Bu durumda, robot hareketi 10 ms örnekleme periyodu ile 50 sn içinde, 25 ms örnekleme periyodu ile 35 sn içinde yapmıştır. Gözlemlenen süreler istenilen sürelerden oldukça uzaktır ve hareket süresinde iyileşme beklenirken daha yavaş bir hareket meydana gelmiştir. Son olarak, üçüncü senaryoda ise veri yazdırma için Asenkron PDO, motorların üzerindeki artımlı enkoderlerin değerlerini okumak için Asenkron PDO ve motorların çektiği akımları okumak içinse Senkron PDO iletişim modu kullanılmıştır. Asenkron PDO ile artımlı enkoderden veri okuma işlemi EKA.2'deki j2mTrans.c programı içindeki sendCanPDOinc() fonksiyonuyla gerçekleştirilir. Akım okuma işlemi ise statikdeneme.c programı içindeki sendCanPDOsync() ve sendCanPDOcurrent() fonksiyonlarıyla gerçekleştirilir. Son senaryoda elde edilen süreler 10 ms için 16 sn, 25 ms için 10 sn ve 15 ms için 10 sn'dir. Dolayısıyla mevcut iletişim protokolüyle en uygun örnekleme periyodu 15 ms olarak elde edilmiştir. Uygulanan örnekleme periyodu için mesaj transfer süresi hesaplanırsa : Her bir sürücü kartı için 4 adet TxPDO ve 4 adet RxPDO gönderilip alınmıştır. PDO mesajları içerisinde toplam 9 baytlık veri taşımaktadır ve uygulanan durum için toplam 108 baytlık veri gönderilip alınmalıdır. Toplam transfer süresinin 12 sürücü kartı için 1728 bit transferi (1728 mikro saniye) ve 1,728 mili saniye olduğu hesaplanır.

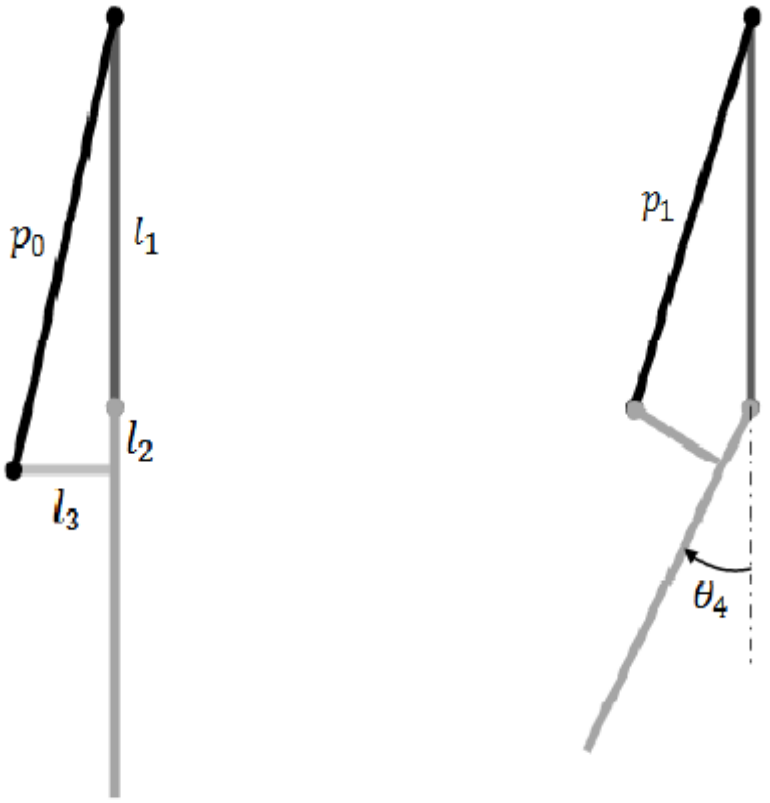
4.2 Eklem Momentlerinin Belirlenmesi

İki ayaklı robota deneysel çalışmalar bazında çeşitli hareket yörüngeleri uygulanmıştır. Bunlar, iki ayak üzerinde 10 cm. çömelip kalkma, gövdenin 7, 10 ve 13cm. uzunluklarla sağa veya sola hareketlerini kapsamaktadır. Hareketler esnasında motorların çektiği akımlar yardımıyla hesaplanan eklem moment davranışlarıyla, kuvvet/moment sensörleri üzerinden alınan ölçümlerle elde edilen eklem moment davranışlarının küçük bir farkla birbirini takip etmesi gerekmektedir. Bu fark, deneysel çalışmalar sırasında öngörülemeyen sürtünme, sensor gürültüsü vs. gibi faktörlerle ortaya çıkan kayıplar olup, ilerleyen çalışmalarda mekanik ve elektronik ekipmanın iyileştirilmesiyle daha küçük seviyelere indirilebilir.

Akımlar yoluyla kalça eklemleri için hesaplanan eklem momentleri (4.1)'deki gibi elde edilir :

$$\tau = Motorun\ çektiği\ akım(A) x Motorun\ tork\ sabiti(\frac{Nm}{A}) \\ x\ Aktarma\ organının\ çevrim\ oranı \tag{4.1}$$

Diz eklemlerindeki momentleri hesaplamak için Şekil 4.1'deki geometrik modelden yararlanılmıştır. Motor çıkışının oluşturduğu dairesel hareketin vidanın doğrusal hareketine çevrilmesi için gerekli kinematik denklemler (4.2) ve (4.3)'ten yararlanılarak bulunabilir[22].



Şekil 4.1 : Diz eklemi momentini hesaplamak için kullanılan geometrik model [23].

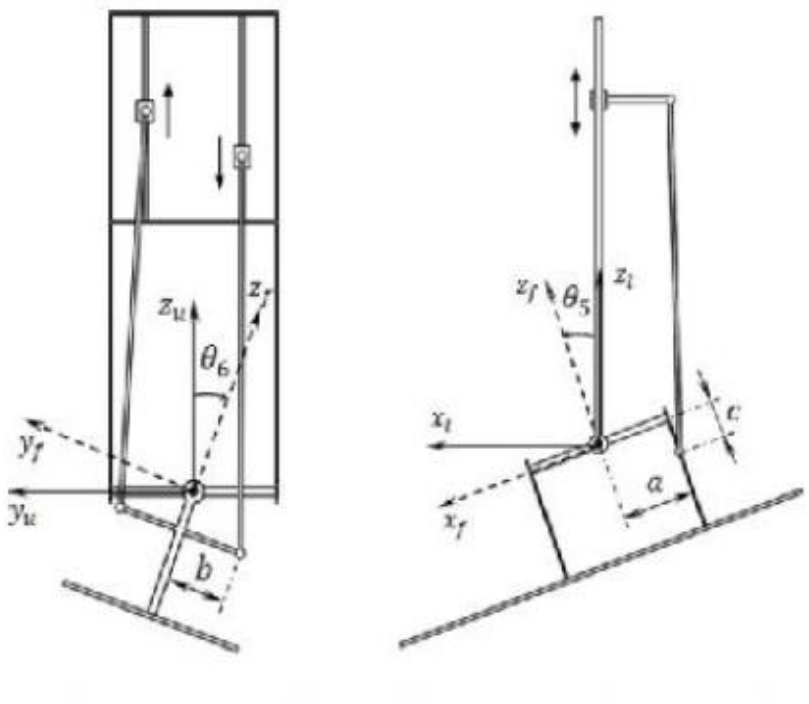
$$\Delta p = p_0 - p_1 = \sqrt{l_3^2 + (l_1^2 + l_2^2)} \\ - \sqrt{l_1^2 + l_2^2 + l_3^2 - 2l_1\sqrt{l_2^2 + l_3^2}\cos\left(\pi - tan^{-1}\left(\frac{l_3}{l_2}\right) - \theta_4\right)} \tag{4.2}$$

Diz eklemi momentini hesaplamak için kullanılan geometrik model nonlinear denklemler içermektedir. Bu durumun sebebi, vidanın bir eklem açısındaki hareket etme miktarı ile başka bir eklem açısındaki hareket etme miktarının eşit olmamasıdır.

$$\frac{\Delta p}{\dot{\theta}_4} = \frac{l_1\sqrt{l_2^2 + l_3^2}\sin\left(\pi - tan^{-1}\left(\frac{l_3}{l_2}\right) - \alpha\right)}{\sqrt{l_1^2 + l_2^2 + l_3^2 - 2l_1\sqrt{l_2^2 + l_3^2}\cos\left(\pi - tan^{-1}\left(\frac{l_3}{l_2}\right) - \theta_4\right)}} \quad \textbf{(4.3)}$$

Denklem (4.2) ve (4.3)'teki Δp vidanın lineer yer değiştirmesini, θ_4 ise diz eklem açısını ifade etmektedir.

Bilek eklemlerindeki momentleri bulmak içinse Şekil 4.2'deki geometrik modelden yararlanılmıştır. Şekil 4.2'deki θ_5 ve θ_6 sırasıyla y ve x eksenlerindeki serbestlik derecelerini belirtmektedir. Bilek eklemindeki aktarma organında yer alan kardan milinin merkezine yerleştirilmiş R_l, R_u, R_f referans eksen takımları, sırasıyla alt bacağa, kardan miline ve ayağa sabitlenmiştir. x_0, y_0, z_0 ve x_1, y_1, z_1, R_l eksen takımında küresel mafsalların koordinatlarıdır[24]. Kinematik model, eksen takımları arasında koordinat dönüşümleri yapılarak ve ilgili parametreler kullanılarak bulunabilir.



Şekil 4.2 : Bilek eklemi momentini hesaplamak için kullanılan geometrik model [25].

Diz ve bilek eklemi için çıkarılan kinematik modeller ayrıca eklem referanslarının motor referanslarına dönüştürüldüğü j2mTrans.c programı içinde kullanılmıştır. Bu program EK A.2'de sunulmuştur.

$$x_0 = a\cos\theta_5 + b\sin\theta_5\sin\theta_6 + c\sin\theta_5\cos\theta_6 \quad \textbf{(4.4)}$$

$$y_0 = b\cos\theta_6 + c\sin\theta_6 \quad \textbf{(4.5)}$$

$$z_0 = -a\cos\theta_5 + b\cos\theta_5\sin\theta_6 + c\cos\theta_5\cos\theta_6 \quad \textbf{(4.6)}$$

$$z_1 = \sqrt{l^2 - (x_1 - x_0)^2 - (y_1 - y_0)^2} + z_0 \quad \textbf{(4.7)}$$

$$\dot{x}_0 = -a\cos\theta_5\dot{\theta}_5 + b\left(\cos\theta_5\sin\theta_6\dot{\theta}_5 + \sin\theta_5\cos\theta_6\dot{\theta}_6\right) + c(\cos\theta_5\cos\theta_6\dot{\theta}_5 + \sin\theta_5\sin\theta_6\dot{\theta}_6) \quad \textbf{(4.8)}$$

$$\dot{y}_0 = -b\sin\theta_6\dot{\theta}_6 - c\cos\theta_6\dot{\theta}_5 \quad \textbf{(4.9)}$$

$$\dot{z}_0 = -a\cos\theta_5\dot{\theta}_5 + b\left(\sin\theta_5\sin\theta_6\dot{\theta}_5 + \cos\theta_5\cos\theta_6\dot{\theta}_6\right) + c(-\sin\theta_5\cos\theta_6\dot{\theta}_5 - \cos\theta_5\sin\theta_6\dot{\theta}_6) \quad \textbf{(4.10)}$$

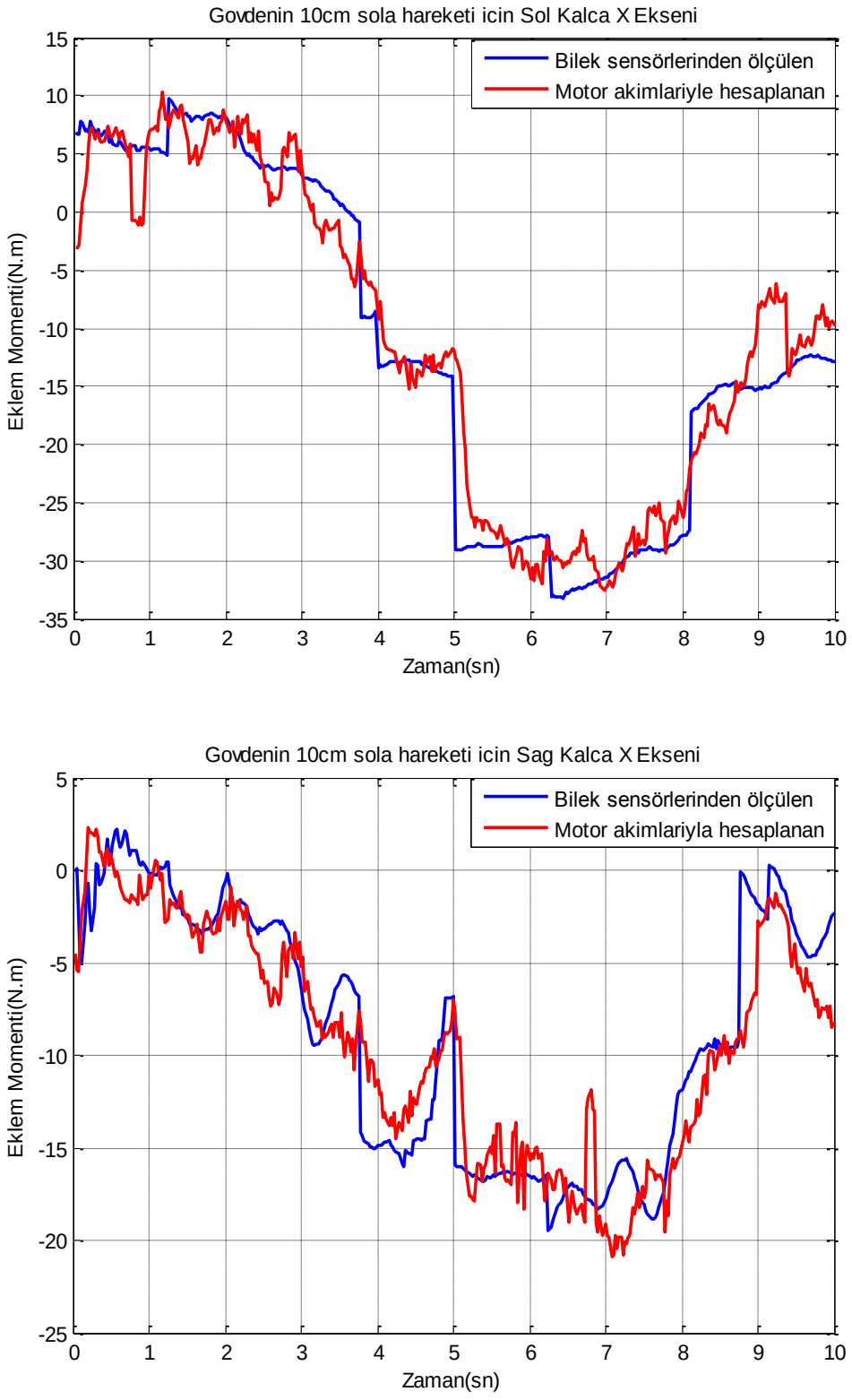
$$\dot{z}_1 = \frac{\dot{x}_0(x_1 - x_0) + \dot{y}_0(y_1 - y_0)}{z_1 - z_0} + \dot{z}_0 \quad \textbf{(4.11)}$$

Ayağa uygulanan tepki kuvvetlerini ve momentlerini ölçen sensörlerden eklem momentlerini elde etmek içinse Bölüm 2'de sunulduğu üzere 3 çizgisel ve 3 açısal hıza bağlı olarak elde edilen (6x6) boyutunda Jakobiyen matrisinden yararlanılır. Bölüm 2'de açıklandığı üzere virtüel iş prensibine dayanan moment hesabından yararlanılarak (4.12) ile hareket sırasında bütün eklemler için moment değerleri bulunabilir.

$$\tau = J^T F \quad \textbf{(4.12)}$$

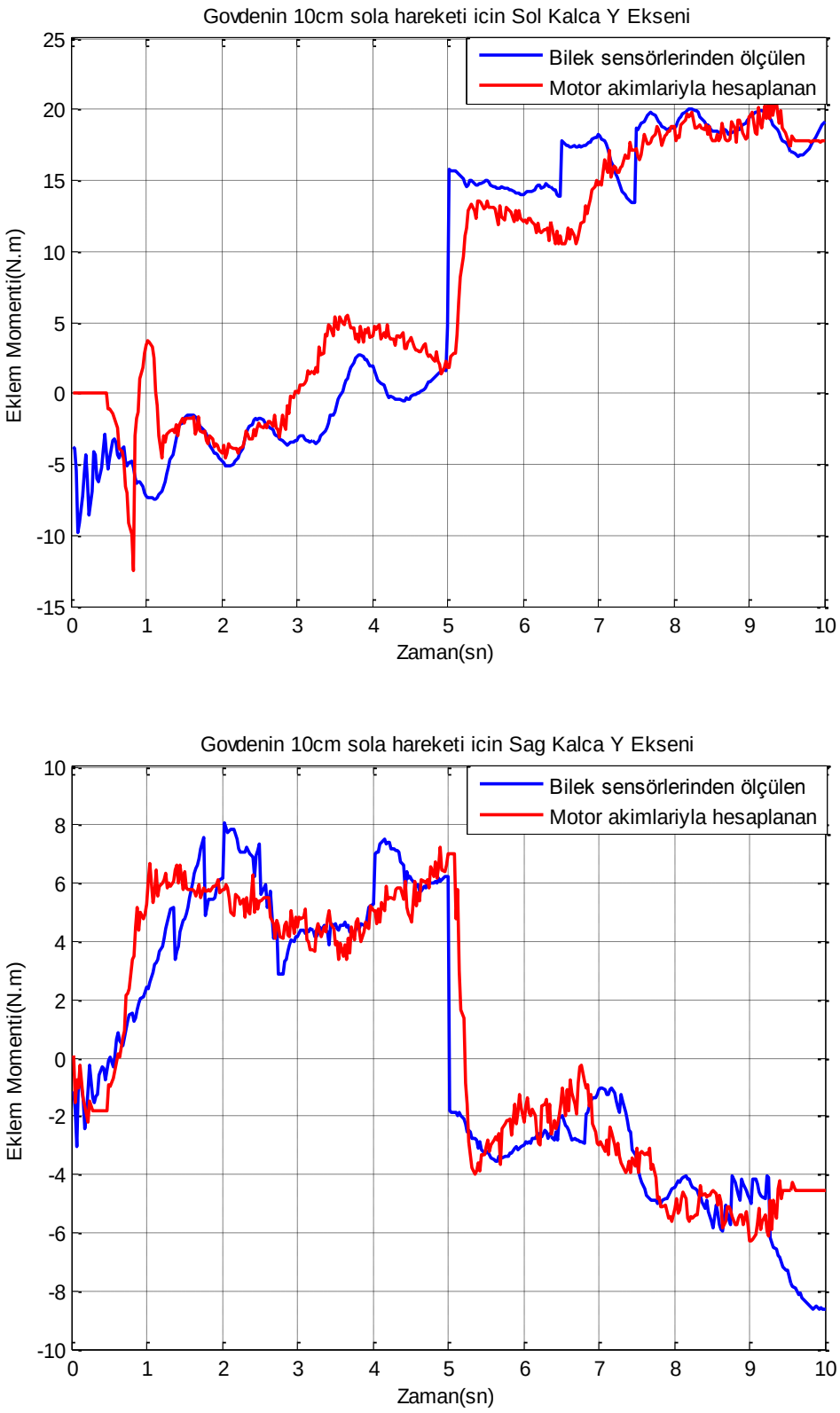
(4.12)'deki parametreler açıklanacak olursa; τ her bir bacağın sahip olduğu (6x1) boyutunda moment vektörü, J^T ise Jakobiyen matrisinin transpozesidir.

(4.12)'deki F 3 eksende ölçülen sırasıyla F_x, F_y, F_z, kuvvet değerleri ve T_x, T_y, T_z moment değerleridir. Bu bölümde gövdenin 10 cm. sola hareketi için bahsedilen her iki yolla da elde edilen eklem moment grafikleri karşılaştırmalı olarak verilmiştir. Gövdenin 10 cm düşey, 10 cm sağ, 13 cm sol ve 13 cm sağ hareketleri için karşılaştırmalı olarak elde edilen eklem moment grafikleri ise EK B'de sunulmuştur.



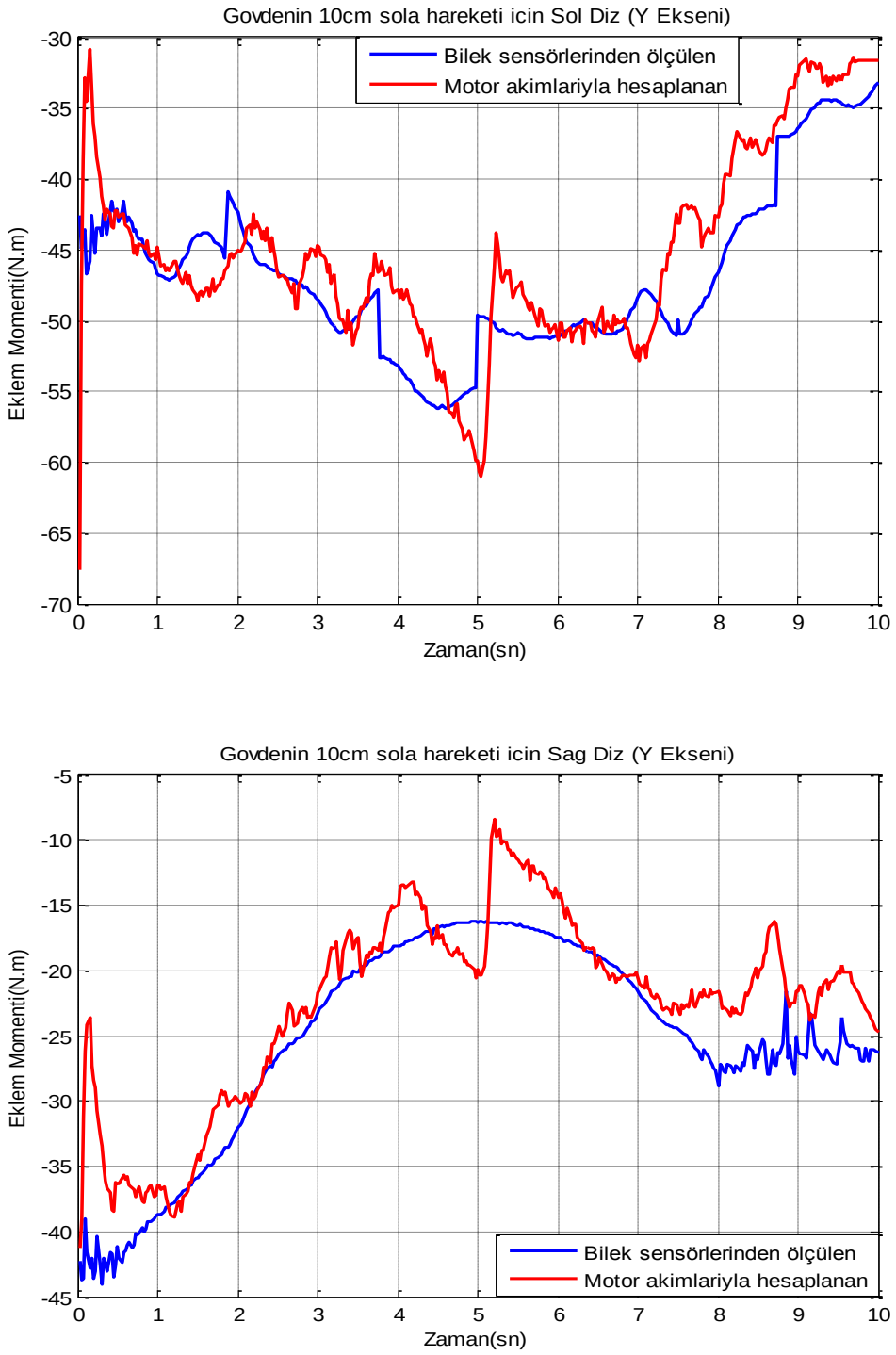
Şekil 4.3 : Gövdenin 10cm sola hareketi için Sol ve Sağ Kalça X ekseni Momentleri.

Şekil 4.3'te görülen mavi renkli grafik kuvvet/moment sensörlerinden ölçülen değerler ve jakobiyen matrisiyle elde edilen eklem momentleri, kırmızı renkli grafik akımlarla hesaplanan eklem momentlerini göstermektedir.



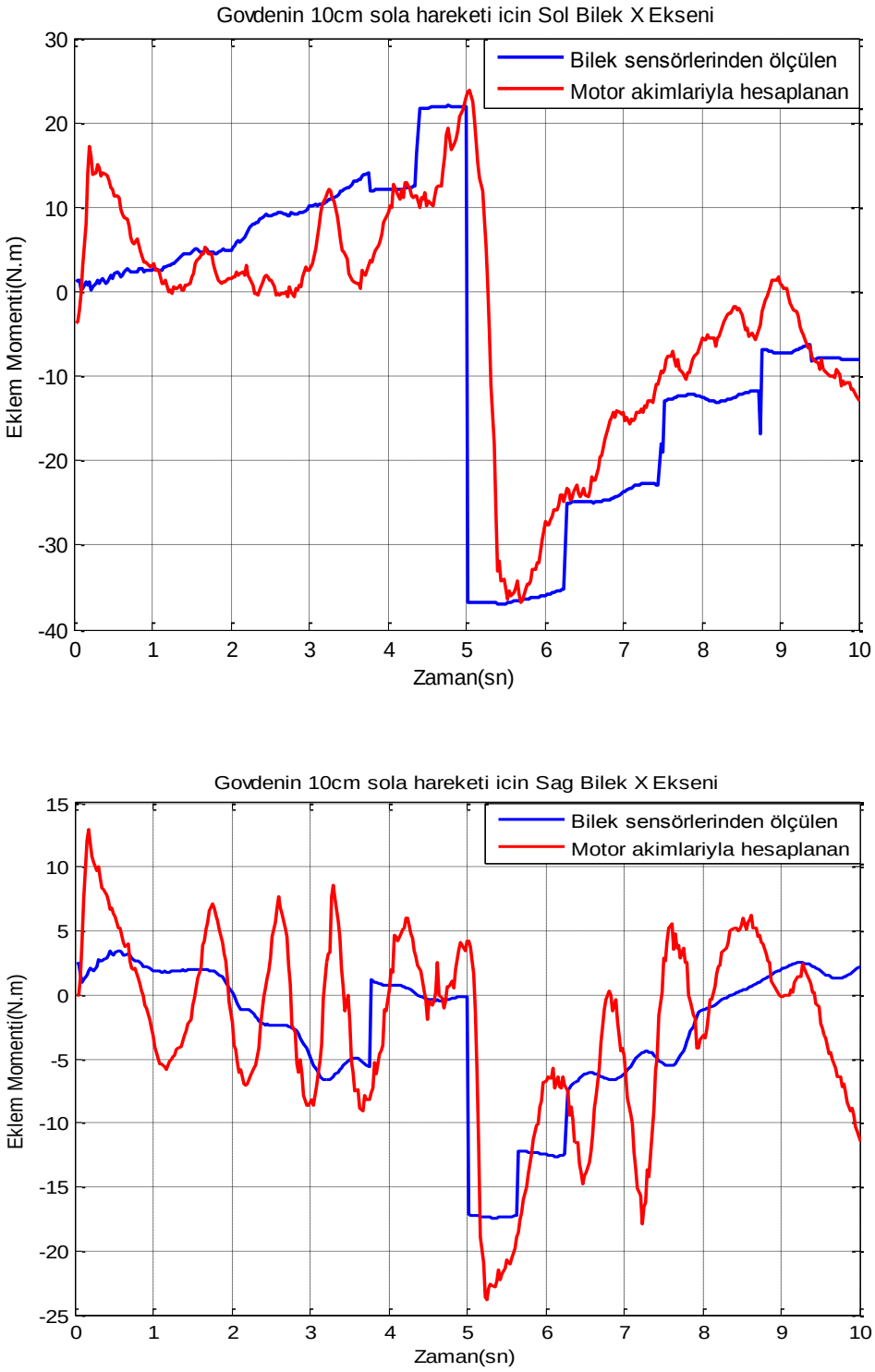
Şekil 4.4 : Gövdenin 10cm sola hareketi için Sol ve Sağ Kalça Y ekseni Momentleri.

Şekil 4.4 kalça Y eksenleri eklem momentlerinin grafiklerini göstermektedir. Kalça eklemlerindeki mekanik yapı diğer eklemlere göre daha basit yapıda olduğu için Şekil 4.3 ve Şekil 4.4'teki gibi davranışlar birbirini küçük farklarla takip etmektedir.



Şekil 4.5 : Gövdenin 10cm sola hareketi için Sol ve Sağ Diz Eklemi Momentleri.

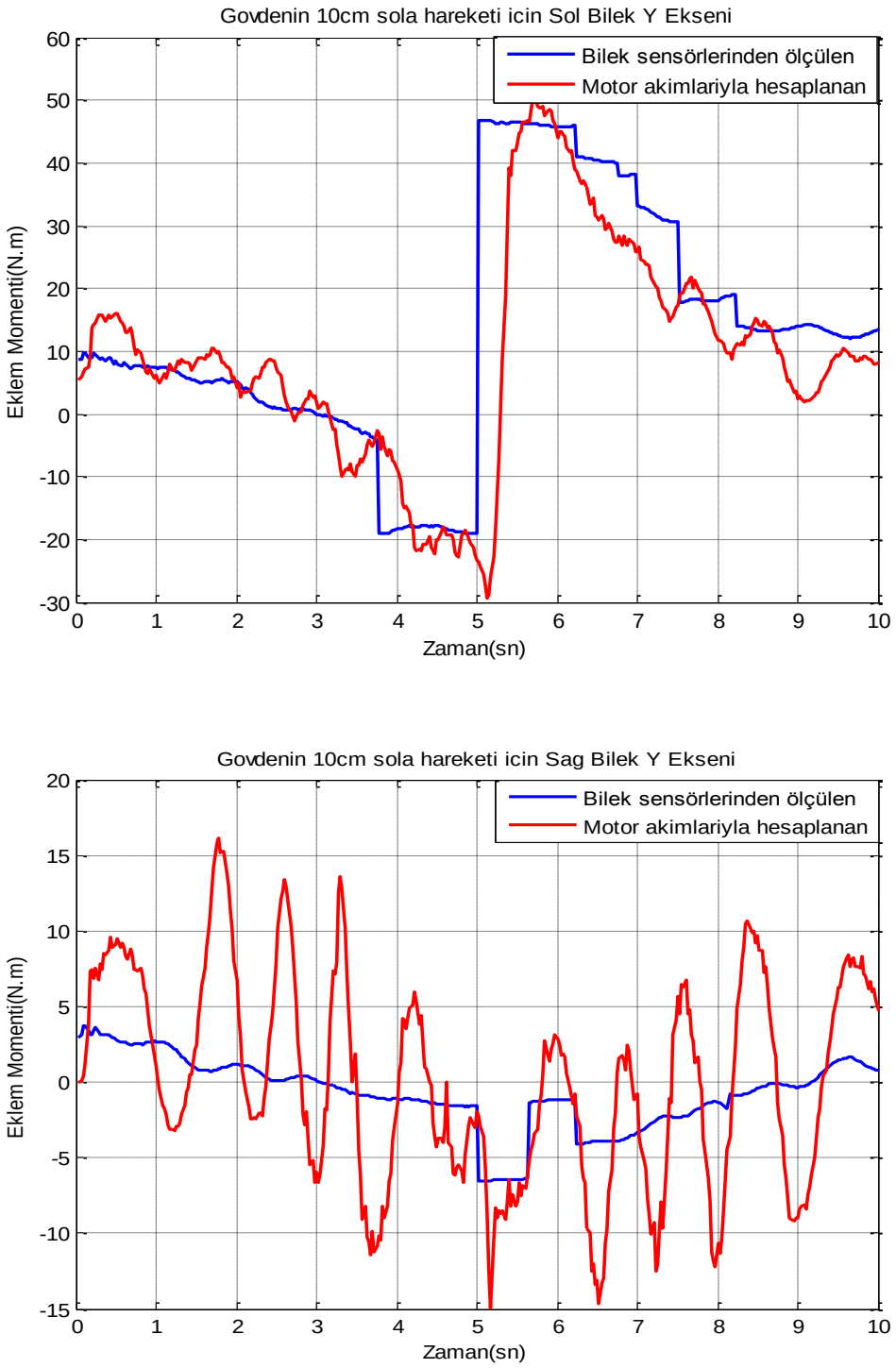
Şekil 4.5 ise sağ ve sol diz eklemlerinin hareket esnasındaki moment değişimlerini göstermektedir. Diz eklemlerinde eğriler birbirini takip etse de farklar grafiklerden de görüldüğü gibi kalça eklemlerinden daha fazladır. Bu durumun başlıca sebebi diz ekleminin aktarma yapısının kalçalardan daha kompleks yapıda olması, ve kurulan kinematik modelin nonlineerlik içermesidir.



Şekil 4.6 : Gövdenin 10cm sola hareketi için Sol ve Sağ Bilek X Ekseni Momentleri.

Kalçalardan bilek eklemine gidildikçe mekanik yapıdaki karmaşıklaşma da artmaktadır. Bölüm 1'de açıklandığı üzere bilek eklemlerinde x ve y'de olmak üzere 2 serbestlik derecesi vardır ve bu serbestlik derecelerindeki hareketleri paralel bir mekanizma gerçekleştirmektedir. Bu durumun sonucu olarak, akımlar yoluyla bilek

eklemlerindeki moment ifadelerini hesaplayabilmek adına mekanik aktarmaları ifade etmek için daha karmaşık ve nonlineer ifadeler içeren kinematik modeller kurulmuştur. Ayrıca bilek eklemlerindeki mekanik pürüzlülükler de sürtünmeyi dolayısıyla iki farklı yoldan çıkarılan eklem momentleri arasındaki farklılıkları arttıran bir diğer önemli faktördür.



Şekil 4.7 : Gövdenin 10cm sola hareketi için Sol ve Sağ Bilek Y Ekseni Momentleri.

Şekil 4.6 ve Şekil 4.7 kurulan modelin getirdiği matematiksel işlem yükü arttıkça hesaplanan ve ölçülen değerler arasındaki farkı açıkça göstermektedir.

5. SONUÇ VE ÖNERİLER

Son birkaç yılda yapılan çalışmalarda iki ayaklı robotun bilgisayar ortamında benzetimi yapılmış, katı modeli oluşturulmuş, mekanik ve elektronik donanımı seçilerek prototipi oluşturulmuştur. Tez kapsamında yapılan ilk çalışmalarda, robotun hareket esnasında daha önce öngörülen örnekleme periyoduna ulaşamadığı ve hareketleri beklenilen sürelerin çok uzağında gerçekleştirdiği görülmüştür. Bu yüzden iletişim protokolünde revizyona gidilerek, CANopen protokolleri arasında geçiş yapılmıştır. Yapılan bir diğer çalışma, gömülü bilgisayar üzerinde C diliyle bir ters kinematik program yazılarak çeşitli hareketler için eklem yörüngelerinin oluşturulmasıdır. Ayrıca, ileri kinematik kullanılarak jakobiyen matrisi elde edilmiş, devreye alınan kuvvet/moment sensörlerinden alınan ölçümler ve jakobiyen matrisi kullanılarak eklem momentleri elde edilerek, motoların çektiği akımlar yoluyla hesaplanan eklem momentleriyle karşılaştırılmıştır. Robota düşey ve yatay hareketler yaptırılarak kurulan kinematik modelin doğruluğu kanıtlanmıştır. Robot yerdeyken düşey ve yatay hareketlerin tümünü başarıyla gerçekleştirmiştir ancak adım atma işlemini kalça bağlantılarındaki esneme nedeniyle gerçekleştirememiştir. Bağlantılardaki sorunlar giderilince yürüme deneylerine tekrar başlanacaktır.

KAYNAKLAR

[1] **Vukobratovic, M., Borovac, B., ve Potkonjak, V.** (2006). ZMP : A review of some basic misunderstandings, International Journal of Humanoid robotics, *Vol. 3, No. 2(2006) pp.153-175.*

[2] **Url-1** <http://world.honda.com/ASIMO>, alındığı tarih 21.09.2012.

[3] **Löffler, K., Gienger, M. ve Pfeiffer, F.**, 2002. The concept of jogging JOHNNIE, IEEE International Conference on Robotics &Automation, Washington, DC, USA, May 2002.

[4] **Url-2** <http://global.kawada.jp/mechatronics/hrp2.html>, alındığı tarih 26.10.2012

[5] **Url-3** <http://en.wikipedia.org/wiki/HUBO>, alındığı tarih 26.10.2012.

[6] **Kajita, S., Kanehiro, F., Kaneko, K.,** 2003. Biped walking pattern generation by using preview control of zero-moment point, IEEE ICRA, Vol. 2, pp. 1620-1626.

[7,12,17] **Tasasız, B.,** 2011. İki ayaklı yürüyen robot prototipin gerçek zamanlı denetimi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü

[8,10] **Craig, J.J.,** 1989. Introduction to robotics, Addison Wesley Longman, New York

[9] **Sciavicco, L., Siciliano, B.,** 1996. Modeling and control of robot manipulators, Springer-Verlag, London

[11] **Shih, C-L., Gruver, W.A., ve Lee, T.T.,** 1992. Inverse kinematics and inverse dynamics for control of a dynamic walking machine, Journal of Robotic Systems, 1992, *pp. 531-555*.

[13-15] **Maxon Motor AG**, 2011. EPOS 2 Communication guide, <http://www.maxonmotor.com> alındığı tarih 14.05.2012

[16] **Url-4** <http:// www.softing.com/home/en/industrial-automation/products/can-bus/more-can-open/process-data-object> alındığı tarih 12.06.2012.

[18] **Gerçek, A.,** 2011, İki ayaklı yürüyen robot tasarımı ve prototip imalatı, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul

[19-21] **ATI Industrial Automation,** 2011. Net F/T Network force/torque sensor system user manual, <http://www.atiia.com/products/ft/ft_literature.aspx>, alındığı tarih 08.12.2012

[22-25] **Bayraktaroglu, Z. Y., Acar, M., Gerçek, A. and Tan, N.,** 2012. Design and Construction of the I.T.U. Biped Robot. Manuscript submitted for publication

EKLER

EK A.1

```
/************************
zeroPos2.c : Robotu bulunduğu konumdan sıfır pozisyonu olarak tanımlanan
noktaya getirmek için gerekli eklem yörüngelerini mutlak enkoder değerlerini
okuyarak üretir.

************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include "can4linux.h"
#define STDDEV "can0"
#define PI 3.14159
#define D2R PI/180.
/* LIMITLERIN TANIMLANMASI */
#define LHZ_max 20 // problem yok!
#define LHZ_set 211.8
#define LHZ_min -20
#define LHZ_dir -1
#define LHX_max 60 //problem yok!
#define LHX_set 71.8
#define LHX_min -60
#define LHX_dir 1
#define LHY_max 90 //kritik! sorunlu gibi islem yap!
#define LHY_set 304.4
#define LHY_min -90
#define LHY_dir 1
#define LK_max 82 // problem yok!
#define LK_set 136.2+2.6
#define LK_min 5
#define LK_dir 1
#define LAY_max -1 // problem yok!
#define LAY_set 194.9+1
#define LAY_min -53
#define LAY_dir -1
#define LAX_max 30 // sorunlu!
#define LAX_set 104 // 345.6 //344.8 zeki hocayla //359.8 labda degistirildi
#define LAX_min -30
#define LAX_dir 1
#define RHZ_max 20 // problem yok!
#define RHZ_set 112.6
#define RHZ_min -20
#define RHZ_dir -1
#define RHX_max 60 //problem yok!
```

```
#define RHX_set 83.6
#define RHX_min -60
#define RHX_dir 1
#define RHY_max 90 //problem yok!
#define RHY_set 218.5
#define RHY_min -90
#define RHY_dir -1
#define RK_max 82 //sorunlu !
#define RK_set 49.9-2
#define RK_min 2
#define RK_dir -1
#define RAY_max -3 // problem yok!
#define RAY_set 269.2-2//268.2
#define RAY_min -52
#define RAY_dir 1
#define RAX_max 30 //? ...sorunlu gibi!
#define RAX_set 28 // 15.4 tekrar guncellendi     //17.2 Zeki hocayla // 28 labda
degistirildi
#define RAX_min -30
#define RAX_dir 1
#define T_FINAL 10
#define RXBUFFERSIZE 100
int fd;
canmsg_t tx;
canmsg_t rx[RXBUFFERSIZE];
double absDegrees[12];
double i_jointDegrees[12];
////////////////////////////////////////////
/* FINAL VALUES OF JOINT DEGREES ARE HERE */
double d_jointDegrees[12]=      { 0, 0, -21.9134, 45.1003, -23.1869 ,0,
                                  0, 0, -21.9134, 45.1003, -23.1869, 0 };
////////////////////////////////////////////
unsigned int OCF[12], COF[12];
clock_t start, end;
unsigned int elaptime;
FILE *f_tra;
void initCan(void);
void sendCan(int ID, char command, int index, int subindex,unsigned long object);
void recieveCan(void);
void myDelayUs(unsigned int start_value,unsigned int delay_value);
void absEncodersRead(void);
int main(void)
{
int i;
double t;
double a1[12], a2[12], b1[12], b2[12], c1[12], c2[12];
if((f_tra=fopen("jointTrajectory.bin","wb")) ==NULL)
{
printf("Cannot open file... \n");
exit(EXIT_FAILURE);
```

```
}
initCan();
absEncodersRead();
printf("LHZ=%.1f\tLHX=%.1f\tLHY=%.1f\tLK=%.1f\tLAY=%.1f\tLAX=%.1f\n",i
_jointDegrees[0],i_jointDegrees[1],i_jointDegrees[2],i_jointDegrees[3],i_jointDegre
es[4],i_jointDegrees[5]);
printf("RHZ=%.1f\tRHX=%.1f\tRHY=%.1f\tRK=%.1f\tRAY=%.1f\tRAX=%.1f\n",i
_jointDegrees[6],i_jointDegrees[7],i_jointDegrees[8],i_jointDegrees[9],i_jointDegre
es[10],i_jointDegrees[11]);
printf("OCF = %d %d %d %d %d %d %d %d %d %d %d %d
\n",OCF[0],OCF[1],OCF[2],OCF[3],OCF[4],OCF[5],OCF[6],OCF[7],OCF[8],OCF[
9],OCF[10],OCF[11]);
printf("COF = %d %d %d %d %d %d %d %d %d %d %d %d
\n",COF[0],COF[1],COF[2],COF[3],COF[4],COF[5],COF[6],COF[7],COF[8],COF[
9],COF[10],COF[11]);
for(i=0;i<12;++i)
{
a1[i]= 2*((d_jointDegrees[i]-i_jointDegrees[i])/(T_FINAL*T_FINAL));
a2[i]= -1*a1[i];
b1[i]= 0;
b2[i]= -2*(a2[i]*T_FINAL);
c1[i]= i_jointDegrees[i];
c2[i]= d_jointDegrees[i]+(a2[i]*(T_FINAL*T_FINAL));
}
for(t=0;t<T_FINAL/2;t=t+0.01)
{
for(i=0;i<12;++i)
fprintf(f_tra,"%.4f\t",D2R*(((a1[i]*t)*t)+c1[i]));
fprintf(f_tra,"\n");
}
for(t=T_FINAL/2;t<T_FINAL;t=t+0.01)
{
for(i=0;i<12;++i)
fprintf(f_tra,"%.4f\t",D2R*(a2[i]*t*t+b2[i]*t+c2[i]));
fprintf(f_tra,"\n");
}
close(fd);
fclose(f_tra);
return 0;
}
void receiveCan()
{
int i,got=0;
while(got<1)
{
got = read(fd,&rx,1);
}
}
```

```
void absEncodersRead(void) /*Sürücü kartlarının dijital giriş ve çıkışları
kullanılarak mutlak kodlayıcılardan eklem açılarının alınabilmesi için
yazılmıştır.*/
{
unsigned long data[12],dataTemp;
int dataBIT,dataBin;
do{                                    //status kontrolü icin ! ! !
printf("Reading Abs.Encoder...\n");
sendCan(1, 'w', 0x2078, 0x02, 0x8000);  // Digital Output Func. Mask
sendCan(2, 'w', 0x2078, 0x02, 0x8000);
sendCan(3, 'w', 0x2078, 0x02, 0x8000);
sendCan(4, 'w', 0x2078, 0x02, 0x8000);
sendCan(5, 'w', 0x2078, 0x02, 0x8000);
sendCan(6, 'w', 0x2078, 0x02, 0x8000);
sendCan(7, 'w', 0x2078, 0x02, 0x8000);
sendCan(8, 'w', 0x2078, 0x02, 0x8000);
sendCan(9, 'w', 0x2078, 0x02, 0x8000);
sendCan(10, 'w', 0x2078, 0x02, 0x8000);
sendCan(11, 'w', 0x2078, 0x02, 0x8000);
sendCan(12, 'w', 0x2078, 0x02, 0x8000);
sendCan(1, 'w', 0x2071, 0x02, 0x8000); // Digital Input Func. Mask
sendCan(2, 'w', 0x2071, 0x02, 0x8000);
sendCan(3, 'w', 0x2071, 0x02, 0x8000);
sendCan(4, 'w', 0x2071, 0x02, 0x8000);
sendCan(5, 'w', 0x2071, 0x02, 0x8000);
sendCan(6, 'w', 0x2071, 0x02, 0x8000);
sendCan(7, 'w', 0x2071, 0x02, 0x8000);
sendCan(8, 'w', 0x2071, 0x02, 0x8000);
sendCan(9, 'w', 0x2071, 0x02, 0x8000);
sendCan(10, 'w', 0x2071, 0x02, 0x8000);
sendCan(11, 'w', 0x2071, 0x02, 0x8000);
sendCan(12, 'w', 0x2071, 0x02, 0x8000);
sendCan(1, 'w', 0x2078, 0x01, 0x8000); // CSn: HIGH
sendCan(2, 'w', 0x2078, 0x01, 0x8000);
sendCan(6, 'w', 0x2078, 0x01, 0x8000);
sendCan(7, 'w', 0x2078, 0x01, 0x8000);
sendCan(8, 'w', 0x2078, 0x01, 0x8000);
sendCan(12, 'w', 0x2078, 0x01, 0x8000);
sendCan(3, 'w', 0x2078, 0x01, 0x8000); // CLK: HIGH
sendCan(4, 'w', 0x2078, 0x01, 0x8000);
sendCan(5, 'w', 0x2078, 0x01, 0x8000);
sendCan(9, 'w', 0x2078, 0x01, 0x8000);
sendCan(10, 'w', 0x2078, 0x01, 0x8000);
sendCan(11, 'w', 0x2078, 0x01, 0x8000);
myDelayUs(start,1000);
for(dataBIT=0;dataBIT<12;++dataBIT)
{
data[dataBIT]=0;
OCF[dataBIT]=0;
COF[dataBIT]=0;
```

```
}
sendCan(1, 'w', 0x2078, 0x01, 0x0000); // CSn: LOW
sendCan(2, 'w', 0x2078, 0x01, 0x0000);
sendCan(6, 'w', 0x2078, 0x01, 0x0000);
sendCan(7, 'w', 0x2078, 0x01, 0x0000);
sendCan(8, 'w', 0x2078, 0x01, 0x0000);
sendCan(12, 'w', 0x2078, 0x01, 0x0000);
start=clock();
myDelayUs(start,1000);
for(dataBIT=18;dataBIT>0;dataBIT--)
{
sendCan(3, 'w', 0x2078, 0x01, 0x0000); // CLK: LOW
sendCan(4, 'w', 0x2078, 0x01, 0x0000);
sendCan(5, 'w', 0x2078, 0x01, 0x0000);
sendCan(9, 'w', 0x2078, 0x01, 0x0000);
sendCan(10, 'w', 0x2078, 0x01, 0x0000);
sendCan(11, 'w', 0x2078, 0x01, 0x0000);
start=clock();
myDelayUs(start,1000);
sendCan(3, 'w', 0x2078, 0x01, 0x8000); // CLK: HIGH
sendCan(4, 'w', 0x2078, 0x01, 0x8000);
sendCan(5, 'w', 0x2078, 0x01, 0x8000);
sendCan(9, 'w', 0x2078, 0x01, 0x8000);
sendCan(10, 'w', 0x2078, 0x01, 0x8000);
sendCan(11, 'w', 0x2078, 0x01, 0x8000);
start=clock();
myDelayUs(start,1000);
sendCan(1, 'r', 0x2071, 0x01, 0x00); // READ DATA: 1
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[0]=dataTemp + data[0];
sendCan(2, 'r', 0x2071, 0x01, 0x00); // READ DATA: 2
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[1]=dataTemp + data[1];
sendCan(3, 'r', 0x2071, 0x01, 0x00); // READ DATA: 3
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[2]=dataTemp + data[2];
sendCan(4, 'r', 0x2071, 0x01, 0x00); // READ DATA: 4
if(rx[0].data[5]==0x80)
```

```
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[3]=dataTemp + data[3];
sendCan(5, 'r', 0x2071, 0x01, 0x00); // READ DATA: 5
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[4]=dataTemp + data[4];
sendCan(6, 'r', 0x2071, 0x01, 0x00); // READ DATA: 6
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[5]=dataTemp + data[5];
sendCan(7, 'r', 0x2071, 0x01, 0x00); // READ DATA: 7
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[6]=dataTemp + data[6];
sendCan(8, 'r', 0x2071, 0x01, 0x00); // READ DATA: 8
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[7]=dataTemp + data[7];
sendCan(9, 'r', 0x2071, 0x01, 0x00); // READ DATA: 9
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[8]=dataTemp + data[8];
sendCan(10, 'r', 0x2071, 0x01, 0x00); // READ DATA: 10
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[9]=dataTemp + data[9];
sendCan(11, 'r', 0x2071, 0x01, 0x00); // READ DATA: 11
if(rx[0].data[5]==0x80)
dataBin=1;
```

```
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[10]=dataTemp + data[10];
sendCan(12, 'r', 0x2071, 0x01, 0x00); // READ DATA: 12
if(rx[0].data[5]==0x80)
dataBin=1;
else
dataBin=0;
dataTemp=dataBin<<(dataBIT-1);
data[11]=dataTemp + data[11];
}
for(dataBIT=0;dataBIT<12;++dataBIT)
{
absDegrees[dataBIT]=(360./4096.)*(data[dataBIT]>>6);
OCF[dataBIT]=(data[dataBIT]>>5)%2;
COF[dataBIT]=(data[dataBIT]>>4)%2;
}
}while( OCF[0]==0 || OCF[1]==0 || OCF[2]==0 || OCF[3]==0 || OCF[4]==0 ||
OCF[5]==0 ||
OCF[6]==0 || OCF[7]==0 || OCF[8]==0 || OCF[9]==0 || OCF[10]==0 || OCF[11]==0
||
COF[0]==1 || COF[1]==1 || COF[2]==1 || COF[3]==1 || COF[4]==1 || COF[5]==1 ||
COF[6]==1 || COF[7]==1 || COF[8]==1 || COF[9]==1 || COF[10]==1 || COF[11]==1
);
/* LHZ */
i_jointDegrees[0] = LHZ_dir*(absDegrees[0]-LHZ_set)*0.6;
/* LHX */
i_jointDegrees[1] = LHX_dir*(absDegrees[1]-LHX_set);
/* LHY */
if(absDegrees[2]<100)
absDegrees[2]= 360+absDegrees[2];
i_jointDegrees[2] = LHY_dir*(absDegrees[2]-LHY_set);
/* LK */
/* diz en asagi konumdayken sifir kabul ediliyor */
i_jointDegrees[3] = LK_dir*(absDegrees[3]-LK_set);
/* LAY */
/* ayak burnu en asagidayken sifir kabul ediliyor */
i_jointDegrees[4] = LAY_dir*(absDegrees[4]-LAY_set);
/* LAX */
/* if(absDegrees[5]<100)
absDegrees[5]= 360+absDegrees[5];*/
i_jointDegrees[5] = LAX_dir*(absDegrees[5]-LAX_set);
/* RHZ */
i_jointDegrees[6] = RHZ_dir*(absDegrees[6]-RHZ_set)*0.6;
/* RHX */
i_jointDegrees[7] = RHX_dir*(absDegrees[7]-RHX_set);
/* RHY */
i_jointDegrees[8] = RHY_dir*(absDegrees[8]-RHY_set);
/* RK */
```

```
/* diz en asagi konumdayken sifir kabul ediliyor */
if(absDegrees[9]>200)
absDegrees[9]= absDegrees[9]-360;
i_jointDegrees[9] = RK_dir*(absDegrees[9]-RK_set);
/* RAY */
/* ayak burnu en asagidayken sifir kabul ediliyor */
i_jointDegrees[10] = RAY_dir*(absDegrees[10]-RAY_set);
/* RAX */
if(absDegrees[11]>200)
absDegrees[11]= absDegrees[11]-360;
i_jointDegrees[11] = RAX_dir*(absDegrees[11]-RAX_set);
}
void initCan(void) /* Bilgisayarın Can modülünün çalıştırılabilmesi için gereken
ayarlamalar için yazılmıştır.*/
{
char device[40];
sprintf(device, "/dev/%s", STDDEV);
if(( fd = open(device, O_RDWR )) < 0 )
{
fprintf(stderr, "Error opening CAN device %s\n", device);
exit(1);
}
}
void sendCan(int ID, char command, int index, int subindex,unsigned long object) /*
Can mesajlarını Sürücü Kartlarının istedigi formatta olusturmak için
yazılmıştır.*/
{
int index_msb,index_lsb;
int value[3];
int sent,i;
tx.id=0x600+ID;
if(command == 'w')
tx.data[0]=0x22;
if(command == 'r')
tx.data[0]=0x40;
index_msb=index/256;
index_lsb=index-index_msb*256;
tx.data[1]=index_lsb;
tx.data[2]=index_msb;
tx.data[3]=subindex;
value[3]=object/16777216;
value[2]=(object-value[3]*16777216)/65536;
value[1]=(object-value[3]*16777216-value[2]*65536)/256;
value[0]=object-value[3]*16777216-value[2]*65536-value[1]*256;
tx.data[4]=value[0];
tx.data[5]=value[1];
tx.data[6]=value[2];
tx.data[7]=value[3];
tx.flags=0;
tx.length=8;
```

```
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
receiveCan();
}
void myDelayUs(unsigned int start_value,unsigned int time_value) /* Bilgisayarın
gerçek zamanlı saatinden aldıgı hassas zaman degeri ile mikro saniye cinsinden
gecikmeler oluşturabilmek için yazılmıştır.*/
{
unsigned int elaptime;
do{
end=clock();
elaptime=end-start_value;
}while(elaptime<time_value);
}
```

EK A.2

```
/*****************************
j2mTrans.c : Bu program,robotun eklem referanslarını motor
referanslarına çevirir. Eklem referanslarını "jointAngles.bin"
dosyasından okur. Olusturdugu motor referanslarını"motorAngles.bin"
dosyasına yazar.*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
////////////////////////////////////////////////////////
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include "can4linux.h"
#define STDDEV "can0"
#define RXBUFFERSIZE 100
int fd;
canmsg_t tx;
canmsg_t rx[RXBUFFERSIZE];
long incDegrees[12];
long shiftValues[12];
////////////////////////////////////////////////////////
double th_hzl, th_hxl, th_hyl, th_kl, th_ayl, th_axl, th_hzr, th_hxr, th_hyr, th_kr,
th_ayr, th_axr;
signed long int th_hzml, th_hxml, th_hyml, th_kml, th_amll, th_amrl, th_hzmr,
th_hxmr, th_hymr, th_kmr, th_amlr, th_amrr;
////////////////////////////////////////////////////////
void initCan(void);
void sendCanPDOinc(int ID);
void sendCanPDONMT(int ID);
void sendCanPDOsync(void);
////////////////////////////////////////////////////////
int main(int argc, char *argv[])
{
const double pi = 3.14159;
const double r2c = 1000/pi;
//Hip Parameters
const int n_hzl = 120;
const int n_hxl = 160;
const int n_hyl = 160;
const int n_hzr = 120;
const int n_hxr = 160;
const int n_hyr = 160;
////////////////
//Knee Parameters
const double Sv_k = 0.002; //knee screw pitch
const double a_k = 0.033;
```

```
const double b_k = 0.035;
const double l_k = 0.337;
////////////////
//Ankle Parameters
const double l_a = 0.17226;
const double Sv_a = 0.001;
const double a_all = -0.036666;
const double b_all = 0.031;
const double c_all = -0.006067;
const double a_arl = -0.036666;
const double b_arl = -0.031;
const double c_arl = -0.006067;
const double x1_all = -0.039;
const double y1_all = 0.0275;
const double x1_arl = -0.039;
const double y1_arl = -0.0275;
const double a_alr = -0.036666;
const double b_alr = 0.031;
const double c_alr = -0.006067;
const double a_arr = -0.036666;
const double b_arr = -0.031;
const double c_arr = -0.006067;
const double x1_alr = -0.039;
const double y1_alr = 0.0275;
const double x1_arr = -0.039;
const double y1_arr = -0.0275;
////////////////////////////////
double x0_all;
double y0_all;
double z0_all;
double z1_all;
double delta_all;
double z1i_all;
double x0_arl;
double y0_arl;
double z0_arl;
double z1_arl;
double delta_arl;
double z1i_arl;
double x0_alr;
double y0_alr;
double z0_alr;
double z1_alr;
double delta_alr;
double z1i_alr;
double x0_arr;
double y0_arr;
double z0_arr;
double z1_arr;
double delta_arr;
```

```
double z1i_arr;
FILE *fj;
FILE *fm;
int count = 0;
int j;
//////////////////////////////////////////
initCan();
for(j=1;j<13;++j)
sendCanPDONMT(j);
for(j=1;j<13;++j)
sendCanPDOinc(j);
printf("LHZ=%ld\tLHX=%ld\tLHY=%ld\tLK=%ld\tLAL=%ld\tLAR=%ld\n",incDe
grees[0],incDegrees[1],incDegrees[2],incDegrees[3],incDegrees[4],incDegrees[5]);
printf("RHZ=%ld\tRHX=%ld\tRHY=%ld\tRK=%ld\tRAL=%ld\tRAR=%ld\n",incD
egrees[6],incDegrees[7],incDegrees[8],incDegrees[9],incDegrees[10],incDegrees[11
]);
//////////////////////////////////////////
if ((fj = fopen("jointTrajectory.bin", "r")) == NULL) {
printf("Cannot open jointAngles.bin file!\n");
exit(EXIT_FAILURE);
}
if ((fm = fopen("motorTrajectory.bin", "w")) == NULL) {
printf("Cannot open motorAngles.bin file!\n");
exit(EXIT_FAILURE);
}
while (fscanf(fj, "%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf", &th_hzl, &th_hxl,
&th_hyl, &th_kl, &th_ayl, &th_axl, &th_hzr, &th_hxr, &th_hyr, &th_kr, &th_ayr,
&th_axr) != EOF) {
//Hip Motors Analysis//
th_hzml = (r2c) * th_hzl * n_hzl;
th_hxml = (-r2c) * th_hxl * n_hxl;
th_hyml = (-r2c) * th_hyl * n_hyl;
th_hzmr = (r2c) * th_hzr * n_hzr;
th_hxmr = (-r2c) * th_hxr * n_hxr;
th_hymr = (r2c) * th_hyr * n_hyr;
//////////////////////////////////////
//Knee Motors Analysis//
th_kml = (r2c) * 2 * pi / Sv_k * (sqrt(powf(b_k,2) + powf((a_k + l_k),2)) -
sqrt(powf(a_k,2) + powf(b_k,2) + powf(l_k,2) - 2*sqrt(powf(a_k,2) + powf(b_k,2))
* l_k * cos(pi-atan(b_k/a_k) - th_kl)));
th_kmr = (r2c) * 2 * pi / Sv_k * (sqrt(powf(b_k,2) + powf((a_k + l_k),2)) -
sqrt(powf(a_k,2) + powf(b_k,2) + powf(l_k,2) - 2*sqrt(powf(a_k,2) + powf(b_k,2))
* l_k * cos(pi-atan(b_k/a_k) - th_kr)));
//////////////////////////////////////
//Ankle Motors Analysis//
x0_all = a_all*cos(th_ayl) + b_all*sin(th_ayl)*sin(th_axl) +
c_all*sin(th_ayl)*cos(th_axl);
y0_all = b_all*cos(th_axl) - c_all*sin(th_axl);
z0_all = -a_all*sin(th_ayl) + b_all*cos(th_ayl)*sin(th_axl) +
c_all*cos(th_ayl)*cos(th_axl);
```

```
z1_all = sqrt(powf(l_a,2) - powf(x1_all-x0_all,2) - powf(y1_all-y0_all,2)) + z0_all;
if (count == 0)
z1i_all = z1_all;
delta_all = z1_all - z1i_all;
th_amll = (r2c) * 2*pi/Sv_a*delta_all;
x0_arl = a_arl*cos(th_ayl) + b_arl*sin(th_ayl)*sin(th_axl) +
c_arl*sin(th_ayl)*cos(th_axl);
y0_arl = b_arl*cos(th_axl) - c_arl*sin(th_axl);
z0_arl = -a_arl*sin(th_ayl) + b_arl*cos(th_ayl)*sin(th_axl) +
c_arl*cos(th_ayl)*cos(th_axl);
z1_arl = sqrt(powf(l_a,2) - powf(x1_arl-x0_arl,2) - powf(y1_arl-y0_arl,2)) + z0_arl;
if (count == 0)
z1i_arl = z1_arl;
delta_arl = z1_arl - z1i_arl;
th_amrl = (r2c) * 2*pi/Sv_a*delta_arl;
x0_alr = a_alr*cos(th_ayr) + b_alr*sin(th_ayr)*sin(th_axr) +
c_alr*sin(th_ayr)*cos(th_axr);
y0_alr = b_alr*cos(th_axr) - c_alr*sin(th_axr);
z0_alr = -a_alr*sin(th_ayr) + b_alr*cos(th_ayr)*sin(th_axr) +
c_alr*cos(th_ayr)*cos(th_axr);
z1_alr = sqrt(powf(l_a,2) - powf(x1_alr-x0_alr,2) - powf(y1_alr-y0_alr,2)) + z0_alr;
if (count == 0)
z1i_alr = z1_alr;
delta_alr = z1_alr - z1i_alr;
th_amlr = (r2c) * 2*pi/Sv_a*delta_alr;
x0_arr = a_arr*cos(th_ayr) + b_arr*sin(th_ayr)*sin(th_axr) +
c_arr*sin(th_ayr)*cos(th_axr);
y0_arr = b_arr*cos(th_axr) - c_arr*sin(th_axr);
z0_arr = -a_arr*sin(th_ayr) + b_arr*cos(th_ayr)*sin(th_axr) +
c_arr*cos(th_ayr)*cos(th_axr);
z1_arr = sqrt(powf(l_a,2) - powf(x1_arr-x0_arr,2) - powf(y1_arr-y0_arr,2)) + z0_arr;
if (count == 0)
z1i_arr = z1_arr;
delta_arr = z1_arr - z1i_arr;
th_amrr = (r2c) * 2*pi/Sv_a*delta_arr;
////////////////////////////////////////////////////////
if(count==0)
{
shiftValues[0]= th_hzml;
shiftValues[1]= th_hxml;
shiftValues[2]= th_hyml;
shiftValues[3]= th_kml;
shiftValues[4]= th_amll;
shiftValues[5]= th_amrl;
shiftValues[6]= th_hzmr;
shiftValues[7]= th_hxmr;
shiftValues[8]= th_hymr;
shiftValues[9]= th_kmr;
shiftValues[10]= th_amlr;
shiftValues[11]= th_amrr;
```

```
}
th_hzml= th_hzml + incDegrees[0] - shiftValues[0];
th_hxml= th_hxml + incDegrees[1] - shiftValues[1];
th_hyml= th_hyml + incDegrees[2] - shiftValues[2];
th_kml=   th_kml + incDegrees[3] - shiftValues[3];
th_amll= th_amll + incDegrees[4] - shiftValues[4];
th_amrl= th_amrl + incDegrees[5] - shiftValues[5];
th_hzmr= th_hzmr + incDegrees[6] - shiftValues[6];
th_hxmr= th_hxmr + incDegrees[7] - shiftValues[7];
th_hymr= th_hymr + incDegrees[8] - shiftValues[8];
th_kmr=   th_kmr + incDegrees[9] - shiftValues[9];
th_amlr= th_amlr + incDegrees[10] - shiftValues[10];
th_amrr= th_amrr + incDegrees[11] - shiftValues[11];
////////////////////////////////////////////////////////
fprintf(fm, "%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\n",
th_hzml, th_hxml, th_hyml, th_kml, th_amll, th_amrl, th_hzmr, th_hxmr, th_hymr,
th_kmr, th_amlr, th_amrr);
count++;
}
printf("Row Number = %d\n",count);
fclose(fm);
close(fd);
return 0;
}
void sendCanPDOinc(int ID) /* NMT State Operational durumuna geçildikten
sonra PDO protokolü Asenkron Remote Transmission Request gönderilerek
artımlı enkoderlerin değerleri okunur.*/
{
double motorDegree;
unsigned long usigndata;
long motorCount;
int value[3];
int got,sent,i;
tx.id=0x380+ID;
tx.flags=MSG_RTR;
tx.length=4;
do{
sent = write(fd,&tx,1);
}while(sent<1);
got=0;
while(got<1) {
got=read(fd,&rx,1);
if(got)
{
got = 0;
if(rx[0].id == 0x380+ID)
{
if(rx[0].length == 4)
{
got = 1;
```

```
}
}
}
}
usigndata=65536*rx[0].data[2]+256*rx[0].data[1]+rx[0].data[0];
if(rx[0].data[2]>128)
motorCount =usigndata-16777216;
else
motorCount =usigndata;
incDegrees[ID-1]=motorCount;
}
void sendCanPDONMT(int ID) /* PDO protokolüyle iletişim için gerekli olan
NMT State Operational moduna geçilir.*/
{
int got,sent,i;
tx.id=0;
tx.data[0]=0x01;
tx.data[1]=ID;
tx.flags=0;
tx.length=2;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
}
void initCan(void) /*Bilgisayarın Can modülünün çalıştırılabilmesi için gereken
ayarlamalar için yazılmıştır.*/
{
char device[40];
sprintf(device, "/dev/%s", STDDEV);
if(( fd = open(device, O_RDWR )) < 0 )
{
fprintf(stderr, "Error opening CAN device %s\n", device);
exit(1);
}
}
```

EK A.3

```
/************************
statikdeneme.c :  Motor yörüngelerini sürücü kartlarına uygulayarak robotun
istenilen hareketi gerçekleştirmesini sağlar.
************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
#include <math.h>
#include "can4linux.h"
#define STDDEV "can0"
#define STDDEV1 "can1"
#define RXBUFFERSIZE 100
#define MAX_SIZE 10000
int fd,fd1,T_F;
canmsg_t tx;
canmsg_t rx[RXBUFFERSIZE];
long start, end;
unsigned int elaptime;
struct timeval timer,timer0,timer1;
int currents[MAX_SIZE][12];
double f_fsol[3],f_tsol[3],f_fsag[3],f_tsag[3];
long maxVel[12];
long motorTra[MAX_SIZE][12];
void initCan(void);
void initCan1(void);
void readCurrent(int,int);
void absEncodersRead(void);
void sendCanPDOposition(int ID,unsigned long object);
void sendCanPDOcntrlword(int ID);
void sendCanPDOcntrlword2(int ID);
void sendCanPDOmodes(int ID);
void sendCanPDOcurrent(int ID);
void sendCanFT1(int ID);
void sendCanFT2(int ID);
void motorDon(int ID,long count);
void motorInit(int ID);
void readTrajectory(void);
void sendCanPDONMT(int ID);
void myDelayUs(unsigned int start_value,unsigned int delay_value);
void myDelayUs2(unsigned int start_value,unsigned int delay_value);
void sendCanPDOsync(void);
void writeCurrent(void);
void writeFT(void);
```

```
int main(void)
{
int j,m,k,a;
char key;
readTrajectory();
printf("op. time=%d s\n",T_F/100);
printf("max. motor speeds (rpm)\n");
printf("%8s%8s%8s%8s%8s%8s\n","HipZ","HipY","HipX","Knee","AnkleL","Ankl
eR");
for(j=0;j<6;++j)
{
printf("%8ld", maxVel[j]);
}
printf("\n");
for(j=6;j<12;++j)
{
printf("%8ld", maxVel[j]);
}
printf("\n");
printf("Press 's' and enter to start motion...\n");
do{
key=getchar();
}while(key!='s' && key!='S');
printf("%8s%8s%8s%8s%8s%8s%8s","
","LHZ","LHY","LHX","LK","LAX","LAY");
printf("%8s%8s%8s%8s%8s%8s\n","RHZ","RHY","RHX","RK","RAX","RAY");
initCan();
for(j=1;j<13;++j)
motorInit(j);
close(fd);
initCan();
initCan1();
for(j=0;j<T_F;++j)
{
gettimeofday(&timer,NULL);
start=timer.tv_usec;
sendCanPDOsync();
for(m=1;m<13;++m)
motorDon(m,motorTra[j][m-1]);
sendCanFT1(32);
sendCanFT2(432);
for(k=1;k<13;++k)
readCurrent(j,k);
if(j%50==0)
{
printf("%4.1f s =",j/100.);
for(k=0;k<12;++k)
{
printf("%8d",currents[j][k]);
}
```

```
printf("\n");
}
myDelayUs(start,10000);
}
writeCurrent();
close(fd);
close(fd1);
return 0;
}
void motorDon(int ID,long count) /* Motorların referans değerlerini PDO protokolüyle sürücü kartlarının nesnelerine yazdırır.*/
{
sendCanPDOposition(ID,count);
}
void motorInit(int ID) /* Motorlar harekete geçmeden önce gerekli düzenlemeler yapılır. EPOS2 Application Notes Collection Device Programming bölümünde ilgili yönlendirme bulunmaktadır.*/
{
sendCanPDOmodes(ID); //Modes of Operation:Position Mode
gettimeofday(&timer,NULL);
myDelayUs(timer.tv_usec,10000);
sendCanPDOcntrlword(ID); // Controlword:Shutdown
gettimeofday(&timer,NULL);
myDelayUs(timer.tv_usec,10000);
sendCanPDOcntrlword2(ID); // Controlword:Switch-on
}
void initCan(void)/* CAN başlatılır. */
{
char device[40];
sprintf(device, "/dev/%s", STDDEV);
if(( fd = open(device, O_RDWR )) < 0 )
{
fprintf(stderr, "Error opening CAN device %s\n", device);
exit(1);
}
}
void initCan1(void) /* Kuvvet-Moment sensörlerinin bağlı olduğu CAN yolu başlatılır. */
{
char device[40];
sprintf(device, "/dev/%s", STDDEV1);
if(( fd1 = open(device, O_RDWR )) < 0 )
{
fprintf(stderr, "Error opening CAN device %s\n",device);
exit(1);
}
}
void sendCanFT1(int ID) /* Sol ayak kuvvet/moment değerleri okunur ve bir dosyaya kaydedilir.*/
{
```

```
FILE *ftsol;
if( (ftsol=fopen("ftsolayak13cmyatay.bin","a")) == NULL)
{
printf("Can not open file\n");
exit(EXIT_FAILURE);
}
int k;
int got,sent,i=0;
unsigned short F[3],T[3];
short f[3],t[3];
double factor[2]={32.77,1310.62};
double f_fsol[3],f_tsol[3];
tx.id=ID;
tx.data[0]=2;
tx.flags=0;
tx.length=1;
do{
sent = write(fd1,&tx,1); //mesajin gonderilmesi
}while(sent<1);
do{
got= read(fd1,&rx,1);
}while(got<1);
F[0]= 256*rx[0].data[1] + rx[0].data[0];
F[1]= 256*rx[0].data[5] + rx[0].data[4];
T[0]= 256*rx[0].data[3] + rx[0].data[2];
T[1]= 256*rx[0].data[7] + rx[0].data[6];
do{
got= read(fd1,&rx,1);
}while(got<1);
F[2]= 256*rx[0].data[1] + rx[0].data[0];
T[2]= 256*rx[0].data[3] + rx[0].data[2];
for(i=0;i<3;++i)
{
t[i]=T[i];
f[i]=F[i];
f_fsol[i]=f[i]/factor[0];
f_tsol[i]=t[i]/factor[1];
}
fprintf(ftsol,"%lf %lf %lf %lf %lf
%lf",f_fsol[0],f_fsol[1],f_fsol[2],f_tsol[0],f_tsol[1],f_tsol[2]);
fprintf(ftsol,"\n");
fclose(ftsol);
}
void sendCanFT2(int ID)/* Sağ ayak kuvvet/moment değerleri okunur ve bir
dosyaya kaydedilir.*/
{
FILE *ftsag;
if( (ftsag=fopen("ftsagayak13cmyatay.bin","a")) == NULL)
{
printf("Can not open file\n");
```

```
exit(EXIT_FAILURE);
}
int got,sent,i=0;
unsigned short F[3],T[3];
short f[3],t[3];
double factor[2]={32.77,1310.62};
double f_fsag[3],f_tsag[3];
tx.id=ID;
tx.data[0]=2;
tx.flags=0;
tx.length=1;
do{
sent = write(fd1,&tx,1); //mesajin gonderilmesi
}while(sent<1);
do{
got= read(fd1,&rx,1);
}while(got<1);
F[0]= 256*rx[0].data[1] + rx[0].data[0];
F[1]= 256*rx[0].data[5] + rx[0].data[4];
T[0]= 256*rx[0].data[3] + rx[0].data[2];
T[1]= 256*rx[0].data[7] + rx[0].data[6];
do{
got= read(fd1,&rx,1);
}while(got<1);
F[2]= 256*rx[0].data[1] + rx[0].data[0];
T[2]= 256*rx[0].data[3] + rx[0].data[2];
for(i=0;i<3;++i)
{
t[i]=T[i];
f[i]=F[i];
f_fsag[i]=f[i]/factor[0];
f_tsag[i]=t[i]/factor[1];
}
fprintf(ftsag,"%lf %lf %lf %lf %lf
%lf",f_fsag[0],f_fsag[1],f_fsag[2],f_tsag[0],f_tsag[1],f_tsag[2]);
fprintf(ftsag,"\n");
fclose(ftsag);
}
void sendCanPDOposition(int ID,unsigned long object) /* İlgili pozisyon değerleri
sürücünün okuması için hexadecimal formata çevrilir*/
{

int value[3];
int got,sent,i;
tx.id=0x400+ID;
value[3]=object/16777216;
value[2]=(object-value[3]*16777216)/65536;
value[1]=(object-value[3]*16777216-value[2]*65536)/256;
value[0]=object-value[3]*16777216-value[2]*65536-value[1]*256;
tx.data[0] = value[0];
```

```
tx.data[1] = value[1];
tx.data[2] = value[2];
tx.data[3] = value[3];
tx.flags=0;
tx.length=4;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
}
void sendCanPDOcntrlword(int ID)  /* Motorların hareketi için gerekli
düzenlemeler yapılır. Can mesajı sürücü için istenilen formata çevrilir.*/
{
int value[3];
int got,sent,i;
tx.id=0x200+ID;
tx.data[0]=0x06;
tx.data[1]=0x00;
tx.flags=0;
tx.length=2;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
}
void sendCanPDOcntrlword2(int ID) /* Motorların hareketi için gerekli
düzenlemeler yapılır. Can mesajı sürücü için istenilen formata çevrilir.*/
{
int value[3];
int got,sent,i;
tx.id=0x200+ID;
tx.data[0]=0x0F;
tx.data[1]=0x00;
tx.flags=0;
tx.length=2;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
}
void sendCanPDOmodes(int ID) /* Motorların hareketi için gerekli düzenlemeler
yapılır. Can mesajı sürücü için istenilen formata çevrilir.*/
{
int got,sent,i;
tx.id=0x300+ID;
tx.data[0]=0xFF;
tx.flags=0;
tx.length=1;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
}
```

```
void sendCanPDOcurrent(int ID) /* Motorların çektiği akımlar sürücü
kartlarının nesnelerinden okunur.*/
{
int value[3];
int got,sent,i;
do{
got = read(fd,&rx,1); //mesajin gonderilmesi
}while(got<1);
}
void readTrajectory() /*Harekete baslamadan önce motor referanslarını
motorTrajectory.bin dosyasından alır ve referansları motorTra isimli iki
boyutlu dizinin içine atar. Böylelikle hareket sırasında dosya okuma islemi ile
zaman kaybedilmez.*/
{
FILE *fl;
int count=0,i,Row[12];
long temp[12],exPos[12];
long velocity;
for(i=0;i<12;i++)
{
exPos[i]=0;
Row[i]=0;
exPos[i]=0;
}
if( (fl=fopen("motorTrajectory.bin","rb")) == NULL)
{
printf("Can not open file\n");
exit(EXIT_FAILURE);
}
while (fscanf(fl, "%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld",&temp[0]
,&temp[1], &temp[2], &temp[3], &temp[4], &temp[5],
&temp[6], &temp[7], &temp[8], &temp[9], &temp[10], &temp[11]) != EOF)
{
for(i=0;i<12;++i)
motorTra[count][i]=temp[i];
if(count==0)
for(i=0;i<12;++i)
exPos[i]=motorTra[count][i];
for(i=0;i<12;++i)
{
velocity= (3*(motorTra[count][i]-exPos[i])); //rpm cinsinden 2000 e ve 0.01 e bol ve
60la carp
if(abs(maxVel[i])<abs(velocity))
{
maxVel[i]=velocity;
Row[i]=count;
}
exPos[i]=motorTra[count][i];
}
count++;
```

```
if(count>MAX_SIZE)
{
printf("Size of trajectory arrays are too small !!!\n");
exit(-1);
}
}
T_F=count;
for(i=0;i<12;++i)
printf("Row=%d\t",Row[i]);
printf("\n");
fclose(fl);
}
void myDelayUs(unsigned int start_value,unsigned int time_value) /*Motor referanslarının sürücü kartlarına belirlenen örnekleme zamanında gönderilmesini saglar. Bunu bilgisayarın gerçek zamanlı saatinden aldıgı hassas zaman degeri ile yapar. Eger sürücü kartlarına referans gönderme isi örnekleme zamanını geçerse hata verir ve programın çalısmasını durdurur.*/
{
do{
gettimeofday(&timer,NULL);
end=timer.tv_usec;
elaptime=end-start_value;
if(elaptime>10000)
printf("!!! elapsed time: %d us !!!\n", elaptime);
}while(elaptime<time_value);
}
void readCurrent(int row,int ID) /* Fonksiyonun içinde sendCanPDOcurrent fonksiyonu çağırılırak akımlar okunur ve decimal formata çevrilir.*/
{
unsigned int udata;
sendCanPDOcurrent(ID);   // Read Average Current
udata= 256*rx[0].data[1]+rx[0].data[0];
if(rx[0].data[1]>128)
currents[row][ID-1] =udata-65536;
else
currents[row][ID-1] =udata;
}
void sendCanPDOsync(void) /* Sync nesnesi çağırılır ve sürücülerden bütün TxPDO'lar aynı anda gönderilir.*/
{
int got,sent,i;
tx.id=0x80;
tx.flags=0;
tx.length=0;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
}
void writeCurrent(void) /* Bu fonksiyon motorların okunan akım degerlerinin oluşturulan bir dosyada saklanması için yazılmıştır.*/
```

```
{
FILE *fcurrents,*fmotorTra;
int i,k;
time_t timer;
struct tm *tptr;
char fileNameCur[50]="currents_";
char fileNameTra[50]="motTra_";
char date[30];
time(&timer);
tptr = localtime(&timer);
sprintf(date,"%02d%02d%02d_%02d%02d%02d\0",tptr->tm_mday,(tptr-
>tm_mon+1),(tptr->tm_year + 1900)%100,tptr->tm_hour,tptr->tm_min,tptr-
>tm_sec);
strcat(fileNameCur,date);
strcat(fileNameTra,date);
strcat(fileNameCur, ".bin");
strcat(fileNameTra, ".bin");
if( (fcurrents=fopen(fileNameCur,"wb")) == NULL)
{
printf("Can not open file\n");
exit(EXIT_FAILURE);
}
if( (fmotorTra=fopen(fileNameTra,"wb")) == NULL)
{
printf("Can not open file\n");
exit(EXIT_FAILURE);
}
for(k=0;k<T_F;++k)
{
for(i=0;i<12;++i)
{
fprintf(fcurrents,"%d\t",currents[k][i]);
}
fprintf(fcurrents,"\n");
}
for(k=0;k<T_F;++k)
{
for(i=0;i<12;++i)
{
fprintf(fmotorTra,"%ld\t",motorTra[k][i]);
}
fprintf(fmotorTra,"\n");
}
fclose(fmotorTra);
fclose(fcurrents);
}
```

EK A.4

```
/*******************
terskinematik.c : Robot sıfır pozisyonuna getirildikten sonra istenen hareketi
yapabilmesi için gerekli eklem yörüngelerini ters kinematik yardımıyla üretir
ve bir dosyaya kaydeder. Kaydedilen dosya j2mTrans.c tarafından okunur ve
motor yörüngelerine çevirilir.
*******************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//Path Planner Parameters
#define T 0.01 //sampling period
#define sL 0.1 //step length
#define sH 0.07 //step height
#define sP 5 //step period
#define sN 2 //total step number
#define tD 0.0628 //torso downwards
#define tS 0.13 //torso sidewards
//KinematicParameters
#define w0 0.13//12.5;%-10;
#define d1 0.422//37.5;%30;
#define d2 0.4//37.5;%30;
void CalcR(double *pth);
void torsoMove(double x,double y,double z,double rotx,double roty,double rotz,
double per);
double *pol3(double fti,double ftf,double ti,double tf);
void stepRight(double stepHeight,double stepLength,double per);
void stepLeft(double stepHeight,double stepLength,double per);
void leftFootMove(double x,double y,double z,double rotx,double roty,double rotz,
double per);
void rightFootMove(double x,double y,double z,double rotx,double roty,double rotz,
double per);
double *pol5(double fti,double ftm,double ftf,double ti,double tf,double tm);
double d3[3] = {-0.03445,0,0.0788}; //[-3.7341 0 7.68]';%[-2.1 0 13]';
double tKeyPntsinit[2][19]= {{0,-0.03445,0,0.838,0,0,0,0,0.13,0,0,0,0,0,-
0.13,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};
double path[19];
double R[3][3];
double detR ;
double invR[3][3];
double RR[3][3];
double RRR[3][3];
double RRR1[3][3];
double RR1[3][3];
double detR1;
double invR1[3][3];
double R1[3][3];
FILE *fkp;
```

```
FILE *fp;
FILE *flp;
int main(int argc, char *argv[])
{
int i;
if ((fkp = fopen("totalKeyPoints2.bin", "w")) == NULL) {
printf("Cannot open totalKeyPoints.bin file!\n");
exit(EXIT_FAILURE);
}
if ((fp = fopen("path.bin", "w")) == NULL) {
printf("Cannot open path.bin file!\n");
exit(EXIT_FAILURE);
}
for (i=0; i < 19; i++) {
fprintf(fkp, "%f ", tKeyPntsinit[0][i]);
}
fprintf(fkp, "\n");
////////////////////////////////////////////////////////////////////////////
/* Yurume planlamalari*/ /* Bu bölümde fonksiyonlar ardarda çağırılarak
yürüme yörüngeleri oluşturulur.*/

torsoMove(0,0.13,0,0,0,0,sP);
//  leftFootMove(0,0,0.1,0,0,0,sP);
//  leftFootMove(0,0,-0.05,0,0,0,sP);
torsoMove(0,-0.13,0,0,0,0,sP);
torsoMove(0,-0.13,0,0,0,0,sP);
torsoMove(0,0.13,0,0,0,0,sP);
/*torsoMove(0,0.13,0,0,0,0,sP);
rightFootMove(0,0,0.02,0,0,0,sP);
rightFootMove(0,0,-0.02,0,0,0,sP);
torsoMove(0,-0.13,0,0,0,0,sP);
*/
/*torsoMove(0,0.05,0,0,0,0,sP);
// stepLeft(sH,sL,sP);
torsoMove(0,-0.05,0,0,0,0,sP);
*/
/*torsoMove(0,0.13,0,0,0,0,sP);
stepRight(sH,sL,sP);
torsoMove(sL/2,-0.13,0,0,0,0,sP);
*/
/*torsoMove(0,-0.13,0,0,0,0,sP);
stepLeft(sH,sL,sP);
torsoMove(sL,0.26,0,0,0,0,sP);
stepRight(sH,sL,sP);
torsoMove(0,-0.13,0,0,0,0,sP);
*/
/*torsoMove(0,0.13,0,0,0,0,sP);
stepRight(sH,sL,sP);
torsoMove(sL,-0.26,0,0,0,0,sP);
stepLeft(sH,sL,sP);
```

```
torsoMove(0,0.13,0,0,0,0,sP);
*/
/*torsoMove(0,0.13,0,0,0,0,sP);
stepRight(stepHeight,stepLength,sP);
torsoMove(stepLength,-0.13,0,0,0,0,sP);
stepLeft(stepHeight,stepLength,sP);
*/
/*rightFootMove(0,0,0.1,0,0,0,sP);
stepRight(sH,0,sP);
stepLeft(sH,0,sP);
torsoMove(sL/2,-2*tS,0,0,0,0,sP);
torsoMove(sL/2,2*tS,0,0,0,0,sP);
stepRight(sH,sL,sP);
torsoMove(sL/2,-2*tS,0,0,0,0,sP);
*/
fclose(fkp);
fclose(fp);
if ((fp = fopen("path.bin", "r")) == NULL) {
printf("Cannot open path.bin file!\n");
exit(EXIT_FAILURE);
}
if ((flp = fopen("jointTrajectory.bin", "w")) == NULL) {
printf("Cannot open jointTrajectory.bin file!\n");
exit(EXIT_FAILURE);
}
while(fscanf(fp,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf
%lf
%lf",path,path+1,path+2,path+3,path+4,path+5,path+6,path+7,path+8,path+9,path+1
0,path+11,path+12,path+13,path+14,path+15,path+16,path+17)!=EOF){
CalcR(path);
}
fclose(fp);
fclose(flp);
return 0;
}
////////////////////////////////////////////////////////////////////////////
void torsoMove(double x,double y,double z,double rotx,double roty,double rotz,
double per) /* Üç eksende çizgisel,açısal koordinatlar ve hareket periyodu
girilerek 3. dereceden polinomlar yardımıyla gövde yörüngeleri üretilir. */
{
int i, k;
double t;
double *pc;
tKeyPntsinit[1][0] = tKeyPntsinit[0][0] + per;
tKeyPntsinit[1][1] = tKeyPntsinit[0][1] + x;
tKeyPntsinit[1][2] = tKeyPntsinit[0][2] + y;
tKeyPntsinit[1][3] = tKeyPntsinit[0][3] + z;
tKeyPntsinit[1][4] = tKeyPntsinit[0][4] + rotx;
tKeyPntsinit[1][5] = tKeyPntsinit[0][5] + roty;
tKeyPntsinit[1][6] = tKeyPntsinit[0][6] + rotz;
```

```
for (i = 7; i < 19; i++) {
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
t = tKeyPntsinit[0][0];
for (i = 0; t <= tKeyPntsinit[1][0]; i++) {
for (k = 1; k < 19; k++) {
if (tKeyPntsinit[0][k] == tKeyPntsinit[1][k]) {
path[k] = tKeyPntsinit[0][k];
}
else {
pc =
pol3(tKeyPntsinit[0][k],tKeyPntsinit[1][k],tKeyPntsinit[0][0],tKeyPntsinit[1][0]);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3);
}
}
for (k = 1; k < 19; k++)
fprintf(fp, "%f ", path[k]);
fprintf(fp, "\n");
t += T;
}
for (i=0; i < 19; i++) {
fprintf(fkp, "%f ", tKeyPntsinit[1][i]);
}
fprintf(fkp, "\n");
for (i = 0; i<19; i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
}
void leftFootMove(double x,double y,double z,double rotx,double roty,double rotz,
double per) /* Üç eksende çizgisel,açısal koordinatlar ve hareket periyodu
girilerek 3. dereceden polinomlar yardımıyla sol ayak yörüngeleri üretilir. */
{
int i, k;
double t;
double *pc;
tKeyPntsinit[1][0] = tKeyPntsinit[0][0] + per;
tKeyPntsinit[1][7] = tKeyPntsinit[0][7] + x;
tKeyPntsinit[1][8] = tKeyPntsinit[0][8] + y;
tKeyPntsinit[1][9] = tKeyPntsinit[0][9] + z;
tKeyPntsinit[1][10] = tKeyPntsinit[0][10] + rotx;
tKeyPntsinit[1][11] = tKeyPntsinit[0][11] + roty;
tKeyPntsinit[1][12] = tKeyPntsinit[0][12] + rotz;
for (i = 1; i < 7 ; i++) {
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
for(i=13;i<19;i++){
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
t = tKeyPntsinit[0][0];
for (i = 0; t <= tKeyPntsinit[1][0]; i++) {
for (k = 1; k < 19; k++) {
```

```
if (tKeyPntsinit[0][k] == tKeyPntsinit[1][k]) {
path[k] = tKeyPntsinit[0][k];
}
else {
pc =
pol3(tKeyPntsinit[0][k],tKeyPntsinit[1][k],tKeyPntsinit[0][0],tKeyPntsinit[1][0]);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3);
}
}
for (k = 1; k < 19; k++)
fprintf(fp, "%f ", path[k]);
fprintf(fp, "\n");
t += T;
}
for (i=0; i < 19; i++) {
fprintf(fkp, "%f ", tKeyPntsinit[1][i]);
}
fprintf(fkp, "\n");
for (i = 0; i<19; i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
}
void rightFootMove(double x,double y,double z,double rotx,double roty,double rotz,
double per) /* Üç eksende çizgisel,açısal koordinatlar ve hareket periyodu
girilerek 3. dereceden polinomlar yardımıyla sağ ayak yörüngeleri üretilir. */
{
int i, k;
double t;
double *pc;
tKeyPntsinit[1][0] = tKeyPntsinit[0][0] + per;
tKeyPntsinit[1][13] = tKeyPntsinit[0][13] + x;
tKeyPntsinit[1][14] = tKeyPntsinit[0][14] + y;
tKeyPntsinit[1][15] = tKeyPntsinit[0][15] + z;
tKeyPntsinit[1][16] = tKeyPntsinit[0][16] + rotx;
tKeyPntsinit[1][17] = tKeyPntsinit[0][17] + roty;
tKeyPntsinit[1][18] = tKeyPntsinit[0][18] + rotz;
for (i = 1; i < 13; i++) {
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
t = tKeyPntsinit[0][0];
for (i = 0; t <= tKeyPntsinit[1][0]; i++) {
for (k = 1; k < 19; k++) {
if (tKeyPntsinit[0][k] == tKeyPntsinit[1][k]) {
path[k] = tKeyPntsinit[0][k];
}
else {
pc =
pol3(tKeyPntsinit[0][k],tKeyPntsinit[1][k],tKeyPntsinit[0][0],tKeyPntsinit[1][0]);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3);
}
}
```

```
for (k = 1; k < 19; k++)
fprintf(fp, "%f ", path[k]);
fprintf(fp, "\n");
t += T;
}
for (i=0; i < 19; i++) {
fprintf(fkp, "%f ", tKeyPntsinit[1][i]);
}
fprintf(fkp, "\n");
for (i = 0; i<19; i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
}
void stepLeft(double stepHeight,double stepLength,double per) /* Adım
yüksekliği,adım uzunluğu ve hareket periyodu girilerek 3. dereceden ve
5.dereceden
polinomlar yardımıyla sol ayak yörüngeleri üretilir. Z ekseninde ayağın yerden
kaldırılıp tekrar indirilmesi için 5. dereceden polinom uydurulur.*/
{
int i, k;
double t;
double *pc;
tKeyPntsinit[1][0] = tKeyPntsinit[0][0] + per;
for (i = 1 ;i < 7; i++) {
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
tKeyPntsinit[1][7] = tKeyPntsinit[0][7]+stepLength ;
tKeyPntsinit[1][8] = tKeyPntsinit[0][8];
tKeyPntsinit[1][9] = tKeyPntsinit[0][9]+stepHeight;
for(i=10; i<19;i++){
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
t = tKeyPntsinit[0][0];
for (i = 0; t <= tKeyPntsinit[1][0]; i++) {
for (k = 1; k < 19; k++) {
if (tKeyPntsinit[0][k] == tKeyPntsinit[1][k]) {
path[k] = tKeyPntsinit[0][k];
}
else if(k==9){
pc =
pol5(tKeyPntsinit[0][k],tKeyPntsinit[0][k]+stepHeight,tKeyPntsinit[0][k],tKeyPntsin
it[0][0],tKeyPntsinit[1][0],tKeyPntsinit[0][0]+per/2);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3)+ pc[4]*pow(t,4)+
pc[5]*pow(t,5);
}
else {
pc =
pol3(tKeyPntsinit[0][k],tKeyPntsinit[1][k],tKeyPntsinit[0][0],tKeyPntsinit[1][0]);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3);
}
}
for (k = 1; k < 19; k++)
```

```
fprintf(fp, "%f ", path[k]);
fprintf(fp, "\n");
t += T;
}
for (i=0; i < 19; i++) {
fprintf(fkp, "%f ", tKeyPntsinit[1][i]);
}
fprintf(fkp, "\n");
for (i = 0; i<9; i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
tKeyPntsinit[0][9] = tKeyPntsinit[0][9];
for (i=10;i<19;i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
}
void stepRight(double stepHeight,double stepLength,double per) /* Adım
yüksekliği,adım uzunluğu ve hareket periyodu girilerek 3. dereceden ve
5.dereceden
polinomlar yardımıyla sağ ayak yörüngeleri üretilir. Z ekseninde ayağın yerden
kaldırılıp tekrar indirilmesi için 5. dereceden polinom uydurulur.*/
{
int i, k;
double t;
double *pc;
tKeyPntsinit[1][0] = tKeyPntsinit[0][0] + per;
for (i = 1; i < 13; i++) {
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
tKeyPntsinit[1][13] = tKeyPntsinit[0][13] + stepLength;
tKeyPntsinit[1][14] = tKeyPntsinit[0][14];
tKeyPntsinit[1][15]=tKeyPntsinit[0][15]+stepHeight;
for (i = 16; i < 19; i++) {
tKeyPntsinit[1][i] = tKeyPntsinit[0][i];
}
t = tKeyPntsinit[0][0];
for (i = 0; t <= tKeyPntsinit[1][0]; i++) {
for (k = 1; k < 19; k++) {
if (tKeyPntsinit[0][k] == tKeyPntsinit[1][k]) {
path[k] = tKeyPntsinit[0][k];
}
else if(k==15){
pc =
pol5(tKeyPntsinit[0][k],tKeyPntsinit[0][k]+stepHeight,tKeyPntsinit[0][k],tKeyPntsin
it[0][0],tKeyPntsinit[1][0],tKeyPntsinit[0][0]+per/2);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3)+ pc[4]*pow(t,4)+
pc[5]*pow(t,5);
}
else {
pc =
pol3(tKeyPntsinit[0][k],tKeyPntsinit[1][k],tKeyPntsinit[0][0],tKeyPntsinit[1][0]);
path[k] = pc[0] + pc[1]*t + pc[2]*pow(t,2) + pc[3]*pow(t,3);
```

```
}
}
for (k = 1; k < 19; k++)
fprintf(fp, "%f ", path[k]);
fprintf(fp, "\n");
t += T;
}
for (i=0; i < 19; i++) {
fprintf(fkp, "%f ", tKeyPntsinit[1][i]);
}
fprintf(fkp, "\n");
for (i = 0; i<15; i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
tKeyPntsinit[0][15] = tKeyPntsinit[0][15];
for(i=16; i<19;i++)
tKeyPntsinit[0][i] = tKeyPntsinit[1][i];
}

void CalcR(double *pth) /* 3. ve 5. dereceden polinomlarla üretilen yörüngeler
bu fonksiyon içinde ters kinematik işlemle eklem yörüngelerine çevirilir ve
bir dosyaya kaydedilir.*/
{
double
x01,x02,x03,alpha0,beta0,gama0,xx1,xx2,xx3,alpha,beta,gama,xx1r,xx2r,xx3r,alphar
,betar,gamar;
x01 = pth[0];
x02 = pth[1];
x03 = pth[2];
alpha0 = pth[3];
beta0 = pth[4];
gama0 = pth[5];
xx1 = pth[6];
xx2 = pth[7];
xx3 = pth[8];
alpha= pth[9];
beta = pth[10];
gama = pth[11];
xx1r = pth[12];
xx2r = pth[13];
xx3r = pth[14];
alphar = pth[15];
betar = pth[16];
gamar = pth[17];
int i,j,k,l,m,n,s;
double sum;
double R[3][3]= {{cos(alpha)*cos(beta), cos(alpha)*sin(beta)*sin(gama) -
cos(gama)*sin(alpha), sin(alpha)*sin(gama) + cos(alpha)*cos(gama)*sin(beta)},
                {cos(beta)*sin(alpha), cos(alpha)*cos(gama) +
sin(alpha)*sin(beta)*sin(gama), cos(gama)*sin(alpha)*sin(beta) -
cos(alpha)*sin(gama)},
```

```
            {-sin(beta),cos(beta)*sin(gama),cos(beta)*cos(gama)}};

double R1[3][3]= {{cos(alphar)*cos(betar), cos(alphar)*sin(betar)*sin(gamar) -
cos(gamar)*sin(alphar), sin(alphar)*sin(gamar) +
cos(alphar)*cos(gamar)*sin(betar)},
            {cos(betar)*sin(alphar), cos(alphar)*cos(gamar) +
sin(alphar)*sin(betar)*sin(gamar), cos(gamar)*sin(alphar)*sin(betar) -
cos(alphar)*sin(gamar)},
            {-sin(betar),cos(betar)*sin(gamar),cos(betar)*cos(gamar)}};
detR= R[0][0]*(R[1][1]*R[2][2]-R[1][2]*R[2][1]) + R[0][1]*(R[1][2]*R[2][0]-
R[2][2]*R[1][0]) + R[0][2]*(R[1][0]*R[2][1]-R[1][1]*R[2][0]);
detR1=R1[0][0]*(R1[1][1]*R1[2][2]-R1[1][2]*R1[2][1]) +
R1[0][1]*(R1[1][2]*R1[2][0]-R1[2][2]*R1[1][0]) + R1[0][2]*(R1[1][0]*R1[2][1]-
R1[1][1]*R1[2][0]);
if(detR!=0 && detR1!=0) {
double invR[3][3] = {{(R[1][1]*R[2][2]-R[1][2]*R[2][1])/detR, (R[0][2]*R[2][1]-
R[0][1]*R[2][2])/detR, (R[0][1]*R[1][2]- R[0][2]*R[1][1])/detR},
          {(R[1][2]*R[2][0]-R[2][2]*R[1][0])/detR, (R[0][0]*R[2][2]-
R[0][2]*R[2][0])/detR, (R[0][2]*R[1][0]- R[0][0]*R[1][2])/detR},
          {(R[1][0]*R[2][1]-R[1][1]*R[2][0])/detR, (R[2][0]*R[0][1]-
R[0][0]*R[2][1])/detR, (R[0][0]*R[1][1]- R[0][1]*R[1][0])/detR}};
double invR1[3][3] = {{(R1[1][1]*R1[2][2]-R1[1][2]*R1[2][1])/detR1,
(R1[0][2]*R1[2][1]- R1[0][1]*R1[2][2])/detR1, (R1[0][1]*R1[1][2]-
R1[0][2]*R1[1][1])/detR1},
          {(R1[1][2]*R1[2][0]-R1[2][2]*R1[1][0])/detR1, (R1[0][0]*R1[2][2]-
R1[0][2]*R1[2][0])/detR1, (R1[0][2]*R1[1][0]- R1[0][0]*R1[1][2])/detR1},
          {(R1[1][0]*R1[2][1]-R1[1][1]*R1[2][0])/detR1, (R1[2][0]*R1[0][1]-
R1[0][0]*R1[2][1])/detR1, (R1[0][0]*R1[1][1]- R1[0][1]*R1[1][0])/detR1}};
double RO[3][3] = {{cos(alpha0)*cos(beta0), cos(alpha0)*sin(beta0)*sin(gama0) -
cos(gama0)*sin(alpha0), sin(alpha0)*sin(gama0) +
cos(alpha0)*cos(gama0)*sin(beta0)},
            {cos(beta0)*sin(alpha0), cos(alpha0)*cos(gama0) +
sin(alpha0)*sin(beta0)*sin(gama0), cos(gama0)*sin(alpha0)*sin(beta0) -
cos(alpha0)*sin(gama0)},
            {-sin(beta0),cos(beta0)*sin(gama0),cos(beta0)*cos(gama0)}};
double x0[3][1]= {{x01},
           {x02},
           {x03}};
double xx[3][1]= {{xx1},
           {xx2},
           {xx3}};
double xxr[3][1]= {{xx1r},
           {xx2r},
           {xx3r}};
double xh[3][1]= {{x0[0][0]+RO[0][1]*w0},
           {x0[1][0]+RO[1][1]*w0},
           {x0[2][0]+RO[2][1]*w0}}; //w0=0.13
double xh1[3][1]= {{x0[0][0]+RO[0][1]*(-w0)},
           {x0[1][0]+RO[1][1]*(-w0)},
           {x0[2][0]+RO[2][1]*(-w0)}};
```

```
double xa[3][1];
double xar[3][1];
xa[0][0]= xx[0][0] + R[0][0]*d3[0]+R[0][1]*d3[1]+R[0][2]*d3[2];
xa[1][0]= xx[1][0] + R[1][0]*d3[0]+R[1][1]*d3[1]+R[1][2]*d3[2];
xa[2][0]= xx[2][0] + R[2][0]*d3[0]+R[2][1]*d3[1]+R[2][2]*d3[2];
xar[0][0]= xxr[0][0] + R1[0][0]*d3[0]+R1[0][1]*d3[1]+R1[0][2]*d3[2];
xar[1][0]= xxr[1][0] + R1[1][0]*d3[0]+R1[1][1]*d3[1]+R1[1][2]*d3[2];
xar[2][0]= xxr[2][0] + R1[2][0]*d3[0]+R1[2][1]*d3[1]+R1[2][2]*d3[2];
double D[3][1];
D[0][0]= invR[0][0]*(xh[0][0]-xa[0][0])+invR[0][1]*(xh[1][0]-
xa[1][0])+invR[0][2]*(xh[2][0]-xa[2][0]);
D[1][0]= invR[1][0]*(xh[0][0]-xa[0][0])+invR[1][1]*(xh[1][0]-
xa[1][0])+invR[1][2]*(xh[2][0]-xa[2][0]);
D[2][0]= invR[2][0]*(xh[0][0]-xa[0][0])+invR[2][1]*(xh[1][0]-
xa[1][0])+invR[2][2]*(xh[2][0]-xa[2][0]);
double D1[3][1];
D1[0][0]= invR1[0][0]*(xh1[0][0]-xar[0][0])+invR1[0][1]*(xh1[1][0]-
xar[1][0])+invR1[0][2]*(xh1[2][0]-xar[2][0]);
D1[1][0]= invR1[1][0]*(xh1[0][0]-xar[0][0])+invR1[1][1]*(xh1[1][0]-
xar[1][0])+invR1[1][2]*(xh1[2][0]-xar[2][0]);
D1[2][0]= invR1[2][0]*(xh1[0][0]-xar[0][0])+invR1[2][1]*(xh1[1][0]-
xar[1][0])+invR1[2][2]*(xh1[2][0]-xar[2][0]);
double u = D[0][0];
double v = D[1][0];
double w = D[2][0];
double u1=D1[0][0];
double v1=D1[1][0];
double w1=D1[2][0];
double theta4 = acos((u*u+v*v+w*w-d1*d1-d2*d2)/(2*d1*d2));  // d1=0.422,
d2=0.4
double theta5 = asin((-d1*sin(theta4))/(sqrt(u*u+v*v+w*w)))-
atan(u/sqrt(w*w+v*v));  // d1=0.422, d2=0.4
double theta6 = atan(v/w);
double theta10 = acos((u1*u1+v1*v1+w1*w1-d1*d1-d2*d2)/(2*d1*d2));  //
d1=0.422, d2=0.4
double theta11 = asin((-d1*sin(theta10))/(sqrt(u1*u1+v1*v1+w1*w1)))-
atan(u1/sqrt(w1*w1+v1*v1));  // d1=0.422, d2=0.4
double theta12 = atan(v1/w1);
double RA[3][3] = {{cos(theta4 + theta5), sin(theta4 + theta5)*sin(theta6), sin(theta4
+ theta5)*cos(theta6)},
                { 0,cos(theta6),-sin(theta6)},
                { -sin(theta4 + theta5), cos(theta4 + theta5)*sin(theta6), cos(theta4 +
theta5)*cos(theta6)}};
double RA1[3][3] = {{cos(theta10 + theta11), sin(theta10 + theta11)*sin(theta12),
sin(theta10 + theta11)*cos(theta12)},
                { 0,cos(theta12),-sin(theta12)},
                { -sin(theta10 + theta11), cos(theta10 + theta11)*sin(theta12),
cos(theta10 + theta11)*cos(theta12)}};
for (i = 0; i <3; i++) {
for (j = 0; j <3; j++) {
```

```
sum = 0;
for (k = 0; k < 3; k++) {
sum += RA[i][k] * invR[k][j];
}
RRR[i][j] = sum;
}
}
for (l = 0; l <3; l++) {
for (m = 0; m <3; m++) {
sum = 0;
for (s = 0; s < 3; s++) {
sum += RRR[l][s] * RO[s][m];
}
RR[l][m] = sum;
}
}
for (i = 0; i <3; i++) {
for (j = 0; j <3; j++) {
sum = 0;
for (k = 0; k < 3; k++) {
sum += RA1[i][k] * invR1[k][j];
}
RRR1[i][j] = sum;
}
}
for (l = 0; l <3; l++) {
for (m = 0; m <3; m++) {
sum = 0;
for (s = 0; s < 3; s++) {
sum += RRR1[l][s] * RO[s][m];
}
RR1[l][m] = sum;
}
}
double theta3 = -atan2(RR[0][2],RR[2][2]);
double theta2 = atan2(RR[1][2],sqrt(RR[1][0]*RR[1][0]+RR[1][1]*RR[1][1]));
double theta1 = -atan2(RR[1][0],RR[1][1]);
double theta9 = -atan2(RR1[0][2],RR1[2][2]);
double theta8 =
atan2(RR1[1][2],sqrt(RR1[1][0]*RR1[1][0]+RR1[1][1]*RR1[1][1]));
double theta7 = -atan2(RR1[1][0],RR1[1][1]);
fprintf(flp,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf
%lf",theta1,theta2,theta3,theta4,theta5,theta6,theta7,theta8,theta9,theta10,theta11,thet
a12);
fprintf(flp,"\n");
}
else {
fprintf(flp,"matrix is not invertible");
}
}
```

```
double *pol3(double fti,double ftf,double ti,double tf) /* Başlangıç zamanındaki
değer, bitiş zamanındaki değer, başlangıç zamanı ve bitiş zamanı girilerek
3. dereceden polinom uydurulur. */
{
static double p[4];
p[0] = (fti*tf*tf*(tf - 3*ti))/((tf - ti)*(tf-ti)*(tf-ti)) + (ftf*ti*ti*(3*tf - ti))/((tf - ti)*(tf -
ti)*(tf- ti));
p[1] = (6*fti*tf*ti)/((tf - ti)*(tf - ti)*(tf - ti)) - (6*ftf*tf*ti)/((tf - ti)*(tf - ti)*(tf - ti));
p[2] = (ftf*(3*tf + 3*ti))/((tf - ti)*(tf - ti)*(tf - ti)) - (fti*(3*tf + 3*ti))/((tf - ti)*(tf -
ti)*(tf - ti));
p[3] = (2*fti)/((tf - ti)*(tf - ti)*(tf - ti)) - (2*ftf)/((tf - ti)*(tf - ti)*(tf - ti));
return p;
}
double *pol5(double fti,double ftm,double ftf,double ti,double tf,double tm) /*
Başlangıç zamanındaki değer, hareketin yarısı bitirilene kadar geçen süredeki
değer,
bitiş zamanındaki değer, başlangıç zamanı,bitiş zamanı ve hareketin yarısına
kadar geçen süre girilerek 5. dereceden polinom uydurulur.*/
{
static double p[6];
p[0] = (fti*tf*tf*tm*tm*(3*tf*ti - tf*tm + 3*ti*tm - 5*ti*ti))/((tf - ti)*(tf - ti)*(tf -
ti)*(ti - tm)*(ti - tm)*(ti - tm)) - (ftf*ti*ti*tm*tm*(3*tf*ti + 3*tf*tm - ti*tm -
5*tf*tf))/((tf - ti)*(tf - ti)*(tf - ti)*(tf - tm)*(tf-tm)*(tf-tm)) + (ftm*tf*tf*ti*ti*(tf*ti -
3*tf*tm - 3*ti*tm + 5*tm*tm))/((tf - tm)*(tf-tm)*(tf-tm)*(ti - tm)*(ti-tm)*(ti-tm));
p[1] = (2*ftf*tf*ti*tm*(3*ti*ti + 4*ti*tm - 5*tf*ti + 3*tm*tm - 5*tf*tm))/((tf - ti)*(tf
- ti)*(tf - ti)*(tf - tm)*(tf - tm)*(tf - tm)) - (2*fti*tf*ti*tm*(3*tf*tf + 4*tf*tm - 5*ti*tf
+ 3*tm*tm - 5*ti*tm))/((tf - ti)*(tf-ti)*(tf-ti)*(ti - tm)*(ti-tm)*(ti-tm)) +
(2*ftm*tf*ti*tm*(3*tf*tf + 4*tf*ti - 5*tm*tf + 3*ti*ti - 5*tm*ti))/((tf - tm)*(tf-
tm)*(tf-tm)*(ti - tm)*(ti-tm)*(ti-tm));
p[2] = (fti*(3*tf*tf*tf*ti + 3*tf*tf*tf*tm - 5*tf*tf*ti*ti + 7*tf*tf*ti*tm +
4*tf*tf*tm*tm - 20*tf*ti*ti*tm + 7*tf*ti*tm*tm + 3*tf*tm*tm*tm - 5*ti*ti*tm*tm +
3*ti*tm*tm*tm))/((tf - ti)*(tf-ti)*(tf-ti)*(ti - tm)*(ti-tm)*(ti-tm)) - (ftf*(- 5*tf*tf*ti*ti
- 20*tf*tf*ti*tm - 5*tf*tf*tm*tm + 3*tf*ti*ti*ti + 7*tf*ti*ti*tm + 7*tf*ti*tm*tm +
3*tf*tm*tm*tm + 3*ti*ti*ti*tm + 4*ti*ti*tm*tm + 3*ti*tm*tm*tm))/((tf - ti)*(tf-
ti)*(tf-ti)*(tf - tm)*(tf-tm)*(tf-tm)) - (ftm*(3*tf*tf*tf*ti + 3*tf*tf*tf*tm +
4*tf*tf*ti*ti + 7*tf*tf*ti*tm - 5*tf*tf*tm*tm + 3*tf*ti*ti*ti + 7*tf*ti*ti*tm -
20*tf*ti*tm*tm + 3*ti*ti*ti*tm - 5*ti*ti*tm*tm))/((tf - tm)*(tf-tm)*(tf-tm)*(ti -
tm)*(ti-tm)*(ti-tm));
p[3] = (ftf*(- 10*tf*tf*ti - 10*tf*tf*tm + 2*tf*ti*ti - 4*tf*ti*tm + 2*tf*tm*tm +
2*ti*ti*ti + 8*ti*ti*tm + 8*ti*tm*tm + 2*tm*tm*tm))/((tf - ti)*(tf-ti)*(tf-ti)*(tf -
tm)*(tf-tm)*(tf-tm)) - (fti*(2*tf*tf*tf + 2*tf*tf*ti + 8*tf*tf*tm - 10*tf*ti*ti -
4*tf*ti*tm + 8*tf*tm*tm - 10*ti*ti*tm + 2*ti*tm*tm + 2*tm*tm*tm))/((tf - ti)*(tf-
ti)*(tf-ti)*(ti - tm)*(ti-tm)*(ti-tm)) + (ftm*(2*tf*tf*tf + 8*tf*tf*ti + 2*tf*tf*tm +
8*tf*ti*ti - 4*tf*ti*tm - 10*tf*tm*tm + 2*ti*ti*ti + 2*ti*ti*tm - 10*ti*tm*tm))/((tf -
tm)*(tf-tm)*(tf-tm)*(ti - tm)*(ti-tm)*(ti-tm));
p[4] = (ftf*(5*tf*tf + 5*tf*ti + 5*tf*tm - 4*ti*ti - 7*ti*tm - 4*tm*tm))/((tf - ti)*(tf-
ti)*(tf-ti)*(tf - tm)*(tf-tm)*(tf-tm)) - (fti*(- 4*tf*tf + 5*tf*ti - 7*tf*tm + 5*ti*ti +
5*ti*tm - 4*tm*tm))/((tf - ti)*(tf-ti)*(tf-ti)*(ti - tm)*(ti-tm)*(ti-tm)) - (ftm*(4*tf*tf +
7*tf*ti - 5*tf*tm + 4*ti*ti - 5*ti*tm - 5*tm*tm))/((tf - tm)*(tf-tm)*(tf-tm)*(ti -
tm)*(ti-tm)*(ti-tm));
```

```
p[5] = (ftf*(2*ti - 4*tf + 2*tm))/((tf - ti)*(tf-ti)*(tf-ti)*(tf - tm)*(tf-tm)*(tf-tm)) -
(fti*(2*tf - 4*ti + 2*tm))/((tf - ti)*(tf-ti)*(tf-ti)*(ti - tm)*(ti-tm)*(ti-tm)) + (ftm*(2*tf
+ 2*ti - 4*tm))/((tf - tm)*(tf-tm)*(tf-tm)*(ti - tm)*(ti-tm)*(ti-tm));
return p;
}
```

EK A.5

```
/************
ftzero.c : sendCanFT1 ve sendCanFT2 fonksiyonları çağırılır ve ilgili Canbus
komutuyla sapma değerleri sıfırlanır.
*************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include "can4linux.h"
#define STDDEV "can1"
#define RXBUFFERSIZE 100
int fd;
canmsg_t tx;
canmsg_t rx[RXBUFFERSIZE];
void initCan(void);
void sendCanFT1(int ID);
void sendCanFT2(int ID);
int main(void)
{
printf("initialising Can...\n");
initCan();
printf("Sending...\n");
sendCanFT1(32);  //Sol ayak
sendCanFT2(432); //Sag ayak
close(fd);
return 0;
}
void initCan(void)
{
char device[40];
sprintf(device, "/dev/%s", STDDEV);
if(( fd = open(device, O_RDWR )) < 0 )
{
fprintf(stderr, "Error opening CAN device %s\n", device);
exit(1);
}
}
void sendCanFT1(int ID)
{
int got,sent,i=0;
unsigned short F[3],T[3];
short f[3],t[3];
double factor[2]={32.77,1310.62};
double f_f[3],f_t[3];
```

```
tx.id=ID;
tx.data[0]=4;
tx.flags=0;
tx.length=1;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
tx.id=ID;
tx.data[0]=2;
tx.flags=0;
tx.length=1;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
do{
got= read(fd,&rx,1);
}while(got<1);
F[0]= 256*rx[0].data[1] + rx[0].data[0];
F[1]= 256*rx[0].data[5] + rx[0].data[4];
T[0]= 256*rx[0].data[3] + rx[0].data[2];
T[1]= 256*rx[0].data[7] + rx[0].data[6];
do{
got= read(fd,&rx,1);
}while(got<1);
F[2]= 256*rx[0].data[1] + rx[0].data[0];
T[2]= 256*rx[0].data[3] + rx[0].data[2];
for(i=0;i<3;++i)
{
t[i]=T[i];
f[i]=F[i];
f_f[i]=f[i]/factor[0];
f_t[i]=t[i]/factor[1];
printf("Fsol[%d] = %.2f \n",i,f_f[i]);
printf("Tsol[%d] = %.2f \n",i,f_t[i]);
}
}
void sendCanFT2(int ID)
{
int got,sent,i=0;
unsigned short F[3],T[3];
short f[3],t[3];
double factor[2]={32.77,1310.62};
double f_f[3],f_t[3];
tx.id=ID;
tx.data[0]=4;
tx.flags=0;
tx.length=1;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
```

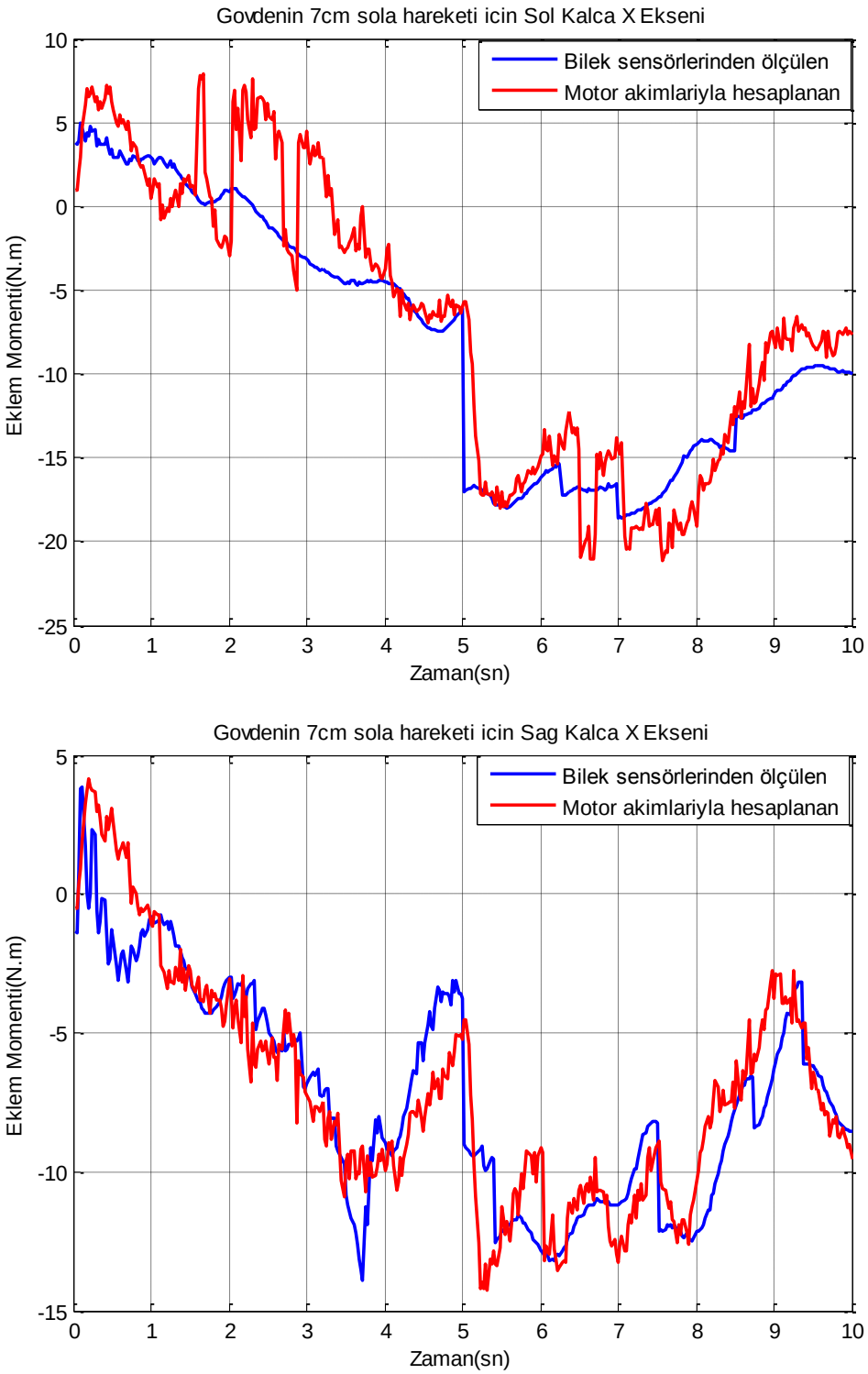
```
tx.id=ID;
tx.data[0]=2;
tx.flags=0;
tx.length=1;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
do{
got= read(fd,&rx,1);
}while(got<1);
F[0]= 256*rx[0].data[1] + rx[0].data[0];
F[1]= 256*rx[0].data[5] + rx[0].data[4];
T[0]= 256*rx[0].data[3] + rx[0].data[2];
T[1]= 256*rx[0].data[7] + rx[0].data[6];
do{
got= read(fd,&rx,1);
}while(got<1);
F[2]= 256*rx[0].data[1] + rx[0].data[0];
T[2]= 256*rx[0].data[3] + rx[0].data[2];
for(i=0;i<3;++i)
{
t[i]=T[i];
f[i]=F[i];
f_f[i]=f[i]/factor[0];
f_t[i]=t[i]/factor[1];
printf("Fsag[%d] = %.2f \n",i,f_f[i]);
printf("Tsag[%d] = %.2f \n",i,f_t[i]);
}
}
```

EK A.6

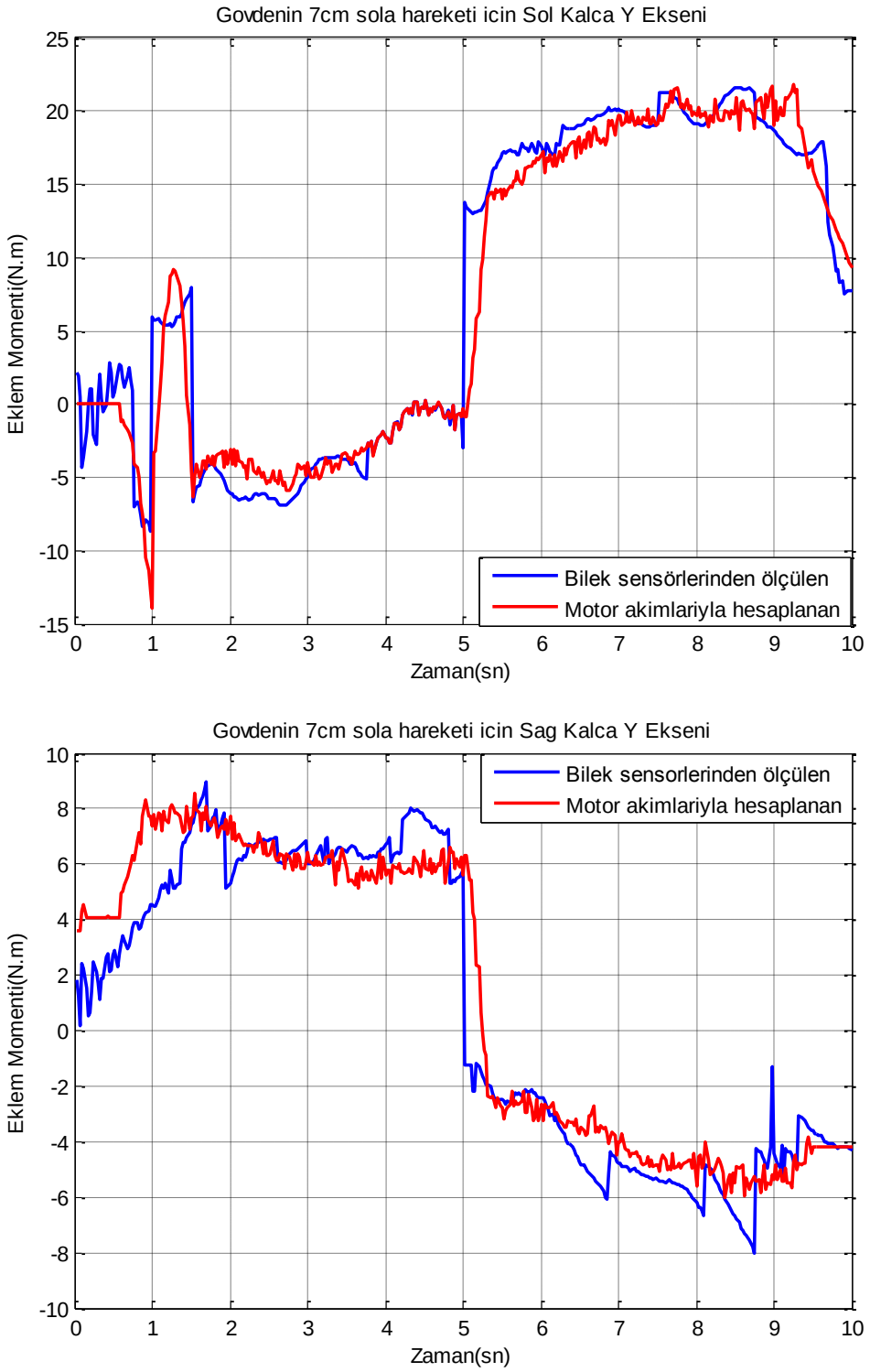
```
/* Bileklerdeki sensörlerden alınan kuvvet/moment değerleri jakobiyen
aracılığıyla eklem momentlerine çevirilir. Bu program ileri kinematik
kullanılarak jakobiyenin elde edilmesini sağlar. */
rotz:=proc(a)  Matrix(3,3,[cos(a),sin(a),0,-sin(a),cos(a),0,0,0,1])  end proc:
roty:=proc(a)  Matrix(3,3,[cos(a),0,-sin(a),0,1,0,sin(a),0,cos(a)])  end proc:
rotx:=proc(a)  Matrix(3,3,[1,0,0,0,cos(a),sin(a),0,-sin(a),cos(a)])  end proc:
tr:=IdentityMatrix(3):
T01:=Matrix(rotz(q[1](t))): T12:=Matrix(rotx(q[2](t))): T23:=Matrix(roty(q[3](t))):
T34:=Matrix(roty(q[4](t))): T45:=Matrix(roty(q[5](t))): T56:=Matrix(rotx(q[6](t))):
T02:=simplify(MatrixMatrixMultiply(T12,T01)):
T03:=simplify(MatrixMatrixMultiply(T23,T02)):
T04:=simplify(MatrixMatrixMultiply(T34,T03)):
T05:=simplify(MatrixMatrixMultiply(T45,T04)):
T06:=simplify(MatrixMatrixMultiply(T56,T05)):   T01T:=Matrix(transpose(T01)):
T02T:=Matrix(transpose(T02)): T03T:=Matrix(transpose(T03)):
T04T:=Matrix(transpose(T04)): T05T:=Matrix(transpose(T05)):
T06T:=Matrix(transpose(T06)):
O01:=Vector[column]([0,-130,0]):   O12:=Vector[column]([0,0,0]):
O23:=Vector[column]([0,0,0]):          O34:=Vector[column]([0,0,-420]):
O45:=Vector[column]([0,0,-400]):  O56:=Vector[column]([0,0,0]):
O6C:=Vector[column]([0,0,-40]):    OC:=Vector[column]([0,0,0]):
OC:=VectorAdd(O01,MatrixVectorMultiply(T01T,O12)):
OC:=VectorAdd(OC,MatrixVectorMultiply(T02T,O23)):
OC:=VectorAdd(OC,MatrixVectorMultiply(T03T,O34)):
OC:=VectorAdd(OC,MatrixVectorMultiply(T04T,O45)):
OC:=VectorAdd(OC,MatrixVectorMultiply(T05T,O56)):
OC:=VectorAdd(OC,MatrixVectorMultiply(T06T,O6C));
#OC:=O01+MatrixVectorMultiply(T01T,O12)+MatrixVectorMultiply(T02T,O23)+
MatrixVectorMultiply(T03T,O34)+MatrixVectorMultiply(T04T,O45)+MatrixVector
Multiply(T05T,O56)+MatrixVectorMultiply(T06T,O6C);                #q[1]:=0;
q[2]:=0; q[3]:=0; q[4]:=0; q[5]:=0; q[6]:=0;   T01:=Matrix(rotz(q[1])):
T12:=Matrix(rotx(q[2])): T23:=Matrix(rotx(q[3])): T34:=Matrix(rotz(q[4])):
T45:=Matrix(roty(q[5])): T56:=Matrix(rotz(q[6])):
T02:=simplify(MatrixMatrixMultiply(T12,T01)):
T03:=simplify(MatrixMatrixMultiply(T23,T02)):
T04:=simplify(MatrixMatrixMultiply(T34,T03)):
T05:=simplify(MatrixMatrixMultiply(T45,T04)):
T06:=simplify(MatrixMatrixMultiply(T56,T05)):   T01T:=Matrix(transpose(T01)):
T02T:=Matrix(transpose(T02)): T03T:=Matrix(transpose(T03)):
T04T:=Matrix(transpose(T04)): T05T:=Matrix(transpose(T05)):
T06T:=Matrix(transpose(T06)):
OC:=O01+MatrixVectorMultiply(T01T,O12)+MatrixVectorMultiply(T02T,O23)+M
atrixVectorMultiply(T03T,O34)+MatrixVectorMultiply(T04T,O45)+MatrixVectorM
ultiply(T05T,O56)+MatrixVectorMultiply(T06T,O6C);
Omega00:=Vector[column]([0,0,0]):       Omega10:=Vector[column]([0,0,dq[1]]):
Omega21:=Vector[column]([dq[2],0,0]):   Omega32:=Vector[column]([0,dq[3],0]):
Omega43:=Vector[column]([0,dq[4],0]):   Omega54:=Vector[column]([0,dq[5],0]):
Omega65:=Vector[column]([dq[6],0,0]):
```

```
Omega[1]:=VectorAdd(Omega10,MatrixVectorMultiply(T01,Omega00)):
Omega[1]:=combine(Omega[1],trig):
Omega[2]:=VectorAdd(Omega21,MatrixVectorMultiply(T12,Omega[1])):
Omega[2]:=combine(Omega[2],trig):
Omega[3]:=VectorAdd(Omega32,MatrixVectorMultiply(T23,Omega[2])):
Omega[3]:=combine(Omega[3],trig):
Omega[4]:=VectorAdd(Omega43,MatrixVectorMultiply(T34,Omega[3])):
Omega[4]:=combine(Omega[4],trig):
Omega[5]:=VectorAdd(Omega54,MatrixVectorMultiply(T45,Omega[4])):
Omega[5]:=combine(Omega[5],trig):
Omega[6]:=VectorAdd(Omega65,MatrixVectorMultiply(T56,Omega[5])):
Omega[6]:=combine(Omega[6],trig):
Omega[6]:=MatrixVectorMultiply(T06T,Omega[6]);
#dT06:=Matrix(3,3,symbol=dt06):
for i to 3 do  for j to 3 do  dt06[i,j]:=-diff(T06[i,j],t)  end do:  end do:  dT06:
Omega6Mat:=MatrixMatrixMultiply(dT06,T06T):
Omega6Vec:=Vector[column]([Omega6Mat[3,2],Omega6Mat[1,3],Omega6Mat[2,1]
]):  Vect:=combine(Omega[6]-Omega6Vec,trig);
V10:=Vector[column]([0,0,0]): V21:=Vector[column]([0,0,0]):
V32:=Vector[column]([0,0,0]):          V43:=Vector[column]([0,0,0]):
V54:=Vector[column]([0,0,0]): V65:=Vector[column]([0,0,0]):
V20:=Vector[column]([0,0,0]): V30:=Vector[column]([0,0,0]):
V40:=Vector[column]([0,0,0]):          V50:=Vector[column]([0,0,0]):
V60:=Vector[column]([0,0,0]): VC6:=Vector[column]([0,0,0]):
V10:=MatrixVectorMultiply(T01,V10):  V[1]:=V10:
V20:=VectorAdd(V[1],V21):
V20:=VectorAdd(V20,CrossProduct(Omega[1],O12)):
V20:=MatrixVectorMultiply(T12,V20):  V[2]:=V20:
V30:=VectorAdd(V[2],V32):
V30:=VectorAdd(V30,CrossProduct(Omega[2],O23)):
V30:=MatrixVectorMultiply(T23,V30):  V[3]:=V30:
V40:=VectorAdd(V[3],V43):
V40:=VectorAdd(V40,CrossProduct(Omega[3],O34)):
V40:=MatrixVectorMultiply(T34,V40):  V[4]:=V40:
V50:=VectorAdd(V[4],V54):
V50:=VectorAdd(V50,CrossProduct(Omega[4],O45)):
V50:=MatrixVectorMultiply(T45,V50):  V[5]:=V50:
V60:=VectorAdd(V[5],V65):
V60:=VectorAdd(V60,CrossProduct(Omega[5],O56)):
V60:=MatrixVectorMultiply(T56,V60):  V[6]:=V60:
VC0:=VectorAdd(V[6],VC6):
VC0:=VectorAdd(VC0,CrossProduct(Omega[6],O6C)):  for i to n do
VC0:=collect(VC0,dq[i]) end do: VC0:=VC0:
VC0:=MatrixVectorMultiply(T06T,VC0);
Jt0:=Matrix(3,n,symbol=jt0):  Jr0:=Matrix(3,n,symbol=jr0):
for i to 3 do  for j to n do  jt0[i,j]:=coeff(VC0[i],dq[j])  end do:  end do:
Jt0:=combine(Jt0,trig);  for i to 3 do  for j to n do
jr0[i,j]:=coeff(Omega[6][i],dq[j])  end do:  end do:  Jr0:=combine(Jr0,trig);
#J0:=Matrix(6,n,[[Jr0],[Jt0]]);
```

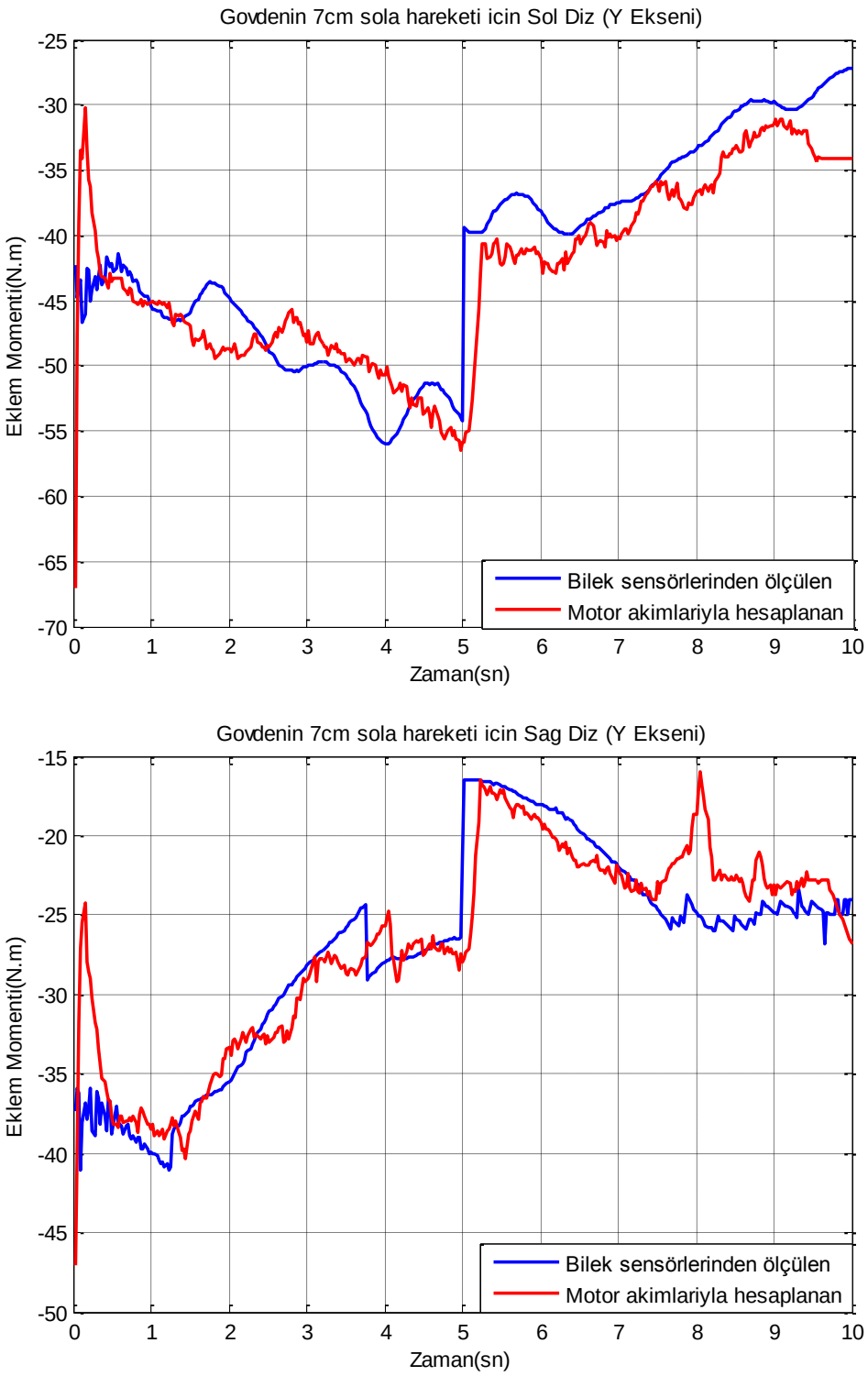
EK B.1



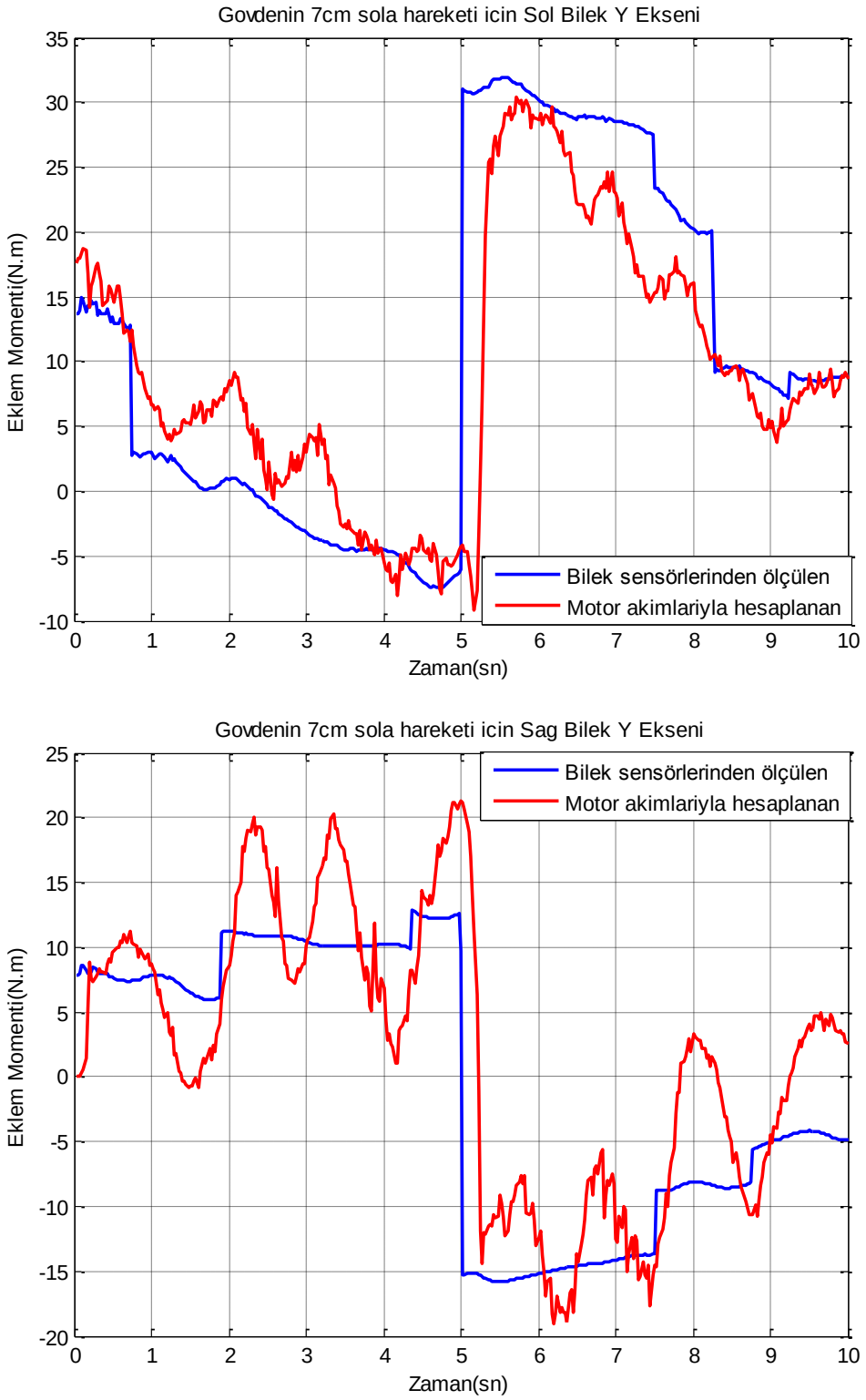
Şekil B.1 : Gövdenin 7 cm sola hareketi için sol ve sağ kalça x ekseni momentleri.



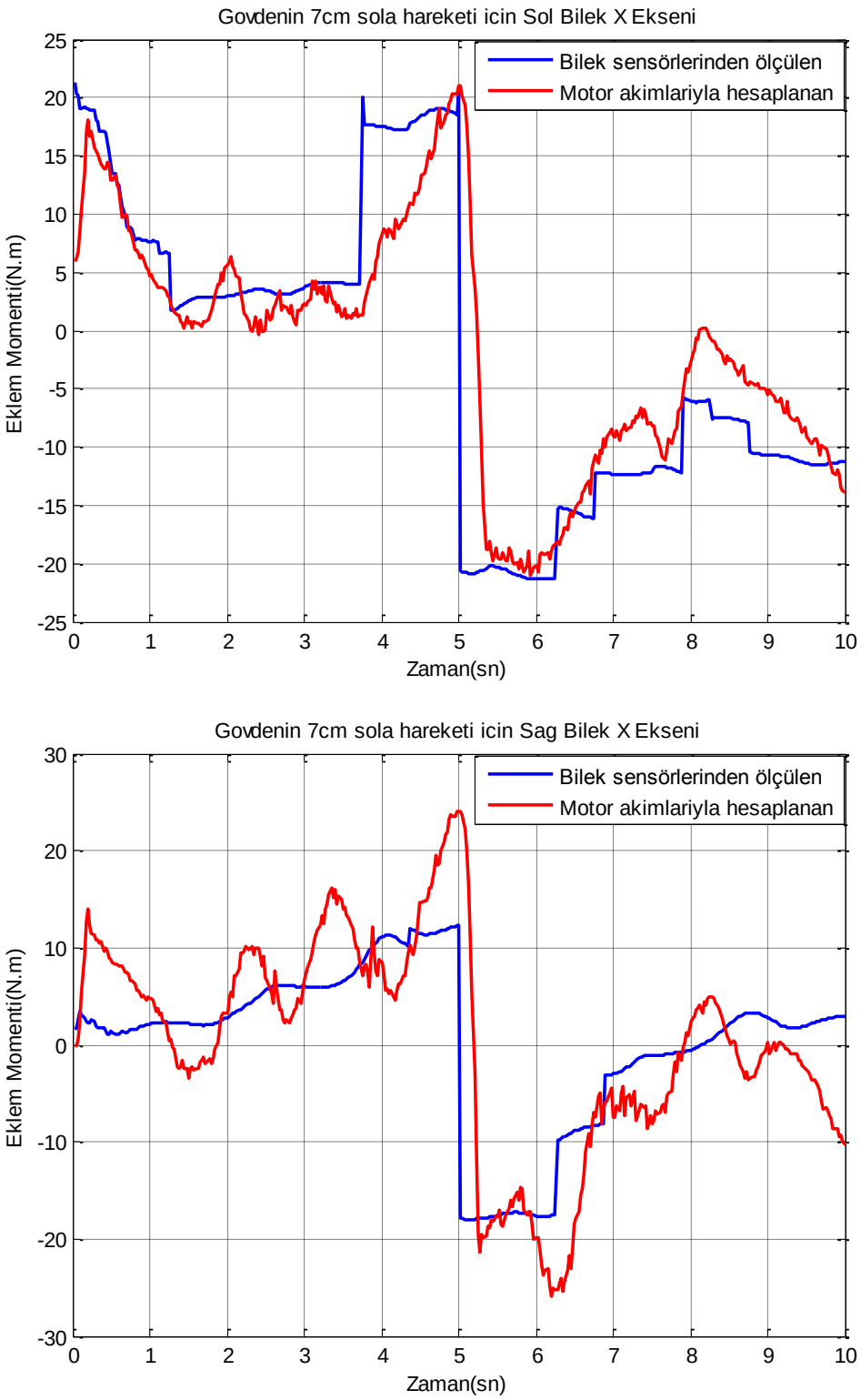
Şekil B.2 : Gövdenin 7 cm sola hareketi için sol ve sağ kalça y ekseni momentleri.



Şekil B.3 : Gövdenin 7 cm sola hareketi için sol ve sağ diz eklemleri momentleri.

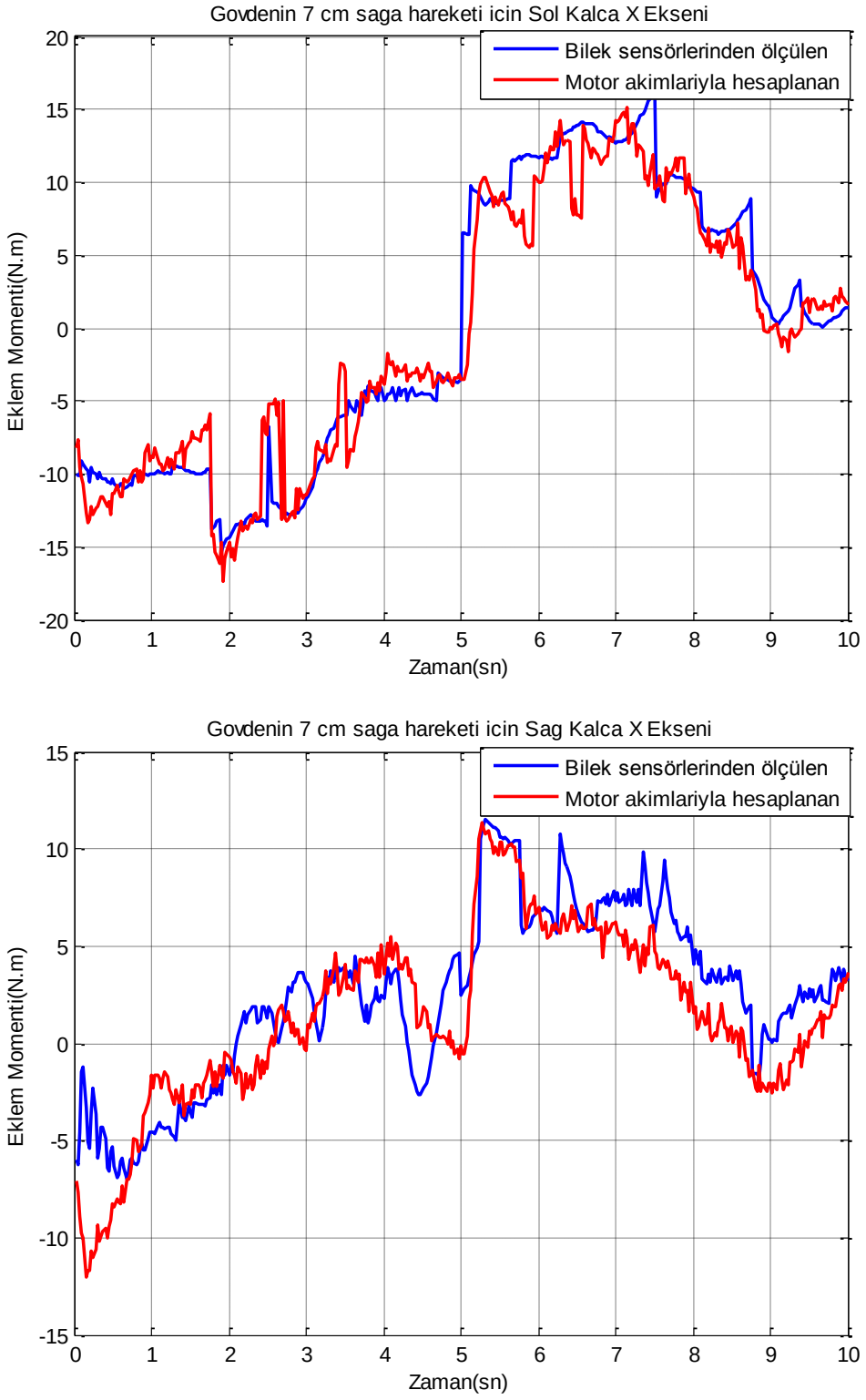


Şekil B.4 : Gövdenin 7 cm sola hareketi için sol ve sağ bilek y ekseni momentleri.

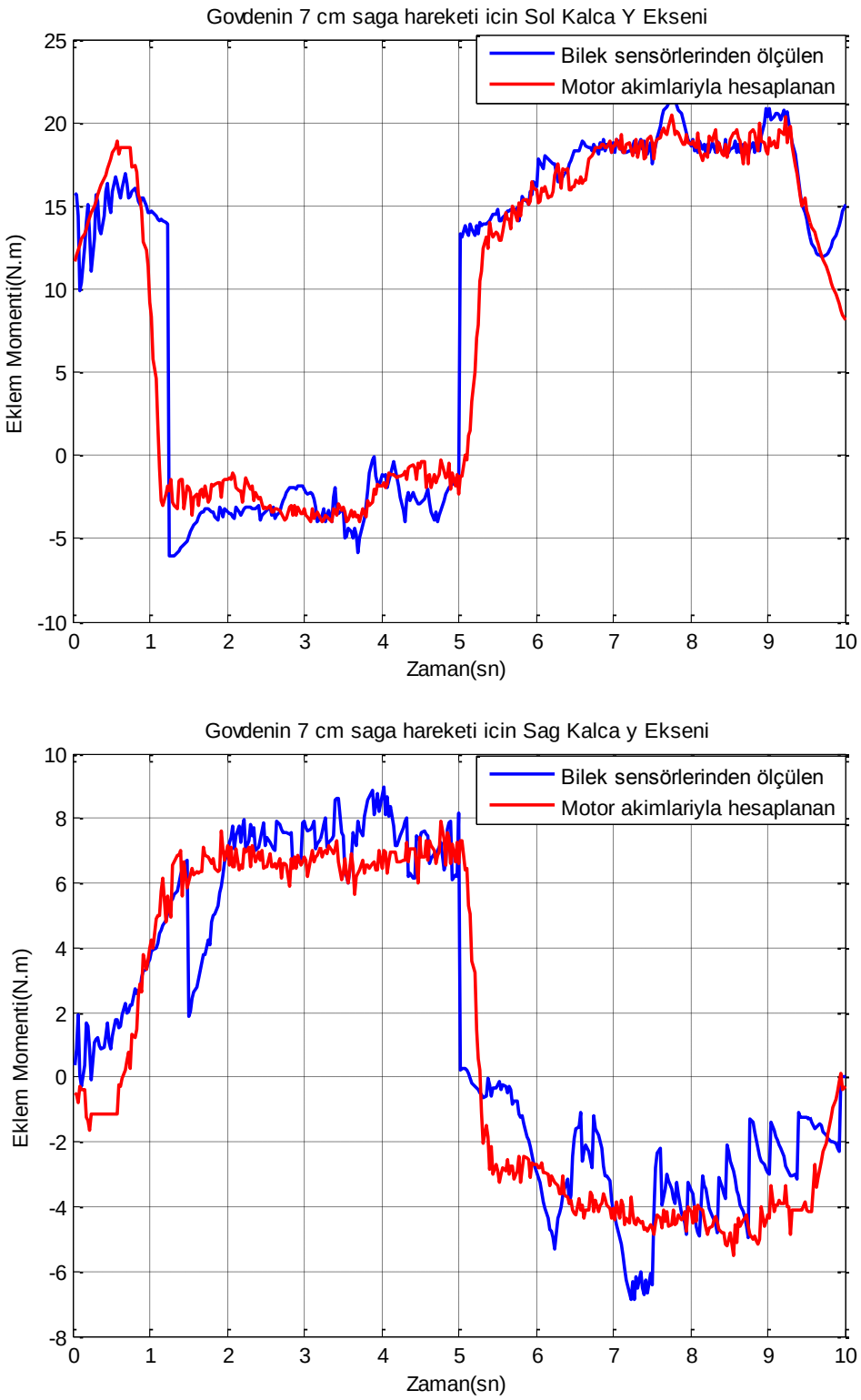


Şekil B.5 : Gövdenin 7 cm sola hareketi için sol ve sağ bilek x ekseni momentleri.

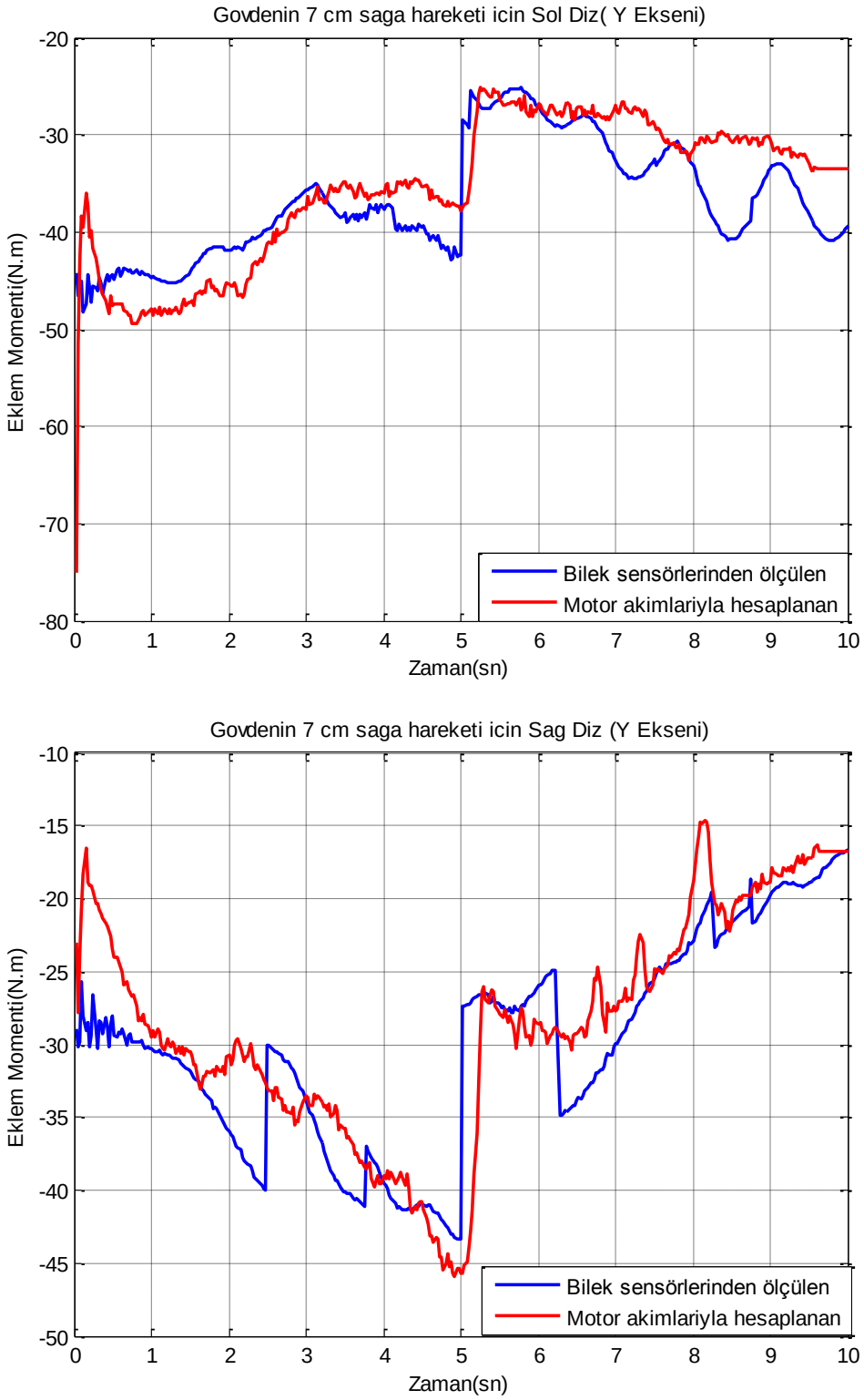
EK B.2



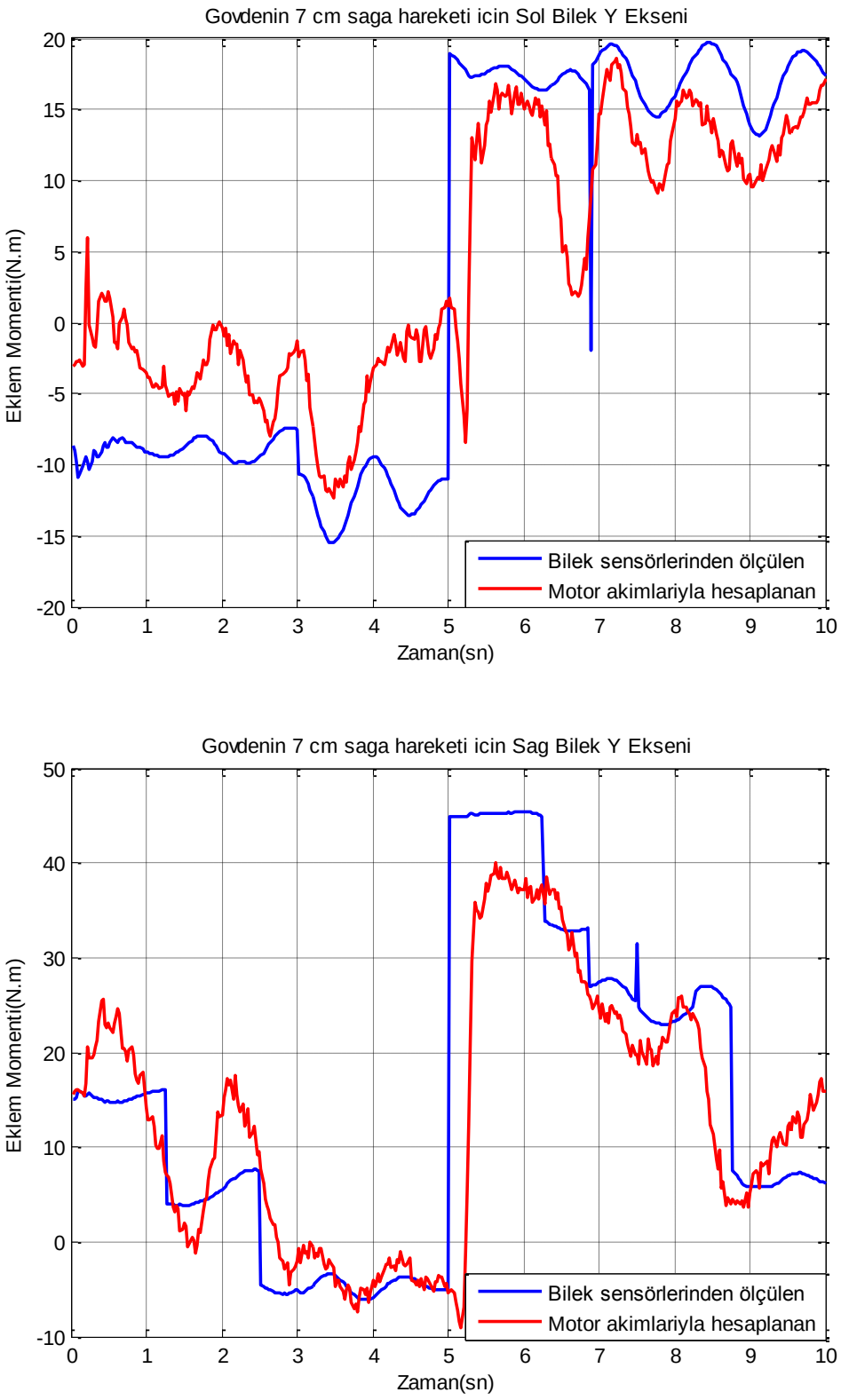
Şekil B.6 : Gövdenin 7 cm sağa hareketi için sol ve sağ kalça x ekseni momentleri.



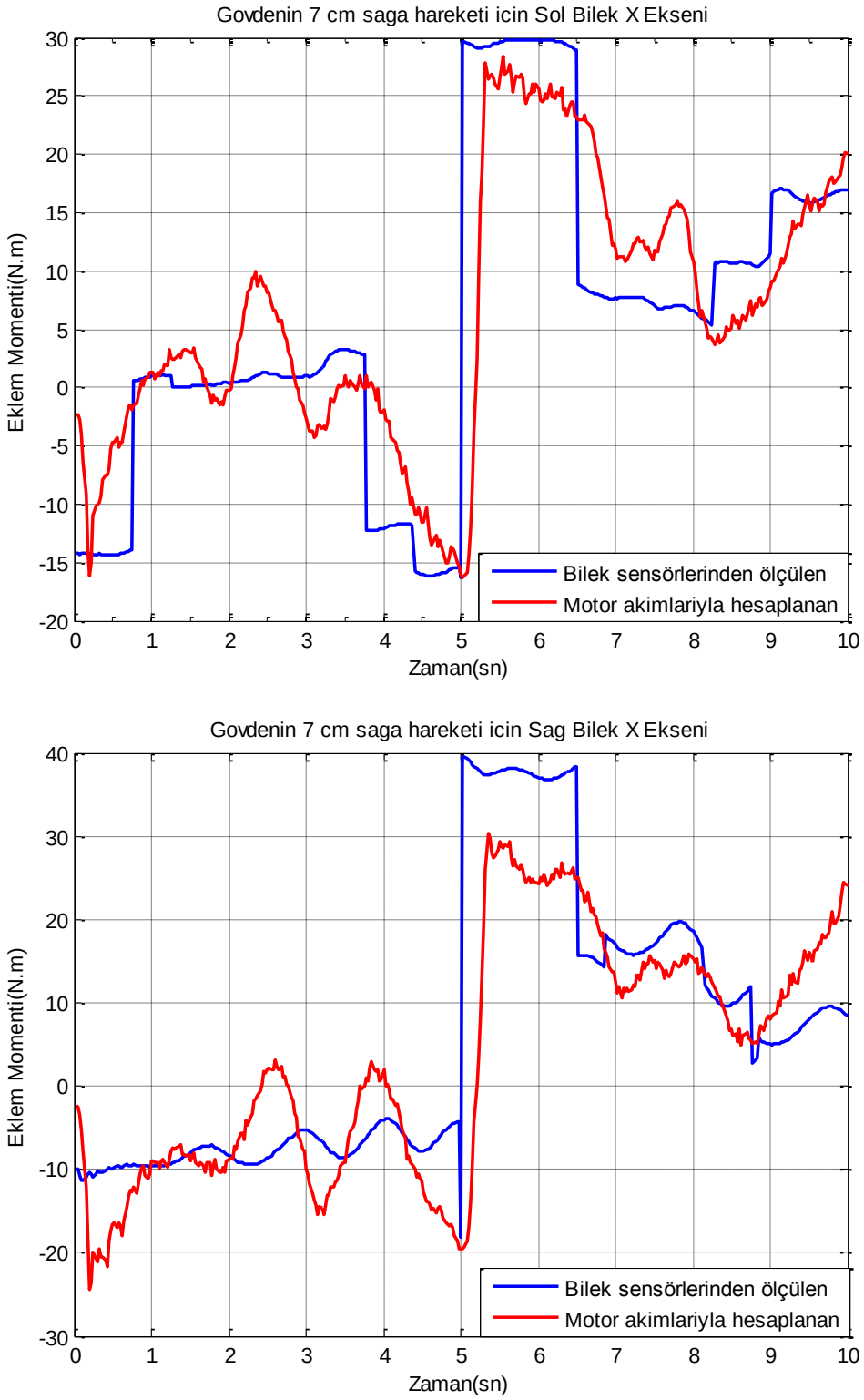
Şekil B.7 : Gövdenin 7 cm sağa hareketi için sol ve sağ kalça y ekseni momentleri.



Şekil B.8 : Gövdenin 7 cm sağa hareketi için sol ve sağ diz eklemi momentleri.

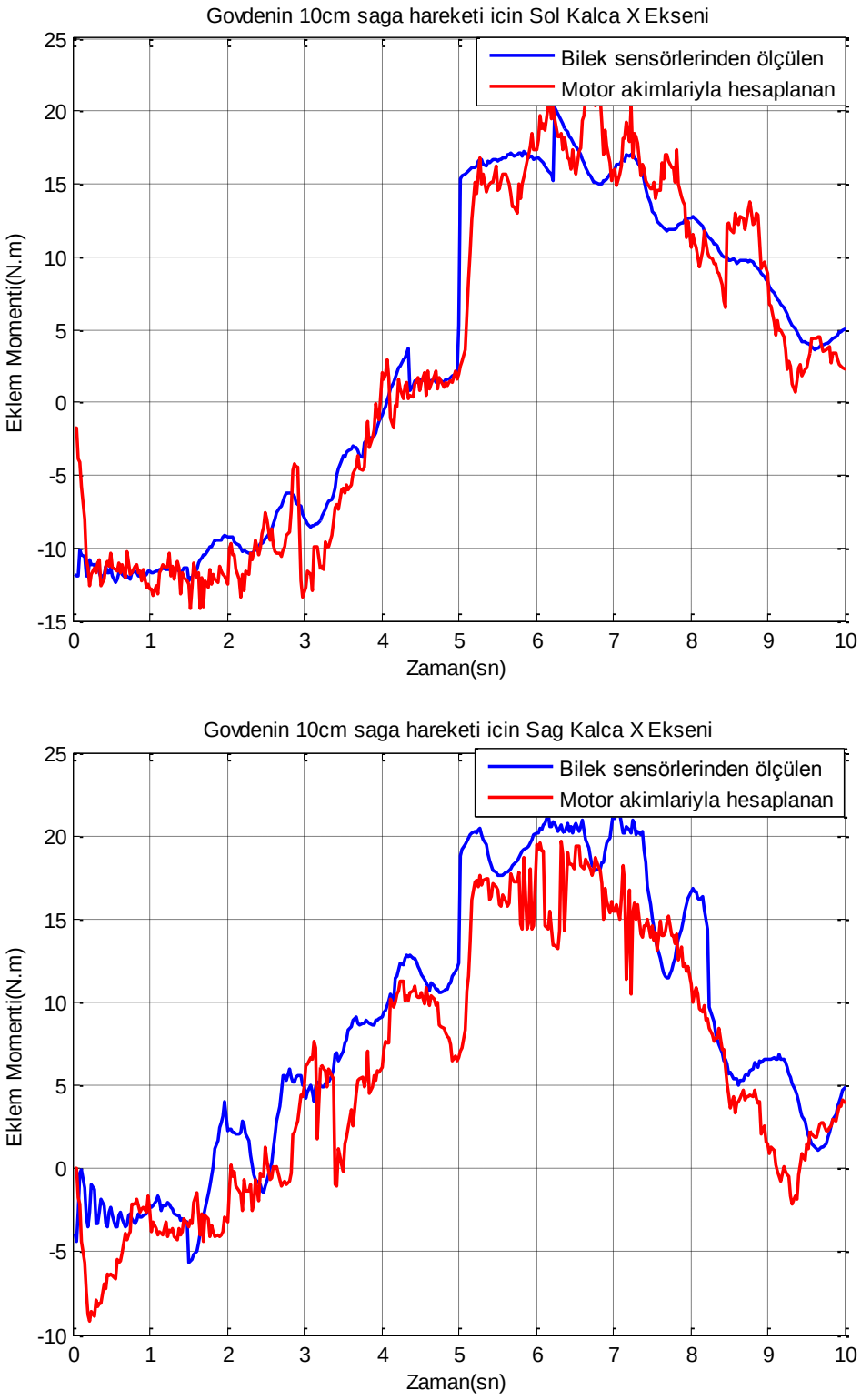


Şekil B.9 : Gövdenin 7 cm sağa hareketi için sol ve sağ bilek y ekseni momentleri.

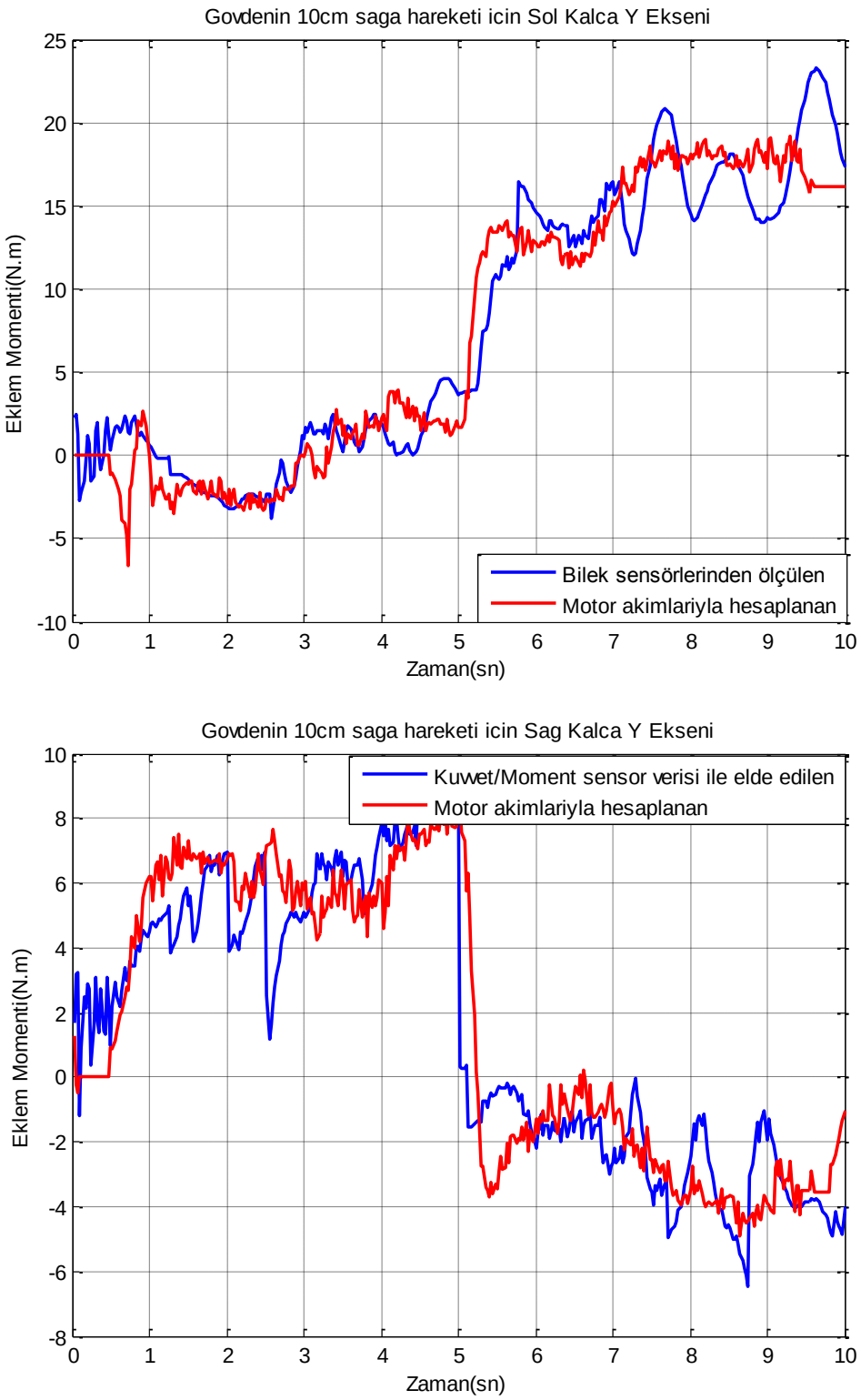


Şekil B.10 : Gövdenin 7 cm sağa hareketi için sol ve sağ bilek x ekseni momentleri.

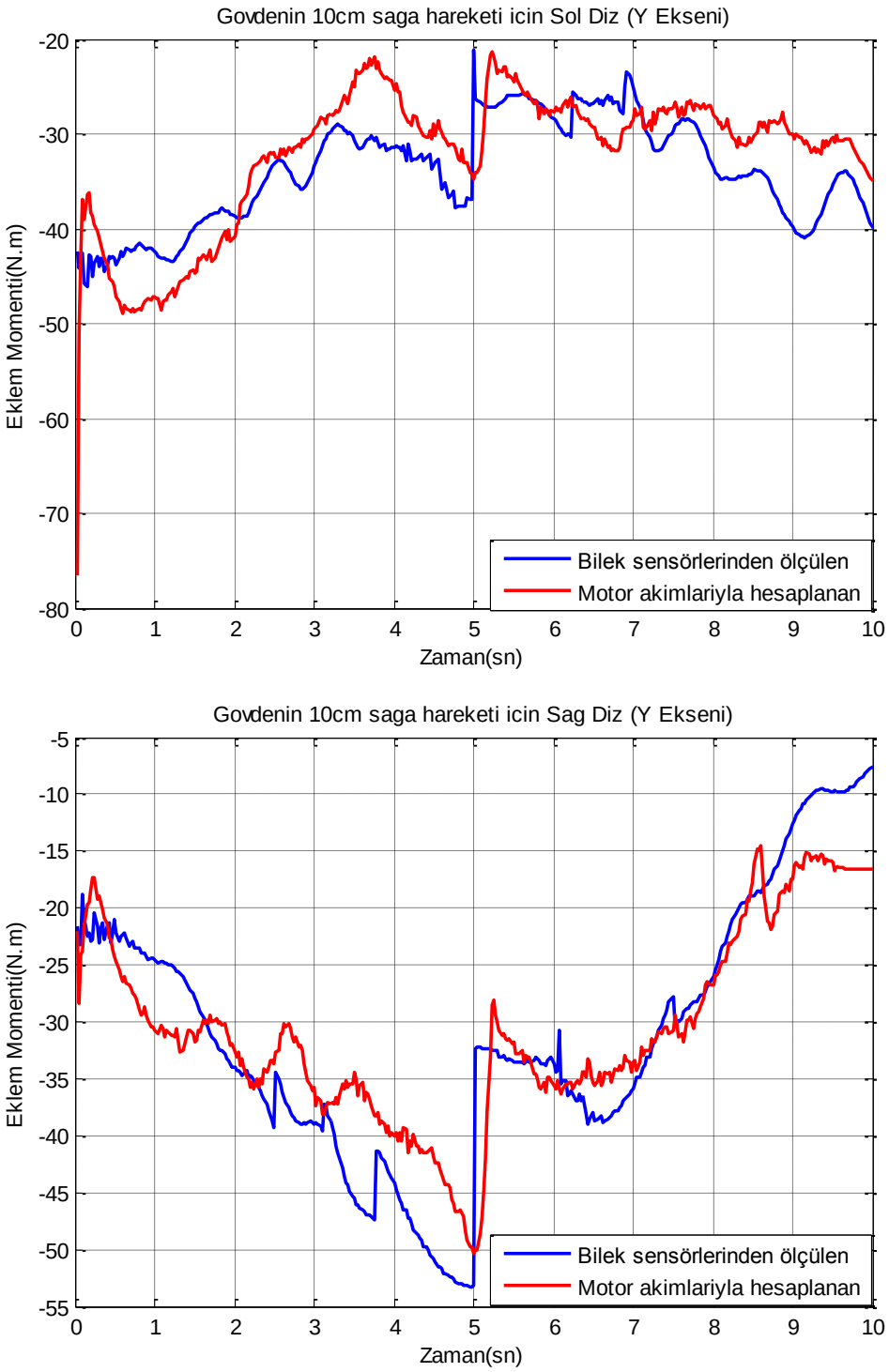
EK B.3



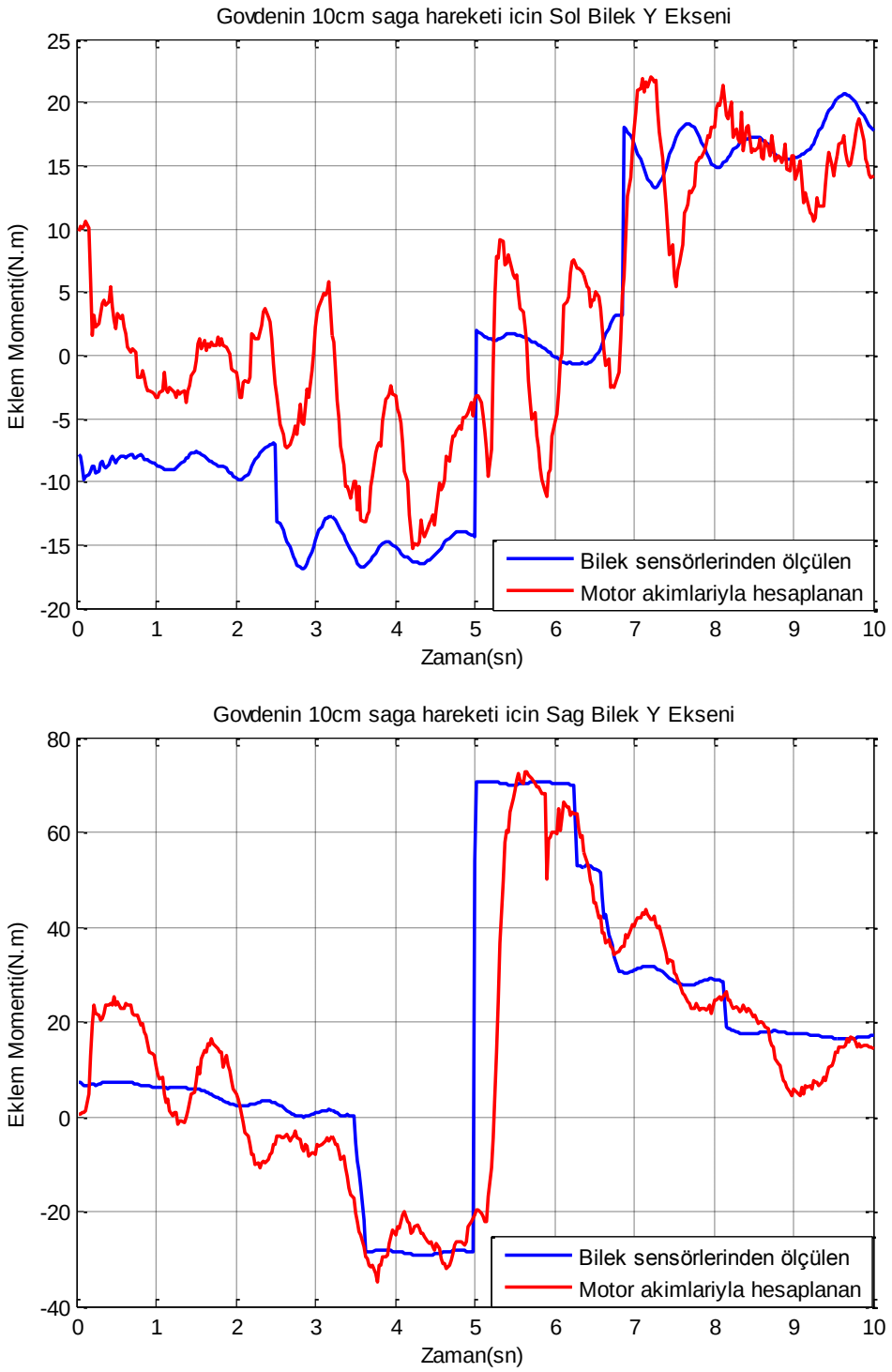
Şekil B.11 : Gövdenin 10 cm sağa hareketi için sol ve sağ kalça x ekseni momentleri.



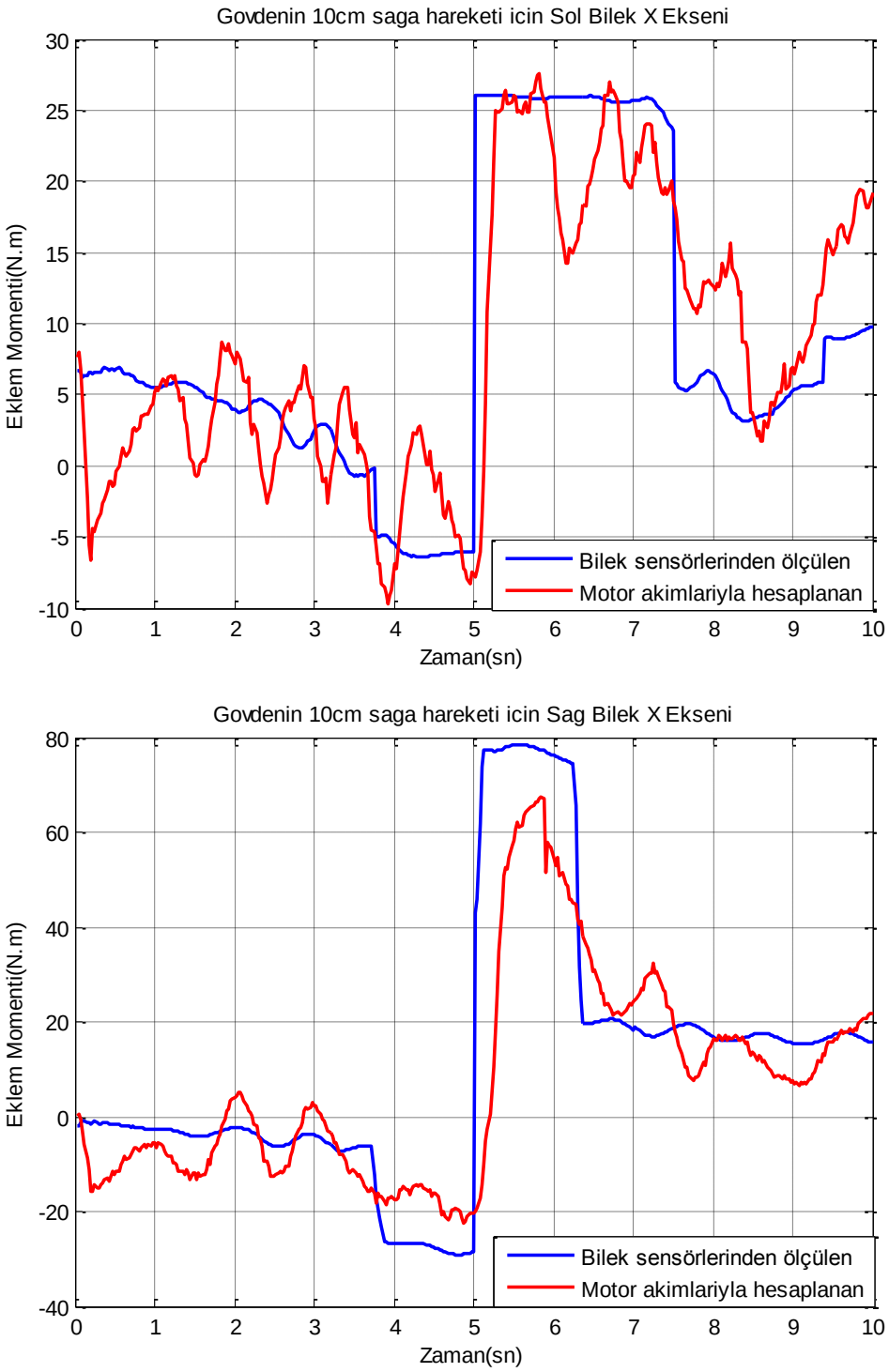
Şekil B.12 : Gövdenin 10 cm sağa hareketi için sol ve sağ kalça y ekseni momentleri.



Şekil B.13 : Gövdenin 10 cm sağa hareketi için sol ve sağ diz eklemi momentleri.

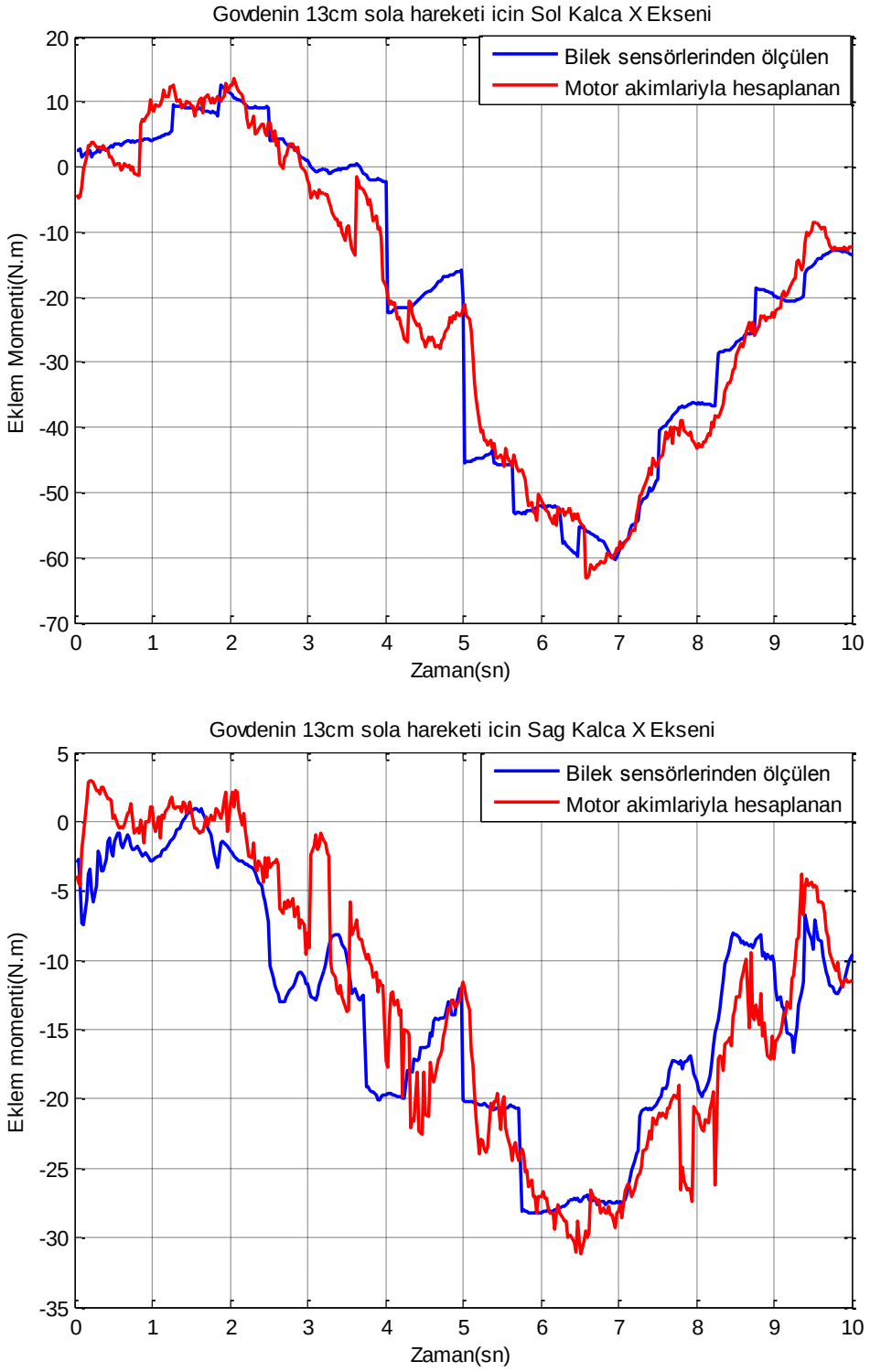


Şekil B.14 : Gövdenin 10 cm sağa hareketi için sol ve sağ bilek y ekseni momentleri.

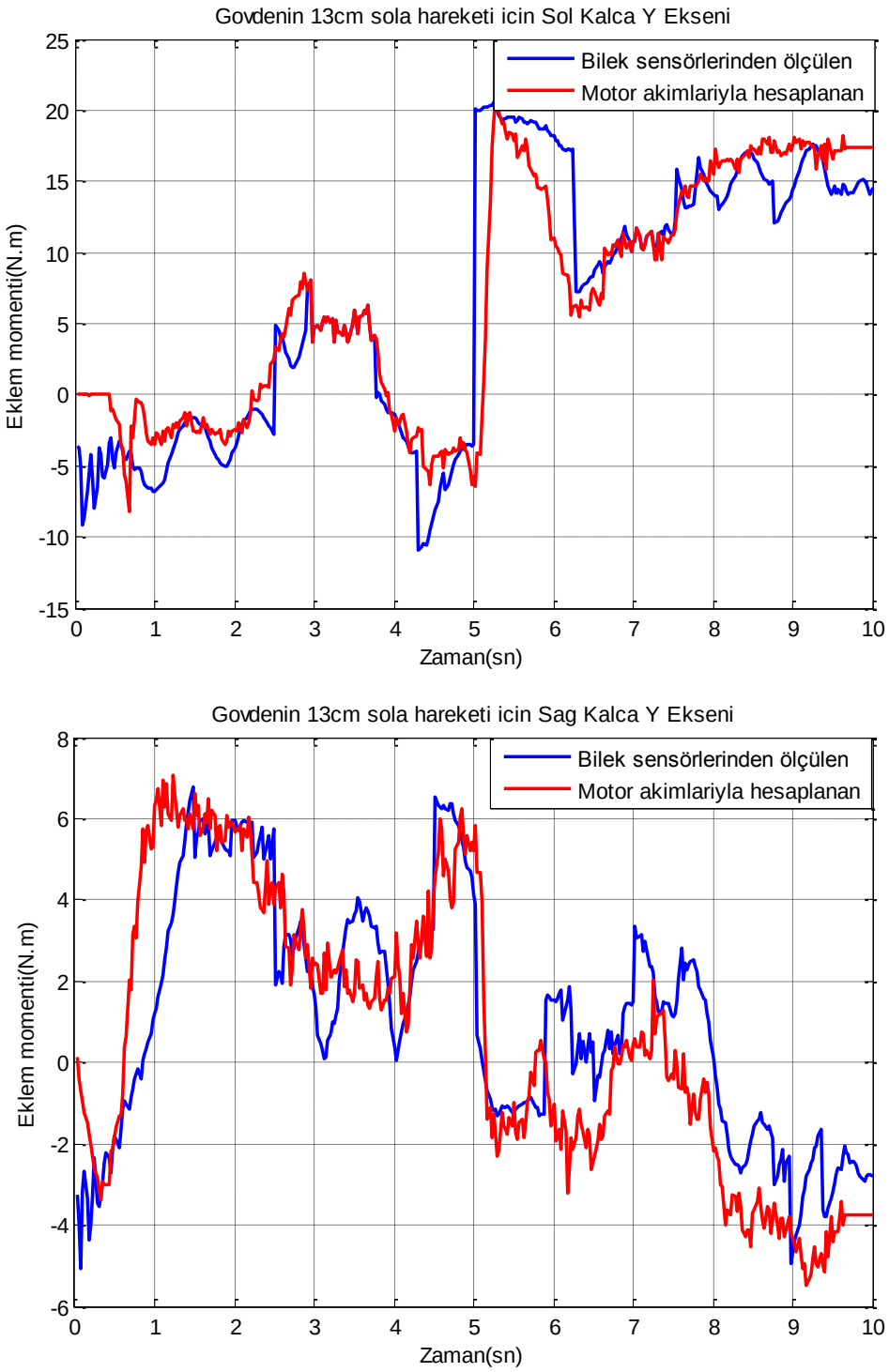


Şekil B.15 : Gövdenin 10 cm sağa hareketi için sol ve sağ bilek x ekseni momentleri.

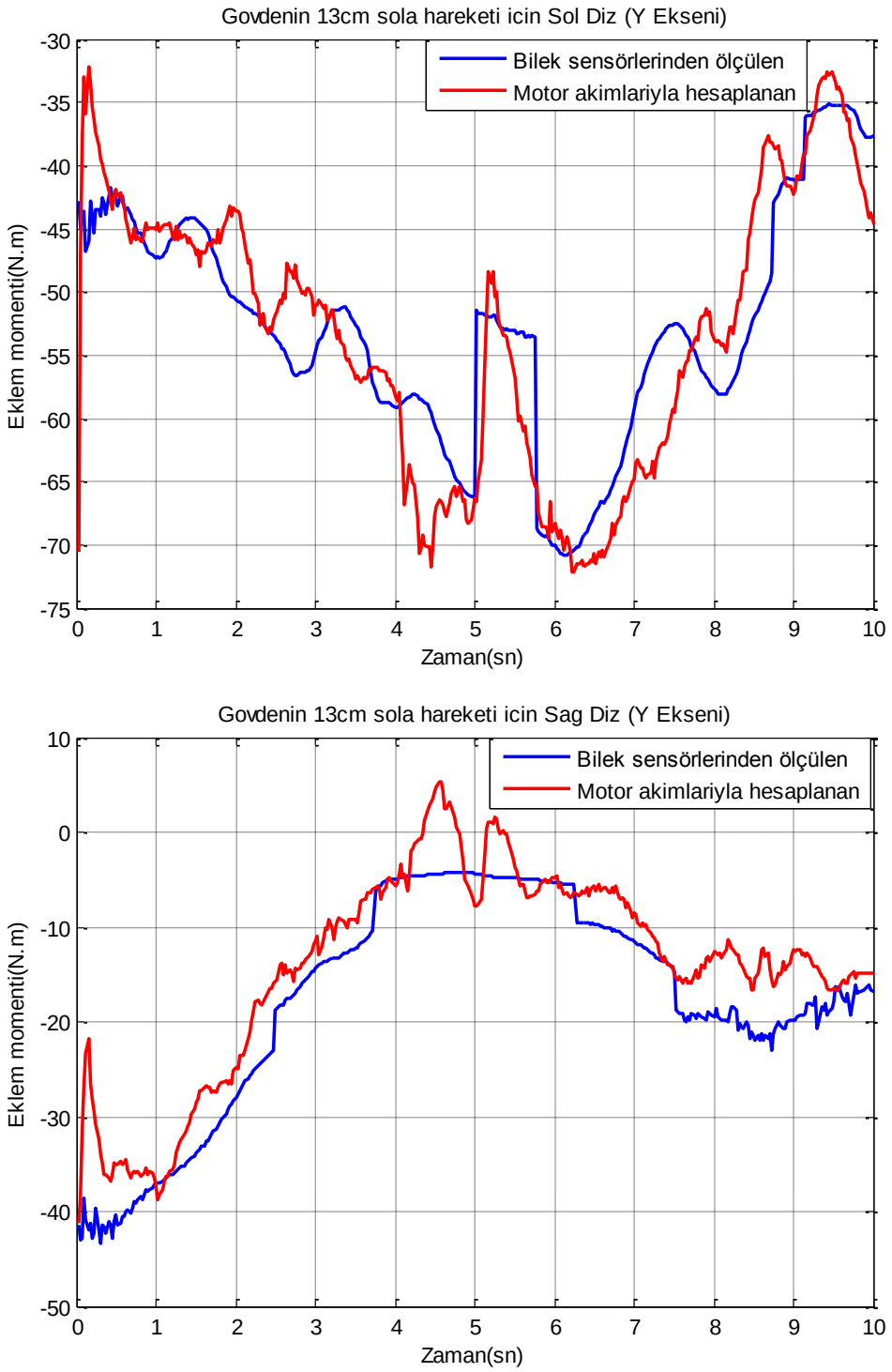
EK B.4



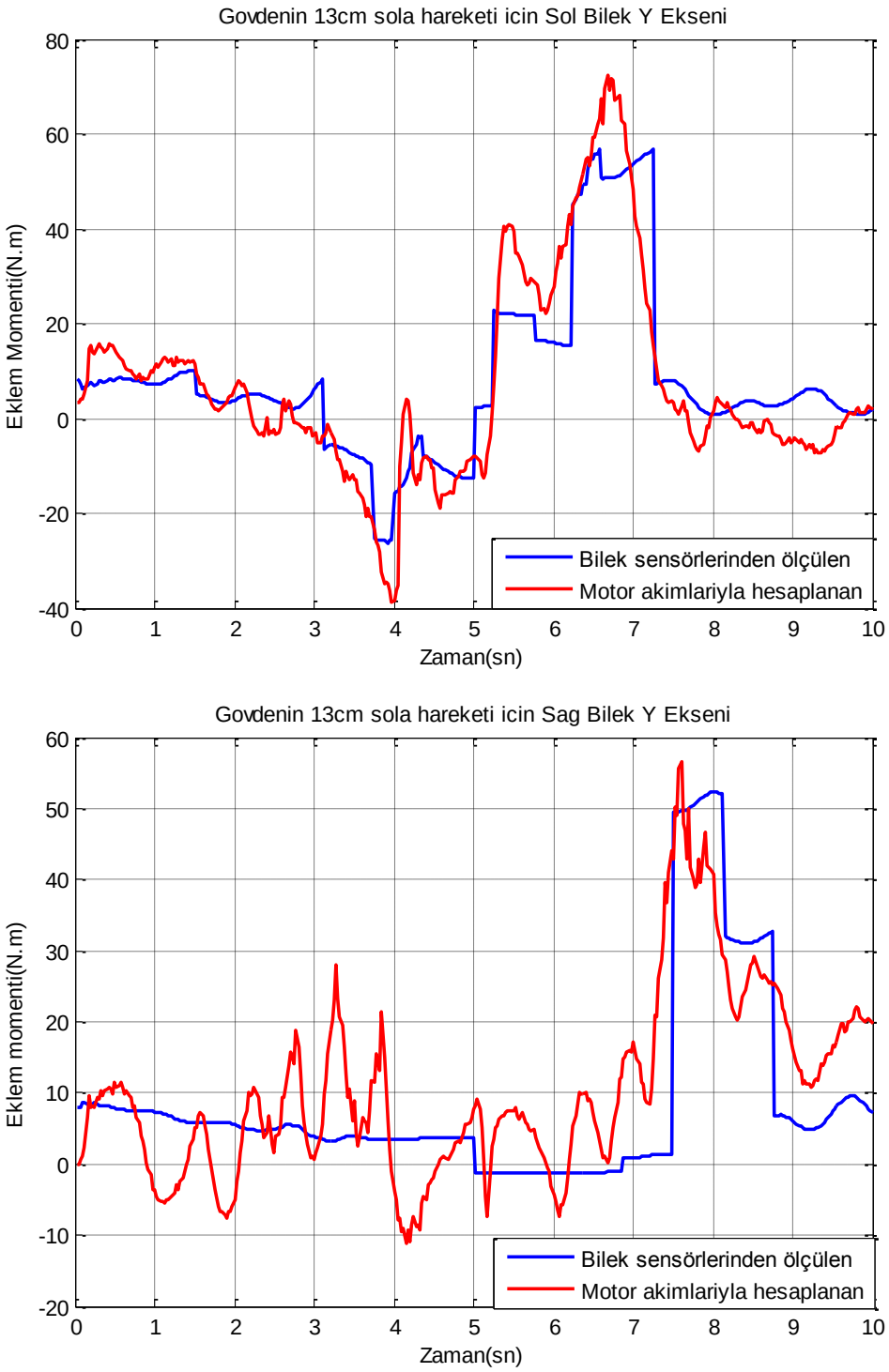
Şekil B.16 : Gövdenin 13 cm sola hareketi için sol ve sağ kalça x ekseni momentleri.



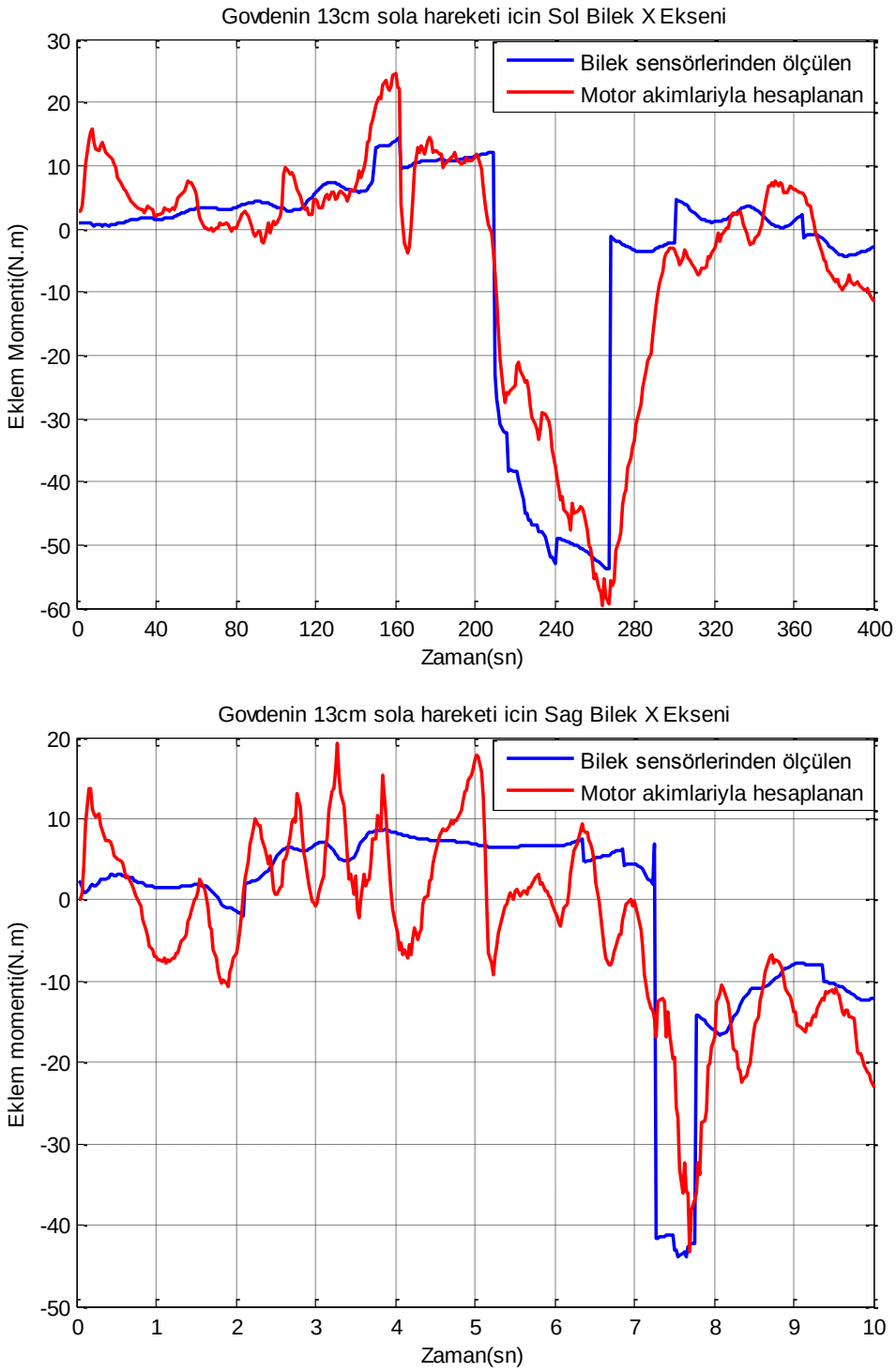
Şekil B.17 : Gövdenin 13 cm sola hareketi için sol ve sağ kalça y ekseni momentleri.



Şekil B.18 : Gövdenin 13 cm sola hareketi için sol ve sağ diz eklemi momentleri.

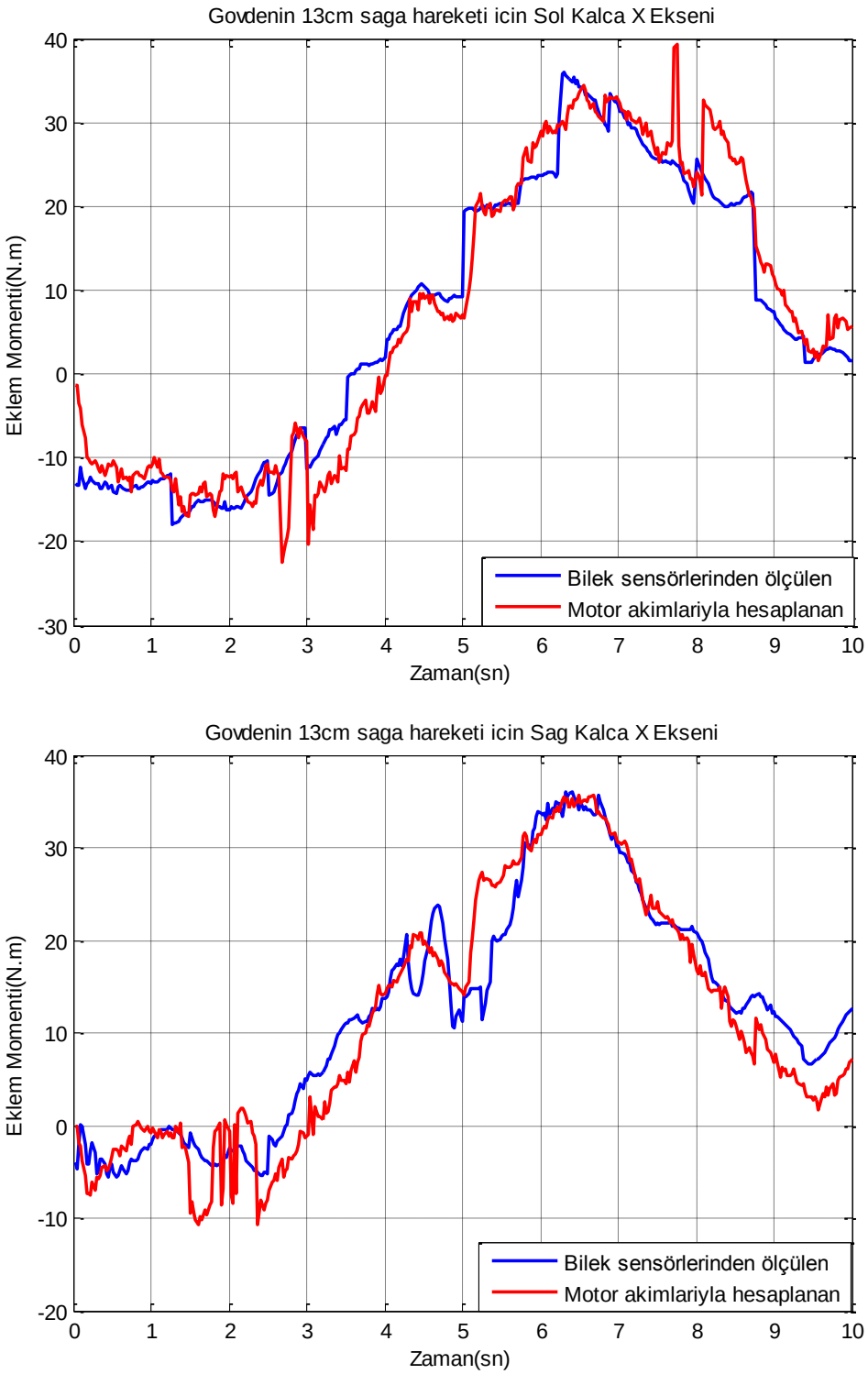


Şekil B.19 : Gövdenin 13 cm sola hareketi için sol ve sağ bilek y ekseni momentleri.

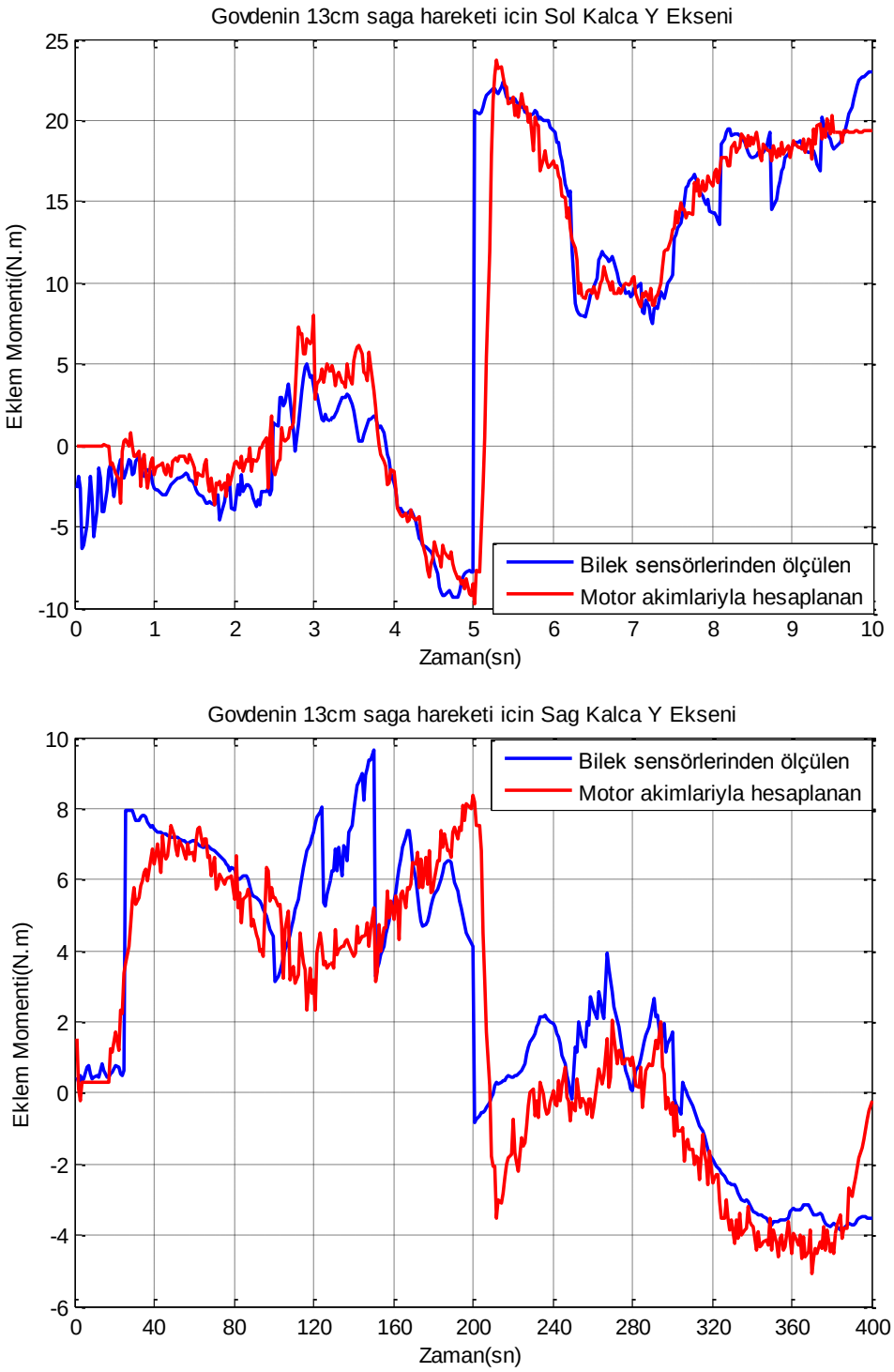


Şekil B.20 : Gövdenin 13 cm sola hareketi için sol ve sağ bilek x ekseni momentleri.

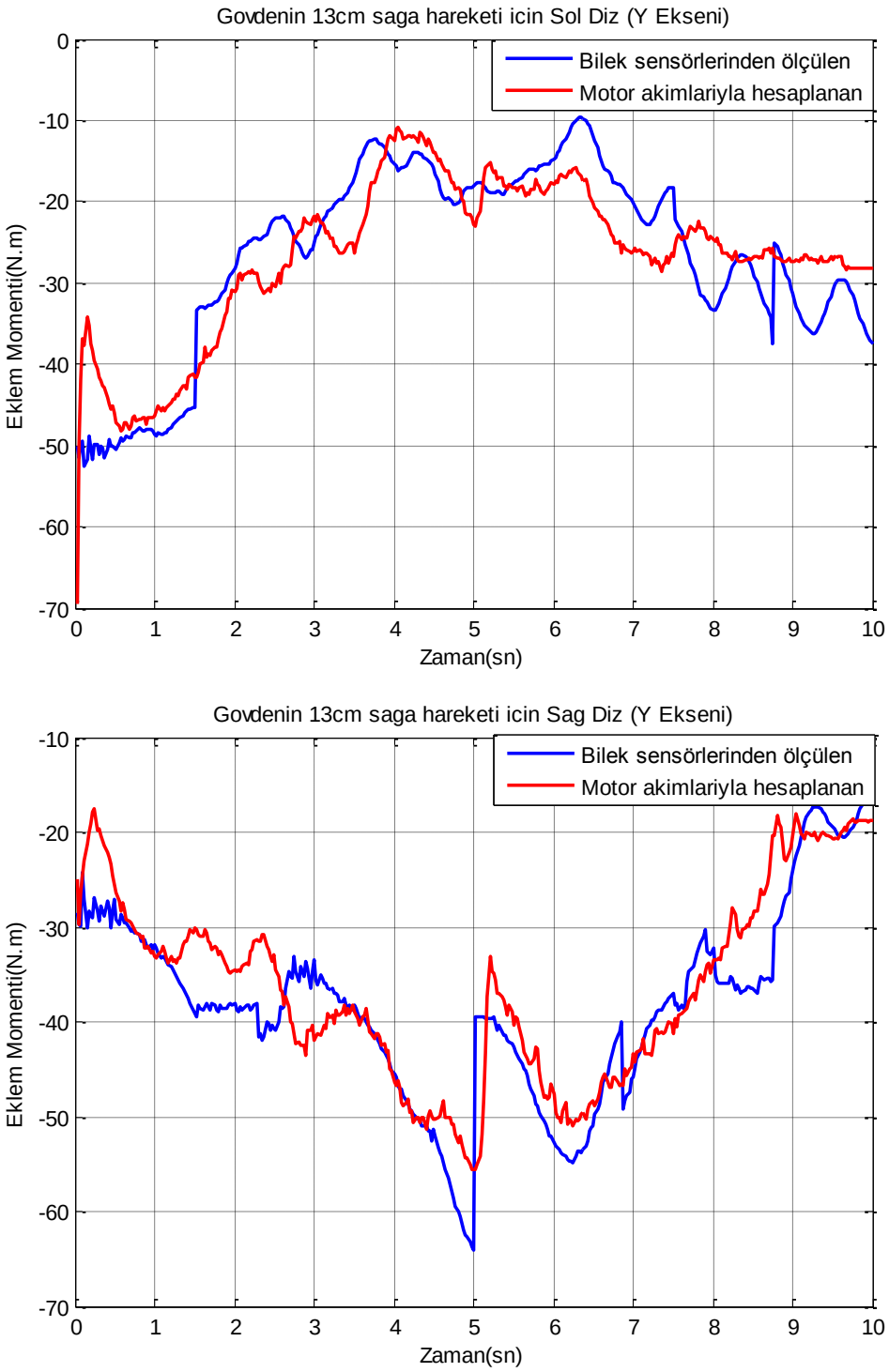
EK B.5



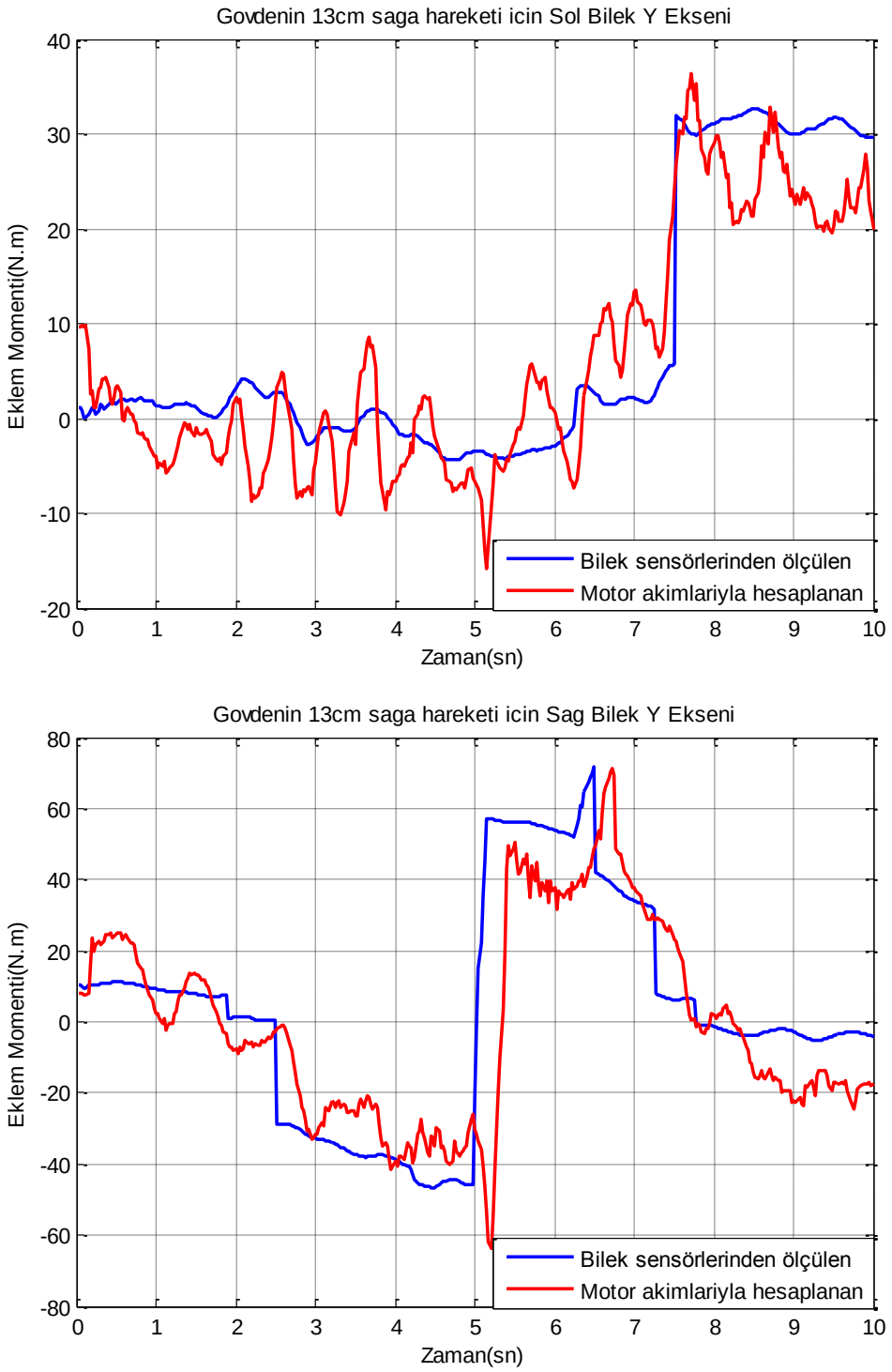
Şekil B.21 : Gövdenin 13 cm sağa hareketi için sol ve sağ kalça x ekseni momentleri.



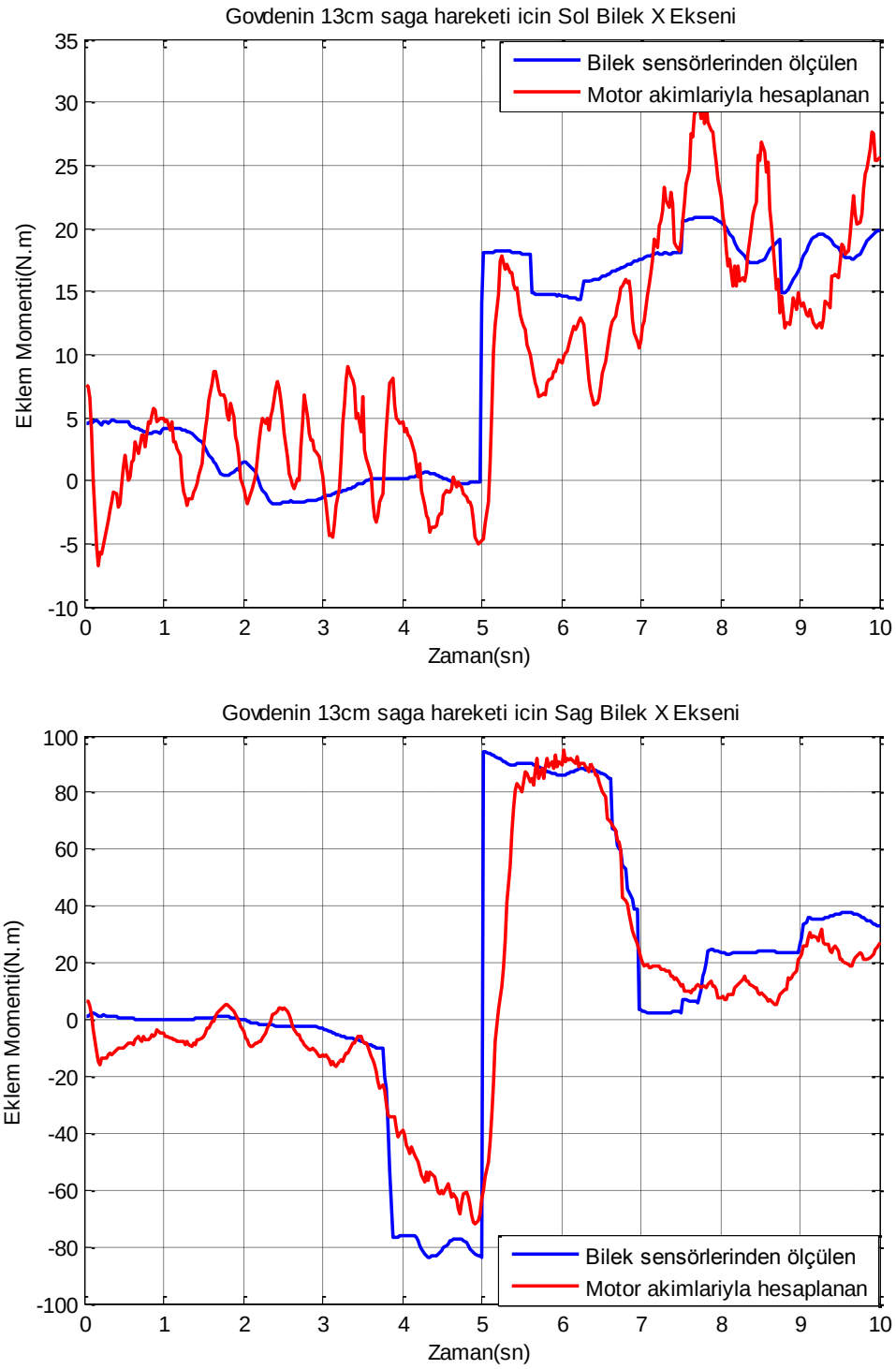
Şekil B.22 : Gövdenin 13 cm sağa hareketi için sol ve sağ kalça y ekseni momentleri.



Şekil B.23 : Gövdenin 13 cm sağa hareketi için sol ve sağ diz eklemi momentleri.

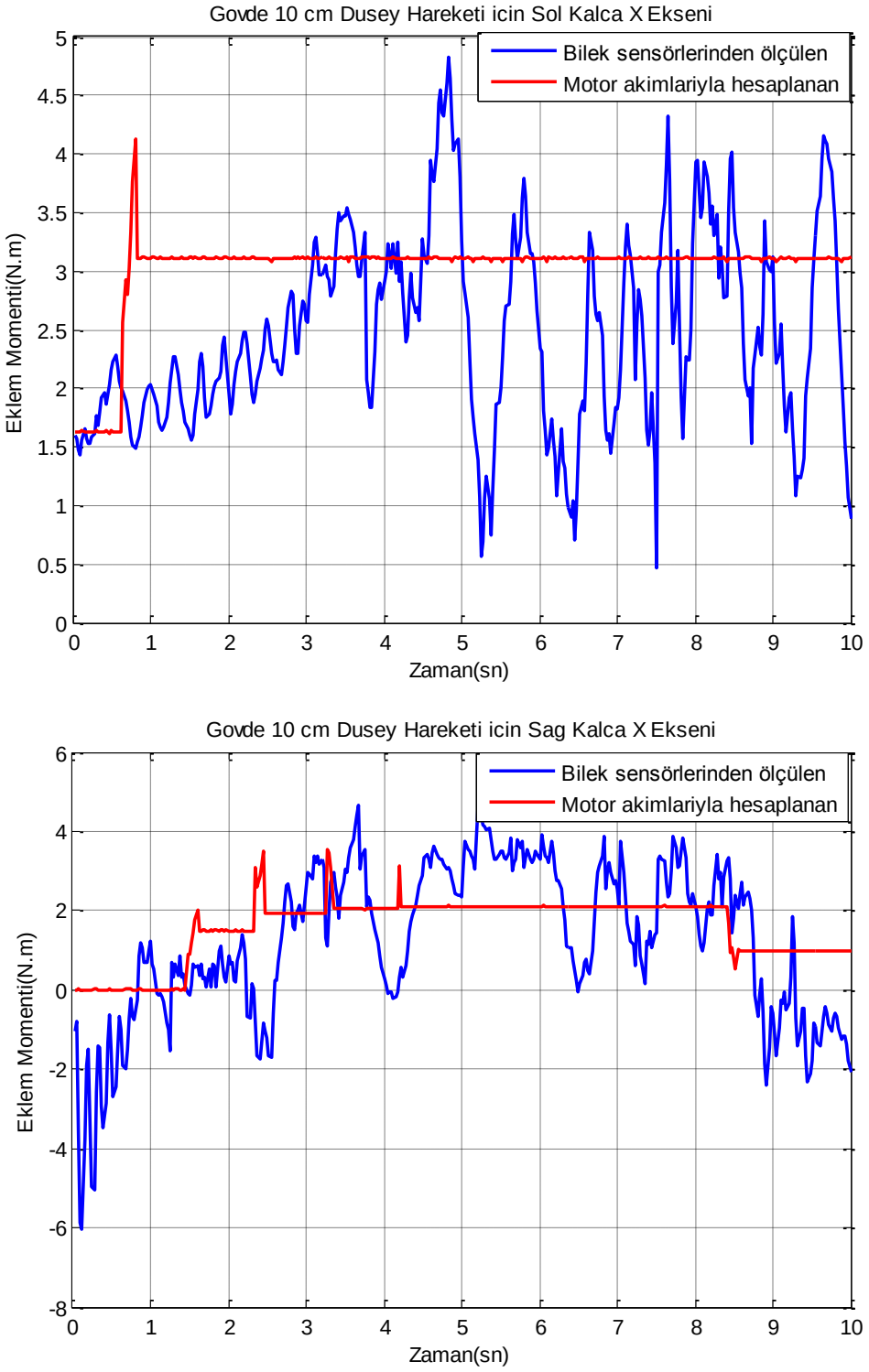


Şekil B.24 : Gövdenin 13 cm sağa hareketi için sol ve sağ bilek y ekseni momentleri.

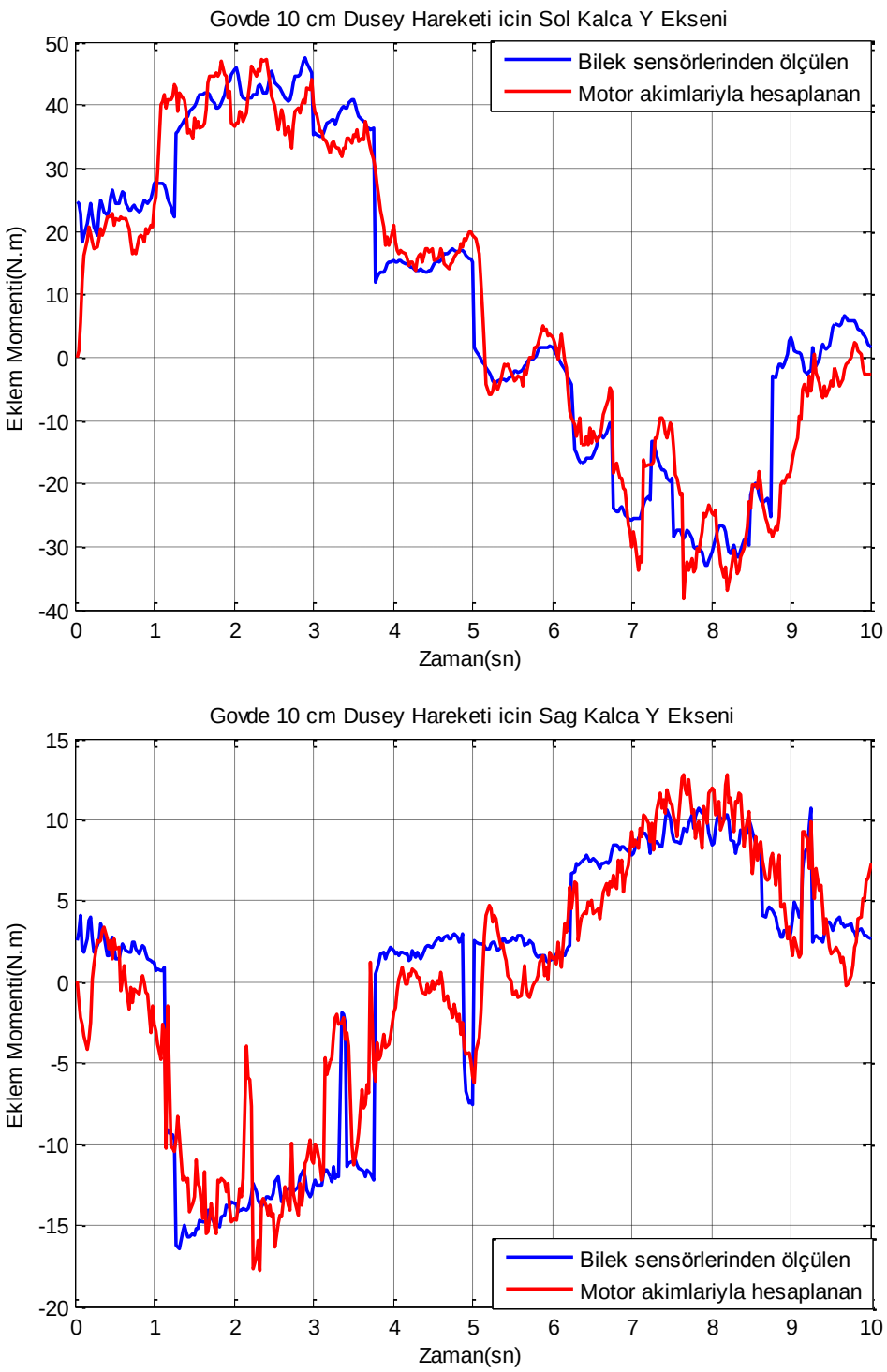


Şekil B.25 : Gövdenin 13 cm sağa hareketi için sol ve sağ bilek x ekseni momentleri.

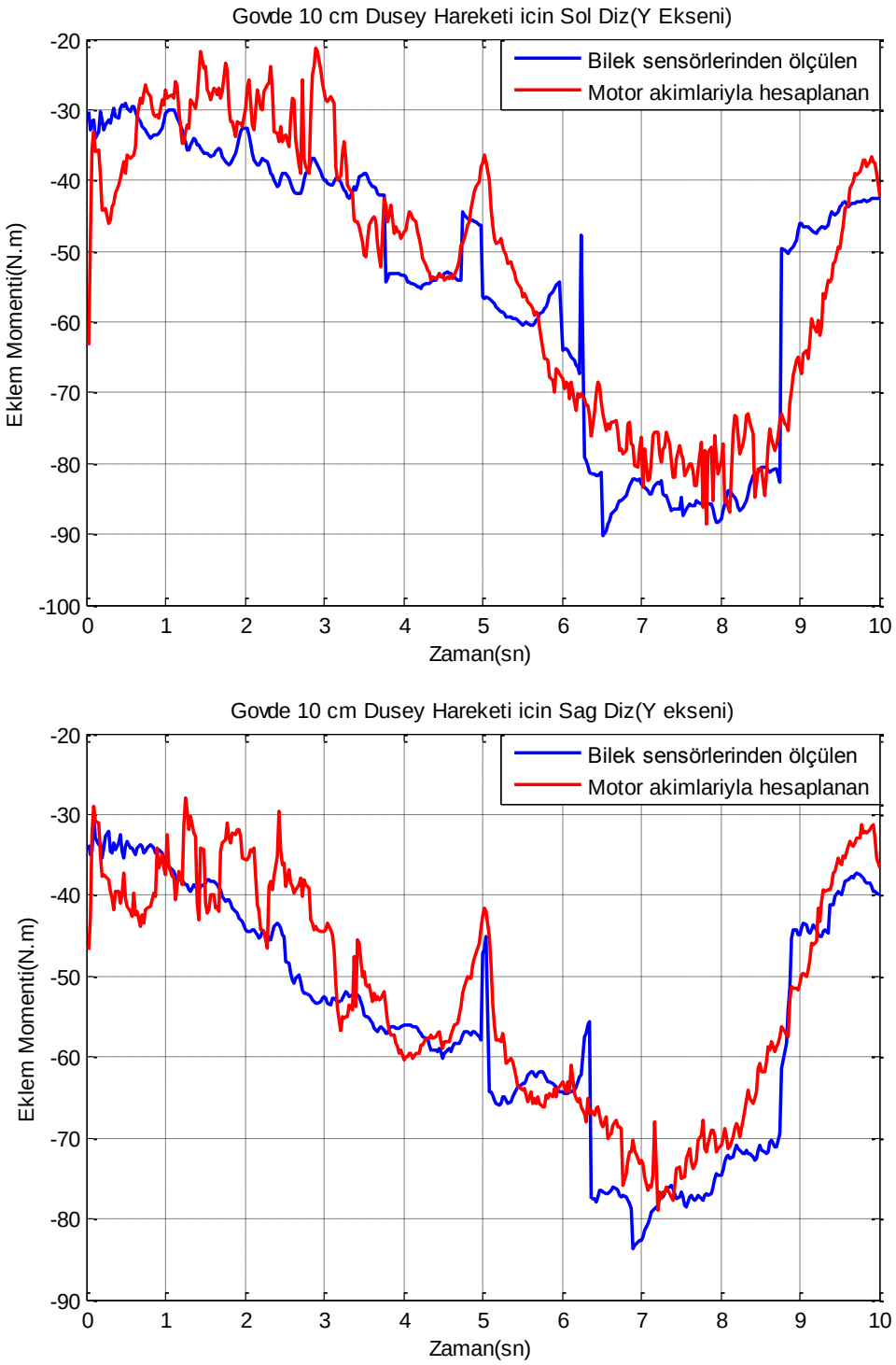
EK B.6



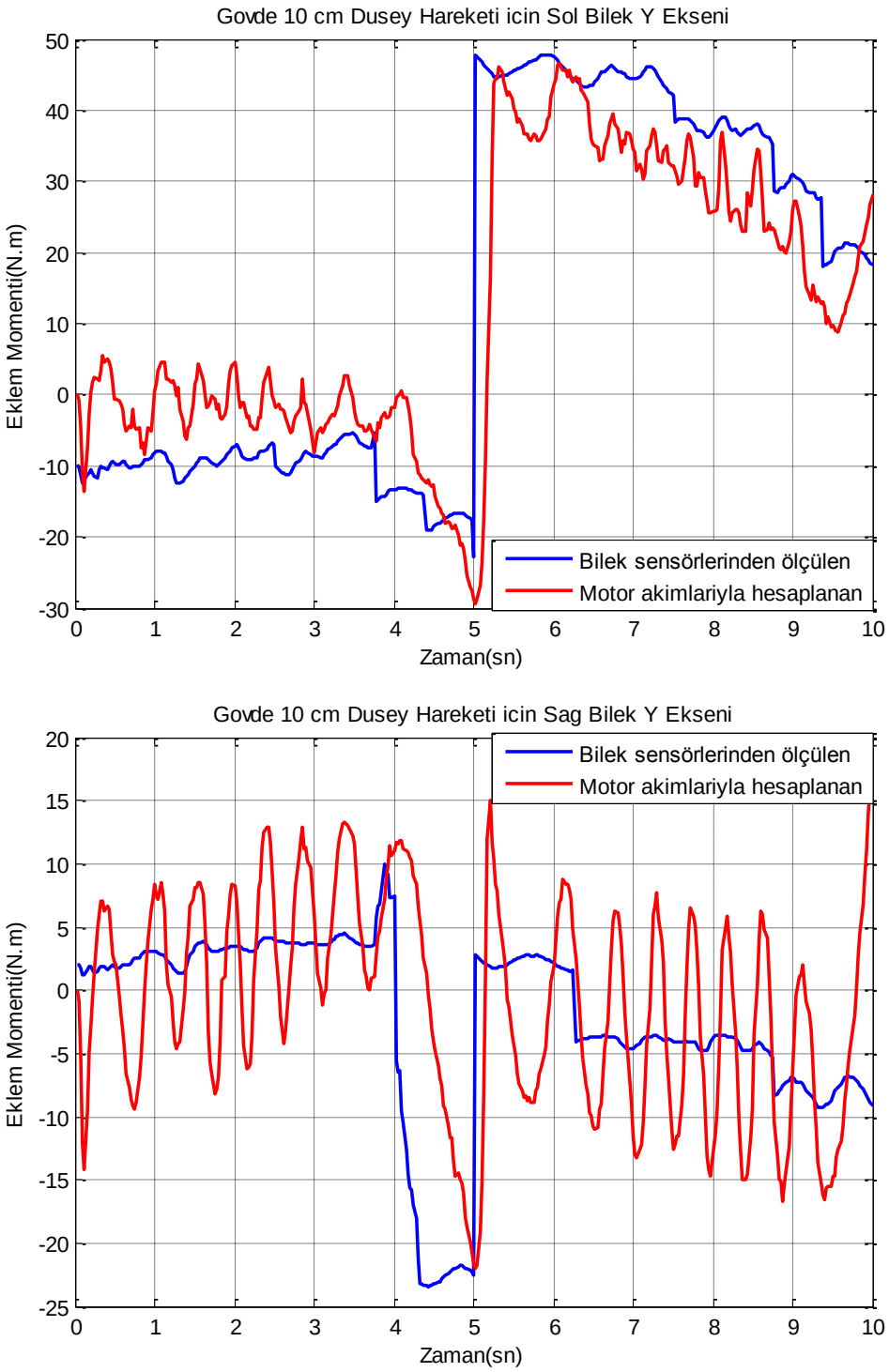
Şekil B.26 : Gövdenin 10 cm düşey hareketi için sol ve sağ kalça x ekseni momentleri.



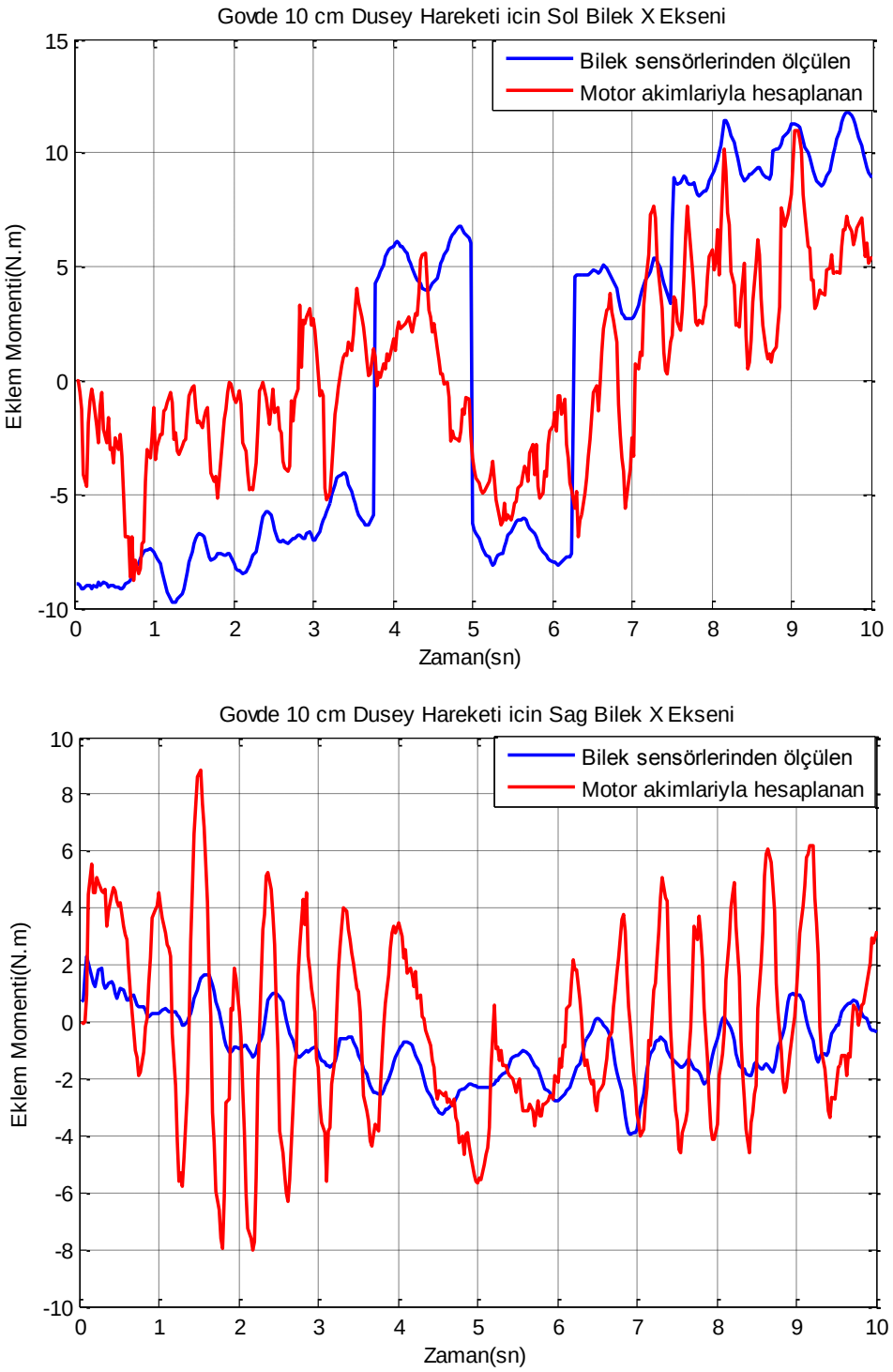
Şekil B.27 : Gövdenin 10 cm düşey hareketi için sol ve sağ kalça y ekseni momentleri.



Şekil B.28 : Gövdenin 10 cm düşey hareketi için sol ve sağ diz eklemi momentleri.



Şekil B.29 : Gövdenin 10 cm düşey hareketi için sol ve sağ bilek y ekseni momentleri.



Şekil B.30 : Gövdenin 10 cm düşey hareketi için sol ve sağ bilek x ekseni momentleri.

ÖZGEÇMİŞ

Ad Soyad: Numan Mert Tan

Doğum Yeri ve Tarihi: 15.04.1987 / Eminönü-İstanbul

Adres: Nişancı Sk. Bahar Apt. No: 4/12 Kızıltoprak/İstanbul

E-Posta: nmert.tan@gmail.com

Lisans: İstanbul Teknik Üniversitesi Makina Mühendisliği