ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

NONLINEAR CONTROL METHODS OF INDUSTRIAL SERIAL ROBOTS

M.Sc. THESIS

Günay YILDIZ

Department of Control and Automation Engineering

Control and Automation Engineering Programme

JANUARY 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE
ENGINEERING AND TECHNOLOGY

NONLINEAR CONTROL METHODS OF INDUSTRIAL SERIAL ROBOTS

M.Sc. THESIS

Günay YILDIZ
(504111116)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Sıddık Murat YEŞİLOĞLU

JANUARY 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL SERİ ROBOTLARIN DOĞRUSAL OLMAYAN KONTROLÜ

YÜKSEK LİSANS TEZİ

Günay YILDIZ
(504111116)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Sıddık Murat YEŞİLOĞLU

OCAK 2014

Günay YILDIZ, a **M.Sc.** student of ITU **Graduate School of Science Engineering and Technology** student ID 504111116, successfully defended the **thesis** entitled "**NONLINEAR CONTROL METHODS OF INDUSTRIAL SERIAL ROBOTS**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Asst. Prof. Dr. Sıddık Murat YEŞİLOĞLU** ....
Istanbul Technical University

Jury Members : **Prof. Dr. Hakan TEMELTAŞ** ....
Istanbul Technical University

Asst. Prof. Dr. Aydemir ARISOY ....
Air Force Academy

Date of Submission : 16 December 2013
Date of Defense : 22 January 2014

FOREWORD

I would like to thank to my advisor Asst. Prof. Dr. Sıddık Murat Yeşiloğlu who has given me a different point of view in the field of robotics and control and has guided me with his valuable information. And I also want to thank to my family for their great love, patience, moral and encouragement during all stages of my educational life.

January 2014 Günay YILDIZ
M.Sc. Control and Automation Eng.

TABLE OF CONTENTS

Page

FOREWORD .... **vii**
TABLE OF CONTENTS .... **ix**
ABBREVIATIONS .... **xi**
LIST OF TABLES .... **xiii**
LIST OF FIGURES .... **xv**
LIST OF SYMBOLS .... **xvii**
SUMMARY .... **xix**
ÖZET .... **xxi**
1. INTRODUCTION **1**
  1.1 Purpose of Thesis 2
  1.2 Literature Review 3
  1.3 Motivation 5
2. RIGID BODY MOTION **7**
  2.1 Rotational Motion in $\mathbb{R}^3$ 8
  2.2 Exponential Coordinates for Rotation 9
  2.3 Rigid Motion in $\mathbb{R}^3$ 11
  2.4 Exponential Coordinates for Rigid Motion 12
3. KINEMATIC AND DYNAMIC MODELLING USING SPATIAL OPERATOR ALGEBRA **17**
  3.1 Spatial Operator Algebra (SOA) 17
  3.2 Serial Manipulator Kinematics on a Fixed Platform 19
  3.3 Serial Manipulator Dynamics on a Fixed Platform 24
4. COMPUTED-TORQUE CONTROL OF ROBOTIC MANIPULATORS **29**
  4.1 Introduction 29
  4.2 Path Generation 30
  4.3 Computed-Torque Control 31
    4.3.1 Derivation of inner feedforward loop 31
    4.3.2 PD outer-loop design 33
  4.4 Stability of the Computed Torque Control Law 35
5. ADAPTIVE CONTROL OF ROBOTIC MANIPULATORS **37**
  5.1 Introduction 37
  5.2 Adaptive Control by a Computed-Torque Approach 37
  5.3 Adaptive Control Summary 42
    5.3.1 Torque controller 42
    5.3.2 Update rule 42
    5.3.3 Stability 43
    5.3.4 Restrictions 43
6. SIMULATION STUDIES **45**
  6.1 Introduction about Simulations 45
  6.2 ABB IRB 6620 46

6.2.1 Specifications and dimensions of ABB IRB6620 48
6.3 How to Get Parameters from SolidWorks 51
6.4 Computed-Torque Control Simulation to Track a Joint Space Trajectory 52
6.5 Adaptive Control Simulation to Track a Joint Space Trajectory 53
6.6 Adaptive Control and Computed-Torque Control Results' Comparison for a Preplanned Joint Space Trajectory Tracking 54
6.6.1 In the case of all masses are fully known 55
6.6.2 In the case of all masses are presumed as 1.5 times heavier 57
6.6.3 In the case of all masses are presumed as 2 times heavier 59
6.6.4 In the case of all masses are presumed as half of the real values 61
6.7 Adaptive Control and Computed-Torque Control Results' Comparison for a Preplanned Operational Space Trajectory Tracking 63
6.7.1 In the case of all masses are fully known 63
6.7.2 In the case of all masses are presumed as 1.5 times heavier 66
6.7.3 In the case of all masses are presumed as 2 times heavier 69
7. CONCLUSIONS AND RECOMMENDATIONS 73
REFERENCES 75
CURRICULUM VITAE 79

ABBREVIATIONS

DOF	: Degree of freedom
VRT	: Virtual reality toolbox
SOA	: Spatial Operator Algebra
$\boldsymbol{SO}(\boldsymbol{n})$	: Special orthogonal group of n
$\boldsymbol{so}(\boldsymbol{n})$	: Skew symmetric matrix of n
$\boldsymbol{SE}(\boldsymbol{n})$	: Special Euclidean group of n
$\boldsymbol{se}(\boldsymbol{n})$	: Twist of the Euclidean group of n
$\mathbb{R}^{\boldsymbol{n}}$	: Euclidean space of dimension n
$\mathbb{C}^{\boldsymbol{n}}$	: Complex space of dimension n
CM	: Center of mass
CT	: Computed-Torque
CTC	: Computed-Torque Control
ACT	: Adaptive Computed-Torque
ACTC	: Adaptive Computed-Torque Control

LIST OF TABLES

Page

Table 6.1 : Technical specification data of IRB 6620 [34] 48
Table 6.2 : Performance data of IRB 6620 [36] 49

LIST OF FIGURES

Page

Figure 2.1 : Pure rotational motion. 8
Figure 2.2 : Exponential coordinates. 9
Figure 2.3 : Rigid motion in $\mathbb{R}^3$. 11
Figure 2.4 : Exponential coordinates for rigid body motion. 13
Figure 3.1 : Vectors associated with link k of the manipulator i. 17
Figure 3.2 : Serial manipulator on a fixed platform. 19
Figure 3.3 : Propagation from joint ^{i}n to tip t of arm i. 21
Figure 4.1 : Computed-torque control scheme, showing inner and outer loops. 33
Figure 5.1 : Adaptive computed torque control block scheme. 38
Figure 6.1 : ABB IRB 6620 (Floor mounted) [35]. 46
Figure 6.2 : ABB IRB 6620 (Ceil mounted) [35]. 46
Figure 6.3 : Working range of IRB 6620 for floor mounted type [35]. 47
Figure 6.4 : Working range of IRB 6620 for ceil mounted type [35]. 47
Figure 6.5 : Axis description of IRB 6620 [34]. 48
Figure 6.6 : Axis of rotations of IRB 6620 [36]. 49
Figure 6.7 : IRB 6620 CAD model with principal distances between axis [36]. 50
Figure 6.8 : DressPack for material handling (A: Connection point at axis 3) [36]. 50
Figure 6.9 : Selection of a link from the SolidWorks. 51
Figure 6.10 : Mass properties screen from SolidWorks for the selected part/parts. 52
Figure 6.11 : Mass properties screen from SolidWorks for the selected part/parts. 52
Figure 6.12 : Interior of Virtual Reality block used in the simulation. 53
Figure 6.13 : Adaptive computed-torque control simulation. 53
Figure 6.14 : Interior of torque limits block used in the simulation. 54
Figure 6.15 : Joint space desired trajectory. 54
Figure 6.16 : Joint space trajectory tracking with computed-torque control. 55
Figure 6.17 : Joint space trajectory tracking with adaptive control. 55
Figure 6.18 : Joint space trajectory tracking position error with CTC. 56
Figure 6.19 : Joint space trajectory tracking position error with adaptive control. 56
Figure 6.20 : Adaptive control rule link masses’ estimation. 56
Figure 6.21 : Joint space trajectory tracking with computed-torque control. 57
Figure 6.22 : Joint space trajectory tracking with adaptive control. 57
Figure 6.23 : Joint space trajectory tracking position error with CTC. 58
Figure 6.24 : Joint space trajectory tracking position error with adaptive control. 58
Figure 6.25 : Adaptive control rule link masses’ estimation. 58
Figure 6.26 : Joint space trajectory tracking with computed-torque control. 59
Figure 6.27 : Joint space trajectory tracking with adaptive control. 59
Figure 6.28 : Joint space trajectory tracking position error with CTC. 60
Figure 6.29 : Joint space trajectory tracking position error with adaptive control. 60
Figure 6.30 : Adaptive control rule link masses’ estimation. 60
Figure 6.31 : Joint space trajectory tracking with computed-torque control. 61

Figure 6.32 : Joint space trajectory tracking with adaptive control. 61
Figure 6.33 : Joint space trajectory tracking position error with CTC...................... 62
Figure 6.34 : Joint space trajectory tracking position error with adaptive control.... 62
Figure 6.35 : Adaptive control rule link masses’ estimation.................................... 62
Figure 6.36 : Corresponding joint space trajectory to produce desired ellipsoid trajectory in Operational space. 63
Figure 6.37 : Operational space trajectory tracking with CTC. 63
Figure 6.38 : Operational space trajectory tracking with adaptive control. 64
Figure 6.39 : Joint space trajectory tracking with CTC. 64
Figure 6.40 : Joint space trajectory tracking with adaptive control. 64
Figure 6.41 : Trajectory tracking position error with CTC. 65
Figure 6.42 : Trajectory tracking position error with adaptive control. 65
Figure 6.43 : Adaptive control rule link masses’ estimation.................................... 65
Figure 6.44 : Operational space trajectory tracking with CTC. 66
Figure 6.45 : Operational space trajectory tracking with adaptive control. 66
Figure 6.46 : Joint space trajectory tracking with CTC. 67
Figure 6.47 : Joint space trajectory tracking with adaptive control. 67
Figure 6.48 : Trajectory tracking position error with CTC. 68
Figure 6.49 : Trajectory tracking position error with adaptive control. 68
Figure 6.50 : Adaptive control rule link masses’ estimation.................................... 68
Figure 6.51 : Operational space trajectory tracking with CTC. 69
Figure 6.52 : Operational space trajectory tracking with adaptive control. 69
Figure 6.53 : Joint space trajectory tracking with CTC. 70
Figure 6.54 : Joint space trajectory tracking with adaptive control. 70
Figure 6.55 : Trajectory tracking position error with CTC. 71
Figure 6.56 : Trajectory tracking position error with adaptive control. 71
Figure 6.57 : Adaptive control rule link masses’ estimation.................................... 71

LIST OF SYMBOLS

$_{n}0$ **:** $n \times n$ zero matrix. (5.9)
$_{n}I$ **:** $n \times n$ identity matrix. (5.9)
$_{3}I$ **:** 3×3 identity matrix. (3.38)
R_{ab} **:** Rotation matrix of B relative to A. (2.1)
ξ : Twist coordinates of twist $\hat{\xi}$. (2.31)
$\hat{\xi}$: Twist. (2.25)
$^{i}\vec{\ell}_{k-1,k}$: Link vector of link $k-1$ of arm i. (3.1)
$^{i}\hat{\ell}_{k-1,k}$: Skew symmetric form of $^{i}\vec{\ell}_{k-1,k}$. (3.1)
$^{i}\vec{l}_{k,c}$: From the origin of the link frame k to the CM of the link. (3.40)
$^{i}\vec{h}_{k}$: Axis of rotation and/or translation vector of link k of arm i. (3.1)
$^{i}\vec{\vec{H}}_{k}$: Axis of rotation and/or translation spatial vector of link k of arm i. (3.7)
^{i}H : Axis of rotation and/or translation matrix of arm i. (3.13)
$^{i}\dot{\theta}_{k}$: Joint rate between link $k-1$ and k of arm i. (3.1)
$^{i}\underline{\dot{\theta}}$: Stacked joint rates of arm i. (3.13)
$^{i}\ddot{\theta}_{k}$: Joint acceleration between link $k-1$ and k of arm i. (3.27)
$^{i}\vec{\omega}_{k}$: Angular velocity vector of link k of arm i. (3.1)
$^{i}\vec{v}_{k}$: Linear velocity vector of link k of arm i. (3.1)
$^{i}\vec{\vec{V}}_{k}$: Spatial velocity vector of link k of arm i. (3.5)
$\vec{\vec{V}}_{b}$: Base spatial velocity vector of arm i. (3.42)
$^{i}\vec{\vec{V}}_{t}$: Tip spatial velocity vector of arm i. (3.18)
$^{i}\underline{V}$: Stacked link spatial velocities arm i. (3.14)
$^{i}\phi_{k,k-1}$: Propagation operator from link $k-1$ to link k of arm i. (3.6)
$^{i}\phi_{b}$: Propagation operator from the first joint of arm i to the base. (3.42)
$^{i}\phi_{t}$: Propagation operator from the first joint to the tip point of arm i.(3.19)
$^{i}\phi$: Propagation operator of arm i. (3.13)
^{i}J : Jacobian operator of arm i. (3.22)
$^{i}J_{\omega}$: The part of the Jacobian operator related to angular velocity. (3.25)
$^{i}J_{v}$: The part of the jacobian operator related to linear velocity. (3.25)
$^{i}J^{\#}$: Pseudo inverse of the jacobian operator of arm i. (3.26)
$^{i}J^{T}$: Transpose of the Jacobian operator of arm i. (3.42)
$^{i}\vec{\vec{a}}_{k}$: Bias spatial acceleration of link k of arm i. (3.30)
$^{i}\underline{a}$: Stacked bias spatial accelerations of the arm i. (3.31)
$^{i}\vec{\vec{b}}_{k}$: Bias spatial forces of link k of arm i. (3.40)

${}^{i}\underline{b}$: Stacked bias spatial forces of arm i. (3.41)
${}^{i}\mathfrak{I}_{k}$: Inertia matrix of link k of arm i. (3.38)
${}^{i}\vec{\vec{F}}_{k}$: Spatial force of link k of arm i. (3.32)
$\rho(x, y, z)$: Mass density of the object. (3.37)
${}^{i}\vec{\tau}_{k}$: Torque vector of link k of arm i. (3.33)
${}^{i}\vec{f}_{k}$: Force vector of link k of arm i. (3.33)
${}^{i}\underline{F}$: Stacked spatial forces of arm i. (3.34)
${}^{i}M_{k}$: Link mass matrix of link k of arm i. (3.38)
${}^{i}m_{k}$: Mass vector of the link k of arm i. (3.38)
${}^{i}M$: Mass matrix of arm i. (3.39)
${}^{i}\mathcal{M}$: Generalized mass matrix of arm i. (3.43)
${}^{i}\underline{C}$: Bias terms including Coriolis and gravity for arm i. (3.44)
${}^{i}\mathcal{M}_{b}$: Mass matrix between the base and the i^{th} arm. (3.45)
${}^{i}\vec{\vec{F}}_{t}$: Spatial tip force of arm i. (3.42)
${}^{i}\underline{\tau}_{d}$: Disturbance torque vector of arm i. (4.1)
$q_{d}(t)$: Desired trajectory in joint space. (4.3)
$q(t)$: Trajectory in joint space. (4.3)
$e(t)$: Trajectory tracking error in joint space. (4.3)
$\omega(t)$: Disturbance function. (4.8)
$u(t)$: Control input in CT. (4.7)
ζ : Damping ratio. (4.20)
ω_{n} : Natural frequency. (4.20)
φ : $r \times 1$ vector of unknown parametric quantities. (5.1)
$W(q, \dot{q}, \ddot{q})$: $n \times r$ known time functions matrix also called as regression matrix. (5.1)
ψ : Tracking error including position and velocity errors in ACT. (5.2)
$\boldsymbol{V}$: Candidate Lyapunov function. (5.10)
Γ : A positive-definite $r \times r$ matrix to gain ACT update rule. (5.11)
${}^{i}\varphi^{\#}$: Pseudo inverse of φ. (5.25)

NONLINEAR CONTROL METHODS OF INDUSTRIAL SERIAL ROBOTS

SUMMARY

The word robot was introduced to the public by the Czech writer Karel Capek in his play R.U.R. (Rossum's Universal Robots) which published in 1920. However, early studies about modern robots started after World War II. In 1960s, the studies about modern robotics started to increase rapidly. At first, to control the robot manipulators simple control techniques were used like PID. However, today very complicated control techniques are being used like Robust Control and Adaptive Control to control the industrial robots.

Classifying the control methods are mainly divided into two groups. The first one is the linear control and the second one is the nonlinear control. Nonlinear control can show much better results than the linear control, however designing a nonlinear controller requires a strong mathematical background.

Robot dynamics is concerned with the relationship between the forces acting on a robot mechanism and the accelerations they produce. There are two main dynamic problems. They are forward dynamics problem and inverse dynamics problem. In forward dynamics, applied tip point forces are given and the problem is to find the joint accelerations of the robot arm; in inverse dynamics, joint accelerations are given and the problem is to find the tip point forces.

In this thesis the tip point torques are calculated by the computed torque or adaptive control. These torque values are limited with the real robot joint torque parameters. This torque values scattered to the links of the robot arm using forward dynamics.

The methodology presented in this thesis is based on Spatial Operator Algebra (SOA). Nonlinear control algorithms applied to an industrial serial robot, which is modeled with SOA. The complexity of dynamic analyze and control structures come from the adaption of the computed torque control and adaptive control into the SOA robot modeling theory, which have not been existed in the literature before this thesis.

In this thesis, all of the control and modeling algorithms are applied to ABB IRB 6620 using real data. For structural modeling of an industrial robot arm, ABB IRB 6620 is chosen because it is favorable for both being widespread in the factories and being durable. The virtual model was constructed from the CAD files, which have been distributed from the ABB’s official web site.

Simulation results are obtained by using MATLAB and animation applications are obtained by using the virtual reality toolbox of MATLAB (VRT). All simulation

results that have been shown in the thesis are obtained by virtualizing ABB IRB 6620, which is a well-known industrial serial robot.

ENDÜSTRİYEL SERİ ROBOTLARIN DOĞRUSAL OLMAYAN KONTROLÜ

ÖZET

Robot sözcüğünü ilk olarak Çekoslovak yazar Karel Capek'in 1920 yılında yazdığı "Rossum'a Universal Robots" adlı tiyatro oyununda kullanmıştır. Çekçe 'de "robota" sözcüğü "iş, zorla çalıştırılan işçi ya da köle" anlamlarına gelmektedir. Genel bir tanım olarak robot, otonom veya önce programlanmış elektronik cihazlar olarak tanımlanabilir. Robot Institute of America robotları "Robot; özel aletleri, parçaları bir yerden başka bir yere götürmek için tasarlanmış, tekrar programlanabilen çok fonksiyonlu bir makinadır" şeklinde tanımlamıştır.

II. Dünya savaşından sonra robotlar üzerine çalışmalar başlamıştır ve kontrol konularındaki çalışmalar da hız kazanmıştır. 1940'larda "Argonne National" laboratuvarlarında radyoaktif maddeleri tutabilmek için basit mekanik yapılara sahip "master-slave" tipte bir robot kol tasarlanmıştı. Bu sistemde "master"ı kontrol eden insanın yaptığı hareketleri "slave" robot izliyordu.

1950'lerin ortalarında elektriksel ve hidrolik robot kolları geliştirilmiştir. George C. Devol 1959'da ilk programlanabilir endüstriyel robotu üretmiştir. 1961 yılında ise ilk endüstriyel robot fabrikada üretimde kullanılmıştır. 1962'de Ernst dokunmayı algılayabilen bir el geliştirmiştir ve "Argonne National" laboratuvarlarında geliştirilen bir robot koluna takılmıştır. 1969 yılında Scheinman, Standford üniversitesinde ilk 6 serbestlik dereceli robot kollarından birini gerçekleştirmiştir. Bu kol bilgisayar ile kontrol edilmekteydi ve tahrik elektrik motorlarıyla sağlanmaktaydı.

Robotlar üzerine yapılan ilk çalışmalarda araştırmacılar robotların kinematik ve dinamik modellemelerinde büyük sorunlar yaşamışlardır. 1968 yılında Pieper, robot kinematiği üzerine, 1971'de Kahn ve Roth, robot dinamiği üzerine bir makale yayınlamışlardır. 1969 yılında ise General Electric ilk yürüyen robotu üretmiştir.

1970'lerde sensör teknolojisindeki gelişmeler hem robot kontrolünü kolaylaştırmış hem de robotların gelişmesini sağlamıştır. 1973'te Bolles ve Paul, bir robota görüntü ve kuvvet geri beslemesi yardımıyla su pompası montajı yapmasını saplamışlardır. 1974'te Inoue robotlarda yapay zekâ uygulamaları ile uğraşmıştır. Be jczy, uzay çalışmalarında kullanılmak üzere bir kontrol metodu geliştirmiş ve bu metodu da Standford robotu üzerinde denemiştir.

1960'lardan beri robotun endüstride kullanılmasıyla görülen faydalar, hem robot teknolojisinde hızlı bir gelişme sağlamıştır, hem de robotların seri üretimdeki paylarını artmıştır.

Endüstride kullanılan makinalar özel amaçlıdır. Yeni bir üretim yapılması gerektiği zaman makinaların değiştirilmesi gerekir. Mevcut makinalar çoğu zaman bu yeni proses için kullanılamaz. Robotlarda durum böyle değildir. Robotların görevleri değiştirilebilir, farklı proseslere adapte edilebilir ve üretim maliyetini düşürmektedir. Bu yüzden sanayide robot kullanımı her geçen gün artmaktadır. Özellikle otomotiv endüstrisinde kullanımı oldukça yaygınlaşmıştır.

Sanayi robotları genel olarak üç başlık altında incelenebilir; seri, paralel ve bunların birleştiği tipte (hibrit) hareket eden robotlar. Seri robotların yapısı genelde basittir ve çalışma uzayları çok geniştir. Paralel robotların ise hassasiyet ve tekrarlanabilirlik değerleri çok yüksek olmasına rağmen, çalışma uzayları çok küçüktür. Hareket kabiliyetleri kısıtlı olduğu için geniş bir çalışma uzayına sahip olamazlar. Paralel robotlar sınır tekilliklerine çok daha kolay yakalanırlar. Seri robotların aksine, paralel robotların ters kinematiği kolaydır, ileri kinematiği zordur.

Engelden sakınma gibi algoritmaların kolaylıkla uygulanabilmesi, geniş çalışma uzayına sahip olması ve çok daha kolay programlanabilmesi sebebiyle genelde sanayide seri robotlar tercih edilse de, yüksek hassasiyet ya da yüksek moment ihtiyacının duyulduğu durumlarda paralel robotlar kullanılmaktadır. Paralel robotların kullanıldığı yerler olarak metal işleme gösterilebilir. Ayrıca, paralel robotlar kendilerine eğlence sektöründe de yer bulmaktadır. 9D, 8D ve 7D sinemalarda, sinemanın alt tabanının tahrik ve kontrol edilebilmesi için bir paralel robot olan Stewart platformu kullanılmaktadır.

Endüstride en yaygın kullanılan robotlar seri robotlar olduğu için bu tez endüstriyel seri robotları ele almaktadır. Bu sistemlerin kinematik ve dinamik analizleri uzaysal operatör cebri (SOA) kullanılarak detaylı olarak açıklanmıştır. Bu bilgiler ışığında modellenen robotlar için doğrusal olmayan kontrolörler tasarlanmıştır.

Uzaysal operatörler, rijit yapıların kinematik ve dinamik modellenmesini ve analizini kolaylaştıran altı boyutlu vektörler olarak tanımlanabilir. Üç boyut açısal hızlardan, diğer üç boyut da lineer hızlardan oluşur. Uzaysal operatörler özyinelemeli yapısı sayesinde çoklu manipülatör sistemlerine kolayca uygulanabilir. Diğer kinematik ve dinamik analiz metotlarına göre daha sistematik ve kolay programlanabilir bir yüksek performanslı hesaplama algoritmasıdır.

Üç boyutlu çalışma uzayında geçerli bütün konfigürasyonlara ulaşmak için manipülatörün serbestlik derecesinin altıdan (üç boyutta dönme ve üç boyutta öteleme) düşük olmaması gerekir. Tekil durumlarda Jakobiyen matrisin rankı düşeceği için genel olarak 6 serbestlik dereceli robotlar tercih edilmektedir.

Modelleme kinematik ve dinamik olmak üzere ikiye ayrılır. Robotların modellenmesinde İleri ve Ters Kinematik olmak üzere iki ana problem vardır. İleri kinematikte eklemlerin hareket etmesi geren hızlar verilir ve manipülatörün uç noktasının açısal ve doğrusal hızlarının hesaplanması incelenir. Ters kinematikte ise, manipülatörün uç noktasının açısal ve doğrusal hızları verilerek eklem hızlarının hesaplanması istenir. Dinamik analizde de İleri ve Ters Dinamik olmak üzere iki ana problem vardır. İleri dinamik analizinde, manipülatörün uç noktasındaki tork vektörü ve kuvvet vektörü verilerek eklemlerin ulaşması gereken ivme değerlerinin hesaplanması istenir. Ters dinamik analizde ise, eklemlerin ulaşması gereken ivmeler verilir ve manipülatörün uç noktasındaki uzaysal tork vektörü hesaplanır.

1960'lardan beri robotun endüstride kullanılmasıyla ortaya çıkan kontrol problemler doğrusal kontrol yerine, doğrusal olmayan kontrolör tasarlama ihtiyacını ortaya çıkarmıştır. Günümüzde ise, çok sayıda lineer olmayan kontrolörler robotlarda kullanılmaktadır.

Kontrolörler iki ana gruba ayrılabilir. Doğrusal olan kontrolörler ve doğrusal olmayan kontrolörler olarak. Doğrusal kontrolörler basit çalışma mantıkları ve haklarındaki geniş literatür ile uzun zaman boyunca kullanılmışlardır. Doğrusal kontrolörlerin ilkel kaldığı durumların ilk olarak uçaklarda ortaya çıkması ile doğrusal olmayan kontrolör tasarlamanın zorunluluğu ortaya çıktığında, artık bu konu üzerinde uğraşılmaya başlandı. Daha sonra bu konuda yapılan çalışmaların sonuç vermesiyle, doğrusal olmayan kontrolörler tasarlanıp, uçaklarda kullanılmaya başlandı. Uçaklardaki başarılı sonuçları görüldükten sonra, diğer lineer olmayan uygulama alanlarında da kendisine kolayca yer bulmuştur. Doğrusal olmayan kontrolörün tasarımı doğrusal kontrolörlere göre çok daha zordur, ama elde edilen sonuçlardaki başarılar bu kontrolörlerin kullanımını yaygınlaştırmıştır.

Literatürde farklı doğrusal olmayan kontrolör çalışması bulunmaktadır, ama robot gibi mekanik sistemler için geri besleme doğrusallaştırmasının özel bir uygulaması olan hesaplanmış tork kontrolü, adaptif kontrol ve robust kontrol tercih edilmektedir. Tercih edilmemesine rağmen, literatürde kayma kipli mod kontrol metodu ile de kontrol edilen robotlar vardır. Kayma kipli mod kontrolün tercih edilmeme sebebi ise, bu yöntemin yüksek frekansa izin veren sistemler için uygun oluşudur. Robot ve robotlar gibi mekanik sistemlerin kontrolörün istediği kadar hızlı cevap veremeyecekleri durumda sistem kararsızlığa gidebilir ya da yüksek frekans ihtiyacı mekanik sistemlere zarar verebilir.

Tüm parametrelerin çok yüksek doğrulukla bilindiği durumda, hesaplanmış tork kontrol yöntemi çok iyi cevap vermektedir. Modellenemeyen belirsizliklerin artması durumunda ise hesaplanmış tork kontrolü etkinliğini kaybetmektedir. Bu durumlarda robust kontrol ya da adaptif kontrol tercih edilmektedir. Robust kontrolün matematiği ile adaptif kontrolün temelleri birbirine benzemektedir.

Yapılan çeşitli çalışmalar, simülasyonlar ve deneyler Adaptif kontrol ile elde edilen sonuçların, Robust kontrol ile elde edilen sonuçlardan çok daha başarılı olduğunu göstermektedir. Yine yapılan literatür taraması sonucunda, modellenemeyen belirsizliklerin artması durumunda, Adaptif kontrol yönteminin hem Robust kontrol hem de Hesaplanmış Tork kontrolünden çok daha iyi sonuç verebildiği açıkça görülmektedir.

Bu tezde, simülasyon çalışmaları MATLAB kullanılarak, animasyon uygulamaları ise MATLAB/Simulink'te bulunun sanal gerçeklik araç kutusu (VRT) kullanılarak gerçekleştirilmiştir. Simülasyonlar ABB tarafından üretilen IRB 6620'nin kataloglarından alınmış gerçek robot kolun parametreleri kullanılarak elde edilmiştir. Elde edilen bu sonuçlar ayrıntılı olarak analiz edilip, incelenmiştir. Sonuçları detaylı şekilde açıklanmıştır.

1. INTRODUCTION

Robots are becoming more and more popular in massive production. Number of robots is rapidly increasing in the industries where safety and fast production are needed. These robots can be mainly classified as three types; serial robots, parallel robots and hybrid of these two mechanisms. There are some pros and cons to each other’s.

Serial robots are the most common ones. They can do various tasks in the industry, like spraying, painting, water jet, flaming, arc welding, spot welding, plasma cutting, cnc feeding, spindle motor milling, foaming, gluing, dispense, bending, laser cutting, die casting, etc.

Parallel robots are for the applications where repeatability and accuracy are vital or very high torques are demanded. Most commonly used parallel robot types are Delta robot and Stewart platform.

Hybrid of these two mechanisms is also used in some applications to provide special requirements, like CNC Machining Centre’s and Modules.

Serial robots can be classified based on the degrees of freedom. Robots are defined in 6-dimensional space.

Robots that have less than 6 DOF can’t go in at least one direction or rotate at one axis. These robots can’t be fully defined at 6 dimensions. Because of this fact, they suffer from singularities more than any other types.

Robots that have more than 6 DOF can be fully defined in 6-dimensional space. Furthermore, this kind of robots can move at all dimensions and obstacle avoidance etc. can be applied to these robots.

Robots that have 6 DOF can be fully defined in 6-dimensional space and can move freely in 6-dimensional space. However, obstacle avoidance algorithms etc. can’t be applied to this kind of robots.

Robots with 6 degrees of freedom are the major group that is being used in the Industry. Because, they can move in 3 axes and rotate in 3 axes, thus they are capable of doing all movements in 6-dimensional space and also industrial applications don’t require additional degrees of freedom over 6 DOF, mostly. Considering these facts with their payload capacity, reachable space and dexterous space values, it is fair to accept, they are the optimal solution for most of the industrial applications.

The organization of the thesis is as follows: Chapter 2 describes general representations of rigid body motion. Chapter 3 presents Kinematic Analysis and Dynamic Analysis of a serial manipulator on a fixed platform using SOA. Chapter 4 explains Computed Torque Control as a successful nonlinear control technique, which is preferable for robotics as many mechanical systems. Chapter 5 explains Adaptive Control to eliminate uncertainties. Chapter 6 explains simulation studies, shows and compares their results under various different circumstances. Finally, Chapter 7 is the conclusion.

1.1 Purpose of Thesis

Industrial serial robots present some problems that exist because of their nature. They are nonlinear systems and they needed to be modeled perfectly. This is impossible from the fact that we cannot know masses, center of masses and friction coefficients fully. That is why Adaptive Control is vital to track the trajectory more correctly by estimating these values. This requires a good understanding of kinematics, dynamics and keen knowledge of the nonlinear control systems. From the controlling point of view, nonlinear control methods require much more effort when compared to linear control methods. However, nonlinear control can provide much better results compared to linear control.

In this thesis, Kinematic Analysis with SOA, Dynamic Analysis with SOA, Computed Torque Control and Adaptive Control of an Industrial serial robot are illustrated and their detailed equations are given.

The purpose of this thesis is to apply Computed Torque and Adaptive Control to the robots, which modeled with SOA. The use of Computed Torque and Adaptive Control methods for the robots modeled with SOA has not been in the literature before this thesis.

1.2 Literature Review

There are various studies about controlling a serial robot, which are well documented in the literature [1], [2], [3] and [4]. Computed torque as feedback linearization has been well clearly covered in many different point aspects. Another important aspect to the robot control method is adaptive computed torque. Some uncertainties come from the nature of robots. Masses, center of masses and friction coefficients can't be known fully. These unknown parameters cause uncertainties. Adaptive control can eliminate these uncertainties' effects during trajectory tracking, i.e. converging path tracking error to zero or at least cause them to be limited in a bound. For a detailed review on various control methods, one can refer to a recent book [1] by L. Sciavicco and B. Siciliano and [4] by L. F. Lewis.

There are a number of methods used for kinematics and dynamics for robot arms. Out of these methods, Denavit-Hartenberg is well-known and well-used in worldwide. Jacques Denavit and Richard Hartenberg introduced this convention in 1955 in order to standardize the coordinate frames for spatial linkages [5] and [6]. The more number of degrees of freedom, the more difficult it gets to apply D-H to robots. When adding this controller calculations, necessary calculations period increase dramatically. D-H modeling is proportional with the square of DOF of the modeled robot manipulator.

Because, D-H modeling calculation difficulty is proportional with the square of DOF, in the meanwhile SOA modeling calculation difficulty is proportional with the DOF. D-H performance can be acceptable even for 5 DOF, but over for 5 DOF robotic manipulators, it is getting more and more difficult to apply, and over 7 DOF it is nearly impossible to apply D-H modeling method for a real-time application. That is way, modeling a robotic manipulator over five degrees of freedom should be modeled with SOA.

All of them show that, for a high performance algorithm to reduce the difficulty for modeling a manipulator with high number of degrees of freedom is needed. It is also essential that this should be a vector-based algorithm so that physical insight can be provided.

One such robot-modeling algorithm is known as "screw theory." The elements of screw theory can be traced up to the work of Chasles and Poinsot in the early 1800s.

Chasles proved that a rigid body can be moved from any position to any other by a movement consisting of rotation about a straight line followed by translation parallel to that line. Robert S. Ball developed a complete theory of screws using the theorems of Chasles and Poinsot as a starting point. Robert S. Ball published his studies in 1900 [7]. There are two main advantages of the screw theory for describing rigid body kinematics and dynamic [8]. The first one is that they allow a global description of rigid body motion, which does not suffer from local singularities due to the use of local coordinates. The second advantage of screw theory is that it provides a very geometric description of rigid motion that greatly simplifies the analysis of mechanisms. In 1983, Featherstone [9] presents a study on the computation of robot dynamics using articulated body inertias utilizing the spatial algebra. After that, Guillermo Rodriguez [10] declares that inverse and forward dynamics of problems for multi-link serial manipulators are solved by using recursive techniques from linear filtering and smoothing theory. In 1991, Abhinandan Jain [11] published in its paper that a unified formulation about serial rigid multibody systems can be developed by utilizing the tools provided by the Spatial Operator Algebra (SOA). Other works of Featherstone et al [12, 13]. Other works of Guillermo Rodriguez et al on SOA are mentioned in [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24].

Also, there are other studies about SOA in the literature which I find them very important [25, 26, 27].

In this thesis in order to obtain Adaptive Control Rule, Lyapunov Stability is used. Lyapunov Stability for robotics has been well documented and illustrated in many books [4, 8].And also Lyapunov Stability has been explained for general nonlinear systems [28, 29].

In this thesis, path planning [30], pseudo inverse [31] and singular value decomposition [32] are used but for the sake of integrity and clarity of nonlinear control, they are not explained in the details.

In this thesis, I analyze dynamics of a serial robot manipulator using a screw theory based algorithm called Spatial Operator Algebra (SOA). I also modify different nonlinear control methods to be compatible with SOA mathematical theory and then I apply these mathematical theories to an industrial robot. All simulation values come from a real industrial robotic manipulator ABB IRB 6620 [33, 34, 35].

1.3 Motivation

The present work is focused on the development of a new control algorithm for a novel robot modeling theory, which was called SOA. The project starts with the modeling an industrial serial robot arm. Then, computed torque control and adaptive control mathematical models have been applied to the robot. First of all, these control methods have been investigated in the literature and then all of these control methods have been modified to achieve the adaption from D-H to the Spatial Operator Algebra.

This work intends to represent to unify the strong nonlinear control theories with the new and fast robot modeling theory, Spatial Operator Algebra. In addition, applying these theories to ABB industrial robot arm simulation using real robot arm parameters is our another focus. A detailed comparison of computed torque control results and adaptive control results are shown.

The use of SOA mathematical models and algorithms fasten the computations. In this way, the algorithms can provide shorter cycle times. With shorter cycle times, robot arms and robotic systems can be controlled more effectively and strongly.

The computed torque control and adaptive control methods provide the nonlinear control for robots. With the addition to the SOA method, both robustness and speed can be provided at the same time.

2. RIGID BODY MOTION

All rigid body motion can be defined using translation and rotation transformations [8]. Let us O be an object described as a subset of $\mathbb{R}^3$. We can define any object in Cartesian coordinates using Euclidean space properties. A rigid body motion of an object can be represented by $g(t): O \to \mathbb{R}^3$ which shows continuous family of mappings. This mapping clearly shows that how individual points in the body can be moved as a function of time relative to some fixed Cartesian coordinate frame. And also this mapping represents that we can use vector definition instead of points. Let us consider two points $p, q \in O$ and the vector $v \in \mathbb{R}^3$ connecting p to q. It is defined to be a direct line segment stretching from $p$to q. In Cartesian coordinates this is given by $v = q - p$ with $p, q \in \mathbb{R}^3$. Even though the obvious fact that they are conceptually quite different, both points and vectors are defined using similar three components in Cartesian coordinates.

A mapping $g: \mathbb{R}^3 \to \mathbb{R}^3$ is a rigid body transformation should satisfy the following properties:

1. Length is preserved: $\|g(p) - g(q)\| = \|p - q\|$ for all points $p, q \in \mathbb{R}^3$
2. The cross product is preserved: $g_*(v \times w) = g_*(v) \times g_*(w)$

If we look at the first property, this gives us the distance between points on a rigid body are not altered by rigid motions. However, this condition is not sufficient because it allows internal reflections, which are not physically realizable [8]. Thus, a rigid body transformation must also satisfy second property as well as to satisfy the first property to preserve orientation.

Although the distance between points is fixed and the cross product between vectors is also fixed, particles in a rigid body can be moved relative to each other. However, those particles in a rigid body cannot be translated; they can only be rotated with respect to each other. Thus, to keep track of the motion, of a rigid body, we need to keep track of the motion of any one particle of the rigid body and the rotation of the body about this point [8].

In order to do this, we represent the configuration of a rigid body by attaching a Cartesian coordinate frame. In this way, it is provided for some points on the rigid body and keeping track of the motion of this body coordinate frame relative to a fixed frame. The motion of the individual particles in the body can then be retrieved from the motion of the body frame and the motion of the point of attachment of the frame to the body [8].

2.1 Rotational Motion in $\mathbb{R}^3$

First of all let us show a pure rotation as in the figure below:

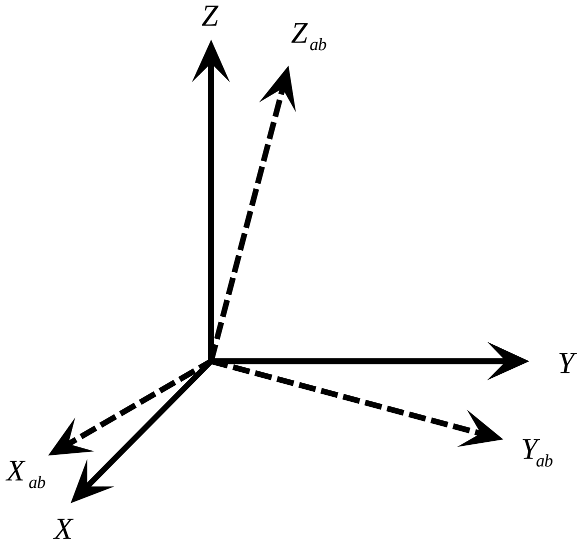


Figure 2.1 : Pure rotational motion.

Orientation of the body is described by giving the relative orientation between a coordinate frame and a fixed or inertial coordinate frame. Main coordinate frame is considered to be attached to the body. From now on, for the sake of clarity all coordinate frames in this thesis will be accepted as right-handed unless otherwise is specially declared.

Let us assume that A is the inertial frame and$B$is the body frame. $x_{ab}, y_{ab}, z_{ab} \in \mathbb{R}^3$ will be the coordinates of the principal axes of B relative to A (Figure 2.1). Using these coordinate vectors, we can obtain a 3 × 3 matrix:

$$R_{ab} = [x_{ab} \quad y_{ab} \quad z_{ab}] \tag{2.1}$$

Constructed matrices as in the Equation 2.1 are called as rotation matrices. Every rotation of the object corresponds to a rotation matrix. There are two major properties of a rotation matrix:

$$RR^T = R^TR = I \tag{2.2}$$

$$detR = +1 \text{(since the coordinate frame is right-handed)} \tag{2.3}$$

The set of all 3×3 matrices which satisfy these two properties are represented in $SO(3)$. SO means special orthogonal.

$$SO(3) = \{R \in \mathbb{R}^{3\times3}: RR^T = I, detR = +1\} \tag{2.4}$$

2.2 Exponential Coordinates for Rotation

In robotics, rotation of a body about a given axis is a very common motion. Let's assume $\omega \in \mathbb{R}^3$ to be a unit vector which specifies the direction of rotation and $\theta \in \mathbb{R}$ be the angle of rotation and its angle unit is in radian. Rotation matrix can be written as a function of w and θ.

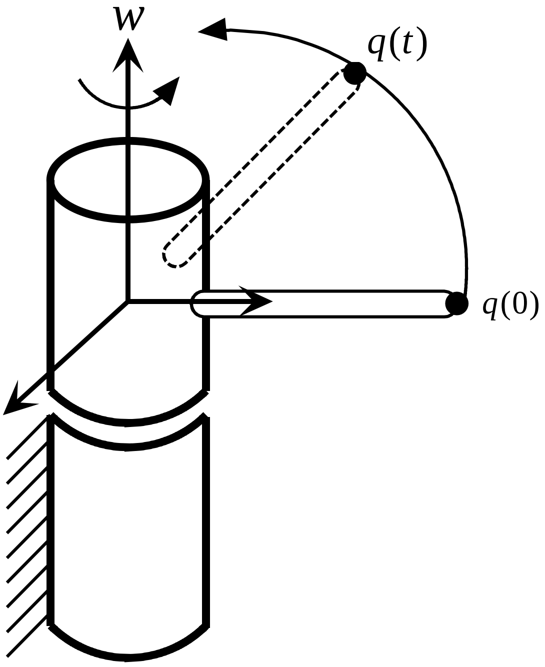


Figure 2.2 : Exponential coordinates.

Let's assume the body to rotate at constant unit velocity about the axis w. Then the velocity of the point $\dot{q}$ may be written as:

$$\dot{q}(t) = \omega \times q(t) = \widehat{\omega} q(t) \tag{2.5}$$

This is a time-invariant linear differential equation. Using the following equation we can propose a $q(t)$ as:

$$q(t) = e^{\widehat{\omega} t} q(0) \tag{2.6}$$

where $q(0)$ is the initial position of the point and $e^{\widehat{\omega} t}$ is the matrix exponential:

$$e^{\widehat{\omega} t} = I + \widehat{\omega} t + \frac{(\widehat{\omega} t)^2}{2!} + \frac{(\widehat{\omega} t)^3}{3!} + \cdots \tag{2.7}$$

If we rotate the axis ω at a unit velocity for θ units of time, then the net rotation could be written as:

$$R(\omega, \theta) = e^{\widehat{\omega} \theta} \tag{2.8}$$

where the $\widehat{\omega}$ is a skew symmetric matrix which satisfies $\widehat{\omega}^T = -\widehat{\omega}$. The vector space of all 3×3 skew matrices are denoted in $so(3)$:

$$so(3) = \{S \in \mathbb{R}^{3\times 3} : S^T = -S\} \tag{2.9}$$

For a given matrix $\widehat{\omega} \in so(3)$, $\|\omega\| = 1$ and a real number $\theta \in \mathbb{R}$, we express $\widehat{\omega}\theta$ as:

$$exp(\widehat{\omega}\theta) = e^{\widehat{\omega}\theta} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \cdots \tag{2.10}$$

This is an infinite series and it is impossible computing all of the terms. To obtain a closed-form expression for $exp(\widehat{\omega}\theta)$, we have to do several deductions using formulas for power of $\hat{a}$.

$$\hat{a}^2 = aa^T - \|a\|^2 I \tag{2.11}$$

$$\hat{a}^3 = -\|a\|^2 \hat{a} \tag{2.12}$$

if we put the terms $a = \omega\theta$, $\|\omega\| = 1$, then the exponential of $\widehat{\omega}\theta$ the equation turns into:

$$e^{\widehat{\omega}\theta} = I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots\right)\widehat{\omega} + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \cdots\right)\widehat{\omega}^2 \tag{2.13}$$

and hence:

$$e^{\widehat{\omega}\theta} = I + \widehat{\omega} sin\theta + \widehat{\omega}^2(1 - cos\theta) \tag{2.14}$$

This previous formula, commonly referred to as *Rodrigues' formula*. *Rodrigues' formula* provides an efficient method to compute $\exp(\widehat{\omega}\theta)$. It can be easily seen that $\exp(\widehat{\omega}\theta)$ is the rotation matrix which expresses rotation by θ about axis ω. We can also find θ and ω for a given any $\exp(\widehat{\omega}\theta)$ rotation matrices as:

$$\exp(\widehat{\omega}\theta) = R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.15}$$

$$\theta = cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{2.16}$$

$$\omega = \frac{1}{2sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{2.17}$$

2.3 Rigid Motion in $\mathbb{R}^3$

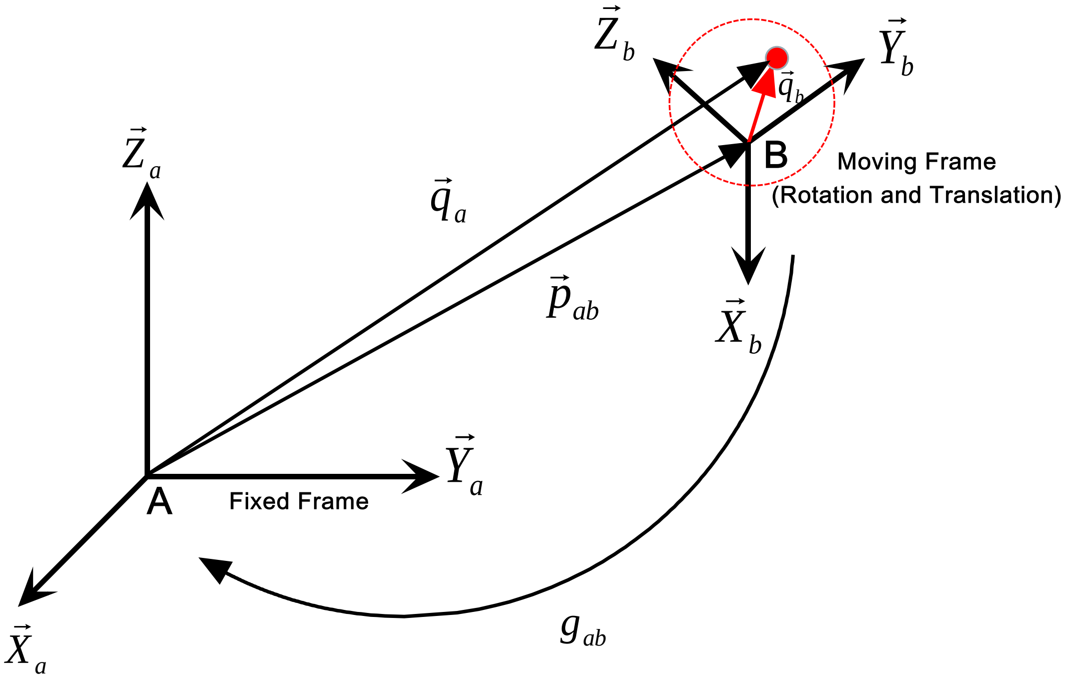


Figure 2.3 : Rigid motion in $\mathbb{R}^3$.

In the Figure 2.3, the representation of general rigid body motion, involving both translation and rotation is shown. In the Figure 2.3, we describe the position and orientation of a coordinate frame B attached to the frame A which is fixed. Let $p_{ab} \in \mathbb{R}^3$ be the position vector stretching from the origin of frame A to the origin of the frame B and $R_{ab} \in SO(3)$ the orientation of frame B is relative to frame A.

A configuration of the system consists of a pair (p_{ab}, R_{ab}). The configuration space of the system is the product space of $\mathbb{R}^3$ with $SO(3)$, It will be expressed as $SE(3)$, Special Euclidean group:

$$SE(3) = \{(p, R) : p \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times SO(3) \tag{2.18}$$

with a given q_b, we can find q_a as shown below:

$$q_a = p_{ab} + R_{ab} q_b \tag{2.19}$$

where $g_{ab} = (p_{ab}, R_{ab}) \in SE(3)$ is the specification of the configuration. By an abuse of notation, we will write $g(q)$ to denote the action of a rigid transformation on a point:

$$g(q) = p + Rq \tag{2.20}$$

so that $q_a = g_{ab}(q_b)$. We may represent it in linear form by writing it as:

$$\bar{q}_a = \begin{bmatrix} q_a \\ 1 \end{bmatrix} = \begin{bmatrix} R_{ab} & p_{ab} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_b \\ 1 \end{bmatrix} =: \bar{g}_{ab} \bar{q}_b \tag{2.21}$$

The 4×4 matrix $\bar{g}_{ab}$ is called the *homogeneous representation* of $g_{ab} \in SE(3)$. In general, if $g = (p, R) \in SE(3)$, then

$$\bar{g} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{2.22}$$

2.4 Exponential Coordinates for Rigid Motion

The notion of the exponential mapping for $SO(3)$ can be generalized to the notation of the Euclidean group $SE(3)$. It can be illustrated with a simple example.

Let us look at an example of a one-link robot in Figure 2.4, where the axis of rotation is $\in \mathbb{R}^3, \|\omega\| = 1$, and $q \in \mathbb{R}^3$ is a point on the axis. It is accepted that the link rotates with unit velocity. Then the velocity of the tip point $p(t)$ is:

$$\dot{p}(t) = \omega \times (p(t) - q) = \omega \times p(t) - \omega \times q \tag{2.23}$$

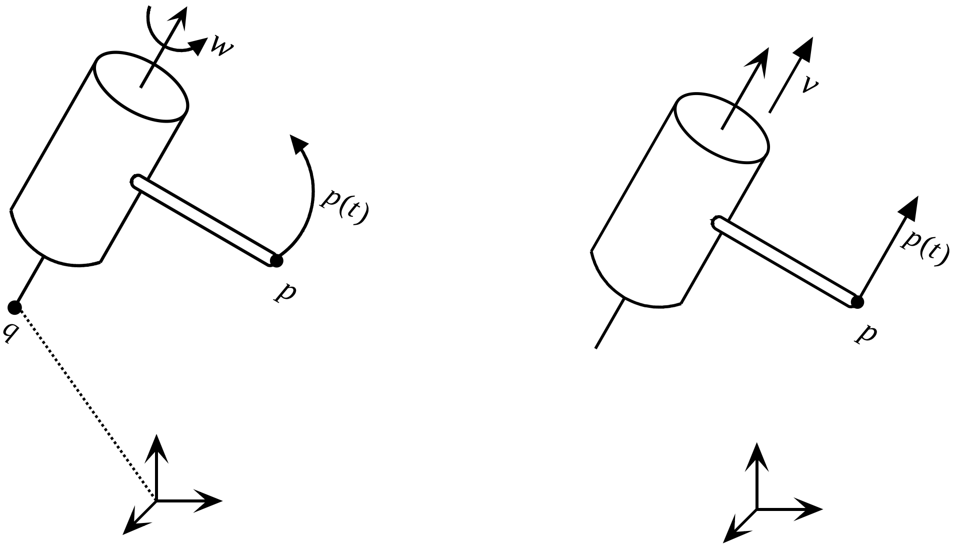


Figure 2.4 : Exponential coordinates for rigid body motion.

The equation (2.23) can be rewritten as:

$$\begin{bmatrix} \dot{p} \\ 0 \end{bmatrix} = \begin{bmatrix} \widehat{\omega} & -\omega \times q \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{2.24}$$

with $v = -\omega \times q$, we can define $\hat{\xi}$ as:

$$-\omega \times q = v \Rightarrow \hat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix} \tag{2.25}$$

and the equation turns into:

$$\begin{bmatrix} \dot{p} \\ 0 \end{bmatrix} = \hat{\xi} \begin{bmatrix} p \\ 1 \end{bmatrix} \Rightarrow \dot{\bar{p}} = \hat{\xi} \bar{p} \tag{2.26}$$

This is a first order differential equation with the solution:

$$\bar{p}(t) = e^{\hat{\xi} t} \bar{p}(0) \tag{2.27}$$

where $\bar{p}(0)$ is the initial position of the point,$\bar{p}(t)$ is current position and $e^{\hat{\xi} t}$ is the matrix exponential of 4×4 matrix. $\hat{\xi} t$ defined as:

$$e^{\hat{\xi} t} = I + \hat{\xi} t + \frac{(\hat{\xi} t)^2}{2!} + \frac{(\hat{\xi} t)^3}{3!} + \cdots \tag{2.28}$$

The scalar t represents the total amount of rotation angle. $\exp(\hat{\xi} t)$ represents the mapping from the initial location of a point to the location after rotating t radians.

The 4×4 matrix $\hat{\xi}$ is a generalized form of the skew-symmetric matrix $\widehat{\omega} \in so(3)$. Similar to the definition of $so(3)$, we can write:

$$se(3) = \{(v, \widehat{\omega}) : v \in \mathbb{R}^3, \widehat{\omega} \in so(3)\} \tag{2.29}$$

We can also write an element $\hat{\xi} \in se(3)$ as:

$$\hat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \tag{2.30}$$

An element of $se(3)$ can be both referred to as a twist, or a (infinitesimal) generator of the Euclidean group. We define the ∨ (vee) operator to extract the 6-dimensional vector:

$$\begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.31}$$

and $\xi := (v, \omega)$ the twist coordinates of $\hat{\xi}$. The inverse operator, ∧ (wedge), forms a matrix in $se(3)$ in $\mathbb{R}^6$:

$$\begin{bmatrix} v \\ \omega \end{bmatrix}^{\wedge} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix} \tag{2.32}$$

Thus, $\xi \in \mathbb{R}^6 : (v, \omega)$ represents the twist coordinates for the twist $\hat{\xi} \in se(3)$.This shows us another way of the notation for the skew-symmetric matrices.

The exponential of a twist shows the relative motion of a rigid body. As a mapping, $\exp(\hat{\xi} t)$ takes points from their initial coordinates $p(0) \in \mathbb{R}^3$, to their final coordinates:

$$p(\theta) = e^{\hat{\xi} \theta} p(0) \tag{2.33}$$

Both $p(0)$ and $p(\theta)$ are stated with a single reference frame. If a coordinate frame B is attached to a rigid body undergoing a screw motion, the configuration of the coordinate frame B related to a fixed frame A for that moment;

$$g_{ab}(\theta) = e^{\hat{\xi}\theta} g_{ab}(0) \qquad \mathbf{(2.34)}$$

This transformation can be interpreted as follows: multiplication by $g_{ab}(0)$ maps the coordinates of a point relative to the B frame into the A frames' coordinates and the exponential map transforms the point to its final destination.

3. KINEMATIC AND DYNAMIC MODELLING USING SPATIAL OPERATOR ALGEBRA

3.1 Spatial Operator Algebra (SOA)

Spatial Operator Algebra (SOA) is a high performance algorithm to model a robotic manipulator with high degrees of freedom. SOA is a recursive method that uses coordinate-free vectorial notation.

To clarify the notation throughout this thesis, it is needed to be explained some notations. Vectors in 3-dimensional space are represented with an over arrow ($\vec{x}$). Spatial vectors in 6-dimensional space are represented with over two arrows ($\vec{\vec{x}}$) and all of the other vectors are represented as underlined ($\underline{x}$).

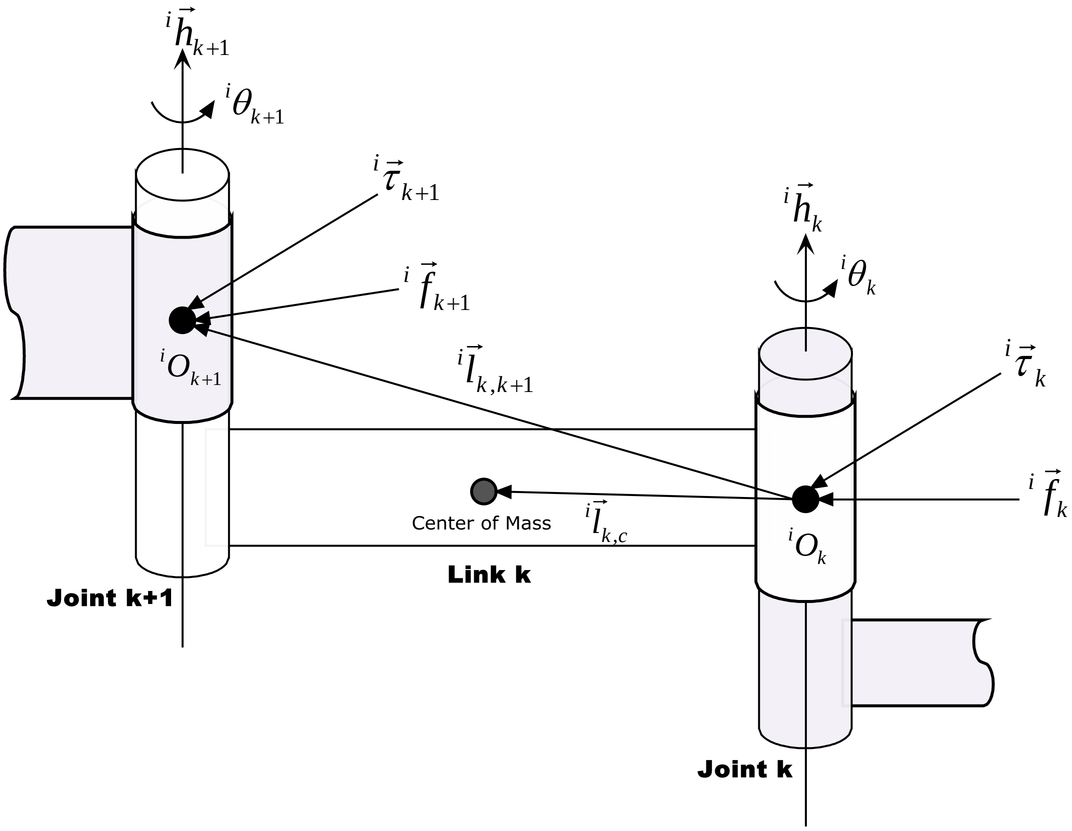


Figure 3.1 : Vectors associated with link k of the manipulator i.

A rigid link is shown in Figure 3.1. Angular and linear velocities of the k^{th} link for the i^{th} manipulator are represented ${}^{i}\vec{\omega}_k$ and ${}^{i}\vec{v}_k$ respectively. They propagate from link $k-1$ to link k for a revolute joint as follows:

$$ {}^{i}\vec{\omega}_k = {}^{i}\vec{\omega}_{k-1} + {}^{i}\vec{h}_k \, {}^{i}\dot{\theta}_k \tag{3.1} $$

$$ \begin{aligned} {}^{i}\vec{v}_k &= {}^{i}\vec{v}_{k-1} + {}^{i}\vec{\omega}_{k-1} \times {}^{i}\vec{\ell}_{k-1,k} = {}^{i}\vec{v}_{k-1} - {}^{i}\vec{\ell}_{k-1,k} \times {}^{i}\vec{\omega}_{k-1} \\ &= {}^{i}\vec{v}_{k-1} - {}^{i}\hat{\ell}_{k-1,k} \, {}^{i}\vec{\omega}_{k-1} \end{aligned} \tag{3.2} $$

where ${}^{i}\vec{h}_k$ is the axis of rotation vector of joint k and ${}^{i}\hat{\ell}_{k-1,k} \triangleq \left({}^{i}\vec{\ell}_{k-1,k} \times \right)$ is an operator called skew symmetric matrix which is given as:

$$ {}^{i}\hat{\ell}_{k-1,k} = \begin{bmatrix} 0 & -\,{}^{i}\vec{\ell}_{(k-1,k)_z} & {}^{i}\vec{\ell}_{(k-1,k)_y} \\ {}^{i}\vec{\ell}_{(k-1,k)_z} & 0 & -\,{}^{i}\vec{\ell}_{(k-1,k)_x} \\ -\,{}^{i}\vec{\ell}_{(k-1,k)_y} & {}^{i}\vec{\ell}_{(k-1,k)_x} & 0 \end{bmatrix} \tag{3.3} $$

where this is the reference frame:

$$ {}^{i}\vec{\ell}_{k-1,k} = \begin{bmatrix} {}^{i}\vec{\ell}_{(k-1,k)_x} \\ {}^{i}\vec{\ell}_{(k-1,k)_y} \\ {}^{i}\vec{\ell}_{(k-1,k)_z} \end{bmatrix} $$

If we unify the equations (3.1) and (3.2), we get:

$$ \begin{bmatrix} {}^{i}\vec{\omega}_k \\ {}^{i}\vec{v}_k \end{bmatrix} = \begin{bmatrix} {}_{3}I & {}_{3}0 \\ -\,{}^{i}\hat{\ell}_{k-1,k} & {}_{3}I \end{bmatrix} \begin{bmatrix} {}^{i}\vec{\omega}_{k-1} \\ {}^{i}\vec{v}_{k-1} \end{bmatrix} + \begin{bmatrix} {}^{i}\vec{h}_k \\ \vec{0} \end{bmatrix} {}^{i}\dot{\theta}_k \tag{3.4} $$

where, spatial velocity of link k is

$$ {}^{i}\vec{\vec{V}}_k \triangleq \begin{bmatrix} {}^{i}\vec{\omega}_k \\ {}^{i}\vec{v}_k \end{bmatrix} \tag{3.5} $$

and the link velocity propagation operator is

$$ {}^{i}\phi_{k,k-1} \triangleq \begin{bmatrix} {}_{3}I & {}_{3}0 \\ -\,{}^{i}\hat{\ell}_{k-1,k} & {}_{3}I \end{bmatrix} \tag{3.6} $$

and, finally, the axis of rotation spatial vector of joint k is defined as depicted;

$$ {}^{i}\vec{\vec{H}}_k \triangleq \begin{bmatrix} {}^{i}\vec{h}_k \\ \vec{0} \end{bmatrix} \tag{3.7} $$

Equation (3.7) is for revolute joints. If the joint is prismatic then ${}^{i}\vec{\vec{H}}_k$ is defined as

$$ {}^{i}\vec{\vec{H}}_k \triangleq \begin{bmatrix} \vec{0} \\ {}^{i}\vec{h}_k \end{bmatrix} \tag{3.8} $$

We can unify spatial velocity equation using definitions (3.5), (3.6), (3.7) and (3.8)

$$ {}^{i}\vec{\vec{V}}_k = {}^{i}\phi_{k,k-1} \; {}^{i}\vec{\vec{V}}_{k-1} + {}^{i}\vec{\vec{H}}_k \; {}^{i}\dot{\theta}_k \tag{3.9} $$

With this unified equation, spatial velocity (angular and linear velocities) of link $k-1$ propagated to the link k. This information helps us to model manipulators much more easily.

3.2 Serial Manipulator Kinematics on a Fixed Platform

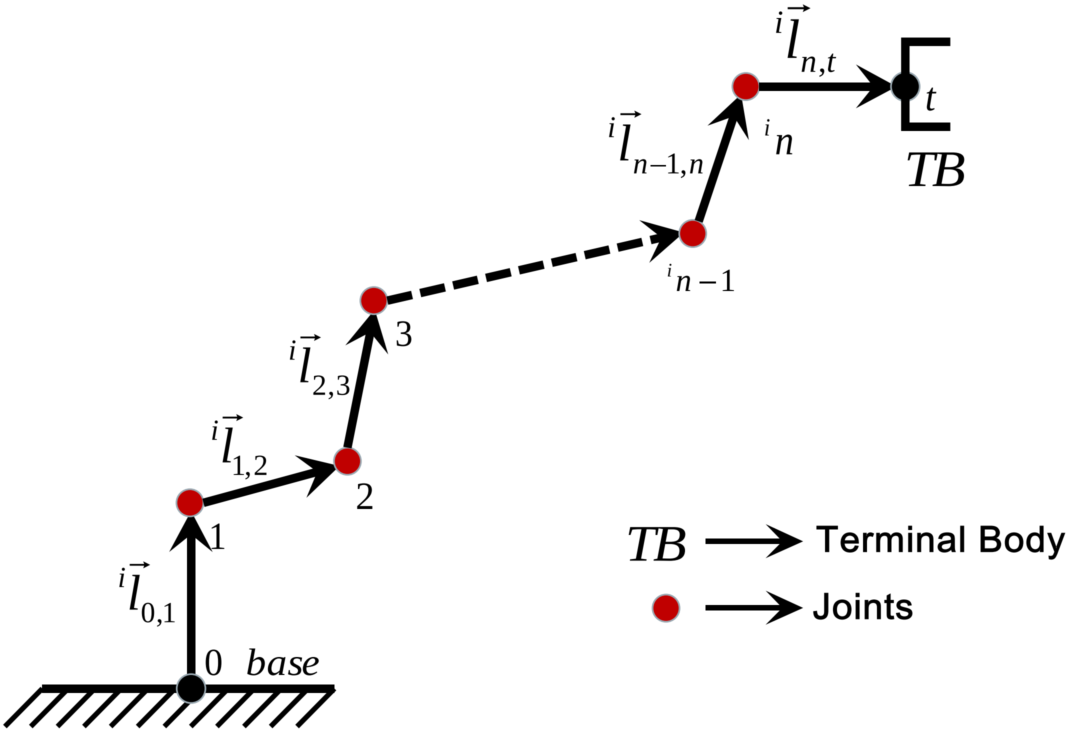


Figure 3.2 : Serial manipulator on a fixed platform.

A serial manipulator represents a serial topology system whose kinematics will be examined and explained in this section. We will propagate the spatial velocities of each joint from base to tip (terminal body) of serial robotic manipulator. We will use arm i notation and accept that robotic arm is on a fixed base.

In Figure 3.2, ${}^{i}n$ means the number of DOF of the i^{th} manipulator. Propagation of spatial velocities from base to the link number ${}^{i}n$ will be detailed. The manipulator is on a fixed platform which means that both angular and linear velocities for the base are zero.

$$\begin{gathered}
{}^{i}\vec{\vec{V}}_{0} = \vec{\vec{0}} \\
{}^{i}\vec{\vec{V}}_{1} = {}^{i}\phi_{1,0}\ {}^{i}\vec{\vec{V}}_{0} + {}^{i}\vec{\vec{H}}_{1}\ {}^{i}\dot{\theta}_{1} \\
{}^{i}\vec{\vec{V}}_{2} = {}^{i}\phi_{2,1}\ {}^{i}\vec{\vec{V}}_{1} + {}^{i}\vec{\vec{H}}_{2}\ {}^{i}\dot{\theta}_{2} \\
\vdots \\
{}^{i}\vec{\vec{V}}_{{}^{i}n} = {}^{i}\phi_{{}^{i}n,\,{}^{i}n-1}\ {}^{i}\vec{\vec{V}}_{{}^{i}n-1} + {}^{i}\vec{\vec{H}}_{{}^{i}n}\ {}^{i}\dot{\theta}_{{}^{i}n}
\end{gathered} \tag{3.10}$$

Using the state transition property of the propagation matrix, which is formulated as ${}^{i}\phi_{a,b}\ {}^{i}\phi_{b,c} = {}^{i}\phi_{a,c}$, we can rewrite the equations in (3.10) and we get:

$$\begin{gathered}
{}^{i}\vec{\vec{V}}_{0} = \vec{\vec{0}} \\
{}^{i}\vec{\vec{V}}_{1} = {}^{i}\vec{\vec{H}}_{1}\ {}^{i}\dot{\theta}_{1} \\
{}^{i}\vec{\vec{V}}_{2} = {}^{i}\phi_{2,1}\ {}^{i}\vec{\vec{H}}_{1}\ {}^{i}\dot{\theta}_{1} + {}^{i}\vec{\vec{H}}_{2}\ {}^{i}\dot{\theta}_{2} \\
{}^{i}\vec{\vec{V}}_{3} = {}^{i}\phi_{3,1}\ {}^{i}\vec{\vec{H}}_{1}\ {}^{i}\dot{\theta}_{1} + {}^{i}\phi_{3,2}\ {}^{i}\vec{\vec{H}}_{2}\ {}^{i}\dot{\theta}_{2} + {}^{i}\vec{\vec{H}}_{3}\ {}^{i}\dot{\theta}_{3} \\
\vdots \\
{}^{i}\vec{\vec{V}}_{{}^{i}n} = {}^{i}\phi_{{}^{i}n,1}\ {}^{i}\vec{\vec{H}}_{1}\ {}^{i}\dot{\theta}_{1} + \cdots + {}^{i}\phi_{{}^{i}n,\,{}^{i}n-1}\ {}^{i}\vec{\vec{H}}_{{}^{i}n-1}\ {}^{i}\dot{\theta}_{{}^{i}n-1} + {}^{i}\vec{\vec{H}}_{{}^{i}n}\ {}^{i}\dot{\theta}_{{}^{i}n}
\end{gathered} \tag{3.11}$$

Equations in (3.11) can be unified into one equation and can be rewritten in a matrix form as follows:

$$\begin{bmatrix} {}^{i}\vec{\vec{V}}_1 \\ {}^{i}\vec{\vec{V}}_2 \\ \vdots \\ {}^{i}\vec{\vec{V}}_{{}^{i}n} \end{bmatrix} = \begin{bmatrix} {}_{6}I & {}_{6}0 & \cdots & {}_{6}0 \\ {}^{i}\phi_{2,1} & {}_{6}I & \cdots & {}_{6}0 \\ \vdots & \vdots & \ddots & \vdots \\ {}^{i}\phi_{{}^{i}n,1} & {}^{i}\phi_{{}^{i}n,2} & \cdots & {}_{6}I \end{bmatrix} \begin{bmatrix} {}^{i}\vec{\vec{H}}_1 & \vec{\vec{0}} & \cdots & \vec{\vec{0}} \\ \vec{\vec{0}} & {}^{i}\vec{\vec{H}}_2 & \cdots & \vec{\vec{0}} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{\vec{0}} & \vec{\vec{0}} & \cdots & {}^{i}\vec{\vec{H}}_{{}^{i}n} \end{bmatrix} \begin{bmatrix} {}^{i}\dot{\theta}_1 \\ {}^{i}\dot{\theta}_2 \\ \vdots \\ {}^{i}\dot{\theta}_{{}^{i}n} \end{bmatrix} \tag{3.12}$$

where;

$$ {}^{i}\underline{V} = \begin{bmatrix} {}^{i}\vec{\vec{V}}_1 \\ {}^{i}\vec{\vec{V}}_2 \\ \vdots \\ {}^{i}\vec{\vec{V}}_{{}^{i}n} \end{bmatrix} \quad {}^{i}\phi = \begin{bmatrix} {}_{6}I & {}_{6}0 & \cdots & {}_{6}0 \\ {}^{i}\phi_{2,1} & {}_{6}I & \cdots & {}_{6}0 \\ \vdots & \vdots & \ddots & \vdots \\ {}^{i}\phi_{{}^{i}n,1} & {}^{i}\phi_{{}^{i}n,2} & \cdots & {}_{6}I \end{bmatrix}$$

$$ {}^{i}H = \begin{bmatrix} {}^{i}\vec{\vec{H}}_1 & \vec{\vec{0}} & \cdots & \vec{\vec{0}} \\ \vec{\vec{0}} & {}^{i}\vec{\vec{H}}_2 & \cdots & \vec{\vec{0}} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{\vec{0}} & \vec{\vec{0}} & \cdots & {}^{i}\vec{\vec{H}}_{{}^{i}n} \end{bmatrix} \quad {}^{i}\underline{\dot{\theta}} = \begin{bmatrix} {}^{i}\dot{\theta}_1 \\ {}^{i}\dot{\theta}_2 \\ \vdots \\ {}^{i}\dot{\theta}_{{}^{i}n} \end{bmatrix} \tag{3.13}$$

We can also write (3.12) in a compact form by the help of definitions in (3.13)

$$ {}^{i}\underline{V} = {}^{i}\phi \; {}^{i}H \; {}^{i}\underline{\dot{\theta}} \tag{3.14}$$

In equation (3.14), it is shown how to propagate the spatial velocities from base to link ${}^{i}n$ of arm i. It is also essential to illustrate how to propagate the spatial velocities from link ${}^{i}n$ to tip t of arm i.

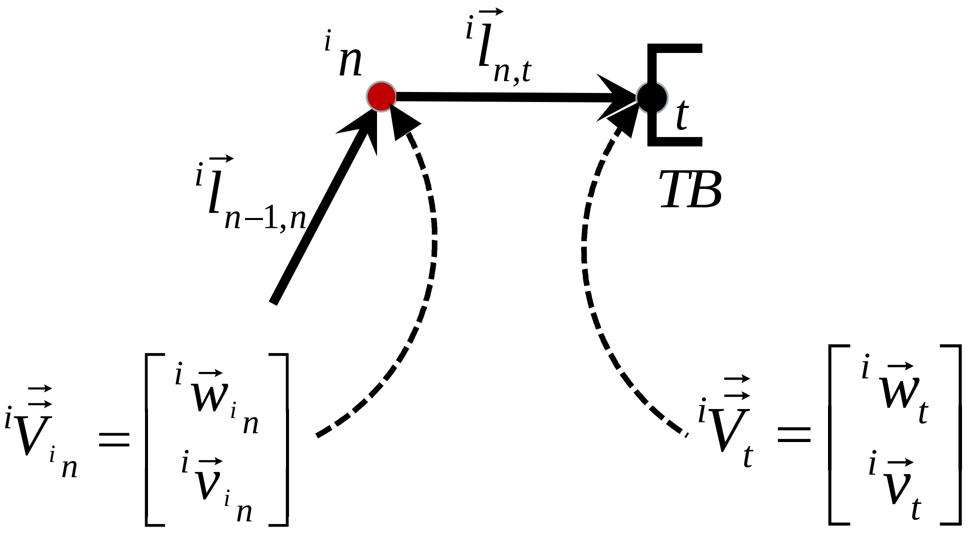

${}^{i}n$
${}^{i}\vec{l}_{n,t}$
t
TB
${}^{i}\vec{l}_{n-1,n}$
${}^{i}\vec{\vec{V}}_{{}^{i}n} = \begin{bmatrix} {}^{i}\vec{w}_{{}^{i}n} \\ {}^{i}\vec{v}_{{}^{i}n} \end{bmatrix}$
${}^{i}\vec{\vec{V}}_{t} = \begin{bmatrix} {}^{i}\vec{w}_{t} \\ {}^{i}\vec{v}_{t} \end{bmatrix}$

Figure 3.3 : Propagation from joint ${}^{i}n$ to tip t of arm i.

$$\begin{gathered} {}^{i}\vec{\omega}_t = {}^{i}\vec{\omega}_{i_n} \\ {}^{i}\vec{v}_t = {}^{i}\vec{v}_{i_n} + {}^{i}\vec{\omega}_{i_n} \times {}^{i}\vec{\ell}_{i_{n,t}} = {}^{i}\vec{v}_{i_n} - {}^{i}\hat{\ell}_{i_{n,t}}\, {}^{i}\vec{\omega}_{i_n} \end{gathered} \tag{3.15}$$

We could unify the equations in (3.15):

$$\begin{bmatrix} {}^{i}\vec{\omega}_t \\ {}^{i}\vec{v}_t \end{bmatrix} = \begin{bmatrix} {}_{3}I & {}_{3}0 \\ -\,{}^{i}\hat{\ell}_{i_{n,t}} & {}_{3}I \end{bmatrix} \begin{bmatrix} {}^{i}\vec{\omega}_{i_n} \\ {}^{i}\vec{v}_{i_n} \end{bmatrix} \tag{3.16}$$

where, the propagation operator ${}^{i}\phi_{t,\,i_n}$ is defined as:

$${}^{i}\phi_{t,\,i_n} = \begin{bmatrix} {}_{3}I & {}_{3}0 \\ -\,{}^{i}\hat{\ell}_{i_{n,t}} & {}_{3}I \end{bmatrix} \tag{3.17}$$

The equation (3.16) can also be rewritten in a compact form as:

$${}^{i}\vec{\vec{V}}_t = {}^{i}\phi_{t,\,i_n}\, {}^{i}\vec{\vec{V}}_{i_n} \tag{3.18}$$

Equation (3.18) can be written in terms of ${}^{i}\underline{V}$ instead of ${}^{i}\vec{\vec{V}}_{i_n}$ as:

$${}^{i}\vec{\vec{V}}_t = {}^{i}\phi_t\, {}^{i}\underline{V} \tag{3.19}$$

where:

$${}^{i}\phi_t = \begin{bmatrix} {}_{6}0 & {}_{6}0 & \cdots & {}^{i}\phi_{t,\,i_n} \end{bmatrix} \tag{3.20}$$

with the equations (3.14) and (3.19) we can deduce:

$${}^{i}\vec{\vec{V}}_t = {}^{i}\phi_t\, {}^{i}\phi\, {}^{i}H\, {}^{i}\underline{\dot{\theta}} \tag{3.21}$$

here we can define the Jacobian operator as:

$${}^{i}\mathcal{J} \triangleq {}^{i}\phi_t\, {}^{i}\phi\, {}^{i}H \tag{3.22}$$

${}^{i}\mathcal{J}$ is the Jacobian operator of the i^{th} manipulator. Jacobian operator is a linear operator which maps joint space to task space. Using the equations (3.21) and (3.22) we can write the general expression of tip velocity as:

$$ {}^{i}\vec{\vec{V}}_t = {}^{i}\mathcal{J} \; {}^{i}\underline{\dot{\theta}} \tag{3.23} $$

That is to say, if the velocities in the joint space are known, operational space velocities can be computed using Jacobian operator. In other saying, the angular and linear velocities of the tip point can be computed using joint velocities. That is referred to as forward kinematics.

Let us partition the six dimensional spatial space back to two 3 dimensional spaces for linear and angular velocities as follows:

$$ \begin{bmatrix} {}^{i}\vec{\omega}_t \\ {}^{i}\vec{v}_t \end{bmatrix} = \begin{bmatrix} {}^{i}\mathcal{J}_\omega \\ {}^{i}\mathcal{J}_v \end{bmatrix} \; {}^{i}\underline{\dot{\theta}} \tag{3.24} $$

where ${}^{i}\mathcal{J}_\omega$ and ${}^{i}\mathcal{J}_v$ are $3 \times n$ matrices (where n is the number of the DOF). These equations can be written as:

$$ \begin{aligned} {}^{i}\vec{\omega}_t &= {}^{i}\mathcal{J}_\omega \; {}^{i}\underline{\dot{\theta}} \\ {}^{i}\vec{v}_t &= {}^{i}\mathcal{J}_v \; {}^{i}\underline{\dot{\theta}} \end{aligned} \tag{3.25} $$

With this representation we can see the general formulation of angular and linear velocities due to the joint velocities.

What is referred to inverse kinematics is essentially the mapping of the velocity vector in joint space based on the spatial velocity of the tip point. This may require the pseudo inverse calculation of the Jacobian, which can be obtained as long as it is full-rank.

$$ {}^{i}\underline{\dot{\theta}} = {}^{i}\mathcal{J}^{\#} \; {}^{i}\vec{\vec{V}}_t \tag{3.26} $$

where ${}^{i}\mathcal{J}^{\#}$ is the pseudo-inverse of the Jacobian.

In the case of Jacobian being a square matrix, inverse of it can be directly computed under the same assumption that it is full-rank. For the sake of simplicity in the equations, we will not impose a restriction on the number of DOF to be equal to the number of dimensions of the task space. Therefore we consider pseudo inverse instead of straight inverse which also brings generality as well as simplicity in the equations.

3.3 Serial Manipulator Dynamics on a Fixed Platform

To make dynamic analysis, we need the time derivatives of angular and linear velocities;

$$ {}^{i}\dot{\vec{\omega}}_k = {}^{i}\dot{\vec{\omega}}_{k-1} + {}^{i}\vec{h}_k\,{}^{i}\ddot{\theta}_k + {}^{i}\vec{\omega}_{k-1} \times {}^{i}\vec{\omega}_k \tag{3.27} $$

$$ {}^{i}\dot{\vec{v}}_k = {}^{i}\dot{\vec{v}}_{k-1} - {}^{i}\vec{l}_{k-1,k} \times {}^{i}\dot{\vec{\omega}}_{k-1} + {}^{i}\vec{\omega}_{k-1} \times ({}^{i}\vec{\omega}_{k-1} \times {}^{i}\vec{l}_{k-1,k}) \tag{3.28} $$

These two above equations can be written as follows:

$$ {}^{i}\dot{\vec{\vec{V}}}_k = {}^{i}\phi_{k,k-1}\,{}^{i}\dot{\vec{\vec{V}}}_{k-1} + {}^{i}\vec{\vec{H}}_k\ddot{\theta}_k + {}^{i}\vec{\vec{a}}_k \tag{3.29} $$

${}^{i}\vec{\vec{a}}_k$ is the spatial bias accelerations of the arm i:

$$ {}^{i}\vec{\vec{a}}_k = \begin{bmatrix} {}^{i}\vec{\omega}_{k-1} \times {}^{i}\vec{\omega}_k \\ {}^{i}\vec{\omega}_{k-1} \times ({}^{i}\vec{\omega}_{k-1} \times {}^{i}\vec{l}_{k-1,k}) \end{bmatrix} \tag{3.30} $$

${}^{i}\underline{a}$ is the stacked bias spatial accelerations of the arm i:

$$ {}^{i}\underline{a} = \begin{bmatrix} {}^{i}\vec{\vec{a}}_1 \\ \vdots \\ {}^{i}\vec{\vec{a}}_{n_i} \end{bmatrix} \tag{3.31} $$

${}^{i}\vec{\vec{F}}_k$ is the link spatial forces matrix of the arm i:

$$ {}^{i}\vec{\vec{F}}_k = {}^{i}\phi^{T}_{k+1,k}\,{}^{i}\vec{\vec{F}}_{k+1} + {}^{i}M_k\,{}^{i}\dot{\vec{\vec{V}}}_k + {}^{i}\vec{\vec{b}}_k \tag{3.32} $$

${}^{i}\vec{\vec{F}}_k$ can also be written as:

$$ {}^{i}\vec{\vec{F}}_k = \begin{bmatrix} {}^{i}\vec{\tau}_k \\ {}^{i}\vec{f}_k \end{bmatrix} \tag{3.33} $$

where ${}^{i}\vec{\tau}_k$ is the torque vector of link k of arm i and ${}^{i}\vec{f}_k$ is the force vector of link k of arm i.

$$ {}^{i}\underline{F} = \begin{bmatrix} {}^{i}\vec{\vec{F}}_1 \\ \vdots \\ {}^{i}\vec{\vec{F}}_{n_i} \end{bmatrix} \tag{3.34} $$

In Equation 3.34, ${}^{i}\underline{F}$ is the stacked spatial forces of the arm i.

There is a formula defines the relation in between applied torques and link spatial forces. That is:

$$ {}^{i}\underline{\tau} = {}^{i}H^{T}\ {}^{i}\underline{F} \tag{3.35} $$

In order to find ${}^{i}\mathfrak{I}_k$ Inertia Matrix of link k of the arm i, let the mass density of the object be a function of position $\rho(x, y, z)$. Then the inertia tensor in the body attached frame can be computed as:

$$ {}^{i}\mathfrak{I}_k = \begin{bmatrix} {}^{i}\mathfrak{I}_{xx} & {}^{i}\mathfrak{I}_{xy} & {}^{i}\mathfrak{I}_{xz} \\ {}^{i}\mathfrak{I}_{yx} & {}^{i}\mathfrak{I}_{yy} & {}^{i}\mathfrak{I}_{yz} \\ {}^{i}\mathfrak{I}_{zx} & {}^{i}\mathfrak{I}_{zy} & {}^{i}\mathfrak{I}_{zz} \end{bmatrix} \tag{3.36} $$

where,

$$ \begin{gathered} {}^{i}\mathfrak{I}_{xx} = \iiint (y^2 + z^2)\rho(x, y, z)\, dx\, dy\, dz \\ {}^{i}\mathfrak{I}_{yy} = \iiint (x^2 + z^2)\rho(x, y, z)\, dx\, dy\, dz \\ {}^{i}\mathfrak{I}_{zz} = \iiint (x^2 + y^2)\rho(x, y, z)\, dx\, dy\, dz \\ {}^{i}\mathfrak{I}_{xy} = {}^{i}\mathfrak{I}_{yx} = -\iiint xy\rho(x, y, z)\, dx\, dy\, dz \\ {}^{i}\mathfrak{I}_{xz} = {}^{i}\mathfrak{I}_{zx} = -\iiint xz\rho(x, y, z)\, dx\, dy\, dz \\ {}^{i}\mathfrak{I}_{yz} = {}^{i}\mathfrak{I}_{zy} = -\iiint yz\rho(x, y, z)\, dx\, dy\, dz \end{gathered} \tag{3.37} $$

The integrals in the above expression are calculated over the region of space occupied by the rigid body. The diagonal elements of the inertia tensor ${}^{i}\mathfrak{I}_{xx}$, ${}^{i}\mathfrak{I}_{yy}$,

${}^i\Im_{zz}$ are called the *Principal Moments of Inertia* which refers to the x, y, z axes, respectively. The off diagonal terms ${}^i\Im_{xy}$, ${}^i\Im_{xz}$, etc., are called the *Cross Products of Inertia*. If the mass distribution of the body is symmetric with respect to the body attached frame then the cross products of inertia are identically zero, i.e. ${}^i\Im_{xy} = 0$, ${}^i\Im_{yz} = 0$ etc.

${}^iM_k$is the link mass matrix of link k of arm i:

$$ {}^iM_k = \begin{bmatrix} {}^i\Im_k & {}^im_k\ {}^iL_{k,c} \\ -\ {}^im_k\ {}^iL_{k,c} & {}_3I\ {}^im_k \end{bmatrix} \tag{3.38} $$

where I is the Identity Matrix (3x3), m_k is the mass vector of the link k of the arm i and $\Im_k$ is the Inertia Matrix of link k of the arm i.

${}^iM$ is the mass matrix of arm i:

$$ {}^iM = \begin{bmatrix} {}^iM_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {}^iM_{n_i} \end{bmatrix} \tag{3.39} $$

${}^i\vec{\vec{b}}_k$ is the bias spatial forces remainder term:

$$ {}^i\vec{\vec{b}}_k = \begin{bmatrix} {}^i\vec{\omega}_k \times\ {}^i\Im_k\ {}^i\vec{\omega}_k \\ {}^im_k\ {}^i\vec{\omega}_k \times (\ {}^i\vec{\omega}_k \times\ {}^i\vec{l}_{k,c}) \end{bmatrix} \tag{3.40} $$

where ${}^i\vec{l}_{k,c}$ is the link vector from the origin of the link frame k to the CM of the link.

${}^i\underline{b}$ is the stacked bias spatial forces of arm i:

$$ {}^i\underline{b} = \begin{bmatrix} {}^i\vec{\vec{b}}_1 \\ \vdots \\ {}^i\vec{\vec{b}}_{n_i} \end{bmatrix} \tag{3.41} $$

All of the above equations give us the final torque formula:

$$ {}^i\underline{\tau} = \ {}^i\mathcal{M}\ {}^i\underline{\ddot{\theta}} + \ {}^i\underline{C} + \ {}^i\mathcal{M}_b\ {}^i\vec{\vec{V}}_b + \ {}^i\mathcal{J}^T\ {}^i\vec{\vec{F}}_t \tag{3.42} $$

where, ${}^{i}\mathcal{M}$ is the generalized mass matrix of the arm i:

$$ {}^{i}\mathcal{M} = {}^{i}H^{T}\ {}^{i}\Phi^{T}\ {}^{i}M\ {}^{i}\Phi\ {}^{i}H \tag{3.43} $$

${}^{i}\underline{C}$ is the bias terms including Coriolis and gravity for the arm i:

$$ {}^{i}\underline{C} = {}^{i}H^{T}\ {}^{i}\Phi^{T}(\ {}^{i}M\ {}^{i}\Phi\ {}^{i}\underline{a} + {}^{i}\underline{b}) \tag{3.44} $$

${}^{i}\mathcal{M}_{b}$ is the mass matrix regarding the dynamic interaction between the base and the i^{th} arm:

$$ {}^{i}\mathcal{M}_{b} = {}^{i}H^{T}\ {}^{i}\Phi^{T}\ {}^{i}M\ {}^{i}\Phi\ {}^{i}\Phi_{b} \tag{3.45} $$

In this thesis, robotic serial manipulator on a fixed platform is studied. For the sake of clarity of nonlinear control, the preferred robot arm is located on a fixed platform. Furthermore, vast majority of industrial robots are installed on a fixed platform. For only one platform i in the formulas, as in ${}^{i}\underline{C}$, ${}^{i}\mathcal{M}$ etc. al.equal to zero. And also, because of the only platform being in the system is fixed, ${}^{i}\vec{\vec{V}}_{b}$ equals to zero matrix.

Kinematic and dynamic modeling of a serial manipulator on a fixed platform is now completed. We will use this knowledge to model a serial manipulator on a fixed platform in the following sections. For further details about kinematics and dynamics, the studies in the literature can also be studied [12, 13] and [25, 26, 27]. For further details, it is also important to look at the references as mentioned in the Introduction Section [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24].

In this thesis, we will also modify the nonlinear control theories to use with this novel robot manipulator modeling theories, which have only been used with D-H robot modeling theory before. Computed-torque control and adaptive control nonlinear theories and their adaptions will be explained and illustrated, respectively.

4. COMPUTED-TORQUE CONTROL OF ROBOTIC MANIPULATORS

In this chapter, we will examine a special feedback linearization type control scheme for robot manipulators that fall under the class known as “computed-torque controllers.” These generally perform well when the robot arm parameters are known quite accurately. Some connections are given with the novel robot modeling and advanced control techniques are provided as well.

4.1 Introduction

A fundamental problem in controlling robots is to make the manipulator to follow a desired trajectory. Before the robot start moving, we must position it in the right place at the right instances. In this chapter, we discuss computed-torque control, which yields a family of easy-to-understand control schemes that often work well and very fast in practice. These schemes involve the analysis of the controls design problem into an inner-loop design and an outer-loop design.

In this section, we show how to use SOA (Spatial Operator Algebra) robot modeling theory in conjunction with computed-torque control. Thus, this chapter could be considered as a bridge between SOA robot modeling algorithm and the advanced controller design techniques to obtain high performance in uncertain environments. We assume here the robot is moving in free space, having no contact with its environment. We will also assume in this chapter that the robot is a considerably well-known rigid system, thus design of controller is based on a well-known model. Control in the existence of uncertainties or unknown parameters (e.g., friction, payload mass) require refined approaches. This problem is dealt with using adaptive control in the following chapters.

Before we can control a robot arm, it is essential to know the desired path for performing a specific task. There are many issues associated with the path planning problem, such as avoiding obstacles, collision avoidance and making sure that the planned path does not require exceeding the voltage and torque limitations of the actuators. To reduce the control problem to its basic components to explain it clearly, it is necessary to assume that the ultimate control objective is to move the robot along a prescribed desired trajectory.

In this chapter, we do not concern ourselves with the actual trajectory-planning problem, however we show how to reconstruct a continuous desired path from a given table of desired points the end effector should pass through.

In most industrial applications, robot controllers are implemented on microprocessors, particularly in view of the complex nature of modern control schemes. In the following sections, we will demonstrate how to simulate a real industrial robot on a computer. This should also be done to verify the effectiveness of any proposed control scheme prior to actual implementation on a real robot manipulator.

4.2 Path Generation

Throughout this chapter, we assume that there is prescribed path $q_d(t)$ the robot arm should follow. We design control schemes that make the manipulator to follow this prescribed desired path or trajectory. *Trajectory Planning* involves finding the prescribed path and is usually considered as a separate design problem involving collision avoidance, concerns about actuator saturation, motor drive's torque and velocity limits and so on.

Trajectory planning is not one of the main concerns for this thesis. However, we need to use feedbacks of trajectory planning. That is why, it is necessary to mention about the *Path Planning*.

First of all, for a specific task, robot manipulators' tip point should have a path in 3-dimensional position space. Then it should be decided at which velocities we will track this path. Then, inverse kinematics will scatter these tip point velocities to the actuators in 6-dimensional velocity space. Thus, joint space velocities can be produced.

In this thesis, there are two different path-tracking simulations. The first one is about the joint space trajectory planning and the second one is about the operational space trajectory which could be called as Cartesian space trajectory too. In computed torque control and adaptive control, we use joint space position, velocity and acceleration values. These values could be taken from joint space trajectory for a given joint space trajectory directly. Nevertheless, for an operational space trajectory, these values have to be deduced with trajectory planning algorithms.

4.3 Computed-Torque Control

In the history of robotics, many sorts of robot control schemes have been offered. Most of them can be considered as special cases of the class of computed-torque controllers. Computed torque can be considered as a special application of feedback linearization of nonlinear systems, which has gained popularity in modern systems theory [Hunt et al. 1983], [Gilbert and Ha 1984]. In fact, one way to classify robot control schemes is to divide them as “computed-torque-like” or “noncomputed-torque-like” [LEWIS, 2004]. Computed-torque like controls can appear in robust control, adaptive control, and fuzzy logic control and so on.

Computed-torque control allows us to derive very effective robot controllers, conveniently. In the meanwhile, this method provides a framework to bring together classical independent joint control and some modern design techniques.

4.3.1 Derivation of inner feedforward loop

The robot arm dynamics are as it was stated in eq. (3.40);

$$ {}^{i}\underline{\tau} = {}^{i}\mathcal{M}(q)\ {}^{i}\underline{\ddot{\theta}} + {}^{i}\underline{C}(q,\dot{q}) + {}^{i}\mathcal{M}_b(q)\ {}^{i}\vec{\vec{V}}_b(q,\dot{q}) + {}^{i}\mathcal{J}^T\ {}^{i}\vec{\vec{F}}_t + {}^{i}\underline{\tau}_d \tag{4.1} $$

with the joint variable $q(t)\epsilon R^n$, ${}^{i}\underline{\tau}_d$ a disturbance. In our system, ${}^{i}\vec{\vec{V}}_b(q,\dot{q}) = 0$ and ${}^{i}\vec{\vec{F}}_t = 0$. Thus the robot arm dynamic equation could be written as;

$$ {}^{i}\underline{\tau} = {}^{i}\mathcal{M}(q)\ {}^{i}\underline{\ddot{\theta}} + {}^{i}\underline{C}(q,\dot{q}) + {}^{i}\underline{\tau}_d \tag{4.2} $$

Let us assume a predetermined desired trajectory in the joint space $q_d(t)$ has been given. Trajectory tracking error would be;

$$ e(t) = q_d(t) - q(t) \tag{4.3} $$

Velocity tracking error:

$$ \dot{e}(t) = \dot{q}_d(t) - \dot{q}(t) \tag{4.4} $$

Acceleration tracking error:

$$\ddot{e}(t) = \ddot{q}_d(t) - \ddot{q}(t) \tag{4.5}$$

Using eq. (4.2) and eq. (4.4) we get:

$$\ddot{e} = \ddot{q}_d + {}^{i}\mathcal{M}^{-1}({}^{i}\underline{C} + {}^{i}\underline{\tau}_d - {}^{i}\underline{\tau}) \tag{4.6}$$

Control input function:

$$u = \ddot{q}_d + {}^{i}\mathcal{M}^{-1}({}^{i}\underline{C} - {}^{i}\underline{\tau}) \tag{4.7}$$

Disturbance function:

$$\omega = {}^{i}\mathcal{M}^{-1}\ {}^{i}\underline{\tau}_d \tag{4.8}$$

and we could define a state $x(t)\epsilon R^{2n}$ by:

$$x = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{4.9}$$

The tracking error dynamics:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} \omega \tag{4.10}$$

This is a linear error system in *Brunovsky Canonical Form* consisting of n pairs of double integrators $1/s^2$, one per joint [Lewis, 2004]. In (4.10) the control input is u(t) and the disturbance is $\omega(t)$.

The feedback linearizing transformation (4.7) could be written as

$${}^{i}\underline{\tau} = {}^{i}\mathcal{M}(\ddot{q}_d - u) + {}^{i}\underline{C} \tag{4.11}$$

Equation (4.11) is *the computed-torque law*.

It can be simply shown that, if we select a control $u(t)$ that stabilizes (4.10), this will make $e(t)$ converge to zero. Thus, the nonlinear control input ${}^{i}\underline{\tau}$ (4.11) can track trajectory in the robot arm (4.1).

Nonlinear transformation, which illustrated in Fig. 4.1, has converted a complicated nonlinear controls design problem into a n decoupled linear subsystem design problem.

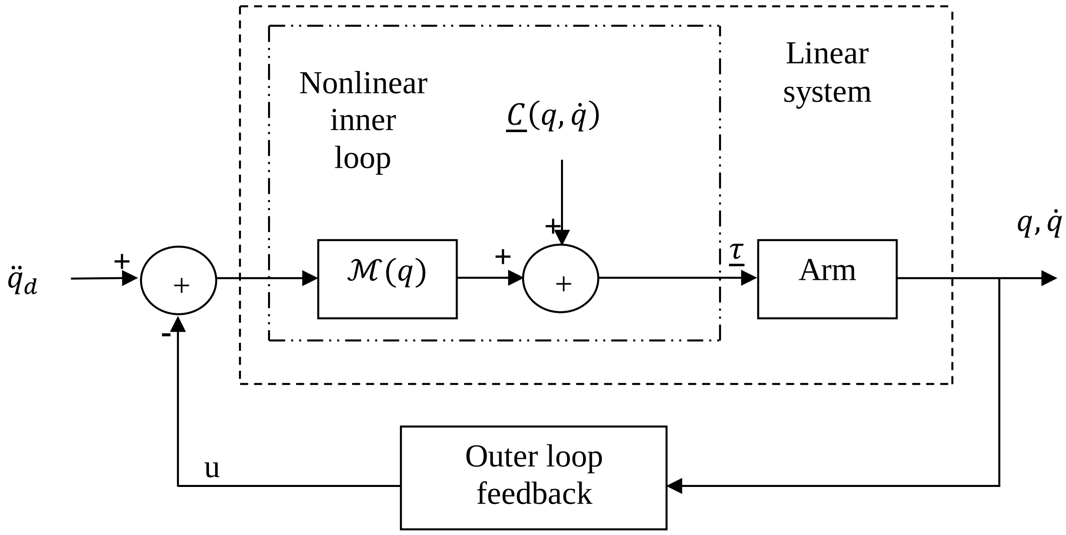


Figure 4.1 : Computed-torque control scheme showing inner and outer loops.

Computed-torque method depends on the inversion of the robot dynamics. That is why it could also be named as *inverse dynamic control* in the literature. Also, in the literature, abbreviation of inverse dynamic control is IDC. But, computed-torque term is much more common.

4.3.2 PD outer-loop design

The auxiliary control signal $u(t)$ could be selected as the proportional-plus derivative (PD) feedback:

$$u = -K_d\dot{e} - K_p e \tag{4.12}$$

Then Computed-Torque Law turns into:

$${}^{i}\underline{\tau} = {}^{i}\mathcal{M}(q)(\ddot{q}_d + K_d\dot{e} + K_p e) + {}^{i}\underline{C}(q,\dot{q}) \tag{4.13}$$

The closed-loop error dynamics are:

$$\ddot{e} + K_d\dot{e} + K_p e = \omega \tag{4.14}$$

or in state-space form:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix}\omega \tag{4.15}$$

The closed-loop characteristic polynomial is:

$$\Delta_c(s) = \left| s^2 I + K_d s + K_p \right| \tag{4.16}$$

It is usual to take the $n \times n$ gain matrices as PD Gains diagonal so that:

$$K_d = diag\{k_{d_i}\} \tag{4.17}$$

$$K_p = diag\{k_{p_i}\} \tag{4.18}$$

then:

$$\Delta_c(s) = \prod_{i=1}^{n} (s^2 + k_{d_i} s + k_{p_i}) \tag{4.19}$$

and the error is asymptotically stable as long as both of the K_{di} and K_{pi} are positive. Therefore, as long as the disturbance $\omega(t)$ is bounded, so is the error $e(t)$.

The standard form for the second-order characteristic polynomial:

$$p(s) = s^2 + 2\zeta\omega_n s + \zeta_n^2 \tag{4.20}$$

with ζ the damping ratio and ω_n the natural frequency. Therefore, to achieve desired performance in each component's PD gains of the error $e(t)$ should be selected according to the following equations:

$$k_{p_i} = \omega_n^2, \quad k_{d_i} = 2\zeta\omega_n \tag{4.21}$$

with *n* the desired damping ratio and natural frequency for joint error i.

It is undesirable for the robot to exhibit overshoot, since this could cause impact. Therefore, the PD gains are usually selected for *critical damping* $\zeta = 1$.

4.4 Stability of the Computed Torque Control Law

If $K_p, K_d \in \mathbb{R}^{n\times n}$ are positive definite and symmetric matrices, then we could say that the control law results in exponential trajectory tracking. The proof will be explained.

Proof: The error dynamics can be written in the form of first-order linear system:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}}_{A}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{4.22}$$

It will be enough to show that each of the eigenvalues of A has negative real part. In the case of $\lambda \in \mathbb{C}$ to be an eigenvalue of A with corresponding eigenvector

$v = (v_1, v_2) \in \mathbb{C}^{2n},\ v \neq 0$. Then,

$$\lambda\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_2 \\ -K_p v_1 - K_d v_2 \end{bmatrix} \tag{4.23}$$

If $\lambda = 0$ then $v = 0$ and hence $\lambda = 0$ is not an eigenvalue of A. Furthermore, if $\lambda \neq 0$, then $v_2 = 0$ refers that $v_1 = 0$. Thus, $v_1, v_2 \neq 0$ and we may assume without loss of generality that $\|v_1\| = 1$. Thus, we can write

$$\lambda^2 = v_1^*\lambda^2 v_1 = v_1^*\lambda v_2 = v_1^*\left(-K_p v_1 - K_d v_2\right) = -v_1^* K_d v_1 - \lambda v_1^* K_d v_1 \tag{4.24}$$

where $*$ denotes complex conjugate transpose. Since $\alpha = v_1^* K_p v_1 > 0$ and

$\beta = v_1^* K_v v_1 > 0$, we have

$$\lambda^2 + \alpha\lambda + \beta = 0 \qquad \alpha, \beta > 0 \tag{4.25}$$

and hence the real part of λ is negative. □

The power of the computed torque control law comes from converting a nonlinear dynamical system into a linear one. It allows the usage of any of a number of strong linear control synthesis tools. Computed torque is a special type of a more general technique known as feedback linearization, where a nonlinear system is rendered linear via full-state nonlinear feedback. One disadvantage of using feedback linearization is that it can be demanding for processors and can increase computation time. For robot manipulators, unboundedness of the inputs is rarely a problem since the inertia matrix of the system is bounded and hence the control torques which must be exerted always remain bounded [8]. In addition to that, different studies, simulations and experimental results show that the computed torque controller has very good performance characteristics. That is why this method is becoming increasingly popular.

5. ADAPTIVE CONTROL OF ROBOTIC MANIPULATORS

In this chapter, proposed adaptive controller is formulated based on separating unknown constant parameters from known time functions in the robot dynamic equation.

5.1 Introduction

Many researchers have focused on the problem of designing adaptive control laws for rigid-robot manipulators that ensure asymptotic trajectory tracking. The development of effective adaptive controllers represents an important step toward high-speed/precision robotic applications [Lewis, 2004]. Even in a well-structured industrial facility, robots could face different uncertainties. These uncertainties may come from grasped loads (e.g., unknown moments of inertia). These parameters are difficult to compute or measure and also some of the uncertainties may change during the time. That is why these uncertainties are inevitable and limit the potential for robots to manipulate accurately objects of considerable size and weight. The accuracy and repeatability in high-speed applications is greatly affected by parametric uncertainties. Adaptive strategies can compensate this parametric uncertainty of robotic manipulators.

5.2 Adaptive Control by a Computed-Torque Approach

In real world applications, exact knowledge of the robot model cannot be known due to many problems associated with model formulation. Two common uncertainties are link masses due to payload disturbances and unknown friction coefficients.

For a powerful Adaptive Control it is necessary change our parameter estimates based on an adaptive update rule that would be a function of the robot configuration and the tracking error. The update rule is derived from the stability analysis of the tracking error system. We ensure stability of the tracking error system by formulating the adaptive update rule and by analyzing the stability of the tracking error system at the same time [Lewis, 2004].

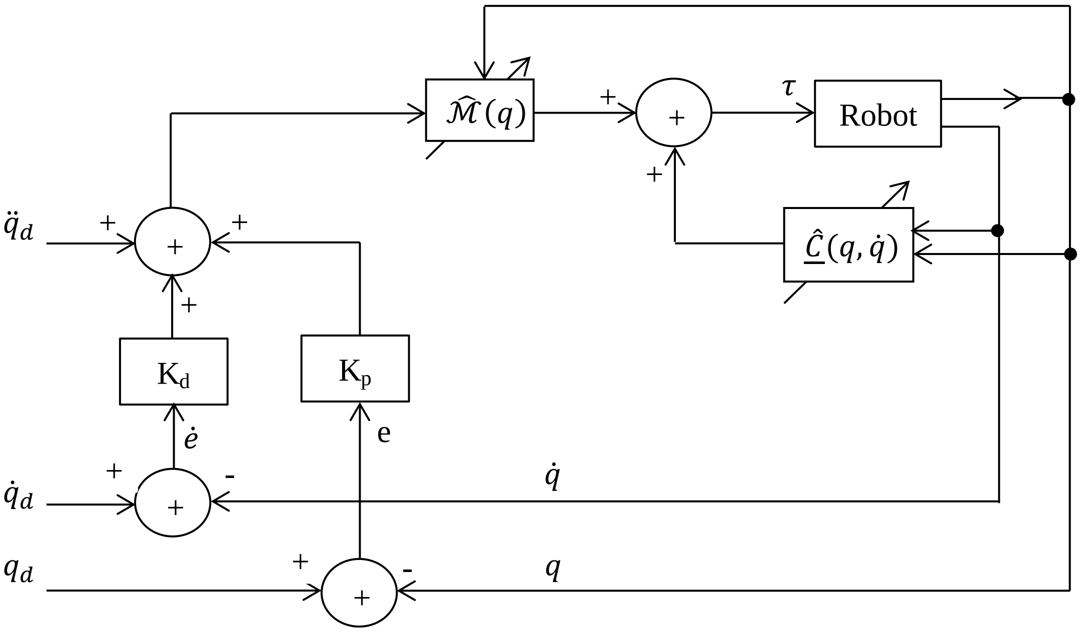


Figure 5.1 : Adaptive computed torque control block scheme.

First of all, we should form the tracking error system to study of the adaptive computed-torque controller. The first adaptive control strategy that we will examine is the method outlined in [Craig 1985]. From [Craig 1985], we can write the robot dynamic equation:

$$\tau = W(q,\dot{q},\ddot{q})\varphi \tag{5.1}$$

where φ is an $r \times 1$ vector of unknown constant parameters and $W(q,\dot{q},\ddot{q})$ is an $n \times r$matrix of known time functions. This property is crucial. Because it illustrates how to separate unknown parameters and the known time functions.

From [Craig 1985], the adaptive computed-torque controller for D-H is proposed by the following formulation:

$$\tau = \hat{M}(q)\left(\ddot{q}_d + K_d\dot{e} + K_p e\right) + \hat{V}_m(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \tag{5.2}$$

where the superscript “ $\widehat{}$ ” denotes the estimated dynamics with the unknown actual parameters replaced by the parameter estimates. The estimated values come from *Adaptive Update Rule*.

For SOA , equation (5.2) turns into:

$$ {}^{i}\underline{\tau} = {}^{i}\widehat{\mathcal{M}}(q)(\ddot{q}_d + K_d\dot{e} + K_p e) + {}^{i}\underline{\hat{C}}(q,\dot{q}) \tag{5.3} $$

Equation (5.2) can be derived as:

$$ {}^{i}\underline{\tau} = {}^{i}\widehat{\mathcal{M}}(q)(\ddot{e} + K_d\dot{e} + K_p e) + W(q,\dot{q},\ddot{q})\hat{\varphi} \tag{5.4} $$

where $\hat{\varphi}$ is an $n \times 1$ vector which represents a time-varying estimate of the unknown constant parameters. The tracking error system:

$$ \ddot{e} + K_d\dot{e} + K_p e = \widehat{\mathcal{M}}^{-1}(q)W(q,\dot{q},\ddot{q})\tilde{\varphi} \tag{5.5} $$

where $\tilde{\varphi}$ called is the parameter error vector:

$$ \tilde{\varphi} = \varphi - \hat{\varphi} \tag{5.6} $$

where the superscript “$\sim$” denotes the difference in between varying estimate of the unknown constant parameters and real values of the constant parameters.

Now for convenience, rewrite (5.5) in the state-space form:

$$ \dot{\psi} = A\psi + B\widehat{\mathcal{M}}^{-1}(q)W(q,\dot{q},\ddot{q})\tilde{\varphi} \tag{5.7} $$

where the tracking error vector is:

$$ \psi = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{5.8} $$

and

$$ B = \begin{bmatrix} {}_{n}0 \\ {}_{n}I \end{bmatrix}, \quad A = \begin{bmatrix} {}_{n}0 & {}_{n}I \\ -K_p & -K_d \end{bmatrix} \tag{5.9} $$

where ${}_{n}I$ is $n \times n$ identity matrix and ${}_{n}0$ is $n \times n$ zero matrix.

In this thesis, *Lyapunov stability analysis* is used to show that the tracking error vector ψ is asymptotically stable with the right choice of adaptive update law.

First of all, a positive-definite candidate Lyapunov function should be proposed:

$$\boldsymbol{V} = \psi^T P \psi + \tilde{\varphi}^T \Gamma^{-1} \tilde{\varphi} \tag{5.10}$$

where $\boldsymbol{V}$ denotes the *Candidate Lyapunov* function, P is an $2n \times 2n$ positive-definite, constant, symmetric matrix, and Γ is a diagonal, positive-definite $r \times r$ matrix.

Γ can be written as:

$$\Gamma = diag(\gamma_1, \gamma_2, \dots, \gamma_r) \tag{5.11}$$

where γ_i's are positive scalar constants.

If we differentiate the positive-definite Lyapunov-like function:

$$\dot{\boldsymbol{V}} = \psi^T P \dot{\psi} + \dot{\psi}^T P \psi + 2\tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \tag{5.12}$$

To derive the above equation we have used:

$$\left[\dot{\tilde{\varphi}}^T \Gamma^{-1} \tilde{\varphi}\right]^T = \tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \tag{5.13}$$

since $\Gamma = \Gamma^T$. We could write:

$$\begin{aligned} \dot{\boldsymbol{V}} = {} & \psi^T P\left(A\psi + B\hat{\mathcal{M}}^{-1}(q)W(.)\tilde{\varphi}\right) + \left(A\psi + B\hat{\mathcal{M}}^{-1}(q)W(.)\tilde{\varphi}\right)^T P\psi \\ & + 2\tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \end{aligned} \tag{5.14}$$

and

$$\dot{\boldsymbol{V}} = -\psi^T Q \psi + 2\tilde{\varphi}^T (\Gamma^{-1} \dot{\tilde{\varphi}} + W^T(.)\hat{\mathcal{M}}^{-1}(q) B^T P \psi) \tag{5.15}$$

where Q is a positive-definite symmetric matrix that satisfies the Lyapunov equation:

$$A^T P + PA = -Q \tag{5.16}$$

To ensure stability, it is always desirable to have the candidate Lyapunov function $\dot{\boldsymbol{V}}$ at least negative semidefinite; therefore, the choice of adaptation update rule deduced, by substituting:

$$\dot{\tilde{\varphi}} = -\Gamma W^T(.)\hat{\mathcal{M}}^{-1}(q) B^T P \psi \tag{5.17}$$

Then, in the Equation 5.17, $\dot{\tilde{\varphi}}$ term causes the candidate Lyapunov function $\dot{\boldsymbol{V}}$ turn into:

$$\dot{\boldsymbol{V}} = -\psi^T Q \psi \tag{5.18}$$

We must do further analysis in order to determine the type of stability. However, (5.17) gives the adaptive update rule for the parameter estimate vector since is equal to zero. Thus, by recalling that the actual unknown parameters are constant, we can obtain the adaptive update rule:

$$\dot{\hat{\varphi}} = \Gamma W^T(.)\widehat{\mathcal{M}}^{-1}(q)B^T P\psi \tag{5.19}$$

for the parameter estimate vector $\hat{\varphi}$.

It can be seen that $\dot{\boldsymbol{V}}$ is negative semidefinite and $\boldsymbol{V}$ is lower bounded by zero. Thus, $\boldsymbol{V}$ remains upper bounded in the time interval $[0, \infty)$

$$\lim_{t\to\infty} \boldsymbol{V} = \boldsymbol{V}_\infty \tag{5.20}$$

where $\boldsymbol{V}_\infty$ is a positive scalar constant.

$\boldsymbol{V}$ is upper bounded, ψ and $\tilde{\varphi}$ are bounded, which also shows that q, $\dot{q}$ and $\hat{\varphi}$ are bounded. We have already assumed $\hat{\varphi}$ is bounded, and we will always assume that the desired trajectory and its first two derivatives are bounded.

Using (5.2) we can write:

$$^{i}\underline{\ddot{\theta}} = {}^{i}\mathcal{M}^{-1}(q)\left({}^{i}\underline{\tau} - {}^{i}\underline{C}(q, \dot{q}) - {}^{i}\underline{\tau}_d\right) \tag{5.21}$$

Based on the equation 5.21, it can be easily said that, $\ddot{q}$ is bounded looking at the fact that the right side of this equation consists of only bounded quantities, such as ${}^{i}\underline{\tau}_d$ and ${}^{i}\underline{\tau}$.

If $\ddot{q}$ is bounded, (5.7) shows that the tracking error vector $\dot{\psi}$ is bounded. If $\dot{\psi}$ is bounded, we can also say that, the first derivative of the Lyapunov function $\dot{\boldsymbol{V}}$ is bounded, looking at the equation (5.18). Therefore, since the Lyapunov function $\boldsymbol{V}$ is lower bounded by zero, $\dot{\boldsymbol{V}}$ is negative semidefinite and $\ddot{\boldsymbol{V}}$ is bounded, then using *Barbalat's lemma*:

$$\lim_{t\to\infty} \dot{\boldsymbol{V}} = 0 \qquad \textbf{(5.22)}$$

which means that by the *Rayleigh-Ritz Theorem* [Lewis, 2004].

$$\lim_{t\to\infty} \lambda_{min}\{Q\}\|\psi\|^2 = 0 \; which\ means\ that \; \lim_{t\to\infty} \psi = 0 \qquad \textbf{(5.23)}$$

5.3 Adaptive Control Summary

5.3.1 Torque controller

Torque controller calculates torque values in order to send torque reference to the servo motor drives for real systems. For simulations, this torque vector is the input for Inverse Dynamics to compute the joint accelerations, velocities and positions.

Torque controller gets estimated values, which is shown as the symbol "$\widehat{\ }$",from *Adaptive Control Update Rule*:

$$^{i}\underline{\tau} = \; ^{i}\widehat{\mathcal{M}}(q)(\ddot{q}_d + K_d\dot{e} + K_p e) + \; ^{i}\underline{\hat{C}}(q,\dot{q}) \qquad \textbf{(5.24)}$$

5.3.2 Update rule

Adaptive Control Update Rule calculates the first derivative of estimated constant vector shown as $\dot{\hat{\varphi}}$.

$$\dot{\hat{\varphi}} = \Gamma W^T(q,\dot{q},\ddot{q})\widehat{\mathcal{M}}^{-1}(q)B^T P\psi \qquad \textbf{(5.25)}$$

where:

$$\Gamma = diag(\gamma_1, \gamma_2, \dots, \gamma_r) \qquad \textbf{(5.26)}$$

$$W(q,\dot{q},\ddot{q}) = \tau\varphi^{\#} \qquad \textbf{(5.27)}$$

$$B = \begin{bmatrix} {}_n0 \\ {}_nI \end{bmatrix} \qquad \textbf{(5.28)}$$

$$P = \begin{bmatrix} P_1\,{}_nI & P_2\,{}_nI \\ P_2\,{}_nI & P_3\,{}_nI \end{bmatrix} = \frac{1}{4}\begin{bmatrix} (2k_p + k_v)\,{}_nI & {}_nI \\ {}_nI & 2\,{}_nI \end{bmatrix} \qquad \textbf{(5.29)}$$

$$\psi = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{5.30}$$

$$\hat{\varphi} = \begin{bmatrix} \hat{m}_1 \\ \vdots \\ \hat{m}_n \end{bmatrix} \tag{5.31}$$

$$A = \begin{bmatrix} {}_n0 & {}_nI \\ -K_p & -K_v \end{bmatrix} \tag{5.32}$$

$$Q = \begin{bmatrix} 0.5 * k_p \, {}_nI & {}_n0 \\ {}_n0 & (K_d + 0.5) \, {}_nI \end{bmatrix} \tag{5.33}$$

$$W(q, \dot{q}, \ddot{q})\hat{\varphi} = \widehat{\mathcal{M}}(q)\ddot{q} + \underline{\hat{C}}(q, \dot{q}) + \hat{F}(\dot{q}) \tag{5.34}$$

We assume that $\hat{F}(\dot{q}) = 0$, then we get

$$W(q, \dot{q}, \ddot{q})\hat{\varphi} = \widehat{\mathcal{M}}(q)\ddot{q} + \underline{\hat{C}}(q, \dot{q}) \tag{5.35}$$

$$A^T P + PA = -Q \tag{5.36}$$

for some positive-definite, symmetric matrices P and Q.

P is symmetric and that it is positive definite if k_v is selected to be greater than 1 (for further details, look at the *Gerschgorin Theorem*) [Lewis, 2004].

5.3.3 Stability

Tracking error vector ψ is asymptotically stable.

5.3.4 Restrictions

Parameter resetting method and measurement of $\ddot{q}$ is required.

6. SIMULATION STUDIES

In this chapter; features, specification and dimensions of an industrial robot explained in details.

In this chapter, two different kinds of simulations are presented. Under all circumstances, “*Adaptive Control*” and “*Computed-Torque Control*” results are compared.

In the first simulation type, joint space trajectory tracking results are compared with and without the existence of uncertainties. In the second simulation type, operational space trajectory tracking results are compared with and without the existence of uncertainties.

6.1 Introduction about Simulations

In the previous chapters, various nonlinear control methods are illustrated. Using the methodology presented in chapter 2 and 3 the robot is modeled. And also using the methodology presented in chapter 4 and 5, the controller are designed.

The system was simulated using MATLAB. MATLAB is chosen for simulations because animation applications can be easily made by using virtual reality toolbox of MATLAB.

In this assignment we used IR6620 serial arm manipulator produced by ABB. 3D CAD models of the robot arm are downloaded from vendor’s official site.

To determine Masses, Center of Masses etc. we have used vendor’s official product specification document.

However, in the case of being missing values or vendor’s catalogue is lack of some critical information; SolidWorks could be used to obtain dynamic parameters such as CM (Center of Mass), mass and Inertia matrices.

6.2 ABB IRB 6620

IRB 6620 is part of the IRB 6600 family produced by ABB. IRB 6620 is a flexible and agile robot with a large workspace for industrial applications. The robot have the different mounting capabilities, i.e. floor standing, tilted or inverted mounted and shelf capability.

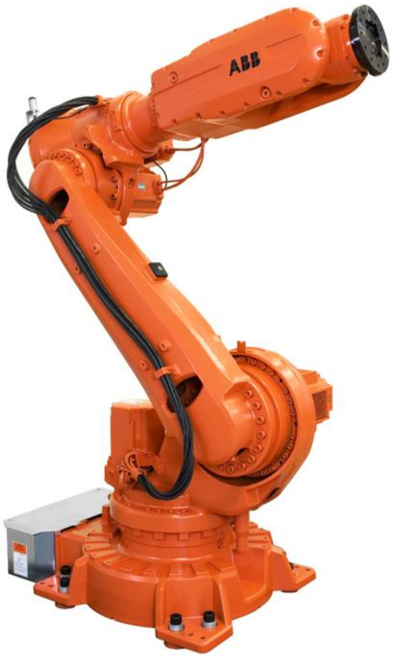


Figure 6.1 : ABB IRB 6620 (Floor mounted) [35].

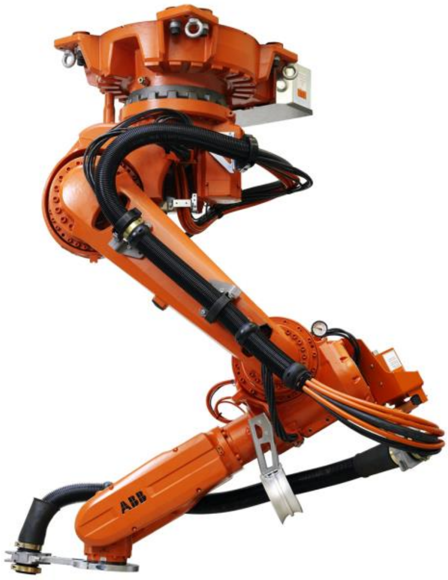


Figure 6.2 : ABB IRB 6620 (Ceil mounted) [35].

Working range is also another criterion defining the design success for robot manipulators.

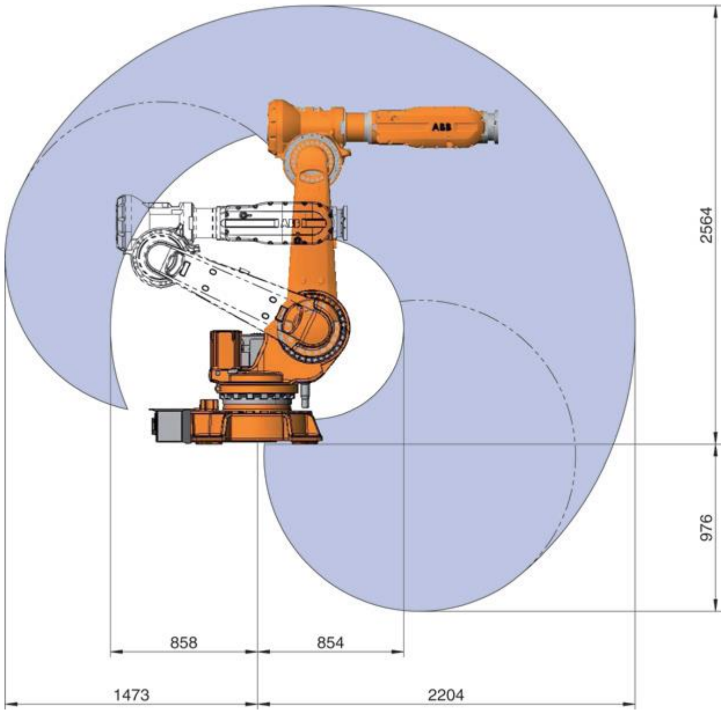


Figure 6.3 : Working range of IRB 6620 for floor mounted type [35].

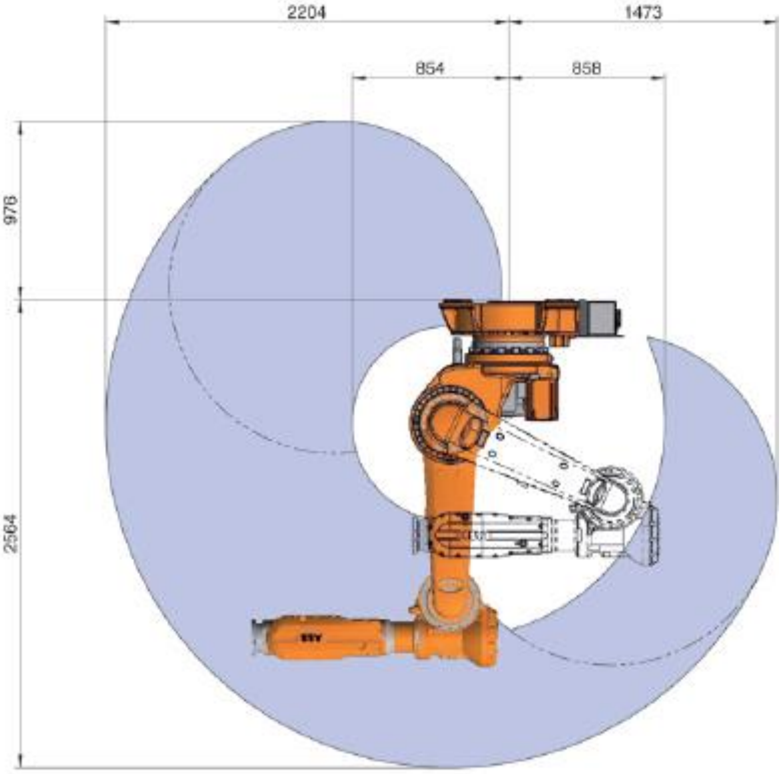


Figure 6.4 : Working range of IRB 6620 for ceil mounted type [35].

6.2.1 Specifications and dimensions of ABB IRB6620

This compact robot opens up opportunities for new flexible and improved line concepts like saving floor space, creating higher robot density and shorter lines.

IRB 6620 is also suitable for machine tending applications such as Die Casting and Injection Molding.

Table 6.1 : Technical specification data of IRB 6620 [34].

SPECIFICATION	
Reach	**2.2 m**
Handling Capacity	**150 kg**
Extra Loads can be mounted on to the robot	**50 kg on to the upper and 100 kg on to the robot base**
Number of Axes	**6**
Protection	**IP 54**
Mounting	**Floor, tilted or inverted**
Position Repeatability	**0.1 mm**

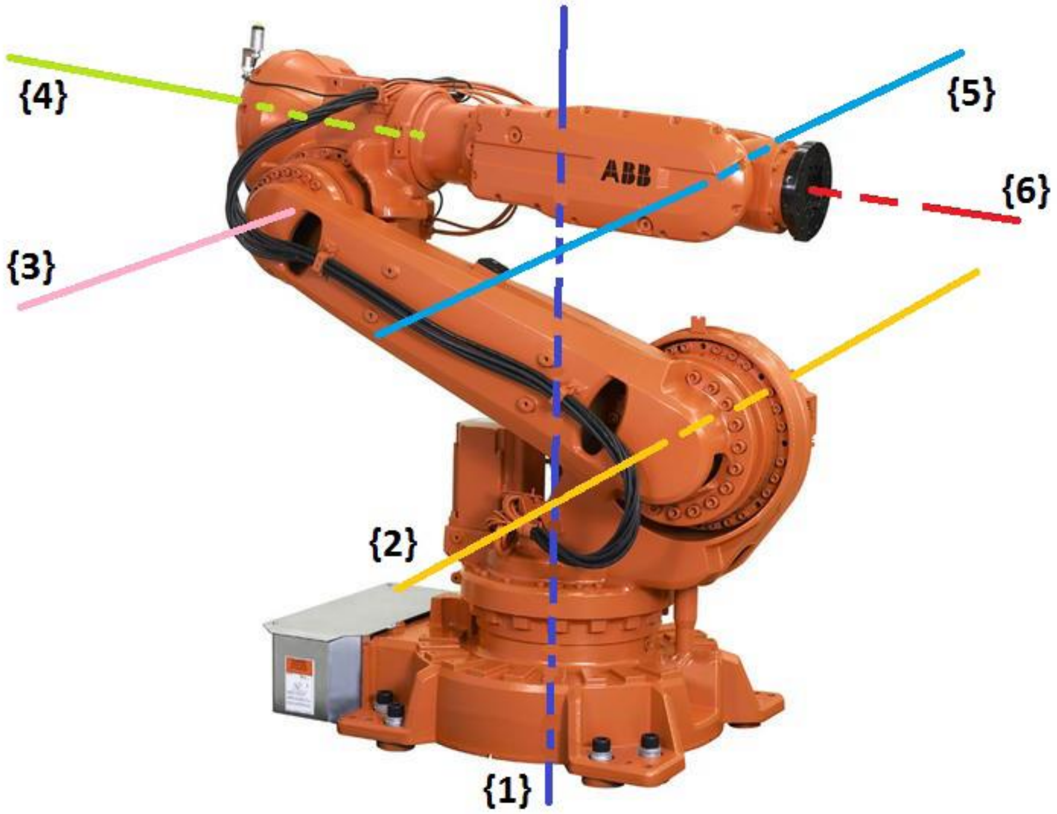


Figure 6.5 : Axis description of IRB 6620 [34].

Table 6.2 : Performance data of IRB 6620 [36].

Axis	Type of Motion	Range of Movement	Maximum Axis Speed	Maximum Permissible Joint Torques
1	**Rotation motion**	**+170° to -170°**	**100°/s**	**4400 Nm**
2	**Arm motion**	**+140° to -65°**	**90°/s**	**15230 Nm**
3	**Arm motion**	**+70° to -180°**	**90°/s**	**2770 Nm**
4	**Wrist motion**	**+300° to -300°**	**150°/s**	**736 Nm**
5	**Bend motion**	**+130° to -130°**	**120°/s**	**736 Nm**
6	**Turn motion**	**+300° to -300°**	**190°/s**	**383 Nm**

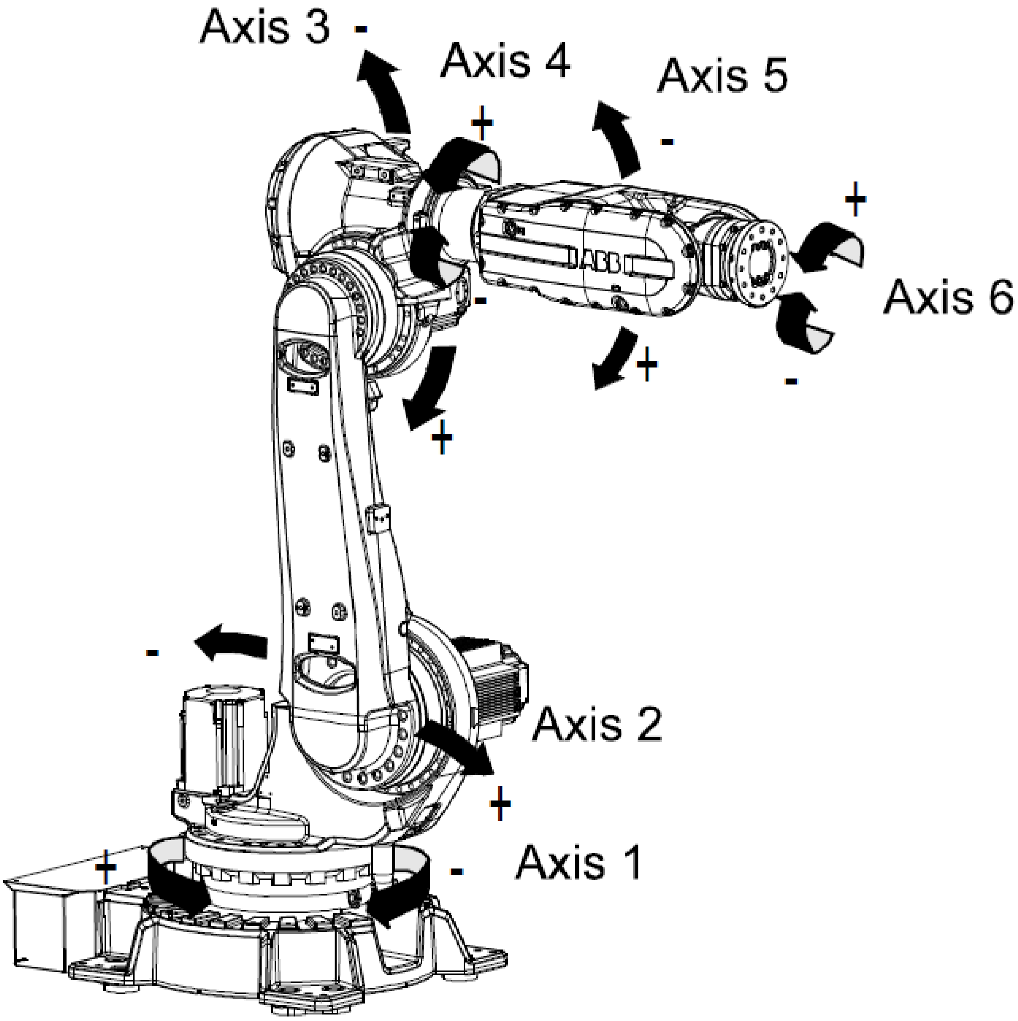


Figure 6.6 : Axis of rotations of IRB 6620 [36].

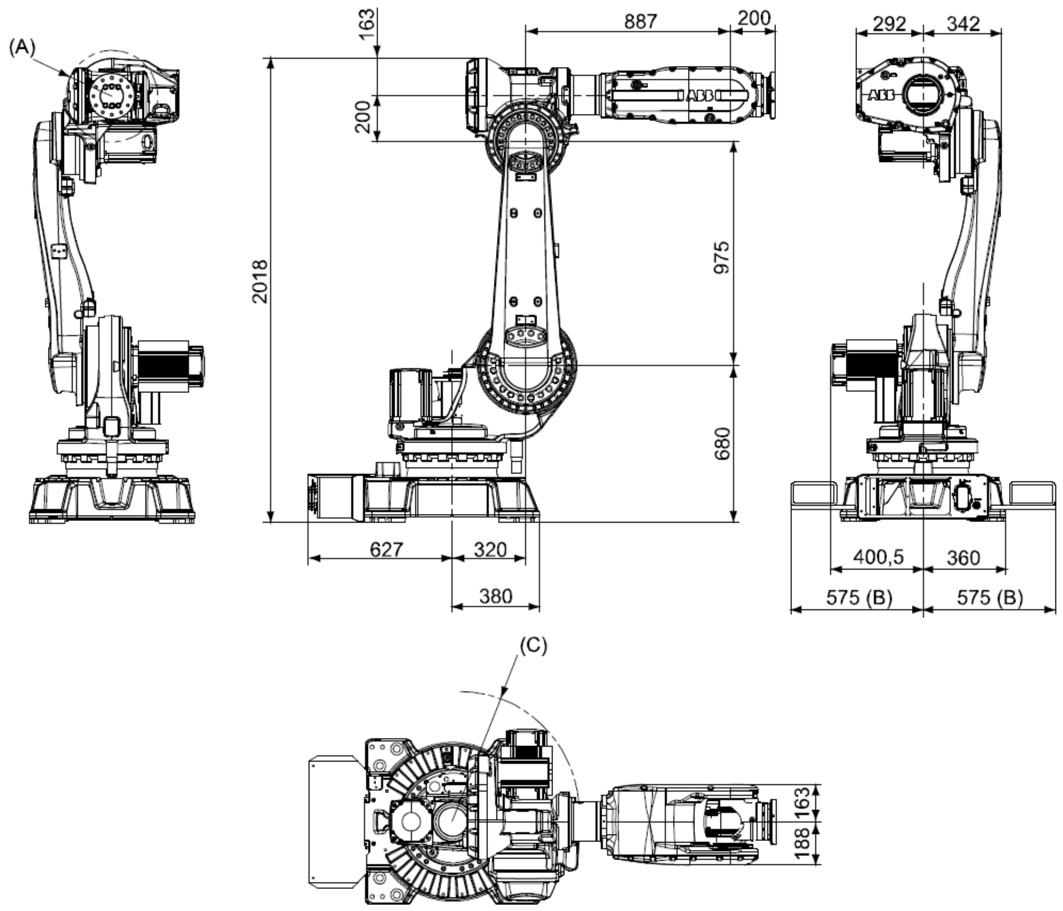


Figure 6.7 : IRB 6620 CAD model with principal distances between axis [36].

In the Figure 6.7; A means R 199 mm for wrist rotation, B means Forklift width 1150 mm and C means R 568 mm for Axis 2 motor.

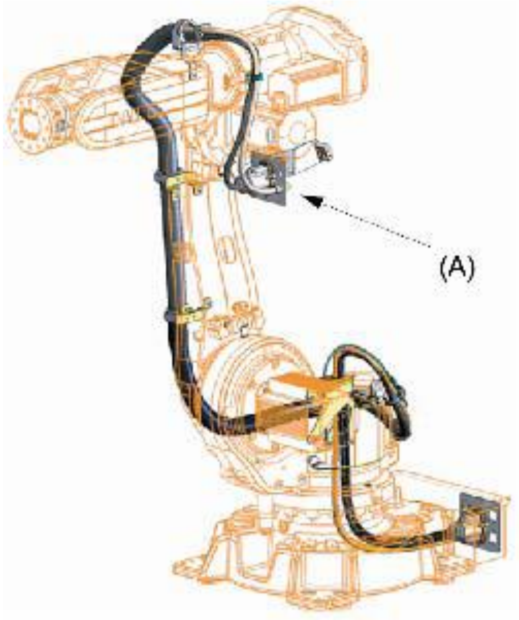


Figure 6.8 : DressPack for material handling (A: Connection point at axis 3) [36].

6.3 How to Get Parameters from SolidWorks

SolidWorks software allows displaying dynamic parameters of object by ‘mass properties’ button. Firstly, click button and related object are chosen from object tree in left side for each link. After that, the selected part must be shown in a different color depicted in the Figure 6.9.

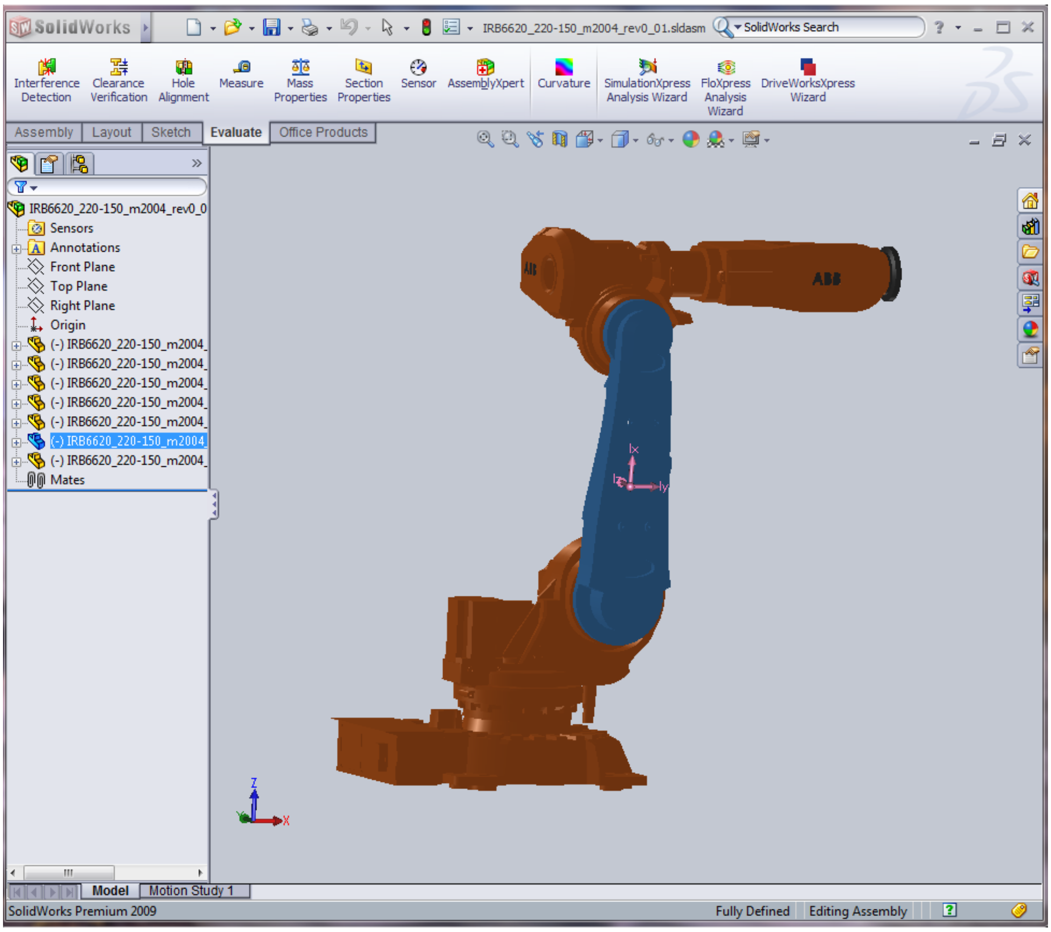
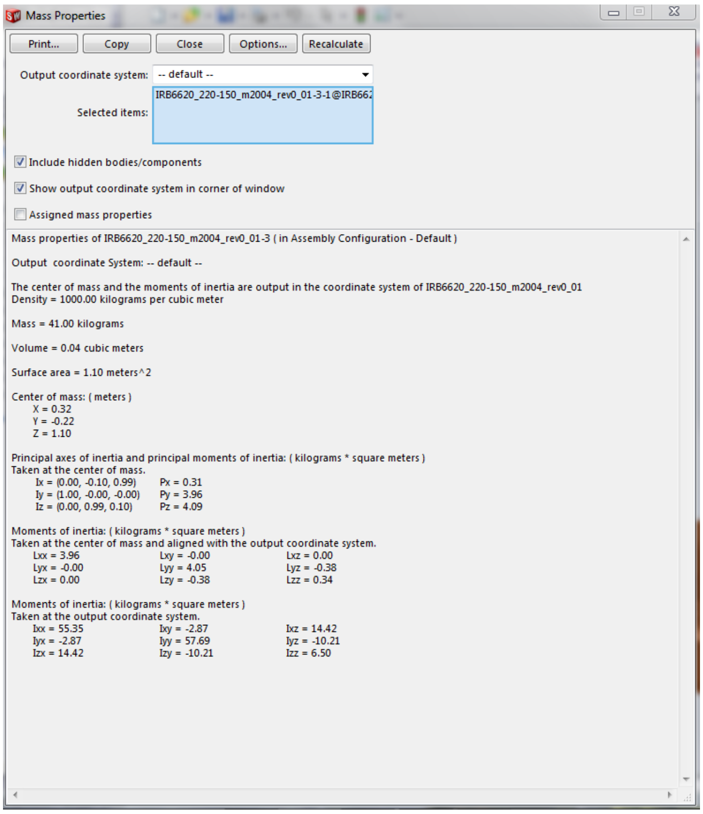


Figure 6.9 : Selection of a link from the SolidWorks.

In the main screen, select “Evaluate” and then “Mass Properties”, then following page will pop up. Then, all dynamic parameters for the selected part will be on the new screen.

In the Figure 6.10, Mass, Center of Mass, Inertia values can be clearly seen. All necessary parameters to model a robotic manipulator can be provided from SolidWorks if 3D CAD model is provided by the vendor. However, for IRB6620 all of the necessary parameters are provided in the official documents, except for Inertia. So, in this thesis only Inertia data is extracted from SolidWorks because of being easier and more accurate.



Output coordinate system: -- default --

Selected items: IRB6620_220-150_m2004_rev0_01-3-1@IRB662

☑ Include hidden bodies/components

☑ Show output coordinate system in corner of window

☐ Assigned mass properties

```
Mass properties of IRB6620_220-150_m2004_rev0_01-3 ( in Assembly Configuration - Default )

Output  coordinate System: -- default --

The center of mass and the moments of inertia are output in the coordinate system of IRB6620_220-150_m2004_rev0_01
Density = 1000.00 kilograms per cubic meter

Mass = 41.00 kilograms

Volume = 0.04 cubic meters

Surface area = 1.10 meters^2

Center of mass: ( meters )
	X = 0.32
	Y = -0.22
	Z = 1.10

Principal axes of inertia and principal moments of inertia: ( kilograms * square meters )
Taken at the center of mass.
	 Ix = (0.00, -0.10, 0.99)   	Px = 0.31
	 Iy = (1.00, -0.00, -0.00)  	Py = 3.96
	 Iz = (0.00, 0.99, 0.10)    	Pz = 4.09

Moments of inertia: ( kilograms * square meters )
Taken at the center of mass and aligned with the output coordinate system.
	Lxx = 3.96     	Lxy = -0.00    	Lxz = 0.00
	Lyx = -0.00    	Lyy = 4.05     	Lyz = -0.38
	Lzx = 0.00     	Lzy = -0.38    	Lzz = 0.34

Moments of inertia: ( kilograms * square meters )
Taken at the output coordinate system.
	Ixx = 55.35    	Ixy = -2.87    	Ixz = 14.42
	Iyx = -2.87    	Iyy = 57.69    	Iyz = -10.21
	Izx = 14.42    	Izy = -10.21   	Izz = 6.50
```

Figure 6.10 : Mass properties screen from SolidWorks for the selected part/parts.

6.4 Computed-Torque Control Simulation to Track a Joint Space Trajectory

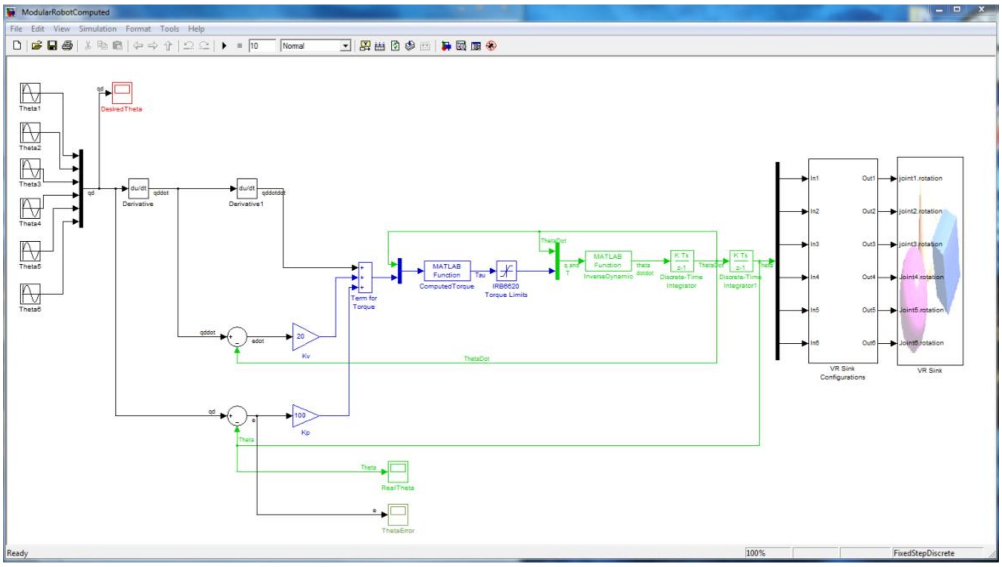


Figure 6.11 : Mass properties screen from SolidWorks for the selected part/parts.

In this thesis, Matlab is used to model and to simulate the robot arm. In chapter 5, Computed-Torque Control is explained. In this part, the simulation which is explained in chapter 5 is shown.

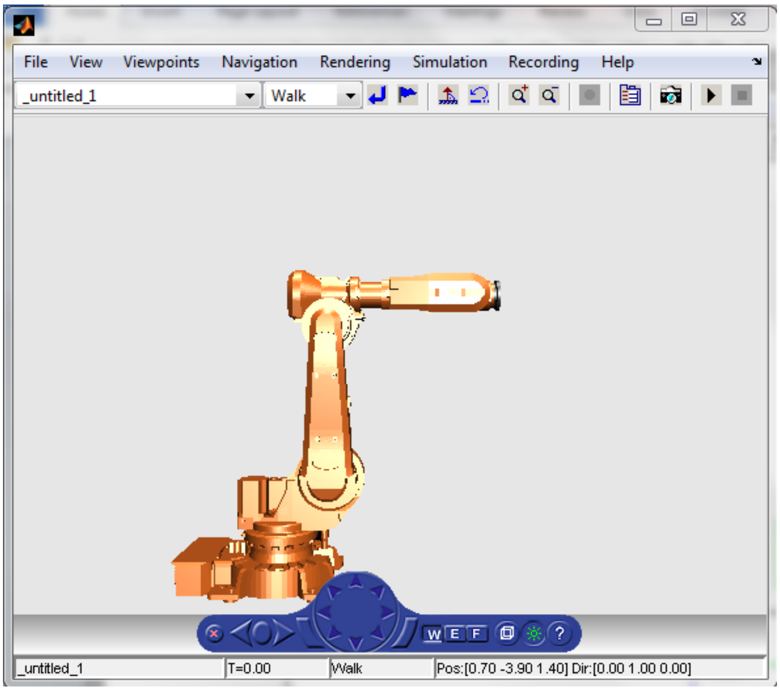


Figure 6.12 : Interior of Virtual Reality block used in the simulation.

In this simulation, K_v is selected as 20 and K_p is selected as 100. Sinus signal is selected as input for all joints. Joint torque limits are also added to the simulation.

6.5 Adaptive Control Simulation to Track a Joint Space Trajectory

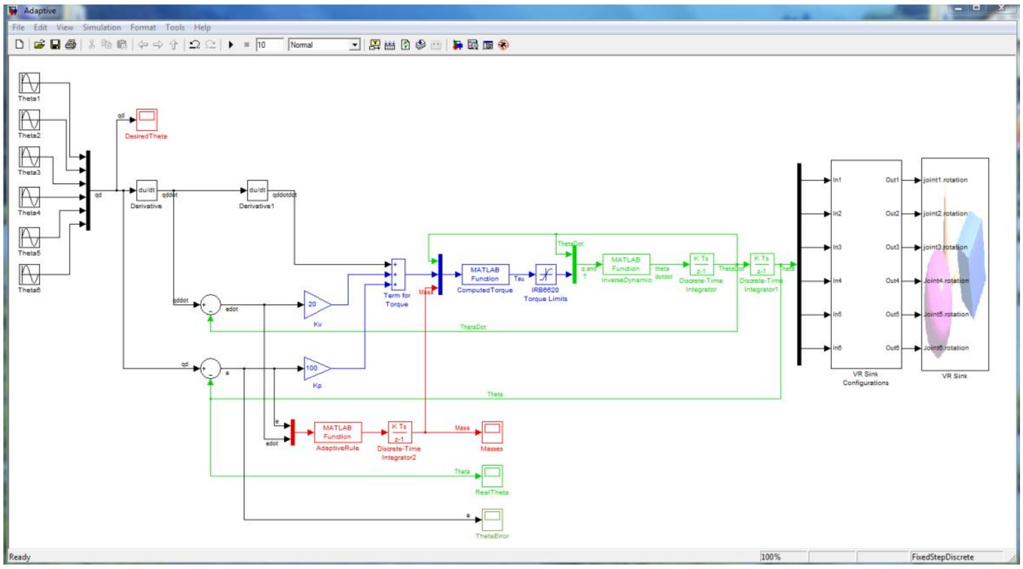
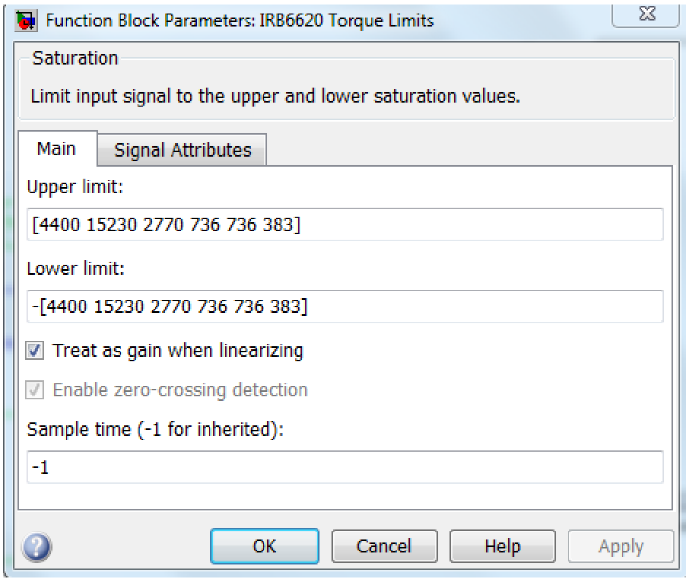


Figure 6.13 : Adaptive computed-torque control simulation.

In this simulation, k_v is selected as 20, k_p is selected as 100 and $\Gamma = 500 * \mathrm{I}$. Sinus signal is selected as input for all joints, because its fluctuating characteristic challenges the algorithm. Thus, we could easily see the limits and capabilities of the algorithm. Joint torque limits are also added to the simulation.



Function Block Parameters: IRB6620 Torque Limits

Saturation

Limit input signal to the upper and lower saturation values.

Upper limit:

[4400 15230 2770 736 736 383]

Lower limit:

-[4400 15230 2770 736 736 383]

☑ Treat as gain when linearizing

☑ Enable zero-crossing detection

Sample time (-1 for inherited):

-1

Figure 6.14 : Interior of torque limits block used in the simulation.

6.6 Adaptive Control and Computed-Torque Control Results' Comparison for a Preplanned Joint Space Trajectory Tracking

For all cases under this heading, "Adaptive Control and Computed-Torque Results' Comparison Tracking a Preplanned Joint Space Trajectory", desired joint space trajectory is depicted in Fig. 6.15.

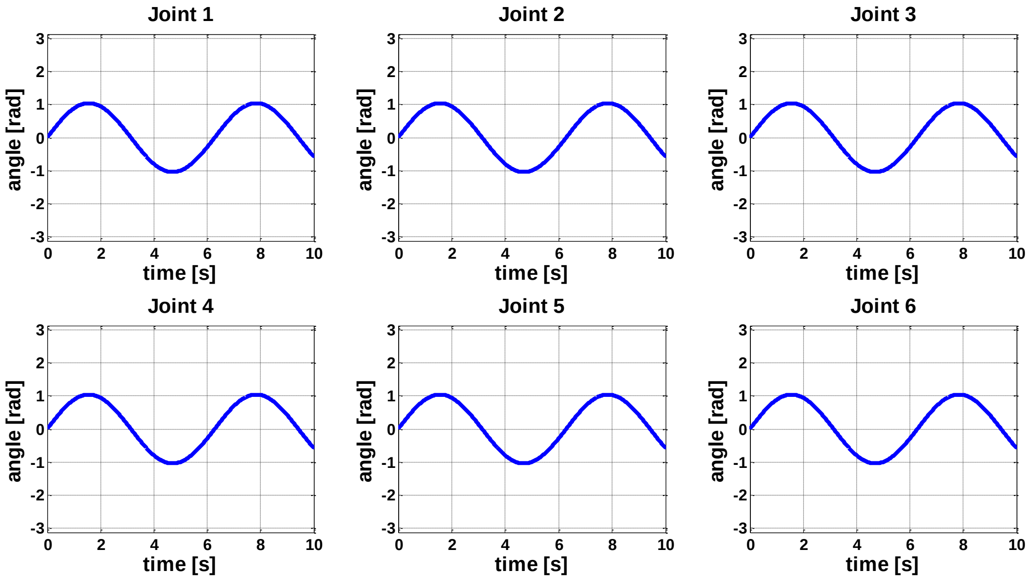


Figure 6.15 : Joint space desired trajectory.

6.6.1 In the case of all masses are fully known

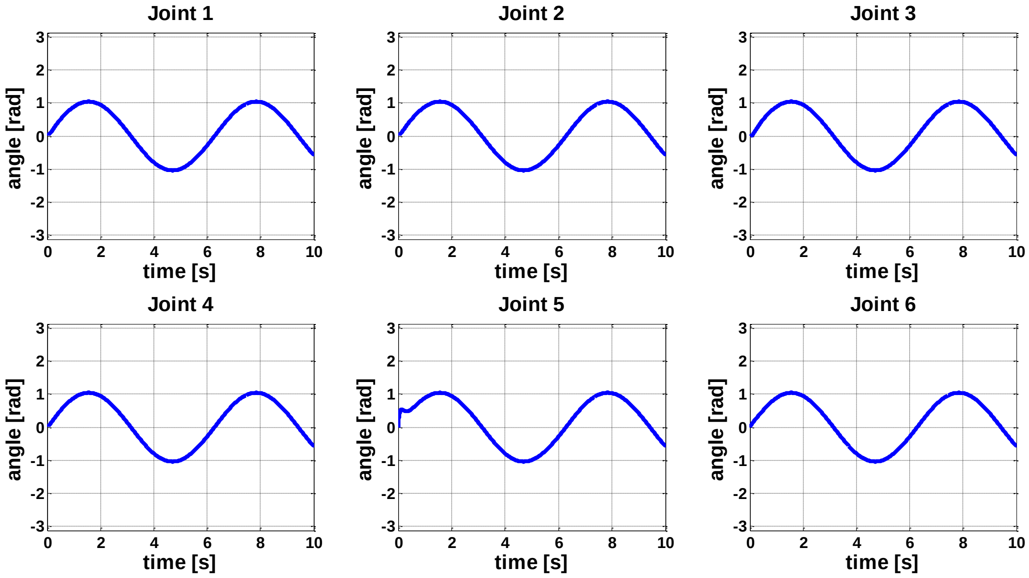


Figure 6.16 : Joint space trajectory tracking with computed-torque control.

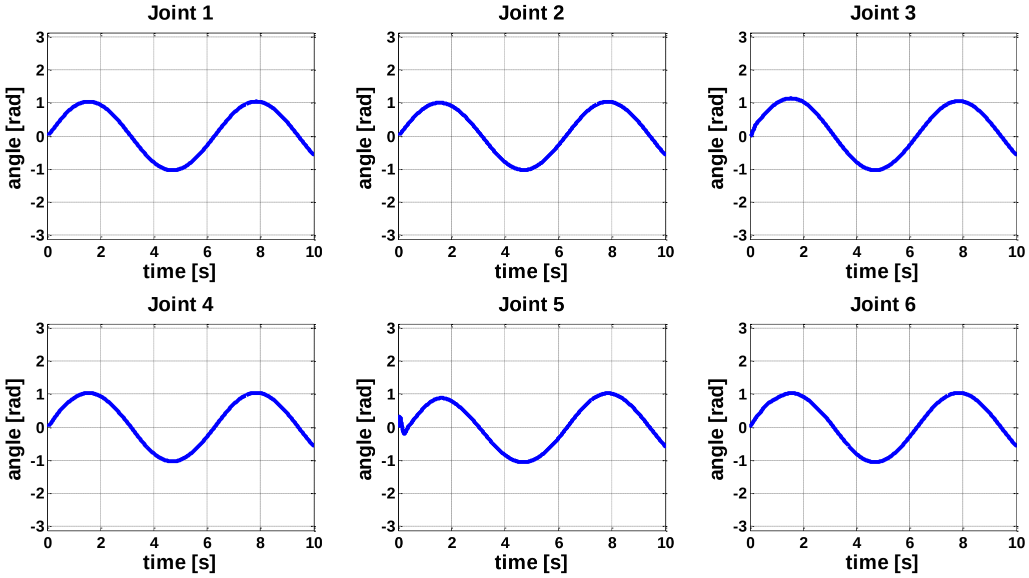


Figure 6.17 : Joint space trajectory tracking with adaptive control.

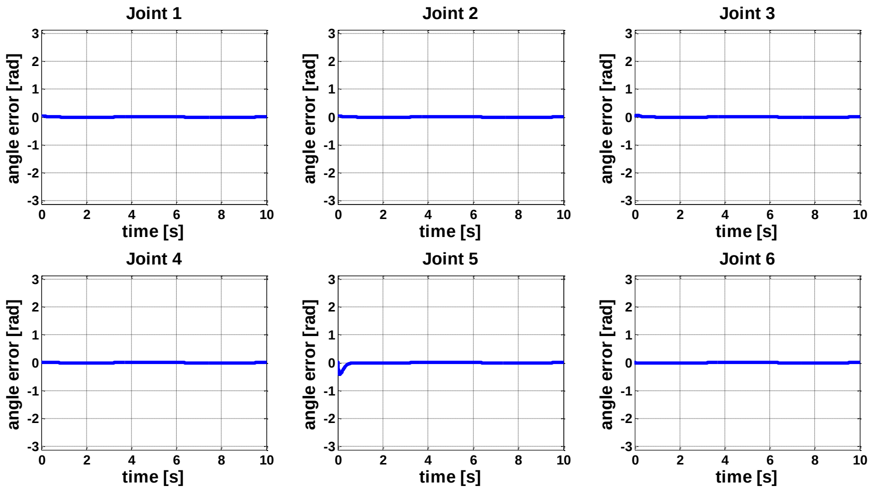


Figure 6.18 : Joint space trajectory tracking position error with CTC.

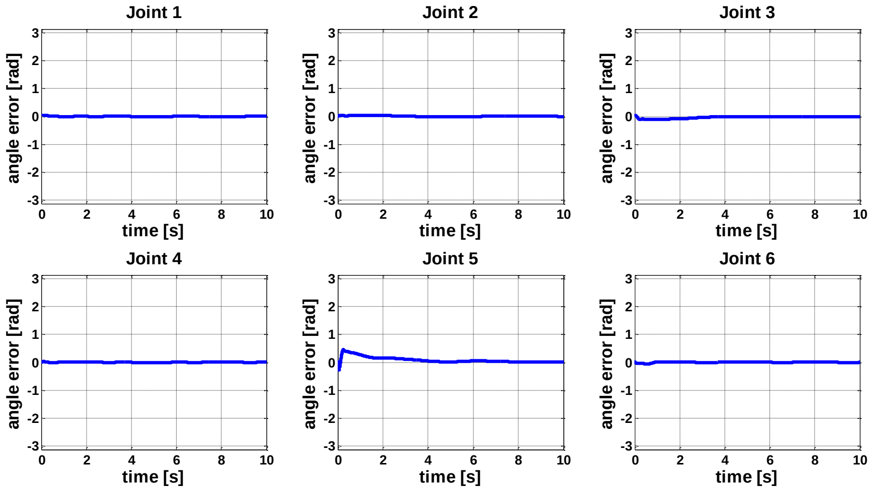


Figure 6.19 : Joint space trajectory tracking position error with adaptive control.

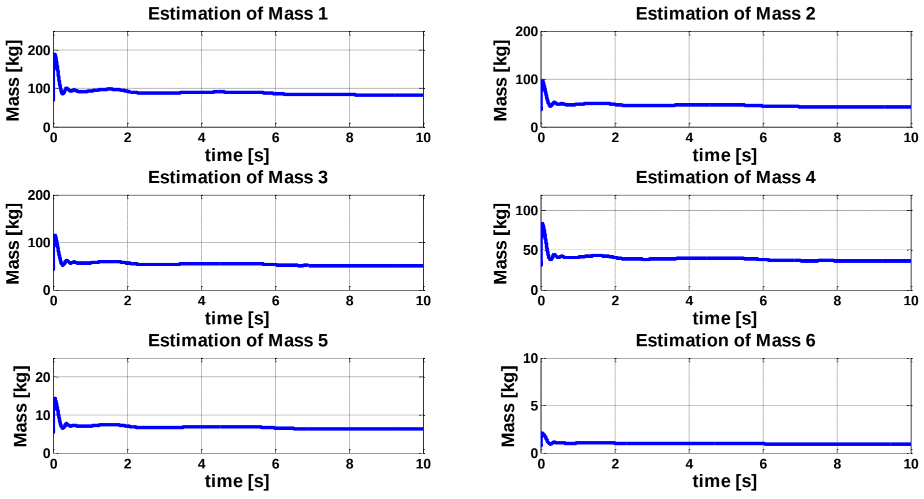


Figure 6.20 : Adaptive control rule link masses’ estimation.

6.6.2 In the case of all masses are presumed as 1.5 times heavier

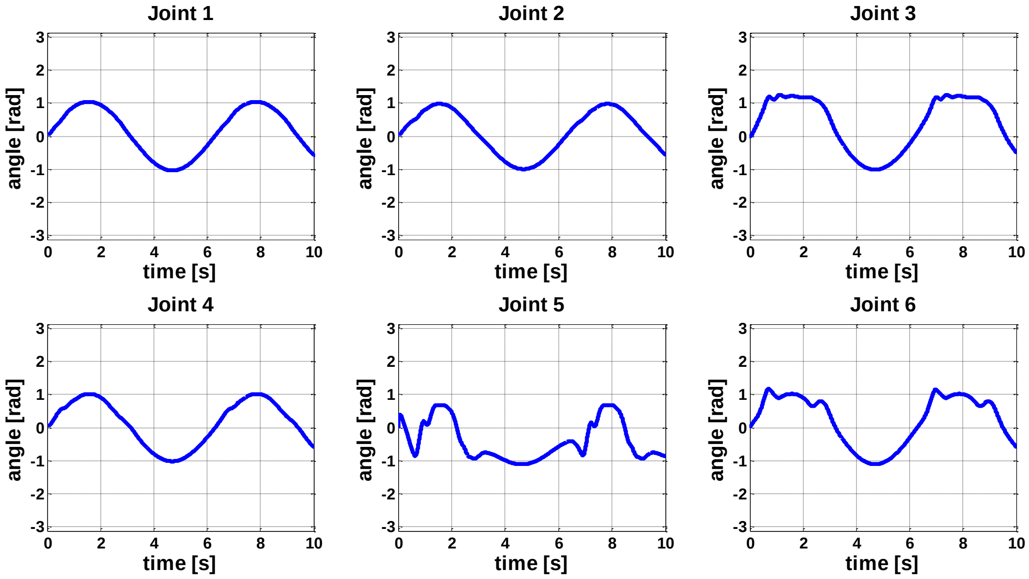


Figure 6.21 : Joint space trajectory tracking with computed-torque control.

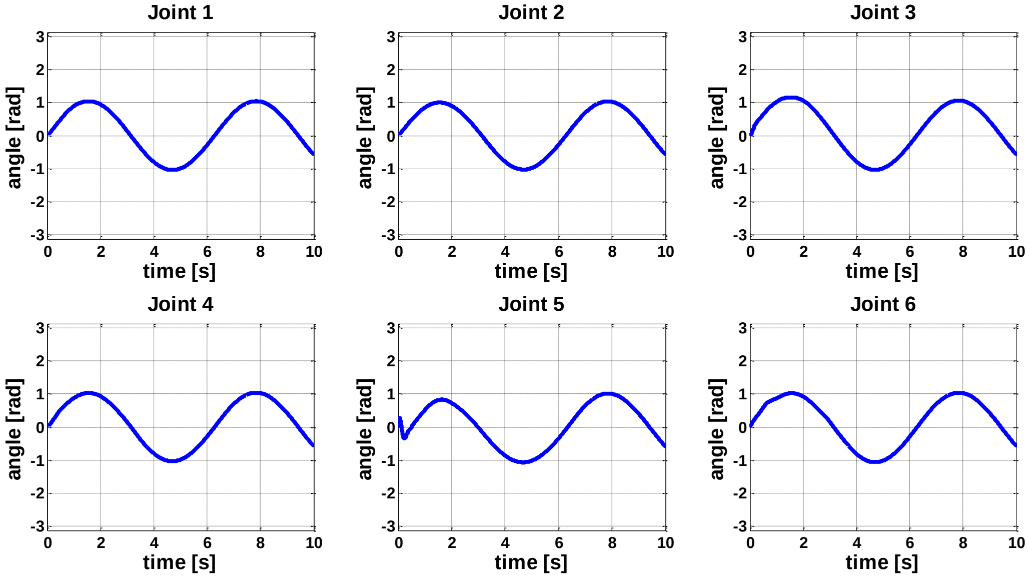


Figure 6.22 : Joint space trajectory tracking with adaptive control.

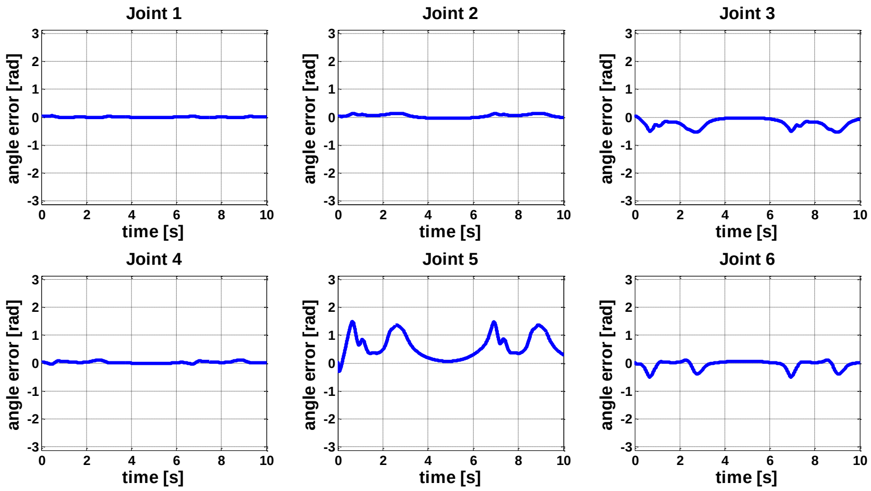


Figure 6.23 : Joint space trajectory tracking position error with CTC.

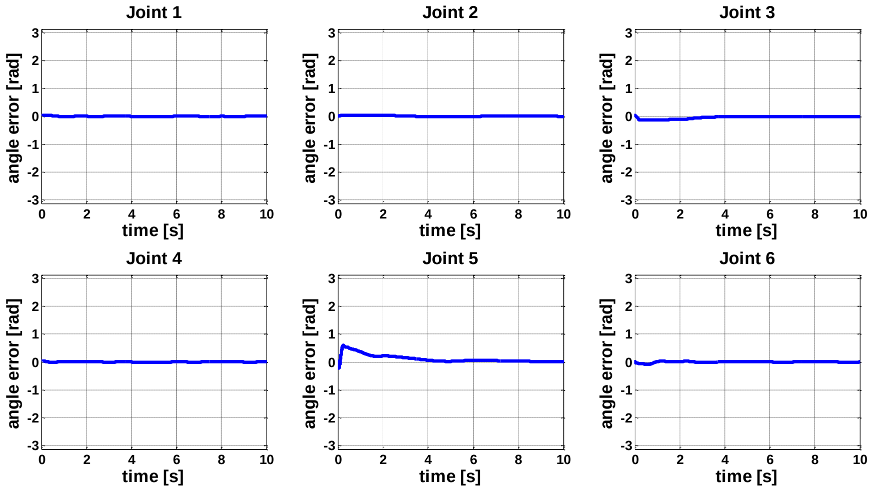


Figure 6.24 : Joint space trajectory tracking position error with adaptive control.

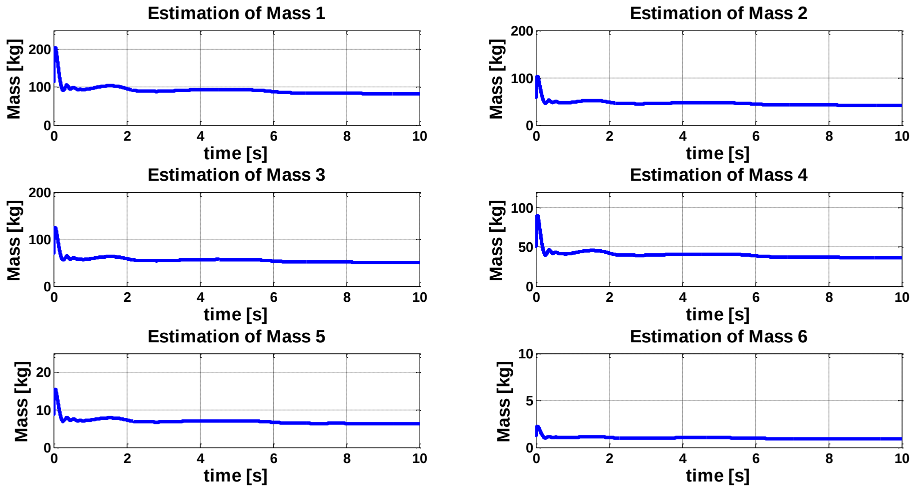


Figure 6.25 : Adaptive control rule link masses’ estimation.

6.6.3 In the case of all masses are presumed as 2 times heavier

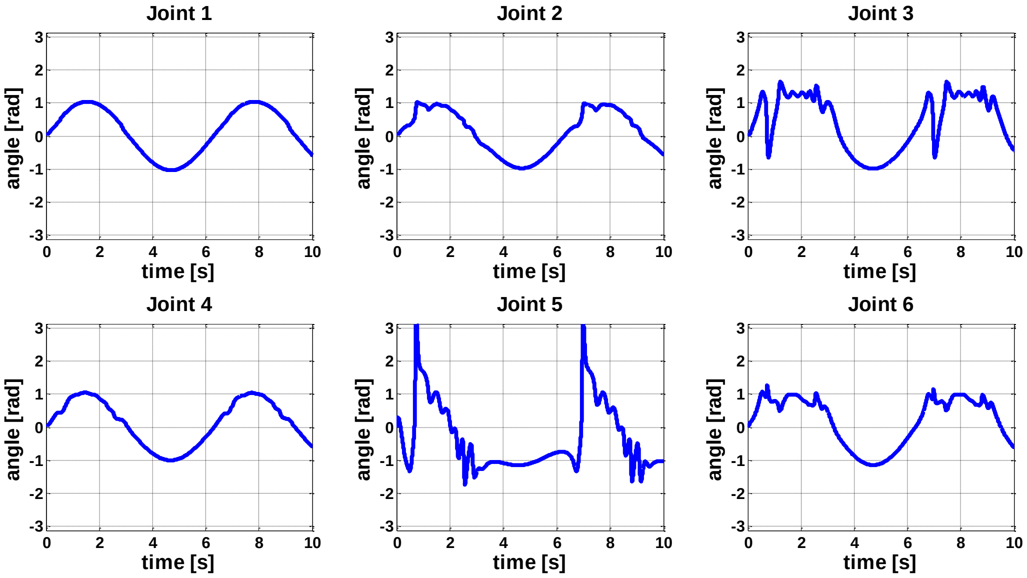


Figure 6.26 : Joint space trajectory tracking with computed-torque control.

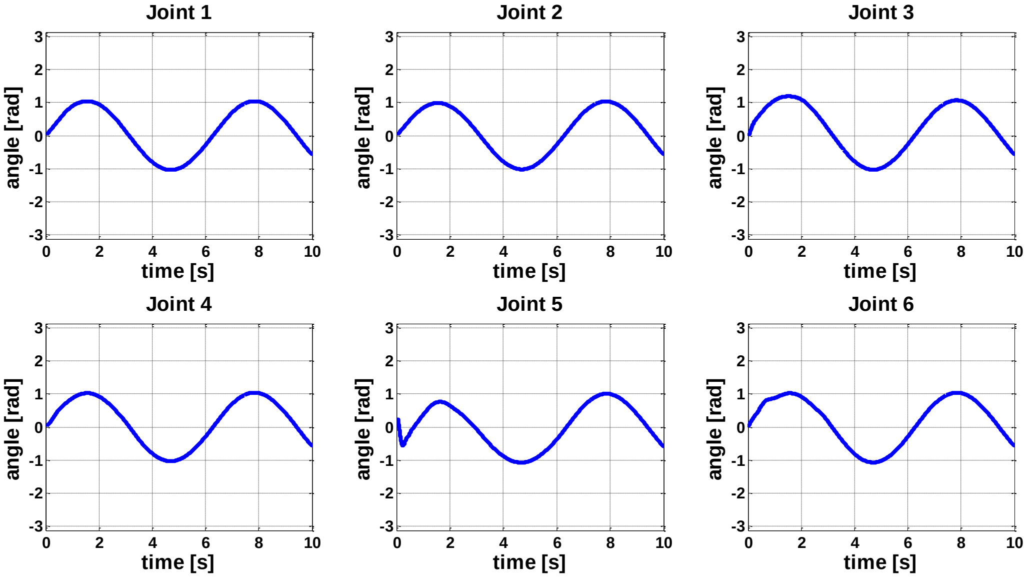


Figure 6.27 : Joint space trajectory tracking with adaptive control.

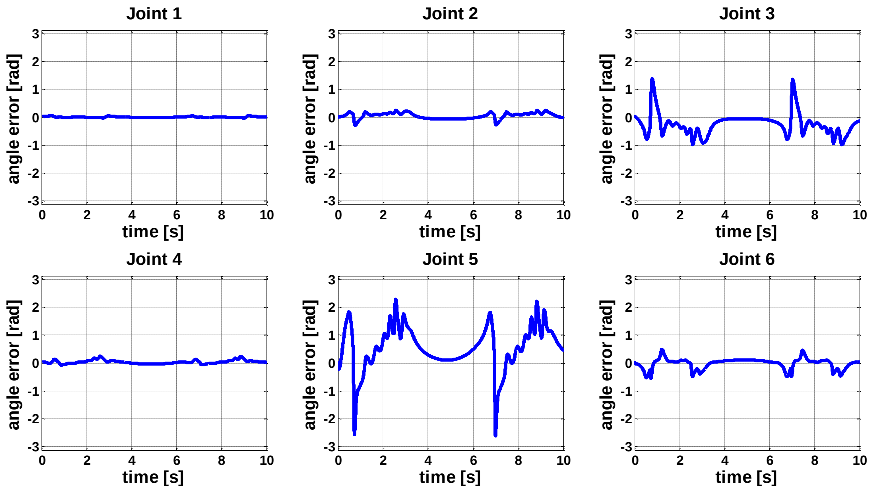


Figure 6.28 : Joint space trajectory tracking position error with CTC.

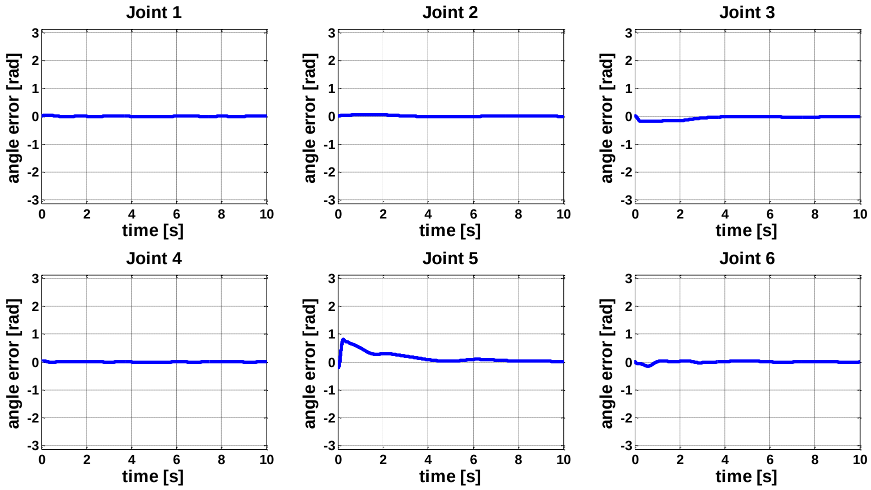


Figure 6.29 : Joint space trajectory tracking position error with adaptive control.

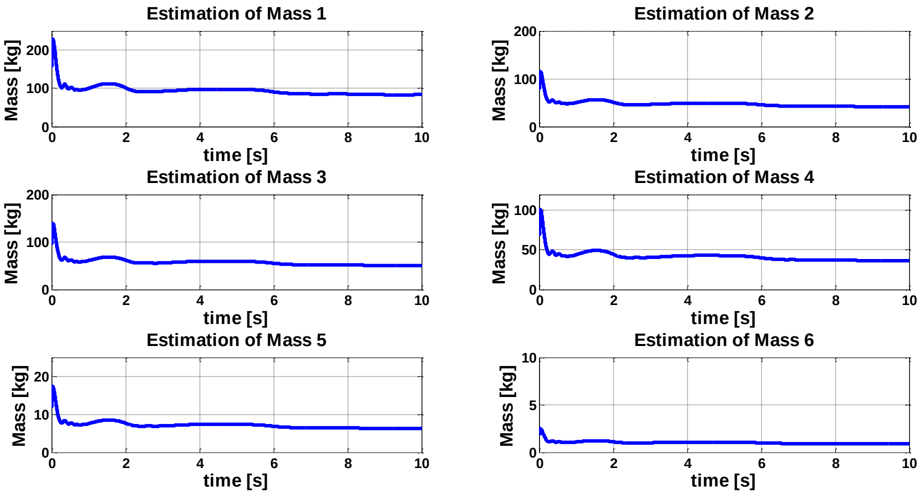


Figure 6.30 : Adaptive control rule link masses' estimation.

6.6.4 In the case of all masses are presumed as half of the real values

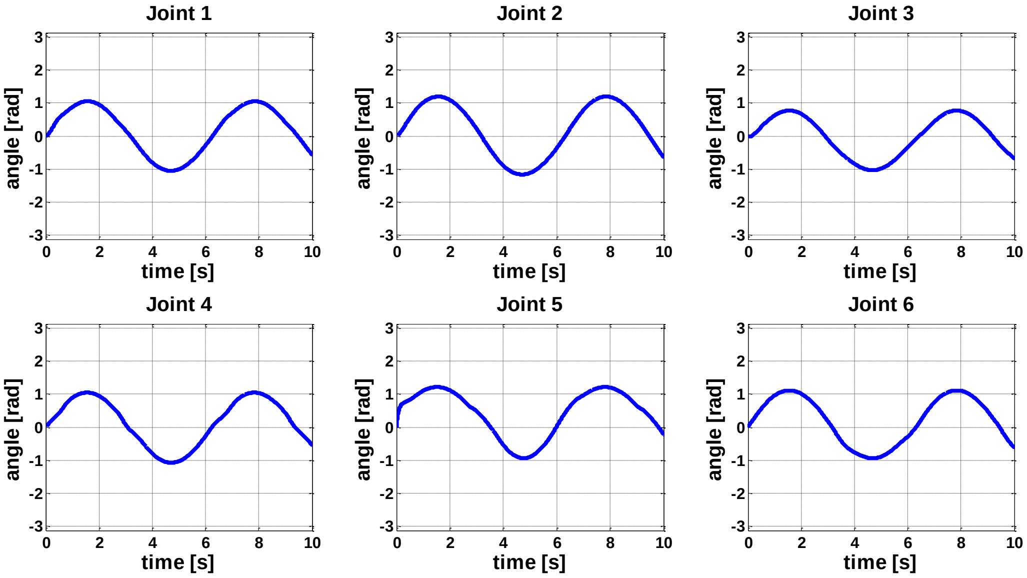


Figure 6.31 : Joint space trajectory tracking with computed-torque control.

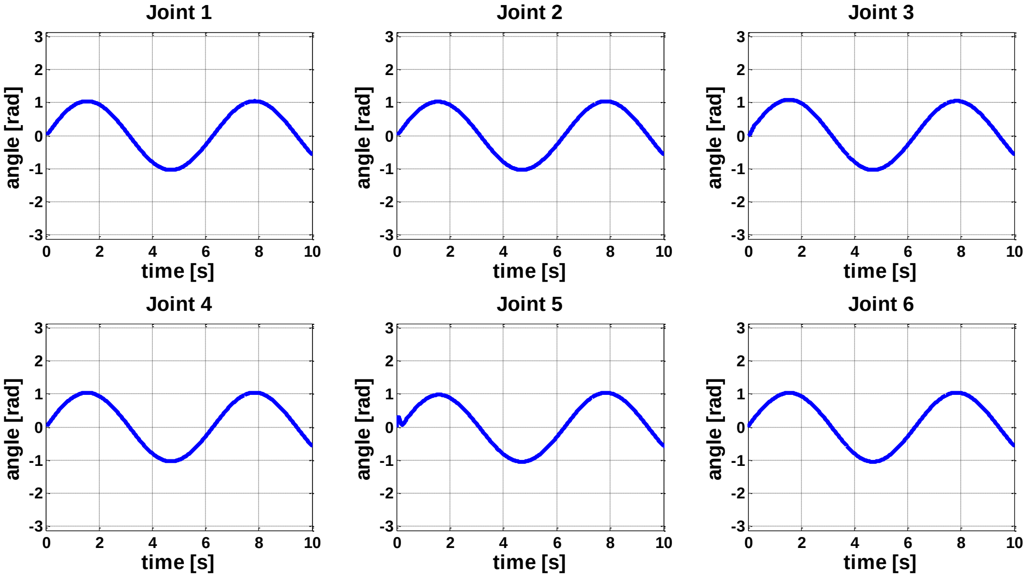


Figure 6.32 : Joint space trajectory tracking with adaptive control.

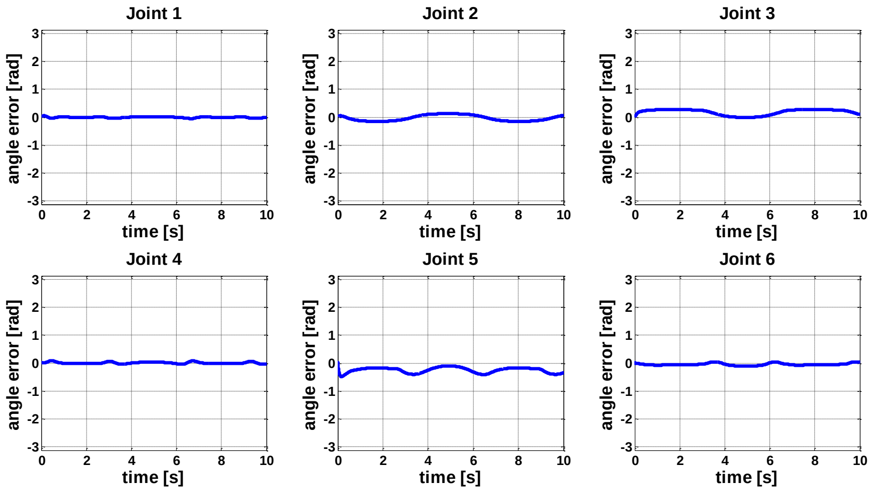


Figure 6.33 : Joint space trajectory tracking position error with CTC.

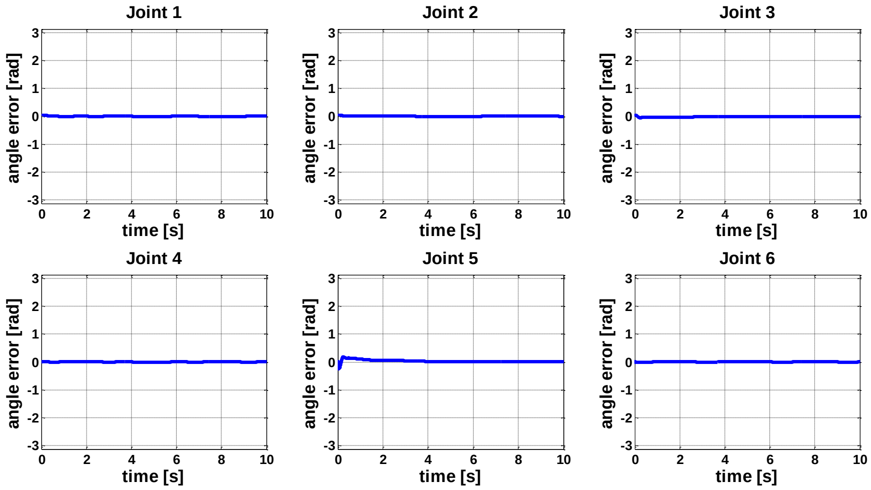


Figure 6.34 : Joint space trajectory tracking position error with adaptive control.

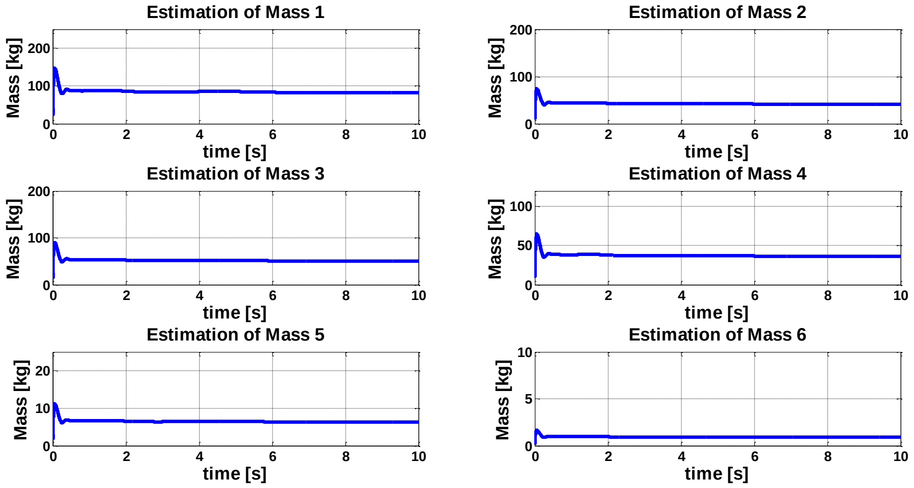


Figure 6.35 : Adaptive control rule link masses’ estimation.

6.7 Adaptive Control and Computed-Torque Control Results' Comparison for a Preplanned Operational Space Trajectory Tracking

For all cases under this heading, "Adaptive Control and Computed-Torque Results' Comparison Tracking a Preplanned Operational Space Trajectory", desired operational space is an ellipsoid trajectory. Desired operational trajectory is shown in green lines and the tracked trajectory is shown in red lines in the figures.

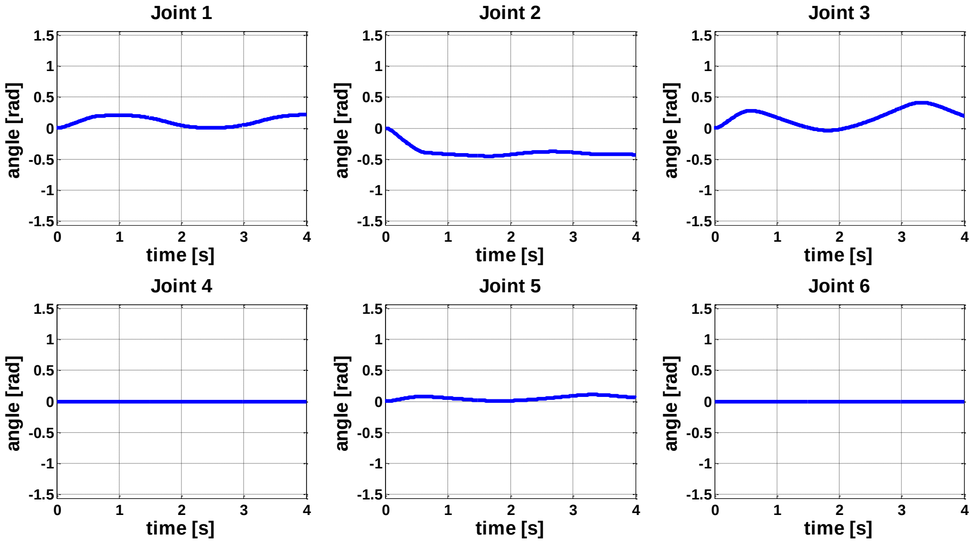


Figure 6.36 : Corresponding joint space trajectory to produce desired ellipsoid trajectory in Operational space.

6.7.1 In the case of all masses are fully known

In simulations, green points show the desired path and the red points show the Tip Point's passed positions.

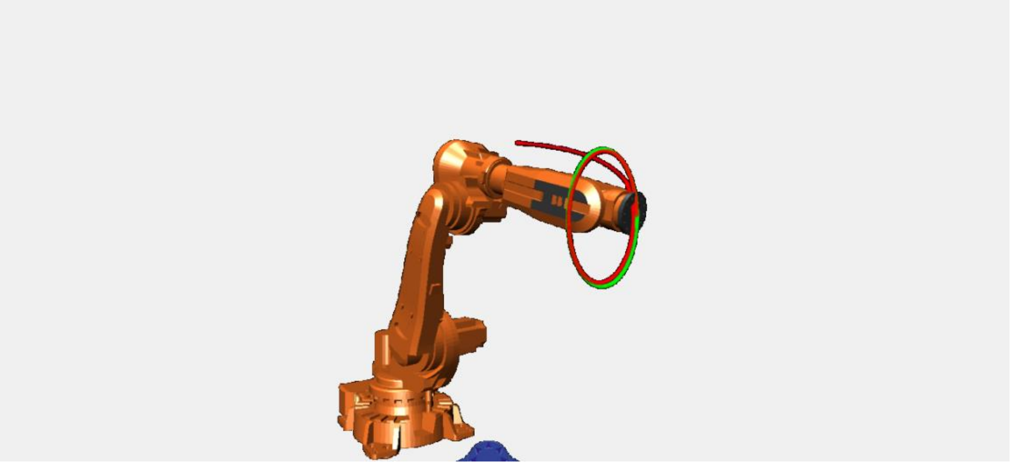


Figure 6.37 : Operational space trajectory tracking with CTC.

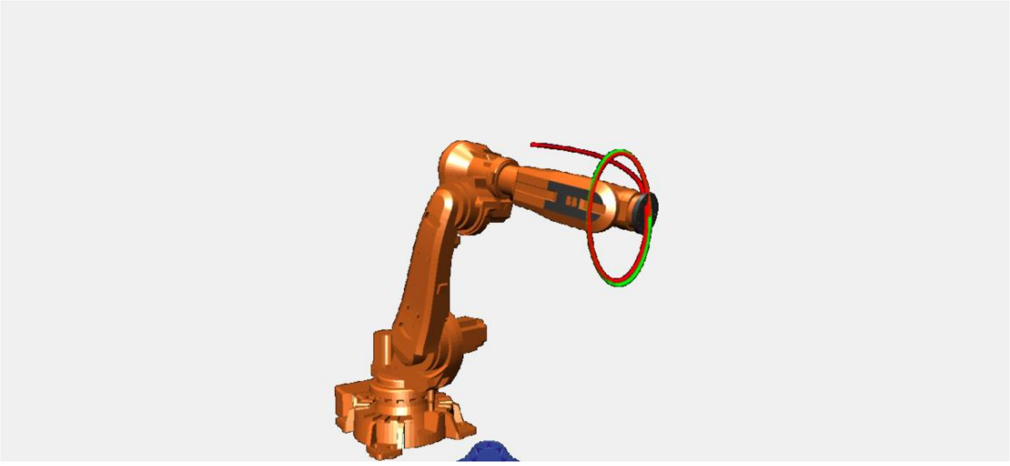


Figure 6.38 : Operational space trajectory tracking with adaptive control.

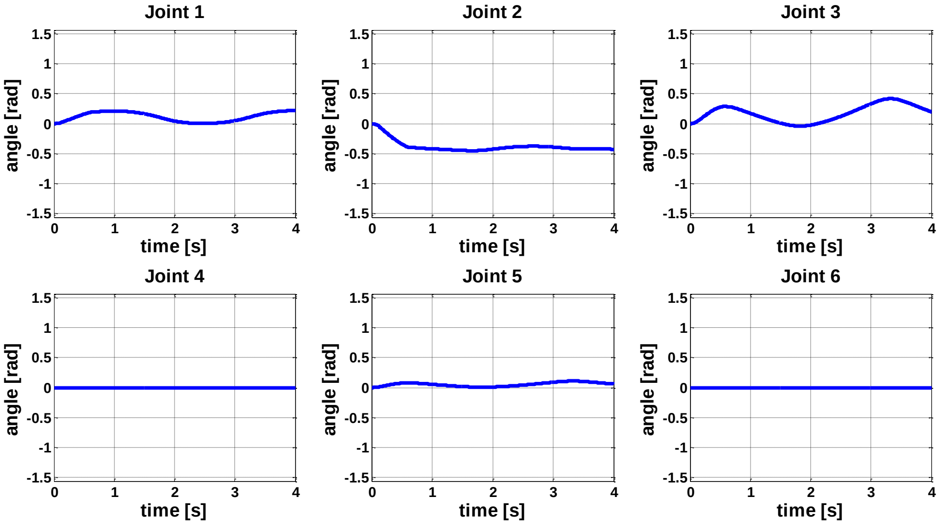


Figure 6.39 : Joint space trajectory tracking with CTC.

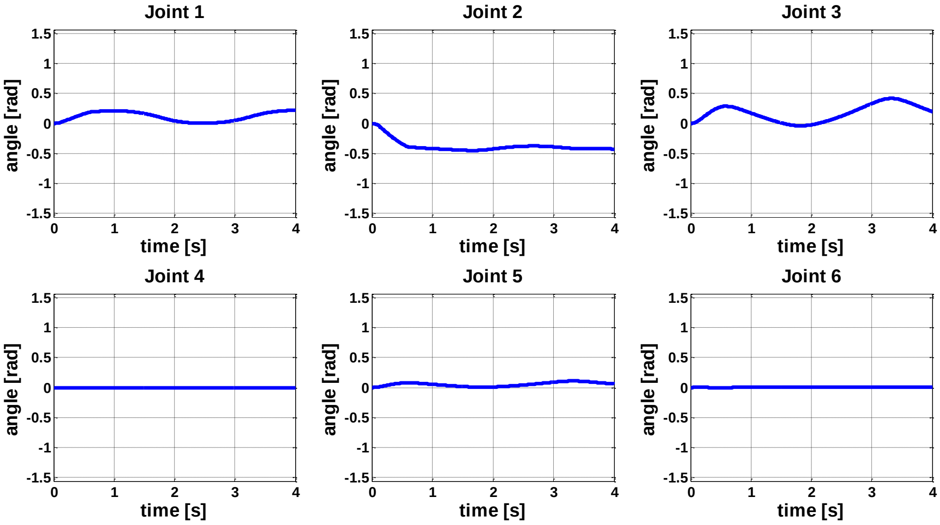


Figure 6.40 : Joint space trajectory tracking with adaptive control.

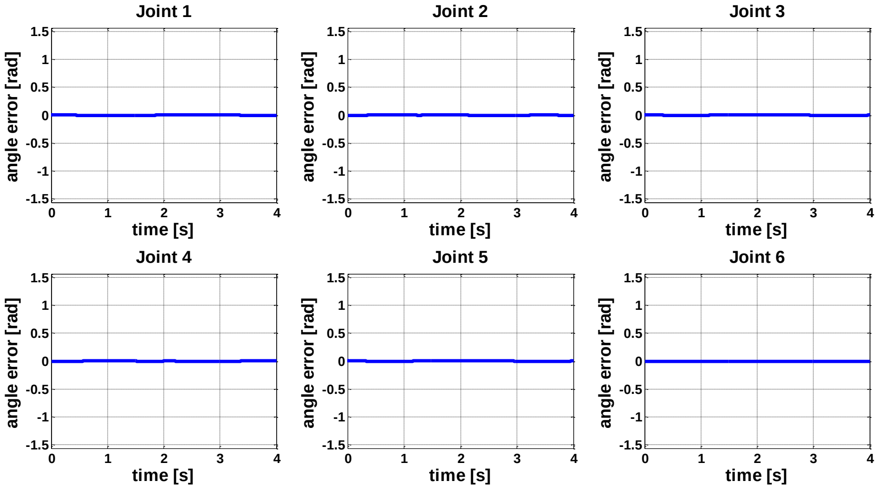


Figure 6.41 : Trajectory tracking position error with CTC.

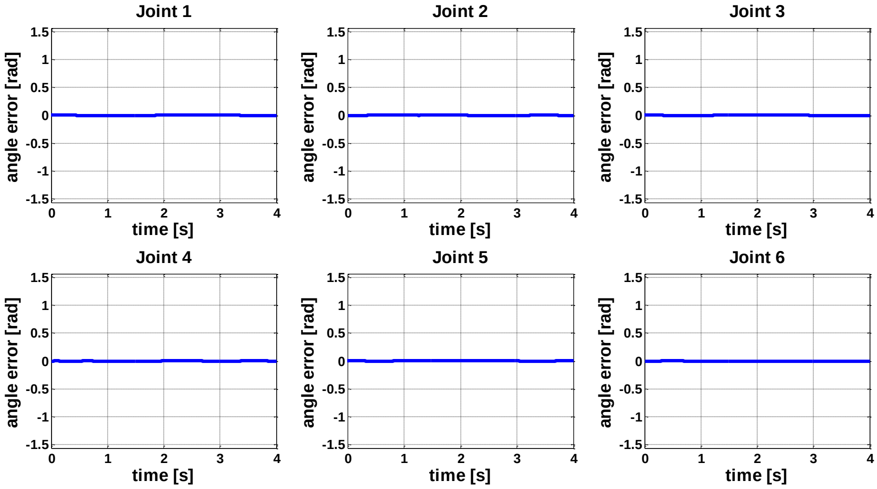


Figure 6.42 : Trajectory tracking position error with adaptive control.

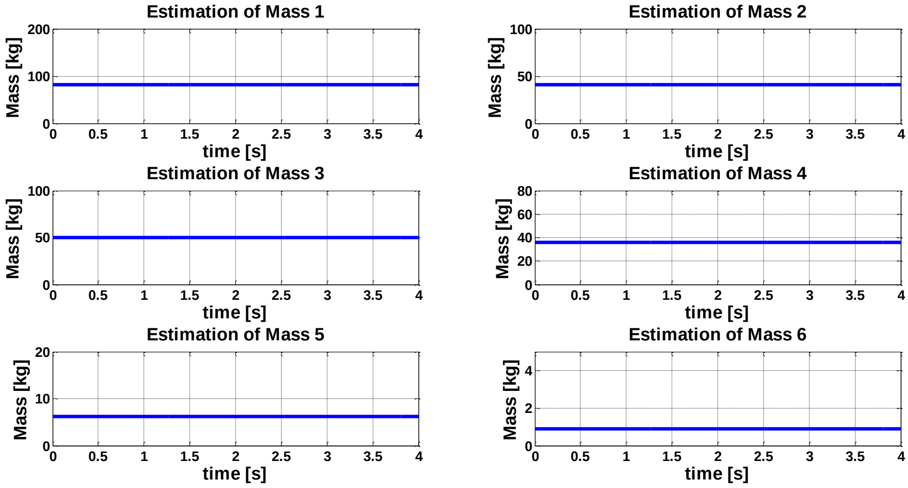


Figure 6.43 : Adaptive control rule link masses’ estimation.

6.7.2 In the case of all masses are presumed as 1.5 times heavier

Green points show the desired path and the red points show the Tip Point's passed positions.

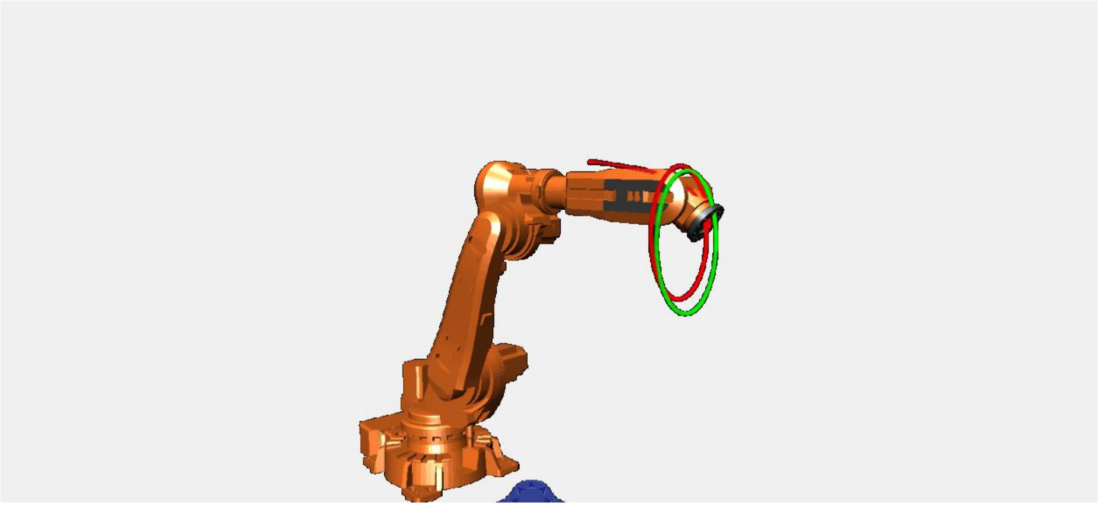


Figure 6.44 : Operational space trajectory tracking with CTC.

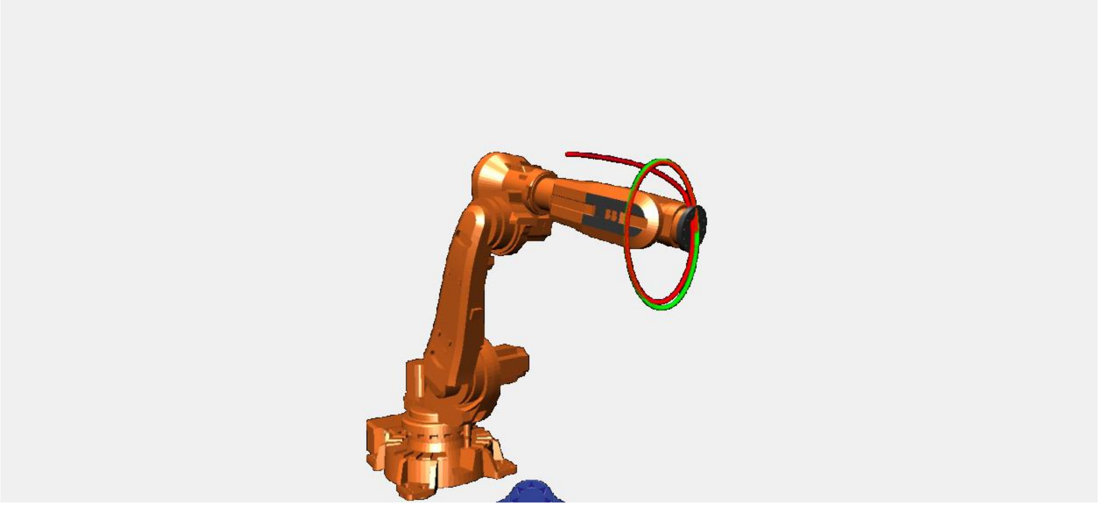


Figure 6.45 : Operational space trajectory tracking with adaptive control.

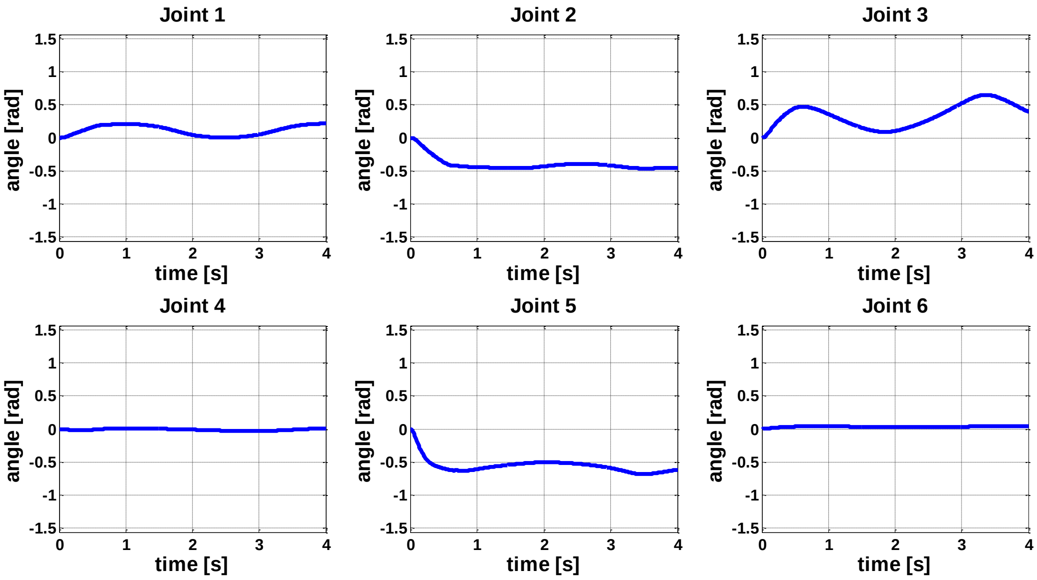


Figure 6.46 : Joint space trajectory tracking with CTC.

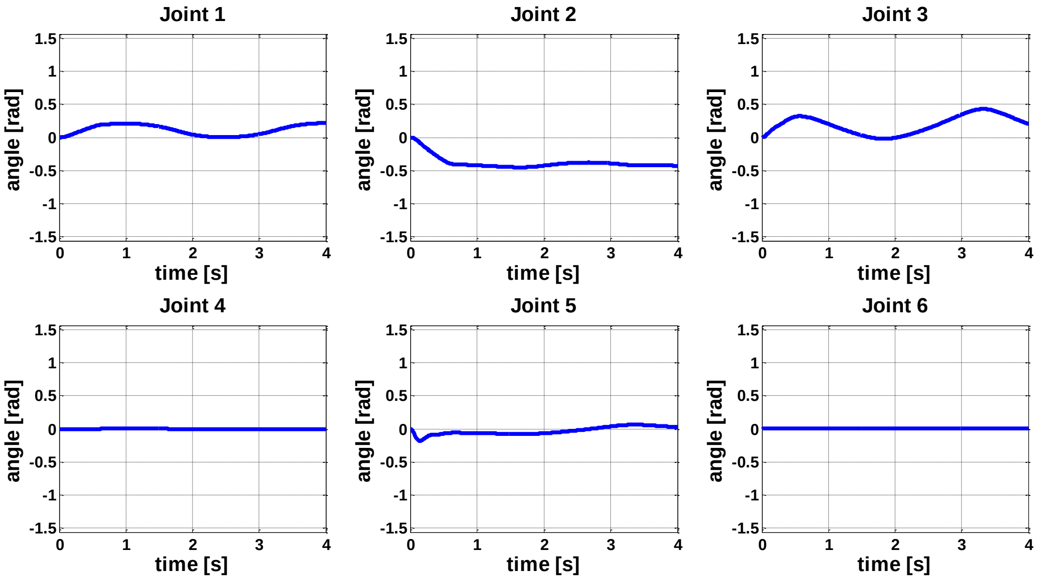


Figure 6.47 : Joint space trajectory tracking with adaptive control.

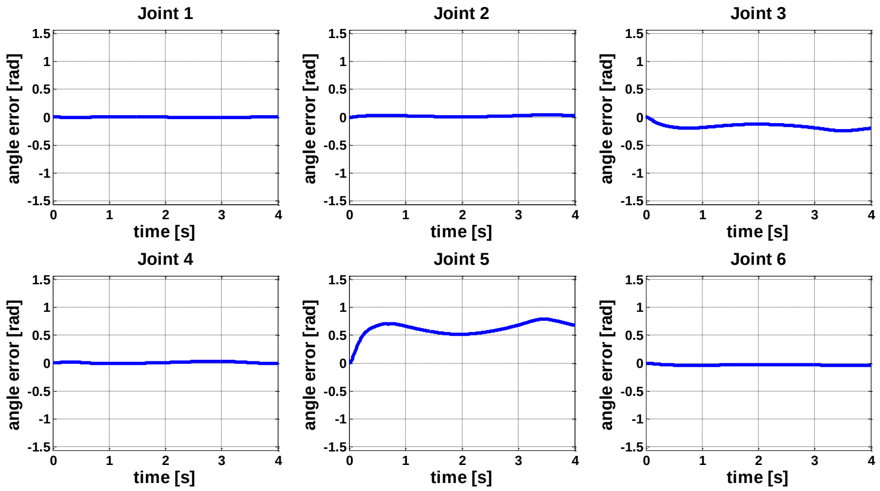


Figure 6.48 : Trajectory tracking position error with CTC.

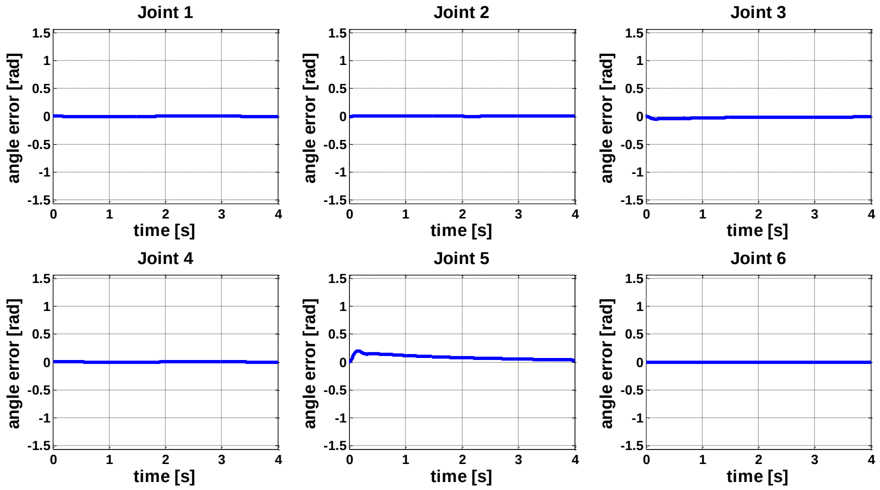


Figure 6.49 : Trajectory tracking position error with adaptive control.

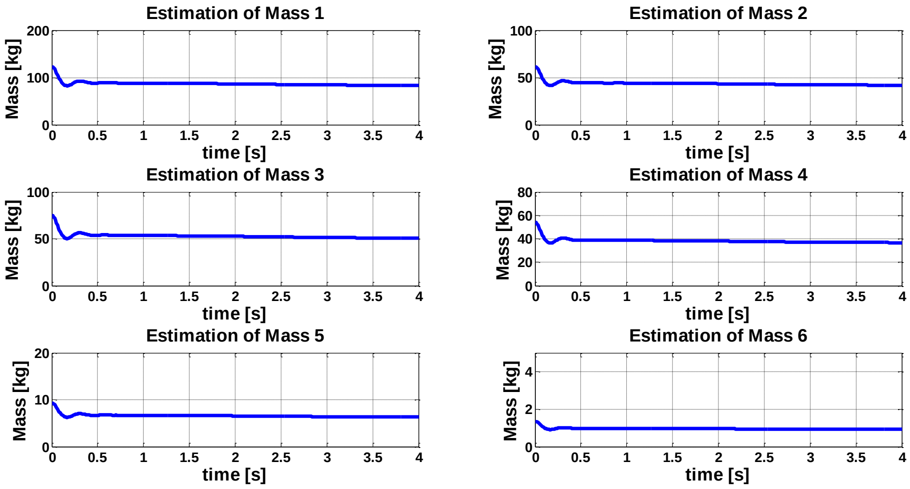


Figure 6.50 : Adaptive control rule link masses’ estimation.

6.7.3 In the case of all masses are presumed as 2 times heavier

Green points show the desired path and the red points show the Tip Point’s real positions.

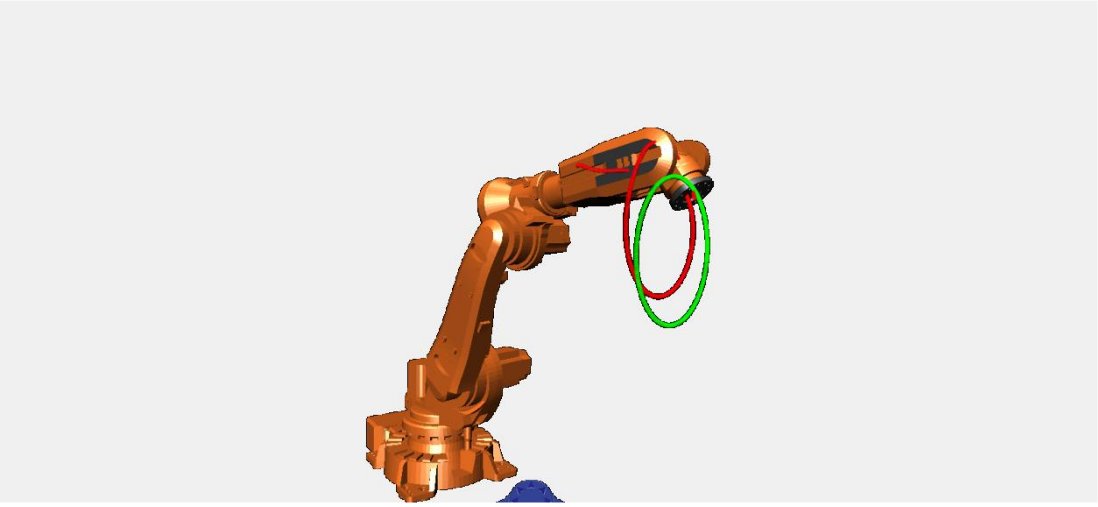


Figure 6.51 : Operational space trajectory tracking with CTC.

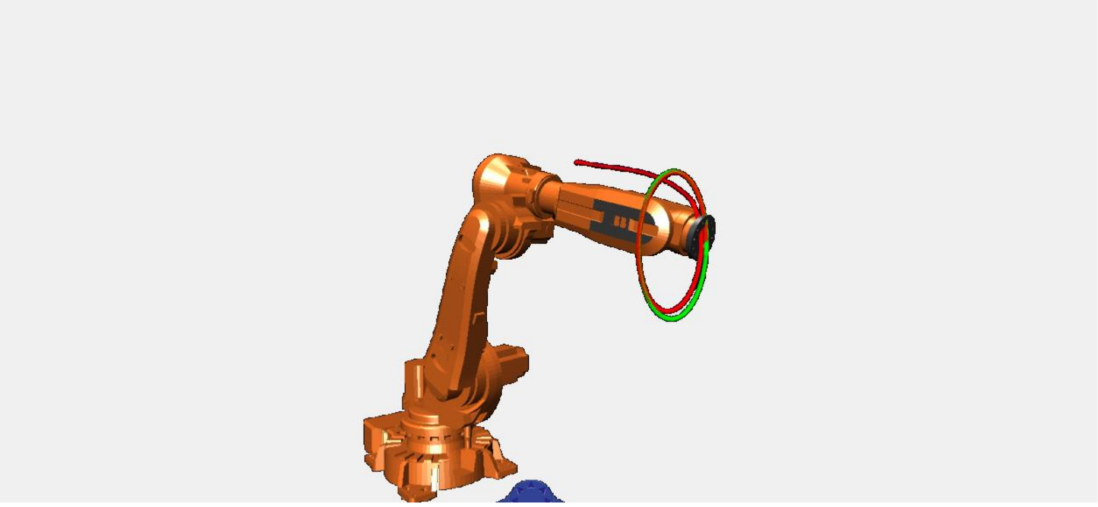


Figure 6.52 : Operational space trajectory tracking with adaptive control.

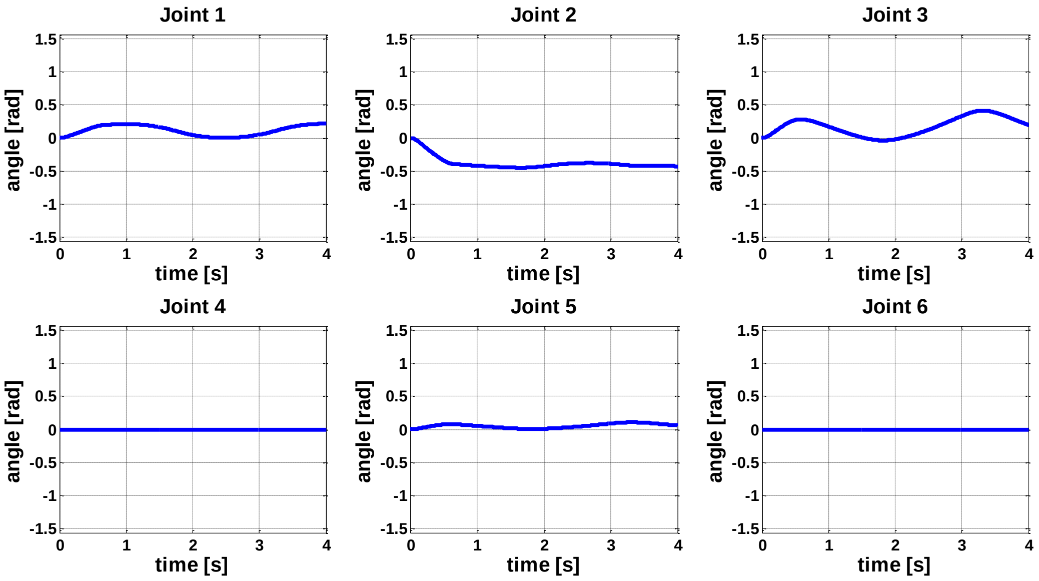


Figure 6.53 : Joint space trajectory tracking with CTC.

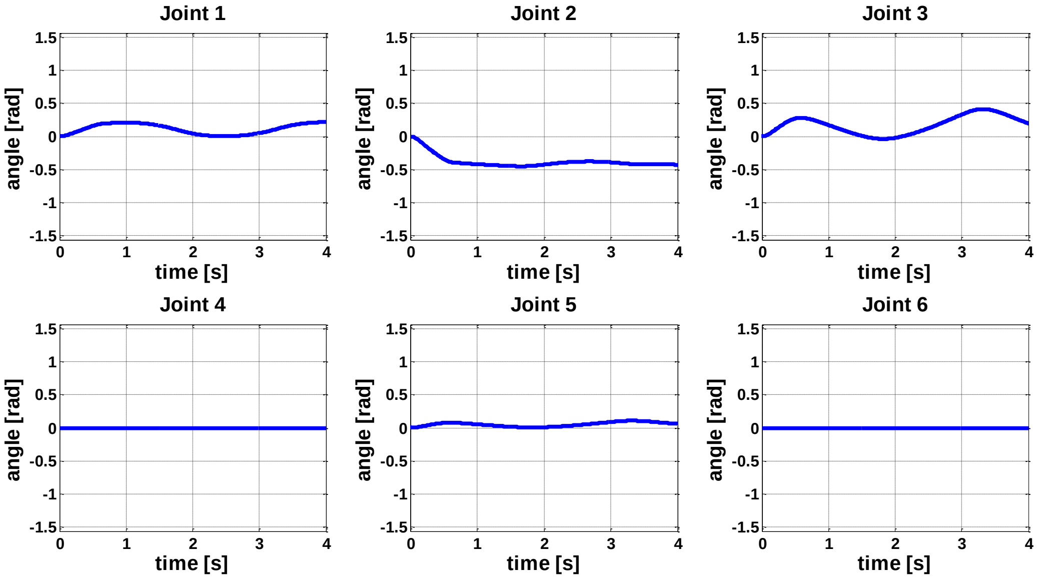


Figure 6.54 : Joint space trajectory tracking with adaptive control.

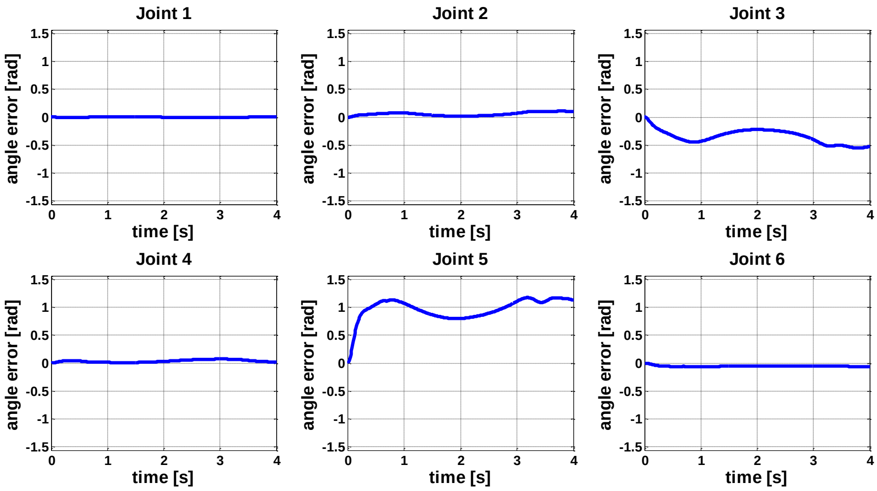


Figure 6.55 : Trajectory tracking position error with CTC.

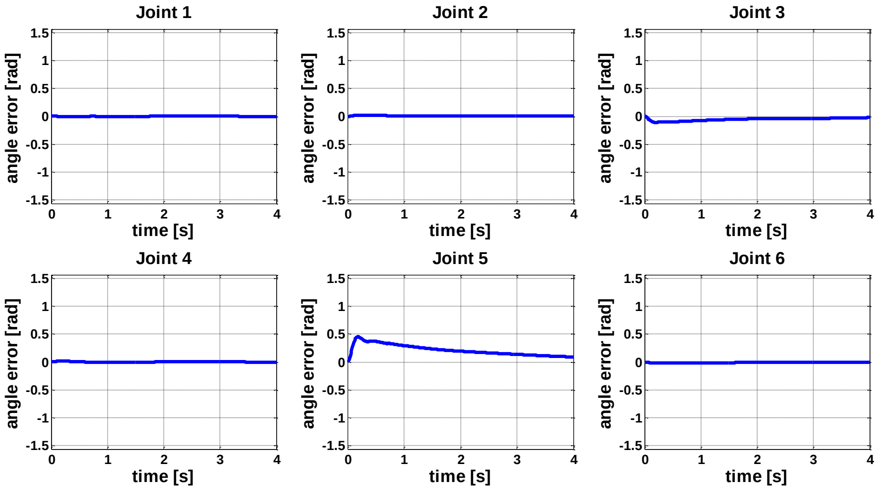


Figure 6.56 : Trajectory tracking position error with adaptive control.

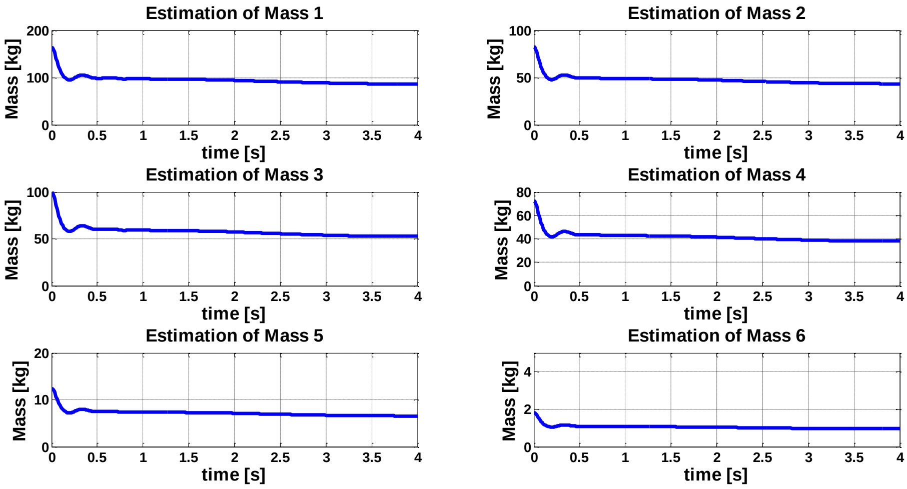


Figure 6.57 : Adaptive control rule link masses’ estimation.

7. CONCLUSIONS AND RECOMMENDATIONS

We analyzed the industrial serial robotic arm with a high performance algorithm (SOA) recursively. Then, we controlled the industrial robotic arm with two strong nonlinear control techniques. The dynamic analysis of serial robot is not very complicated; however when nonlinear control algorithms are included, the system turns out to be a highly complex system. Using this algorithm, the model and its controllers could be easily implemented into any kind of real world robot applications. The visualization of the system was done using the "Virtual Reality Toolbox" in the environment of MATLAB/Simulink. The performance of the robot control algorithms was shown and results were compared in Chapter 7.

This study was very important for two main reasons. The first contribution is to show the possibility to adapt strong background of nonlinear control methods, which were designed for D-H, to be compatible with the usage of SOA, which is novel robot modeling algorithm. The second major importance is to provide us to see the possible results in the simulation and test the algorithms before the production.

In the simulations results, it can be clearly seen that, Adaptive Control can eliminate the disturbance. In this way, it shows much better performance than Computed Torque. However, if the expectations are low, computed torque can be preferred for lightweight robots where masses and frictions are small. Computed torque can also be preferred for the case of faster calculations are demanded, i.e. shorter cycle times are needed, too.

Now that we have the successful models of the controllers, results of their outputs and their 3D simulations, we could easily test these algorithms on a real robotic manipulator in the future studies.

This work can also be extended to include parallel and cooperating manipulators in the future studies. In the future studies, *Delta Robots* and *Stewart Platform* can be our focus as industrial parallel robot. *Baxter* from *Rethink Robotics* could also be our new focus as a cooperating robotic manipulator.

REFERENCES

[1] **Siciliano, B., Sciavicco, L. etc.** (2000). *Modelling and Control of Robot Manipulators,* Springer-Verlag London Limited.

[2] **Craig, J. J.** (2005). *Introduction to Robotics Mechanics and Control*, Third Edition, Pearson Education International, Prentice Hall.

[3] **Spong, M. W., Hutchinson, S. and Vidyasagar, M.** (2005). *Robot Modeling and Control*,First Edition, JOHN WILEY & SONS, INC.

[4] **Lewis, L. F., MUNRO, N.** (2004). *Robot Manipulator Control Theory and Practice*, Second Edition, Revised and Expanded, Marcel Dekker Inc.

[5] **Hartenberg, R. S., and Denavit, J.** (1964). *Kinematic Synthesis of Linkages, New York*, McGraw-Hill.

[6] **Hartenberg, R. S., and Denavit, J.** (1964). *Kinematic Synthesis of Linkages, New York*, McGraw-Hill.

[7] **Ball, R. S.** (1990). *A Treatise on the Theory of Screws*, Cambridge University Press, London.

[8] **Murray, M., Li, Z., and Sastry, S. S.** (1994). *A mathematical introduction to robotic manipulation*, Boca Raton FL:CRC Press, 19-51.

[9] **Featherstone, R.** (1983). The calculation of robot dynamics using articulated-body inertias, *The International Journal of Robotics Research, **2**, 13-30.*

[10] **Rodriguez, G.** (1987). Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics, *IEEE Journal of Robotics and Automation, **6**, 624-639.*

[11] **Jain, A.** (1991). Unified formulation of dynamics for serial rigid multibody systems, *Journal of Guidance, **14**, 531-542*

[12] **Featherstone, R.** (2008). *Rigid Body Dynamics Algorithms*, Springer.

[13] **Featherstone, R. and Orin, D.** (2000). Robot Dynamics: Equations and Algorithms, *Proc. IEEE Int. Conf. Robotics & Automation, San Francisco, CA, 2000, pp. 826-834.*

[14] **Jain , A.** (2011). *Robot and Multibody Dynamics*, Analysis and Algorithms, Springer.

[15] **Rodriguez, G. and Jain, A.** (1991). Spatially Recursive Dynamics for Flexible Manipulators, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation Sacramento.*

[16] **Rodriguez, G., Jain, A., and Kreutz, K.** (1989). Spatial Operator Algebra for Manipulator Modeling and Control, *Jet Propulsion Laboratory/*

California Institute of Technology, 4800 Oak Grove Drive, MS 198-330, Pasadena, CA 91109.

[17] **Rodriguez, G. and Jain, A.** (1991). Kinematics and Dynamics of Under-Actuated Manipulators, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation Sacramento, California - April 1991.*

[18] **Rodriguez, G. and Jain, A.** (1990). Recursive Linearization of Manipulator Dynamics Models, *Jet Propulsion Laboratory/California Institute of Technology 4800 Oak Grove Drive, Pasadena, CA 91109.*

[19] **Rodriguez, G. and Kreutz, K.** (1991). A Spatial Operator Algebra for Manipulator Modeling and Control, *The International Journal of Robotics Research, Vol. 10, No. 4, August 1991.*

[20] **Rodriguez, G., Jain, A., and Kreutz, K.** (1992). Spatial Operator Algebra for Multibody System Dynamics, *The Journal of the Astronautical Sciences, vol. 40, pp. 27-50.*

[21] **Jain, A.** (1991). Unified Formulation of Dynamics for Serial Rigid Multibody Systems, *Journal of Guidance, **14**, 531-542.*

[22] **Rodriguez, G., Kreutz-Delgado, K., and Jain, A.** (1991). Spatial Operator Algebra for Manipulator Modelling and Control, *International Journal of Robotics Research, **10**, 371-381.*

[23] **Rodriguez, G., Meldrum, D. R., and Franklin, F. G.** (1991). An order(n) recursive inversion of the jacobian for an n-link serial manipulator, *In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1175-1180, San Francisco, CA.*

[24] **Jain, A., Rodriguez, G.** (1993). Recursive dynamics algorithm for multibody systems with prescribed motion, *Journal of Guidance, Control and Dynamics, **16**(5), 830-837.*

[25] **Hopler, R. and Thümmel, M.** (2004). Symbolic Computation of the Inverse Dynamics of Elastic Joint Robots, *Proceedings of the 2004 IEEE International Conference on Robotics & Automation New Orleans, LA April 2004.*

[26] **Yesiloglu, S. M.** (2007). *High Performance Dynamical Modeling of Complex Topology Systems*, Ph. D. Thesis, Istanbul Technical University/ Graduate School of Science Engineering and Technology, Istanbul.

[27] **Ozakyol, H.** (2012). Kinematics of Quadruped Walking, M.Sc. Thesis, Istanbul Technical University/ Graduate School of Science Engineering and Technology, Istanbul.

[28] **Khalil, H. K.** (2001).*Nonlinear Systems* (3rd Edition), Macmillan.

[29] **Vidyasagar, M.** (2002). *Nonlinear Systems Analysis (Classics in Applied Mathematics)*, second edition, Prentice-Hall.

[30] **Carter, T. J.** (2009). *The Modeling of a Six Degree-of-Freedom Industrial Robot for the Purpose of Efficient Path Planning*, M.Sc. Thesis, the Pennsylvania State University/ Graduate School of The Harold and

Inge Marcus Department of Industrial and Manufacturing Engineering, Pennsylvania.

[31] **Sabes, P. N.,** (2001). *Linear Algebraic Equations, SVD, and the Pseudo-Inverse*, Neuroscience and Physiology at University of California, San Francisco.

[32] **Baker, K.** (2013). *Singular Value Decomposition Tutorial*, The Ohio State University,Ohio.

[33] **Valverde, S. and Martinez, D.** (2011). *Fundamentals of Robotics Study of a Robot*, Universitat de Girona, Girona.

[34] **ABB Robotics** (2013), IRB 6620's Datasheet, Main Applications.

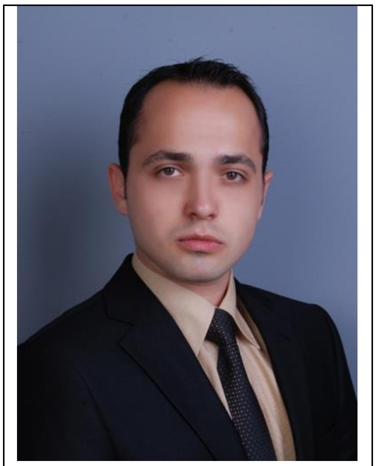
[35] **ABB Robotics** (2013), IRB 6620's Datasheet, Industrial Robot, The agile spot welder.

[36] **ABB Robotics** (2013), IRB 6620's Product Specification Document, Articulated Robot.

Url-1<*http://www.abb.com/product/seitp327/b863009bb251fe17c1257227002f552f.aspx*>, date retrieved 13.12.2013.

Url-2<*http://www.rethinkrobotics.com/products/baxter/*>, date retrieved 12.12.2013.

Url-3<*http://dartslab.jpl.nasa.gov/References/*>, date retrieved 15.12.2013.



CURRICULUM VITAE

Name Surname: Günay YILDIZ

Place and Date of Birth: Kardzhali, Bulgaria, 1988

Address: Bayrampasa/ ISTANBUL

E-Mail: gunayyildiz.itu@gmail.com

B.Sc.: Mechatronics Engineering, Kocaeli University (KOU)