

INTRODUCTION TO

ROBOTICS

Second Edition

About the Author



Born in Malda, West Bengal (India), **Subir Kumar Saha** completed most of his school studies from Vidyasagar Vidyapith, Midnapore (also in West Bengal). He obtained a BE (Mech.) from RE College (now NIT), Durgapur in 1983, followed by a Master's from IIT Kharagpur in 1985. He then obtained a PhD degree from McGill University, Canada, in 1991, and immediately joined the R&D Center of Toshiba Corporation in Japan. At Toshiba, he worked on space robot dynamics. In 1995, he returned to IIT Madras as a Visiting Faculty, before joining IIT Delhi in 1996 as an Assistant Professor. Since 2006 he is a Professor at IIT Delhi, and presently he holds the Naren Gupta Chair Professorship at IIT Delhi.

Prof. Saha is in the Dean's Honors' List of McGill University for his Excellent PhD thesis, and received the Humboldt Fellowship during 1999-2000 when he was at the University of Stuttgart, Germany. He has also been a visiting faculty/researcher to several universities abroad, for example, McGill University, Canada, Monash University, Australia, University of Verona, Italy, and Waseda-IPS, Japan.

Prof. Saha is actively engaged in teaching, research, and technology development. His present book's (*Introduction to Robotics*) first edition was widely acclaimed by readers in India and abroad. The RoboAnalyzer software (<http://www.roboanalyzer.com>) that complements the book was also well received and is very popular. He established the SAE-IIT Delhi Chapter in 1997, Robotics Club in 2002, Mechatronics Laboratory in July 2001, and contributed significantly in setting up the Programme for Autonomous Robotics (PAR) Laboratory in May 2010. His research, consultancy, and training activities with many private companies like Asahi India, Sona Steering, Minda-Huf, SAMTEL Colour Tubes, and public sectors like BHEL, CEPC, and government agencies like DST, MIT, CWDB, Simulator Development Division, BARC/BRNS are clear indicators of the industries' confidence in Prof. Saha's commitment. Prof. Saha has more than 185 research publications in reputed journals and conference proceedings, and has delivered more than 160 invited lectures.

Two of Prof. Saha's special interests are (1) popularizing the concept of engineering education through participation in robotics competitions. In this regard, he has been guiding students of IIT Delhi since 2003 to take part in Doordarshan-Robocon competitions. The team was the champion in 2007 and represented India in the international competition held in Hanoi, Vietnam. To strengthen the concept, Prof. Saha has introduced the concept of *Robotics Competition Based Education in Engineering* (RoCK-BEE) on which he has already delivered about 58 lectures, and published a fiction with his student (www.pothi.com). (2) In order to convert engineering problems faced by the rural people of India and the world into research topics or design challenges, and solve them using modern tools like a software or theory, the concept of *Multibody Dynamics for Rural Applications* or MuDRA was conceived. A monograph on *Dynamics and Balancing of Multibody Systems* by Springer, Germany, was published to demonstrate the concept, and a total of 41 lectures were delivered in India and abroad (Poland, the USA, Mexico, and Japan) in the last seven years.

INTRODUCTION TO ROBOTICS

Second Edition

S K Saha

Professor

*Department of Mechanical Engineering
Indian Institute of Technology (IIT) Delhi
New Delhi*



McGraw Hill Education (India) Private Limited
NEW DELHI

McGraw Hill Education Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto



McGraw Hill Education (India) Private Limited

Published by McGraw Hill Education (India) Private Limited
P-24, Green Park Extension, New Delhi 110 016

Introduction to Robotics, 2e

Copyright © 2014, 2008, by McGraw Hill Education (India) Private Limited.

No part of this publication may be reproduced or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a database or retrieval system without the prior written permission of the publishers. The program listing (if any) may be entered, stored and executed in a computer system, but they may not be reproduced for publication.

This edition can be exported from India only by the publishers,
McGraw Hill Education (India) Private Limited.

ISBN (13): 978-93-3290-280-0
ISBN (10): 93-3290-280-1

Managing Director: *Kaushik Bellani*

Head—Higher Education Publishing and Marketing: *Vibha Mahajan*

Senior Publishing Manager—SEM & Tech Ed.: *Shalini Jha*

Editorial Executive: *Harsha Singh*

Manager—Production Systems: *Satinder S Baveja*

Assistant Manager—Editorial Services: *Sohini Mukherjee*

Senior Production Executive: *Suhail Ali*

Assistant General Manager—Higher Education: *Vijay Sarathi*

Senior Product Specialist: *Sachin Tripathi*

Senior Graphic Designer—Cover: *Meenu Raghav*

General Manager—Production: *Rajender P Ghansela*

Production Manager—*Reji Kumar*

Information contained in this work has been obtained by McGraw Hill Education (India), from sources believed to be reliable. However, neither McGraw Hill Education (India) nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw Hill Education (India) nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw Hill Education (India) and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Typeset at Text-o-Graphics, B-1/56, Aravali Apartment, Sector-34, Noida 201 301, and printed at

Cover Printer:

Contents

<i>Preface</i>	<i>ix</i>
<i>Walk Through</i>	<i>xviii</i>
1. Introduction	1
1.1 History	1
1.2 Robots	2
1.3 Robot Usage	7
<i>Summary</i>	18
<i>Exercises</i>	18
<i>Web-Based Exercises</i>	18
2. Industrial Robots and Their Applications	19
2.1 Robot Subsystems	22
2.2 Classification of Robots	29
2.3 Industrial Applications	34
<i>Summary</i>	40
<i>Exercises</i>	40
<i>Web-Based Exercises</i>	40
3. Actuators and Grippers	41
3.1 Electric Actuators	42
3.2 Hydraulic Actuators	63
3.3 Pneumatic Actuators	66
3.4 Selection of Motors	68
3.5 Grippers	69
<i>Summary</i>	74
<i>Exercises</i>	75
<i>Web-Based Exercises</i>	75
4. Sensors, Vision and Signal Conditioning	76
4.1 Sensor Classification	77
4.2 Internal Sensors	77
4.3 External Sensors	87
4.4 Vision	90
4.5 Signal Conditioning	106
4.6 Sensor Selection	111
<i>Summary</i>	114
<i>Exercises</i>	115
<i>Web-Based Exercises</i>	115

5. Transformations	116
5.1 Robot Architecture	116
5.2 Pose of a Rigid Body	122
5.3 Coordinate Transformation	140
5.4 Denavit and Hartenberg (DH) Parameters	146
5.5 A Variant of DH Parameters	155
5.6 DH Parametrization of Euler angles	159
<i>Summary</i>	159
<i>Exercises</i>	160
<i>MATLAB and RoboAnalyzer Based Exercises</i>	161
6. Kinematics	162
6.1 Forward Position Analysis	163
6.2 Inverse Position Analysis	175
6.3 Velocity Analysis: The Jacobian Matrix	193
6.4 Link Velocities	197
6.5 Jacobian Computation	198
6.6 Jacobian Using the Decoupled Natural Orthogonal Complement (DeNOC)	202
6.7 Forward and Inverse Velocity Analyses	205
6.8 Acceleration Analysis	206
<i>Summary</i>	208
<i>Exercises</i>	208
<i>MATLAB and RoboAnalyzer Based Exercises</i>	208
7. Statics and Manipulator Design	209
7.1 Forces and Moments Balance	210
7.2 Recursive Calculations	212
7.3 Equivalent Joint Torques	213
7.4 Role of Jacobian in Statics	219
7.5 Manipulator Design	222
7.6 Functional Requirements of a Robot	223
7.7 Kinematic and Kinetostatic Measures	225
7.8 Structural Measures	231
7.9 Dynamics and Control Measures	233
<i>Summary</i>	234
<i>Exercises</i>	234
<i>MATLAB Based Exercises</i>	234
8. Dynamics	235
8.1 Inertia Properties	236
8.2 Euler—Lagrange Formulation	243
8.3 Newton—Euler Formulation	257
8.4 Recursive Newton—Euler Algorithm	261
8.5 Dynamic Algorithms	270
<i>Summary</i>	281
<i>Exercises</i>	281
<i>Web-Based Exercises</i>	282

<i>MATLAB Based Exercises</i>	282
<i>RoboAnalyzer Based Exercises</i>	282
9. Recursive Robot Dynamics	283
9.1 Dynamic Modeling	284
9.2 Analytical Expressions	289
9.3 Recursive Inverse Dynamics of RoboAnalyzer	299
9.4 Recursive Forward Dynamics and Simulation	309
<i>Summary</i>	316
<i>Exercises</i>	316
10. Linear Control	318
10.1 Control Techniques	319
10.2 Dynamic Systems	320
10.3 Transfer Function and State-Space Representation	327
10.4 A Robotic Joint	333
10.5 Performance and Stability of Feedback Control	338
10.6 Proportional-Derivative-Integral (PID) Control of a Moving Block	346
10.7 Selection of PID Controller Gains	351
10.8 State-feedback Control	352
10.9 Joint Controllers	357
<i>Summary</i>	363
<i>Exercises</i>	363
<i>MATLAB Based Exercises</i>	364
11. Nonlinear and Force Controls	365
11.1 Control of a Moving Block	366
11.2 Multivariable Robot Control	368
11.3 Stability of Multi-DOF Robot	370
11.4 Linearized Control	371
11.5 Proportional-Derivative (PD) Position Control	372
11.6 Computed-torque (Inverse Dynamics) Control	374
11.7 Feedforward Control	377
11.8 Robust Control	380
11.9 Adaptive Control	380
11.10 Cartesian Control	381
11.11 Force Control	385
11.12 Hybrid Control	394
<i>Summary</i>	395
<i>Exercises</i>	395
<i>MATLAB Based Exercises</i>	396
12. Motion Planning	397
12.1 Joint Space Planning	398
12.2 Cartesian Space Planning	416
12.3 Path Primitives	419
12.4 Cartesian Trajectories	422
12.5 Point-to-Point vs. Continuous Path Planning	424
<i>Summary</i>	424

<i>Exercises</i>	425
<i>MATLAB Based Exercises</i>	425
13. Control Hardware and Robot Programming	426
13.1 Control Considerations	426
13.2 Hardware Architecture	429
13.3 Hardware for Joint Controllers	431
13.4 Computational Speed	433
13.5 Robot Languages	436
13.6 Robot Programming	442
<i>Summary</i>	455
<i>Exercises</i>	455
<i>Web-Based Exercises</i>	455
14. Student Projects	456
14.1 Robotic Contests	457
14.2 Hardware Developments	460
14.3 Software Projects	466
14.4 Commercial Products	472
<i>Summary</i>	474
<i>Exercises</i>	474
<i>Web-Based Exercises</i>	474
Appendix A: Mathematical Fundamentals	475
A.1 Function ‘Atan2’	475
A.2 Vectors	475
A.3 Matrices	478
<i>Summary</i>	479
Appendix B: MATLAB with Allied Toolbox and Software	480
B.1 Use of MATLAB	480
B.2 MuPAD	484
B.3 Simulink	486
B.4 Robotics Toolbox	486
B.5 Recursive Dynamics Simulator (ReDySim)	488
<i>Summary</i>	488
Appendix C: Use of RoboAnalyzer	489
C.1 Visualize Denavit-Hartenberg Parameters	489
C.2 Forward Kinematics	490
C.3 Inverse Kinematics	490
C.4 Inverse Dynamics	491
C.5 Forward Dynamics and Simulation	491
C.6 Joint Trajectory	492
C.7 Virtual Robot Module	492
<i>Summary</i>	492
References	493
Index	498

Preface

Introduction to Robotics is a book which aims to understand the underlying concepts used in designing and building a robot, and to make it work. There are a number of books available in the market which typically cater to either researchers working mainly on the analyses aspects of robots, e.g., kinematics, dynamics, control, etc., or practicing engineers interested in the feasibility study of using robots for particular applications, procurement of robots, their programming, economics, etc. In an undergraduate curriculum of Robotics, it is important that the students are exposed to both the aspects of analyses and applications. Hence, the need was felt for a book which would cover both the aspects in a lucid manner.

Since the first edition of this book *Introduction to Robotics* published in 2008, it has received several praises, mainly from students and faculty in India and abroad. The book has been widely used in Singapore, Mexico (translated in Spanish by Prof. Jorge Eduardo Aguirre Aquilar and Prof. Francisco Javier Standoval Palafox), and the People's Republic of China. I have received comments from several senior well-reputed professors who felt the book to be of very high quality.

Need for a Revised Edition

I have used the book in teaching several courses at IIT Delhi, namely, *Robotics Engineering* to the undergraduate (UG) students, and *Robotics, Mechatronics Product Design* and *Multibody Systems and Vibration Design* to the postgraduate (PG) students. I am sure the book has been used by many in their universities and institutes either as a part of a core or elective course. During my interactions with professors and students of other institutions, it was clear that many of them do not cover the mathematical content, i.e., transformations, forward and inverse kinematics, dynamics, etc., of the book in great detail due to relative difficulty in understanding those contents. In fact, these topics are extremely important from the programming, control, and design point of view. That was the time when the need of a software like RoboAnalyzer for three-dimensional visualization of robot motion was felt. It was developed over the past five years and became popular for its ability to teach robotics in a fun way. It teaches a student the physics of robots first before he or she gets into the mathematical aspects. For example, in Inverse Kinematics, one can learn the concept of multiple solutions by clicking several buttons on the window interface of the software. The software is made available not only through the online portal of this book but also through a dedicated website www.roboanalyzer.com, which gets updated whenever a new module is developed.

Besides, it was noticed that in many institutions, in courses like *Robotics and Automation* or *Applications of Robotics*, etc., there is a need of wider coverage of topics on industrial applications of robots, types of grippers, robot programming, and the like, which are missing in the first edition. Hence, those aspects are included in the present edition. These additions, however, do not majorly change the prescribed roadmap of the first edition to the teachers of UG- and PG-level students belonging to various disciplines.

Suggested Roadmap

A suggested roadmap for this edition is given below:

For UG-level Courses

- At junior level (typically, 2nd and 3rd year students)
 - ◆ Mechanical discipline: Chapters 1–5, 13
 - ◆ Electrical and Computer Science disciplines: Chapters 1–5, 10
- At senior level (typically, 3rd and 4th year students) with above background
 - ◆ Mechanical discipline: Chapters 2, 6–9, 12
 - ◆ Electrical and Computer Science disciplines: Chapters 2, 5–6, 8, 11–12

For PG-level (Master's and Ph. D students) Courses

- With no exposure to a UG-level Robotics course
 - ◆ Mechanical discipline: Chapters 1–5, 8
 - ◆ Electrical discipline: Chapters 1–5, 12
- With exposure to a UG-level Robotics course
 - ◆ Mechanical discipline: Chapters 5–9, 12
 - ◆ Electrical and Computer Science disciplines: Chapters 5–8, 11–12

Besides, the book can be used in courses like “Mechatronics and Automation” (MA), “Multi Body Dynamics” (MBD), and others. For example, Chapters 3–4, 8, 10, and 13–14 would constitute about 90% of MA, whereas Chapters 5–6, 8–9 can cover upto 80% of MBD.

The book can also be used by practicing engineers, as each chapter contains many figures, tables, examples, references, web-based exercises, and on-line support materials.

New to this Edition

The new topics and the features of this second edition are highlighted chapter-wise.

Chapter 1: Introduction Eight new application areas (e.g., Medical, Underwater, etc.) with photos of the robots. The statistics of robots and other dynamic information are updated.

Chapter 2: Serial Robots and Their Applications The title is changed to reflect the addition of new topics on five industrial applications of robots.

Chapter 3: Actuators and Grippers The topic of grippers is added and accordingly, the title of the chapter is changed.

Chapter 4: Sensors, Vision and Signal Processing Vision is expanded to help the students of robotics and/or automation courses, and the industry users of robot systems. Since each sensor signal needs some kind of conditioning before they are displayed or used by the robots for feedback control, etc., they are included in this edition.

Chapter 5: Transformations To understand this topic in a fun way, numerical examples using RoboAnalyzer (RA) software are included. Examples were also solved using MATLAB software and its module MuPAD. An important correlation between the definitions of the Denavit and Hartenberg (DH) parameters presented in this book and some other popular books is drawn which will enable readers of those books to use the RA software and compare their results with this book.

Chapter 6: Kinematics Many RoboAnalyzer (RA)-based examples are included. The inverse kinematics of a spatial six-degrees-of-freedom wrist-partitioned robot adopted from the book on *Fundamentals of Robotic Mechanical Systems* by J Angeles, Springer-Verlag, New York, 2003, are also included because of its implementation in the RA software.

Chapter 7: Statics and Manipulator Design An important aspect of robotics, i.e., its design, was missing in the first edition. The same is now included which will be of immense benefit to practicing engineers working in the area of robotics and robot designs.

Chapters 8 (Dynamics) and 9 (Recursive Robot Dynamics) No major change, except that many more examples including those using RoboAnalyzer, MATLAB, etc., are added.

Chapter 10: Linear Control This chapter is mainly the first part of the earlier Control chapter. It is completely re-written bringing almost all aspects of Linear Control Theory using simple examples of a moving block and a robotic joint.

Chapter 11: Non-linear Control It is a new chapter covering the remaining part of the earlier Control chapter but in a much more expanded form.

Chapter 12: Motion Planning Spline trajectory is re-written, and cycloidal trajectory is added which was used throughout the book to generate numerical results for kinematics and dynamics. It is also the default trajectory in RoboAnalyzer software.

Chapter 13: Control Hardware and Robot Programming This chapter is renamed from “Computers for Robots” (Chapter 12 of the first edition) to justify the new topics, mainly, the expanded descriptions of Robot Programming languages. Examples and illustrations using a real industrial KUKA KR-5 robot available in the Programme for Autonomous Robotics (PAR) Laboratory of IIT Delhi are also included.

Chapter 14: Student Projects It is a new chapter added in this edition, based on the contents of Appendix C in the first edition. New projects, both by the students and those possible using commercially available robots, are included for the benefit of applying the knowledge covered throughout this book into practice.

Appendices “Appendix B: Use of MATLAB and RIDIM” of the first edition is separated into two appendices, namely, “Appendix B: MATLAB and Allied Toolbox and Software” and “Appendix C: Use of RoboAnalyzer.” In the allied toolbox, how to use a freely available **Robotics Toolbox** developed by Prof. Peter Corke of Queensland University of Technology, Australia, is included for the general awareness of students and readers of this book.

References Since the present edition mentions many useful online resource materials, e.g., videos, software, etc., their references are listed separately under the category ‘Web References’ (WR) at the end of the list of references.

Besides, this second edition has been enriched with more figures (310 from 190, i.e., up by ~43%), tables (37 from 23, i.e., up by ~60%), examples (176 from 98, i.e., up by ~80%) and exercises (263 from 195, i.e., up by ~35%) to help the readers grasp the subject in a lucid manner, and practice for better understanding of the concepts explained in the book.

Online Learning Center

The comprehensive OLC can be accessed at <http://www.mhhe.com/saha/ir2> and contains the following material:

For Students

- *RoboAnalyzer Software*
- List of Abbreviations, List of Symbols, author’s website link, Web links for further reading
- Chapter Summary, Web-Based Exercises, Student Projects, Case Studies, Use of MATLAB, Allied Toolbox and Software, and RoboAnalyzer

For Faculty

- *RoboAnalyzer Software*
- Solution Manual
- Chapterwise PowerPoint slides

Acknowledgements

It is my moral responsibility to acknowledge those without whose support the completion of this second edition of the book would not have been possible within a short span of one and a half years. First, the Department of Mechanical Engineering at IIT Delhi and my colleagues are thanked who allowed me to take the sabbatical leave during 2013 by sharing my academic and administrative responsibilities. The book grant from the Quality Improvement Programme of IIT Delhi was a welcome offer to cover the incidental costs of writing the book. While proofreading the typed version of this edition, I must thank my following students who, almost on a war footing, read the manuscripts, corrected/drew figures, created examples whenever required, and typed the equations:

1. Mr Rajeevlochana G Chittawadigi (Graduated as MS student in 2013, and presently working with the 3D PLM Software, Bangalore)
2. Mr Abdullah Aamir Hayat (Presently, a PhD student)
3. Mr Arun Dayal Udai (Presently, a PhD student at IIT Delhi, and a faculty in BIT, Mesra)
4. Mr Riby Abrahama Boby (Presently, a PhD student)

The name of Mr Mangal Sharma (a project staff) must be mentioned here, as he took off all the burden of printing the chapters from me while I was concentrating on writing the chapters.

The names of Mr Dharmendra Jaitly (Technical Staff in the Mechatronics Laboratory), and my other contemporary PhD and MTech students (Mr Paramanand Nandihal, Mr Majid Hameed Koul, Mr Vinay Gupta, Mr Anil Sharma, Mr Vikas

Kharolia) needed special mention due to their time-to-time praise, motivation, and feedback, particularly, during our tea sessions in the laboratory. I wanted to mention the name of my ex-PhD student and presently a faculty member of IIIT Hyderabad (Dr Suril V Shah) who was instrumental in preparing the PowerPoint presentations of the lecture slides in 2010 after the publication of the first edition in 2008. The untold names of the students of various courses taught by me at IIT Delhi or during my lecturing of selected topics on Robotics at IITDM, Jabalpur; IIITM, Gwalior; NIT and CMERI, Durgapur; IISc, Bangalore; IIT Madras; and Waseda-IPS, Japan, who gave important feedback are also acknowledged.

At this stage, I must also thank the reviewers for their valuable comments which have truly raised the standard of this book. Their names are mentioned below:

S Sanyal	<i>National Institute of Technology (NIT) Raipur, Chhattisgarh</i>
Vikas Rastogi	<i>Delhi Technological University, New Delhi</i>
Ashish Dutta	<i>Indian Institute of Technology (IIT) Kanpur, Uttar Pradesh</i>
Arun Dayal Udai	<i>Birla Institute Technology (BIT), Mesra, Jharkhand</i>
S S Roy	<i>National Institute of Technology (NIT) Durgapur, West Bengal</i>
U S Dixit	<i>Indian Institute of Technology (IIT) Guwahati, Assam</i>
Sanjiv Arul	<i>Amrita School of Engineering, Coimbatore, Tamil Nadu</i>
Ajith R R	<i>College of Engineering, Thiruvananthapuram, Kerala</i>
Ramesh S	<i>R V College of Engineering, Bangalore, Karnataka</i>

It will remain incomplete if I do not mention the McGraw Hill Education (India) staff for readily agreeing to bring out the second edition, especially Ms Vibha Mahajan, Ms Harsha Singh, Ms Sohini Mukherjee, Mr Suhaib Ali, and many others in the background, for their persuasion and help.

Finally, I should not forget my family (wife *Bulu* and daughter *Esha*) for their support and patience for not getting my attention while I was engrossed with the book writing.

Feedback Request

At the end, I am quite confident that the readers will find the improved second edition much more reader-friendly and value-added for its detailed content and lucid presentation, particularly, due to the use of RoboAnalyzer software. Even though all efforts were made to minimize typos or any other mistakes, the readers are requested to point out any mistake, if still left, and give their feedback to me directly at saha@mech.iitd.ac.in or sahaitd@gmail.com.

Happy Reading!

SUBIR KUMAR SAHA

Publisher's Note

Remember to write to us! We look forward to receiving your feedback, comments and ideas to enhance the quality of this book. You can reach us at info.india@mducation.com. Kindly mention the title and author's name as the subject. In case you spot piracy of this book, please do let us know.

List of Abbreviations

ABU	Asian Broadcasting Union	ISO	International Standard of Organization
AC	Alternating Current	ISRO	Indian Space Research Organization
ADC	Analog-to-Digital converter	JIRA	The Japan Industrial Robot Association
AGV	Automatic Guided Vehicle	JV	Joint Variable
BRA	British Robot Association	KE	Kinetic Energy
CCD	Charged Coupled Device	KRL	KUKA Robot Language
CID	Charge Injection Device	LED	Light Emitting Diode
CMERI	Central Mechanical Engineering Research Institute	LSPB	Linear Segments with Parabolic Blend
CNC	Computer Numerical Controlled	LSI	Large Scale Integrations
CP	Continuous Path	LTI	Linear Time Invariant
CPL	Computer Programming Language	LVDT	Linear Variable Differential Transformer
CPU	Central Processing Unit	MCI	Matrix of Convective Inertia
CSG	Constructive Solid Geometry	MDH	Modified DH parameters
D	Derivative	MIMO	Multi-Input Multi-Output
DAC	Digital-to-Analog converter	MOS	Metal Oxide Semi-conductor
DAE	Differential Algebraic Equations	MTTF	Mean Time to Failure
DAQ	Data Acquisition	MTTR	Mean Time to Repair
DC	Direct Current	NASA	National Aeronautic Society of America
DeNOC	Decoupled Natural Orthogonal Complement	NC	Normally Closed
det.	Determinant	NE	Newton-Euler
DH	Denavit and Hartenberg	NO	Normally Open
DOF	Degrees-of-freedom	NOC	Natural Orthogonal Compliment
DRDO	Defence Research and Development Organization	ODE	Ordinary Differential Equations
DSP	Digital Signal Processing	OLP	Off-line Programming
EL	Euler-Lagrange	P	Proportional
EMF	Electromotive Force	PAR	Programme for Autonomous Robotics
FKin	Forward Kinematics	PD	Proportional-Derivative
GIM	Generalized Inertia Matrix	PE	Potential Energy
HaPRA	Hanging Planar Robotic Arm	PI	Proportional-Integral
HTM	Homogeneous Transformation Matrix	PID	Proportional-Integral-Derivative
IDyn	Inverse Dynamics		
IKin	Inverse Kinematics		

PLC	Programmable Logic Controller	RUR	Rossum's Universal Robots
PM	Permanent Magnet	RVDT	Rotary Variable Differential
PO	Percentage Overshoot	Transformer	
PR	Prismatic-Revolute	SCARA	Selective Compliance Assembly
PSD	Position-Sensitive Detector	Robot Arm	
PTP	Point-to-Point	SBED	Shoulder-Back/Elbow-Down
PUMA	Programmable Universal Manipulator for Assembly	SBEU	Shoulder-Back/Elbow-Up
RA	RoboAnalyzer (software)	SFED	Shoulder-Front/Elbow-Down
RAM	Random Access Memory	SFEU	Shoulder-Front/Elbow-Up
ReDySim	Recursive Dynamic Simulator	SiRoD	Simple Robot Designer
RIA	Robotics Institute of America	SISO	Single-Input Single-Output
RIDIM	Recursive Inverse Dynamics for Industrial Manipulators	SLI	Structural Length Index
RNE	Recursive Newton-Euler	SPD	Symmetric Positive Definite
ROBOCON	ROBOtic CONtests	TF	Transfer Function
ROV	Remotely Operated Vehicle	WMR	Wheeled Mobile Robot
RP	Revolute-Prismatic	VCI	Vector of Convective Inertia
RPL	Robot Programming Language	VLSI	Very Large Scale Integrations
		5P	Proper Planning Prevents Poor Performance

List of Symbols

γ	the n -dimensional vector of gravitational accelerations
Π^C	power due to the constraint wrenches, \mathbf{w}^C
θ	the n -dimensional vector of generalized coordinates
$\dot{\theta}, \ddot{\theta}$	the n -dimensional vectors of generalized rates and accelerations, respectively
ρ	a Spline function
τ_i	torque at each joint
τ	the n -dimensional vector of generalized forces due to external forces and moments
Ω	the $6n \times 6n$ matrix of angular velocities different than \mathbf{W}
Ω_i	the 6×6 angular velocity matrix for the i^{th} body different than \mathbf{W}_i
ω_e	the 3-dimensional angular velocity vector of the end-effector
ω_i	the 3-dimensional angular velocity vector of the i^{th} body
$\omega_i, \dot{\omega}_i$	the 3-dimensional vectors of angular velocity and acceleration of link i , respectively
\mathbf{a}_i	the 3-dimensional vector denoting point O_{i+1} from O_i
\mathbf{B}_{ij}	the 6×6 twist propagation matrix from $\#i$ to $\#j$
b	damping coefficient
$b_i, \theta_i, a_i, \alpha_i$	Four DH parameters
\mathbf{C}	the $n \times n$ matrix for convective inertia terms (MCI)
c	total number of constraints imposed by p joints
c_i	number of constraints imposed by each joint
\mathbf{c}_i	the 3-dimensional vector denoting the mass center C_i from the origin of fixed frame, F
$\dot{\mathbf{c}}, \ddot{\mathbf{c}}_i$	The 3-dimensional vectors of linear velocity and acceleration of the mass center of link i , respectively
\mathbf{d}_i	the 3-dimensional vector of the mass center, C_i , from the origin point, O_i
e_{ss}	steady-state error
\mathbf{e}_i	the 3-dimensional unit vector along the axis of the i^{th} joint
\mathbf{f}_{ij}	the 3-dimensional vector of resulting force exerted on link i by link j at O_i
\mathbf{g}	the 3-dimensional vector of acceleration due to gravity
h	the n -dimensional vector of convective inertia terms (VCI)
\mathbf{I}	the $n \times n$ Generalized Inertia Matrix (GIM)
\mathbf{I}_i	the 3×3 inertia tensor of the i^{th} body or link about its mass center, C_i
$\tilde{\mathbf{I}}_i$	the 3×3 inertia tensor of the i^{th} composite body consisting of rigidly connected links, $\#i$ to $\#n$
\mathbf{J}	the $6 \times n$ Jacobian matrix for n -DOF robot
k	spring constant

k_p, k_v, k_i	proportional, derivative, and integral gains, respectively
L	Lagrangian
\mathbf{M}	the $6n \times 6n$ generalized mass matrix
$\tilde{\mathbf{M}}$	the $6n \times 6n$ matrix of composite mass matrix
\mathbf{M}_i	the 6×6 mass matrix for the i^{th} body
$\tilde{\mathbf{M}}_i$	the 6×6 mass matrix of the i^{th} ‘composite body’
$\tilde{\mathbf{M}}_w, \tilde{\mathbf{M}}_p, \tilde{\mathbf{M}}_e$	the $6n \times 6n$ matrices, which constitute MCI
m_i	mass of i^{th} body
\mathbf{m}	linear momentum
$\tilde{\mathbf{m}}$	angular momentum
\mathbf{N}	the $6n \times n$ NOC matrix
\mathbf{N}_d	the $6n \times n$ block diagonal matrix of the DeNOC matrices
\mathbf{N}_l	the $6n \times 6n$ lower block triangular matrix of the DeNOC matrices
n	degree of freedom of the whole system
n_i	relative degree of freedom of each joint
\mathbf{n}_{ij}	the 3-dimensional vector of resulting moment exerted on link i by link j at O_i
p	number of kinematic pairs or joints in the system
\mathbf{p}	position vector of a point P . Position vector of any other point is similarly represented.
\mathbf{p}_i	the 6-dimensional joint-motion propagation vector
$[\mathbf{p}]_F$	representation of vector P in the fixed frame F . Similar representation for other frames.
\mathbf{Q}	the 3×3 rotation matrix
\mathbf{Q}_i	the 3×3 rotation matrix transforming frame i to frame $i+1$, i.e., any vector representation in frame $i+1$ is pre-multiplied by this matrix to find the representation in frame i
r	number of rigid bodies or links in the system
\mathbf{r}_i	the 3-dimensional vector of the origin O_{i+1} from the mass center C_i
s	dimension of working space (For planar, $s = 3$; spatial, $s = 6$)
T	kinetic energy
\mathbf{T}	the 4×4 homogeneous transformation matrix
\mathbf{t}	the $6n$ -dimensional vector of generalized twist
\mathbf{t}_e	the 6-dimensional twist vector of the end-effector
\mathbf{t}_i	the 6-dimensional twist vector associated with the i^{th} body
$\dot{\mathbf{t}}_i$	the 6-dimensional twist-rate vector of the i^{th} body
U	Potential energy
\mathbf{W}	the $6n \times 6n$ generalized matrix of the angular velocities
\mathbf{W}_i	the 6×6 angular velocity matrix for the i^{th} body
\mathbf{w}_i	the 6-dimensional vector of wrench acting on the i^{th} body
\mathbf{w}	the $6n$ -dimensional vectors of generalized wrench
$\mathbf{w}^E, \mathbf{w}^C$	the 6-dimensional vectors of external and constraint wrenches, respectively
$\tilde{\mathbf{w}}_i^E$	the 6-dimensional vector of external wrench for the i^{th} composite body
\mathbf{x}, \mathbf{y} , and \mathbf{z}	unit vectors along axes, X , Y and Z , respectively. Similar notations for other unit vectors.
$\mathbf{1}$	the identity matrix of compatible dimension
$\mathbf{0}$	the matrix of compatible dimension whose all elements are zeros
$\mathbf{0}$	the vector of compatible dimension whose all elements are zeros
$[\bullet]^T$	transpose of the argument $[\bullet]$

Walk Through

1

Introduction



The subject of 'Robotics' is extremely relevant in today's engineering curriculum because it is a multidisciplinary science. It is not just concerned with robots, but also with what it means to relieve a human worker from boring, unpleasant, hazardous, or too precise jobs. A robot is normally designed to assist a human worker. In contrast to the general belief, a robot is actually not as fast as humans in most applications. It, however, attains the speed over a very long period of time. As a result, productivity increases if the number of pieces to be produced is very large. Moreover, the intelligence of today's most advanced robot is nowhere near human intelligence. Thus, the introduction of a robot without real understanding of its benefits will be disastrous and is not advisable.

1.1 HISTORY

Even though the idea of robots goes back to ancient times, 3000 years ago, in Indian legend of mechanical elephants (Fuller, 1999), the first use of the word 'robot' appeared in 1921 in the play *Rossum's Universal Robots* (RUR) written by the Czech writer Karel Čapek (1890–1938).

In the play RUR (Capek, 2001), a fictional manufacturer of mechanical creatures designed a robot to replace human workers.

Efficient but totally lacking in emotion, these robots were first thought to be an improvement on human beings since they did as they were told without question.

These robots eventually turned on their masters. They destroyed the human race, saved one man so that he could continue to produce robots. The formula, unfortunately, went lost in the destruction.

The feeling of hatred towards robots seems to exist in many minds even today.

The fear that robots will take away people's jobs might have been the main no immediate development in this field. However, Karel Čapek in his science-fiction stories during the 1940s envisioned the robot as a helper of humankind and postulated three basic rules for robots. These are generally known as the *Laws of Robotics*.

Laws of Robotics

1. A robot must not harm a human being, nor through inaction allow one to come to harm.

Origin of the word 'Robot'
Origin of the word 'robot' can be traced in the Czech word 'robota,' which means forced or compulsory labour.'

Textboxes in each chapter provide historical facts and topic-related information to complement the knowledge gained through the text of the book.

other than typical factory environment. For example, a serial robot mounted on a spacecraft used for retrieval of a faulty satellite or putting it back after repair can be considered a special-purpose robot. The rover Sojourner in Fig. 1.2(b) can also be treated as a special-purpose robot.



(a) Pathfinder Lander with the microrover outside on-board



(b) Sojourner microrover

[Courtesy: <http://mars.jpl.nasa.gov/mfp/>]

Fig. 1.2 Robotic systems for Mars exploration

Other special-purpose robots are classified as follows:

1. Automatic Guided Vehicles (AGVs)

These are mobile robotic systems commonly used in factories for material-handling purposes.

Figures 1.4(a) and (b) show how one such AGV and

its application in material handling, respectively.

An Automatic Guided Vehicle (AGV) is one type of mobile robot which has wheels for its locomotion.

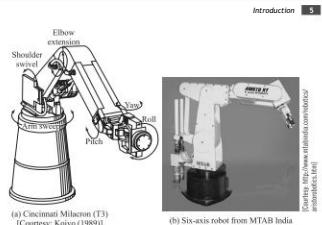


Fig. 1.3 Industrial robots

features like omnidirectional movement capability, etc., are also available, as shown in Fig. 1.5. The Mecanum or omnidirectional wheels, Fig. 1.5(a), in contrast to the two degrees of freedom (DOF) wheels used in the automobile and other AGVs, provide three-DOF; i.e., such AGVs can move sideways also, as illustrated in Fig. 1.5(b). AGVs are also used in hospitals for nursing, security, and other applications.

AGV vs. Walking Robot
A walking robot is another type of mobile robot, where legs are used for locomotion, instead of wheels. In the AGV, however, a walking robot is different than an AGV or a WMR.



(a) Stand-alone



(b) At work (picking up car doors)

[Courtesy: <http://www.agvsoft.com>]

Photos of practical robots and their real applications give true understanding and utility of the subject covered in this book.

Control Hardware and Robot Programming 455

model can be moved around to generate a sequence of robot movements that can be calibrated and programmed for an actual peg in a hole-insertion task.

SUMMARY

In this chapter, the software and hardware requirements for robot control are explained. Different robot-programming languages are explained, along with online and offline robot-programming methodologies. Advantages and disadvantages of both programming are mentioned. Examples with a KUKA robot are also provided.

EXERCISES

13.1 What are the hardware and software requirements for a robot controller?
 13.2 How can computing speed associated with trigonometric function be enhanced?
 13.3 Explain controller hardware structure of PUMA robot.
 13.4 Describe online programming.
 13.5 What are the advantages and disadvantages of online programming?
 13.6 What are the types of offline programming?
 13.7 What is the difference between lead-through and walk-through programming?
 13.8 What are the typical features of a teach pendant?
 13.9 What are the types of offline programming?
 13.10 What are the features offline programming environments should have?
 13.11 When one should prefer online over offline programming?
 13.12 Name a few robot languages and mention their features.
 13.13 What are the generations of robot languages?
 13.14 Mention the feature of generation of robot languages.
 13.15 Why is task-level programming difficult?

WEB-BASED EXERCISES

13.16 What are the types of robot-programming languages used by commercial robot manufacturers?
 13.17 Name some robot simulation software for offline programming (other than mentioned in the chapter).
 13.18 Find control architecture of KUKA KR C2 controller.
 13.19 Which research facilities attempted programming using augmented reality?
 13.20 Explain at least two assistive devices for walk-through programming.

Summary at the end of each chapter gives a glimpse of what has been explained in that chapter.

Web-based exercises will keep the readers up-to-date with the latest developments in this area.

Selection of components and necessary calculations give a student a practical approach to the subject, and provide a practicing engineer useful information for real product applications.

7.1 Introduction to Robotics

$$\rho = \frac{110187.12}{2 \times \pi \times (0.1)^2} = 175.35 \text{ N/m}^2 \quad (3.7)$$

3.5.4 Adhesive Grippers

An adhesive substance used for a grasping action can be used to handle fabrics and other lightweight materials. One of the limitations is that the adhesive substance loses its effectiveness with repeated use. Hence, it has to be continuously fed like a mechanical typewriter's ribbon which needs to be attached to the robot's wrist.

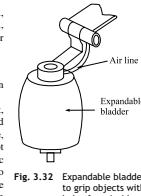


Fig. 3.32 Expandable bladder to grip objects with holes from inside

3.5.6 Selection of Grippers

Some of the criteria to be used in selecting an appropriate gripper are highlighted below:

- One needs to decide the drive system. In fact, in an industrial scenario where electric and pneumatic power sources are easily available, one needs to decide the type. If fast but not so accurate motion is desired then pneumatic should be preferred. Otherwise, one can go for electric. For heavy objects, certainly one must prefer hydraulic, as pointed in Section 3.2.
- Object or the part surface to be gripped must be reachable by a gripper.
- The size variation should be kept in mind. For example, a cast object may be difficult to put in a machine chuck.
- The gripper should tolerate some dimension change. For example, before and after machining a workpiece, their sizes change.
- Quality of surface area must be kept in mind. For example, a mechanical gripper may damage the surface area of an object.
- Force should be sufficient to hold an object and should not fly away while moving with certain accelerations. Accordingly, the materials on the surfaces of the fingers should be chosen.

SUMMARY

A robot is moved by actuators. Different forms of actuators, namely, electric, hydraulic and pneumatic types are explained. For electric motors, their typical specifications and

282 Introduction to Robotics

WEB-BASED EXERCISES

Based on Web search find the answers to the following questions:

8.11 Find at least three commercial software that are capable of performing dynamics of a robot manipulator.
 8.12 What are other possible dynamic formulations (methodologies) for robot dynamics?
 8.13 What are the commercial software for a specific architecture of a robot?

MATLAB BASED EXERCISES

8.14 Find the joint torques for the manipulator in Exercise 8.3.
 8.15 Find simulation results for the three-link manipulator while no torque is applied but the gravity is acting. The initial conditions for the generalized coordinates are $\dot{\theta}_1(0) = \dot{\theta}_2(0) = \dot{\theta}_3(0) = 0$ rad; $\ddot{\theta}_1(0) = \ddot{\theta}_2(0) = \ddot{\theta}_3(0) = 0$ rad/sec.
 Take the geometric and inertial parameters as
 $a_1 = a_2 = 1$ m; $a_3 = 0.5$ m; $m_1 = m_2 = 1$ kg; $m_3 = 0.5$ kg
 8.16 Write a program to find joint torques and forces for the PR manipulator based on the equations of motion derived in Exercise 8.4. While the parameters are defined using Eq. (1.22), Take $\theta(0) = 0$; $\dot{\theta}(0) = \pi/2$; $\ddot{\theta}(0) = 0$ rad; a and b are respectively the joint variables for the revolute and prismatic joints. Consider $T=10$ sec.
 8.17 Repeat Exercise 8.16 for the PR robot manipulators of Exercise 8.6.

ROBOANALYZER BASED EXERCISES

8.18 Validate the results of Exercise 8.14. Visualize the animation.
 8.19 Validate the forward dynamics results of two-link RP and PR arms whose dynamic equations were derived in Exercises 8.5 and 8.6, respectively. Take numerical data from Exercises 8.16.
 8.20 Perform inverse and forward dynamics of the KUKA KR-5 robot available in RoboAnalyzer environment.

MATLAB-based exercises enhance the problem-solving skill with ease.

C

Use of RoboAnalyzer¹



In this appendix, the steps to test the RoboAnalyzer software developed by the author and his students at IIT-Delhi are explained. It is an improved version of the BiDDIM (Reactive Inverse Dynamics for Industrial Manipulators) programs appeared in the first edition of this book in 2008. RoboAnalyzer (RA) has the visualization feature through 3-dimensional models of robots, including many standard robots like KUKA, ABB, Fanuc, and others. It can be used to learn DH parameters, kinematics, and dynamics of robots. It generates the 3-dimensional (3D) animation and graph plots as outputs. In essence, learning the physics of robotics with joy is emphasized through RA, rather than only mathematical derivations.

RoboAnalyzer can be installed on a computer with windows operating system by downloading from the website mentioned in the footnote.



Fig. C.1 Robot-model selection and redefine the DH parameters

The following sections explain the main features of RoboAnalyzer.

C.1 VISUALIZE DENAVIT-HARTENBERG PARAMETERS

After selecting a robot and redefining its Denavit-Hartenberg (DH) parameters, as shown in Fig. C.1, users can visualize each DH parameter by selecting a joint and

¹ RoboAnalyzer (RA) software and the users' manual can be downloaded free from <http://www.roboanalyzer.com>. Also available from <http://www.iaitbhu22.iitd.ac.in/~seaslab/ra22>. Seated students since 2008 (IAIT-BHU).

MATLAB programming examples help a student to master solving complex problems with ease.

In-house developed software usage not only allows the reader to get results for complex problems but also provides the possibility of writing new programs based on the existing ones.



Example 6.19 Inverse Kinematics of the Articulated Arm Using MATLAB

In order to solve Example 6.18, a MATLAB program can be written, as shown in Fig. 6.16, which can be stored in a file, say, 'ch6ikin3aa.m' that can be run to yield the above results. Moreover, by changing values of "a2," "a3," and "px," "py," "pz" that correspond to the robot's DH parameters and the end-effector position, respectively, many other solutions can be generated.

```
%Non-zero constant DH parameters
a2=1; a3=1;
%Input
px=1; py=0; pz=0;
%Intermediate calculation
delxy = px*pz-py*pz; delz=delxy+pz*pz;
%Calculations for theta_1
th11atan2(px,py);
th12=pi+atan2(py,pz);
%Calculations for theta_2
c21=(a2*a3*cos(th11))+px-a3*s3*delxy/del; c21=(a2*a3*cos(th11));
s22 = (- ( a2 + a3 * cos ( th11 ) ) * p z + a3 * s3 * delx ) / del;
c22=(a2*a3*cos(th11))+delxy-a3*s3*px/del;
th21=atan2(c21, th22*tan2(s22, c22));
th22=pi-th21; th23=pi-th22;
%angles in degrees
r2d=180/pi;
th11=th11*r2d; th12=th12*r2d; th21=th21*r2d; th22=th22*r2d;
th23= th23*r2d; th24 = th24*r2d; th31=th31*r2d; th32=th32*r2d
```

Fig. 6.16 Texts of file 'ch6ikin3aa.m' for inverse kinematics of articulated arm

6.2.4 A Wrist

Consider the wrist architecture shown in Figs. 5.38 and 6.16, whose kinematic relations are given in Eq. (6.12b). It is desired to find the joint variable, θ_1 , θ_2 , and θ_3 , corresponding to a given end-effector orientation, \mathbf{Q} , with the following form:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \end{bmatrix} \quad (6.41)$$

can be interpreted as the projection of \mathbf{a} onto the vector \mathbf{b} whose result is multiplied by the magnitude of the latter. Now, if these two vectors are orthogonal to each other, i.e., if $\theta = 90^\circ$, the dot product vanishes, namely, $\mathbf{a} \cdot \mathbf{b} = 0$.

A.2.3 Vector or Cross Product

A vector or cross product between two Cartesian vectors, say, \mathbf{a} and \mathbf{b} , denoted by \mathbf{c} , is defined as

$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k} \quad (\text{A.11a})$$

where \times denotes the symbol for the cross product, whereas $\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$ represents the determinant of the arguments \mathbf{a} and \mathbf{b} . The result of Eq. (A.11a) can also be expressed as

$$\mathbf{c} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} \quad (\text{A.11b})$$

In Eq. (A.11a), the result of the cross product is also a vector \mathbf{c} , as indicated in Eq. (A.11b), which is orthogonal to the two vectors \mathbf{a} and \mathbf{b} . The magnitude of vector \mathbf{c} is denoted as c that can be given by

$$c = ab \sin \theta \quad (\text{A.12})$$

where θ is the angle between the vectors \mathbf{a} and \mathbf{b} , as shown in Fig. A.3. In order to obtain the direction of the resultant vector \mathbf{c} , the right-hand rule is applied, i.e., if the palm of a right hand is placed along the vector \mathbf{a} , and then turned towards the vector \mathbf{b} , the thumb points out in the direction of the vector \mathbf{c} . It is also shown in Fig. 13.13.

The cross product has the following properties:

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}; \mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a}' \mathbf{b}') \mathbf{c} - (\mathbf{a}' \mathbf{c}') \mathbf{b}; \mathbf{a}' (\mathbf{b} \times \mathbf{c}) = \mathbf{a} \mathbf{b}' \mathbf{c}' \quad (\text{A.13})$$

A.2.4 Cross-Product Matrix

A cross-product matrix is always associated with a three-dimensional Cartesian vector, say, \mathbf{a} . When the matrix is pre-multiplied with another three-dimensional Cartesian vector, say, \mathbf{b} , it results in the cross product between the two vectors, as shown in Section A.2.3. If $\mathbf{a} \times \mathbf{I}$ denotes the cross-product matrix associated with the vector \mathbf{a} then

$$(\mathbf{a} \times \mathbf{I})\mathbf{b} = \mathbf{a} \times \mathbf{b} \quad (\text{A.14a})$$

The 3×3 matrix $(\mathbf{a} \times \mathbf{I})$ is a skew-symmetric matrix and singular. Its representation in terms of the components of the vector $\mathbf{a} = [a_1 \ a_2 \ a_3]^T$ is given by

$$\begin{bmatrix} 0 & -a_3 & a_2 \\ 0 & -a_1 & a_3 \\ 0 & -a_2 & a_1 \end{bmatrix}$$

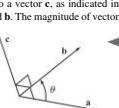


Fig. A.3 Vector (cross)-product of two Cartesian vectors

Mathematical Fundamentals quickly revise the basics of mathematics required for the topics covered in the book without the need of referring other books immediately.



Robots built by the students give readers the confidence to build their own robots that will certainly enhance their knowledge about the subject.



used a pulley arrangement to lift the gripper assembly, while a four-bar parallelogram mechanism was used in the manual robot. The latter could extend almost about half a meter in front of the robot to be able to place the blocks inside the boundary line of the 10-sided polygon. Once the design was done, the next challenge was to fabricate, assemble, program, and make them run successfully. Steps similar to Section 14.1.1 were followed as per the Gantt chart shown in Fig. 14.6.

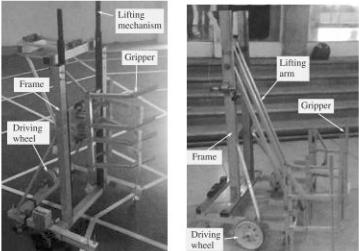


Fig. 14.5 Robots for ROBOCON 2007

14.2 HARDWARE DEVELOPMENTS

In this section, several other hardware developments of robots are explained.

14.2.1 RoboMuse

RoboMuse, shown in Fig. 14.7 is a live-following mobile robot capable of handling few kilograms of payload. It was originally manufactured as a part of ROBOCON 2008 by IIT Delhi's robotics team. After participation in the event, a live 24x7 demo of the robot was planned in the Student Activity Centre of the campus. The salient features of the RoboMuse were as follows:

1. It had three Maxon motors. Two were for driving wheels and one to lift the charging rod up and down.
2. Supply voltage was 12 V dc from a lead-acid battery (current capacity: 4.3

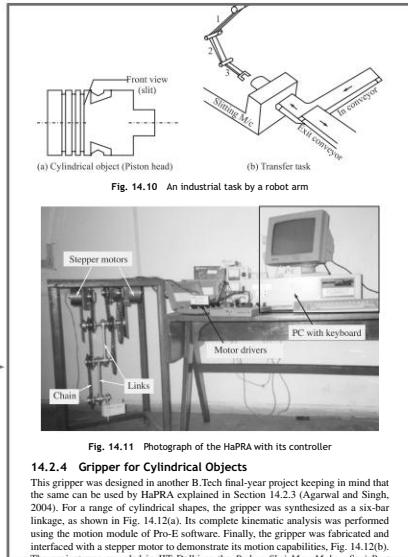


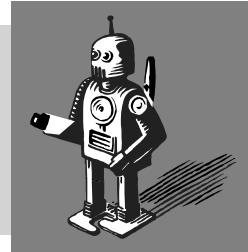
Fig. 14.10 An industrial task by a robot arm

Fig. 14.11 Photograph of the HaPRA with its controller

14.2.4 Gripper for Cylindrical Objects

This gripper was designed in another B.Tech final-year project keeping in mind that the same can be used by HaPRA explained in Section 14.2.3 (Agarwal and Singh, 2004). For a range of cylindrical shapes, the gripper was synthesized as a six-bar linkage, as shown in Fig. 14.12(a). Its complete kinematic analysis was performed using the motion module of Pro-E software. Finally, the gripper was fabricated and interfaced with a stepper motor to demonstrate its motion capabilities. Fig. 14.12(b).

Introduction



The subject of ‘Robotics’ is extremely relevant in today’s engineering curriculum because of the robots’ ability to perform tireless dangerous jobs. A robot is meaningful only when it is meant to relieve a human worker from boring, unpleasant, hazardous, or too precise jobs. A robot is normally designed to assist a human worker. In contrast to the general belief, a robot is actually not as fast as humans in most applications. It, however, maintains the speed over a very long period of time. As a result, productivity increases if the number of pieces to be produced is very large. Moreover, the intelligence of today’s most advanced robot is nowhere near human intelligence. Thus, the introduction of a robot without real understanding of its benefits will be disastrous and is not advisable.

1.1 HISTORY

Even though the idea of robots goes back to ancient times, 3000 years ago, in Indian legend of mechanical elephants (Fuller, 1999), the first use of the word ‘robot’ appeared in 1921 in the play *Rossum’s Universal Robots* (RUR) written by the Czech writer Karel Capek (1890–1938).

In the play RUR (Capek, 2001), a fictional manufacturer of mechanical creatures designed a robot to replace human workers. Efficient but totally lacking in emotion, these robots were first thought to be an improvement over human beings since they did as they were told without question. These robots eventually turned on their masters. They destroyed the human race, saved one man so that he could continue to produce robots. The formula, unfortunately, had been lost in the destruction.

The feeling of hatred towards robots seems to exist in many minds even today. The fear that robots will take away people’s jobs might have resulted in no immediate development in this area. However, Isaac Asimov in his science-fiction stories during the 1940s envisioned the robot as a helper of humankind and postulated three basic rules for robots. These are generally known as the ‘Laws of Robotics.’

Laws of Robotics

1. A robot must not harm a human being, nor through inaction allow one to come to harm.

Origin of the word ‘Robot’

Origin of the word ‘robot’ can be traced in the Czech word ‘robota,’ which means ‘forced’ or ‘compulsory labour.’

2. A robot must always obey human beings, unless that is in conflict with the first law.
 3. A robot must protect from harm, unless that is in conflict with the first two laws.
- A fourth law was later introduced by Fuller (1999) as
4. A robot may take a human being's job but it may not leave that person jobless!

Attempts are being made to adhere to these laws of robotics, but there is no automatic way to implement them. For instance, the military robot, by its very nature, is likely to be designed with the intention of breaking these laws. Most industrial robots of today are designed to work in environments which are not safe and very difficult for human workers. For example, a robot's hand can be designed to handle very hot or very cold objects that the human hand cannot handle safely. Inspired by Asimov's books on robots, Joseph F. Engelberger tried to design a working robot in the 1950s. He, along with George C. Devol, started the UNIMATION Robotics Company in the USA in 1958. The first Unimate robot, however, was installed in

Why the name UNIMATION?

The word UNIMATION is the contraction of the two terms, UNIversal and autoMATION, because of the belief that the robot is a universal tool that can be used for many kinds of tasks.

1961 in a General Motors automobile factory in New Jersey. It was an automated die-casting mould that dropped red-hot door handles and other such car parts into pools of cooling liquid on a line that moved them along to workers for trimming and buffing. Its most distinct feature was a grip on a steel armature that eliminated the need for a person to touch car parts just made from molten steel. It had five degrees-of-freedom (DOF), but there were few applications in which six-DOF were required. Figure 1.1 shows such a six-DOF Unimate robot.

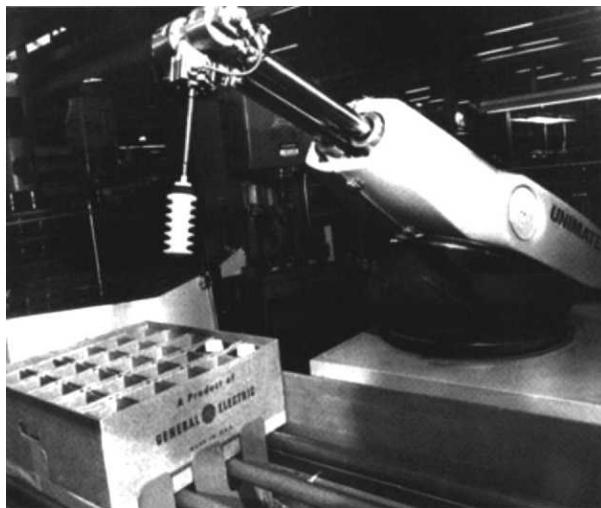
Since then robotics has evolved in multitude directions, starting from using it in welding, painting, assembling, machine-tool loading and unloading, inspection, agriculture, for nursing purposes, medical surgery, military, security, undersea and space explorations, and many more. A majority of them are still used in welding (~25%) and assembly (~33%). Figures 1.2 (a) and (b), respectively, show the Mars Pathfinder Lander and the Sojourner microrover by the National Aeronautics and Space Administration (NASA), USA.

1.2 ROBOTS

A robot is formally defined by the International Standard of Organization (ISO) as a *reprogrammable, multifunctional manipulator* designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks. There exist several other definitions too given by other societies, e.g., by the Robotics Institute of America (RIA), The Japan Industrial Robot Association (JIRA), British Robot Association (BRA), and others. All definitions have two points in common. They are 'reprogrammability' and 'multifunctionality' of robots.

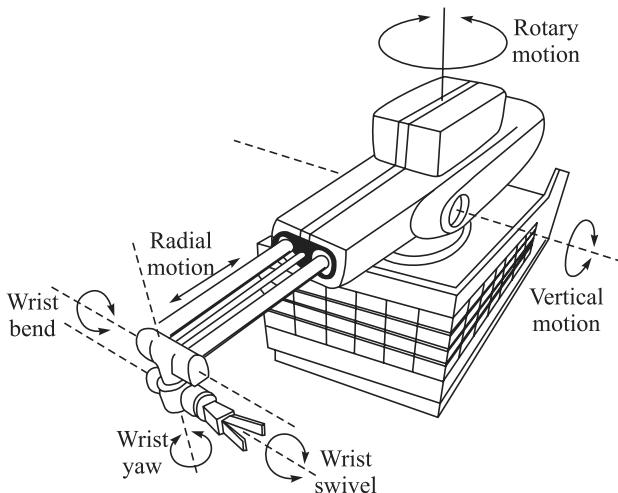
CNC machine tools vs. Robots

'Reprogrammability' and 'Multifunctionality' abilities of robots make them flexible, and different from the Computer Numerical Controlled (CNC) machine tools.



[Courtesy:<http://www.robottalloffame.org/unimate.html>]

(a) Photo of a Unimate robot



(b) Sketch of Unimate robot [Courtesy: Critchlow (1985)]

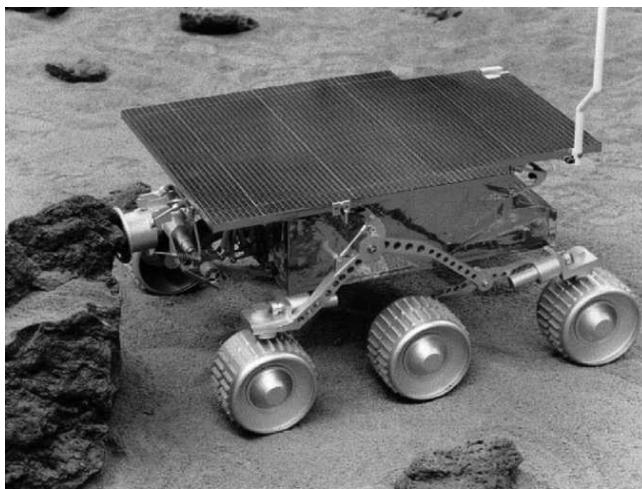
Fig. 1.1 Unimate robot

Robots are broadly classified as *industrial* and *non-industrial* or *special-purpose*. A typical industrial robot made by Cincinnati Milacron of the United States of America (USA) is shown in Fig. 1.3(a), whereas an Indian robot suitable for light industrial applications is shown in Fig. 1.3(b). Strictly speaking, a manipulator which when controlled by a computer is called a robot. During robotics study, however, a manipulator like the one shown in Fig. 1.3 (a) or (b) is always assumed to be computer controlled. Hence, it can be regarded as a robot. Industrial robots are intended to serve as general-purpose, unskilled or semiskilled labour, e.g., for welding, painting, machining, etc. Alternatively, a special-purpose robot is the one that is used in places

other than typical factory environment. For example, a serial robot mounted on a spacecraft used for retrieval of a faulty satellite or putting it back after repair can be considered a special-purpose robot. The rover Sojourner in Fig. 1.2(b) can also be treated as a special-purpose robot.



(a) Pathfinder Lander with the microrover outside on-board



(b) Sojourner microrover

[Courtesy: <http://mars.jpl.nasa.gov/MPF/>]

Fig. 1.2 Robotic systems for Mars exploration

Other special-purpose robots are classified as follows:

1. Automatic Guided Vehicles (AGVs)

These are mobile robotic systems commonly used in factories for material-handling purposes. Figures 1.4(a) and (b) show one such AGV and its application in material handling, respectively. Such AGVs generally follow a wire-guided path on shop floors. There are also autonomous AGVs that do not require a wired path. Additional

Is AGV same as WMR?

An Automatic Guided Vehicle (AGV) is one type of mobile robot, which has wheels for its locomotion. Hence, it is also called Wheeled Mobile Robot (WMR). As a result, an AGV is same as WMR.

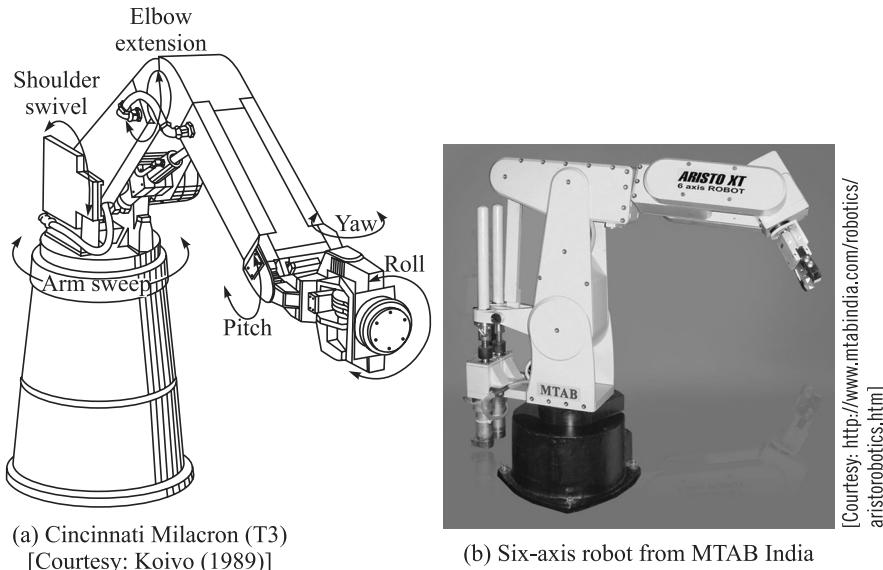


Fig. 1.3 Industrial robots

features like omnidirectional movement capability, etc., are also available, as shown in Fig. 1.5. The Mecanum or omnidirectional wheels, Fig. 1.5(a), in contrast to the two degrees-of-freedom (DOF) conventional wheels used in the automobiles and other AGVs, provide three-DOF, i.e., such AGVs can move sideways also, as illustrated in Fig. 1.5(b). AGVs are also used in hospitals for nursing, security, and other applications.

AGV vs. Walking Robot

A walking robot is another type of mobile robot, where legs are used for its locomotion, instead of wheels in the AGV. Hence, a walking robot is different than an AGV or a WMR.

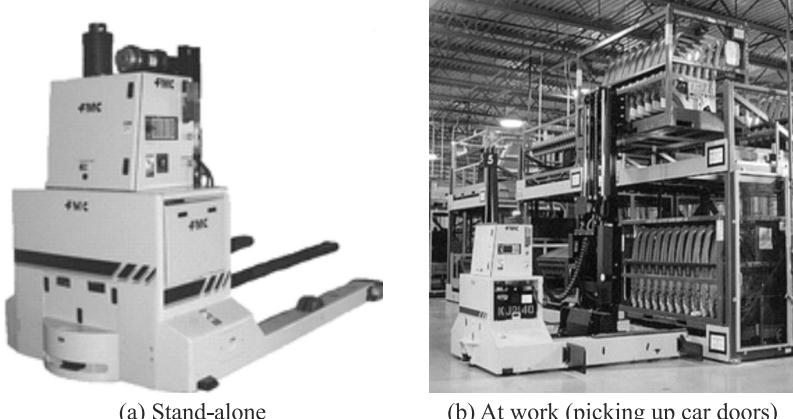


Fig. 1.4 An AGV

2. Walking Robots These robots walk like human beings, as shown in Fig. 1.6. They are used in military, undersea exploration, and places where rough terrains exist.

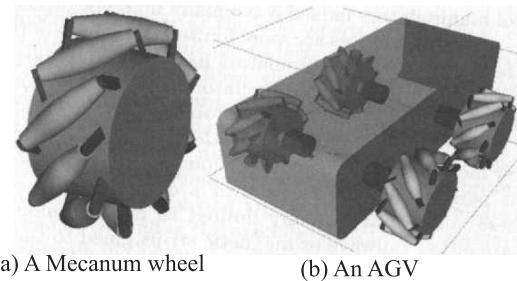


Fig. 1.5 An AGV with Mecanum wheels [Courtesy: Angeles (2003)]

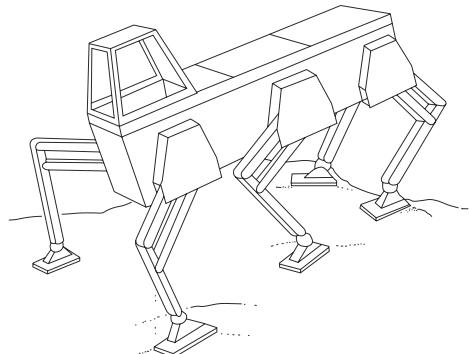


Fig. 1.6 The Ohio State University walking robot, Hexapod [Courtesy: Todd (1985)]

3. Parallel Robots As the name suggests, these robots have parallel configuration, in contrast to the serial-like structure of an industrial robot shown in Fig. 1.1 or 1.3. In this sense, a walking robot with all its legs touching the ground is a parallel robot. However, a parallel structure with a fixed base is used as a flight simulator to train pilots of aeroplanes. As shown in Fig. 1.7, it has six legs to control the moving platform. It is also used as a driving simulator for truck and other types of ground vehicles. Such robots find applications in machine tools and medical surgery to reduce the trembling of a surgeon's hand during operation, etc. Figure 1.8 shows the use of a parallel robot as a milling machine.



Fig. 1.7 A parallel robot as a flight simulator



[Courtesy: <http://biotsavart.tripod.com/hexapod.htm#HISTORY>]

Fig. 1.8 A parallel robot as a milling machine

In this book, only serial-type industrial robots shown in Figs. 1.1 and 1.3 will be studied. However, the tools and knowledge acquired from this book can be used to study and analyze other robots like AGVs, parallel robots, and others.

1.3 ROBOT USAGE

Robots of any type, industrial or non-industrial, are neither as fast nor as efficient as special-purpose automated machines. However, they are easily re-trained or re-programmed to perform an array of different tasks, whereas an automated special-purpose machine, including a CNC machine, can perform only a very limited class of tasks. It is the degree of re-programmability that differentiates a robot from a CNC machine tool. There is, however, no internationally recognized demarcation line. The question then remains when to consider a person, a robot, or a specialized machine to perform a certain job. The answer to this question is not simple and straightforward. Some rules of thumb can help suggest significant factors to be kept in mind.

Thumb Rules on the Decision of Robot Usage

1. The first rule to consider is known as *Four D's of Robotics*, i.e., is the task dirty, dull, dangerous, or difficult? If so, a human will probably not be able to do the job efficiently for hours. Therefore, the job is appropriate for automation or robotic labor.
2. The second rule is that a robot may not leave a human jobless. Robotics and automation must serve to make our lives more enjoyable, not miserable.
3. A third rule involves asking whether you can find people who are willing to do the job. If not, the job is a candidate for automation or robotics. Indeed this should be a primary reason for the growth of automation and robotics.
4. A fourth rule of thumb is that the use of robots or automation must make short-term and long-term economic sense.

So, as a general starting point, the following may be considered: A task that has to be done only once or a few times and is not dangerous probably is best done by a human. After all, a human is the most flexible of all machines. A task that has to be done a few hundred to a few hundred thousand times is probably best done by a flexible automated machine such as an industrial robot. A task that has to be done one million times or more is probably best handled by building a special-purpose hard-automated machine.

1.3.1 Applications

Throughout the world, robots are mainly used in automobile and other manufacturing industries, as depicted in Fig. 1.9. Most of them are serial in nature like a human arm. Different serial-type robots and their applications in industries, for example, in welding, painting, debarring, etc., are explained in Chapter 2. However, the present-day robots are expected to perform other type of jobs as well, e.g., medical surgery, space explorations, etc. They are explained next.

1. Medical Medical robots have found applications mainly in surgery. As shown in Fig. 1.10, a robot is posing to perform a laparoscopic surgery. The goal of surgical robotics is not to replace the surgeon with a robot, but to provide the surgeon with a new set of very versatile tools that extend his or her ability to treat patients. Hence, medical robot systems are called *surgical assistants* that work cooperatively with surgeons. Currently, there are two main varieties of surgical assistant robots. The first one is called a *surgeon extender*, which augments the surgeon's ability to manipulate surgical instruments during surgery. This eliminates hand tremor or the ability to perform dexterous operations inside the patient's body. As a result, casualty rates are reduced, and operative times are shortened. The second variety is called *auxiliary surgical support* which works side by side with the surgeon and performs functions like holding endoscopes, etc. A surgical robot must be compatible with the operating theatre. It must be placed where it can work on the patient while also allowing access by clinical staff.

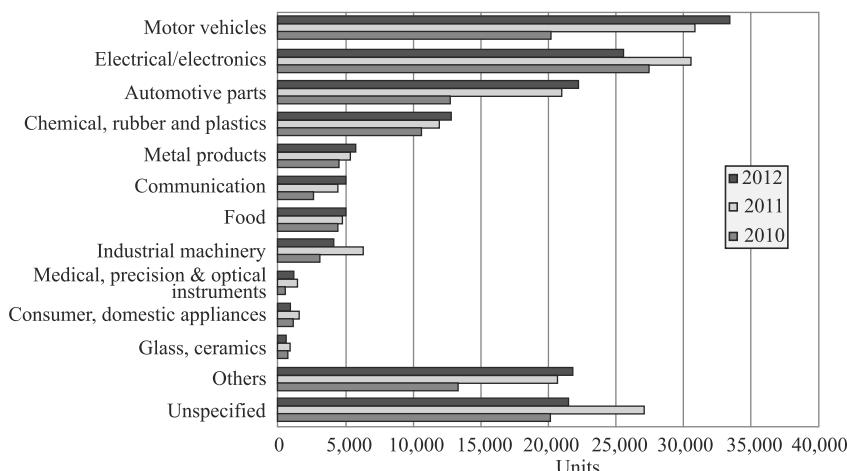
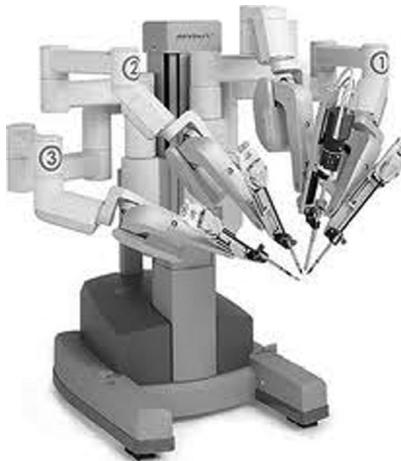


Fig. 1.9 Applications of industrial robots during 2010–12

[Courtesy: <http://www.ifr.org/industry-robots/statistics/>]



[Courtesy: <http://www.intuitivesurgical.com/products/>]

Fig. 1.10 The *da Vinci* robot for minimally invasive surgery

2. Mining In order to enhance productivity, access to unworkable mineral seams, and reduce human exposure to the inhospitable environment of dust, noise, gas, water, moving equipment and roof-fall robots are used. In coal mining, room-and-pillar mining is accomplished by repetition of a well-defined cycle of cutting coal, removing coal, and supporting the roof. Figure 1.11 shows an autonomous truck used in mining industries.

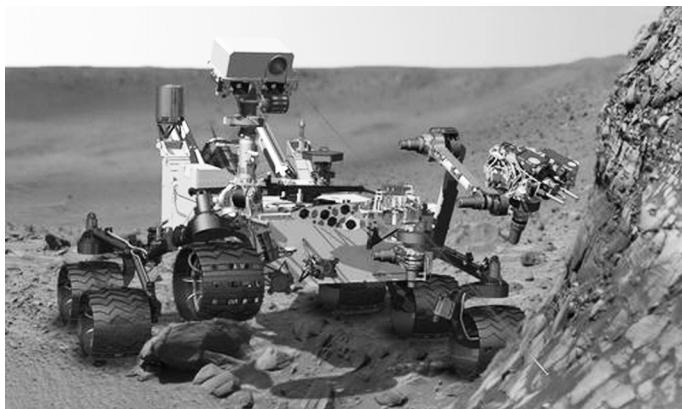


[Courtesy: <http://www.rec.rl.cmu.edu/projects/ahs/>]

Fig. 1.11 An autonomous truck in a mine

3. Space Explorations of the planets, moons, and other near bodies in space are a clear goal for international space scientists. Figure 1.2 shows the Mars Pathfinder of NASA (National Aeronautics and Space Agency) in the United States of America, a space robot. A more recent Mars rover by NASA is Curiosity, as shown in

Fig. 1.12, whereas ‘Chandrayan’ and ‘Mangalyan’ are attempts by the ISRO (Indian Space Research Organization) towards space robotics to explore the moon and Mars, respectively. A robotic approach to explorations has the benefit of achieving many tasks without endangering human lives. Space robots can perform space manipulation (servicing equipment in space), surface mobility (planetary exploration), and scientific experiments with the environment (drilling rocks and testing the composition), etc.



[Courtesy: <http://mars.jpl.nasa.gov/msl/mission/rover/>]

Fig. 1.12 NASA’s Curiosity

4. Underwater Underwater applications of robots involve prospecting for minerals on the floor of the ocean (e.g., nodules of manganese), salvaging of sunken vessels, and repair of ships either at sea or in the dry dock. Figure 1.13 shows a Remotely Operated Vehicle (ROV) by CMERI (Central Mechanical Engineering Research Institute) in Durgapur, India. The ROV is actually an underwater robot planned to perform the following tasks:

- Routine inspection of underwater structures using a camera



[Courtesy: <http://www.cmeri.res.in/rnd/rov.html>]

Fig. 1.13 Remotely operated underwater robot by CEMRI, Durgapur

- Pipeline inspection, debris cleaning, water testing for removal of marine growth, etc.
- Mapping and photo-documentation in marine geological survey, marine life survey, etc.

5. Defence The defence people, namely, the air force, navy, and army are interested in mobile firefighters. These devices would be equipped with infrared sensors and could react more quickly than people in an emergency and in extremely hazardous situations. They are also used for bomb disposals, as shown in Fig. 1.14. It was developed by India's DRDO (Defence Research and Development Organization). Other defence applications of robots will be on the battlefield itself. More realistic short-term applications from the defence point of view would be in the areas of surveillance (e.g., guard and sentry duty), mine sweeping, and artillery-loading devices.

6. Security The application of security robots is mainly for surveillance and guard power generating plants, oil refineries, and other large civilian facilities that are potential targets of terrorist groups. Figure 1.15 shows one such robot which is DRDO's NETRA, an Unmanned Aerial Vehicle, or UAV, robot. The robots for these applications must be mobile (running on wheels and legs as shown in Figs. 1.5 and 1.6, respectively, or tracks or even in air as shown in Fig. 1.15), equipped with some form of vision system and other types of sensors (e.g., infrared), and even have defensive and/or offensive capability.

7. Domestic Though robots were first used for industrial purposes, recently, they have also gained popularity for domestic usage. Such robots can be used to perform day-to-day tasks at home. Some examples are cleaning robots, robots to assist aged people in performing daily chores, vacuum-cleaner robots, etc. A robotic vacuum cleaner by LG Electronics is shown in Fig. 1.16.



[Courtesy: http://en.wikipedia.org/wiki/DRDO_Daksh]

Fig. 1.14 Defence robot 'Daksh' by DRDO



[Courtesy: http://en.wikipedia.org/wiki/DRDO_Netra]

Fig. 1.15 DRDO's unmanned aerial vehicle, NETRA



[Courtesy: <http://www.lg.com/in/vacuum-cleaners/lg-VR6170LVM>]

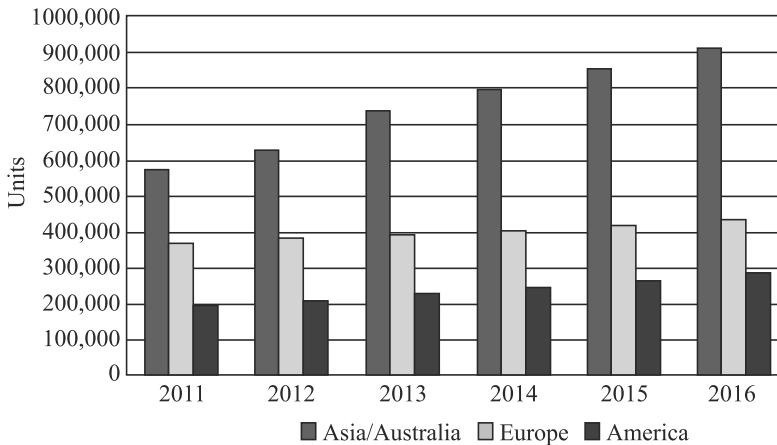
Fig. 1.16 Robotic vacuum cleaner HOM BOT by LG Electronics

8. Entertainment Robots are also finding applications to entertain people, be it in amusement parks, joy rides, sports, etc. Motion platforms are used to seat people wearing 3D glasses and are taken on simulator rides to entertain them. Recently, an industrial robot has been used in an amusement ride named KUKA Robocoaster, as shown in Fig. 1.17. Several rides in Disneyland and other popular amusement parks are also driven by robots these days. Robotics competitions, such as ROBOCON, can also be seen as an entertainment where students of engineering institutes develop robots that compete against each other by performing the laid-down tasks. In addition, there are some popular robots such as Honda's Asimo, Sony's Aibo, etc., which are also used for entertainment.



[Courtesy: <http://www.kuka-entertainment.com/en/products/robocoaster>]

Fig. 1.17 KUKA Robocoaster used as an amusement ride

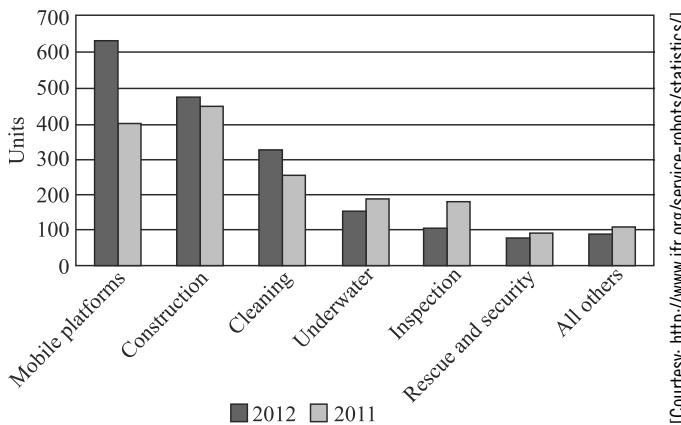


[Courtesy: <http://www.ifr.org/industrial-robots/statistics/>]

Fig. 1.18 Estimated populations of industrial robots

1.3.2 Population

Figure 1.18 shows the estimated population of industrial robots in different continents, whereas Table 1.1 shows the country-wise distribution of robots and future predictions. The statistics of the number of service robots, used for applications other than those in industries, are shown in Fig. 1.19.



[Courtesy: <http://www.ifr.org/service-robots/statistics/>]

Fig. 1.19 Number of service robots sold in 2011 and 2012

1.3.3 Economics

Industrial robots are becoming cheaper and cheaper, and the industrial robot prices reflect this. Consider that a Swiss-made ABB IRB 6000 that is priced at ₹4,00,000 [₹ is a symbol for India's currency, (Rupee)], which in US currency would be equivalent to about US\$ 7,200 (as per the exchange rate of

Are robots threat to employment?

Statistics show that largest robot users in the world are not those having highest unemployment. Hence, it can be viewed that the robots are not a direct threat to employment.

One US\$ = ₹55) as on 20 May 2013. The same robot was over ₹72,00,000 when it was new in 1993. A new equivalent robot will cost between ₹20,00,000 (US\$ 36,000) to ₹55,00,000 (US\$ 100,000) depending on the manufacturer and specifications. For example, a new German-made KUKA KR-5 robot with 6-DOF and 5 kg payload was procured at IIT Delhi in 2011 for its Programme for Autonomous Robotics. The robot is shown in Fig. 2.1 whose price was ₹22,00,000 (about US\$ 40,000). The prices of robots are becoming so cheap that for some countries, a new robot is even cheaper than the cheapest labourer. However, the success of any industry is measured in terms of financial performance. Hence, the best technical innovation can be a failure if it results in no return. Robots are no exception to this rule.

Table 1.1 Estimated stock of industrial robots during 2011-12 and forecast for 2013 and 2016

Country	2011	2012	2013*	2016*
America	192,966	207,017	226,550	281,000
Brazil	6,971	7,576	9,170	17,400
North America (Canada, Mexico, USA)	184,679	197,962	215,650	260,800
Other America	1,316	1,479	1,730	2,800
Asia/Australia	576,545	628,889	733,500	908,500
China	74,317	96,924	121,200	215,800
India	6,352	7,840	9,300	16,300
Japan	307,201	310,508	309,400	312,900
Republic of Korea	124,190	138,883	155,300	201,700
Taiwan	29,837	32,455	35,800	43,000
Thailand	13,088	17,116	20,600	32,600
Other Asia/Australia	21,560	25,163	81,900	86,200
Europe	369,965	380,546	388,800	431,700
Czech Rep.	5,890	6,830	7,800	11,000
France	34,461	33,624	33,000	33,200
Germany	157,241	161,988	165,800	177,900
Italy	62,245	60,750	58,600	55,400
Spain	29,847	28,911	27,300	27,100
United Kingdom	13,641	15,046	15,500	20,000
Other Europe	66,640	73,397	80,800	107,100
Africa	2,495	2,858	3,300	4,900
Not specified by countries**	11,126	16,079	20,850	33,400
Total	1,153,097	1,235,389	1,373,000	1,659,500

Sources: IFR, national robot associations,

*forecast

**reported and estimated sales which could not be specified by countries

Irrespective of the social benefits, smartness of the technology, the look of the robot, the investment in robotics has to yield financial gains. Same criteria in this direction can provide a framework for the management to decide whether to invest on robotics installation or not.

1. Price of a Robot The purchase price of a robot is highly variable, particularly, if one requires a robot to simply pick and place an object with few articulations. The range might extend from ₹20,00,000–55,00,000 (US\$36,000–100,000) depending upon the number of articulations, operating area, weight-handling capacity, and control sophistication. Generally speaking, the higher priced robots are capable of more demanding jobs and their control sophistication assures that they can be adapted to new jobs when original assignments are completed. So, the more expensive and more sophisticated robot will normally require less special tooling and lower installation cost. Some of the pick-and-place robots are considered as no more than an adjustable component to the automation system. A robot may contribute over 20% of the total system automation cost.

2. Special Tooling Special tooling may include an indexing conveyor, weld guns, transformers, clamps, and a supervisory computer for a complex task of robots involved in spot welding of car bodies. For assembly automation, the special parts may cost well in excess of the robot equipment cost.

3. Installation Cost Installation cost is sometimes charged fully to a robot project, but is often carried as overhead because plant layout may change. As a model changes, there are usually installation costs to be absorbed even if equipment is to be manually operated. There is no logic to penalising the robot installation for any more than a differential cost which is inherent in the robotizing process.

4. Maintenance To keep the functioning of a robot proper, there is a need for regular maintenance, a periodic need for more sweeping overhaul, and a random need to correct unscheduled downtime incidents. A rule of thumb for well-designed production equipment operated in two shifts continually is a total annual cost of about 10% of the purchase price. There is a variability of course depending upon the demands of the job and the environment. Maintenance costs in a foundry are greater than those experienced in plastic moulding.

5. Operating Power Operating power is easily calculated as the product of overall power drain and the hours worked.

6. Finance In some cost-justification formulae, one takes into account the current cost of money. In others, one uses an expected return on investment to establish economic viability.

7. Depreciation Robots, like any other equipment, will exhibit a useful life and it is ordinary practice to depreciate the investment over this useful life. Since a robot tends to be a general-purpose equipment, there is ample evidence that an 8 to 10 years life running multi-shift is a conservative treatment.

8. Enhanced Productivity The prime issue in justifying a robot is increased productivity. Industries are interested in shielding workers from hazardous working

conditions, but the key motivator is the increased productivity by introducing a robot that can operate for more than one shift continuously and thereby multiply the production rate.

9. Improved Quality If a job is in a hazardous environment, or is physically demanding, or is simply mind-numbing, there is a good chance that product quality will suffer according to the mood of a human worker. A robot may well be more consistent on the job and, therefore, it may produce a higher quality output.

1.3.4 Safety

Industrial robots can be dangerous. They are exceptionally powerful devices, especially models with larger capacity and reach. Their misuse may constitute a risk to life and limb or cause damage to the robots and to other material property. This means that safety is paramount during the installation and in production. Safety guidelines vary from country to country and are essential to ensure that any installation complies with local legislation. Mostly, safety is about isolating personnel from the robot's work envelope and ensuring that the movements can be easily halted in an emergency. It should be noted that almost all robots have electrically operated disc brakes on each axis. These are 'on' whenever power is not applied to release them. Therefore, in the event of a power failure or if the emergency stop is applied the robot stops dead, within split second, in position. It is said that robots take over hazardous jobs, but at the same time new risks are involved. Accidents involving robots can happen just as with any other machinery. Some of the possible ways accidents can happen due to electrical noise, oil-pressure valve troubles, encoder-related problems, electronic malfunctions, and mistakes by human workers are as follows:

- The arm of a robot suddenly moves as the oil pressure is cut off after the robot completes its work, or there is a power failure.
- A robot executes a motion that was not a part of its program.
- A robot starts moving as soon as its power source was switched on, although its interlock conditions were still not ready.
- When operating alone, a robot destroyed the work it was to weld because of a mistake in program instruction.
- High summer temperature leading to erratic behavior.

Note that for programming, monitoring, tool changing, inspection, correct problems with peripheral equipment, accidental halt, inspect the work, investigate the trouble, and other operations, humans must enter the workspace of robots. Hence, the issue of robot safety is of paramount importance. It mainly deals with three aspects, namely, the design of a reliable control system to prevent malfunctions, the design of the workstation layout, and training of plant personnel (programmers, operators, and maintenance staff). While the first aspect depends on the robot manufacturer, the other two must be taken care of in the plant. The following guidelines can help

Safety First

It is the phrase we see almost in every factory. But do we sincerely follow it?

remove hazardous situations to robot personnel, factory workers, visitors, and to the robot itself:

- The robot working area should be closed by permanent barriers (e.g., fences, rolls, and chains) to prevent people from entering the area while the robot is working. The advantage of a fence-type barrier is that it is also capable of stopping a part which might be released by the robot's gripper while in motion.
- Access gates to the closed working area of the robot should be interlocked with the robot control. Once such a gate is opened, it automatically shuts down the robot system.
- An illuminated working sign, stating 'robot at work', should be automatically turned on when the robot is switched on.
- Emergency stop buttons must be provided in easily accessible locations as well as on the robot's teach box and control console. Hitting the emergency button stops power to the motors and causes the brakes on each joint to be applied.
- Pressure-sensitive pads can be put on the floor around the robot that, when stepped on, turn the robot controller off.
- Emphasize safety practices during robot maintenance. In addition, the arm can be blocked up on a specially built holding device before any service work is started.
- Great care must be taken during programming with the manual reaching mode. The reach box must be designed so that the robot can move as long as a switch is pressed by the operator's finger. Removing the finger must cease all robot motions. The motion during the teaching mode is restricted to slow speeds in some robots. Programmers should stay out of areas where they might be hurt in the event of a robot malfunction.
- The robot's electrical and hydraulic installation should meet proper standards. This includes efficient grounding of the robot body. Electric cables must be located where they cannot be damaged by the movements of the robot. This is especially important when the robot carries electrical tools such as a spot-welding gun.
- Power cables and signal wires must not create hazards if they are accidentally cut during the operation of the robot.
- If a robot works in cooperation with an operator, for example, when a robot forwards parts to a human assembler, the robot must be programmed to extend its arm to the maximum when forwarding the parts so that the worker can stand beyond the reach of the arm.
- Mechanical stoppers, interlocks, and sensors can be added to limit the robot's reach envelope when the maximum range is not required. If a wall or a piece of machinery not served by the robot is located inside the reach envelope, the robot can be prevented from entering into this area by adding photoelectric devices, stoppers, or interlock switches in the appropriate spots.

Another approach states that only robots themselves are able to detect the approach of humans. Therefore, the solution to the safety problem is to provide a sensor system that can detect intruders that enter the robot area while it is operating.

SUMMARY

Robots, their history, definitions, categories, special applications like in space, defence, etc., and their populations are presented in this chapter. The laws of robotics and when to use robots are also presented. Criteria for economic viabilities of robotic installation were outlined, along with the safety aspects.

EXERCISES

- 1.1** What is a robot?
- 1.2** What are the types of robots?
- 1.3** Give a typical example of flying robot.
- 1.4** What is the purpose of the ‘Daksh’ robot?
- 1.5** Where was ‘Curiosity’ used?
- 1.6** Name an entertainment robot.
- 1.7** Name a few typical applications of an industrial robot.
- 1.8** What are the differences between a robot and a CNC machine tool?
- 1.9** How to decide the introduction of a robot for a particular job?
- 1.10** What are the four D’s of robotics?
- 1.11** What is RUR?
- 1.12** What are the ‘Laws of Robotics?’
- 1.13** Write down different applications of industrial robots?
- 1.14** What are the criteria for robotic installation from the economic point of view?
- 1.15** What are the safety issues in robot usage?

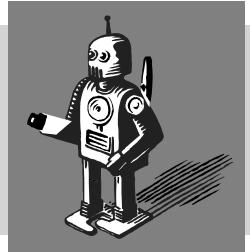
WEB-BASED EXERCISES

Based on Web search, find the answers to the following questions:

- 1.16** Find names of different robot manufacturers.
- 1.17** Find out the price of an industrial robot (other than KUKA KR-5).
- 1.18** What is the robot population in the world?
- 1.19** Name some robot applications that are not listed in this chapter?
- 1.20** What are different industrial robot models of a company (other than KUKA)?

2

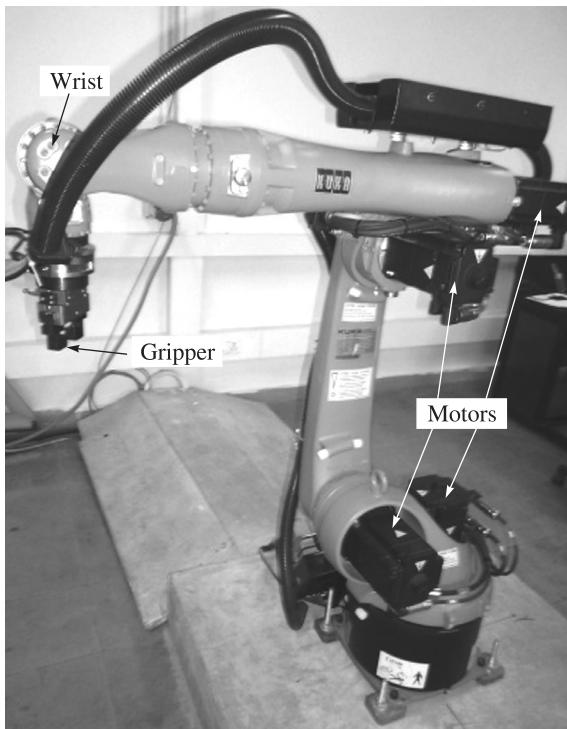
Industrial Robots and Their Applications



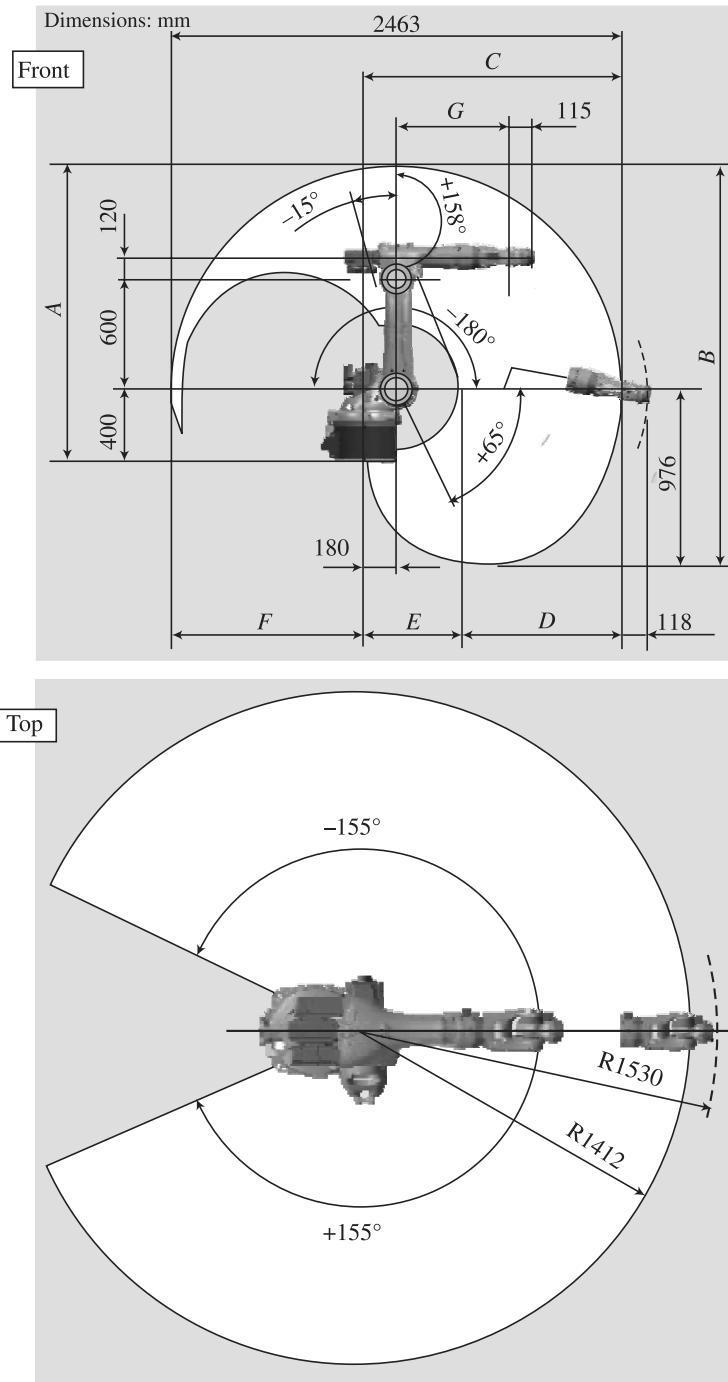
An industrial robot is typically serial in nature, as shown in Fig. 1.1 or 1.3 or 2.1 (a). As a result, such robots are also referred to as serial robots. A robot, either serial or otherwise, consists of several subsystems, e.g., motion subsystem, recognition subsystem, etc. These are explained in Section 2.1. Specifications of one such serial robot, namely, the German-made KUKA KR-5, housed in the Programme for Autonomous Robotics (PAR) Laboratory of IIT Delhi in 2010 are given in Fig. 2.1 (b) and Table 2.1. From the company's literature, it is clear that the robot is

Work Envelope or Workspace?

They mean the same, i.e., the reachable points by the end-effector of a robot.



(a) KUKA KR-5 Arc robot in the PAR Lab. of IIT Delhi



(b) Front and top views of work volume

Fig. 2.1 An industrial KUKA KR-5 robot [Courtesy: Company's Technical Data sheet]

to be used for arc-welding purposes, as emphasized by the word ‘Arc’ in its type ‘KR-5 Arc’, whereas 5 stands for the payload capacity, i.e., 5 kg. One can classify this robot as *arc-welding* robot. Note that the operating volume of the arm is shown in Fig. 2.1 (b) and Table 2.1 (top two rows). They show how far and where the arm can move, and defines a geometric shape. Another way to specify a robot is based on its geometric work envelope or arm configuration or coordinate system. There are other types of classifications depending on different aspects of a robot, which are explained in Section 2.2.

Table 2.1 Technical data of KUKA KR-5 Arc [Courtesy: Company’s Technical Data sheet]

Work Envelope	A (mm)	B (mm)	C (mm)	D (mm)	E (mm)	F (mm)	G (mm)	Volume
KR-5 Arc	1,632	2,207	1,412	881	531	1,052	620	8.4 m ³
Type		KR-5 arc						
Maximum reach		1,411 mm						
Rated payload		5 kg						
Suppl. load, arm/link arm/rotating col.		12/-/20 kg						
Suppl. load, arm + link arm, max.		-						
Maximum total load		37 kg						
Number of axes		6						
Mounting position		Floor, ceiling						
Variant		-						
Positional repeatability (ISO 9283)		±0.04 mm						
Path repeatability (ISO 9283)								
Controller		KR C2 edition 2005						
Weight (excluding controller), approx.		127 kg						
Temperature during operation		+10 °C to +55 °C						
Protection classification		IP 54, IP 65 (In-line wrist)						
Robot footprint		324 mm × 324 mm						
Connection		7.3 kVA						
Noise level		< 75 dB						
Axis data	Range (software)			Speed with rated 5 kg payload				
Axis 1 (A1)	±155°			154°/s				
Axis 2 (A2)	+65°/-180°			154°/s				
Axis 3 (A3)	+158°/-15°			228°/s				
Axis 4 (A4)	±350°			343°/s				
Axis 5 (A5)	±130°			384°/s				
Axis 6 (A6)	±350°			721°/s				

2.1 ROBOT SUBSYSTEMS

As illustrated in Fig. 2.2, a robotic system generally consists of three subsystems, namely, a motion subsystem, a recognition subsystem, and a control subsystem. While some aspects of hardware implementations are explained in Chapter 13, their functions are described below.

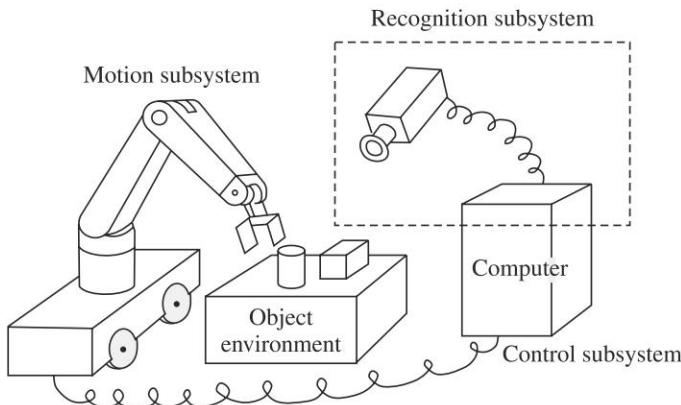


Fig. 2.2 Robot subsystems and their interactions
[Courtesy: Yoshikawa (1990)]

1. A Motion Subsystem The motion subsystem is the physical structure of the robot that carries out desired motion similar to human arms, as illustrated in Fig. 2.3.

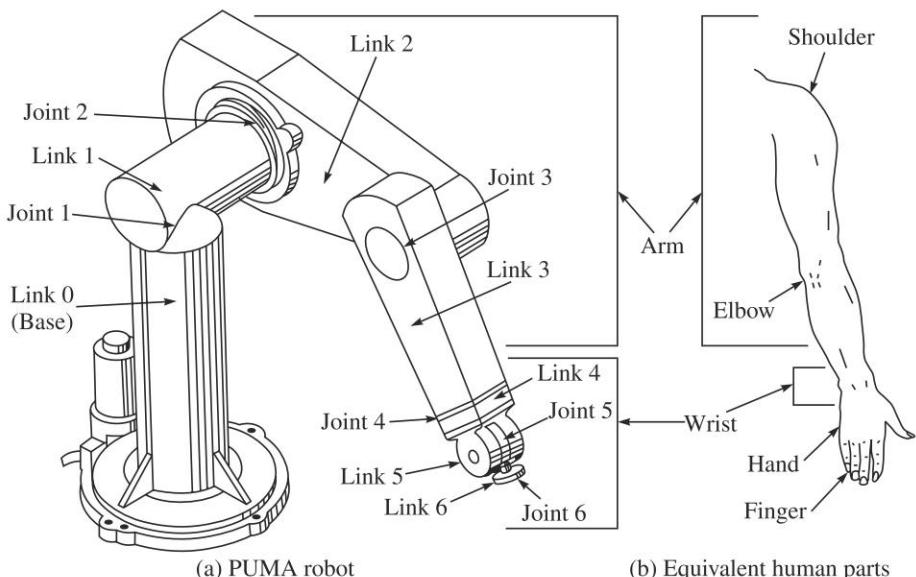


Fig. 2.3 A robot manipulator and its equivalent human parts

2. A Recognition Subsystem The recognition subsystem uses various sensors to gather information about the robot itself and any object being acted upon, and about the environment. Based on sensor data, it recognizes the robot's state, the objects, and the environment.

3. A Control Subsystem The control subsystem influences the robot's motion to achieve a given task using the information provided by the recognition subsystem.

What is PUMA?

PUMA stands for *Programmable Universal Manipulator for Assembly*.

It may be useful to point out here that a person with mechanical engineering background normally works on motion subsystem area, whereas the people with computer science and electrical engineering knowledge focus on the recognition and control subsystems, respectively. This is due to the subjects taught in their academic curricula. However, robotics is an interdisciplinary area and a comprehensive knowledge of all three will certainly help to design and develop a better robotic system. As a result, it is not uncommon to see people crossing their boundaries of specialization. It is often seen that a mechanical engineering specialist works on Artificial Intelligence (Recognition Subsystem), while the one with electrical engineering or computer science background deals with dynamic simulation and design of robots (motion subsystem).

Why Robotics is interdisciplinary?

Since it requires the knowledge of Mechanical Engineering, Electrical Engineering, Computer Science Engineering, and Information Technology.

2.1.1 Motion Subsystem

The elements of the motion subsystem are as follows:

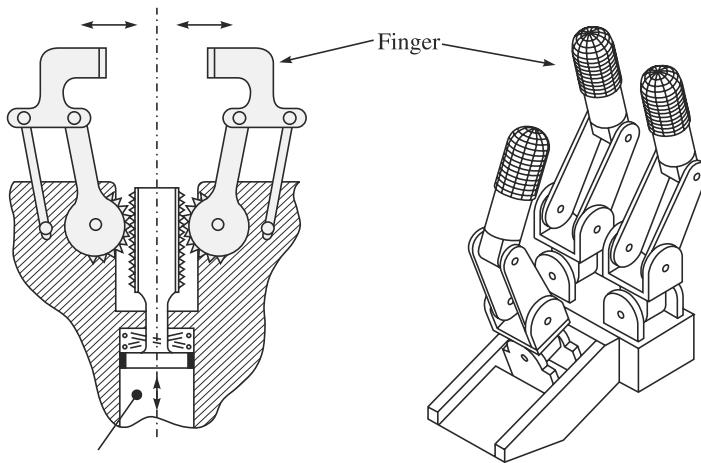
1. Manipulator This is the physical structure, which moves around. It comprises of *links* (also referred as *bodies*) and *joints* (also called *kinematic pairs*) normally connected in series, as shown in Figs. 1.1, 1.3 and 2.1 or the PUMA robot as shown in Fig. 2.3(a). Each link is either made of steel or aluminum. Other materials can also be used depending on the requirements. The joints are generally *rotary* or *translatory* types. In the study of robotics and mechanisms, these joints are referred to as *revolute* and *prismatic* joints, as explained in Chapter 5. Whereas an example of a revolute joint is the hinge of a door, a prismatic joint is the piston-cylinder arrangement of an Internal Combustion (IC) engine used in automobiles.

Robot or Manipulator?

In this book, many times both the words are used interchangeably. However, as explained in this chapter, *manipulator* is the mechanical structure of the robot.

Like a human arm, wrist, and hand arrangement of Fig. 2.3(b), a robot manipulator has also three parts. The first two, i.e., the arm and wrist, are shown in Fig. 2.3(a), respectively, whereas the third one, i.e., the hand, is shown in Fig. 2.4(a). More gripper or end-effectors performing the task of a hand are explained in Chapter 3. The function of an arm is to place an object in a certain position in the three-dimensional Cartesian space, where the wrist orients it. For a typical six degrees-of-freedom

(DOF) robot, as shown in Fig. 1.1, 1.3(a), 2.1 and 2.3(a), the first three links and joints form the *arm*, and the last three mutually intersecting joints make the *wrist*.



(a) A simple gripper [Mair (1988)] (b) A three-fingered hand [Angeles (2003)]

Fig. 2.4 Robot hands

2. End-effector This is the part attached at the end of a robot manipulator. Hence, the name follows. This is equivalent to the human hand. An end-effector could be a mechanical hand that manipulates an object or holds it before they are moved by the robot arm. Two typical hands are shown in Figs. 2.4(a-b). Figure 2.4(a) shows a simple two-fingered gripper that holds simple objects, whereas a multi-fingered hand shown in Fig. 2.4(b) can perform complex tasks. More grippers of the type shown in Fig. 2.4(a) are explained in Chapter 3. Also, the specialized tools like welding electrode, gas-cutting torch, painting brush, deburring tool, or grinding wheel attached to the end of a manipulator arm to perform specific tasks, are also considered end-effectors.

End-effector or Hand?

Mechanical hands (Fig. 2.4) are basically the end-effectors of a robot. End-effector is a more generic term used to specify what is attached at the end of a robot manipulator. Other examples of end-effectors are welding gun, paintbrush, etc.

3. Actuator The actuators of a robot actually provide motion to the manipulator links and the end-effector. They are classified as pneumatic, hydraulic, or electric, based on their principle of operation, which are explained in Chapter 3. Note here that an electric motor, e.g., dc or ac, when coupled to motion transmission elements, e.g., gears, etc., is called an actuator. However, a pneumatic or a hydraulic system which can directly impart motions to the robot links and the end-effectors is called an actuator, not motor.

4. Transmission As the term conveys, these elements transmit motion from the electric motors and pneumatic/hydraulic actuators to the actual links of the manipulator. With electric motors these elements, mainly, the gears, are used to step down the speed. Note that electric motors are efficient at higher speeds. However,

the robot links move at relatively slow speed. Hence, the transmission gears are used to reduce the speed of the electric motors. Typical transmission elements are the following:

Belt and Chain Drives Belt drives are widely used in robotics, particularly, the synchronous belt shown in Fig. 2.5(a). However, their life is short as they rely on belt tension to produce grip over the pulley. Alternatively, chains shown in Fig. 2.5(b) are generally cheaper. They have higher load capacities and service lives compared to belt drives, but lower in relation to the gears.

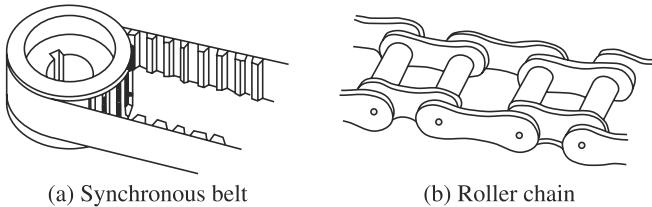


Fig. 2.5 Belt and chain drives

Gears Of all mechanical transmissions, gears shown in Fig. 2.6 are the most long-lasting and reliable, although their backlash must be carefully taken into account during the design stage.

What is a mechanism?

A series of links, generally rigid, coupled by joints that allow relative motions between any two links form a mechanism.

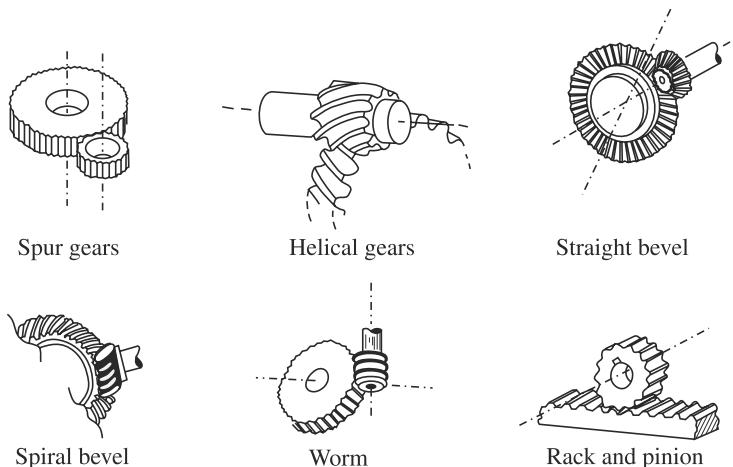


Fig. 2.6 Gears

Link Mechanisms In order to reduce the flexibility and weight of the above transmission elements, link mechanisms shown in Fig. 2.7(a) are used.

Screw Mechanism Figure 2.7(b) shows how ball-screws with a four-bar mechanism (indicated with 1, 2, 3 and 4) bars are used to transmit motion.

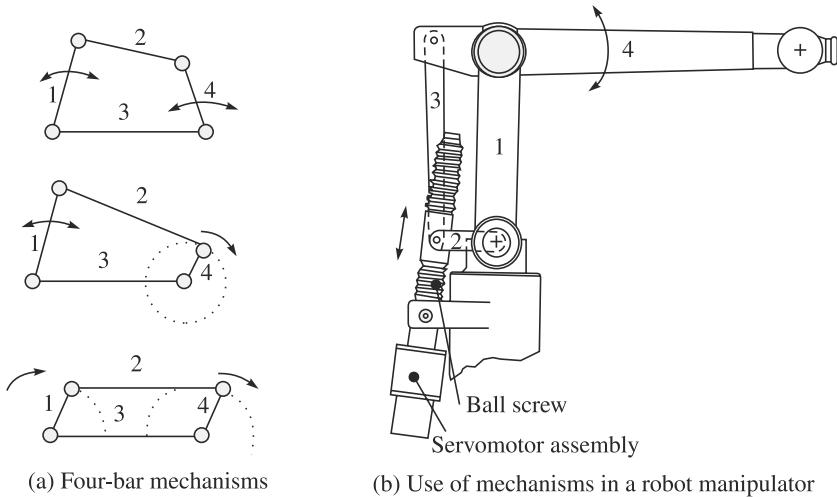


Fig. 2.7 Mechanisms and their use in a robot manipulator

Example 2.1 Gear Ratio

Assume a pair of spur gears shown in Fig. 2.6 has 20 and 100 teeth, respectively. If the smaller gear (pinion) rotates at 200 rpm (revolutions per minute), the speed of the larger gear or gear, denoted with ω_g , can be obtained as

$$\omega_g = (1/\eta)\omega_p = (20/100)200 = 40 \text{ rpm} \quad (2.1)$$

Pinion vs. Gear?

In a pair of two gears, the smaller gear is called 'pinion,' whereas the larger one is simply referred as only 'gear.'

where η is the gear ratio defined as the ratio between the number of teeth of pinion and gear, respectively, i.e., $\eta = n_g/n_p - n_p$ and n_g being the number of teeth in pinion and gear, respectively, whereas ω_p is the speed of the pinion. Such gear transmission is called step-down gear arrangement and specified as 1:5.

Example 2.2 Screw Transmission

Figure 2.7(b) shows a screw mechanism for the transmission of motion from the motor to link 4 via link 3. If the nut on Link 3 has to be translated by 50 mm while moving on the screw of 5 mm pitch, the screw should be turned by θ times. The value of θ can be obtained from the following formula:

$$x = l\theta, \text{ where } l = \alpha p \quad (2.2)$$

In Eq. (2.2), x is the displacement of the nut for θ rotations of the screw, and l is the lead of the screw which is equal to pitch p multiplied by the number of starts α . Assuming, $\alpha = 1$, i.e., single start screw, the number of times the screw needs to rotate can be easily obtained from Eq. (2.2) as $\theta = 10$.

2.1.2 Recognition Subsystem

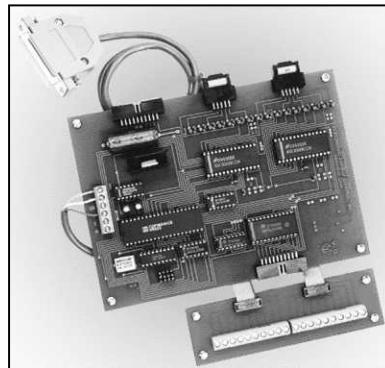
The most important element in the recognition subsystem is the sensor, which is like our eyes or nose. Inclusion of sensors to a robot changes its dumb nature to

an intelligent one. In order to process the sensed signal, which is mostly analog, by a digital controller, an Analog-to-Digital Converter (ADC) is required. Some of the aspects of signal conditioning will be elaborated in Chapter 4. A recognition subsystem typically consists of the following elements.

1. Sensors Most of the sensors are essentially transducers. Transducers convert one form of signal to another. For example, the human eye converts light patterns into electrical signals. Sensors fall into one of the several general areas: vision, touch, range and proximity detection, navigation, speech recognition, etc. Each of these areas is an individual research area in itself. However, some of them are explained in relative details in Chapter 4.

2. Analog-to-Digital Converter (ADC)

(ADC) This electronic device interfaces the sensors with the robot's controller. For example, the ADC converts the voltage signal due to the strain in a strain gauge to a digital signal, i.e., 0 or 1, so that the digital robot controller can process these information. They physically look like any other computer interface card inside the Central Processing Unit (CPU) box, as depicted in Fig. 2.8.



[Courtesy: <http://www.eec1.com/adc-16p.htm>]

Fig. 2.8 An analog-to-digital converter

2.1.3 Control Subsystem

The role of control in a robot is depicted in Fig. 2.9(a). It primarily consists of the following items.

1. Digital Controller The digital controller is a special electronic device that has a CPU, memory, and sometimes hard disk to store programmed data. In robotic systems, these components are kept inside a box referred as controller, as shown in Fig. 2.9(b). Figure 2.9(b) is the controller KR C2 for KUKA robots whose inside is shown in Fig. 2.9(c). It is used to control the movement of the manipulator and end-effector. A robot controller is like the supervisor in a factory. Since a computer has the same characteristics as those of a digital controller, it can be used as a robot controller. A controller processes the user-programmed commands and sends signals to the actuators through the Digital-to-Analog Converters (DAC). The programming languages can be same as computers, i.e., BASIC, Fortran, C, and C++. However, commercial robots use their domain-specific languages based on their manufacturers. For example, KUKA, Germany, uses Kuka Robot Language (KRL), whereas Fanuc, Japan, uses *Karel* robot programming language. This is mainly to introduce specific features into the robotic systems so that the products are different.

2. Digital-to-Analog Converter (DAC)

A Digital-to-Analog Converter, or DAC, converts the digital signal from the robot controller to an analog signal to activate the actuators. In

ADC vs. DAC

A DAC serves the purpose opposite to an ADC.

order to actually drive the actuators, e.g., a dc electric motor, the digital controller is coupled with a DAC to convert its signal back to an equivalent analogue signal, e.g., the electric voltage for the dc motor. Physical appearance of a DAC is shown in Fig. 2.10.

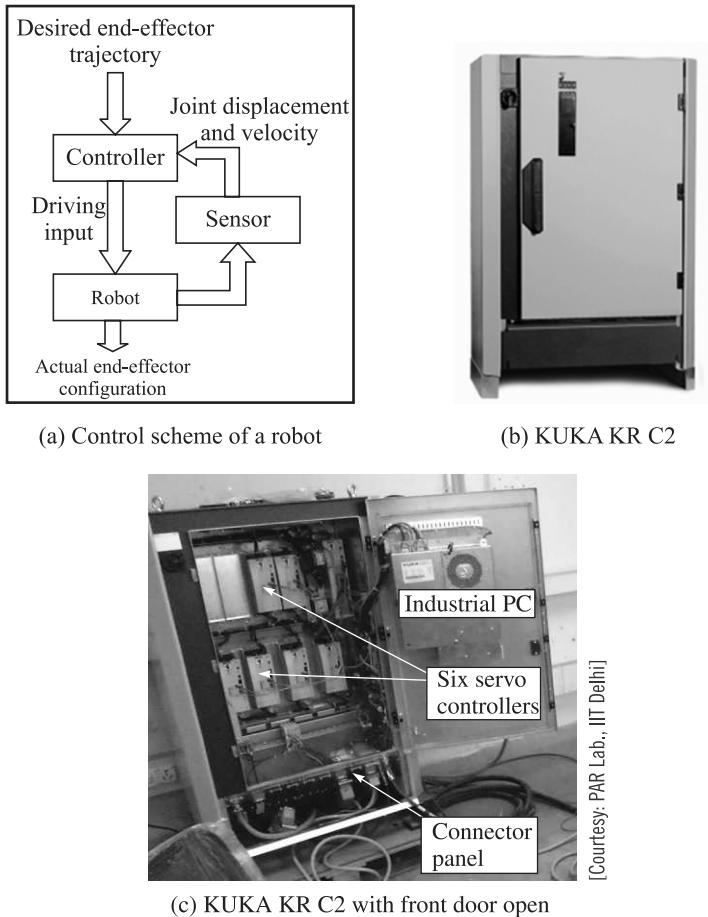


Fig. 2.9 Control subsystem

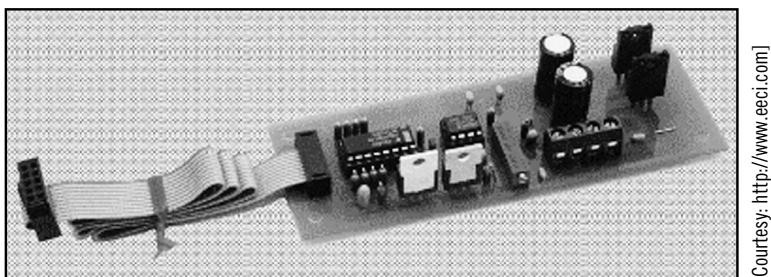


Fig. 2.10 A digital-to-analog converter

3. Amplifier Since the control commands from the digital controller converted to analog signals by the DAC are very weak, they need to be amplified to drive, e.g., the electric motors of the robot manipulator.

2.2 CLASSIFICATION OF ROBOTS

As mentioned in the introduction of this chapter, there are several ways to classify a robot, for example, based on a coordinate system, i.e., Cartesian or cylindrical, etc., or control of a robot or based on its applications, i.e., assembling or welding, etc. Whereas classifications of robots based on the coordinate system, actuators, control scheme, and programming methodology are explained in this section, those based on industrial applications are separately covered in Section 2.3.

2.2.1 Coordinate Systems

Classification of a robot based on its coordinate system is also referred to as classification by arm configuration or geometric work envelope. It actually classifies the arm of a robot without considering the wrist and its end-effector or hand. It tells the volume of reachable coordinates of a point on the end-effector, rather than its orientations. There are four such fundamental types, namely, Cartesian, cylindrical, spherical or polar, and articulated or revolute. They are explained next.

1. Cartesian When the arm of a robot moves in a rectilinear mode, that is, to the directions of x , y , and z coordinates of the rectangular right-handed Cartesian coordinate system, as shown in Fig. 2.11(a), it is called Cartesian or rectangular type. The associated robot is then called Cartesian robot. The movements are referred to as travel x , height or elevation y , and reach z of the arm. Its workspace has the shape of a rectangular box or prism, as indicated in Fig. 2.11(b). A Cartesian robot needs a large volume to operate. It has, however, a rigid structure and provides an accurate position of the end-effector. Maintenance of such robots is difficult, as the rectilinear motions are generally obtained through the sets of rotary electric actuators coupled with nut-and-ball screws. Dust accumulated on the screws may jam the smooth motion of the robot. Hence, they have to be covered with bellows. Moreover, maintaining the straightness of the screw demands higher rigidity in those components. Hence, such robots tend to be more expensive.

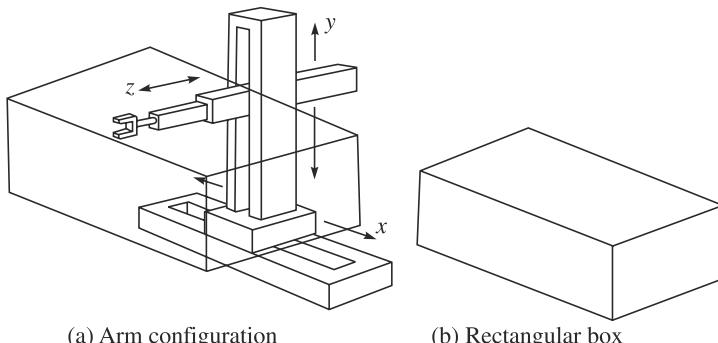


Fig. 2.11 A Cartesian robot arm with its workspace

2. Cylindrical When the arm of a robot possesses one revolute and two prismatic joints, i.e., the first prismatic joint of the Cartesian type, Fig. 2.11(a), is replaced by a revolute one with its axis rotated by 90° about the reach z -axis, the points that it can reach conveniently be specified by the cylindrical coordinates, i.e., angle θ , height y , and radius z , as in Fig. 2.12(a). A robot with this type of arm is termed as cylindrical robot whose arm moves, θ , y , and z , i.e., it

has base rotation, elevation, and reach, respectively. Since the coordinates of the arm can assume values between specified upper and lower limits, its end-effector can move in a limited volume that is a cut section from the space between the two concentric cylinders, as shown in Fig. 2.12(b). Note that for a Cartesian arm, this is not the case. The workspace is a solid box given by Fig. 2.11(b). The dotted line in Fig. 2.12(b) completes the boundary of the workspace volume for better visualization. It has no other purpose. A robot of this type may have difficulties in touching the floor near the base. Cylindrical robots are successfully used when a task requires reaching into small openings or working on cylindrical surfaces, e.g., welding pipes.

3. Spherical or Polar When the arm of a robot can change its configuration by moving its two revolute joints and one prismatic joint, i.e., the second prismatic joint along the height y of the cylindrical type is replaced by a revolute joint with its axis rotated by 90° about the reach z -axis, the arm position is conveniently described by means of the spherical coordinates, i.e., θ , ϕ , and z . The arm is shown in Fig. 2.13(a), and is termed a spherical or polar robot arm. The arm movements represent the base rotation, elevation angles, and reach, respectively. Its workspace is indicated in Fig. 2.13(b).

Why different coordinate systems to define robot architectures?

No special reason except the convenience of defining a point in the three-dimensional Cartesian space.

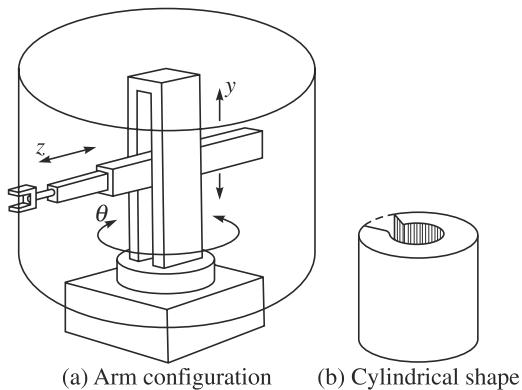


Fig. 2.12 A cylindrical robot arm with its workspace

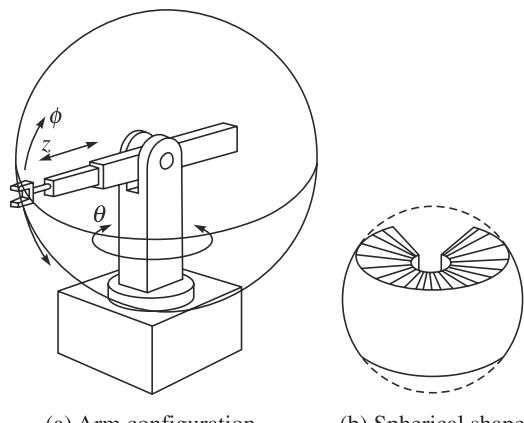


Fig. 2.13 A spherical robot arm with its workspace

4. Articulated or Revolute When a robot arm consists of links connected by revolute joints only, i.e., the third prismatic joint of the spherical type is also replaced by another revolute joint with its axis rotated by 90° about θ -axis, it is called articulated or revolute jointed arm. It is shown in Fig. 2.14(a). Its spherelike workspace is shown in Fig. 2.14(b) whose internal surface is difficult to determine. Such robots are relatively simple to fabricate and maintain, as the robot's actuators are directly coupled through a set of rotary gear or belt elements. However, achieving a task of the Cartesian coordinates requires mathematical transformations. These aspects are discussed in Chapter 6.

Why some workspaces, e.g., Fig. 2.12(b), are hollow?

Hollow portions are those where the end-effector of the robot cannot reach. They are called singular zones (see Chapter 6 for the definition).

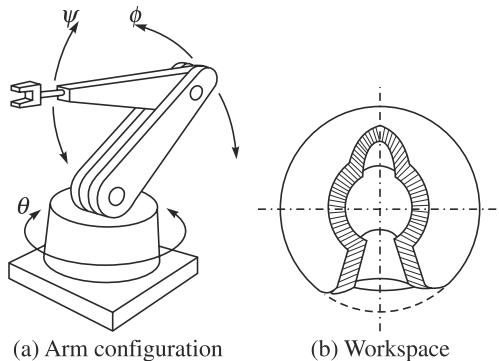


Fig. 2.14 An articulated robot arm with its workspace

Table 2.2 Transformation of robot-arm types

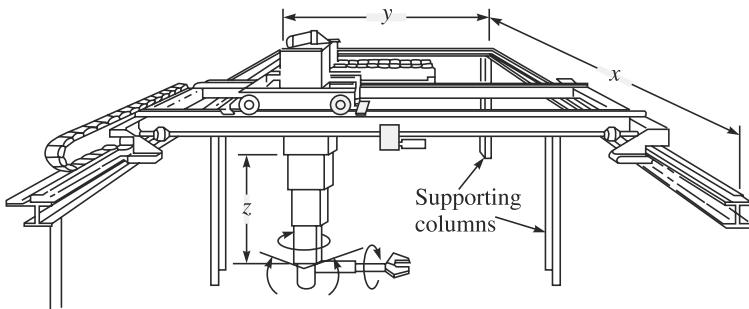
Type	Joints		
	1 (base): Motion	2 (elevation): Motion	3 (reach): Motion
Cartesian ↓	P : travel x $\downarrow -P+R+90^\circ @z$	P : height y \downarrow	P : reach z \downarrow
Cylindrical ↓	R : rotation θ \downarrow	P : -do- $\downarrow -P+R+90^\circ @z$	P : -do- \downarrow
Spherical ↓	R : -do- \downarrow	R : rotation ϕ \downarrow	P : -do- $\downarrow -P+R+90^\circ @\theta\text{-axis}$
Revolute	R : -do-	R : -do-	R : rotation ψ

-P: Remove prismatic joint; +R: Add revolute joint; $+90^\circ @z$: Rotate the revolute joint axis about z -axis by 90° .

It is interesting to note here that how the above four fundamental arm architectures can be derived from one another. This is explained in Table 2.2, whereas Table 2.3 provides the advantages and disadvantages of those basic robot arms. Some literatures also classify robots as Gantry and SCARA (Selective Compliance Assembly Robot Arm), as shown in Figs. 2.15 and 2.16, respectively, and others. This is truly not required, as the fundamental types will help one to understand such types. For example, the arm in the gantry robot is a Cartesian type whose base is mounted overhead, i.e., the robot is placed upside down. This robot is large, versatile in its operation, and expensive. The SCARA, on the other hand, is a cylindrical type whose reach is obtained using a revolute, instead of a prismatic joint [compare Figs. 2.12(a) and 2.16]. A SCARA robot is very suitable for assembly operations, and is, therefore, extensively used in several industries.

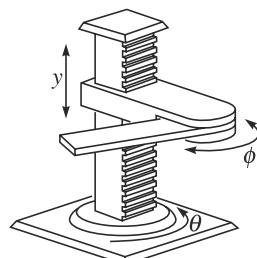
Table 2.3 Comparison of fundamental robot arms [Courtesy: Fuller (1999)]

Configuration	Advantages	Disadvantages
<i>Cartesian</i> (3 linear axes) x : base travel y : height z : reach	- Easy to visualize - Rigid structure - Easy offline programming - Easy mechanical stops	- Reach only front and back - Requires large floor space - Axes are hard to seal - Expensive
<i>Cylindrical</i> (1 rotation and 2 linear axes) θ : base rotation y : height z : reach	- Can reach all around - Rigid y , z -axes - θ -axis easy to seal	- Cannot reach above itself - Less rigid θ -axis - y , z -axes hard to seal - Won't reach around obstacles - Horizontal motion is circular
<i>Spherical</i> (2 rotating and 1 linear axes) θ : base rotation ϕ : elevation angle z : reach	- Can reach all around - Can reach above or below obstacles - Large work volume	- Cannot reach above itself - Short vertical reach
<i>Articulated</i> (3 rotating axes) θ : base rotation ϕ : elevation angle ψ : reach angle	- Can reach above or below objects - Largest work volume for least floor space	- Difficult to program off-line - Two or more ways to reach a point - Most complex robot

**Fig. 2.15** A Gantry robot [Courtesy: Koivo (1989)]

2.2.2 Actuation Systems

Robots are driven by either electric power or fluid power. The latter category can be further subdivided into pneumatic and hydraulic. Today, the most common drive method is electric with various types of motors, e.g., stepper, dc servo, and brushless ac servo. Pneumatic robots are used in light assembly or packing work but are not usually suitable for heavy-duty tasks or where speed control is necessary. On the other hand, hydraulic

**Fig. 2.16** A SCARA arm
[Courtesy: Fuller (1999)]

robots are used in heavy payload applications because of their high power to size ratios. Actuators are explained in detail in Chapter 3.

2.2.3 Control Methods

Here, control could mean two things. One is motion control strategy, i.e., whether a robot is servo controlled or not, and the other one is how the motion path is achieved, i.e., point-to-point or continuous.

1. Servo/Non-servo Control Robots are either servo controlled (closed loop) or non-servo controlled (open loop). To gain full advantage of digital or microprocessor control, achieve good precision under heavy load conditions, and to carry out complex tasks with confidence, full servo control is necessary. In this method of control, commands are sent to the arm drives to move each axis the requisite amount. The actual movement is monitored for both displacement and velocity and compared with the command signal. The difference between the command and the action, defined as the error, is used as feedback to the controller to enable further commands to be modified accordingly. Most electric and hydraulic robots are servo controlled.

Pneumatic robots are usually non-servo controlled. In this case, a command signal is sent and it is assumed that the robot arm reaches its intended position. Non-servo control is adequate where position control of light loads only is required. However, if velocity, acceleration, and torque are to be controlled or if the movement against heavy loads is necessary then non-servo control is usually not possible. The majority of industrial robots today are servo controlled. This control problem requires knowledge of Proportional-Derivative (PD), Proportional-Integral (PI), Proportional-Integral-Derivative (PID), fuzzy, neural network, and other control theories. Some of the control algorithms are presented in Chapter 10.

2. Motion Control In Point-To-Point (PTP) motion control, the robot arm moves from one desired point to the next without regard to the path taken between them. The actual path taken may be the result of a combination of arm link movements calculated to provide the minimum travel time between the points. Point-to-point control is widely used for assembly, palletizing, and machine-tool loading/unloading. Spot-welding robots also use point-to-point control.

In Continuous Path (CP) control, a robot moves along a continuous path with specified orientations. For example, for welding two metal parts along a straight line or a specified curve, CP control is used. Signals from the sensors located at the joints are constantly monitored by the robot controller for appropriate motion control.

Theoretical development of motion control is explained in Chapter 12.

2.2.4 Robot Programming

Industrial robots can be programmed by various means. For example, they can be programmed either on-line or off-line. On-line methods require the direct use of the robot and utilize teach pendant for point-to-point or PTP programming, and slave arms or pistol grip attachments for continuous path or CP programming. More recent robots have the ability to be programmed off-line, i.e., the robot can continue working on a particular task while a program for a new task is prepared on a computer terminal using the robot programming language, for example, VAL, ALU, KRL, and others. Detail descriptions on robot programming appear in Chapter 13 of this book.

2.3 INDUSTRIAL APPLICATIONS

As mentioned in Section 1.3.1, serial robots are mainly used in industries for welding, assembling, machining, etc. As more and more robots are designed for a specific task, they can be classified based on their tasks, e.g., ‘welding robot’, ‘assembly robot’, etc. Since these robots are designed for a specified work, they cannot be made readily adaptable to other applications. For example, an assembly robot cannot be used for welding purposes unless a major change is done. Note that some suppliers provide a complete robot system, e.g., for welding, with welding equipment and other material-handling facilities like turntables, etc., as an integrated unit. Such integrated robotic system is also called a ‘welding robot’ even though its discrete manipulator unit could be adapted to a variety of tasks. Some robots are specifically designed for heavy load manipulation, and are labeled as ‘heavy-duty robots’.

There are certain industrial tasks which robots do very well like material transfer, machine loading, spot welding, continuous arc welding, spray coating, material removing, cutting, assembling, inspection, sorting, cleaning and polishing parts, and a dozen more specialized tasks. In this section, some of these applications will be examined as per their actions of end-effector (Groover et al., 2012), namely, the following:

Material Handling In these applications, the robot grasps an object, e.g., a machined component, with the help of a gripper, and moves it to another location, say, on a conveyor belt.

Processing Here, the robot uses a tool, e.g., an electrode, rather than a gripper in order to perform some processing task, say, welding.

Assembling These tasks are more complex than other two, as the robot has to interact with a fixture or another robot in order to put together two or more components or sub-assemblies to complete an assembly task.

2.3.1 Material Handling

During the sixties and the seventies, automation affected primarily the manufacturing process but not the auxiliary functions such as handling, set-up, loading, unloading, etc. The time spent to transfer a workpiece from one station to the next is still high. Up to 95 percent of the time involved in manufacturing a part is composed of transfer and waiting time, and only about 5 percent of the total time is actual processing. Whereas the processing time has been reduced considerably by automation, much less progress has been made in handling and loading. The fully automatic systems that were developed for mass production (e.g., transfer lines in the automobile industry) are rigid (hard automation) and not suitable for batch production (in the order of 50 to 100,000 parts annually). A more flexible automation technology that takes into account frequent changes in production is needed for this category of industrial production, which accounts for about 75 percent of the manufactured parts. A new solution was offered to the handling and machine-tool loading of small and medium-size parts with the development of industrial robots. Actually, loading and unloading of machine tools are the major applications of robots. Robots are utilized to load and unload machine tools in two basic configurations:

- (a) A robot tending a single machine, and
- (b) a robot serving several machines.

The first configuration is applied when typical machining times per part is short. The second configuration is applicable when a chain of operations must be executed to complete a part. For many applications, the cylindrical, polar, and revolute types are equally suitable whereas in some light-duty work in restricted areas, rectangular types may be used. For material-handling applications, point-to-point control is necessary. A typical material-handling robot is shown in Fig. 2.17.

2.3.2 Welding (Processing)

Welding is a manufacturing process in which two metal pieces are joined usually by heating and fusing. The welding operations performed by robots are thermal processes in which the metal pieces are joined by melting or fusing their contacting surfaces. These processes can be grouped under two classes, namely, where no filler material is added to the joint interface, and in which a filler material of the same parent metal is added. Accordingly, there are two types of welding operations performed by the robots, namely, spot and arc welding, respectively, as shown in Fig. 2.18.

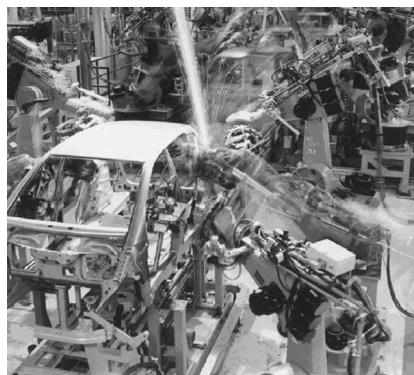
1. Spot Welding

In spot welding, two metal pieces are joined only at certain points by melting or fusing their contacting surfaces. The required heat is generated by the passage of an electric current through the metals at the point where they are to be joined. It is frequently used in the automotive industries to join thin sheet metals. A spot-welding robot has to carry the



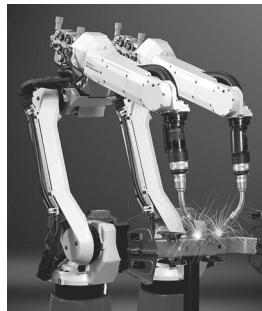
[Courtesy: <http://www.packworld.com/machinery/palletizing/hassia-usa-inc-robotic-palletizer>]

Fig. 2.17 A pick-and-place material-handling robot



[Courtesy: http://www.kukarobotics.com/en/solutions/solutions_search/L_R142_Body_shop_Mercedes_Benz_A_Class.htm]

(a) Spot welding of car parts in an assembly plant



[Courtesy: <http://www.cnc-arena.com/en/newmachines/907/jointed-arm-robot/fanucrobotics/m-10ia>]

(b) A pair of robots performing arc welding

Fig. 2.18 Welding robots

welding gun, which consists of electrodes, cables required to conduct the high current, and sometimes a water-cooling system for the electrodes. Since the welding gun is relatively heavy (10 to 80 kg), most of the spot-welding robots are hydraulically powered as dc-motor-driven robots cannot handle such heavy loads. The control system for spot-welding robots is of a point-to-point (PTP) type with reasonable positional accuracy and repeatability of ± 1 mm. This repeatability of a robot is much higher than that of a human welder. Hence, a robotic welding is preferred over manual one for uniformity of the products at hand. Further, the operation of robotized spot-welding is very fast.

Spot-welding robots, as shown in Fig. 2.18 (a) for a car assembly, affected the car industry most. However, they can be used in fabrication of metal products, domestic appliances, metal furniture, containers which do not require liquid-tight joints, etc. The first spot-welding robots were installed in 1969 at a General Motor's plant for welding of car bodies (Engelberger, 1980). Since then spot-welding robots have proved to be very profitable. Moreover, parts of the robots can be suspended from the ceiling, which saves expensive floor space. Several robots can operate simultaneously on the same car body, which increases the efficiency of the assembly line. A human operator might miss a weld or make it in an incorrect location, and, therefore, many times extra welds are added at the design stage. With robot operation, the work is consistent and all the welds are placed in the right location and, therefore, the required body strength can be achieved by specifying fewer welds. A typical assembly line produces between 50 to 90 cars per hour, and the work is performed while the car bodies are continuously moving on conveyors, which means that the weld locations specified by the task programs should be synchronized with the velocity of the assembly line.

2. Arc Welding Arc welding falls in the category in which two metals pieces are joined along a continuous path by adding a filler material of the same type as the parent metal. The required heat is provided by an electric arc generated between the electrode and the metals. It is needed, for example, in sealing a container against leakage. While most robotic-arc welding uses a consumable wire electrode, e.g., in MIG welding, with an automatic wire feeder, welding with non-consumable tungsten electrodes with shielding gas, as in TIG welding, is also in use.

In arc welding, the robot shown in Fig. 2.18(b) uses the welding gun as a tool. The consumable electrode, which provides the filler material, is in the form of a wire (coiled on a drum) of the same composition as the material to be welded. Wire diameters of $1/32$ to $3/16$ in (0.8 to 4.8 mm) are commonly used. The wire is automatically fed by a motor with adjustable speed at a preset rate that is determined by the arc voltage. The wire feed increases with an increase in the voltage applied between the work and the electrode. This voltage can be monitored and used to maintain a constant arc length by varying the speed of the motor which feeds the wire. In order to keep the electrode cooler and permit higher currents to be used, the

Accuracy vs. Repeatability

Accuracy is defined as the difference between the commanded and actual positions attained by the robot end-effector, whereas repeatability is defined as the closeness between the actual positions attained by the end-effector for the same command.

shielding gas flows in a tube along the electrode. The tube is terminated in a nozzle at the end of the gun from which the gas flows into the arc region. Robotic welding systems sometimes use water-cooled guns. The weight of the welding gun is usually not heavy (unless the water-cooled type is used) and, therefore, dc servomotor-driven robots are typically used in arc welding, although hydraulically drive robot are also sometimes found. Welding speeds range from about 10 to over 120 in/min (0.25 to 3 m/min). The welding current usually ranges between 100 and 300 A, but with the larger electrodes (3/16 in or about 4.75 mm) the current may be as high as 1200 A, resulting in a very deep penetration of the weld. The control system for robots in arc welding is usually of a Continuous Path (CP) type. Nevertheless, PTP control systems are also used.

2.3.3 Spray Painting (Processing)

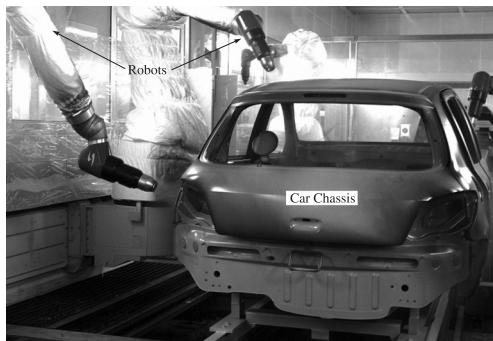
Spray painting is also a kind of processing used in manufacturing industries. The unhealthy and unpleasant environment of the painting booth in industry made this process an ideal candidate for the application of robots. The solvent materials that are used in spray painting are toxic, and, therefore, the operators must be protected by masks and be provided with fresh-air ventilation. The painting area must be dust-free and temperature-controlled, and consequently the painting booth is small in size and inconvenient for the operators. Furthermore, the noise arising from the air discharge through the painting nozzles can cause irreversible damage to the ears. For all these reasons, spray painting became one of the first applications of robots. The requirement for robots in spray painting are different from those of other robot applications, and, therefore, many robot manufacturers offer a robot dedicated to this one application. The spray-painting robots are of CP capability and have the following characteristics:

- (a) high level of manipulator dexterity,
- (b) large working volume for small-base manipulator,
- (c) compact wrist,
- (d) small payload, and
- (e) low accuracy and repeatability.

Figure 2.19 shows the spray painting of a car chassis by robots. The painting robot must be able to carry any type of spray gun. Spray guns, however, are light weight and, therefore, painting robots are designed for small payloads (e.g., 1 kg). Finally, the requirements for repeatability and resolution are the least severe in painting robots. Therefore, a repeatability of 2 mm throughout the working volume is regarded as sufficient for spray-painting robots.

2.3.4 Machining (Processing)

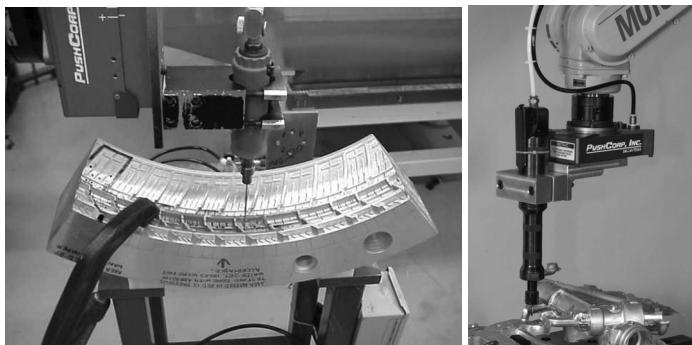
Machining is another type of manufacturing process. There are five basic types of machine tools to perform machining, namely, drilling machine, lathe or turning machine, milling machine, shaper, and grinder. Out of all these machining operations, only drilling is being successfully done with robots, and mainly in the aircraft industry. Another application related to machining which is performed by robots is deburring of metal parts.



[Courtesy: [http://www.roboticsbible.com/
spray-painting-robot.html](http://www.roboticsbible.com/spray-painting-robot.html)]

Fig. 2.19 Spray-painting robots with car chassis

1. Drilling Robots can replace the manual operators if the template hole is provided with a chamfered guide. The gripper of the robot holds a portable drill and guides from hole to hole. At each hole, a fixed drill cycle is performed, and then the robot moves the drill to the next hole. Programming the robot to perform the task is quite simple. Since drilling is a PTP operation, the manual teaching method is appropriate. Figure 2.20(a) shows a robotized drilling operation.



[Courtesy: [http://www.romheld.com.au/
sub_products.php](http://www.romheld.com.au/sub_products.php)]

(a) Drilling

(b) Deburring

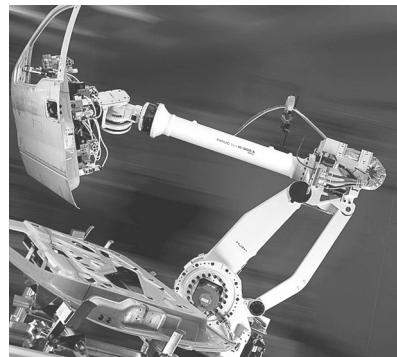
Fig. 2.20 Machining robots

2. Deburring Burrs are generated almost always when machining is performed on metal parts. Burrs can be generated between a machined surface and a raw surface or at the intersection between two machined surfaces. The removal of these burrs is an expensive operation. Most deburring is performed manually by workers equipped with appropriate tools. By closely following the manual method, the industrial robots can solve most deburring problems. Figure 2.20(b) shows deburring operation by a robot. There are two basic ways to perform robotized deburring. If the part is relatively lightweight, it can be picked up by the robot and brought to the deburring tool. If the part is heavy then the robot holds the tool. The support of the tool is very important, whether it is held by the robot or mounted on the work table. In both cases, the relative motion between the tool and the part is of a CP type with high repeatability (approximately 0.2 mm) and highly controlled speed. Therefore, deburring is one of the most difficult tasks for robots.

2.3.5 Assembling

Assembling with industrial robots is mainly used for small products such as electrical switches and small motors. Robotized assembly systems are programmable and, therefore, provide a cost-effective solution for the assembly of small batch sizes and for batches containing different products. Although industrial robots require the same fixtures, feeders, and other equipment for positioning the parts as conventional assembly machines, simpler workpiece feeder and fixtures may be used because of the robots' programmability feature. Furthermore, tactile or optical sensors may be added to the assembly robot to tackle more complex assembly tasks. Some assembly tasks require the participation of more than one robot. In order to reduce the cost per arm, there are systems in which several Cartesian arms can use the same base and share the same controller.

Assembly robots, e.g., the one similar to Fig. 2.21, can be designed in any coordinate system: Cartesian, cylindrical, spherical, or articulated. However, many tasks require only vertical assembly motions, such as the assembly of printed circuit boards. For these applications, the 4-DOF robot shown in Fig. 2.22 can be adequate. Its arm has two articulated motions, and the wrist has two axes of motion: a linear vertical displacement and a roll movement. This robot can pick up parts located on the horizontal plane, bring them to the assembly location, orient them with the roll motion of the wrist, and finally insert them in a vertical motion. This class of robot is known as the SCARA type which was first developed in Japan.



[Courtesy: <http://www.fanucrobotics.com>]

Fig. 2.21 Assembly robot



[Courtesy: <http://www.systemantics.com/products.htm>]

Fig. 2.22 A 4-DOF robot from Systemantics India

Example 2.3 Robot Considerations for an Application

There are several points which should be taken into account while choosing a robot for an application. Some of the important ones are listed below:

- Degrees-of-freedom (DOF) of the robot for the planned application
- Robot work volume
- Payload capacity
- Type of end-effector, i.e., gripper for material handling, etc.
- For grippers, its total DOF, number of fingers or jaws
- Actuator type, i.e., electric, hydraulic, or pneumatic
- Point-to-point or PTP (mostly used in material-handling applications) or continuous-path or CP (welding, spraying and similar applications)

- Programming language
 - Safety features
 - Initial investment and running cost
 - Payback period
-

SUMMARY

In this chapter, several subsystems of a robotic system are presented. A robot is classified based on its workspace, controlling method, etc. Industrial applications of the robots in the area of manufacturing, etc., are also presented in detail. Classification of a robot, either based on workspace or application, will enable a user to select/order a robot for his or her requirements.

EXERCISES

- 2.1** What are the different subsystems of a robotic system?
- 2.2** Why are transmissions used in a robot manipulator?
- 2.3** When should a belt-pulley be used over gears?
- 2.4** Show an arrangement of conveying boxes.
- 2.5** What are the advantages of using a mechanism over other transmission arrangements?
- 2.6** What are different ways of classifying a robot?
- 2.7** Why are ADC and DAC required in robots?
- 2.8** What kind of robot is suitable for painting a shaft? Why?
- 2.9** What is the shape of the workspace of the SCARA robot similar to Fig. 2.22? Draw it.
- 2.10** Why are some workspaces empty?
- 2.11** What is the full form of PUMA?
- 2.12** What type of basic arm does the PUMA robot have?
- 2.13** What type of control is suitable in welding operations?
- 2.14** Name typical industrial applications of robots.
- 2.15** What is the repeatability demanded in a spraying robot?

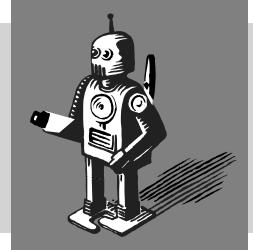
WEB-BASED EXERCISES

Based on Web search, find the answers to the following questions:

- 2.16** Name some robot manufacturers and their robot programming languages.
- 2.17** What are the typical gear ratios used in industrial robots?
- 2.18** Find some industrial applications where gantry robots are used.
- 2.19** Name some industries which employ SCARA robots and why.
- 2.20** Find out the model numbers of KUKA, ABB, and Kawasaki robots for the payload capacity of up to 15 kg.
- 2.21** What are the accuracy and repeatability of the above robots?
- 2.22** What are typical prices of industrial robots used in welding of car bodies?

3

Actuators and Grippers

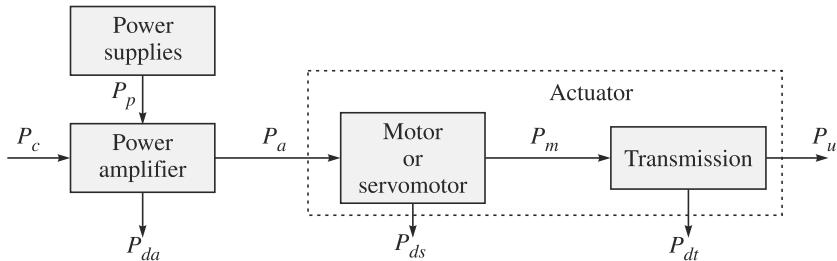


Actuators are devices which drive a robot including its grippers. They are like the muscles of a human arm and hand. While the human arm provides motion, the hand is used for object manipulation. As illustrated in Fig. 2.3, the same is true with a robot. Its arm, be Cartesian or anthropomorphic or of other type, which are explained in Section 2.2, provides motion, whereas its gripper, which is equivalent to a human hand, manipulates objects. Both require some kind of actuator system powered by electricity or compressed air or pressurized fluid. These systems are presented in this chapter, along with different types of grippers used in robotic applications.

An actuator system comprises of several subsystems, namely,

- (i) a power supply;
- (ii) a power amplifier;
- (iii) a servomotor; and
- (iv) a transmission system.

The connections between all the actuator components are depicted in Fig. 3.1.



P_p : Primary source of power (electricity or pressurized fluid or compressed air); P_c : Input control power (usually electric); P_a : Input power to motor (electric or hydraulic or pneumatic type); P_m : Power output from motor; P_u : Mechanical power required; P_{da} , P_{ds} and P_{dt} : Powers lost in dissipation for the conversions performed by the amplifier, motor, and transmission

Fig. 3.1 An actuator system

To choose an actuator, it is worth starting from the requirements imposed on the mechanical power P_u by the force and velocity that describe the joint motion. Based on the source of input power P_a the actuators can be classified into three groups:

Actuator vs. Motor

A motor together with transmissions and other accessories, if any, is referred as an actuator (see Fig. 3.1). However, in many instances, they are used interchangeably to mean that they move a robot link.

1. Electric Actuators The primary input power supply is the electric energy from the electric distribution system.

2. Hydraulic Actuators They transform hydraulic energy stored in a reservoir into mechanical energy by means of suitable pumps.

3. Pneumatic Actuators They utilize pneumatic energy, i.e., compressed air, provided by a compressor and transform it into mechanical energy by means of pistons or turbines.

A portion of the motor input power P_a is converted to the output as a mechanical power P_m , and the rest P_{ds} is dissipated because of mechanical, electrical, hydraulic, or pneumatic motor losses. In a robotic application, an actuator should have the following characteristics:

- Low inertia
- High power-to-weight ratio
- Possibility of overload and delivery of impulse torques
- Capacity to develop high accelerations
- Wide velocity ranges
- High positioning accuracy
- Good trajectory tracking and positioning accuracy

Based on wide application of electric actuators in industrial robots and their easy availability in the stores next door selling toys or repairing photocopiers, washing machines, etc., they are presented first. It will be followed by hydraulic and pneumatic actuators. While hydraulic actuators are suitable for applications where the requirement is high power-to-weight ratio, pneumatic actuators are often used by electrically actuated robots for their grippers requiring only on-off motions of the jaws. Use of a pneumatic gripper makes the robot system a little lighter and cost effective.

Who is ABB?

ABB is a Zurich, Switzerland-based multinational company operating in power and automation technology areas. It was formed in 1988 through the merger of ASEA of Sweden and Brown, Boveri and Cie of Switzerland. It has operations in more than 100 countries, and is one of the major industrial robot manufacturers in the world.

3.1 ELECTRIC ACTUATORS

Electric actuators are generally those where an electric motor drives robot links through some mechanical transmission, e.g., gears, etc. In the early years of industrial robotics, hydraulic robots were the most common, but recent improvements in electric-motor design have meant that most new robots are of all-electric construction. The first commercial electrically driven industrial robot was introduced in 1974 by ABB. The advantages and disadvantages of an electric motor are the following:

Advantages

- Widespread availability of power supply.
- Basic drive element in an electric motor is usually lighter than that for fluid power, i.e., pressurized fluid or compressed air.
- High power-conversion efficiency.

- No pollution of working environment.
- Accuracy and repeatability of electric drive robots are normally better than fluid power robots in relation to cost.
- Being relatively quiet and clean, they are very acceptable environmentally.
- They are easily maintained and repaired.
- Structural components can be lightweight.
- The drive system is well suited to electronic control.

Disadvantages

- Electrically driven robots often require the incorporation of some sort of mechanical transmission system.
- Additional power is required to move the additional masses of the transmission system.
- Unwanted movements due to backlash and plays in the transmission elements.
- Due to the increased complexity with the transmission system, we need complex control requirement and additional cost for their procurement and maintenance.
- Electric motors are not intrinsically safe, mainly, in explosive environments.

The above disadvantages are gradually being overcome with the introduction of direct-drive motor system, in which the electric motor is a part of the relevant robot arm joint, thus, eliminating the transmission elements. Moreover, the introduction of newer brushless motors allow electric robots to be used in some fire-risk applications such as spray painting, as the possibility of sparking at the motor brushes is eliminated. Different types of electric motors are stepper motors, and the dc and ac motors, which are explained next.

dc vs. ac

dc stands for *direct current*, whereas ac means *alternating current*. Batteries in flash lights, cars, etc., supply dc. For domestic use, however, the electric supply is ac with 230 volts, 50 cycles (in India and other countries), 110 volts, 60 cycles (USA and others), and similar.

3.1.1 Stepper Motors

Stepper motors (also called stepping motors or step motors) were first used for remote control of the direction indicators of torpedo tubes and guns in British warships and later, for a similar purpose, in the US Navy. A variable reluctance-type stepper motor was first patented in 1919 by C.L. Walker, a Scottish civil engineer. However, its commercial production did not start until 1950.

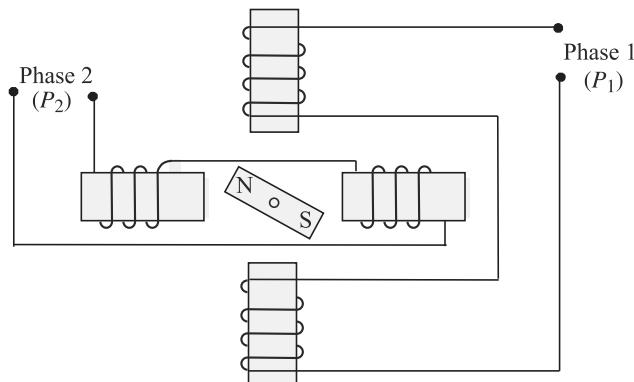
Such motors are used with small and medium robots and with teaching and hobby robots. These motors are widely used in other industrial applications mainly due to their advantage of not requiring any feedback system and, accordingly not incurring the associated cost. Stepper motors are, however, compatible with many feedback devices, if so desired, and are used in full servo-control configurations in medium-duty industrial robots. Because they are digitally controlled, they can be referred to as *digital motors* or *actuators*. These motors do not require the expense of digital-to-analog conversion equipment when connected to a computer-control system. A commercially available stepper motor is shown in Fig. 3.2.



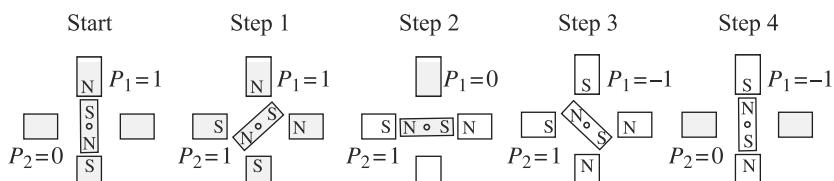
[Courtesy: <http://www.pololu.com>]

Fig. 3.2 Stepper motor (Bipolar, 200 Steps/Rev, 20 × 30 mm, 3.9 V, 0.6 A/Phase)

Normally, the shaft of a stepper motors rotates incrementally in equal steps in response to a programmed input pulse train. As shown in Fig. 3.3(a), current in any of the two phases, i.e., with P_1 and P_2 , will magnetize the pair into north and south poles, indicated with N and S, respectively. Accordingly, the permanent magnet at the centre will rotate in order to align itself along a particular phase, which is demonstrated in Fig. 3.3(b). Switching the currents in the two phases in an appropriate sequence can produce either clockwise (CW) or counterclockwise (CCW) rotations, as tabulated in Table 3.1. The switching sequence of Table 3.1 corresponds to what is known as *half-stepping* with a step angle of 45° , whereas the full-stepping corresponds to 90° in which one phase is energized at a time. See Example 3.3. It is pointed out here that



(a) A two-phase stepper motor



(b) Half-stepping sequence (both phases energized simultaneously)

Fig. 3.3 Working principle of a two-phase stepper motor

one can also achieve micro-stepping (non-identical steps) up to 1/125 of full-step by changing the currents in small steps instead of switching them on and off, as in the case of half- and full-stepping. While micro-stepping is advantageous from the point of view of accurate motion control using a stepper motor, it has the disadvantage of reduced motor torque.

Note that the steps are achieved through phase activation or switching sequences triggered by the pulse sequences. The switching logic that decides the states of the phases of a given step can be generated using a look-up table given in Table 3.1. The same sequences can also be generated using a logic circuitry which is typically an electronic device.

Table 3.1 Half-stepping sequence of a two-phase stepper motor
(both phases energized simultaneously)

Step \ Phases	1	2	3	4	5	6	7	8	Rotation
P_1	1	0	-1	-1	-1	0	1	1	CW
P_2	1	1	1	0	-1	-1	-1	0	CCW

As the rotor indexes round a specific amount for each control pulse, any error in positioning is noncumulative. To know the final position of the rotor, all that is required is to count the number of pulses fed into the stator's phase winding of the motor. The number of pulses per unit time determines the motor speed. The rotor can be made to index slowly, coming to rest after each increment or it can move rapidly in a continuous motion termed *slewing*. Maximum dynamic torque in a stepper motor occurs at low pulse rates. Therefore, it can easily accelerate a load. Once the required position is achieved and the command pulses cease, the shaft stops without the need for clutches or brakes. The actual rotational movements or step angles of the shaft are obtainable typically from 1.8° to 90° depending on the particular motor choices. Thus, with a nominal step angle of 1.8° , a stream of 1000 pulses will give an angular displacement of 1800° or five complete revolutions. They have also a low velocity capability without the need for gear reduction. For instance, if the previously mentioned motor is driven by 500 pulses per second, it will rotate at 150 rpm. Other advantages of the stepper motor are that the motor inertia is often low, and also if more than one stepper motor is driven from the same source then they will maintain perfect synchronizations.

Some disadvantages of stepper motors are that they have a lower output and efficiency compared to other motors, namely, dc and ac, and drive inputs and circuitry have to be carefully designed in relation to the torque and speed required. There are various types of stepper motors which are based on the nature of a motor's rotor, e.g., variable reluctance, permanent magnet, and hybrid. A good account of stepper motors is given in de Silva (2004), whereas an online video showing their

working principles are available in WR-Stepper (2013)¹. A typical specification of a stepper motor is given in Table 3.2, whereas a general torque-speed plot is shown in Fig. 3.4.

Table 3.2 Specifications of a stepper motor [Courtesy: www.robokits.co.in]

Brand and Model	NEMA 23, RMCS-115
Step angle	1.8°
Configuration	6 wire stepper motor
Holding torque	10 kg-cm bipolar
Rated voltage	3.3 VDC
Phase current	2 Amp
Resistance/Phase	1.65 E
Inductance/Phase	2.2 mH
Rotor inertia	275 gcm ²
Detent torque	0.36 kg-cm
Length	51 mm
Weight	650 grams

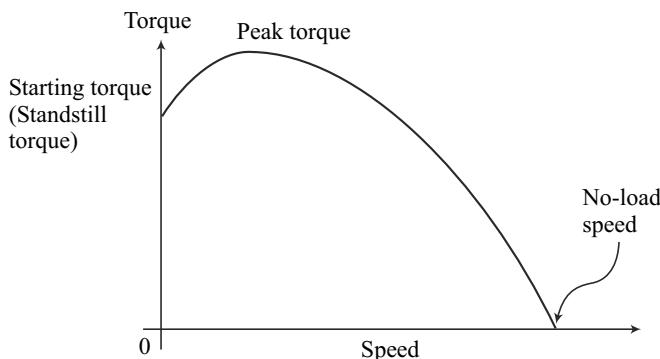


Fig. 3.4 Torque-speed characteristics of a stepper motor

1. Variable-Reluctance Stepper Motor Magnetic reluctance, or simply *reluctance*, is the analog of electrical resistance. Just as current occurs only in a closed loop, so magnetic flux occurs only around a closed path, although this path may be more varied than that of the current. Figure 3.5(a) shows the basic form of the variable-reluctance stepper motor. The rotor is made of soft steel and it has four poles, whereas the stator has six

Detent Position

It is the equilibrium configuration when the rotor assumes the minimum reluctance position for that step.

¹ WR stands for Web Reference given at the end of the list of References. They mainly contain the links to online materials including videos, which are expected to enhance the understanding of a concept to the reader.

poles. When one of the phases, say AA' , is excited due to a dc current passing through the coils around the poles, the rotor positions itself to complete the flux path shown in Fig. 3.5(a). Note that there is a main flux path through the aligned rotor and stator teeth, with secondary flux paths occurring as indicated. When rotor and stator teeth are aligned in this manner, the reluctance is minimized and the rotor is at rest in this position. This flux path can be considered rather like an elastic thread and always trying to shorten itself. The rotor will move until the rotor and stator poles are lined up. This is termed as the position of minimum reluctance. To rotate the motor counterclockwise, the phase AA' is turned off and phase BB' is excited. At that point, the main flux path has the form indicated in Figs. 3.5(b-c). This form of a stepper motor generally gives step angles of 7.5° or 15° , which are referred as *half-stepping* and *full-stepping*, respectively. Note that a disadvantage of variable-reluctance stepper motors is that it has zero holding torque when the stator windings are not energized (power off) because the rotor is not magnetized. Hence, it has no capacity to hold a load in power-off mode unless mechanical brakes are employed.

Holding Torque vs. Detent Torque

Holding torque is the amount of torque that the motor produces when it has rated current flowing through the windings but the motor is at rest, whereas the *detent torque* is the amount of torque that the motor produces when it is not energized, i.e., no current is flowing through the windings.

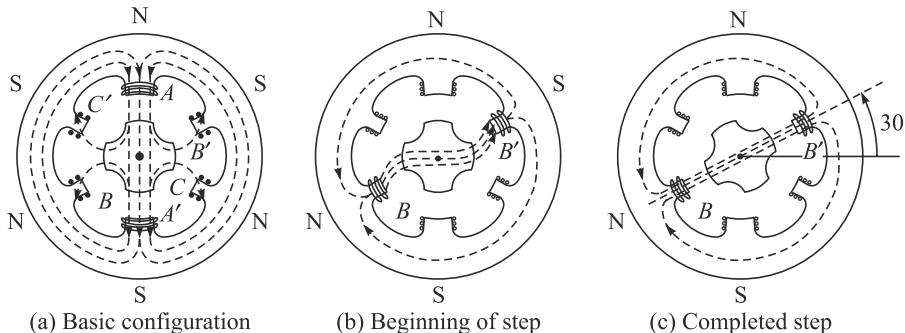


Fig. 3.5 Variable-reluctance stepper motor

2. Permanent-Magnet Stepper Motor The basic method of operation of a permanent-magnet type is similar to the variable-reluctance type. As illustrated in Fig. 3.6, there are two coils A and B , each of them having four poles but displaced from each other by half a pole pitch. The rotor is of permanent-magnet construction and has four poles. Each pole is wound with field winding, the coils on opposite pairs of poles being in series. Current is supplied from a dc source to the winding through switches. It can be seen in Fig. 3.6(a) that the motor is at rest with the poles of the permanent magnet rotor held between the

Harmonic drive and Ratchet-and-pawl vs. Stepper motors

Harmonic drive and *ratchet-and-pawl* are two transmission mechanisms which produce intermittent motions purely through mechanical means. They can be loosely categorized as stepper motors.

residual poles of the stator. In this position, the rotor is locked unless a turning force is applied. If the coils are energized and, in the first pulse, the magnetic polarity of the poles of the coil A is reversed, the rotor will experience a torque and will rotate counterclockwise, as shown in Fig. 3.6(b). The reverse poles are shown as A'.

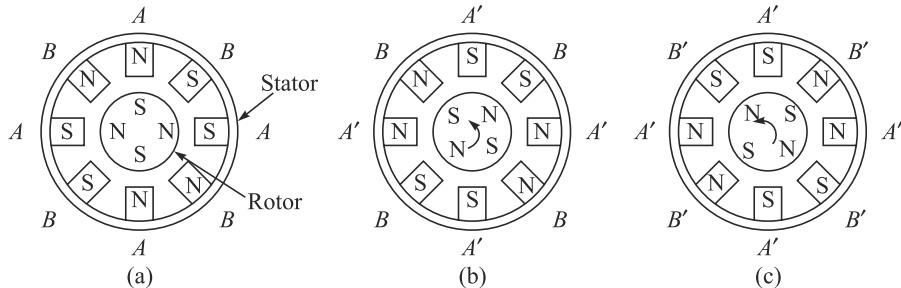


Fig. 3.6 Permanent-magnet stepper motor

If the coil B poles are now reversed to B', as shown in Fig. 3.6(c), the rotor will again experience a torque, and once more the poles of the rotor are positioned midway between the stator poles. Thus, by switching the currents through the coils, the rotor rotates by 45° . If in the first pulse, the poles of the coil B had been reversed then the motor would have rotated clockwise. With this type of motor, commonly produced step angles are 1.8° , 7.5° , 15° , 30° , 34° , 90° .

3. Hybrid Stepper Motor Hybrid stepper motors are the most common variety of stepper motors in engineering applications. They combine the features of both the variable reluctance and permanent-magnet motors, having a permanent magnet encaged in iron caps which are cut to have teeth, as shown in Fig. 3.7. A hybrid stepper motor has two stacks of rotor teeth on its shaft (WR-Stepper, 2013). The two rotor stacks are magnetized to have opposite polarities, while two stator segments surround the two rotor stacks. The rotor sets itself in the minimum reluctance position in response to a pair of stator coils being energized. Typical step angles are 0.9° and 1.8° .

From the descriptions of stepper motors, it is, therefore, apparent that the rate at which the pulses are applied determines the motor speed, the total number of pulses determines the angular displacement, and the order of energizing the coils in the first instance determines the direction of rotation. It is because of this ease of driving using direct digital control that stepper motors are well suited for use in a computer-controlled robot, although the motor does require interfacing with a high-current pulse source.

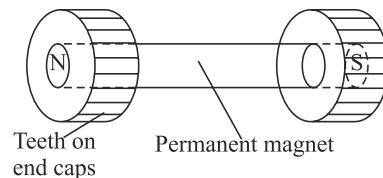


Fig. 3.7 Hybrid stepper motor

Example 3.1 Pitch Angles of Rotors and Stators

If the angles of two adjacent teeth of a rotor and a stator, called the pitch angles, are denoted by θ_r and θ_s , respectively, they can be calculated as

$$\theta_r = \frac{360^\circ}{n_r} \text{ and } \theta_s = \frac{360^\circ}{n_s} \quad (3.1)$$

where by n_r and n_s are the corresponding number of teeth in rotors and stators.

Stepper Motor Model

Model of a stepper motor is described by its torque expression with respect to its rotor angle, i.e., $\tau = -\tau_{\max} \sin(n_r \theta)$, where τ_{\max} is maximum torque during a step (holding torque), n_r is the number of rotor teeth, and θ is any angular position.

Example 3.2 Step Angle

Suppose, in a variable reluctance stepper motor the number of steps required is, $n = 200$. Then, the step angles $\Delta\theta$ can be calculated as

$$\Delta\theta = \frac{360^\circ}{n} = 1.8^\circ \quad (3.2)$$

Example 3.3 Sequence for Full-step Angle

As explained in the beginning of Section 3.1.1, the full-stepping requires that one phase is energized at a time. For the two-phase motor of Fig. 3.3, the stepping sequence is given in Table 3.3. The rotor positions are shown in Fig. 3.8.

Table 3.3 Full-stepping sequence for a two-phase motor

Step Phases	1	2	3	4	Rotation
P_1	0	-1	0	1	CW
P_2	1	0	-1	0	CCW

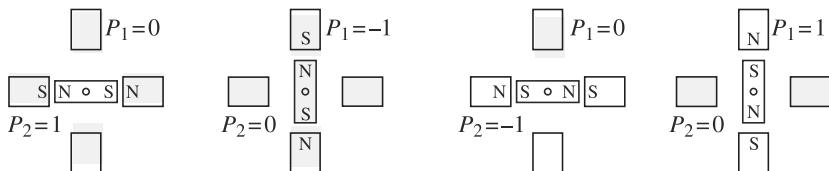


Fig. 3.8 Stepping sequence (one phase energized at a time)

3.1.2 dc Motors

People in their daily lives experience electric motors, whether they are dc (often battery operated) motors to run a child's toy or ac (mains operated) motors to turn the blenders in the kitchen, etc. The first commercial electrically driven industrial robot was introduced in 1974 by the Swedish giant corporation ASEA. Traditionally, roboticists have employed electrically driven dc (direct-current) motors for robots because, not only are powerful versions available, but they are also easily controllable with relatively simple electronics. Although direct current is needed, batteries are rarely used (for nonmobile robots) but instead the ac supply is *rectified* into a suitable dc equivalent. The operation of any electric motor is based upon the principle which states that a conductor will experience a force if an electric current in that conductor flows at right angles to a magnetic field. Therefore, to construct a motor, two basic components are required. One to produce the magnetic field usually termed the *stator*, and another to act as the conductor usually termed the *armature* or the *rotor*. The principle is shown in Fig. 3.9(a) for one element of a dc motor, whereas a two-pole dc motor is shown in Fig. 3.9(b). The magnetic field may be created either by *field coils* wound on the stator or by permanent magnets. The field coils, if used, would be provided with an electric current to create magnetic poles on the stator. In Fig. 3.9(a), the current is supplied to a conductor via the *brushes* and *commutators*. The current passing through the field produces a torque, or more accurately static torque on the conductor, which can be given by

$$\tau = 2fr \sin \sigma \quad (3.3a)$$

where f is the magnetic force exerted on the conductor, r is the radius of the rotor, and σ is the angle between the flux density vector of stator magnetic field \mathbf{b} and the unit vector normal to the plane of the coil \mathbf{n} , as shown in Fig. 3.9(a). The angle σ is known as *torque angle*. In view of Lorentz's law, the imbalanced magnetic force f that is generated on the conductor, normal to the magnetic plane is given by

$$f = Bi_a l \quad (3.3b)$$

in which B is the flux density of the original field, i_a is the current through the conductor, i.e., coil rotor or armature formed by the conductors, and l is the length of

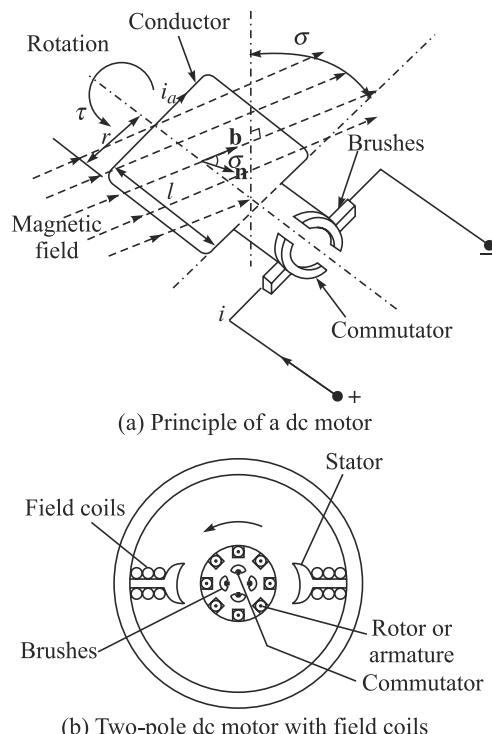


Fig. 3.9 dc motors

the conductor. Substituting Eq. (3.3b) into Eq. (3.3a), the torque τ generated in the rotor is given by

$$\tau = A i_a b \sin \sigma \quad (3.3c)$$

In Eq. (3.3c), $A \equiv 2lr$, which is nothing but the face area of the planar rotor, as shown in Fig. 3.9(a). Moreover, unit vectors \mathbf{b} and \mathbf{n} represent the field direction and normal to the plane of the rotor.

According to Eq. (3.3c), τ is maximum when $\sigma = 90^\circ$. Moreover, the higher the voltage supplied to the stator coils of the motor, the faster the motor turns, providing a very simple method of speed control. Similarly, varying the current to the armature controls the torque. Reversing the polarity of the voltage causes the motor to turn in the opposite direction. Some larger robots utilize field control dc motors, i.e., motors in which the torque is controlled by manipulating the current to the field coils. These motors allow high power output at high speeds and can give a good power-to-weight ratio. A typical specification of a dc motor is given in Table 3.4, whereas its speed-torque characteristic curve is shown in Fig. 3.10 for different voltage values.

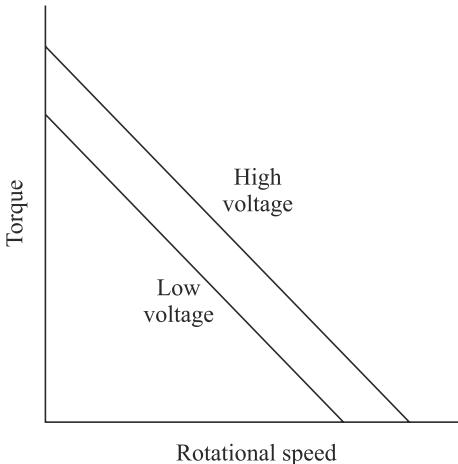
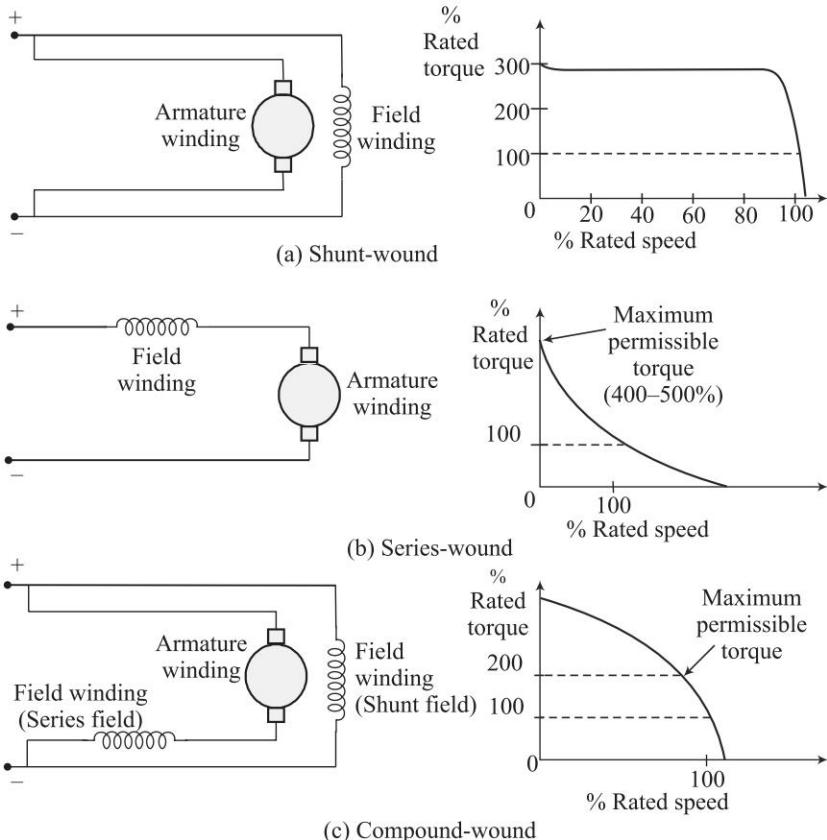


Fig. 3.10 Speed-torque speed characteristics of a dc motor

Table 3.4 Specifications of a dc motor [Courtesy: <http://uk.rs-online.com>]

Brand	Parvalux
Manufacturer part no.	PM2 160W511109
Type	Industrial dc Electric Motors
Shaft size (S, M, L)	M
Speed (rpm)	4000 rpm
Power rating (W)	160 W
Voltage rating (Vdc)	50 V(dc)
Input current	3.8 A
Height (mm)	78 mm
Width (mm)	140 mm
Length (mm)	165 mm

Note that the shape of steady-state speed-torque curves will be modified depending on how the windings of the rotor and the stator are connected. There are three typical ways to do that, namely, shunt-wound motor, series-wound motor, and compound-wound motor. Figure 3.11 shows the above three windings.

**Fig. 3.11** Types of windings of dc motors

In a shunt-wound motor, the armature windings and field windings are connected in parallel, as shown in Fig. 3.11(a). At steady state, the back electromotive force (e.m.f.) depends directly on the supply voltage. Since the back e.m.f. is proportional to the speed, it follows that the speed controllability is good with the shunt-wound configuration. In series-wound motors, as the name suggests, they are connected in series. This is shown in Fig. 3.11(b). The relation between the back e.m.f. and supply voltage is coupled through both the armature windings and the field windings. Hence, its speed controllability is relatively poor. But in this case, a relatively large current flows through both windings at low speeds of the motor, giving higher starting torque. Also, the operation is approximately at constant power. In the compound-wound motor, a part of the field windings is connected with the armature windings in series and the other part is connected in parallel, as shown in Fig. 3.11(c). This kind of motors provides a compromise performance between the extremes of speed controllability and higher starting torque characteristics, as provided by the shunt-wound and series-wound motors, respectively. Table 3.5 summarizes the properties derived from the above three types of windings. For an industrial robot, in general, it is said that the current excited field control methods involve too slow a response

time and incur losses that make permanent-magnet fields and armature control more attractive, which are explained next.

Table 3.5 Steady-state characteristics of a dc motor [Courtesy: de Silva, 2004]

Serial No.	Windings in a dc motor	Field-coil resistance	Speed controllability	Starting torque
1	Shunt-wound	High	Good	Average
2	Series-wound	Low	Poor	High
3	Compound-wound	Parallel high, series low	Average	Average

1. Permanent-Magnet (PM) dc Motors The permanent-magnet dc motor, which is also referred as *torque motor*, can provide high torque. Here, no field coils are used and the field is produced by the permanent magnets themselves. These magnets should have high-flux density per unit yielding a high torque/mass ratio. Typical materials with desired characteristic of such dc motors are rare-earth materials such as samarium cobalt, etc. Some PM motors do have coils wound on the magnet poles but these are simply to recharge the magnets if their strength fails. Due to the field flux being a constant, the torque of these motors is directly proportional to the armature current. Some other advantages are: excitation power supplies for the field coils are not required, reliability is improved as there are no field coils to fail, and no power loss from field coils means efficiency and cooling are improved. However, these types of motors are more expensive. Two types of PM configurations are shown in Fig. 3.12. They are cylindrical and disk types.

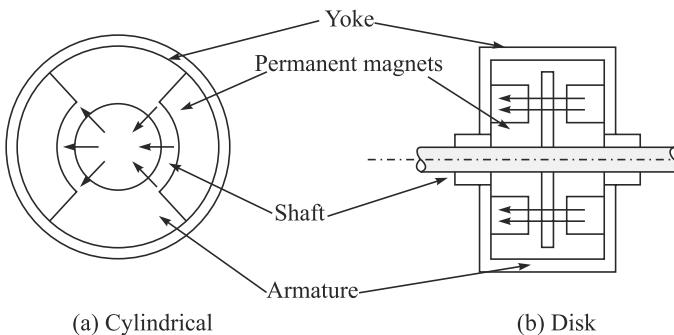


Fig. 3.12 Two types of permanent-magnet dc motor configurations

The cylindrical motor operates in a similar manner to that already described for other dc motors except that there are no field coils, whereas the disk motor has a large diameter, short-length armature of nonmagnetic material. It is the cylindrical motor that is more commonly used in industrial robots.

2. Brushless Permanent-Magnet dc Motors A problem with dc motors is that they require a commutator and brushes in order to periodically reverse the current through each armature coil. The brushes make sliding contacts with the commutators and as a consequence sparks jump between the two and they suffer wear. Brushes

thus have to be periodically changed and the commutator resurfaced. To avoid such problems, brushless motors have been designed. Essentially, they consist of a sequence of stator coiled and a permanent magnet rotor. A current-carrying conductor in a magnetic field experiences a force; likewise, as a consequence of Newton's third law of motion, the magnet will also experience an opposite and equal force. With the conventional dc motor, the magnet is fixed and the current-carrying conductors made to move. With the brushless permanent-magnet dc motor the reverse is the case, the current-carrying conductors are fixed and the magnetic field moves. The rotor is a ferrite or ceramic permanent magnet. In concept, brushless dc motors are somewhat similar to permanent-magnet stepper motors explained in Section 3.1.1 and to some type of ac motors, as in Section 3.1.3. Figure 3.13 shows the basic form of such a motor. The current to the stator coils is electronically switched by transistors in sequence round the coils, the switching being controlled by the position of the rotor so that there are always forces acting on the magnet causing it to rotate in the same direction.

The *brushless* motors have many advantages over conventional dc motors. For example,

- They have better heat dissipation, heat being more easily lost from the stator than the rotor.
- There is reduced rotor inertia. Hence, they weigh less and low mass for a specified torque rating.
- The motors in themselves are less expensive.
- They are more durable and have longer life.
- Low maintenance.
- Lower mechanical loading.
- Improved safety.
- Quieter operation.
- They are of smaller dimensions of comparable power.
- The absence of brushes reduces maintenance costs due to brush and commutator wear, and also allows electric robots to be used in hazardous areas with flammable atmospheres such as are found in spray-painting applications.

One disadvantage is that the control systems for brushless motors are relatively expensive due to sensing and switching.

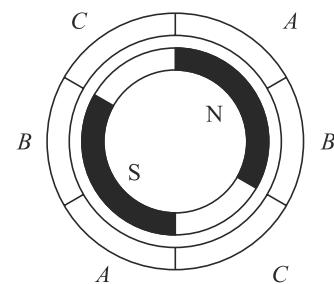


Fig. 3.13 Brushless permanent-magnet dc motor

3. dc Servomotors and their Drivers Servomotors are motors with motion feedback control, which are able to follow a specified motion trajectory. In a dc servomotor, both angular position and speed might be measured using, say, shaft encoders, tachometers, resolvers, potentiometers, etc., and compared with the desired position and speed. The error signal which is the difference between the desired minus actual responses is conditioned and compensated using analog circuitry or

is processed by a digital hardware processor or a computer, and supplied to drive the servomotor toward the desired response. Motion control implies indirect torque control of the motor that causes the motion. In some applications like grinding, etc., where torque itself is a primary output, direct control of motor torque would be desirable. This can be accomplished using feedback of the armature current or the field current because those currents determine the motor torque. A typical layout of a dc servomotor is given in Fig. 3.14.

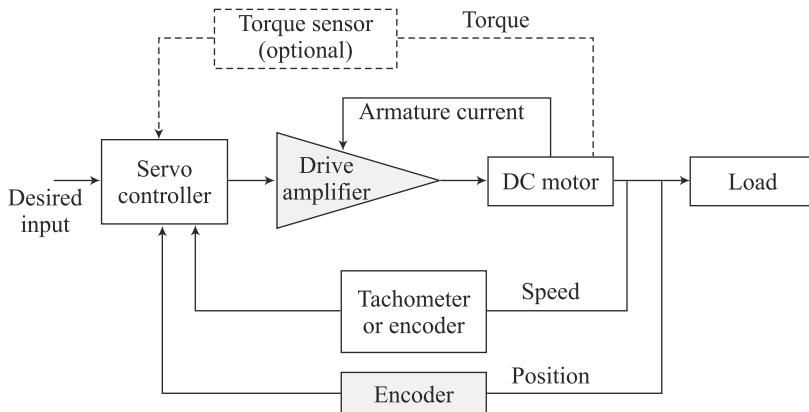


Fig. 3.14 A dc servomotor [Courtesy: de Silva, 2004]

Note that the control of a dc motor is achieved by controlling either the stator field flux or the armature flux. If the armature and field windings are connected through the same circuit, i.e., one of the winding types shown in Fig. 3.11, both techniques are implemented simultaneously. Two methods of control are armature control and field control. In armature control, the field current in the stator circuit is kept constant and the input voltage to the rotor is varied in order to achieve a desired performance. In the field control, on the other hand, the armature voltage is kept constant and input voltage to the field circuit is varied. These winding currents are generated using a *motor driver*. It is a hardware unit that generates necessary current to energize the windings of the motor. By controlling the current generated by the driver, the motor torque can be controlled. By receiving feedback from a motion sensor (encoder, tachometer, etc.), the angular position and the speed of the motor can be controlled.

The drive unit of a dc servomotor primarily consists of a driver amplifier (commercially available amplifiers are linear amplifier or pulse-width-modulation, i.e., PWM, amplifier), with additional circuitry and a dc power supply. The driver is commanded by a control input provided by a host computer through an interface (input/output) card. A suitable arrangement is shown in Fig. 3.15. Also, the driver parameters like amplifier gains are

Impedance vs. Inertia Matching

In electrical systems, impedance matching is to equate the input impedance of a load (or output impedance) with the input impedance to maximize the power transfer. Similarly, in a motor-gearbox load configuration, the inertia (or mechanical impedance) of the load seen by the motor should be the same as its inertia in order to obtain maximum power transfer or acceleration.

software programmable and can be set by the host computer. The control computer receives a feedback signal of the motor motion, through the interface board, and generates a control signal, using a suitable control law explained in Chapter 10. The signal is then provided to the drive amplifier, again through the interface board. An interface board or Data Acquisition (DAQ) card is a hardware module with Digital-to-Analog (DAC) and Analog-to-Digital (ADC) capabilities built-in. These are generally parts of a robot's control system, as mentioned in Section 2.1.3.

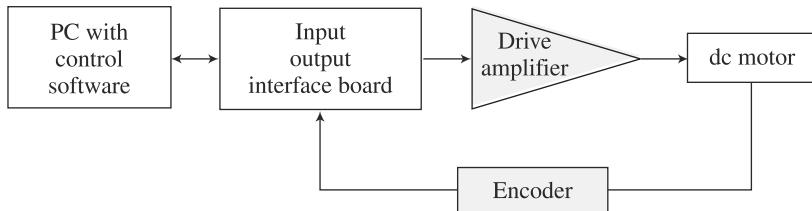


Fig. 3.15 Controller of a dc servomotor

The final control of a dc motor is accomplished by controlling the supply voltage to either the armature circuit or the field circuit. A dissipative method of achieving this involves using a variable resistor in series with the supply source to the circuit. This method has disadvantages of high heat generation, etc. Instead, the voltage to a dc motor is controlled by using a solid-state switch known as a *thyristor* to vary the off time of fixed voltage level, while keeping the period of pulse signal constant. Specifically, the duty cycle of a pulse signal is varied, as demonstrated in Fig. 3.16. This is called *pulse-width-modulation* or PWM. A commercial dc servomotor along with its controller unit is shown in Fig. 3.17.

Duty Cycle
It is given in percentage as $\frac{\text{On period}}{\text{Pulse period}} \times 100\%$

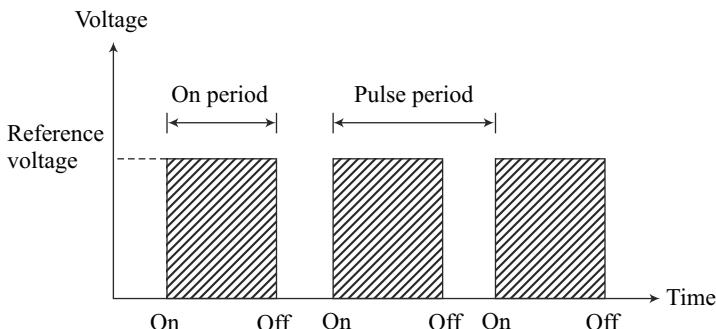
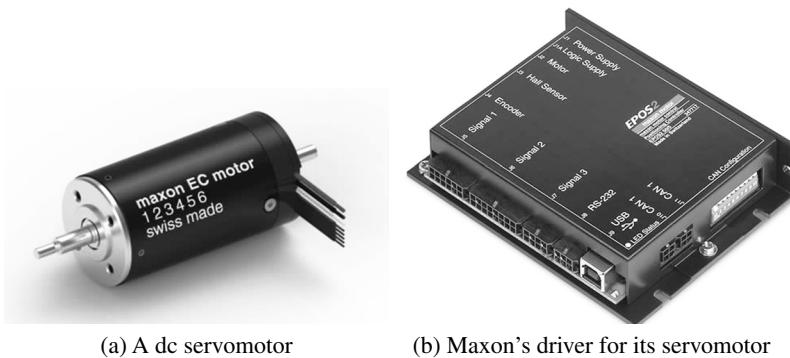


Fig. 3.16 Duty cycle of a PWM signal



[Courtesy: <http://www.maxonmotor.in/maxon/view/content/ec-max-motors>]

Fig. 3.17 A dc servomotor and its driver controller

Example 3.4 Stable and Unstable Operating Points of a dc Motor

Figure 3.18 shows the steady-state characteristic curve of a dc motor for a given load to be driven at constant power under steady-state operating condition using a separately wound dc motor with constant supply voltages to the field and armature windings. The constant power curve for the load is a hyperbola because $\tau\omega_m = \text{constant}$. The two curves intersect at *A* and *B*. At *A*, if there is a slight decrease in the speed of operation, the motor (magnetic) torque will increase to τ_m^A and the load torque demand will increase to τ_l^A . But since $\tau_m^A > \tau_l^A$, the system will accelerate back to the point *A*. This shows that the point *A* is stable. On the other hand, at *B*, if speed drops slightly, the magnetic torque of the motor will increase to τ_m^B and at the same time, the load torque demand will also increase to τ_l^B , where $\tau_m^B < \tau_l^B$. As a result, the system will decelerate further leading to stalling the motor. Hence, the point *B* is unstable.

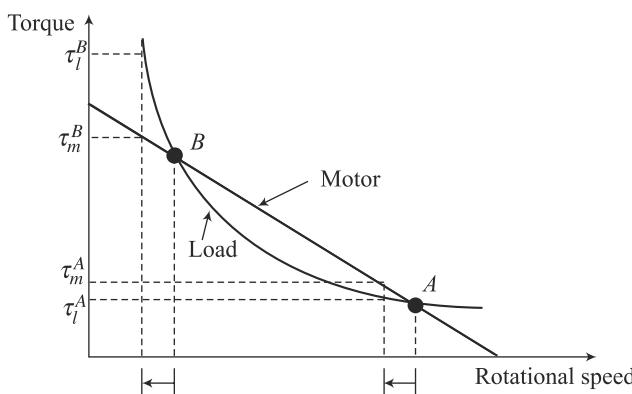


Fig. 3.18 Steady-state characteristic curve of a dc motor

3.1.3 ac Motors

Until recently, ac (alternating current) motors have not been considered suitable for robots because of the problems involved in controlling their speeds. In its simplest form, an ac motor consists of external electromagnets around a central rotor, but without any form of mechanical switching mechanism for the electromagnets. However, because alternating current (such as the mains electricity supply) is constantly changing polarity (first flowing one way, then the opposite way, several times a second, e.g., 50 in India, and 60 in the USA), it is possible to connect the ac supply directly to the electromagnets. The alternating reversal of the direction of current through the coils will perform the same task of polarity changing in ac motors. The magnetic field of the coils will appear to *rotate* (almost as if the coils themselves were being mechanically rotated).

A typical specification of an ac motor is given in Table 3.6. Moreover, speed-torque characteristics of ac motors are shown in Fig. 3.19. Typical advantages of an ac motor over its dc counterpart are listed below:

- Cheaper.
- Convenient power supply.
- No commutator and brush mechanism. Hence, virtually no electric spark generation or arcing (safe in hazardous environment like spray painting and others)
- Low power dissipation, and low rotor inertia.
- High reliability, robustness, easy maintenance, and long life.

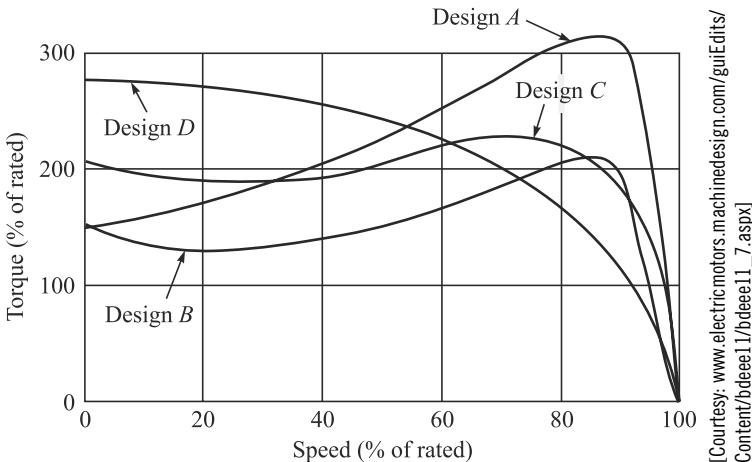
Some of the disadvantages are the following:

- Low starting torque.
- Need auxiliary devices to start the motor.
- Difficulty of variable-speed control or servo control unless modern solid-state and variable-frequency drives with field feedback compensation are used.

Table 3.6 Specifications of an ac motor [Courtesy: <http://uk.rs-online.com>]

Brand	ABB
Manufacturer part no.	1676687
Type	Industrial single and three-phase electric motors
Supply voltage	220 – 240 V _{ac} 50 Hz
Output power	180 W
Input current	0.783 A
Shaft diameter	14 mm
Shaft length	30 mm
Speed	1370 rpm
Rated torque	1.3 Nm
Torque starting	1.3 Nm
Height	150 mm
Length	213 mm
Width	120 mm

Due to the complexity in speed control of an ac motor, a speed-controlled dc drive generally works out cheaper than a speed-controlled ac drive, though the price difference is steadily dropping as a result of technological developments and the reduction in price of solid-state devices. A brief account of an ac servo motor is given at the end of this subsection.



[Courtesy: www.electricmotors.machinedesign.com/guEdits/Content/bdeee1/bdeee1_7.aspx]

Fig. 3.19 Typical speed-torque characteristics for four different designs of an ac induction motor

Alternating current (ac) motors can be classified into two groups, single-phase and poly-phase, with each group being further subdivided into *induction* or *asynchronous* and *synchronous* motors. Single-phase motors tend to be used for low power requirements while poly-phase motors are used for higher powers. Induction motors tend to be cheaper than synchronous motors and are thus very widely used.

1. Single-phase Squirrel-cage Induction Motor It consists of a squirrel-cage rotor, this being copper or aluminum bars that fit into slots in end rings to form complete electrical circuits, as shown in Fig. 3.20. There are no external electrical connections to the rotor. The basic motor consists of this rotor with a stator having a set of windings. When an alternating current passes through the stator windings, an alternative magnetic field is produced, which appears like a rotating magnetic field. The rotating field in the stator intercepts the rotating windings, thereby generating an induced current due to mutual induction or transformer action (hence, the name induction motor). The resulting secondary magnetic flux interacts with the primary, rotating magnetic flux, thereby producing a torque in the direction of rotation of the stator field. This torque drives the rotor. As the rotor speed increases, initially the motor torque also increases because of secondary interactions between the stator circuit and the rotor circuit even though the relative speed of the rotating field with respect to the rotor decreases, which reduces the rate of change of flux linkage and,

dc motor or ac motor?

The dc motors are easy to control compared to the ac motors. Hence, the former is preferred in robotic applications.

hence, the direct transformer action. Note that the relative speed is termed as slip rate given by

$$S = \frac{\omega_f - \omega_m}{\omega_f} \quad (3.4)$$

where ω_f and ω_m are the speed of the field and motor's rotor, respectively. Quite soon, the maximum torque will be reached. Further increase in the rotor speed (i.e., a decrease in slip rate) sharply decreases the motor torque, until at synchronous speed (zero slip rate), the motor torques becomes zero.

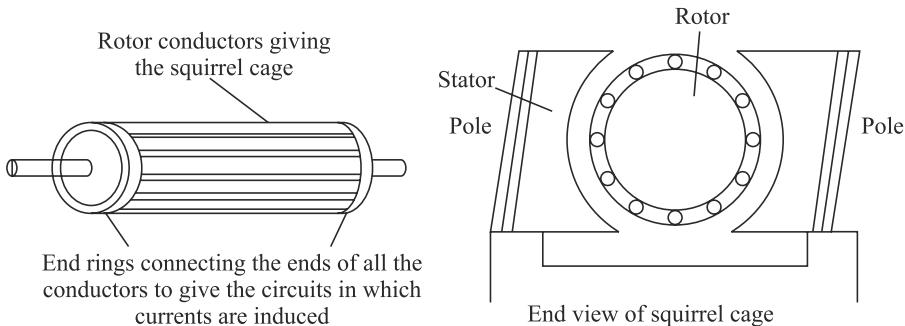


Fig. 3.20 Single-phase induction motor

For a single-phase supply, when the rotor is stationary initially, the forces on the current-carrying conductors or the rotor in the magnetic field of the stator are such as to result in no net torque. Hence, the motor is not self-starting. A number of methods are used to make the motor self-starting and give this initial impetus to start it. For example, to provide the starting torque, most single-phase motors have a main and auxiliary winding. The auxiliary winding current from the main winding is phase-shifted. Connecting a capacitor in series with the auxiliary winding causes the motor to start rotating. The rotor rotates at a speed determined by the frequency of the alternating current applied to the stator. For a constant frequency supply to a two-pole single-phase motor, the magnetic field will alternate at this frequency. This speed of rotation of the magnetic field is termed *synchronous speed*. The rotor will never quite match this frequency of rotation, typically differing from it by about 1 to 3%. For a 50 Hz supply, the speed of rotation of the rotor, i.e., ω_m , will be almost 50 revolutions per second.

Not Self-Starting of a Single-Phase Motor?

The stator winding of a single-phase motor can only produce the field (flux), which is only alternating but not rotating. Hence, it cannot produce any rotation.

2. Three-phase Induction Motor As shown in Fig. 3.21(a), it is similar to the single-phase induction motor but has a stator with three windings located 120° apart, each winding being connected to one of the three lines of the supply. Because the three phases reach their maximum currents at different times, the magnetic field

can be considered to rotate round the stator poles, completing one rotation in one full cycle of the current. The rotation of the field is much smoother than with the single-phase motor. The three-phase motor has a great advantage over the single-phase motor in being self-starting. The direction of rotation is reversed by interchanging any two of the line connections, thus changing the direction of rotation of the magnetic field.

3. Synchronous Motor A synchronous motor has a stator similar to those described above for induction motors, but a rotor is a permanent magnet as shown in Fig. 3.21(b). The magnetic field produced by the stator rotates and so the magnet rotates with it. With one pair of poles per phase of the supply, the magnetic field rotates through 360° in one cycle of the supply and so the frequency of rotation with this arrangement is the same as the frequency of the supply. Synchronous motors are used when a precise speed is required. They are not self-starting and some system has to be employed to start them.

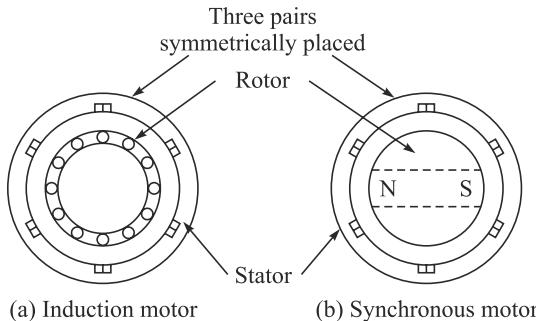


Fig. 3.21 An ac three-phase motor

4. ac Servomotor Generally, a modern servomotor refers to an ac permanent-magnet synchronous servomotor. It consists of a stator winding plus rotor with feedback units like encoders, resolvers, etc. These motors have typical advantages of ac motors as mentioned in the beginning of Section 3.1.3. Speed control of ac motors is based on the provision of a variable frequency supply, since the speed of such motors is determined by the frequency of the supply. In principle, the higher the frequency of the alternating current applied to the motor, the faster it will rotate. Providing varying frequency supplies to a number of axis drives simultaneously has been, until recently, largely impractical. In some special cases, e.g., wound-rotor ac induction motors, speed can be controlled by accessing the rotor circuit where different values of resistance can be inserted in the rotor circuit. Electromagnetic is used to provide regenerative braking to cut down the deceleration times, and minimize axis overrun. Many industrial robots, e.g., KUKA KR-5 of Fig. 2.1, use ac servomotors today. A typical ac servomotor is shown in Fig. 3.22, whereas its servo controllers look like those shown in Fig. 2.9(c) for the KUKA robot. Typical specification of an ac servomotor used is shown in Table 3.7.



[Courtesy: <http://motorsforautomationportal.info/kuka-robot-servo-motor/>]

Fig. 3.22 A typical ac servomotor

Table 3.7 Specifications of an ac servo motor [Courtesy: <http://uk.rs-online.com>]

Brand	Siemens
Manufacturer part no.	1FK7042-5AF21-1PA3
Type	ac synchronous servomotor
Supply voltage	230 V 50 Hz
Power rating	820 W
Maximum output torque	2.6 Nm
Output speed	3000 rpm
Shaft diameter	19 mm
Stall torque	3 Nm
Height	96 mm
Length	162 mm
Width	90 mm
Encoder system for motor	14-bit resolver (with DRIVE-CLiQ-interface)

3.1.4 Linear Actuators

Linear actuators like solenoids are used widely in robotic and other automation applications for on-off of the gripper and other devices. Electrically powered stepper and dc/ac linear actuators can also be used in motion generation of Cartesian robots, etc.

A solenoid shown in Fig. 3.23 has a coil and a soft-iron core. When the coil is activated by a dc signal, the soft core becomes magnetized. This electromagnet can serve as on-off (push-pull) actuator.

Solenoids are rugged and inexpensive devices. Common applications of solenoids are valve actuators (as explained in Section 3.2 and 3.3 for fluid-power actuators), mechanical switches, relays, etc. Most commonly generated linear

Backlash

Backlash is the term used to describe the unwanted play in transmission components from their intended position, when put under load conditions, due to wear or clearances between surfaces.

motions are with the help of an electrically powered rotary motor, as explained in sections 3.1.1–3.1.3, coupled with transmission mechanisms like nut and ball-screw, cam-follower, rack-and-pinion, etc. These devices inherently have problems of friction and backlash. Additionally, they add inertia and flexibility to the driven load, thereby generating undesirable resonances and motion errors. Proper inertia matching is also essential. In order to avoid the above difficulties of using a transmission system, direct rectilinear electromechanical actuators are desirable. They can be based on any of the principles mentioned for the rotary actuators, i.e., stepper or dc or ac motors. In these actuators, flat stators and rectilinearly moving elements (in place of rotors) are employed. These types of actuators are also referred to as electric cylinders in line with hydraulic or pneumatic cylinders that are explained next.

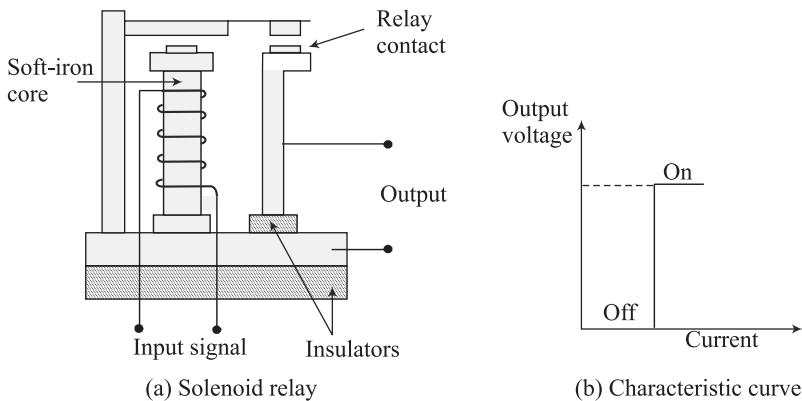


Fig. 3.23 A solenoid and its characteristic curve

3.2 HYDRAULIC ACTUATORS

Hydraulic actuators are one of the two types of fluid power devices for industrial robots. The other type is pneumatic, described in the next section. Hydraulic actuators utilize high-pressure fluid such as oil to transmit forces to the point of application desired. A hydraulic actuator is very similar in appearance to that of pneumatically driven one. Different components are shown in Fig. 3.24, whereas typical specifications of a cylinder are given in Table 3.8. Hydraulic actuators designed to operate at much higher pressures (typically between 70 and 170 bar). They are suitable for high power applications. Advantages and disadvantages of a hydraulic actuator are as follows:

Advantages

- High efficiency and high power-to-size ratio.
- Complete and accurate control over speed, position, and direction of actuators are possible.

What is 1 Bar?

It is a unit of pressure.
One Bar = one atm = 14.7 Psi = 101356.5 N/m^2 (Pa) or approximately 0.1 MPa.

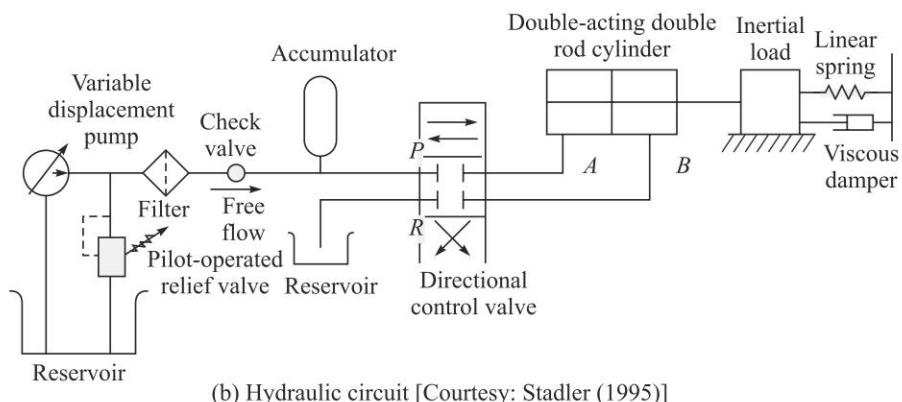
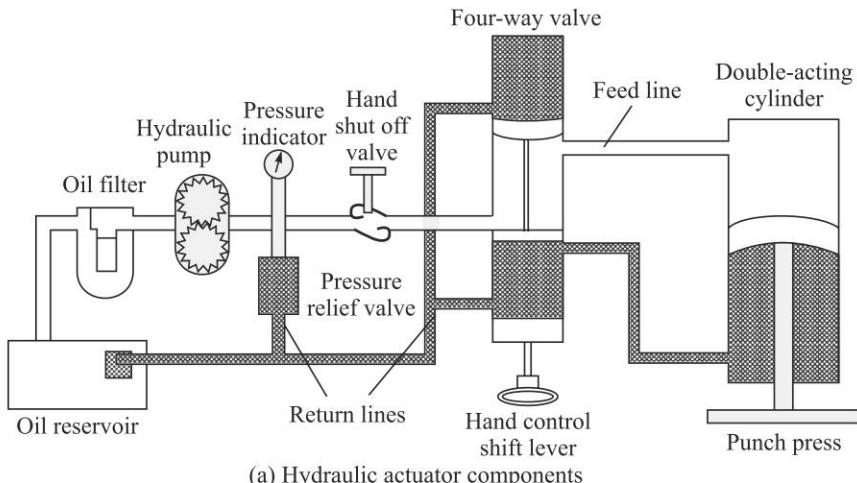


Fig. 3.24 A hydraulic actuator

- Few backlash problems occur due to the stiffness and incompressibility of the fluid, especially, when the actuator acts as the joint itself. Large forces can be applied directly at the required locations.
- They generally have a greater load-carrying capacity than electric and pneumatic robots.
- No mechanical linkage is required, i.e., a direct drive is obtained with mechanical simplicity.
- Self-lubricating (low wear) and non-corrosive.
- Due to the presence of an accumulator, which acts as a ‘storage’ device, the system can meet sudden demands in power.
- Hydraulic robots are more capable of withstanding shock loads than electric robots.

Disadvantages

- Leakages can occur to cause loss in performance, and general contamination of the work area. There is also higher fire risk.

- The power pack can be noisy, typically about 70 decibel (dBA) or louder if not protected by an acoustic muffler.
- Changes in temperature alter the viscosity of the hydraulic fluid. Thus, at low temperatures, fluid viscosity will increase, possibly, causing sluggish movement of the robot.
- For smaller robots, hydraulic power is usually not economically feasible as the cost of hydraulic components do not decrease in proportion to size.
- Servo control of hydraulic systems is complex and is not as widely understood as electric servo control.

How noisy is 70 decibels?

The level of 70 decibels is about the noise level of heavy traffic.

Hydraulic systems power the strongest and the stiffest robots and, hence, the bulk modulus of the oil is an extremely important attribute to be selected. A high bulk modulus implies a stiff, quickly responding system with a corresponding quick pressure build-up, while a low bulk modulus may result in a system that is too loose because of the high compressibility of the oil. Hydraulic systems or circuits have always four essential components: a reservoir to hold the fluid, pumps to move it, valves to control the flow, and an actuator to carry out the dictates of the fluid on some load. Rotary hydraulic actuators are also available in the market. A commercially available hydraulic cylinder is shown in Fig. 3.25(a), whereas its design using software is available in WR-Hydraulic (2013). A video of such cylinders is also available at WR-Hydraulic-video (2013).

Table 3.8 Specifications of a hydraulic cylinder [Courtesy: <http://uk.rs-online.com>]

Brand	Rexroth
Manufacturer part no.	CDL1M00/32/18/50/C1X/B1CHUMS
Type	Cylinder-rod ends bearing
Body	Carbon steel
Bore size	32 mm
Stroke	50 mm
Port size	G 1/4
Static proof pressure	240 bar
Working pressure	160 bar
Maximum force at 160 bar	12.8 kN
Maximum flow at 0.1 mm/s	4.8 l/min
Overall length	200 mm
Overall length (piston extended)	250 mm
Rod-end diameter	18 mm
Rod-end thread size	18 mm
Seal material	NBR/Polyurethane
Recommended hydraulic medium	Mineral-oil-based fluids
Hydraulic fluid operating temperature range	-20 to +80°C



(a) Hydraulic cylinder

[Courtesy: www.meritindustriesltd.com]

(b) Pneumatic cylinder

[Courtesy: www.festo.com]**Fig. 3.25** Commercially available cylinders

3.3 PNEUMATIC ACTUATORS

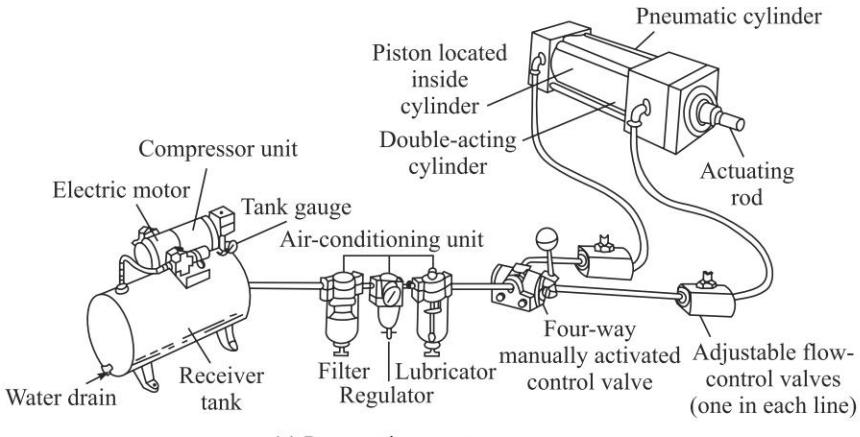
Pneumatic actuators are the other type of fluid power devices for industrial robots. Pneumatic actuators utilize compressed air for the actuation of cylinders as shown in Fig. 3.25 (b), and are widely used for typical opening and closing motions of jaws in the gripper of a robot, as shown in Fig. 2.4 (a), or for the actuation of simple robot arms used in applications where continuous motion control is not of concern. A pneumatic actuator comprising of a pneumatic cylinder and other accessories are shown in Fig. 3.26, whereas the specifications of a cylinder are given in Table 3.9. Typical advantages and disadvantages of pneumatic actuators are as follows:

Advantages

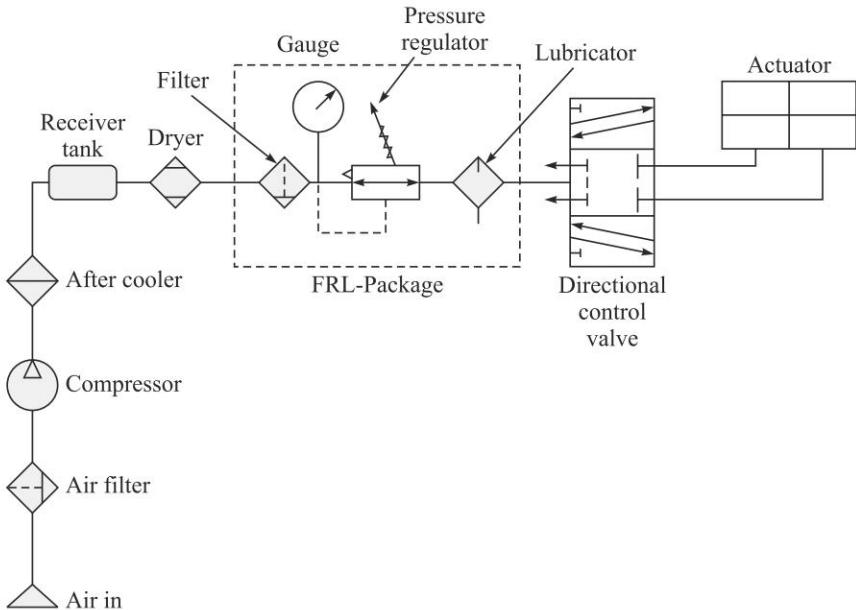
- It is the cheapest form of all actuators. Components are readily available and compressed air is normally an already existing facility in factories.
- Compressed air can be stored and conveyed easily over long distances.
- Compressed air is clean, explosion-proof and insensitive to temperature fluctuations, thus, lending itself to many applications.
- They have few moving parts making them inherently reliable and reducing maintenance costs.
- Since pneumatic systems are common throughout industry, therefore, relevant personnel are often very familiar with the technology.
- Very quick in action and response time, thus, allowing fast work cycles.
- No mechanical transmission is usually required.
- Pneumatics can be intrinsically safe in explosive areas as no electrical control is required. Also in wet conditions there is no danger of electrocution.
- The systems are usually compact.
- Control is simple, e.g., mechanical stops are often used.
- Individual components can be easily interconnected.

Disadvantages

- Since air is compressible, precise control of speed and position is not easily obtainable unless more complex electromechanical devices are incorporated into the system. This means that only a limited sequence of operation at a fixed speed is often available.



(a) Pneumatic actuator components



(b) Pneumatic circuit [Courtesy: Stadler (1995)]

Fig. 3.26 A pneumatic actuator

- If mechanical stops are used, resetting the system can be slow.
- Pneumatics is not suitable for moving heavy loads under precise control due to the compressibility of air. This compressibility necessitates the application of more force than would normally be necessary to ensure that the actuator is firmly in position against its stop under load conditions.
- If moisture penetrates the units and ferrous metals have been used then damage to individual components can result.

The compressibility of the air in a pneumatic actuator does not allow for sophisticated control, but the same can be used to advantage, preventing damage

due to overload and providing compliance that may be required in many practical applications.

Table 3.9 Specifications of a pneumatic cylinder for a gripper
 [Courtesy: <http://uk.rs-online.com>]

Brand	SMC
Manufacturer part no.	MHC2-25D
Type	Gripper (double acting)
Bore	25 mm
Operating angle	30° (open) to -10° (Closed)
Pressure range	1 to 6 Bar
Connection port size	M5 m
Temperature range	0 to 60°C
Media	Non lubricated air
Holding moment	1.4 Nm at 5 bar
Repeatability at closing position	±0.1 mm
Maximum operating frequency	80 cpm (cycles per minute)

3.4 SELECTION OF MOTORS

For any application of robots, one must decide which of available actuators is most suitable. Positioning accuracy, reliability, speed of operation, cost, and other factors must be considered. Electric motors are inherently clean and capable of high precision if operated properly. In contrast, pneumatic systems are not capable of high precision for continuous-path operation and hydraulic actuators require the use of oil under pressure. Hydraulic system can generate greater power in a compact volume than electric motors. Oil under pressure can be piped to simple actuators capable of extremely high torque and rapid operations. Also, the power required to control an electro-hydraulic valve is small. Essentially, the work is done in compressing the oil and delivering it to the robot-arm drives. All the power can be supplied by one powerful, efficient electric motor driving the hydraulic pump at the base of the robot or located at some distance away. Power is controlled in compact electro-hydraulic valves. However, high-precision electro-hydraulic valves are more expensive and less reliable than low-power electric amplifiers and controllers. On the other hand, electric motors must have individual controls capable of controlling their power. In large robots, this requires switching of 10 to 50 amperes at 20 to 100 volts. Current switching must be done rapidly; otherwise there is large power dissipation in the switching circuit that will cause excessive heat. Small electric motors use simple switching circuits and are easy to control with low-power circuits. Stepper motors are especially simple for open-loop operation. The single biggest advantage of a hydraulic system is their intrinsically safe operation.

As a rule of thumb, hydraulic actuators are preferred where rapid movement at high torques is required, at power ranges of approximately over 3.5 kW unless the slight possibility of an oil leakage cannot be tolerated. Electric motors are preferred

at power levels under about 1.5 kW unless there is danger due to possible ignition of explosive materials. At ranges between 1-5 kW, the availability of a robot in a particular coordinate system with specific characteristics or at a lower cost may determine the decision. Reliability of all types of robots made by reputable manufacturers is sufficiently good that this is not a major determining factor.

Example 3.5 Selection of a Motor

Simple mathematical calculations are needed to determine the torque, velocity, and power characteristics of an actuator or a motor for different applications. Torque is defined in terms of force times distance or moment. A force, f , at distance, a , from the center of rotation has a moment or torque, τ , i.e., $\tau = fa$. In general terms, power, P , transmitted in a drive shaft is determined by the torque, τ , multiplied by the angular velocity, ω . Power P is expressed as, $P = \tau \omega$. For an example, a calculation can tell one what kilowatt or horsepower is required in a motor used to drive a 2-meter robot arm lifting a 25 kg mass at 10 rpm. If the mass of the arm is assumed zero then, $P = (25 \times 9.81 \times 2) \times (2\pi \times 10/60) = 0.513$ kW. The use of simple equations of this type is often sufficient to make a useful approximation of a needed value. More detailed calculations can take place using the equations of statics and dynamics that apply.

3.5 GRIPPERS

Grippers are end-effectors, as introduced in Section 2.1.1, which are used to grasp an object or a tool, e.g., a grinder, and hold it. Tasks required by the grippers are to hold workpieces and load/unload from/to a machine or conveyer. Grippers can be mechanical in nature using a combination of mechanisms driven by electric, hydraulic, or pneumatic powers, as explained in earlier sections. Grippers can be classified based on the principle of grasping mechanism. For example, grippers can hold with the help of suction cups, magnets, or by other means. A gripper is then accordingly referred to as *pneumatic gripper*, *magnetic gripper*, etc. Another way to classify a gripper is based on how it holds an object, i.e., based on grasping the object on its exterior (*external gripper*) or interior (*internal gripper*) surface.

3.5.1 Mechanical Grippers

As shown in Fig. 2.4(a), mechanical grippers have their jaw movements through pivoting or translational motion using a transmission element, e.g., linkages or gears, etc. This is illustrated in Fig. 3.27. The gripper can be of single or double type. While the former has only one gripping device at the robot's wrist, the latter type has two. The double grippers can be actuated independently and are especially useful in machine loading and unloading. As illustrated in Groover et al. (2012), suppose a particular job calls for a raw part to be loaded from a conveyor onto a machine and the finished part to be unloaded onto another conveyor. With a single gripper, the robot would have to unload the finished part before picking up the raw part. This would consume valuable time in the production cycle because the machine would remain idle during these handling motions. With a double gripper, the robot can pick up the part from the incoming conveyor with one of its gripping devices and have it ready to exchange for the finished part on the machine. When the machine cycle is completed, the robot can reach in for the finished part with the available grasping

device, and insert the raw part into the machine with the other grasping device. The amount of time spent in exchanging the parts or the time to keep the machine idle is minimized.

A gripper uses its fingers or jaws to hold an object, as illustrated in Fig. 3.28. The function of a gripper mechanism is to translate some form of power input, be it electric, hydraulic or pneumatic, into the grasping action of the fingers against the part. Note that there are two ways a gripper can hold an object, i.e., either by physical constriction as shown in Fig. 3.28(a) or by friction, as demonstrated in Fig. 3.28(b). In the former case, contacting surfaces of the fingers are made of approximately the same shape of the part geometry, while in the latter case the fingers must apply sufficient force to retain the part against gravity or accelerations. The friction method of holding a part is less complex and hence less expensive. However, they have to be designed properly with its surfaces having sufficient coefficient of friction so that the parts do not slip during motion. Example 3.6 illustrates the situation.

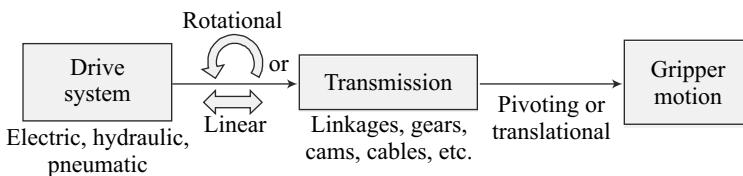


Fig. 3.27 Motion of a mechanical gripper

Example 3.6 Friction-based Gripper

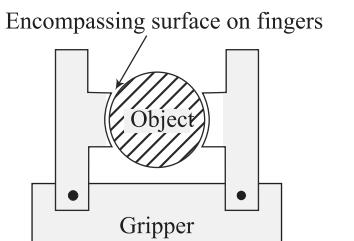
Consider the weight w of an object to be carried by a parallel-fingered gripper shown in Fig. 3.28(b). Gripper force can be calculated using the following force balance:

$$\mu n f = w \quad (3.5a)$$

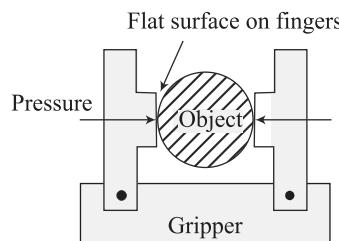
where μ is the coefficient of friction of the object and finger surfaces, whereas n , f , and w are the number of contacting surfaces, finger forces, and the weight of the object to be held by the gripper. If $w = 140$ kg, $n = 2$, and $\mu = 0.2$, f is computed simply as

$$f = \frac{140 \times 9.81}{2 \times 0.2} \cong 3500 \text{ N} \quad (3.5b)$$

Note that the value of 9.81 m/s^2 is the value of g , i.e., acceleration due to gravity.



(a) Physical or encompassing



(b) Friction load

Fig. 3.28 Fingers gripping objects

Example 3.7 Gripper Accelerating with an Object

Assume the object of Example 3.6 is to be lifted up with an acceleration of 10 m/s^2 . The gripper will require almost twice the force as the acceleration to be experienced by it is $(9.81 + 10) = 19.81 \cong 2g$. Rest of the calculation is same as in equation (3.5a-b).

Note that the motion of the fingers in mechanical grippers is obtained by means of any of the following transmission elements: (i) Linkage; (ii) Gear and rack; (iii) Cam; (iv) Screw; and (v) Cable and pulley, etc. Figure 3.29 shows some of these arrangements, where the transmission elements are either driven by an electric motor or a hydraulic/ pneumatic actuator. The final gripping action of the fingers is, however, achieved by one of the following means:

1. Pivoting Movement In this arrangement, the fingers rotate about fixed pivot points on the gripper to open and close. This is indicated in Fig. 3.29(a). The motion is usually achieved by some kind of linkage mechanism.

2. Translational Movement In the translational or linear motion, the fingers open and close by moving parallel to each other, as shown in Fig. 3.29 (b-d).

It is interesting to note that in the literature, e.g., in Groover et al. (2012), Deb and Deb (2010), and others, the mechanical grippers were classified as per either the transmission elements used or the type of finger movements, besides the input power, i.e., electric or pneumatic. For example, cam-operated gripper or pivoting gripper, etc. In the true sense, they should fall in the specifications of a mechanical gripper. Moreover, mechanical grippers with more fingers are also in use in industries, e.g., to hold spherical objects. One such robotic gripper, namely, a three-fingered hand, is shown in Fig. 2.4(b).

3.5.2 Magnetic Grippers

Unlike mechanical grippers, the principle of a magnetic gripper is based on the magnetic property of a gripper. Hence, they can be used only for ferrous objects. They have the following advantages:

- Variations in object sizes can be tolerated.
- Operations are very fast.
- Require only one surface to hold an object.

The disadvantages with magnetic grippers are, however, the difficulty to pick thin sheets one at a time because the magnetic force penetrates through more than one sheet. As a result, more than one sheet is picked up. To overcome such disadvantages, one needs to take care during the design stage itself either by limiting the magnetic force by the gripper or by introducing some means (mechanical or otherwise) not to allow more than one sheet to be picked up.

Magnetic grippers can have either (i) permanent magnets, or (ii) electromagnets. While electromagnetic grippers are easy to control requiring only a dc power source, grippers with permanent magnets do not require any external power source to operate the magnets. Besides, no electric sparks in handling hazardous materials. They, however, require an external stripping mechanism during the release of the object. One such mechanism is illustrated in Fig. 3.30.

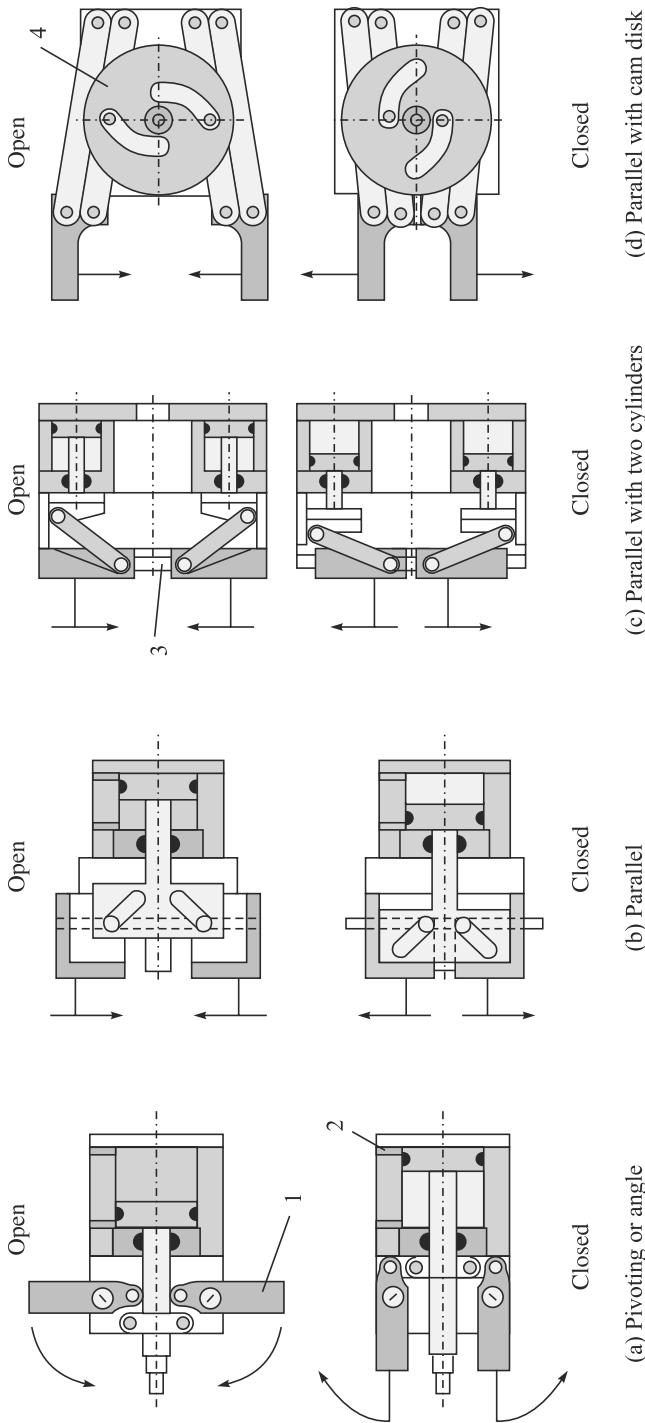


Fig. 3.29 Pivoting and translating mechanical grippers

[Courtesy: http://media.wiley.com/product_data/excerpt/90/35274061/3527406190.pdf]

1: Finger or jaw; 2: Pneumatic cylinder; 3: Straight guideway; 4: Cam disk

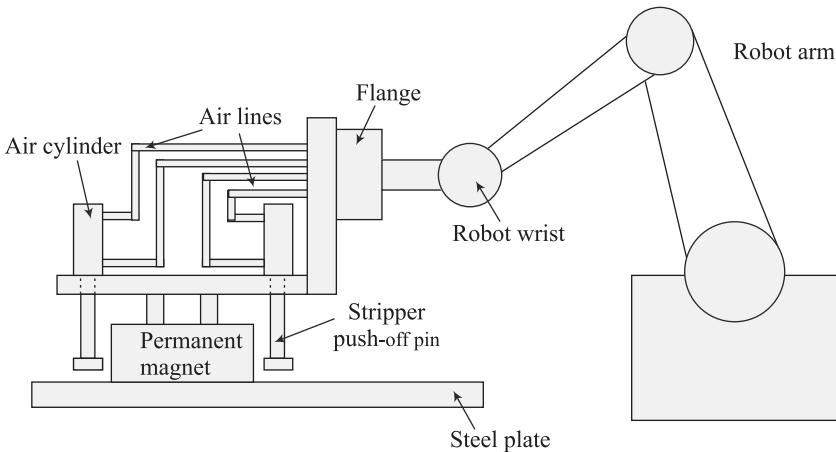


Fig. 3.30 Stripping of a sheet from a magnetic gripper

3.5.3 Vacuum Grippers

Such grippers are suitable to handle large flat objects. The material of an object is of no concern with vacuum grippers, except that the object's surface should not have any holes. An example of vacuum gripper which uses suction cups made of elastic materials is shown in Fig. 3.31. For a vacuum gripper, lifting capacity can be determined from the negative pressure and the effective area of the cups as

$$f = pA \quad (3.6)$$

where f is the force or lift capacity, p is the negative pressure, and A is the total effective area of the suction cups.

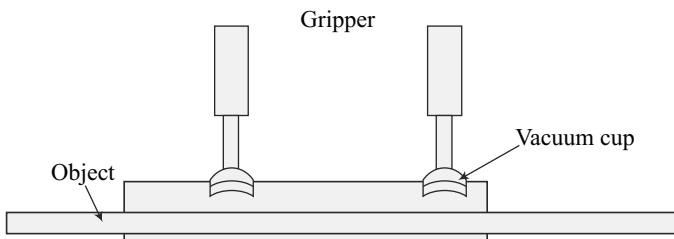


Fig. 3.31 Vacuum gripper

Example 3.8 Lifting Capacity

Consider lifting of four steel plates, each piece is 6 mm thick and measures $6\text{ m} \times 10\text{ m}$. The weight of the four steel plates is given by

$$w = 4 \times (0.006 \times 6 \times 10) \times 7800 \times 9.81 = 110187.12 \text{ N}$$

where 7800 kg/m^3 is the mass density of the steel plates, whereas 9.81 m/s^2 is the acceleration due to gravity. Assuming the diameter of vacuum cups as 100 mm, the required negative pressure can be obtained from Eq. (3.6) as

$$p = \frac{110187.12}{2 \times \pi \times (0.1)^2} = 175.35 \text{ N/m}^2 \quad (3.7)$$

3.5.4 Adhesive Grippers

An adhesive substance used for a grasping action can be used to handle fabrics and other lightweight materials. One of the limitations is that the adhesive substance loses its effectiveness with repeated use. Hence, it has to be continuously fed like a mechanical typewriter's ribbon which needs to be attached to the robot's wrist.

3.5.5 Hooks, Scoops, and Others

There exist other types of gripping devices, e.g., hooks, scoops or ladles, inflatable devices, etc., based on the need of item to be handled. For example, Fig. 3.32 shows one of these devices.

3.5.6 Selection of Grippers

Some of the criteria to be used in selecting an appropriate gripper are highlighted below:

- One needs to decide the drive system. In fact, in an industrial scenario where electric and pneumatic power sources are easily available, one needs to decide the type. If fast but not so accurate motion is desired then pneumatic should be preferred. Otherwise, one can go for electric. For heavy objects, certainly one must prefer hydraulic, as pointed in Section 3.2.
- Object or the part surface to be gripped must be reachable by a gripper.
- The size variation should be kept in mind. For example, a cast object may be difficult to put in a machine chuck.
- The gripper should tolerate some dimension change. For example, before and after machining a workpiece, their sizes change.
- Quality of surface area must be kept in mind. For example, a mechanical gripper may damage the surface area of an object.
- Force should be sufficient to hold an object and should not fly away while moving with certain accelerations. Accordingly, the materials on the surfaces of the fingers should be chosen.

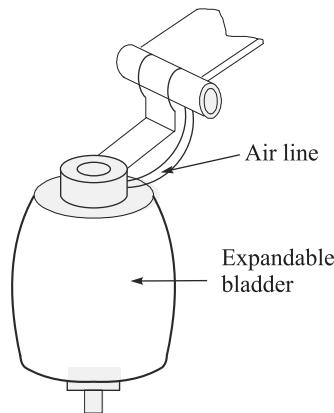


Fig. 3.32 Expandable bladder to grip objects with holes from inside

SUMMARY

A robot is moved by actuators. Different forms of actuators, namely, electric, hydraulic and pneumatic types are explained. For electric motors, their typical specifications and speed-torque characteristics are presented. How to select a suitable motor is also outlined. Besides, grippers used to hold an object in order to transfer it to a new location are explained. Important aspects to select an appropriate gripper are also highlighted.

EXERCISES

- 3.1** Name the components of a joint actuating system.
- 3.2** What is an actuator?
- 3.3** Give the requirements of an actuator for robotic applications.
- 3.4** What are different actuators?
- 3.5** List the advantages and disadvantages of electric motors.
- 3.6** What are the types of stepper motors?
- 3.7** Why are dc motors preferred as servomotors?
- 3.8** Describe the functional differences of steppers, dc, and ac motors.
- 3.9** Draw a typical speed-torque plot of a dc motor.
- 3.10** Distinguish a stable operating point from the unstable one of a dc motor.
- 3.11** When are hydraulic actuators preferred in a robot system?
- 3.12** What fluid is used in hydraulic actuators and what is the pressure range?
- 3.13** State different types of hydraulic actuators.
- 3.14** What are the risks of using hydraulic actuators?
- 3.15** What is the function of direction control valve?
- 3.16** List any three advantages and disadvantages of a pneumatic actuator.
- 3.17** State the typical applications of pneumatic actuators.
- 3.18** Why are pneumatic actuators preferred in factory set-up robots?
- 3.19** Why is a storage tank necessary in pneumatic actuators?
- 3.20** Which component basically provides the mechanical motion?
- 3.21** How is a motor selected?
- 3.22** What is a mechanical gripper? Show a sketch to hold a cylindrical object.
- 3.23** What are the limitations of friction-based grippes?
- 3.24** State one example for the use of a hook.
- 3.25** What is the main drawback of using adhesive gripper?

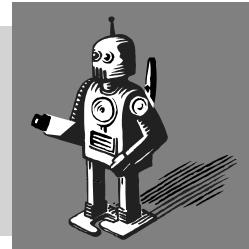
WEB-BASED EXERCISES

Based on Web search, find the answers to the following questions:

- 3.26** Find at least three manufacturers of dc and ac electric motors. What are their prices?
- 3.27** What is a typical value of the dc motor constant?
- 3.28** Find out if there is any software that can provide motor specifications based on the load and speed requirement of a robot. Name it.
- 3.29** Find out two companies who make direct-drive motors.
- 3.30** Name three companies who make both pneumatic and hydraulic actuators. What are the prices for these actuators?

4

Sensors, Vision and Signal Conditioning



Sensors in robots are like our nose, ears, mouth, and skin, whereas vision or robot vision can be thought of as our eyes. Based on the function of human organs, for example, eyes, skin, etc., terminology like vision, tactile, etc., have cropped up in robot sensors. Robots, like humans, must gather extensive information about their environment in order to function effectively. They must pick up an object and know it has been picked up. As the robot arm moves through the 3-dimensional Cartesian space, it must avoid obstacles and approach items to be handled at a controlled speed. Some objects are heavy, others are fragile, and others are too hot to handle. These characteristics of objects and the environment must be recognized, and fed to the computer that controls a robot's movement. For example, to move the end-effector of a robot, with or without a payload along a desired trajectory and to exert a desired force on an object, the end-effector and sensors (normally located at the joints and at the end-effector) work in coordination with the robot controller (a microprocessor, or a computer or a microcontroller).

Note that in industrial applications of robots, as explained in Chapter 2, sensors must play important roles. For example, they must at least provide the following functions:

1. Safe Operation Sensors must protect human workers who work in the vicinity of the robot or other equipment. For example, one can provide a sensor on the floor of a work cell where a robot is working so that if anybody steps in the robot's power should be switched off.

2. Interlocking This is required to coordinate the sequence of operations on a component. For example, unless a turning is done on a component, it should not be transferred to the conveyor.

3. Inspection This is essential for quality control. For example, one can use a vision system to measure the length of a component to check if it is within the acceptable tolerance or not.

4. Part Configuration If a robot is used to weld two parts of a car body, the sensors must identify the correct

Sensor vs. Transducer

Sensor and *transducer* are used interchangeably to denote a sensor-transducer unit. They are the functional stages of sensing. First, a *measurand* is *felt* or *sensed* before it is transduced or converted from one type of energy to another for various purposes including the measurement of physical parameters like position, velocity, etc.

configurations, i.e., position and orientation of the parts, before the robot starts welding.

There could be other scenarios like identifying the color code of a particular car model before painting with that color is done by the robot, etc.

4.1 SENSOR CLASSIFICATION

The major capabilities required by a robot are as follows:

Simple Touch The presence or absence of an object.

Taction or Complex Touch The presence of an object plus some information on its size and shape.

Simple Force Measured force along a single axis.

Complex Force Measured force along two or more axes.

Proximity Noncontact detection of an object.

Simple Vision Detection of edges, holes, corners, and so on.

Complex Vision Recognition of shapes.

Based on the type of signals a sensor or transducer receives and processes, it can be classified as *analog* or *digital*. In analog sensors, with the variation of input there is a continuous variation of output, whereas in case of digital sensors, the output is of digital or discrete nature. For example, potentiometers, tacho-generators located at the joints and strain-gauge-based sensors located at the end-effector of a robot fall in the category of analog sensors, whereas encoders, located at the robot's joints, are digital sensors. In this book, sensors are, however, classified based on what they sense, i.e., internal or external state of the robots, etc., as shown Fig. 4.1.

4.2 INTERNAL SENSORS

Internal sensors, as the name suggests, are used to measure internal state of a robot, i.e., its position, velocity, acceleration, etc., at a particular instant. Based on these information, control command is decided by the controller. Depending on the quantities it measures, a sensor is termed as the position, velocity, acceleration, or force sensor.

4.2.1 Position Sensors

Position sensors measure the position of each joint, i.e., joint angle of a robot. From these joint angles, one can find the end-effector configuration, namely, its position and orientation, through forward kinematics which will be taken up in Chapter 6. Different position sensors are explained next.

1. Encoder The encoder is a digital optical device that converts motion into a sequence of digital pulses. By counting a single bit or by decoding a set of bits, the pulses can be converted to relative or absolute measurements. Thus, encoders are of *incremental* or *absolute* type. Further, each type may be again *linear* and *rotary*.

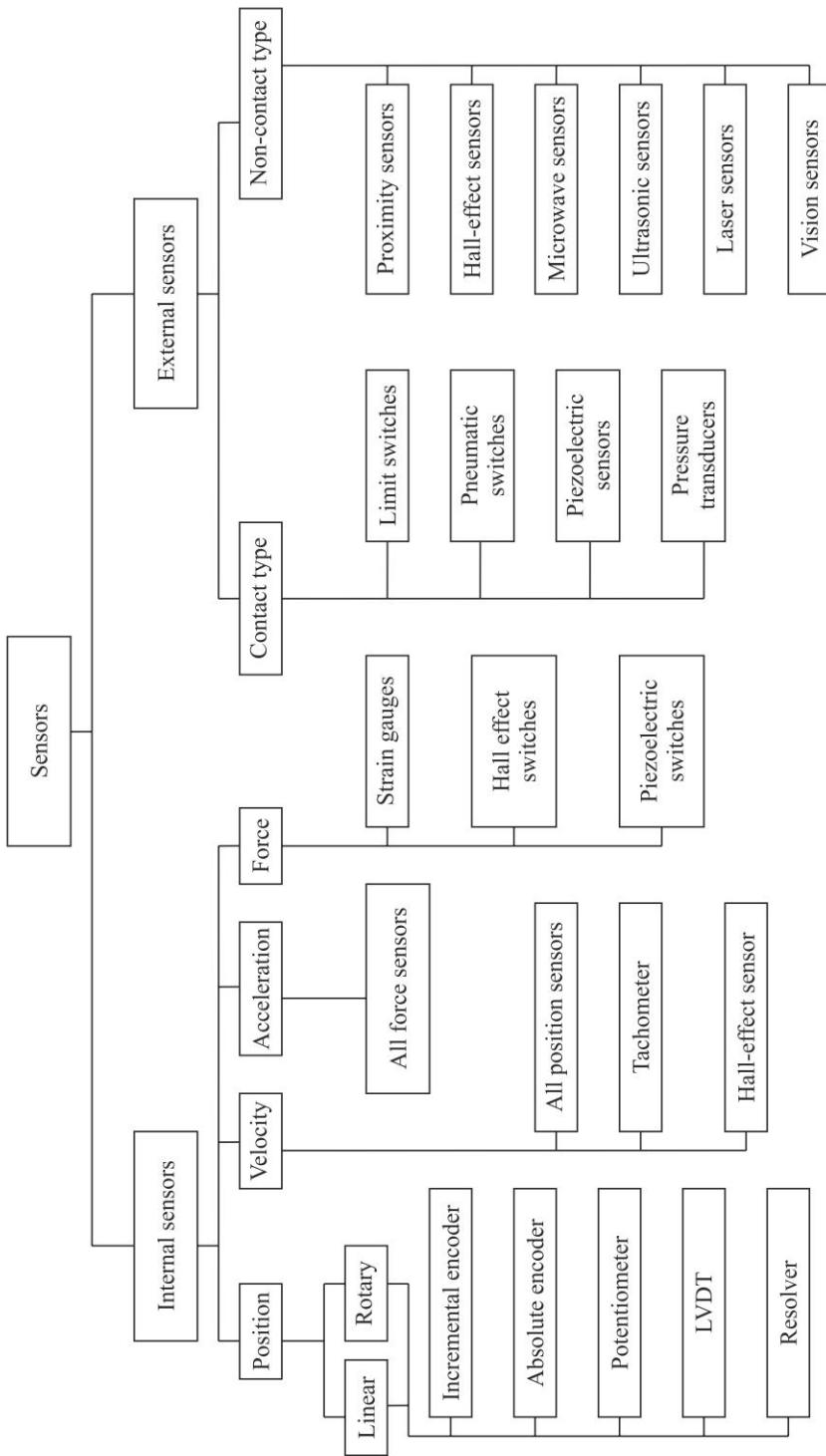


Fig. 4.1 Classification of sensors

Incremental Linear Encoder As shown in Fig. 4.2(a), it has a transparent glass scale with opaque grating. The thickness of grating lines and the gap between them is made same, which are in the range of microns. One side of the scale is provided with a light source and a condenser lens. On the other side there are light-sensitive cells. The resistance of the cells (photodiodes) decreases whenever a beam of light falls on them. Thus, a pulse is generated each time a beam of light is intersected by the opaque line. This pulse is fed to the controller, which updates a counter (a record of the distance traveled).

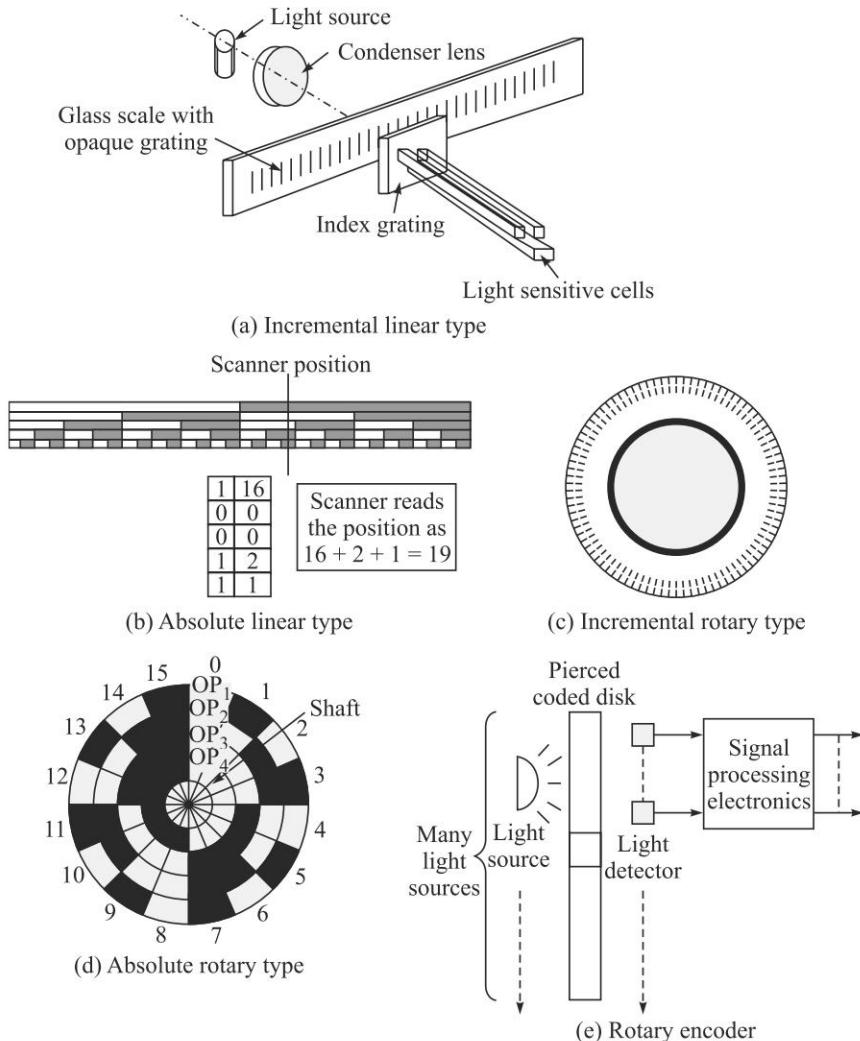
Absolute Linear Encoder It is similar in principle as the incremental linear encoder. The difference is that it gives absolute value of the distance covered at any time. Thus, the chance of missing the pulses at high speeds is less. The output is digital in this case. The scale is marked in a sequence of opaque and transparent strips, as shown in Fig. 4.2(b). In the scale shown, if the opaque block represents 1 (one) and the transparent block as 0 (zero) then the leftmost column will show a binary number as 00000, i.e., a decimal value of 0, and the rightmost column will show a binary number 11111, i.e., a decimal value of 61.

Incremental Rotary Encoder It is similar to the linear incremental encoder with a difference that the gratings are now on a circular disc, as in Fig. 4.2(c). The common value of the width of transparent spaces is 20 microns. There are two sets of grating lines on two different circles which detect direction of rotation, and one can also enhance the accuracy of the sensor. There is another circle, which contains only one grating mark. It is used for measurement of full circles.

Absolute Rotary Encoder Similar to the absolute linear encoder, the circular disk is divided into a number of circular strips and each strip has definite arc segments, as shown in Fig. 4.2(d). This sensor directly gives the digital output (absolute). The encoder is directly mounted on the motor shaft or with some gearing to enhance the accuracy of measurement. To avoid noise in this encoder, a gray scale is sometimes used. A Gray code, unlike binary codes, allows only one of the binary bits in a code sequence to change between radial lines. It prevents confusing changes in the binary output of the absolute encoder when the encoder oscillates between points. A sample Gray code is given in Table 4.1 for some numbers. Note the difference between the Gray and binary codes. The basic arrangement of the rotary encoder is shown in Fig. 4.2(e).

Table 4.1 Sample Gray codes

Decimal	Binary code	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110

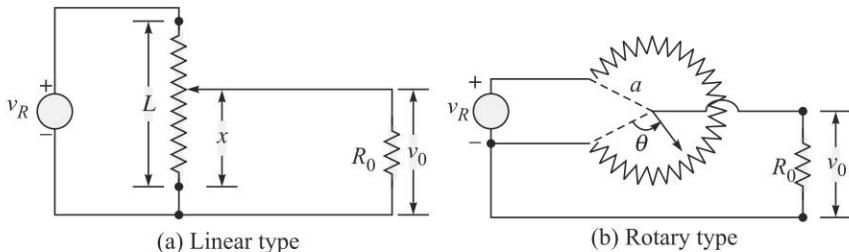
**Fig. 4.2** Encoders**Example 4.1** Angular Position of an Encoder

If maximum count possible is p pulses and the range of a rotary encoder is $\pm\theta_m$, then the angular position corresponding to a count of n pulses is given as follows:

$$\theta = \frac{n}{p} \theta_m \quad (4.1)$$

2. Potentiometer A potentiometer, also referred as simply *pot*, is a variable-resistance device that expresses linear or angular displacements in terms of voltage, as shown in Figs. 4.3(a-b), respectively. It consists of a wiper that makes contact with a resistive element, and as this point of contact moves, the resistance between the

wiper and end leads of the device changes in proportion to the displacement, x and θ for linear and angular potentiometers, respectively.



v_R : Reference voltage; v_0 : Measured voltage

L, x, R_0, a, θ : Other physical parameters

Fig. 4.3 Potentiometers

3. LVDT The Linear Variable Differential Transformer (LVDT) is one of the most used displacement transducers, particularly when high accuracy is needed. It generates an ac signal whose magnitude is related to the displacement of a moving core, as indicated in Fig. 4.4. The basic concept is that of a ferrous core moving in a magnetic field, the field being produced in a manner similar to that of a standard transformer. There is a central core surrounded by two identical secondary coils and a primary coil, as shown in Fig. 4.4. As the core changes position with respect to the coils, it changes the magnetic field, and hence the voltage amplitude in the secondary coil changes as a linear function of the core displacement over a considerable segment. A Rotary Variable Differential Transformer (RVDT) operates under the same principle as the LVDT is also available with a range of approximately $\pm 40^\circ$.

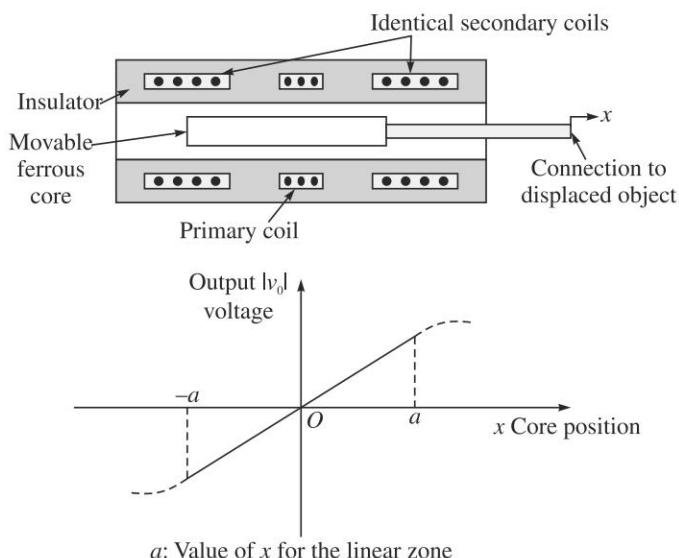


Fig. 4.4 Principle of LVDT

4. Synchros and Resolver While encoders give digital output, synchros and resolvers provide analog signal as their output. They consist of a rotating shaft (rotor) and a stationary housing (stator). Their signals must be converted into the digital form through an analog-to-digital converter before the signal is fed to the computer.

As illustrated in Fig. 4.5, synchros and resolvers employ single-winding rotors that revolve inside fixed stators. In a simple synchro, the stator has three windings oriented 120° apart and electrically connected in a Y-connection. Resolvers differ from synchros in that their stators have only two windings oriented at 90°. Because synchros have three stator coils in a 120° orientation, they are more difficult than resolvers to manufacture and are, therefore, more costly.

Modern resolvers, in contrast, are available in a brushless form that employ a transformer to couple the rotor signals from the stator to the rotor. The primary winding of this transformer resides on the stator, and the secondary on the rotor. Other resolvers use more traditional brushes or slip rings to couple the signal into the rotor winding. Brushless resolvers are more rugged than synchros because there are no brushes to break or dislodge, and the life of a brushless resolver is limited only by its bearings. Most resolvers are specified to work over 2 V to 40 V rms (root mean square) and at frequencies from 400 Hz to 10 kHz. Angular accuracies range from 5 arc-minutes to 0.5 arc-minutes.

What are arc-minute and arc-second?

They are the measures of small angles.
one degree = 60 arc-minutes and
one arc-minute = 60 arc-seconds.

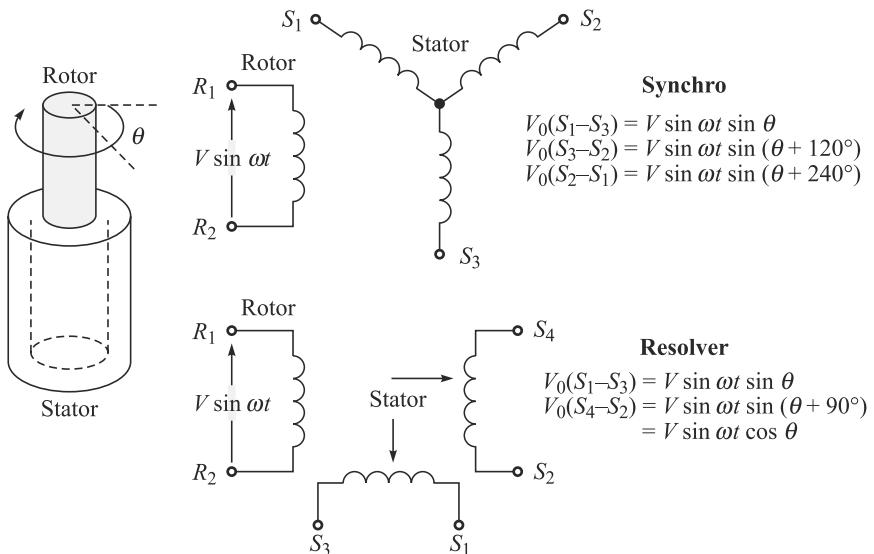


Fig. 4.5 Synchros and resolvers

In operation, synchros and resolvers resemble rotating transformers. The rotor winding is excited by an ac reference voltage, at frequencies up to a few kHz. The magnitude of the voltage induced in any stator winding is proportional to the sine

of the angle θ between the rotor-coil axis and the stator-coil axis. In the case of a synchro, the voltage induced across any pair of stator terminals will be the vector sum of the voltages across the two connected coils. For example, if the rotor of a synchro is excited with a reference voltage, $V \sin(\omega t)$, across its terminal R_1 and R_2 , the stator's terminal will see voltages denoted as V_0 in the form:

$$V_0(S_1 - S_3) = V \sin(\omega t) \sin \theta \quad (4.2a)$$

$$V_0(S_3 - S_2) = V \sin(\omega t) \sin(\theta + 120^\circ) \quad (4.2b)$$

$$V_0(S_2 - S_1) = V \sin(\omega t) \sin(\theta + 240^\circ) \quad (4.2c)$$

where S_1 , S_2 , etc., denotes the stator terminals. Moreover, V and ω are the input amplitude and frequency, respectively, whereas θ is the shaft angle. In the case of a resolver, with a rotor ac reference voltage of $V \sin(\omega t)$, the stator's terminal voltages will be

$$V_0(S_1 - S_3) = V \sin(\omega t) \sin \theta \quad (4.3a)$$

$$V_0(S_4 - S_2) = V \sin(\omega t) \sin(\theta + 90^\circ) = V \sin(\omega t) \cos \theta \quad (4.3b)$$

As said earlier, the output of these synchros and resolvers must be first digitized. To do this, analog-to-digital converters are used. These are typically 8-bit or 16-bit. An 8-bit means that the whole range of analog signals will be converted into a maximum of $2^8 = 256$ values.

4.2.2 Velocity Sensors

Velocity or speed sensors measure by taking consecutive position measurements at known time intervals and computing the time rate of change of the position values or directly finding it based on different principles.

1. All Position Sensors Basically, all position sensors when used with certain time bounds can give velocity, e.g., the number of pulses given by an incremental position encoder divided by the time consumed in doing so. But this scheme puts some computational load on the controller which may be busy in some other computations.

2. Tachometer Such sensors can directly find the velocity at any instant of time, and without much of computational load. This measures the speed of rotation of an element. There are various types of tachometers in use but a simpler design is based on the Fleming's rule, which states 'the voltage produced is proportional to the rate of flux linkage.' Here, a conductor (basically a coil) is attached to the rotating element which rotates in a magnetic field (stator). As the speed of the shaft increases, the voltage produced at the coil terminals also increases. In other ways, as shown in Fig. 4.6, one can put a magnet on the rotating shaft and a coil on the stator. The voltage produced is proportional to the speed of rotation of the shaft. This information is digitized using an analog-to-digital converter and passed on to the computer.

3. Hall-effect Sensor Another velocity-measuring device is the Hall-effect sensor, whose principle is described next. If a flat piece of conductor material, called *Hall chip*, is attached to a potential difference on its two opposite faces, as indicated in Fig. 4.7 then the voltage across the perpendicular faces is zero. But if a magnetic field is imposed at right angles to the conductor, the voltage is generated on the two

other perpendicular faces. Higher the field value, higher the voltage level. If one provides a ring magnet, the voltage produced is proportional to the speed of rotation of the magnet.

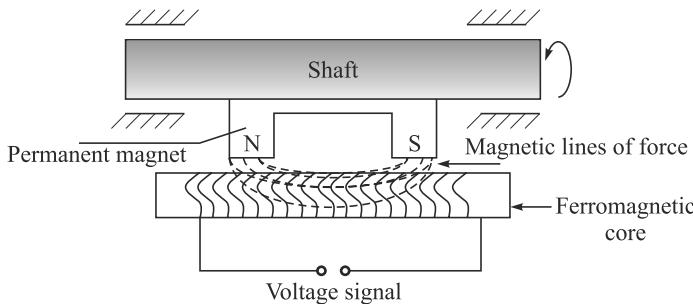


Fig. 4.6 Schematic diagram of a tachometer

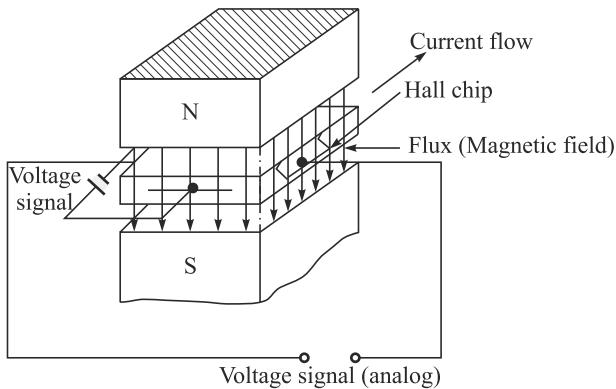


Fig. 4.7 Principle of Hall-effect sensor

4.2.3 Acceleration Sensors

Similar to measurements of velocity from the information of position sensors, one can find the accelerations as the time rate of change of velocities obtained from velocity sensors or calculated from the position information. But this is not an efficient way to calculate the acceleration because this will put a heavy computational load on the computer and that can hamper the speed of operation of the system. Another way to compute the acceleration is to measure the force which is the result of mass times acceleration. Forces are measured, for example, using strain gauges for which the formula is

Why acceleration from force but not from velocity?

Finding acceleration from force involves integration which suppresses any noise in the force signal. However, if the velocity signal is used to determine acceleration, differentiation has to be performed which amplifies the noise in the velocity signal. Hence, the latter is never advisable.

$$F = \frac{\Delta RAE}{RG} \quad (4.4)$$

where F is force, ΔR is the change in resistance of the strain gauge, A is the cross-sectional area of the member on which the force being applied, E is the elastic modulus of the strain-gauge material, R is the original resistance of the gauge, and G is gauge factor of the strain gauge. Then, the acceleration a is the force divided by mass of the accelerating object m , i.e.,

$$a = \frac{F}{m} = \frac{\Delta RAE}{RCm} \quad (4.5)$$

It is pointed out here that the velocities and accelerations that are measured using position sensors require differentiations. It is generally not desirable, as the noise in the measured data, if any, will be amplified. Alternatively, the use of integrators to obtain the velocity from the acceleration, and consequently the position, are recommended. Integrators tend to suppress the noise.

What is Gauge Factor?

It is a measure of sensitivity for the strain gauges, and defined by

$$G = \frac{1}{\varepsilon} \frac{\Delta R}{R}$$

where G is the gauge factor, and ε is strain.

Example 4.2 Change in Resistance

If the gauge factor $G = 2$, resistance of the unreformed wire $R = 100 \Omega$, and strain $\varepsilon = 10^{-6}$, then change in resistance is given by

$$\Delta R = G\varepsilon R = 2 \times 10^{-6} \times 100 = 0.0002 \Omega \quad (4.6)$$

4.2.4 Force Sensors

A spring balance is an example of a force sensor in which a force, namely, the weight, is applied to the scale pan that causes displacement, i.e., the spring stretches. The displacement is then a measure of the force. There exist other types of force sensors, e.g., strain-gauge based, Hall-effect sensor, etc.

1. Strain-gauge Based The principle of this type of sensors is that the elongation of a conductor increases its resistance. Typical resistances for strain gauges are 50–100 ohms. The increase in resistance is due to

- Increase in the length of the conductor; and
- Decrease in the cross-section area of the conductor.

Strain gauges are made of electrical conductors, usually wire or foil, etched on a base material, as shown in Fig. 4.8. They are glued on the surfaces where strains are to be measured, e.g., R_1 and R_2 of Fig. 4.9(a). The strains cause changes in the resistances of the strain gauges, which are measured by attaching them to the Wheatstone bridge circuit as one of the four resistances, $R_1 \dots R_4$ of Fig. 4.9(b). The principle of a Wheatstone bridge is explained in Section 4.5.5. It is a cheap and accurate method of measuring strain. But care should be taken for the temperature changes. In order to enhance the output voltage and cancel away the resistance changes due to the change in temperature, two strain gauges are used, as shown in Fig. 4.9(a), to measure the force at the end of the cantilever beam.

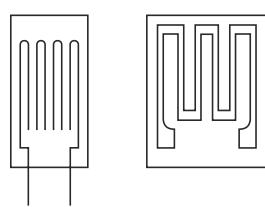


Fig. 4.8 Strain gauges

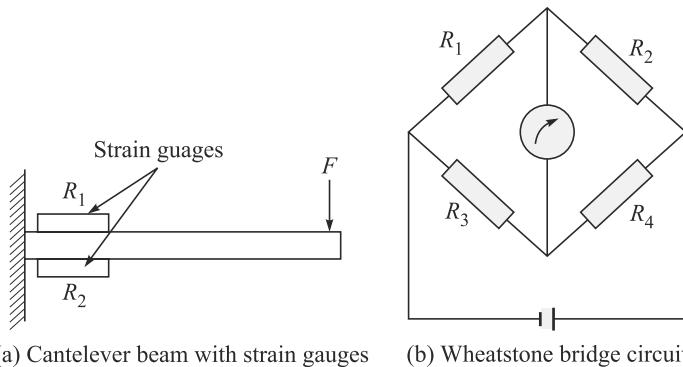


Fig. 4.9 Strain measurement

2. Piezoelectric Based A piezoelectric material exhibits a phenomenon known as the *piezoelectric effect*. This effect states that when asymmetrical, elastic crystals are deformed by a force, an electrical potential will be developed within the distorted crystal lattice, as illustrated in Fig. 4.10. This effect is reversible. That is, if a potential is applied between the surfaces of the crystal, it will change its physical dimensions. The magnitude and polarity of the induced charges are proportional to the magnitude and direction of the applied force. The piezoelectric materials are quartz, tourmaline, Rochelle salt, and others. The range of forces that can be measured using piezoelectric sensors are from 1 to 20 kN and at a ratio of 2×10^5 . These sensors can be used to measure an instantaneous change in force (dynamic forces).

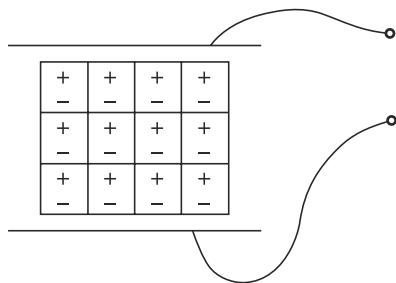


Fig. 4.10 Piezoelectric sensor

3. Current Based Since the torque provided by an electric motor is a function of the current drawn, its measurement, along with the known motor characteristics, gives the torque sensing.

Example 4.3 Force and Acceleration using Strain Gauges

If the strain gauge of Example 4.2 is used to measure applied force, F on a member of cross-sectional area, $A = 10 \text{ mm}^2$, and Young's modulus, $E = 6.9 \times 10^{10} \text{ N/m}^2$ (for aluminium), then using Eq. (4.4)

$$F = \frac{AE\Delta R}{GR} = \frac{(10 \times 10^{-6})(6.9 \times 10^{10})(0.0002)}{2(100)} = 0.69 \text{ N} \quad (4.7)$$

If the mass of the member $m = 3 \text{ kg}$ then using equations (4.5) and (4.7)

$$a = \frac{0.69}{3} = 0.21 \text{ m/s}^2 \quad (4.8)$$

4.3 EXTERNAL SENSORS

External sensors are primarily used to learn more about a robot's environment, especially the objects being manipulated. External sensors can be divided into the following categories:

- Contact type, and
- Noncontact type.

4.3.1 Contact Type

In this section, a contact-type force sensor is explained.

Limit Switch A limit switch is constructed much as the ordinary light switch used at homes and offices. It has the same on-off characteristics.

The limit switch usually has a pressure-sensitive mechanical arm, as shown in Fig. 4.11(a). When an object applies pressure on the mechanical arm, the switch is energized. An object might have an attached magnet that causes a contact to rise and close when the object passes over the arm. As shown in Fig. 4.11(b), the pull-up register keeps the signal at +V until the switch closes, sending the signal to ground. Limit switches can be either

Normally Open (NO) or Normally Closed (NC), and may have multiple-poles. A normally open switch has continuity when pressure is applied. A single-pole switch allows one circuit to be opened or closed upon contact, whereas a multi-pole switch allows multiple switch circuits to be open or closed. Limit switches are mechanical devices which have problems like

- they are subjected to mechanical failure,
- their mean time between failures is low compared to noncontact sensors, and
- the speed of operation is relatively slow compared to the speed of switching of photoelectric micro-sensors which is up to 3000 times faster.

Limit switches are used in robots to detect the extreme positions of the motions, where the link reaching an extreme position switches off the corresponding actuator, thus, safeguarding any possible damage to the mechanical structure of the robot arm.

4.3.2 Noncontact Type

Here, noncontact-type force sensors and their principles are presented.

1. Proximity Sensor Proximity sensing is the technique of detecting the presence or absence of an object with an electronic noncontact-type sensor. Proximity sensors are of two types, *inductive* and *capacitive*. Inductive proximity sensors are used in place of limit switches for noncontact sensing of metallic objects, whereas capacitive proximity sensors are used on the same basis as inductive proximity sensors. However, these can also detect nonmetallic objects.

Inductive Proximity Sensor All inductive proximity sensors consist of four basic elements, namely, the following:

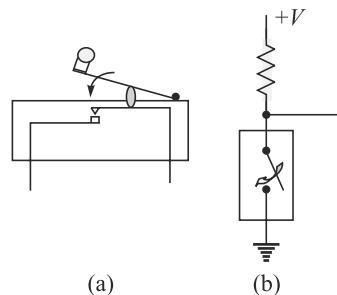


Fig. 4.11 Limit switch

- Sensor coil and ferrite core
- Oscillator circuit
- Detector circuit
- Solid-state output circuit

As shown in Fig. 4.12, the oscillator circuit generates a radio-frequency electromagnetic field. The field is centred around the axis of the ferrite core, which shapes the field and directs it at the sensor face. When a metal target approaches the face and enters the field, eddy currents are induced into the surface of the target. This results in a loading or damping effect that causes a reduction in amplitude of the oscillator signal. The detector circuit detects the change in the oscillator amplitude. The detector circuit will ‘switch on’ at specific operating amplitude. This signal ‘turns on’ the solid-state output circuit. This is often referred to as *damped condition*. As the target leaves the sensing field, the oscillator responds with an increase in amplitude. As the amplitude increases above a specific value, it is detected by the detector circuit, which is ‘switched off’ causing the output signal to return to the normal or ‘off’ state. The sensing range of an inductive proximity sensor refers to the distance between the sensor face and the target. It also indicates the shape of the sensing field generated through the coil and the core. There are several mechanical and environmental factors that affect the sensing range. The usual range is up to 10–15 mm but some sensors have ranges as high as 100 mm.

Capacitive Proximity Sensor A capacitive proximity sensor operates much like an inductive proximity sensor. However, the means of sensing is considerably different. Capacitive sensing is based on dielectric capacitance. Capacitance is the property of insulators to store the charge. A capacitor consists of two plates separated by an insulator, usually called a *dielectric*. When the switch is closed, a charge is stored on the two plates. The distance between the plates determines the ability of the capacitor to store the charge and can be calibrated as a function of stored charge to determine discrete ON and OFF switching status. Figure 4.13 illustrates the principle of a capacitive sensor. One capacitive plate is part of the switch, the sensor face is the insulator, and the target is the other plate. Ground is the common path. The

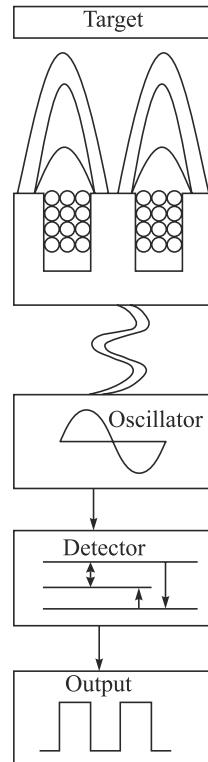


Fig. 4.12 Inductive proximity sensor

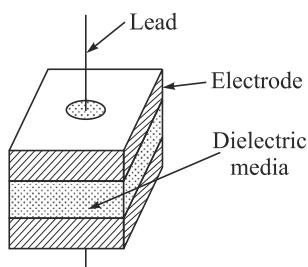


Fig. 4.13 Principle of capacitive sensors

capacitive switch has the same four elements as the inductive sensor, i.e., sensor (the dielectric media), oscillator circuit, detector circuit, and solid-state output circuit.

The oscillator circuit in a capacitive switch operates like one in an inductive switch. The oscillator circuit includes capacitance from the external target plate and the internal plate. In a capacitive sensor, the oscillator starts oscillating when sufficient feedback capacitance is detected. Major characteristics of the capacitive proximity sensors are as follows:

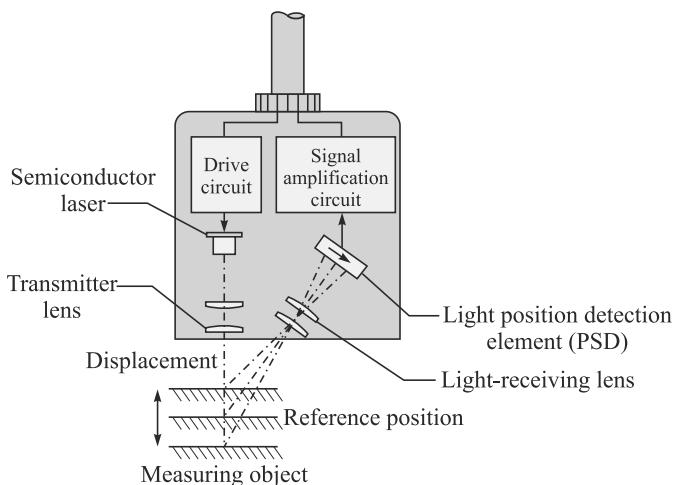
- They can detect non-metallic targets.
- They can detect lightweight or small objects that cannot be detected by mechanical limit switches.
- They provide a high switching rate for rapid response in object counting applications.
- They can detect limit targets through nonmetallic barriers (glass, plastics, etc.).
- They have long operational life with a virtually unlimited number of operating cycles.
- The solid-state output provides a bounce-free contact signal.

Capacitive proximity sensors have two major limitations.

- The sensors are affected by moisture and humidity, and
- They must have extended range for effective sensing.

Capacitive proximity sensors have a greater sensing range than inductive proximity sensors. Sensing distance for capacitive switches is a matter of plate area, as coil size is for inductive proximity sensors. Capacitive sensors basically measure a dielectric gap. Accordingly, it is desirable to be able to compensate for the target and application conditions with a sensitivity adjustment for the sensing range. Most capacitive proximity sensors are equipped with a sensitivity adjustment potentiometer.

2. Semiconductor Displacement Sensor As shown in Fig. 4.14, a semiconductor displacement sensor uses a semiconductor Light Emitting Diode (LED) or laser as a light source, and a Position-Sensitive Detector (PSD). The laser



[Courtesy: <http://www.sensortcentral.com/displacement/laser02.php>]

Fig. 4.14 Semiconductor-based sensor

beam is focused on the target by a lens. The target reflects the beam, which is then focused on to the PSD forming a beam spot. The beam spot moves on the PSD as the target moves. The displacement of the workpiece can then be determined by detecting the movement of the beam spot.

4.4 VISION

Vision can be defined as the task of extracting information about the external world from light rays imaged by a camera or an eye. Vision, also referred in the literature as *computer vision* or *machine vision* or *robot vision*, is a major subject of research and many textbooks, e.g., by Haralick and Shapiro (1992, 1993), and others. A good coverage on the topic has also appeared in Niku (2001). There are also dedicated journals, e.g., *Computer Vision, Graphics, and Image Processing*, and conferences in the area of robot vision. The area is so vast that it cannot be covered in one section or chapter of a book. However, an attempt is made here to introduce the basic concepts and techniques so that one is able to understand the systems and methodologies used in robot vision. For detailed study and research, other references in the area should be consulted.

Note in Fig. 4.1 that the vision systems or vision sensors are classified as external noncontact type. They are used by robots to let them look around and find the parts, for example, picking and placing them at appropriate locations. Earlier, fixtures were used with robots for accurate positioning of the parts. Such fixtures are very expensive. A vision system can provide alternative economic solution. Other tasks of vision systems used with robots include the following:

1. Inspection Checking for gross surface defects, discovery of flaws in labeling, verification of the presence of components in assembly, measuring for dimensional accuracy, checking the presence of holes and other features in a part.

2. Identification Here, the purpose is to recognize and classify an object rather than to inspect it. Inspection implies that the part must be either accepted or rejected.

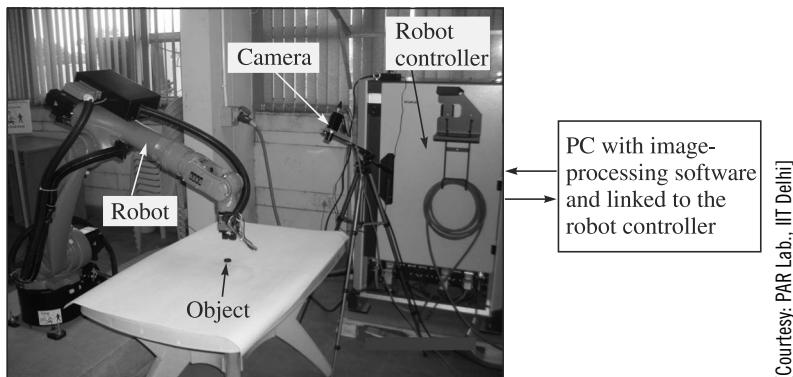
3. Visual Servoing and Navigation Control The purpose here is to direct the actions of the robot based on its visual inputs, for example, to control the trajectory of the robot's end-effector toward an object in the workspace. Industrial applications of visual servoing are part positioning, retrieving parts moving along a conveyor, seam tracking in continuous arc welding, etc.

All of the above applications somehow require the determination of the configuration of the objects, motion of the objects, reconstruction of the 3D geometry of the objects from their 2D images for measurements, and building the maps of the environments for a robot's navigation. Coverage of vision system is from a few millimetres to tens of metres with either narrow or wide angles, depending upon the

Computer Vision vs. Computer Graphics

Computer vision can be thought of as 'inverse computer graphics.' Computer graphics deals with how to generate images from a specification of the visual scene (e.g., objects, scene structures, light sources), whereas computer vision inverts this process to infer the structure of the world from the observed image(s).

system needs and design. Figure 4.15 shows a typical visual system connected to an industrial robot.



[Courtesy: PAR Lab., IIT Delhi]

Fig. 4.15 Hardware components of a vision system

4.4.1 Elements in a Vision Sensor

In vision systems, the principal imaging component is a complete camera including sensing array, associated electronics, output signal format, and lens, as shown in Fig. 4.16.

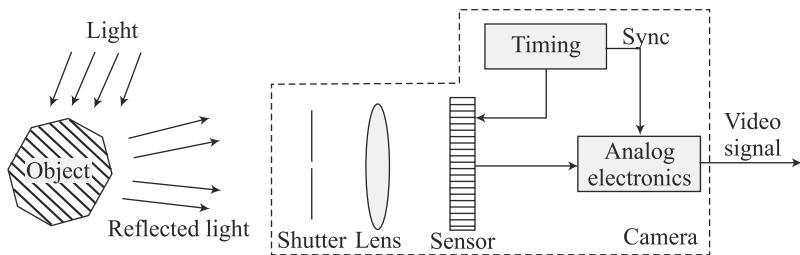


Fig. 4.16 Principal of a camera or vision system [Courtesy: Siciliano, 2010]

The task of the camera as a vision sensor is to measure the intensity of the light reflected by an object, as indicated in Fig. 4.16, using a photosensitive element termed pixel (or photosite). A pixel is capable of transforming light energy into electric energy. The sensors of different types like CCD, CMOS, etc., are available depending on the physical principle exploited to realize the energy transformation. Depending on the application, the camera could be RS-170/CCIR, NTSC/PAL (These are American RS-170 monochrome, European/Indian CCIR monochrome, NTSC color, PAL color television standard signal produced by the video cameras, respectively) progressive scan, variable scan, or line scan. Five major system parameters which govern the choice of camera are field of view, resolution, working distance, depth of field, and image data acquisition rate. As a rule of thumb, for size measurement, the sensor should have a number of pixels at least twice the ratio of the largest to smallest object sizes of interest.

1. Camera Systems As indicated in Fig. 4.16, a camera is a complex system comprising of several devices inside it. Other than the photosensitive sensor, there are shutter, a lens, and analog preprocessing electronics. The lens is responsible for focusing the light reflected by the object on the plane where the photosensitive sensors lies, called the image plane. In order to use it to compute the position and/or orientation of an object, the associated coordinate transformations, etc., need to be performed using the knowledge of Chapter 5, i.e., Transformation. This is generally carried out by a software residing inside a personal computer which saves the images. Specific calculations related to camera calibration, etc., can be found in Haralick and Shapiro (1993), Siciliano et al. (2011), and others. They are not covered in this introductory book on robotics.

Note that there are two types of video cameras: *analog* and *digital*. Analog cameras are not in common anymore. However, if it is used, a frame grabber or video capture card, usually a special analog-to-digital converter adopted for video signal acquisition in the form of a plug-in board which is installed in the computer, is often required to interface the camera to a host computer. The frame grabber will store the image data from the camera on-board, or system memory, and performs sampling and digitizing of the analog data as necessary. In some cases, the camera may output digital data, which is compatible with a standard computer. So a separate frame grabber may not be needed. Vision software is needed to create the program which processes the image data. When an image has been analyzed, the system must be able to communicate the result to control the process or to pass information to a database. This requires a digital input/output interface. The human eye and brain can identify objects and interpret scenes under a wide variety of conditions. Robot-vision systems are far less versatile. So the creation of a successful system requires careful consideration of all elements of the system and precise identification of the goals to be accomplished, which should be kept as simple as possible.

Vidicon Camera Early vision systems employed vidicon cameras, which were bulky vacuum tube devices. They are almost extinct today but explained here for the sake of completeness in the development of video cameras. Vidicons are also more sensitive to electromagnetic noise interference and require high power. Their chief advantages are higher resolution and better light sensitivity. Figure 4.17 shows the schematic diagram of a vidicon camera.

The mosaic reacts to the varying intensity of a light by varying its resistance. Now, as the electric gun generates and sends a continuous cathode beam to the mosaic passing through two pairs of orthogonal capacitors (deflectors), the electron beam gets deflected up or down, and left or right based on the charge on each pair of capacitors. As the beam scans the image, at each instant, the output is proportional to the resistance of the mosaic or the light intensity on the mosaic. By reading the output voltage continuously, an analog representation of the image can be obtained. Please note that the analog signal of vidicon needs to be converted to digital signal using analog-to-digital converters (ADC), as mentioned in Section 2.1.2 in order to process the image further using a PC. The ADC which actually performs the digitization of the analog signal requires mainly three steps, i.e., sampling, quantization, and encoding.

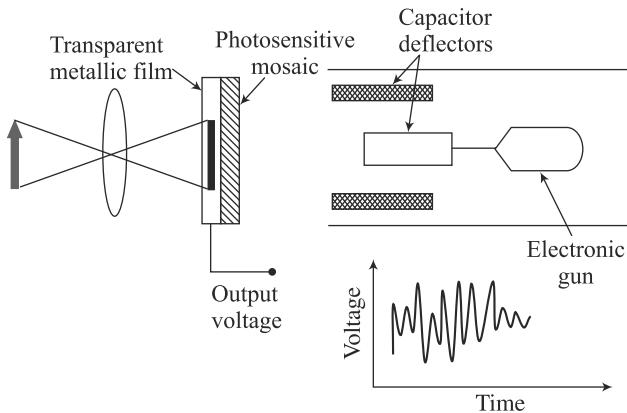


Fig. 4.17 Schematic diagram of a vidicon camera [Courtesy: Niku (2001)]

In sampling, a given analog signal is sampled periodically to obtain a series of discrete-time analog signal, as illustrated in Fig. 4.18. By setting a specified sampling rate, the analog signal can be approximated by the sampled digital outputs. However, while reconstructing the original signal from the sample data, one may end up with a completely different signal. This loss of information is called *aliasing*, and it can be a serious problem. In order to prevent aliasing, according to the sampling theorem, the sampling rate must be at least twice the largest frequency in the original video signal if one wishes to reconstruct that signal exactly.

In quantization, each sampled discrete time voltage level is assigned to a finite number of defined amplitude levels. These levels correspond to the Gray scale used in the system. The predefined amplitude levels are characteristics to a particular ADC and consist of a set of discrete values of voltage levels. The number of quantization levels is defined by 2^n , where n is the number of bits of the ADC. For example, a 1-bit ADC will quantize only at two values, whereas with an 8-bit ADC, it is possible to quantize at $2^8 = 256$ different values. Note that a large number of bits enables a signal to be represented more precisely. Moreover, sampling and quantization resolutions are completely independent of each other.

Finally, encoding does the job of converting the amplitude levels that are quantized into digital codes, i.e., 0 or 1. The ability of the encoding process to distinguish between various amplitude levels is a function of the spacing of each quantization level.

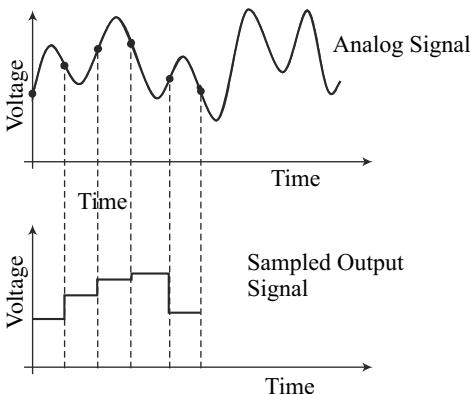


Fig. 4.18 Sampling and digitalization

Example 4.4 Quantization

An 8-bit ADC is to be used to represent a range of video signal between 0 and 5 V. The resolution or quantization spacing is given by

$$\frac{5}{2^8} = 0.0195 \text{ V} \quad (4.9)$$

Accordingly, quantization error can be given by,

$$\pm \frac{0.0195}{2} = 0.00975 \text{ V} \quad (4.10)$$

Digital Camera A digital camera is based on solid-state technology. The main part of these cameras is a solid-state silicon wafer image area that has hundreds of thousands of extremely small photosensitive areas called *photosites* printed on it. Each small area of the wafer is a pixel. As the image is projected onto the image area, at each pixel location of the wafer, a charge is developed that is proportional to the intensity of the light at that location. Thus, a digital camera is also called a *Charged Coupled Device* (CCD) camera or *Charge Integrated Device* (CID) camera. The collection of charges, as shown in Fig. 4.19, if read sequentially, would be a representation of the image pixels. The output is a discrete representation of the image as a voltage sampled in time. Solid-state cameras are smaller, more rugged, last longer, and have less inherent image distortion than vidicon cameras. They are also slightly more costly, but prices are coming down.

What is a Pixel?

A pixel is a short form for *picture element*. It is a single point in a graphic image.

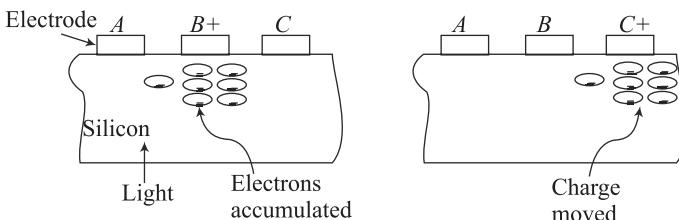


Fig. 4.19 Basic principle of a CCD device (Image acquisition)

Both the CCDs and CID chips use large transfer techniques to capture an image. In a CCD camera, light impinges on the optical equivalent of a Random Access Memory (RAM) chip. The light is absorbed in a silicon substrate, with charge build-up proportional to the amount of light reaching the array. Once sufficient amount of energy has been received to provide a picture, the charges are read out through built-in control registers. Some CCD chips use an interline charge-transfer technique. Others use frame-transfer approach, which is more flexible for varying the integration period.

The CID camera works on a similar principle. A CID chip is a Metal Oxide Semiconductor (MOS) based device with multiple gates similar to CCDs. The video signal is the result of a current pulse from a recombination of carriers. CIDs produce

a better image (less distortion) and use a different read-out technique than CCDs which require a separate scanning address unit. CIDs are, therefore, more expensive than CCDs. The principle difference between a CCD and a CID camera is the method of generating the video signal.

2. Lighting Techniques One of the key questions in robot vision is what determines how bright the image of some surface on the object will be? It involves radiometry (measurement of the flow and transfer of radiant energy), general illumination models, and surface having both diffuse and specular reflection components. Different points on the objects in front of the imaging system will have different intensity values on the image, depending on the amount of incident radiance, how they are illuminated, how they reflect light, how the reflected light is collected by a lens system, and how the sensor camera responds to the incoming light. Figure 4.20 shows the basic reflection phenomenon. Hence, proper illumination of the scene is important. It also affects the complexity level of the image-processing algorithm required. The lighting techniques must avoid reflections and shadow unless they are designed for the purpose of image processing. The main task of lighting is to create contrast between the object features to be detected. Typical lighting techniques are explained below.

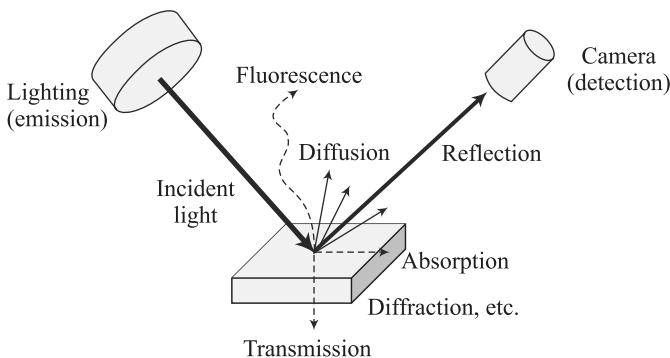


Fig. 4.20 Light reflections

Direct Incident Lighting This simple lighting technique can be used for non-reflective materials which strongly scatter the light due to their matte, porous, fibrous, non-glossy surface. Ideally, a ring light is chosen for smaller illuminated fields that can be arranged around the lens. Shadows are avoided to the greatest extent due to the absolutely vertical illumination. Halogen lamps and large fluorescence illumination can be used too.

Diffuse Incident Lighting Diffused light is necessary for many applications, e.g., to test reflective, polished, glossy, or metallic objects. It is particularly difficult if these surfaces are not glossy, perfectly flat, but individually shaped, wrinkled, curved, or cylindrical. To create diffused lighting, one may use incident light with diffusers, coaxial illumination, i.e., light is coupled into the axis of the camera by means of a beam splitter or half-mirror, or the dome-shaped illumination where light is diffused by means of a diffused coated dome in which the camera looks through an opening in the dome onto the workpiece.

Lateral Lighting Light from the side can be radiated at a relatively wide or narrow angle. The influence on the camera image can be significant. In an extreme case, the image information can almost be inverted.

Dark Field Lighting At first sight, images captured using dark field illumination seem unusual to the viewer. The light shines at a shallow angle. According to the principle of *angle of incidence* equals the *angle of reflection*, all the light is directed away from the camera. The field of view, therefore, remains dark. Inclined edges, scratches, imprints, slots, and elevations interfere with the beam of light. At these anomalies, the light is reflected towards the camera. Hence, these defects appear bright in the camera image.

Backlighting Transmitted light illumination is the first choice of lighting when it is necessary to measure parts as accurately as possible. The lighting is arranged on the opposite side of the camera, the component itself is put in the light beam.

Example 4.5 Pixel Processing

Assume a vidicon tube generates a signal for 512 lines of a faceplate. If the sampling capability of an ADC is 100 nanoseconds, i.e., the time required to process one pixel, and 1/30 seconds are required to scan all 512 lines then scanning rate for each line is,

$$\frac{1}{(30 \times 512)} = 65.1 \text{ microsecond per line} \quad (4.11)$$

Hence, the number of pixels which can be processed per line is given below:

$$\frac{65.1 \times 10^{-6}}{100 \times 10^{-9}} = 651 \text{ pixels per line} \quad (4.12)$$

Example 4.6 Memory Requirement

If a video image requires 256×256 pixels, the total number of pixels in an image is,

$$256 \times 256 = 65,536 \quad (4.13a)$$

Assuming each pixel is digitalized at the rate of 8 bits for 256 shades of gray then it would require

$$65,536 \times 8 = 524,288 \text{ bits or } 65,536 \text{ bytes} \quad (4.13b)$$

In case the picture is in three colors of red, green, and blue (RGB), and the video image changes at the rate of 30 images per second then the memory requirement is,

$$65,536 \times 3 \times 30 = 5,898,240 \text{ bytes per second} \quad (4.13c)$$

4.4.2 Steps in a Vision System

As depicted in Fig. 4.21, vision sensing has two steps, namely, image acquisition and image processing. They are explained below.

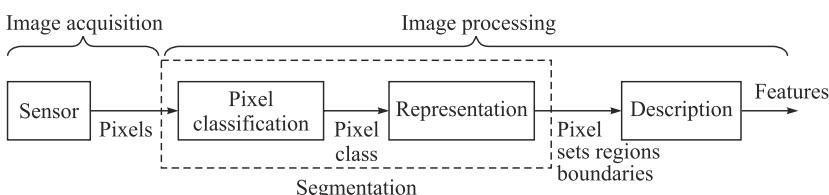


Fig. 4.21 Steps in vision

1. Image Acquisition In image acquisition, an image is acquired from a vidicon which is digitized or from a digital camera (CCD or CID), as explained in the previous section. The image is stored in computer memory (also called a frame buffer) in the format such as TIFF, JPG, Bitmap, etc. The buffer may be a part of the frame grabber card or in the computer itself. Note that the image acquisition is primarily a hardware function, however, software can be used to control light intensity, focus, camera angle, synchronization, field of view, read times, and other functions. Image acquisition has four principle elements, namely, a light source, either controlled or ambient, which is explained in Section 4.4.1, a lens that focuses reflected light from the object on to the *image sensor*, an image sensor that converts the light image into a stored electrical image, and the electronics to read the sensed image from the image-sensing element, and after processing, transmit the image information to a computer for further processing. A typical acquired image is shown in Fig. 4.22.

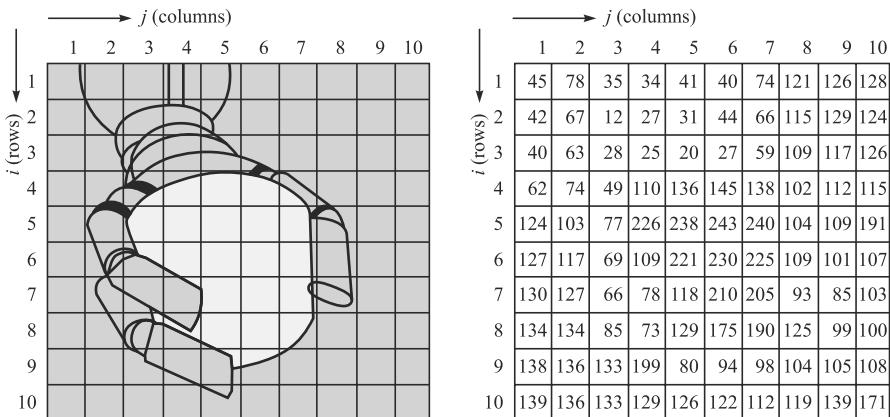


Fig. 4.22 Digitized picture

Example 4.7 Image Storage and Retrieval

Note that an image is generally acquired at the rate of 30 frames per second or each image stored in every $1/30$ second. If a vision system has a pixel density of 350 pixels per line and 280 lines, there will be a total of $350 \times 280 = 98,000$ pixels. Moreover, for a 6-bit register to represent various gray levels of a pixel, there is a requirement of storage of $98,000 \times 6 = 588,000$ bits of data, which is to be processed in every $1/30$ second. For storage, a combination of row and column counters is used in the frame grabber. To read the information stored in the frame buffer, the data is grabbed via a signal sent from the computer to the address corresponding to a row-column combination.

2. Image Processing Image-processing techniques are used to enhance, improve, or otherwise alter an image and to prepare it for image analysis. Usually, during image processing, information is not extracted from the image. The intention is to remove faults, trivial information, or information that may be important, and to improve the image. Image processing examines the digitized data to locate and

recognize an object within the image field. It is divided into several sub-processes, which are discussed below:

Image Data Reduction Here, the objective is to reduce the volume of data. As a preliminary step in the data analysis, the schemes like digital conversion or windowing can be applied to reduce the data. While the digital conversion reduces the number of gray levels used by the vision system, windowing involves using only a portion of the total image stored in the frame buffer for image processing and analysis. For example, in windowing, to inspect a circuit board, a rectangular window is selected to surround the component of interest and only pixels within that window are analyzed.

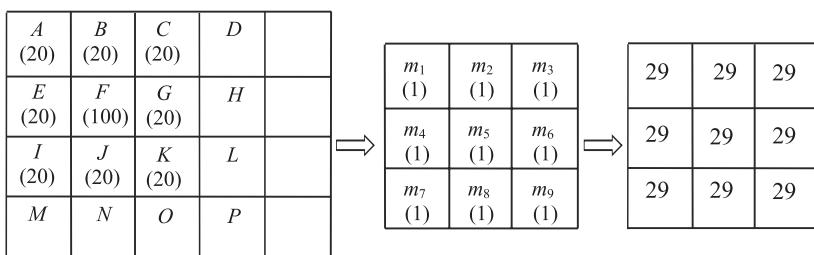
Histogram Analysis A histogram is a representation of the total number of pixels of an image at each gray level. Histogram information is used in a number of different processes, including thresholding. For example, histogram information can help in determining a cut-off point when an image is to be transformed into binary values.

Thresholding It is the process of dividing an image into different portions or levels by picking a certain grayness level as a threshold. Comparing each pixel value with the threshold, and then assigning the pixel to the different portions or level, depending on whether the pixel's grayness level is below the threshold ('off' or 0, or not belonging) or above the threshold ('on' or 1, or belonging).

Masking A mask may be used for many different purposes, e.g., filtering operations and noise reduction, and others. It is possible to create masks that behave like a low-pass filter such that higher frequencies of an image are attenuated while the lower frequencies are not changed very much. This is illustrated in Example 4.8. Thereby, the noise is reduced. Masking an image considers a portion of an imaginary image shown in Fig. 4.23(a), which has all the pixels at a gray value of 20 except the one at a gray level of 100. The one with 100 may be considered noise. Applying the 3×3 mask shown in Fig. 4.23(b) over the corner of the image yields the following value:

$$X = \frac{(m_1A + m_2B + m_3C + m_4E + m_5F + m_6G + m_7I + m_8J + m_9K)}{S} = 29 \quad (4.14)$$

where $S \equiv m_1 + m_2 + \dots + m_9 = 9$, and values of A, B, \dots , and m_1, m_2, \dots are shown in Figs. 4.23 (a) and (b), respectively. As a result of application of the mask, the pixel with the value of 100 changes to 29, as indicated in Fig. 4.23(c). The large difference



(a) Gray value of an image

(b) 3×3 masking

(c) Change in pixel value

Fig. 4.23 Masking of an image

between the noisy pixel and the surrounding pixels, i.e., 100 vs. 20, becomes much smaller, namely, 29 vs. 20, thus reducing the noise. With this characteristic, the mask acts as a low-pass filter. Note that the above reduction of noise has been achieved using what is referred as *neighborhood averaging*, which causes the reduction of the sharpness of the image as well.

Edge Detection Edge detection is a general name for a class of computer programs and techniques that operate on an image and result in a line drawing of the image. The lines represent changes in values such as cross section of planes, intersections of planes, textures, lines, etc. In many edge-detection techniques, the resulting edges are not continuous. However, in many applications, continuous edges are preferred, which can be obtained using the Hough transform. It is a technique used to determine the geometric relationship between different pixels on a line, including the slope of the line. Consider a straight line in the xy -plane, as shown in Fig. 4.24, which is expressed as

$$y = mx + c \quad (4.15)$$

where m is the slope and c is the intercept. The line represented by Eq. (4.15) can be transformed into a Hough plane of $m - c$ with x and y as its slope and intercept, respectively. Thus, a line in the xy -plane with a particular slope and intercept will transform into a point in the Hough plane. For the Hough or $m - c$ plane, if a line passes through (m_1, c_1) , as shown in Fig. 4.24(b), it represents a point (x_1, y_1) in the x - y plane of Fig. 4.24(a). In that way, all lines through a point in the Hough plane will transform into a single line in the x - y plane. Alternatively, if a group of points is collinear, their Hough transform will all intersect. By examining these properties in a particular case, it can be determined whether a cluster of pixels is on a straight line or not, whereupon the orientation of an object in a plane can be determined by calculating the orientation of a particular line in the object.

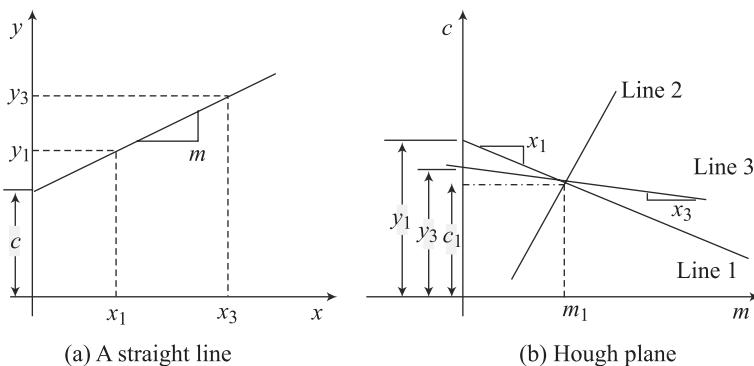
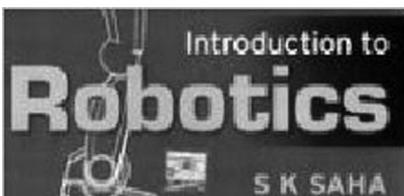


Fig. 4.24 Hough transform

Segmentation Segmentation is a generic name for a number of different techniques that divide the image into segments of its constituents. The purpose of segmentation is to separate the information contained in the image into smaller entities that can be used for other purposes. Segmentation includes edge detection, as mentioned

above, region growing and splitting, and others. While region growing works based on the similar attributes, such as gray-level ranges or other similarities, and then try to relate the regions by their average similarities, region splitting is carried out based on thresholding in which an image is split into closed areas of neighborhood pixels by comparing them with thresholding value or range.

Morphology Operations Morphology operations are a family of operations which are applied on the shape of subjects in an image. They include many different operations, both for binary and gray images, such as thickening, dilation, erosion, skeletonization, opening, closing, and filing. These operations are performed on an image in order to aid in its analysis, as well as to reduce the ‘extra’ information that may be present in the image. For example, Fig. 4.25(a) shows the object which after skeletonization is shown in Fig. 4.25(b).



(a) Object



(b) After reducing image to lines

Fig. 4.25 Skeletonization

3. Image Analysis Image analysis is a collection of operations and techniques that are used to extract information from images. Among these are feature extraction; object recognition; analysis of the position, size, orientation; extraction of depth information, etc. Some techniques can be used for multiple purposes. For example, moment equations may be used for object recognition, as well as to calculate the position and orientation of an object. It is assumed that image processing has already been performed to the image and available for the image analysis. Some of the image-analysis techniques are explained below.

Feature Extraction Objects in an image may be recognized by their features that uniquely characterize them. These include, but are not limited to, gray-level histograms, morphological features such as perimeter, area, diameter, number of holes, etc., eccentricity, cord length, and moments. As an example, perimeter of an object may be found by first applying an edge-detection routine and then counting the number of pixels on the perimeter. Similarly, the area can be calculated by region-growing techniques, whereas diameter of a non circular object is obtained by the maximum distance between any two points on any line that crosses the identified area of the object. In order to know the thinness of an object, it can be calculated using either of the two ratios:

$$\text{Thickness} = \frac{(\text{Perimeter})^2}{\text{Area}} \quad \text{or} \quad \text{Thinness} = \frac{\text{Diameter}}{\text{Area}} \quad (4.16)$$

Moreover, moment of an image, denoted with $n_{p,q}$, can be calculated as

$$n_{p,q} = \sum_{x,y} x^p y^q \quad (4.17)$$

where x and y are the coordinates of each pixel that is turned on within an image, and p and q are their respective powers, as illustrated in Fig. 4.26. The coordinates x and y are measured either from a designated coordinate frame located at the edge of the image or are measured from the frame formed by the first row and column of the image. As an example, in Fig. 4.26, the first ‘on-pixel’ appears in the second row and fourth column. As per xy coordinate frame, it is at $(4, 2)$ location, whereas the x distance of the pixel with respect to the first column is 3. For calculation purposes, only one coordinate frame should be used.

According to Eq. (4.17), when $p = q = 0$, the moment $n_{0,0}$ can be calculated as

$$n_{0,0} = \sum_{x,y} 1 = \text{Number of pixels, i.e., area} \quad (4.18)$$

Similarly, one can find $n_{0,1}$ as the summation of y -coordinates which is same as the summation of each pixel area multiplied by its distance from the x -axis. Hence, it is similar to the first moment of the area relative to the x -axis. Then, the location of the center of the area relative to the x -axis can be calculated immediately as

$$\bar{y} = \frac{1}{\text{Area}} \sum y = \frac{n_{0,1}}{n_{0,0}} \quad (4.19a)$$

Following Eq. (4.19a), one can also find the location of the center of the area relative to the y -axis as

$$\bar{x} = \frac{1}{\text{Area}} \sum x = \frac{n_{1,0}}{n_{0,0}} \quad (4.19b)$$

In a similar manner, one can also find the second moment of area with respect to x and y axes by putting $p = 2; q = 0$, and $p = 0; q = 2$, respectively. Note here that the first moments are independent of orientation, whereas the second moments are not.

Example 4.8 Moments Calculation

Referring to Fig. 4.26 and Eqs. (4.17–4.19),

$$n_{0,0} = \sum_{x,y} 1 = 13 \quad (4.20a)$$

$$n_{0,1} = \sum y = \sum x^0 y^1 = 1(2) + 5(3) + 5(4) + 1(5) + 1(6) = 48 \quad (4.20b)$$

$$n_{1,0} = \sum x = \sum x^1 y^0 = 2(2) + 2(3) + 5(4) + 2(5) + 1(6) + 1(7) = 53 \quad (4.20c)$$

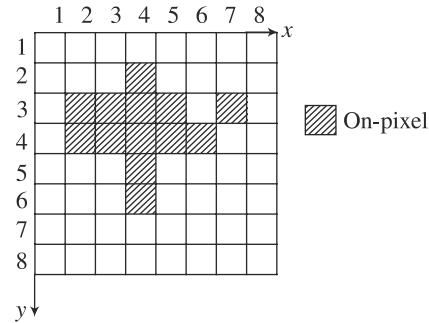


Fig. 4.26 Moment of an image

$$n_{0,2} = \sum x^0 y^2 = 1(2)^2 + 5(3)^2 + 5(4)^2 + 1(5)^2 + 1(6)^2 = 190 \quad (4.20d)$$

$$n_{2,0} = \sum x^2 y^0 = 2(2)^2 + 2(3)^2 + 5(4)^2 + 2(5)^2 + 1(6)^2 + 1(7)^2 = 241 \quad (4.20e)$$

Using Eqs (4.20a-c), one can easily find the center of the picture as,

$$\bar{x} = \frac{n_{1,0}}{n_{0,0}} = \frac{53}{13} = 4.1; \text{ and } \bar{y} = \frac{n_{0,1}}{n_{0,0}} = \frac{48}{13} = 3.7 \quad (4.21)$$

Object Recognition The next in image analysis is to identify the object that the image represents based on the extracted features. The recognition algorithm should be powerful enough to uniquely identify the object. Typical techniques used in the industries are *template matching* and *structural technique*.

In *template matching*, the features of the object in the image, e.g., its area, diameter, etc., are compared to the corresponding stored values. These values constitute the stored template. When a match is found, allowing for certain statistical variations in the comparison process, the object has been properly classified. For example, the area of an image given by the moment $n_{0,0}$ can be used to determine nature of the object, and to distinguish it from other objects that have different areas. Similarly, the first and second moments can be used to identify an object's position and orientation so that it can be picked up by the robot. Note that, instead of calculating the moment values which change as the object moves, a look-up table can be prepared. For example, the moment of inertia of an object varies significantly as the object rotates about its center. Hence, the values of $n_{2,0}$ and $n_{0,2}$ will also change. Since each orientation creates a unique value, a look-up table that contains these values can later be used to identify the orientation of the object.

An important aspect here is that similar shapes, as shown in Fig. 4.27 cannot be distinguished with their first and second moments, as they are similar or close to each other. In such a situation, a small difference between the two objects may be exaggerated through higher-order moments, making identification of the object possible.

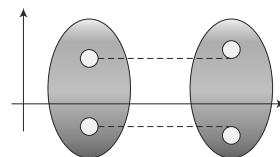


Fig. 4.27 Two similar objects

Structural techniques of pattern recognition rely on the relationship between features or edges of an object. For example, if the image of an object can be subdivided in four straight lines connected at their end points, and the connected lines are at right angles then the object is a rectangle. This kind of technique is known as *syntactic pattern recognition*.

In case depth information of an object is required, it can be obtained either by using a range finder in conjunction with a vision system, or using a stereo vision. The details of these systems are beyond the scope of this book. Readers are encouraged to look at Haralick and Shapiro (1992) and others.

4.4.3 Hierarchy of a Vision System

The collection of processes involved in visual perception are often perceived as a hierarchy spanning the range from 'low' via 'intermediate' to 'high-level' vision. The notion of 'low' and 'high' vision are used routinely, but there is no clear definition of

the distinction between what is considered ‘high’ as opposed to ‘low-level’ vision. As shown in Fig. 4.28, a vision is classified as ‘low’, ‘intermediate’ or ‘high-level’ vision based on specific activities during the image-processing stage. They are explained below.

1. Low-level Vision The sequence of steps from image formation to image acquisition, etc., described above, along with the extraction of certain physical properties of the visible environment, such as depth, three-dimensional shape, object boundaries, or surface-material properties, can be classified as a process of low-level vision. Activity in the low-level vision is to process images for feature extraction (edge, corner, or optical flow). Operations carried out on the pixels in the image to extract the above properties with respect to intensity or depth at each point in the image. One may, for example, be interested in extracting uniform regions, where the gradient of the pixels remains constant, or first-order changes in gradient, which would correspond to straight lines, or second-order changes which could be used to extract surface properties such as peaks, pits, ridges, etc. A number of characteristics that are typically associated with low-level vision processes are as follows:

- They are spatially uniform and parallel, i.e., with allowance for the decrease in resolution from the center of the visual field outwards, similar process is applied simultaneously across the visual field. For example, processing involved in edge detection, motion, or stereo vision, often proceed in parallel across the visual field, or a large part of it.
- Low-level visual processes are also considered ‘bottom-up’ in nature. This means that they are determined by the data, i.e., data driven, and are relatively independent of the task at hand or knowledge associated with specific objects. As far as the edge detection is concerned, it will be performed in the same manner for images of different objects, with no regard to whether the task to do with moving around, looking for a misplaced object, or enjoying the landscape.

2. Intermediate-level Vision In this level, objects are recognized and 3D scenes are interpreted using the features obtained from the low-level vision. The intermediate-level processing is fundamentally concerned with grouping entities together. The simplest case is when one groups pixels into lines. One can then express the line in a functional form. Similarly, if the output of the low-level information is a depth map, one may further need to distinguish object boundaries, or other characteristics. Even in the simple case where one is trying to extract a single sphere, it is not an easy process to go from a surface-depth representation to a center-and-radius representation. In contrast to higher-level vision, the process here does not depend on the knowledge about specific object.

3. High-level Vision High-level vision, which is equivalent to image understanding, is concerned mainly with the interpretation of scene in terms of the objects in it, and is usually based on knowledge of specific objects and relationships. It is concerned primarily with the interpretation and use of information in the image rather than the direct recovery of physical properties.

In high-level vision, interpretation of a scene goes beyond the tasks of line extraction and grouping. It further requires decisions to be made about types of boundaries, such as which are occluding, and what information is hidden from the

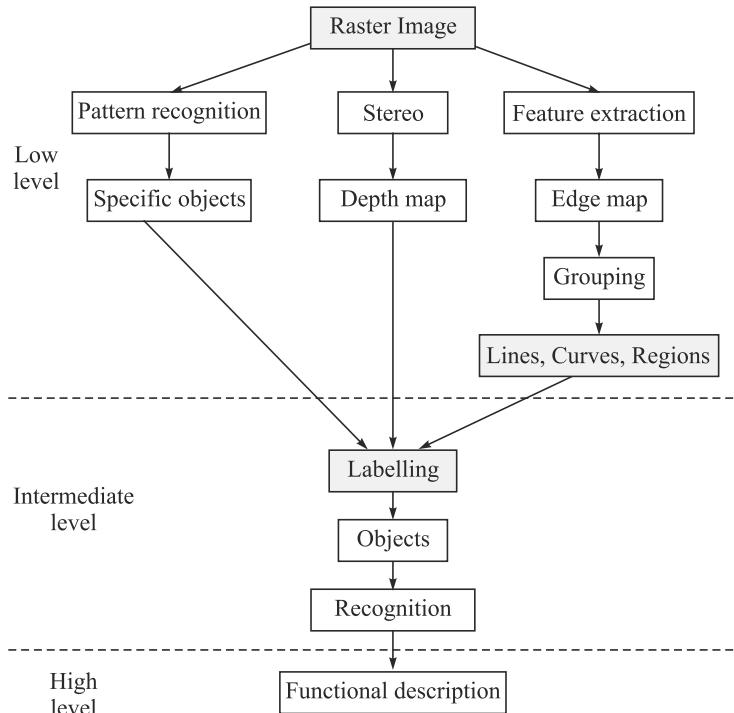


Fig. 4.28 Hierarchy of a vision system

user. Further grouping is essential at this stage since one may still need to be able to decide which lines group together to form an object. To do this, it is necessary to further distinguish lines which are part of the object structure, from those which are part of a surface texture, or caused by shadows. High-level systems are, therefore, object oriented, and sometimes called ‘top-down’. High-level visual processes are applied to a selected portion of the image, rather than uniformly across the entire image, as done in low- and intermediate-level visions. They almost always require some form of knowledge about the objects of the scene to be included.

4.4.4 Difficulties in Vision and Remedies

A vision system cannot uniquely represent or process all available data because of computational problem, memory, and processing-time requirements imposed on the computer. Therefore, the system must compromise. Other problems include variation of light, part-size, part placement, and limitations in the dynamic ranges available in typical vision sensors. A vision system requires specialized hardware and software. It is possible to purchase just the hardware with little or no vision application programming. In fact, a few third-party programs are available. A hardware-only approach is less expensive and can be more flexible for handling usual vision requirements. But, since this approach requires image processing expertise, it is only of interest to users who wish to retain the responsibility of image interpretation. It is usual practice to obtain the hardware and application software together from the supplier. However, the user might still need custom programming for an application.

Major vision system suppliers specialise in providing software for only a few application areas.

Every vision system requires a sensor to cover the visual image into an electronic signal. Several types of video sensors are used, including vidicon cameras, vacuum tube devices, and solid-state sensors. Many of these vision systems were originally designed for other applications, such as television so the signal must be processed to extract the visual image and remove synchronization information before the signal is sent to the computer for further processing. The computer then treats this digital signal as the array pixels, and processes this data to extract the desired information. Image processing can be very time consuming. For a typical sensor of 200,000 or more pixels, a vision system can take many seconds, even minutes, to analyze the complete scene and determine the action to be taken. The number of bits to be processed is quite large, for example, a system with 512×512 pixels array and an 8-bit intensity per pixel yields over two million of bits to be processed. If continuous image at a 30 FPS frame rate were being received, data bytes would be received at an 8 MHz rate. Few computers can accept inputs at these data rates and, in any case, there would be no time left to process the data. When higher resolution system, color system, or multiple camera systems are considered, data-handling requirements become astronomical. Several methods can be used to reduce the amount of data handled and, therefore, the processing time. They are explained as follows:

- (a) One approach is the binary vision, which is used when only black-and-white information is processed (intensity variations and shades of gray are ignored). In binary vision, a picture is converted into a binary image by *thresholding*, as illustrated in Fig. 4.29. In thresholding, a brightness level is selected. All data with intensities equal to or higher than this value are considered white, and all other levels are considered black.
- (b) Another method of shortening process time is to control object placement so that objects of interest cannot overlap in the image. Complicated algorithms to separate images are then not necessary, and the image-processing time is reduced.



(a) Initial image

(b) 30% threshold

Fig. 4.29 Image thresholding [Courtesy: Shell and Hall (2000)]

- (c) A third approach reduces data handling by processing only a small window of the actual data; that is, the object is located in a predefined field of view. For example, if the robot is looking for a mark on the printed circuit board, the system can be held in such a way that the mark is always in the upper right corner.
- (d) A fourth approach takes a statistical sample of data and makes decisions on this data sample. Unfortunately, all of these approaches ignore some of the available data and, in effect, produce a less robust system. Processing time is saved, but some types of complex objects cannot be recognized.

Note that some of the above steps are actually a part of image processing explained in Section 4.4.2.

4.5 SIGNAL CONDITIONING

The basic information or data generated by the transducers (or sensors as mentioned in ‘Sensors vs. Transducers’ text box in the beginning) generally requires ‘conditioning’ or ‘processing’ of one sort or another before they are presented to the observer as an indication or a record, or to be used by a robot controller for further action. Some of the commonly used devices used for signal conditioning of those obtained from the sensors explained in this chapter are explained next. The detailed descriptions are out of the scope of this book. However, they are available in the books by Doebelin (1990), de Silva (2004), Nakra and Chaudhry (2009), and others.

4.5.1 Amplifiers

Since the electrical signals produced by most transducers of a sensor are at a low voltage, generally they require amplification before they are suitably recognized by a monitoring system like a data processor, controller, or data logger. While the discussion is aimed mainly at users (rather than designers) of amplifiers, the use of operational amplifiers with the sensors will be explained here.

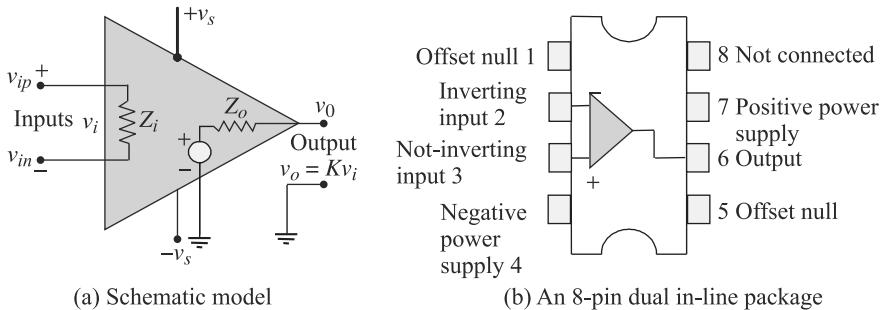
The operational amplifier (op-amp) is the most widely utilized analog electronic sub-assembly. It is the basis of instrumentation amplifier, filters, and a myriad of analog and digital data-processing equipment. An op-amp could be manufactured in the discrete-element form using, say, ten bipolar junction transistors and as many discrete resistors or alternatively (and preferably) in the monolithic form as in IC (Integrated Circuit) chip that may be equivalent to over 100 discrete elements. In any form, the device has an input impedance Z_i , an output impedance Z_o and a gain K , as indicated in Fig. 4.30(a). A common 8-pin dual in-line package (DIP) or V-package is shown in Fig. 4.30(b). From Fig. 4.30(a), the open-loop condition yields

$$v_o = Kv_i \quad (4.22)$$

where the input voltage v_i is the differential input voltage defined as the algebraic difference between the voltages at the +ve and –ve leads. Thus,

$$v_i = v_{ip} - v_{in} \quad (4.23)$$

The open-loop voltage gain K is typically very high ($10^5 - 10^9$), whereas values of Z_i and Z_o are typically $2\text{ M}\Omega$ (could be as high as $10\text{ M}\Omega$) and $10\text{ }\Omega$ (may reach about 75Ω), respectively. Since v_0 is typically $1-15\text{ V}$, from Eq. (4.22), it follows that $v_i \approx 0$ since K is very large. Hence, from Eq. (4.23) we have $v_{ip} \approx v_{in}$. In other words,

**Fig. 4.30** Operational amplifier

the voltages at the two input leads are nearly equal. If a large voltage differential v_i (say, 10 V) at the input then according to Eq. (4.22), the output voltage should be extremely high. This never happens in practice, because the device saturates quickly beyond moderate output voltages in the order of 15 V. From Eqs (4.22–4.23), it is clear that if the –ve input lead is grounded, i.e., $v_{in} = 0$ then

$$v_o = Kv_{ip} \quad (4.24a)$$

and if the +ve input lead is grounded, i.e., $v_{ip} = 0$ then

$$v_o = -Kv_{in} \quad (4.24b)$$

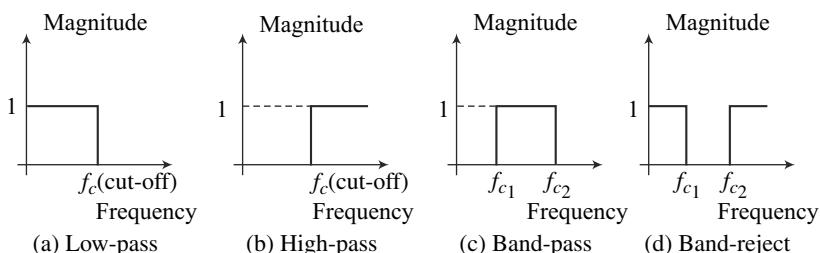
This is the reason why v_{ip} is termed non-inverting input and v_{in} is termed inverting input.

4.5.2 Filters

A filter is a device that allows through only the desirable part of a signal, rejecting the unwanted part. Unwanted signals can seriously degrade the performance of a robotic system. External disturbances, error components in excitations, and noise generated internally within system components and instrumentation are such spurious signals, which may be removed by a filter. There are four broad categories of filters, namely, low-pass filters, high-pass filters, band-pass filters, and band-reject (or notch) filters, which are shown in Fig. 4.31.

Dynamic System vs. Analog Filter

The way an analog sensor is defined, any dynamic system can be interpreted as an analog filter.

**Fig. 4.31** Filter characteristics

An analog filter contains active components like transistors or op-amps. It is a physical dynamic system, typically an electric circuit, whose dynamics will determine which (desired) signal components would be allowed through and which (unwanted) signal components would be rejected. In a way, output of the dynamic system is the filtered signal. An analog filter can be represented as a differential equation with respect to time.

Filtering can be achieved through digital filters as well which employ digital signal processing. Digital filtering is an algorithm by which a sampled signal (or sequence of numbers), acting as an input, is transformed to a second sequence of numbers called the output. It is a discrete-time system and can be represented as a difference equation. Digital filtering has the usual digital benefits of accuracy, stability, and adjustability by software (rather than hardware) changes.

4.5.3 Modulators and Demodulators

Signals are sometimes deliberately modified to maintain the accuracy during their transmission, conditioning, and processing. In modulation, the data signal which is referred to as modulating signal is used to vary a property, say, amplitude or frequency, of a carrier signal. Thus, the carrier signal is modulated by the data signal. After transmitting or conditioning, the data signal is recovered by removing the carrier signal from the modulated signal. This step is known as *demodulation*. The carrier signal can be either sine or square wave, and its frequency should be 5–10 times the highest frequency of the data signal.

Figure 4.32 shows some typical modulation techniques in which the amplitude of a high-frequency sinusoidal carrier signal is varied according to the amplitude of the data signal. Figure 4.32(a) shows the data signal which needs to be transmitted. The carrier signal's frequency is kept constant, while amplitude is same as that of the data signal. This is called Amplitude Modulation (AM), and the resulting amplitude-modulated signal is shown in Fig. 4.32(b). In Fig. 4.32(c), however, the frequency of

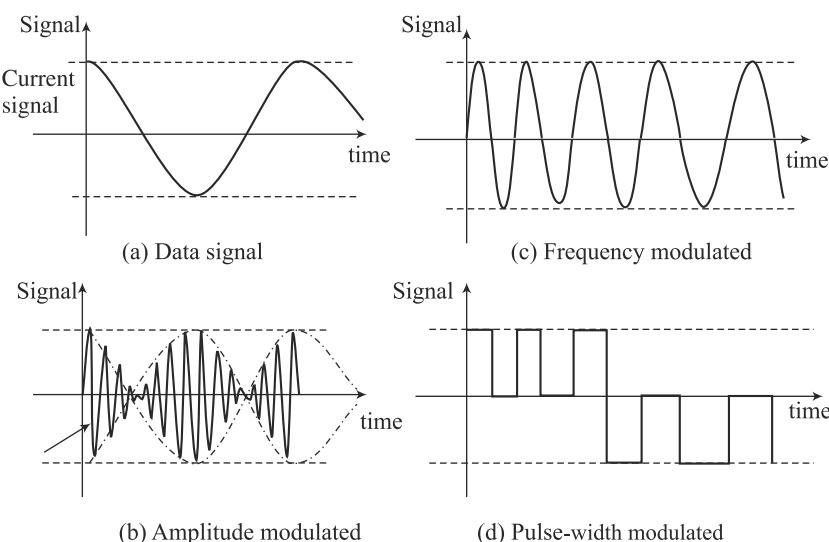


Fig. 4.32 Modulation techniques

the carrier signal is varied in proportion to the amplitude of the data signal (modulating signal), while keeping the amplitude of the carrier signal constant. This is called *Frequency Modulation (FM)*. An FM is less susceptible to noise than AM. In *Pulse-Width Modulation (PWM)*, the carrier signal is a pulse sequence, and its width is changed in proportion to the amplitude of the data signal, while keeping the pulse spacing constant. This is shown in Fig. 4.32(d). The PWM signals can be used directly in controlling a process without having to demodulate them. There also exists *Pulse-Frequency Modulation (PFM)*, where the carrier signal is a pulse sequence. Here, the frequency of the pulses is changed in proportion to the value of the data signal, while keeping the pulse width constant. It has the advantages of ordinary FM.

Demodulation or detection is the process of extracting the original data signal from a modulated signal. A simple and straightforward method of demodulation is by detection of the envelope of the modulated signal.

4.5.4 Analog and Digital Conversions

Most sensors have analog output while much data processing is done using digital computers. Hence, devices for conversion between these two domains have to be performed. These can be achieved using an analog-to-digital converter (ADC) and a digital-to-analog converter (DAC), as mentioned in Chapter 2. Explanation on DAC is given first as some ADC uses DAC as its component.

1. Digital-to-Analog Converter (DAC) The function of a digital-to-analog-converter or DAC is to convert a sequence of digital words stored in a data register, typically in the straight binary form, into an analog signal. A typical DAC unit is an active circuit in the integrated circuit (IC) form and consists of a data register (digital circuits), solid-state switching circuits, resistors, and op-amps powered by an external power supply. The IC chip that represents the DAC is usually one of many components mounted on a Printed Circuit Board (PCB), which is the I/O board or card. This board is plugged into a slot of the PC having DAQ. A typical commercial DAC is shown in Fig. 2.10.

2. Analog-to-Digital Converter (ADC) An analog-to-digital converter or ADC, on the other hand, converts an analog signal into the digital form, according to an appropriate code, before the same is used by a digital processor or a computer. The process of analog-to-digital conversion is more complex and time consuming than the digital-to-analog conversion. ADCs are usually more costly, and their conversion rate is slower than DACs. Several types of ADCs are commercially available. However, their principle of operation may vary depending on the type. A typical commercial ADC is shown in Fig. 2.8.

Note that the most fundamental property of any DAC or ADC is their number of bits for which it is designed, since this is a basic limit on resolution. Units of 8 to 12 bits are most common even though higher bits are also available. Both DAC and ADC are elements in typical input/output board (or I/O board, or data acquisition and control card, i.e., DAC or DAQ), which are usually situated on the same digital interface board.

4.5.5 Bridge Circuits

Various bridge circuits are employed widely for the measurement of resistance, capacitance, and inductance, as many transducers convert physical variables to these quantities. Figure 4.33 shows a purely resistive (Wheatstone) bridge in its simplest form. The basic principle of the bridge may be applied in two different ways, namely, the null method and the deflection method.

If the resistances are adjusted so that the bridge is balanced then there is

no voltage across AC, i.e., $v_{AC} = 0$. This happens when $R_1/R_4 = R_2/R_3$. Now if one of the resistors, say, R_1 , changes its resistance then there will be unbalance in the bridge and a voltage will appear across v_{AC} causing a meter reading. This meter reading is an indication of the change in R_1 of the transducer element, and can be utilized to compute the change. This method of measurement is called the *deflection method*. In the *null method*, one of the resistors is adjustable manually. Thus, if R_1 changes causing a meter deflection, R_2 can be adjusted manually till its effect just cancels that of R_1 and the bridge is returned to its balanced position. Here, the change in R_1 is directly related to the change in R_2 required to effect the balance. Note that both the deflection and null methods require a calibration curve so that one knows the numerical values of R_1 or R_2 that has caused the imbalance or balance, respectively. Note that the measurements of rapid dynamic phenomenon can be done using the deflection method. Moreover, based on the alternate current (ac) and direct current (dc) excitations in the bridge, there are ac bridges and dc bridges, respectively.

4.5.6 Signal Analyzer

Modern signal analyzers employ digital techniques of signal analysis to extract useful information that is carried by the signal. Digital Fourier analysis using Fast Fourier Transform (FFT) is perhaps the singlemost common procedure that is used in the vast majority of signal analyzers. Fourier analysis produces the frequency spectrum of a time signal, which is explained here in brief.

Any periodic signal $f(t)$ can be decomposed into a number of sines and cosines of different amplitudes, a_n and b_n , and frequencies $n\omega t$, for $n = 1, 2, \dots, \infty$, which is expressed as

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega t) + \sum_{n=1}^{\infty} b_n \sin(n\omega t) \quad (4.25)$$

If one adds sine and cosine functions together, the original signal can be reconstructed. Equation (4.25) is called a Fourier series, and the collection of different frequencies present in the equation is called the *frequency spectrum* or *frequency content* of the signal. Even though the signal is in amplitude-time domain, the frequency spectrum is in the amplitude-frequency domain. For example, for the function, $f(t) = \sin(t)$ of Fig. 4.34(a), which consists of only one frequency with

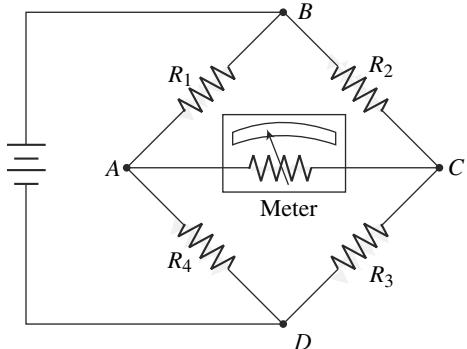


Fig. 4.33 Wheatstone bridge

constant amplitude, the plotted signal would be represented by a single line at the given frequency, as shown in Fig. 4.34(b). If the plot with given frequency and amplitude is represented by the arrow in Fig. 4.34(b), the same sine function can be reconstructed. The plots in Fig. 4.35 are similar and represent the following:

$$f(t) = \sum_{n=1,3,\dots,15} \frac{1}{n} \sin(nt) \quad (4.26)$$

where $\omega = 1$.

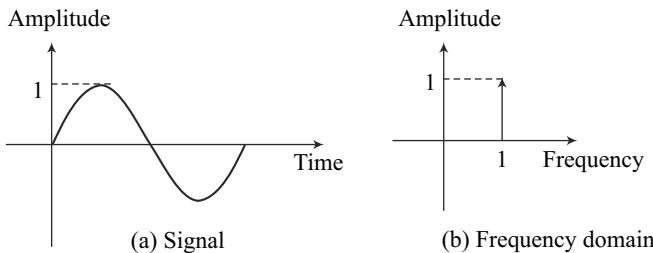


Fig. 4.34 Time domain vs. frequency domain

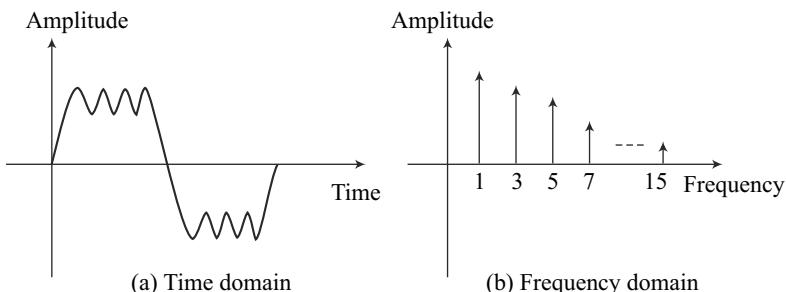


Fig. 4.35 Complex functions

The frequencies are also plotted in the frequency-amplitude domain. Clearly, when the number of frequencies contained in $f(t)$ increases, the summation becomes closer to a square function. Theoretically, to reconstruct a square wave from sine functions, an infinite number of sines must be added together. In practice, however, some of the major frequencies within the spectrum will have larger amplitudes. These major frequencies or harmonics are used in identifying and labeling a signal, including recognizing shapes, objects, etc.

4.6 SENSOR SELECTION

In using sensors, one must first decide what the sensor is supposed to do and what result one expects. A sensor detects the quantity to be measured (the *measurand*). The transducer converts the detected measurand into a convenient form for subsequent use, e.g., for control or actuation. The transducer signal may be filtered, amplified and suitably modified using the devices mentioned in Section 4.5. Selection of suitable sensors for robotic applications relies heavily on their performance specifications.

Majority of manufacturers provide what are actually static parameters. However, dynamic parameters are also important. In this section, static characteristics will be studied, whereas the dynamic characteristics can be described using the transient response of a system explained in Chapter 10 (Control), and the associated definitions appear therein, as a sensor is also a dynamic system whose behavior changes with the input signals. The following definitions will help a user or a designer select appropriate sensors for a robotic application.

1. Range Range or span is a measure of the difference between the minimum and maximum values of its input or output (response) so as to maintain a required level of output accuracy. For example, a strain gauge might be able to measure output values over the range from 0.1 to 10 Newtons.

2. Sensitivity Sensitivity is defined as the ratio of the change of output to change in input. As an example, if a movement of 0.025 mm in a linear potentiometer causes an output voltage by 0.02 volt then the sensitivity is 0.8 volts per mm. It is sometimes used to indicate the smallest change in input that will be observable as a change in output. Usually, maximum sensitivity that provides a linear and accurate signal is desired.

3. Linearity Perfect linearity would allow output versus input to be plotted as a straight line on a graph paper. Linearity is a measure of the constancy of the ratio of output to input. In the form of an equation, it is

$$y = mx \quad (4.27)$$

where x is input and y is output, and m is a constant. If m is a variable, the relationship is not linear. For example, m may be a function of x , such as $m = a + bx$ where the value of b would introduce a nonlinearity. A measure of the nonlinearity could be given as the value of b .

4. Response Time Response time is the time required for a sensor to respond completely to a change in input. The response time of a system with sensors is the combination of the responses of all individual components, including the sensor. An important aspect in selecting an appropriate sensor is to match its time response to that of the complete system. Associated definitions like rise time, peak time, settling time, etc., with regard to the dynamic response of a sensor are given in Chapter 10.

5. Bandwidth It determines the maximum speed or frequency at which an instrument associated with a sensor or otherwise is capable of operating. High bandwidth implies faster speed of response. Instrument bandwidth should be several times greater than the maximum frequency of interest in the input signals.

6. Accuracy Accuracy is a measure of the difference between the measured and actual values. An accuracy of ± 0.025 mm means that under all circumstances considered, the measured value will be within 0.025 mm of the actual value. In positioning a robot and its end-effector, verification of this level of accuracy would require careful measurement of the position of the end-effector with respect to the base reference location with an overall accuracy of 0.025 mm under all conditions of temperature, acceleration, velocity, and loading. Precision-measuring equipment,

carefully calibrated against secondary standards, would be necessary to verify this accuracy. Accuracy describes ‘closeness to true values.’

7. Repeatability and Precision Repeatability is a measure of the difference in value between two successive measurements under the same conditions, and is a far less stringent criterion than accuracy. As long as the forces, temperature, and other parameters have not changed, one would expect the successive values to be the same, however poor the accuracy is.

An associated definition is precision, which means the ‘closeness of agreement’ between independent measurements of a quantity under the same conditions without any reference to the true value, as done above. Note that the number of divisions on the scale of the measuring device generally affects the consistency of repeated measurement and, therefore, the precision. In a way, precision describes ‘repeatability.’ Figure 4.36 illustrates the difference between accuracy and precision.

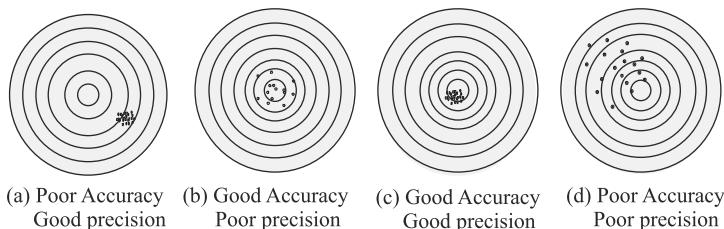


Fig. 4.36 Interpretation of accuracy and precision

8. Resolution and Threshold Resolution is a measure of the number of measurements within a range from minimum to maximum. It is also used to indicate the value of the smallest increment of value that is observable, whereas threshold is a particular case of resolution. It is defined as the minimum value of input below which no output can be detected.

9. Hysteresis It is defined as the change in the input/output curve when the direction of motion changes, as indicated in Fig. 4.37. This behavior is common in loose components such as gears, which have backlash, and in magnetic devices with ferromagnetic media, and others.

10. Type of Output Output can be in the form of a mechanical movement, an electrical current or voltage, a pressure, or liquid level, a light intensity, or another form. To be useful, it must be converted to another form, as in the LVDT (Linear Variable Differential Transducer) or strain gauges, which are discussed earlier.

In addition to the above characteristics, the following considerations must also be made while selecting a sensor.

11. Size and Weight Size and weight are usually important physical characteristics of sensors. If the sensor is to be mounted on the robot hand or arm, it becomes a part

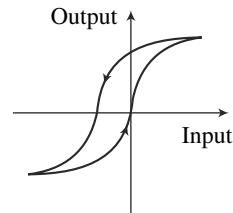


Fig. 4.37 Hysteresis

of the mass that must be accelerated and decelerated by the drive motors of the wrist and arm. So, it directly affects the performance of the robot. It is a challenge to sensor designers to reduce size and weight. An early wrist force-torque sensor, for example, was about 125 mm in diameter but was reduced to about 75 mm in diameter through careful redesign.

12. Environmental Conditions Power requirement and its easy availability should be considered. Besides, conditions like chemical reactions including corrosion, extreme temperatures, light, dirt accumulation, electromagnetic field, radioactive environments, shock and vibrations, etc., should be taken into account while selecting a sensor or considering how to shield them.

13. Reliability and Maintainability Reliability is of major importance in all robot applications. It can be measured in terms of Mean Time To Failure (MTTF) as the average number of hours between failures that cause some part of the sensor to become inoperative. In industrial use, the total robot system is expected to be available as much as 98 or 99% of the working days. Since there are hundreds of components in a robot system, each one must have a very high reliability. Some otherwise good sensors cannot stand the daily environmental stress and, therefore, cannot be used with robots. Part of the requirement for reliability is ease of maintenance. A sensor that can be easily replaced does not have to be as reliable as one that is hidden in the depths of the robot. Maintainability is a measure in terms of Mean Time To Repair (MTTR).

14. Interfacing Interfacing of sensors with signal-conditioning devices and the controller of the robot is often a determining factor in the usefulness of sensors. Nonstandard plugs or requirements for nonstandard voltages and currents may make a sensor too complex and expensive to use. Also, the signals from a sensor must be compatible with other equipment being used if the system is to work properly.

15. Others Other aspects like initial cost, maintenance cost, cost of disposal and replacement, reputation of manufacturers, operational simplicity, ease of availability of the sensors and their spares should be taken into account. In many occasions, these nontechnical considerations become the ultimate deciding factor in the selection of sensors for an application.

SUMMARY

Necessity of sensors in robotics, different types of sensors used in robotic applications and the selection criteria are presented in this chapter. Functional aspects, rather than their mathematical models, are emphasized so that the reader is able to decide which sensor to use and when. Different sensors are classified from their functional uses, for example, position, velocity, acceleration, force, sensors, etc. Special attention has been paid to vision system, its elements, how it can be processed, etc. Signal-conditioning units, their characteristics, and when to use what are also explained. Finally guidelines are provided to select appropriate sensors.

EXERCISES

- 4.1** Define sensitivity and linearity.
- 4.2** Distinguish accuracy from repeatability.
- 4.3** State the physical characteristics in sensor selection.
- 4.4** What are the essential components of a sensor?
- 4.5** Why are the terms internal and external used to classify sensors?
- 4.6** Classify internal sensors.
- 4.7** Name some velocity sensors?
- 4.8** Why are position sensors not preferred to obtain velocity and acceleration?
- 4.9** Is there any advantage of external sensors over internal types?
- 4.10** Name some contact and noncontact type sensors.
- 4.11** What are the advantages of capacitive proximity sensors?
- 4.12** What is a machine vision?
- 4.13** What are the major components in a vision system?
- 4.14** What is the sensor component in a vision system?
- 4.15** What are different types of cameras? Explain the vidicon camera.
- 4.16** What are the typical difficulties in a vision system?
- 4.17** What is thresholding in vision?
- 4.18** Why is signal processing important?
- 4.19** Name at least three devices used in signal conditioning.
- 4.20** Define different types of filters.
- 4.21** What are the characteristics one should check while selecting a sensor?
- 4.22** What is Hough transform?
- 4.23** What are the static characteristics a sensor should have?

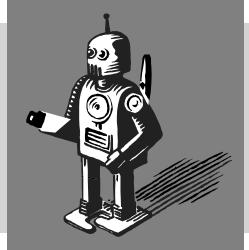
WEB-BASED EXERCISES

Based on Web search, find the answers for the following questions:

- 4.24** Name at least five commercial companies who make different types of sensors. List the sensors against each company's name.
- 4.25** What are the typical prices of encoders, tachogenerators, and accelerometers?
- 4.26** What is the price of a CCD camera?
- 4.27** Name two camera companies and mention which camera you will select for industrial robot system involved in accurate pick-and-place operation?
- 4.28** Name a robot manufacturing company who implements vision system in their robots.
- 4.29** Give a typical significance of a force/torque sensor.
- 4.30** What command MATLAB uses for digital filtering and image processing?

5

Transformations



For the purpose of controlling a robot, it is necessary to know the relationships between the joints' motion (input) and the end-effector motion (output), because the joint motions control the end-effector's movements. Thus, the study of kinematics is important, where transformations between the coordinate frames attached to different links of the robot need to be performed.

5.1 ROBOT ARCHITECTURE

A robot under study is made up of several links connected serially by joints. The robot's Degree Of Freedom (DOF) depends on the number of links and joints, their types, and the kinematic chain of the robot.

5.1.1 Links and Joints

The individual bodies that make up a robot are called *links*. Here, unless otherwise stated, all links are assumed to be rigid, i.e., the distance between any two points within the body does not change while it is moving. A rigid body in the three-dimensional Cartesian space has six DOF. This implies that the position of the body can be described by three translational, and the orientation by three rotational coordinates. For convenience, certain nonrigid bodies, such as chains, cables, or belts, which when serve the same function as rigid bodies, may be considered as links. From the kinematic point of view, two or more members when connected together without any relative motion between them are considered a single rigid link. For example, an assembly of two gears connected by a common shaft is treated as one link.

Links of a robot are coupled by kinematic pairs or joints. A *joint* couples two links and provides physical constraints on the relative motion between the links. It is not a physical entity but just a concept that allows one to specify how one link moves with respect to another one. For example, a hinge joint of a door allows it to move relative to the fixed wall about an axis. No other motion is possible. The type of relative motion permitted by a joint is governed by the form of the contact surface between the members, which can be a surface, a line, or a point. Accordingly, they are termed as either *lower* or *higher* pair joint. If two mating links are in surface contact, the joint is referred to as a lower pair joint. On the contrary, if the links are in line or point contact, the joint is called higher pair joint. As per the definition, the hinge joint of the door is a lower pair joint, whereas a ball rolling on a plane makes a higher pair joint. Frequently used lower and higher pairs are listed in Table 5.1.

1. Revolute Joint, R A revolute joint, also known as a *turning pair* or a *hinge* or a *pin joint*, permits two paired links to rotate with respect to each other about the axis of the joint, say, Z, as shown in Fig. 5.1. Hence, a revolute joint imposes five constraints, i.e., it prohibits one of the links to translate with respect to the other one along the three perpendicular axes, X, Y, Z, along with the rotation about two axes, X and Y. This joint has one degree of freedom (DOF).

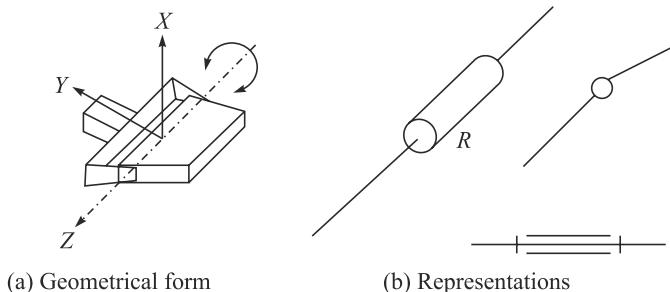


Fig. 5.1 A revolute joint

2. Prismatic Joint, P A *prismatic joint* or a *sliding pair* allows two paired links to slide with respect to each other along its axis, as shown in Fig. 5.2. It also imposes five constraints and, hence, has one DOF.

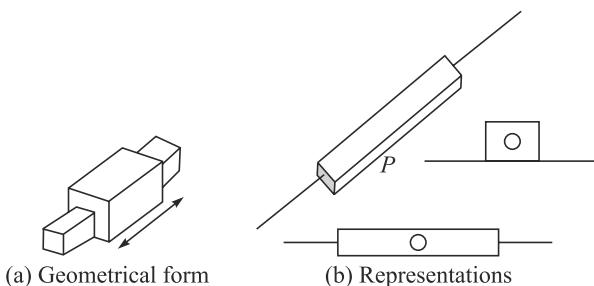


Fig. 5.2 A prismatic joint

3. Helical Joint, H As shown in Fig. 5.3, a helical joint allows two paired links to rotate about and translate at the same time along the axis of the joint. The translation is, however, not independent. It is related to the rotation by the pitch of the screw. Thus, the helical joint also has five constraints, and accordingly one DOF.

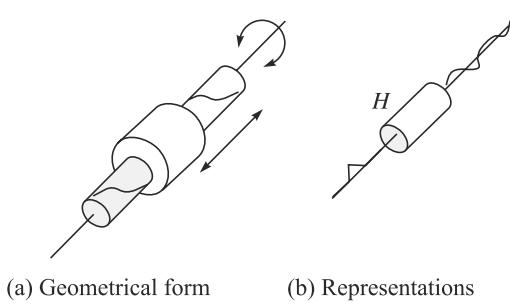


Fig. 5.3 A helical joint

4. Cylindrical Joint, C It permits rotation about, and independent translation along, the axis of the joint, as shown in Fig. 5.4. Hence, a cylindrical joint imposes four constraints on the paired links, and has two DOF.

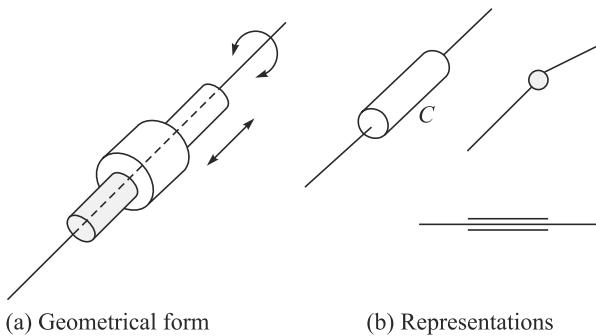


Fig. 5.4 A cylindrical joint

5. Spherical Joint, S It allows one of the coupled links to rotate freely in all possible orientation with respect to the other one about the center of a sphere. No relative translation is permitted. Hence, it imposes three constraints and has three DOF.

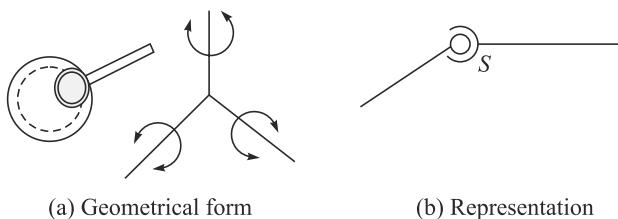


Fig. 5.5 A spherical joint

6. Planar Joint, L This three DOF joint allows two translations along the two independent axes of the plane of contact and one rotation about the axis normal to the plane, Fig. 5.6.

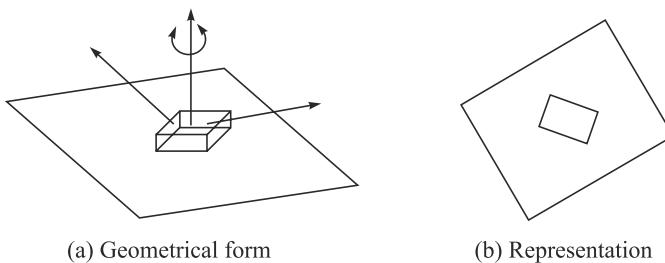
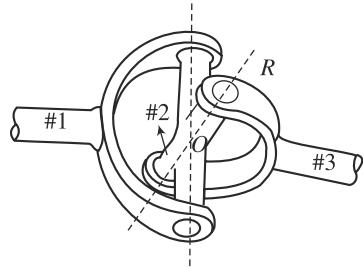


Fig. 5.6 A planar joint

Table 5.1 Lower pair joints

Name	Symbol	Geometric Form and Representations	DOF	Common Surface
Revolute	R	Fig. 5.1	1	Cylinder
Prismatic	P	Fig. 5.2	1	Prism
Helical	H	Fig. 5.3	1	Screw
Cylindrical	C	Fig. 5.4	2	Cylinder
Spherical	S	Fig. 5.5	3	Sphere
Planar	L	Fig. 5.6	3	Plane

Table 5.1 summarizes the basic lower pair joints, where all the joints have surface contact between the interconnecting links. Another commonly used lower pair joint in robotics is the two DOF *universal joint*, as shown in Fig. 5.7. This is the combination of two intersecting revolute joints. Examples of higher pair joints in robots are *gears* and *cams with roller followers*, where they make line contacts.

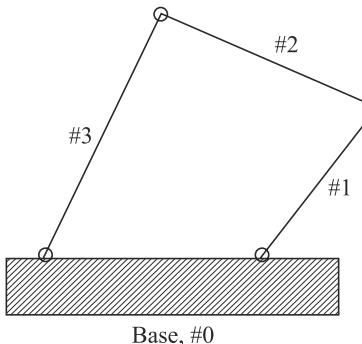
**Fig. 5.7** A universal joint

5.1.2 Kinematic Chain

A *kinematic chain* is a series of links connected by joints. When each and every link in a kinematic chain is coupled to at most two other links, the chain is referred to as *simple kinematic chain*. A simple kinematic chain can be either *closed* or *open*. It is closed if each and every link is coupled to two other links as shown in Fig. 5.8. A kinematic chain is open if it contains exactly two links, namely, the end ones that are coupled to only one link. A robotic manipulator shown in Fig. 5.9 falls in this category.

Mechanism vs. Manipulator

Whereas a mechanism can be open- and closed-chain mechanical system; a manipulator is treated here as an open-chain system.

**Fig. 5.8** A four-bar mechanism

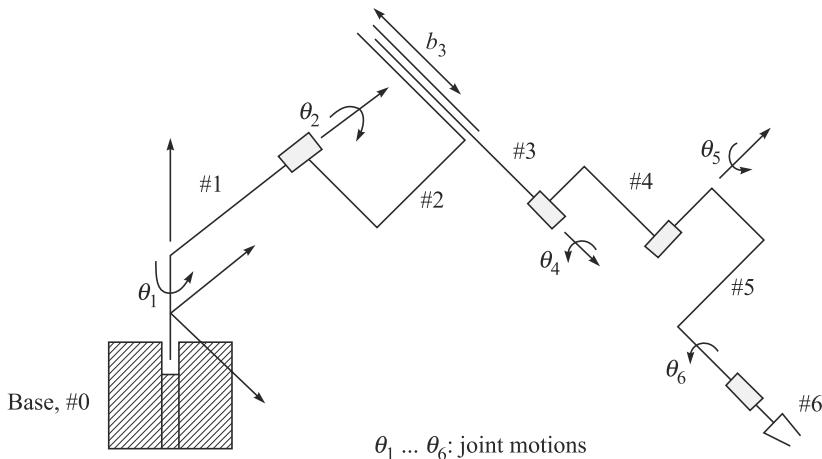


Fig. 5.9 A robot manipulator

5.1.3 Degree of Freedom

Formally, the degree of freedom (DOF) of a mechanical system is defined as the number of independent coordinates or minimum coordinates required to fully describe its pose or configuration. Thus, a rigid body moving in the three-dimensional Cartesian space has six DOF, three each for position and orientation. Several methodologies exist to determine the DOF. One such method was given by Grubler in 1917 for planar mechanisms, which was later generalized by Kutzbach in 1929 for spatial mechanisms. Together they are known as Grubler Kutzbach criterion, which is derived below:

Assume,

- s : dimension of working space (for planar, $s = 3$; spatial, $s = 6$)
- r : number of rigid bodies or links in the system
- p : number of kinematic pairs or joints in the system
- c_i : number of constraints imposed by each joint
- c : total number of constraints imposed by p joints
- n_i : relative degree of freedom of each joint
- n : degree of freedom (DOF) of the whole system

The DOF of the whole system n is then determined from the total DOF associated with all the moving links minus the total number of constraints imposed by all the joints, i.e.,

$$n = s(r - 1) - c, \text{ where } c \equiv \sum_{i=1}^p c_i \quad (5.1)$$

Note that -1 in Eq. (5.1) corresponds to the fixed link of the system which has zero DOF. Moreover, for the coupled links, the sum of the number of constraints imposed by each joint, c_i , and the relative degree of freedom permitted by that joint, n_i , is

equal to the dimension of the working space, s , i.e., $s = c_i + n_i$. Hence, the total number of constraints imposed by all the joints can be re-written as

$$c = \sum_{i=1}^p c_i = \sum_{i=1}^p (s - n_i) = sp - \sum_{i=1}^p n_i \quad (5.2)$$

Upon substitution of Eq. (5.2) into Eq. (5.1), one obtains the DOF of the system, n , as

$$n = s(r - p - 1) + \sum_i^n \quad (5.3a)$$

which is the well-known Grubler Kutzbach criterion for the determination of a system's DOF. A simple and easy-to-use form of Eq. (5.3a) for the planar and spatial motions of a system consisting of only one DOF kinematic pairs, e.g., revolute or prismatic, can be given by

$$n = 3(r - 1) - 2p: \text{For planar; } n = 6(r - 1) - 5p: \text{For spatial} \quad (5.3b)$$

Note that the link dimensions are not entered in the formula, which sometimes affect the DOF of the mechanism due to its geometrical arrangement of the links.

Example 5.1 Four-bar Mechanism

Figure 5.8 shows a planar four-bar mechanism whose links, including the ground, are connected in a loop by four revolute joints. Here, $s = 3$, $r = 4$, and $p = 4$. Equation (5.3a) gives

$$n = 3(4 - 4 - 1) + (1 + 1 + 1 + 1) = 1 \quad (5.4)$$

Hence, the planar four-bar linkage has one DOF. One can easily verify the result of Eq. (5.4) using the expression of Eq. (5.3b) for planar motion too.

Example 5.2 A Robot Manipulator

Figure 5.9 shows the schematic diagram of a Stanford robot (Angeles, 2003). Its DOF can be determined using Eq. (5.3a) as $s = 6$, $r = 7$ (including the fixed base), $p = 6$. Then, from Eq. (5.3a), the DOF of the robot is

$$n = 6(7 - 6 - 1) + 6 \times 1 = 6 \quad (5.5)$$

One can easily verify the result of Eq. (5.5) using the expression of Eq. (5.3b) for spatial motion too.

Example 5.3 Five-bar Mechanism

Figure 5.10 shows a planar five-bar mechanism. There are five links connected in a loop by five revolute joints. Hence, $s = 3$, $r = 5$ (including the base), and $p = 5$. Equation (5.3a) yields

$$n = 3(5 - 5 - 1) + 5 \times 1 = 2 \quad (5.6)$$

The planar five-bar mechanism has two DOF, which is often used as a robot manipulator also.

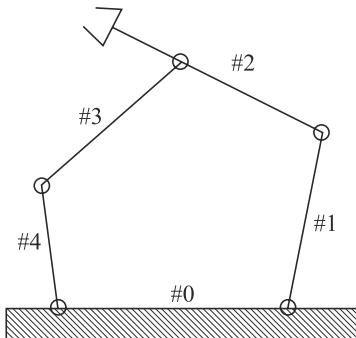


Fig. 5.10 A five-bar mechanism

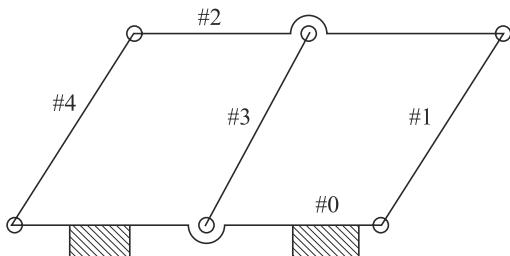


Fig. 5.11 A double parallelogram mechanism

Example 5.4 A Double Parallelogram Mechanism

Figure 5.11 shows a double parallelogram mechanism, where $s = 3$, $r = 5$ (including the fixed base), $p = 6$. Then, Eq. (5.3a), gives

$$n = 3(5 - 6 - 1) + 6 \times 1 = 0 \quad (5.7)$$

which interprets the system as a structure with zero DOF. This is not true, as the mechanism is known to have one DOF. Actually, the link #5 has length equal to that of link #2. Hence, #5 is kinematically redundant and can be removed without destroying the basic kinematic chain and subsequently the input-output behavior of the mechanism.

In general, if the Grubler–Kutzbach criterion, Eq. (5.3), yields $n > 0$, the mechanism has n DOF. If the criterion yields $n = 0$, it is a structure with zero DOF, and if $n < 0$, the system is a statically indeterminate structure with redundant constraints. Exceptions, however, occur as the one shown for Fig. 5.11 which does not obey the Grubler–Kutzbach criterion. For such systems, different mobility criterion, e.g., loop mobility criterion, as described by Tsai (1999), etc., can be employed to determine their DOF.

5.2 POSE OF A RIGID BODY

Rigid-body motion in the three-dimensional Cartesian space comprises of translation and rotation. Whereas translation is defined using three Cartesian coordinates, the rotation needs three angular coordinates. Hence, the rigid-body motion can be defined completely using six coordinates. In the study of the kinematics of robot manipulators, one constantly deals with the position and orientation of several bodies in space. The bodies of interest include the links of the manipulator, tools, and workpiece. To identify the position and orientation of a body, i.e., its *pose* or *configuration*, a fixed reference coordinate system is established, which is called the *fixed frame*. Next, a Cartesian coordinate system attached to the moving body is employed to describe its pose.

Pose vs. Configuration

They mean the same in this book, i.e., the position of a point on a rigid body and its orientation.

The pose or the position and orientation of a rigid body with respect to the reference coordinate system are known from the six independent parameters. As shown in Fig. 5.12, let the *X-Y-Z*-coordinate system be the *fixed reference frame*. The *U-V-W*-coordinate system is attached to the moving body and referred as the *moving frame*. Clearly, the pose or configuration of the rigid body is known if the pose of the moving frame with respect to the fixed frame is known. This pose is determined from the position of any point on it, say, the origin, *O*, or point *P*, and the orientation of the moving frame with respect to the fixed frame.

5.2.1 Position Description

The position of any point, *P*, on a rigid body in motion with respect to the fixed reference frame can be described by the three-dimensional Cartesian vector, \mathbf{p} , as indicated in Fig. 5.12. If the coordinates of the point *P* or the components of the vector \mathbf{p} are, p_x, p_y, p_z , in the fixed frame *F*, it is denoted as

$$[\mathbf{p}]_F \equiv \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (5.8)$$

where the subscript *F* stands for the reference frame where the vector \mathbf{p} is represented.

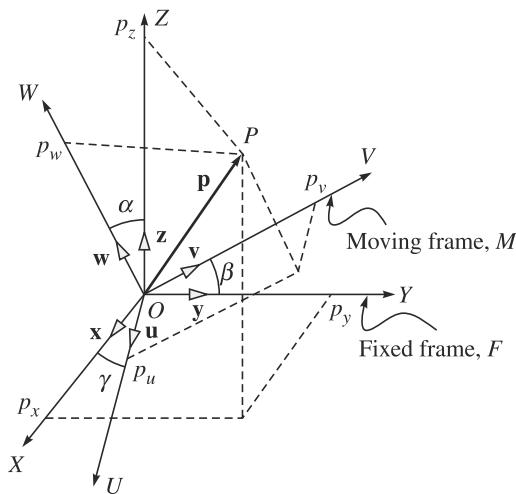


Fig. 5.12 Spatial description

The subscripts *x*, *y* and *z* represent the projections of the position vector \mathbf{p} onto the coordinate axes of the fixed reference frame, namely, along *X*, *Y* and *Z*, respectively. Vector \mathbf{p} can alternatively be expressed as

$$\mathbf{p} = p_x \mathbf{x} + p_y \mathbf{y} + p_z \mathbf{z} \quad (5.9)$$

where \mathbf{x} , \mathbf{y} and \mathbf{z} denote the unit vectors along the axes, *X*, *Y* and *Z* of the frame *F*, respectively, as indicated in Fig. 5.12. Their representations in the frame *F*, namely, $[\mathbf{x}]_F$, $[\mathbf{y}]_F$ and $[\mathbf{z}]_F$, are as follows:

$$[\mathbf{x}]_F \equiv \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, [\mathbf{y}]_F \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{ and } [\mathbf{z}]_F \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (5.10)$$

Substituting Eq. (5.10) into Eq. (5.9), it can be shown that the expression of the vector \mathbf{p} in the frame F , i.e., $[\mathbf{p}]_F$, is same as that of Eq. (5.8). Note that if the vector \mathbf{p} is represented in another fixed frame different from the frame F then the vector \mathbf{p} will have different components along the new coordinate axes even though the actual position of the point P has not changed. Thus, a vector will normally be written without mentioning any frame, e.g., as done in Eq. (5.9) which, in contrast to Eqs (5.8) and (5.10), is termed *frame invariant representation*, i.e., independent of the choice of any particular reference frame. It is, however, important to note that when a numerical problem is solved, one must choose a suitable coordinate frame.

Example 5.5 Position of a Point in the Fixed Frame

If coordinates of a point P in the fixed coordinate frame are $p_x = 3$, $p_y = 4$, and $p_z = 5$ then the position vector \mathbf{p} from Eq. (5.8) can be given by

$$[\mathbf{p}]_F \equiv \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

5.2.2 Orientation Description

The orientation of a rigid body with respect to the fixed frame can be described in different ways. For example,

1. Direction cosine representation
2. Fixed-axes rotations
3. Euler-angles representation
4. Single- and double-axes rotations
5. Euler parameters, and others

Each one has its own limitations. If necessary, one may switch from one to other representation during a robot's motion to avoid the former's limitations. These representations are explained below.

1. Direction Cosine Representation To describe the orientation or rotation of a rigid body, consider the motion of a moving frame M with respect to a fixed frame F , with one point fixed, say, the origin of the fixed frame, O , as shown in Fig. 5.12. Let \mathbf{u} , \mathbf{v} and \mathbf{w} denote the three unit vectors pointing along the coordinates axes, U , V and W of the moving frame M , respectively, similar to the unit vectors, \mathbf{x} , \mathbf{y} , and \mathbf{z} , along X , Y and Z of the fixed frame F , respectively. Since each of the unit vectors, \mathbf{u} , \mathbf{v} or \mathbf{w} , denotes the position of a point at a unit distance from the origin on the axes of frame M , they are expressed using their projections on the X , Y , Z axes of the frame F as

$$\mathbf{u} = u_x \mathbf{x} + u_y \mathbf{y} + u_z \mathbf{z} \quad (5.11a)$$

$$\mathbf{v} = v_x \mathbf{x} + v_y \mathbf{y} + v_z \mathbf{z} \quad (5.11b)$$

$$\mathbf{w} = w_x \mathbf{x} + w_y \mathbf{y} + w_z \mathbf{z} \quad (5.11c)$$

where u_x , u_y and u_z are the components of the unit vector, \mathbf{u} , along X , Y and Z axes, respectively. Similarly, v_x , v_y , v_z , and w_x , w_y , w_z are defined for the unit vectors \mathbf{v} and \mathbf{w} , respectively. Now, the point P of the rigid body, as shown in Fig. 5.12 and given by Eq. (5.8), is expressed in the moving frame, M , as

$$\mathbf{p} = p_u \mathbf{u} + p_v \mathbf{v} + p_w \mathbf{w} \quad (5.12)$$

where p_u , p_v , and p_w are the components of the vector, \mathbf{p} , along U , V , W axes of the moving frame, M . Substitution of Eqs (5.11a-c) into Eq. (5.12) yields

$$\mathbf{p} = (p_u u_x + p_v v_x + p_w w_x) \mathbf{x} + (p_u u_y + p_v v_y + p_w w_y) \mathbf{y} + (p_u u_z + p_v v_z + p_w w_z) \mathbf{z} \quad (5.13)$$

Comparing the right-hand sides of Eqs. (5.8) and (5.13), the following identities are obtained:

$$p_x = u_x p_u + v_x p_v + w_x p_w \quad (5.14a)$$

$$p_y = u_y p_u + v_y p_v + w_y p_w \quad (5.14b)$$

$$p_z = u_z p_u + v_z p_v + w_z p_w \quad (5.14c)$$

Equations (5.14a-c) are written in a matrix form as

$$[\mathbf{p}]_F = \mathbf{Q} [\mathbf{p}]_M \quad (5.15)$$

where $[\mathbf{p}]_F$ and $[\mathbf{p}]_M$ are the representations of the three-dimensional vector \mathbf{p} in frames F and M , respectively, and \mathbf{Q} is the 3×3 rotation or orientation matrix transforming the representation of the vector \mathbf{p} from the frame M to F . They are given as follows:

$$[\mathbf{p}]_F \equiv \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}, [\mathbf{p}]_M \equiv \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}, \text{ and } \mathbf{Q} \equiv \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} = \begin{bmatrix} \mathbf{u}^T \mathbf{x} & \mathbf{v}^T \mathbf{x} & \mathbf{w}^T \mathbf{x} \\ \mathbf{u}^T \mathbf{y} & \mathbf{v}^T \mathbf{y} & \mathbf{w}^T \mathbf{y} \\ \mathbf{u}^T \mathbf{z} & \mathbf{v}^T \mathbf{z} & \mathbf{w}^T \mathbf{z} \end{bmatrix} \quad (5.16)$$

Note the columns of the matrix \mathbf{Q} . They are nothing but the components of the orthogonal (meaning 90° to each other) unit vectors \mathbf{u} , \mathbf{v} , and \mathbf{w} in the fixed frame F , which must satisfy the following six orthogonal conditions:

$$\mathbf{u}^T \mathbf{u} = \mathbf{v}^T \mathbf{v} = \mathbf{w}^T \mathbf{w} = 1, \text{ and } \mathbf{u}^T \mathbf{v} (\equiv \mathbf{v}^T \mathbf{u}) = \mathbf{u}^T \mathbf{w} (\equiv \mathbf{w}^T \mathbf{u}) = \mathbf{v}^T \mathbf{w} (\equiv \mathbf{w}^T \mathbf{v}) = 0 \quad (5.17)$$

Moreover, for the three orthogonal vectors, \mathbf{u} , \mathbf{v} and \mathbf{w} , the following hold true:

$$\mathbf{u} \times \mathbf{v} = \mathbf{w}, \mathbf{v} \times \mathbf{w} = \mathbf{u}, \text{ and } \mathbf{w} \times \mathbf{u} = \mathbf{v} \quad (5.18)$$

Hence, the 3×3 rotation matrix \mathbf{Q} , denoting the orientation of the moving frame, M , with respect to the fixed frame, F , is called *orthogonal*. It satisfies the following properties due to Eq. (5.17):

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{Q} \mathbf{Q}^T = \mathbf{1}; \text{ where } \det(\mathbf{Q}) = 1, \text{ and } \mathbf{Q}^{-1} = \mathbf{Q}^T \quad (5.19)$$

The term $\mathbf{1}$ denotes the 3×3 identity matrix. Moreover, if one is interested to find the rotation description of the frame, F , with respect to the frame M denoted by \mathbf{Q}' , it can be derived similarly. It can be shown that $\mathbf{Q}' = \mathbf{Q}^T$. Also note from Eq. (5.16), that the (1, 1) element of \mathbf{Q} is the cosine of the angle between the vectors \mathbf{u} and \mathbf{x} , i.e., $\mathbf{u}^T \mathbf{x}$. The same holds true with other elements of \mathbf{Q} . Hence, this rotation matrix is known as the *direction cosine representation* of the rotation matrix. Such representation requires nine parameters, namely, the elements of the 3×3 matrix, \mathbf{Q} . However, not all the nine parameters are independent as they must satisfy the six

conditions of Eq. (5.17). Thus, only three parameters are independent and should be sufficient to define the three DOF rotational motion. It is, however, difficult to choose the set of three independent parameters. This is the drawback of the direction cosine representation.

Example 5.6 Elementary Rotations

Suppose that a reference frame M coincides with the fixed frame F . Now, the frame M is rotated by an angle α about the axis Z , as shown in Fig. 5.13(a). The unit vectors of the new frame M can be described in terms of their components in the reference frame F as

$$[\mathbf{u}]_F \equiv \begin{bmatrix} C\alpha \\ S\alpha \\ 0 \end{bmatrix}, [\mathbf{v}]_F \equiv \begin{bmatrix} -S\alpha \\ C\alpha \\ 0 \end{bmatrix}, \text{ and } [\mathbf{w}]_F \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (5.20)$$

where, $S \equiv \sin$; $C \equiv \cos$. Hence, the rotation matrix denoted by \mathbf{Q}_Z is given by

$$\mathbf{Q}_Z \equiv \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.21)$$

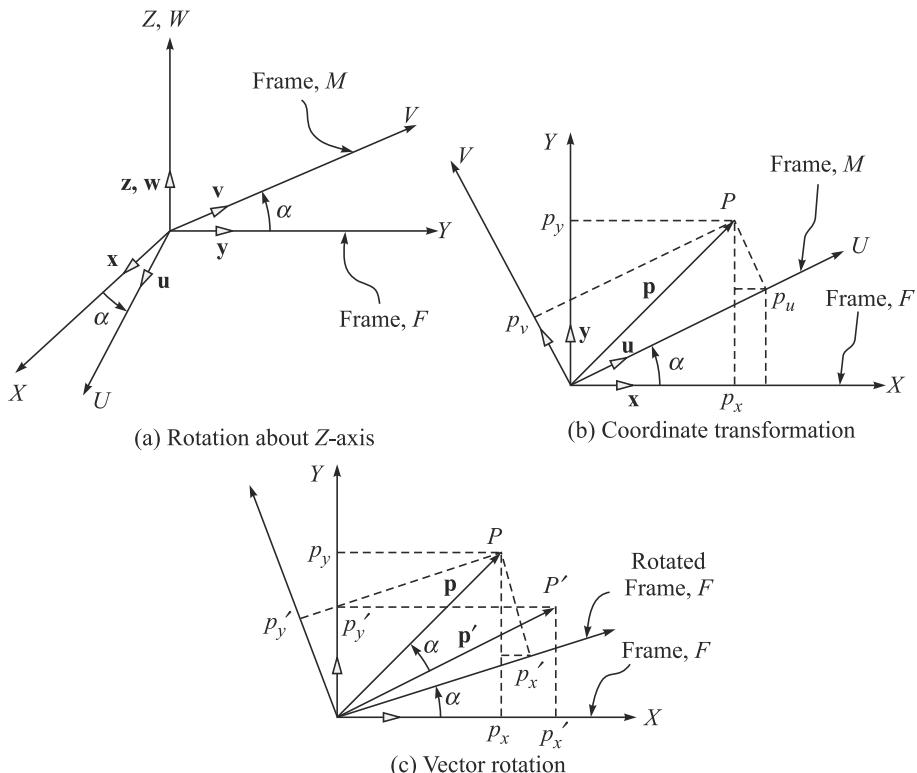


Fig. 5.13 Alternative interpretations of rotation matrix

In a similar manner, it can be shown that the rotations of angle β about the axis Y , and by an angle γ about the axis X are, respectively, given by

$$\mathbf{Q}_Y \equiv \begin{bmatrix} C\beta & 0 & S\beta \\ 0 & 1 & 0 \\ -S\beta & 0 & C\beta \end{bmatrix}, \text{ and } \mathbf{Q}_X \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\gamma & -S\gamma \\ 0 & S\gamma & C\gamma \end{bmatrix} \quad (5.22)$$

The above matrices in Eqs. (5.21–5.22) are called the *elementary rotations* and are useful to describe any arbitrary rotation while the angles with respect to the coordinate axes are known.

Example 5.7 Properties of Elementary Rotation Matrices

From Eqs. (5.21–5.22), matrix multiplication of, say, \mathbf{Q}_z^T and \mathbf{Q}_z results in the following:

$$\mathbf{Q}_z^T \mathbf{Q}_z = \begin{bmatrix} C\alpha & S\alpha & 0 \\ -S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using the definition of determinant given in Appendix A, it can be easily calculated that $\det(\mathbf{Q}_z) = 1$. Hence, \mathbf{Q}_z satisfies both the properties of a rotation matrix. Similarly, the other two matrices can be shown to have the properties of a rotation matrix as well.

Example 5.8 Coordinate Transformation

Consider two coordinate frames with a common origin. One of them is rotated by an angle α about the axis Z . Let $[\mathbf{p}]_F$ and $[\mathbf{p}]_M$ be the vector representations of the point P in frames F and M , respectively, as shown in Fig. 5.13(b). On the basis of simple geometry, the relationships between the coordinates of the point P in the two coordinate frames are given by

$$p_x = p_u C\alpha - p_v S\alpha \quad (5.23)$$

$$p_y = p_u S\alpha + p_v C\alpha \quad (5.24)$$

$$p_z = p_w \quad (5.25)$$

where p_x, p_y, p_z and p_u, p_v, p_w are the three coordinates of the point P along the axes of frames F and M , respectively. It is easy to recognize from Eqs. (5.23–5.25) that the vector $[\mathbf{p}]_F \equiv [p_x, p_y, p_z]^T$ is nothing but

$$[\mathbf{p}]_F = \mathbf{Q}_Z [\mathbf{p}]_M \quad (5.26)$$

where $[\mathbf{p}]_M \equiv [p_u, p_v, p_w]^T$ and \mathbf{Q}_Z is given by Eq. (5.21). Matrix \mathbf{Q}_Z not only represents the orientation of the frame M with respect to the fixed frame F , as in Fig. 5.13(a), but also transforms the representation of a vector from frame M , say, $[\mathbf{p}]_M$, to another frame F with the same origin, i.e., $[\mathbf{p}]_F$.

Example 5.9 Vector Rotation

Consider the vector \mathbf{p} which is obtained by rotating a vector \mathbf{p}' in the X - Y -plane by an angle α about the axis Z of the reference frame, F of Fig. 5.13(c). Let p'_x, p'_y, p'_z be the coordinates of the vector \mathbf{p}' in the frame F , i.e., $[\mathbf{p}']_F \equiv [p'_x, p'_y, p'_z]^T$. Then, the vector \mathbf{p} in the frame F , $[\mathbf{p}]_F$, has the following components:

$$p_x = p'_x C\alpha - p'_y S\alpha \quad (5.27)$$

$$p_y = p'_x S\alpha + p'_y C\alpha \quad (5.28)$$

$$p_z = p'_z \quad (5.29)$$

which can be obtained by rotating the fixed frame F with which the vector \mathbf{p}' is attached to by an angle α counter-clockwise about the Z -axis so that the vector \mathbf{p}' reaches \mathbf{p} , as indicated in Fig. 5.13(c). Equations (5.27–5.29) can be written in compact form as

$$[\mathbf{p}]_F = \mathbf{Q}_Z [\mathbf{p}']_F \quad (5.30)$$

where \mathbf{Q}_Z is given by Eq. (5.21).

2. Fixed-axes Rotations Rotations about three fixed axes of an inertial coordinate frame, denoted with frame F , to represent the orientation of a frame, say, M , using three angles are conceptually simple because a reader finds it similar to three translations along three fixed axes of F . However, there is an important difference between the translations and rotations. The latter does not conform to the commutative property, as pointed out in Section 5.2.3 and shown in Figs 5.20–5.21. The fixed-axes rotations, unlike direction cosine representation, however, constitute a *minimal* set representation of the orientation. The overall orientation of M with respect to F is obtained by composing three elementary rotations with respect to the axes of fixed frames, namely, X , Y and Z of the frame F shown in Fig. 5.14.

Roll, Pitch and Yaw

Rotations about fixed X , Y , and Z -axes are also referred to as roll, pitch and yaw angles, respectively. This nomenclature is commonly used in aeronautical engineering.

The XYZ Fixed-Axes Set For the rotations about fixed X , Y , and Z -axes, one can reach the frame M from the fixed frame F through a series of elementary rotations about individual X -, Y - and Z -axis, as shown in Fig. 5.14. Let ψ , θ , and ϕ be the angles about X -, Y -, and Z -axis, respectively. The overall rotation described by these angles is then obtained as the composition of elementary rotations, as explained below. Referring to Fig. 5.14,

- Rotate the fixed frame F by the angle ψ about the axis X , as indicated in Fig. 5.14(a). This rotation is described by the rotation matrix \mathbf{Q}_X , similar to the one derived in Eq. (5.22), i.e.,

$$\mathbf{Q}_X \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\psi & -S\psi \\ 0 & S\psi & C\psi \end{bmatrix} \quad (5.31a)$$

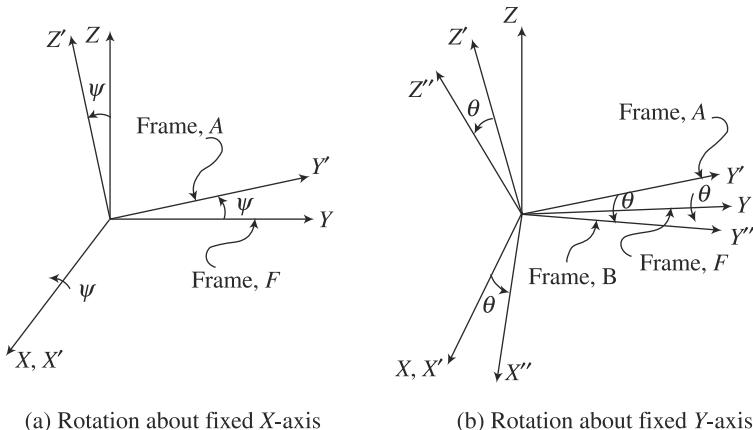


Fig. 5.14 Rotations about fixed XYZ-axes

- Rotate the current frame A by an angle θ about its Y -axis, Fig. 5.14(b). This rotation is denoted by \mathbf{Q}_Y and described by the rotation matrix, \mathbf{Q}_Y of Eq. (5.22), i.e.,

$$\mathbf{Q}_Y \equiv \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \quad (5.31b)$$

- Rotate the current frame B by an angle ϕ about its Z -axis, Fig. 5.14(c). This rotation is denoted by \mathbf{Q}_Z and described by the rotation matrix, \mathbf{Q}_Z of Eq. (5.21), i.e.,

$$\mathbf{Q}_Z \equiv \begin{bmatrix} C\phi & -S\phi & 0 \\ S\phi & C\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.31c)$$

The resulting orientation of the frame M , denoted by \mathbf{Q} , is obtained from the composition of the three elementary rotations. The resulting orientation matrix is obtained via pre-multiplications of the successive rotation matrices, i.e.,

$$\mathbf{Q} = \mathbf{Q}_Z \mathbf{Q}_Y \mathbf{Q}_X \quad (5.31d)$$

whose elements are computed below:

$$\mathbf{Q} \equiv \begin{bmatrix} C\theta C\phi & S\theta S\psi C\phi - C\psi S\phi & S\theta C\psi C\phi + S\psi S\phi \\ C\theta S\phi & S\theta S\psi S\phi + C\psi C\phi & S\theta C\psi S\phi - S\psi C\phi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix} \quad (5.31e)$$

Since the elementary rotations do not commute, any change in their order will lead to different representations for the overall rotation matrix \mathbf{Q} . Accordingly, one can define a total of 12 sets indicated in Table 5.2.

Table 5.2 Rotation matrices for fixed-axes rotations and Euler angles

SN	Fixed-axes [$\psi\theta\phi$]	Euler angles [$\phi\theta\psi$]	Rotation Matrix, \mathbf{Q}
Nonsymmetric sets			
1	XYZ [$\mathbf{Q}_Z \mathbf{Q}_Y \mathbf{Q}_X$]	ZYX [$\mathbf{Q}_Z \mathbf{Q}_Y \mathbf{Q}_{X''}$]	$\begin{bmatrix} C\theta C\phi & S\theta S\psi C\phi - C\psi S\phi & S\theta C\psi C\phi + S\psi S\phi \\ C\theta S\phi & S\theta S\psi S\phi + C\psi C\phi & S\theta C\psi S\phi - S\psi C\phi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix}$
2	YZX [$\mathbf{Q}_X \mathbf{Q}_Z \mathbf{Q}_Y$]	XZY [$\mathbf{Q}_X \mathbf{Q}_Z' \mathbf{Q}_{Y''}$]	$\begin{bmatrix} C\theta C\psi & -S\theta & C\theta S\psi \\ C\psi S\theta C\phi + S\psi S\phi & C\theta C\phi & S\theta S\psi C\phi - C\psi S\phi \\ C\psi S\theta S\phi - S\psi C\phi & C\theta S\phi & S\theta S\psi S\phi + C\psi C\phi \end{bmatrix}$
3	ZXY [$\mathbf{Q}_Y \mathbf{Q}_X \mathbf{Q}_Z$]	YXZ [$\mathbf{Q}_Y \mathbf{Q}_X' \mathbf{Q}_{Z''}$]	$\begin{bmatrix} S\psi S\theta S\phi + C\psi C\phi & C\psi S\theta S\phi - S\psi C\phi & C\theta S\phi \\ C\theta S\psi & C\theta C\psi & -S\theta \\ S\psi S\theta C\phi - C\psi S\phi & C\psi S\theta C\phi + S\psi S\phi & C\theta C\phi \end{bmatrix}$
4	ZYX [$\mathbf{Q}_X \mathbf{Q}_Y \mathbf{Q}_Z$]	XYZ [$\mathbf{Q}_X \mathbf{Q}_Y' \mathbf{Q}_{Z''}$]	$\begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta \\ C\psi S\theta S\phi + S\psi C\phi & -S\theta S\psi S\phi + C\psi C\phi & -C\theta S\phi \\ -S\theta C\psi C\phi + S\psi S\phi & S\psi S\theta C\phi + C\psi S\phi & C\theta C\phi \end{bmatrix}$
5	XZY [$\mathbf{Q}_Y \mathbf{Q}_Z \mathbf{Q}_X$]	YZX [$\mathbf{Q}_Y \mathbf{Q}_Z' \mathbf{Q}_{X''}$]	$\begin{bmatrix} C\theta C\phi & -C\psi S\theta C\phi + S\psi S\phi & S\theta S\psi C\phi + C\psi S\phi \\ S\theta & C\theta C\psi & -C\theta S\psi \\ -C\theta S\phi & C\psi S\theta S\phi + S\psi C\phi & -S\theta S\psi S\phi + C\psi C\phi \end{bmatrix}$
6	YXZ [$\mathbf{Q}_Z \mathbf{Q}_X \mathbf{Q}_Y$]	ZXY [$\mathbf{Q}_Z \mathbf{Q}_X' \mathbf{Q}_{Y''}$]	$\begin{bmatrix} -S\phi S\psi S\theta + C\phi C\psi & -C\theta S\phi & S\theta C\psi S\phi + S\psi C\phi \\ C\phi S\psi S\theta + S\phi C\psi & C\theta C\phi & -C\phi C\psi S\theta + S\phi S\psi \\ -C\theta S\psi & S\theta & C\psi C\theta \end{bmatrix}$

(Contd.)

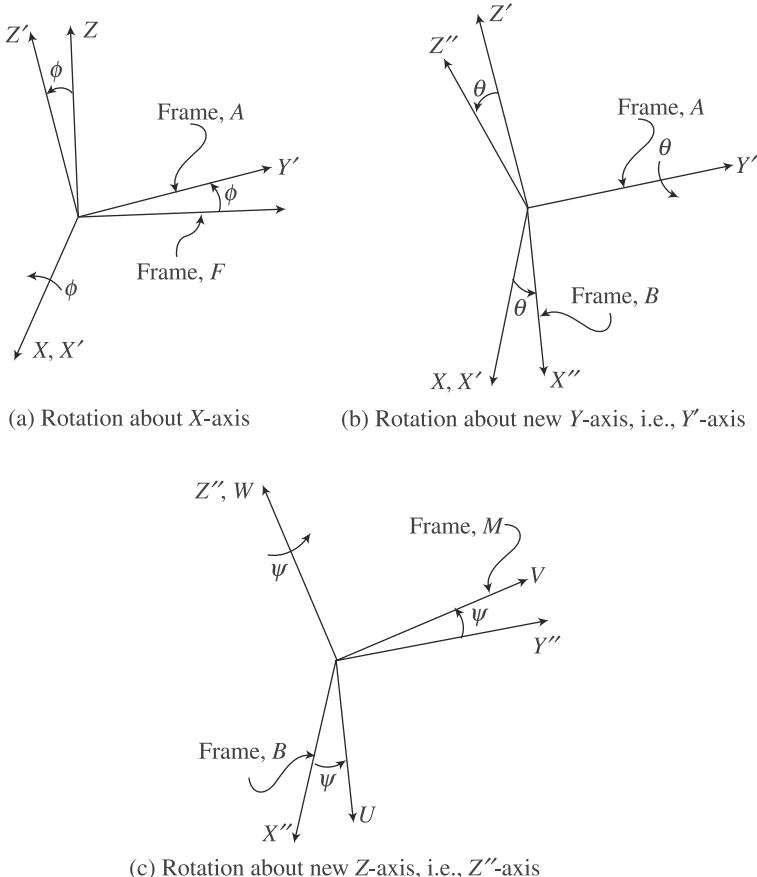
Table 5.2 (Contd.)

SN	Fixed-axes [$\psi\theta\phi$]	Euler angles [$\phi\theta\psi$]	Rotation Matrix, \mathbf{Q}
Symmetric sets			
7	XYY [$\mathbf{Q}_X \mathbf{Q}_Y \mathbf{Q}_X$]	XYY [$\mathbf{Q}_X \mathbf{Q}_{Y'} \mathbf{Q}_{X''}$]	$\begin{bmatrix} C\theta & S\theta S\psi & S\theta C\psi \\ S\theta S\phi & -C\theta S\psi S\phi + C\psi C\phi & -C\theta C\psi S\phi - S\psi C\phi \\ -S\theta C\phi & C\theta S\psi C\phi + C\psi S\phi & C\theta C\psi C\phi - S\psi S\phi \end{bmatrix}$
8	XZX [$\mathbf{Q}_X \mathbf{Q}_Z \mathbf{Q}_X$]	XZX [$\mathbf{Q}_X \mathbf{Q}_{Z'} \mathbf{Q}_{X''}$]	$\begin{bmatrix} C\theta & -S\theta C\psi & S\theta S\psi \\ S\theta C\phi & C\theta C\psi C\phi - S\psi S\phi & -C\theta S\psi C\phi - C\psi S\phi \\ S\theta S\phi & C\theta C\psi S\phi + S\psi C\phi & -C\theta S\psi S\phi + C\psi C\phi \end{bmatrix}$
9	YZY [$\mathbf{Q}_Y \mathbf{Q}_Z \mathbf{Q}_Y$]	YZY [$\mathbf{Q}_Y \mathbf{Q}_{Z'} \mathbf{Q}_{Y''}$]	$\begin{bmatrix} C\theta C\psi C\phi - S\psi S\phi & -S\theta C\phi & C\theta S\psi C\phi - S\psi S\phi \\ S\theta C\psi & C\theta & S\theta S\psi \\ -C\theta C\psi S\phi - S\psi C\phi & S\theta S\phi & -C\theta S\psi S\phi + C\psi C\phi \end{bmatrix}$
10	YXY [$\mathbf{Q}_Y \mathbf{Q}_X \mathbf{Q}_Y$]	YXY [$\mathbf{Q}_Y \mathbf{Q}_{X'} \mathbf{Q}_{Y''}$]	$\begin{bmatrix} -C\theta S\psi S\phi + C\psi C\phi & S\theta S\phi & C\theta C\psi S\phi + S\psi C\phi \\ S\theta S\psi & C\theta & -S\theta C\psi \\ -C\theta S\psi C\phi - C\psi S\phi & S\theta C\phi & C\theta C\psi C\phi - S\psi S\phi \end{bmatrix}$
11	ZXZ [$\mathbf{Q}_Z \mathbf{Q}_X \mathbf{Q}_Z$]	ZXZ [$\mathbf{Q}_Z \mathbf{Q}_{X'} \mathbf{Q}_{Z''}$]	$\begin{bmatrix} -C\theta S\psi S\phi + C\psi C\phi & -C\theta C\psi S\phi - S\psi C\phi & S\theta S\phi \\ C\theta C\phi S\psi + C\psi S\phi & C\theta C\psi C\phi - S\psi S\phi & -S\theta C\phi \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix}$
12	ZYZ [$\mathbf{Q}_Z \mathbf{Q}_Y \mathbf{Q}_Z$]	ZYZ [$\mathbf{Q}_Z \mathbf{Q}_{Y'} \mathbf{Q}_{Z''}$]	$\begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\phi C\theta S\psi - S\phi C\psi & S\theta C\phi \\ S\phi C\theta C\psi + C\phi S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\theta S\phi \\ -S\theta C\psi & S\theta S\psi & C\theta \end{bmatrix}$

3. Euler Angles Representation Euler angles also constitute a minimal representation of orientation obtained by composing three elementary rotations with respect to the axes of current frames. Unlike the fixed-axes rotations, here the rotations are performed with respect to the current frames. As a result, it can be shown that the resulting matrix representing the orientation of the frame M with respect to the fixed frame F , which is denoted with the matrix \mathbf{Q} , can be obtained via post-multiplications of the successive rotation matrices.

For example, as shown in Fig. 5.15, if the frame F is rotated first by about the axis Z , followed by Y' next, and finally by X'' , the resulting rotation matrix \mathbf{Q} can be obtained from the composition of the three elementary rotations, namely, \mathbf{Q}_X , \mathbf{Q}_Y and $\mathbf{Q}_{Z''}$, with respect to their current frames, i.e.,

$$\mathbf{Q} = \mathbf{Q}_X \mathbf{Q}_Y \mathbf{Q}_{Z''} \quad (5.32)$$

**Fig. 5.15** The XYZ Euler angles

Comparison of Eqs. (5.31d–5.32) suggests that if the notations for the angles about first and third fixed axes are interchanged to denote the Euler angles, i.e., instead of ψ , θ and ϕ about first, second and third (not always about axes X , Y and Z though!) fixed axes, if ϕ , θ and ψ are used to denote the Euler angles about first, second and third axes the resulting rotation matrix representations are same. For example, if the rotation matrix due to XYZ fixed-axes rotations is given by Eq. (5.31d) then according to Eq. (5.32), the expression will be same as ZXY Euler angles, i.e.,

$$\mathbf{Q}_Z(\phi)\mathbf{Q}_Y(\theta)\mathbf{Q}_X(\psi) = \mathbf{Q}_Z(\phi)\mathbf{Q}_{Y'}(\theta)\mathbf{Q}_{X''}(\psi) \quad (5.33)$$

where $\mathbf{Q}_i(\cdot)$ represents the elementary rotation matrix about the i th axis by an angle (\cdot) , which are given by Eqs. (5.21–5.22) or Eqs. (5.31a–c). The elements of the resulting matrix expression given by Eq. (5.33) are shown in Eq. (5.31e). The resulting rotation matrices for all twelve possible sets of fixed-axes and Euler angles representations are shown in Table 5.2.

The ZYZ Euler Angles Set Even though twelve sets of Euler angle combinations exist, the ZYZ set is one of the most commonly used representations. Hence, the

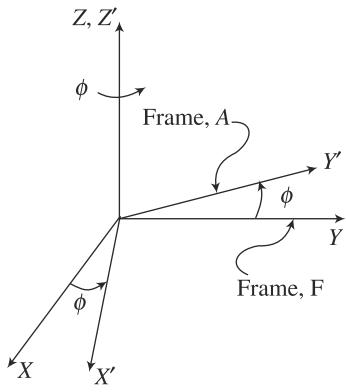
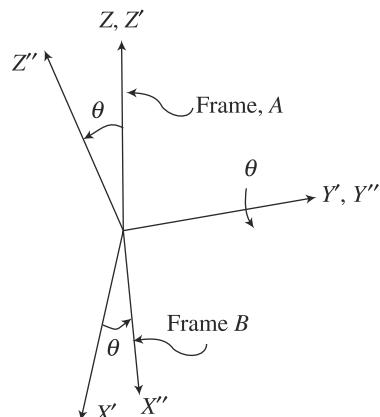
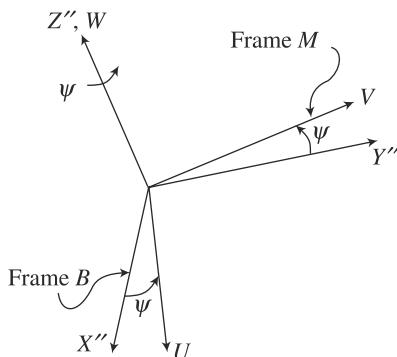
derivation of the rotation matrix \mathbf{Q} for the ZYZ Euler angles is separately derived here. Referring to Fig. 5.16,

- Rotate the fixed frame F by the angle ϕ about the axis Z , as indicated in Fig. 5.16(a). This rotation is described by the rotation matrix \mathbf{Q}_Z . This is similar to Eq. (5.21) or (5.31c), i.e.,

$$\mathbf{Q}_Z \equiv \begin{bmatrix} C\phi & -S\phi & 0 \\ S\phi & C\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.34a)$$

- Rotate the current frame A by an angle θ about its Y' axis, Fig. 5.16(b). This rotation is denoted by $\mathbf{Q}_{Y'}$, which is similar to the rotation matrix \mathbf{Q}_Y of Eq. (5.22) or Eq. (5.31b), i.e.,

$$\mathbf{Q}_{Y'} \equiv \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \quad (5.34b)$$

(a) Rotation about Z -axis(b) Rotation about current Y' -axis, i.e., Y' -axis(c) Rotation about current Z'' -axis, i.e., Z'' -axis**Fig. 5.16** The ZYZ Euler angles

- Rotate the current frame B by an angle ψ about its Z'' axis, Fig. 5.16(c). This rotation is denoted by $\mathbf{Q}_{Z''}$ and described similar to Eq. (5.34a) above, i.e.,

$$\mathbf{Q}_{Z''} \equiv \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.34c)$$

The resulting orientation of the frame M , denoted by \mathbf{Q} , is obtained from the composition of the three elementary rotations \mathbf{Q}_Z , \mathbf{Q}_Y and $\mathbf{Q}_{Z''}$ with respect to their current frames. It is obtained via post-multiplications of the successive rotation matrices, as done for the XYZ Euler angles given by Eq. (5.33), i.e.,

$$\mathbf{Q} = \mathbf{Q}_Z \mathbf{Q}_Y \mathbf{Q}_{Z''} \quad (5.34d)$$

whose elements are computed below:

$$\mathbf{Q} \equiv \begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\phi C\theta S\psi - S\phi C\psi & S\theta C\phi \\ S\phi C\theta C\psi + C\phi S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\theta S\phi \\ -S\theta C\psi & S\theta S\psi & C\theta \end{bmatrix} \quad (5.34e)$$

The expression in Eq. (5.34e) is same as the 12th set of Table 5.2, i.e., ZYZ symmetric set of Euler angles.

The drawback of the minimal set of Euler angles representation given by, say, Eq. (5.34e) or any of the sets in Table 5.2, is that it sometimes fails to find the solution of an inverse problem, i.e., for a given rotation matrix \mathbf{Q} , find the equivalent Euler angles. For example, if \mathbf{Q} is given as follows:

$$\mathbf{Q} \equiv \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \quad (5.35a)$$

then the angle ϕ can be obtained from the comparison of (1, 3) and (2, 3) elements of \mathbf{Q} given in Eq. (5.34e), i.e.,

$$\phi = \text{atan2}\left(\frac{q_{23}}{S\theta}, \frac{q_{13}}{S\theta}\right) \quad (5.35b)$$

where “*atan2(y, x)*” is the two-argument inverse tangent function that yields one unique solution for the angle. The solution of Eq. (5.35b) exists provided $S\theta \neq 0$, i.e.,

MuPAD

It is MATLAB's symbolic computation software.

when $\theta \neq 0$ or $n\pi$, for $n = 1, 2, \dots$. Such non-existence of solutions refers to numerical singularity, and is known as one of the major disadvantages of Euler angles or for that matter any minimal set representation of rotations.

Example 5.10 Verify a Rotation Matrix using MuPAD

Figure 5.17 shows MuPAD screenshot for the symbolic evaluation of the rotation matrix of the ZYZ Euler angles. It validates the expression shown in Eq. (5.34e). One can also use MATLAB's symbolic commands to evaluate the same as well. The MATLAB commands and their output are shown as follows:

**** This program in MuPAD computes ZYZ Rotation Matrix ****

Define Greek variables

```
[ ph := `&phi;`  
[ φ
```

```
[ th := `&theta;`  
[ θ
```

```
[ ps := `&psi;`  
[ ψ
```

Define Matrices

```
[ Qz:=matrix([[cos(ph), -sin(ph), 0], [sin(ph), cos(ph), 0], [0, 0, 1]])  
[ 
$$\begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```

```
[ Qyp:=matrix([[cos(th), 0, sin(th)], [0, 1, 0], [-sin(th), 0, cos(th)]])  
[ 
$$\begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

```

```
[ Qzpp:=matrix([[cos(ps), -sin(ps), 0], [sin(ps), cos(ps), 0], [0, 0, 1]])  
[ 
$$\begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```

Calculates resultant matrix

```
[ Q:=Qz*Qyp*Qzpp  
[ 
$$\begin{pmatrix} \cos(\phi) \cos(\psi) \cos(\theta) - \sin(\phi) \sin(\psi) & -\cos(\psi) \sin(\phi) - \cos(\phi) \cos(\theta) \sin(\psi) & \cos(\phi) \sin(\theta) \\ \cos(\phi) \sin(\psi) + \cos(\psi) \cos(\theta) \sin(\phi) & \cos(\phi) \cos(\psi) - \cos(\theta) \sin(\phi) \sin(\psi) & \sin(\phi) \sin(\theta) \\ -\cos(\psi) \sin(\theta) & \sin(\psi) \sin(\theta) & \cos(\theta) \end{pmatrix}$$

```

Fig. 5.17 MuPAD notebook to verify a rotation matrix w.r.t. ZYZ Euler angles

```
>> syms p t s  
>> Qz = [cos(p), -sin(p), 0; sin(p), cos(p), 0; 0, 0, 1]  
>> Qyp = [cos(t), 0, sin(t); 0, 1, 0; -sin(t), 0, cos(t)]  
>> Qzpp = [cos(s), -sin(s), 0; sin(s), cos(s), 0; 0, 0, 1]  
>> Q = Qz*Qyp*Qzpp  
  
Q =  
[cos(p)*cos(s)*cos(t) - sin(p)*sin(s), -cos(s)*sin(p) - cos(p)*cos(t)*sin(s),  
cos(p)*sin(t)]  
[cos(p)*sin(s) + cos(s)*cos(t)*sin(p), cos(p)*cos(s) - cos(t)*sin(p)*sin(s),  
sin(p)*sin(t)]  
[-cos(s)*sin(t), sin(s)*sin(t), cos(t)]
```

where ‘*p*’, ‘*t*’, and ‘*s*’ respectively are the angles ϕ , θ , and ψ representing the ZYZ Euler angles.

Example 5.11 Extract ZYZ Euler Angles

Given a rotation matrix \mathbf{Q} as follows:

$$\mathbf{Q} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \quad (5.36)$$

Its ZYZ Euler angles, namely, ϕ , θ , ψ , can be calculated using the comparisons between the elements of Eqs. (5.34e) and (5.36) as follows:

- The (3, 3) element: $C\theta = 0 \Rightarrow \theta = 90^\circ$
- The (1, 3) and (2, 3) elements: $S\theta C\phi = 0$; $S\theta S\phi = 1 \Rightarrow S\phi = 1$ (since $S\theta = 1$)
 $\Rightarrow \phi = 90^\circ$
- The (3, 1) element: $-S\theta C\psi = -1 \Rightarrow \psi = 0^\circ$

The ZYZ Euler angles representing the rotation of the frame M with respect to the frame F is given by $(\phi, \theta, \psi) = (90^\circ, 90^\circ, 0^\circ)$. The relative orientation of the two frames is shown in Fig. 5.18.

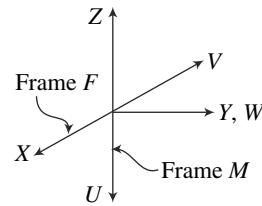


Fig. 5.18 Relation between frames M and F

4. Single- and Double-axes Rotations Using single-axis-rotation representation, the relation between any two frames, say, the fixed frame F and the moving frame M having the same origin, can be given by the pair $[\mathbf{e}, \alpha]$, where $\mathbf{e} \equiv [e_x, e_y, e_z]^T$ represents the unit vector parallel to the axis of rotation and α is the angle of rotation about \mathbf{e} , as indicated in Fig. 5.19(a). Note that the single-axis rotation uses four parameters instead of three parameters, as in the case of representations using fixed-axes rotations or Euler angles. The corresponding rotation matrix, denoted with $\mathbf{Q}(\mathbf{e}, \alpha)$, is given by

$$\mathbf{Q}(\mathbf{e}, \alpha) = \begin{bmatrix} e_x^2 C'\alpha + C\alpha & e_x e_y C'\alpha - e_z S\alpha & e_x e_z C'\alpha + e_y S\alpha \\ e_x e_y C'\alpha + e_z S\alpha & e_y^2 C'\alpha + C\alpha & e_y e_z C'\alpha - e_x S\alpha \\ e_x e_z C'\alpha - e_y S\alpha & e_y e_z C'\alpha + e_x S\alpha & e_z^2 C'\alpha + C\alpha \end{bmatrix} \quad (5.37)$$

where $S\alpha \equiv \sin \alpha$, $C\alpha \equiv \cos \alpha$, and $C'\alpha \equiv 1 - C\alpha$. The expression for the rotation matrix $\mathbf{Q}(\mathbf{e}, \alpha)$ given by Eq. (5.37) can be derived by noting the following:

- Referring to Fig. 5.19, assume that the unit vector \mathbf{e} is parallel to the axis W of the moving frame M .
- Matrix \mathbf{Q} denotes the rotation of Frame M with respect to Frame F .
- Rotation matrix $\mathbf{Q}(\mathbf{e}, \alpha)$ is then equivalent to
 - Rotation of frame M to coincide with frame F . This will make the unit vector \mathbf{e} or the axis W parallel to the axis Z of the fixed frame F . This rotation is denoted with matrix \mathbf{Q}^T ;
 - Rotation about the new axis W or axis Z by the angle α . This is denoted with $\mathbf{Q}(\mathbf{z}, \alpha)$, where \mathbf{z} is the unit vector parallel to axis Z ; and
 - Reversal of the rotation of frame M to frame F which will take care of the initial rotation of item (a) above. This can be done from the fact that the rotation of frame F to frame M can be represented by the matrix \mathbf{Q} .

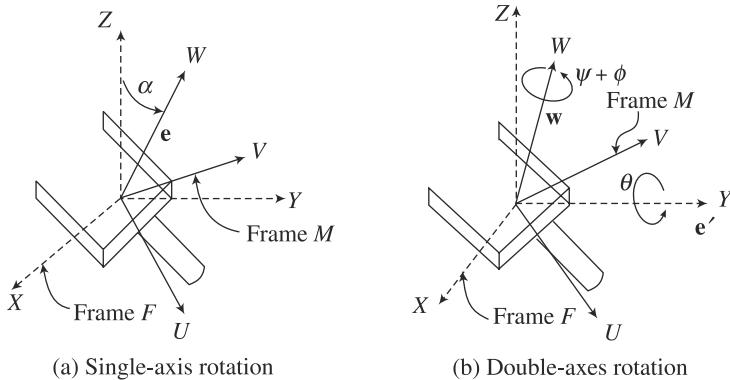


Fig. 5.19 Single- and double-axis rotations

Hence, the final expression for the single-axis rotation in frame F can be given by

$$\mathbf{Q}(\mathbf{e}, \alpha) = \mathbf{Q}\mathbf{Q}(\mathbf{z}, \alpha)\mathbf{Q}^T \quad (5.38a)$$

where the elements of 3×3 matrix \mathbf{Q} are shown below:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & e_x \\ q_{21} & q_{22} & e_y \\ q_{31} & q_{32} & e_z \end{bmatrix} \quad (5.38b)$$

It is a simple matter to note that the third column of the matrix \mathbf{Q} is nothing but the components of the unit vector \mathbf{e} , as it is parallel to the axis W of the frame M . Performing the matrix multiplications of Eq. (5.38a) and using the orthogonal properties of \mathbf{Q} in (5.38b), i.e.,

$$q_{11}^2 + q_{21}^2 + q_{31}^2 = q_{12}^2 + q_{22}^2 + q_{32}^2 = e_x^2 + e_y^2 + e_z^2 = 1$$

$$q_{11}^2 + q_{12}^2 + e_x^2 = q_{21}^2 + q_{22}^2 + e_y^2 = q_{31}^2 + q_{32}^2 + e_z^2 = 1$$

$$q_{11}q_{12} + q_{21}q_{22} + q_{31}q_{32} = q_{11}e_x + q_{21}e_y + q_{31}e_z = q_{12}e_x + q_{22}e_y + q_{32}e_z = 0 \quad (5.38c)$$

$$q_{11}q_{21} + q_{12}q_{22} + e_xe_y = q_{11}q_{31} + q_{12}q_{32} + e_xe_z = q_{21}q_{31} + q_{22}q_{32} + e_ye_z = 0$$

$$q_{21}q_{32} - q_{22}q_{31} = e_x; q_{31}q_{12} - q_{11}q_{32} = e_y; q_{11}q_{22} - q_{21}q_{12} = e_z$$

one can then immediately prove the expression of the matrix $\mathbf{Q}(\mathbf{e}, \alpha)$ given by Eq. (5.37). In Eq. (5.38c), the first two lines correspond to the unit magnitudes of the columns and rows of the matrix \mathbf{Q} , whereas the next two lines correspond to the zero values of the dot products between any two columns or rows. The last line of Eq. (5.38c) gives the result of the cross-product between the first and second columns of \mathbf{Q} as the third column.

The interpretation of Eq. (5.37) or (5.38a) is straightforward. First note that

$$\mathbf{Q}(\mathbf{e}, \alpha) = [\mathbf{u} \quad \mathbf{v} \quad \mathbf{w}] \quad (5.39)$$

where, \mathbf{u} , \mathbf{v} , and \mathbf{w} are the unit vectors parallel to the axes U , V and W of frame M expressed in frame F . Moreover, let \mathbf{r}_1 be an arbitrary vector in fixed-frame F , and

\mathbf{r}_2 be the same vector obtained after rotating \mathbf{r}_1 by an angle α about \mathbf{e} . One can have their relation from the fundamentals of rotational kinematics (Angeles, 2003) as

$$\mathbf{r}_2 = (\mathbf{e}^T \mathbf{r}_1) \mathbf{e} + C\alpha [\mathbf{r}_1 - (\mathbf{e}^T \mathbf{r}_1) \mathbf{e}] + S\alpha (\mathbf{e} \times \mathbf{r}_1) \quad (5.40a)$$

Now, if $\mathbf{r}_1 = \mathbf{i}$ --- the unit vector \mathbf{i} being parallel to the axis X of the frame F which is nothing but $\mathbf{i} \equiv [1 \ 0 \ 0]^T$ --- then the result of \mathbf{r}_2 can be given from Eq. (5.40a) as

$$\mathbf{r}_2 \equiv [e_x^2 C'\alpha + C\alpha, \ e_x e_y C'\alpha + e_z S'\alpha, \ e_x e_z C'\alpha + e_y S'\alpha]^T \quad (5.40b)$$

which is nothing but the first column of $\mathbf{Q}(\mathbf{e}, \alpha)$ in Eq. (5.37) or the representation of \mathbf{u} in frame F , as expected.

In double-axes rotation, first suppose that the final orientation of a frame M with respect to the frame F is denoted with ZYZ Euler angles given by (ϕ, θ, ψ) . Using Eq. (5.37), one can check that the expression for the matrix \mathbf{Q} given by Eq. (5.34e) is equivalent to $\mathbf{Q}(\mathbf{e}', \theta) \mathbf{Q}(\mathbf{w}, \psi + \phi)$, where $\mathbf{e}' \equiv [-\sin \phi, \cos \phi, 0]^T$ and $\mathbf{w} \equiv [0, 0, 1]^T$ are the unit vectors about which two rotations are performed, as depicted in Fig. 5.19(b). The unit vector \mathbf{w} is parallel to the Z-axis after the rotation by $\mathbf{Q}(\mathbf{e}', \theta)$.

5. Euler Parameters Another four-parameters representation of a rotation matrix, similar to the single-axis rotation, is using Euler parameters (not same as Euler angles!). These parameters, also referred in the literature as *Euler–Rodrigues parameters* (e.g., in Angeles, 2003) or *unit quaternion* (e.g., in Craig, 2009), are defined by

$$\mathbf{r} = \mathbf{e} \left(\sin \frac{\alpha}{2} \right), r_0 = \cos \frac{\alpha}{2} \quad (5.41)$$

where $\mathbf{r} \equiv [r_x \ r_y \ r_z]^T$ has three components representing the scaled vector parallel to the axis of rotation \mathbf{e} , whereas r_0 is the scalar term associated with the cosine of the half-angle of rotation α . These four parameters are constrained through $r_x^2 + r_y^2 + r_z^2 + r_0^2 = 1$, whereas the 3×3 rotation matrix \mathbf{Q} is represented as

$$\mathbf{Q} = (r_0^2 - \mathbf{r}^T \mathbf{r}) \mathbf{1} + 2\mathbf{r}\mathbf{r}^T + 2r_0 \mathbf{r} \times \mathbf{1} \quad (5.42a)$$

where $\mathbf{1}$ is the 3×3 identity matrix, whereas $\mathbf{r} \times \mathbf{1}$ is the 3×3 skew-symmetric matrix which denotes a cross-product tensor, i.e., for any 3-dimensional vector \mathbf{x} , $(\mathbf{r} \times \mathbf{1}) \mathbf{x} = \mathbf{r} \times \mathbf{x}$. Substituting Eq. (5.41) into Eq. (5.42a) yields the following:

$$\mathbf{Q} = \begin{bmatrix} 1 - 2(r_y^2 + r_z^2) & 2(r_x r_y - r_0 r_z) & 2(r_x r_z + r_0 r_y) \\ 2(r_x r_y + r_0 r_z) & 1 - 2(r_x^2 + r_z^2) & 2(r_y r_z - r_0 r_x) \\ 2(r_x r_z - r_0 r_y) & 2(r_y r_z + r_0 r_x) & 1 - 2(r_x^2 + r_y^2) \end{bmatrix} \quad (5.42b)$$

Comparing the scalar elements of the Euler parameters given by Eq. (5.41) and the expression of matrix \mathbf{Q} in Eq. (5.42b), one can extract the Euler parameters for a given rotation matrix as

$$\begin{aligned} r_0 &= \frac{1}{2} \sqrt{1 + q_{11} + q_{22} + q_{33}} \\ r_x &= \frac{1}{4r_0} (q_{32} - q_{23}); r_y = \frac{1}{4r_0} (q_{13} - q_{31}); r_z = \frac{1}{4r_0} (q_{21} - q_{12}) \end{aligned} \quad (5.43)$$

where q_{ij} , for $i, j = 1, \dots, 3$, are the elements of the matrix \mathbf{Q} . Equation (5.43) fails when r_0 vanishes, which happens when $\alpha = \pi$. This is the limitation of the use of Euler parameters like it happens in the case of Euler angles, as mentioned after Eq. (5.35b).

5.2.3 Noncommutative Property of Rotation

An important aspect of the rotation representation, either using direction cosine or Euler angles or any other, is that unlike vectors, it is noncommutative, i.e., the order of rotations is important in deriving the correct representation. As illustrated in Fig. 5.20(a-c), the rotation of a box about Z and Y axes is different from the rotation about Y and Z axes, as shown in Fig. 5.21(a-c). In order to show it mathematically, the resultant matrix for the rotations about Z and Y-axes, denoted by \mathbf{Q}_{ZY} , can be given by

$$\mathbf{Q}_{ZY} = \mathbf{Q}_Y \mathbf{Q}_Z = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (5.44a)$$

where \mathbf{Q}_Y and \mathbf{Q}_Z are as follows:

$$\mathbf{Q}_Y = \begin{bmatrix} C90^\circ & 0 & S90^\circ \\ 0 & 1 & 0 \\ -S90^\circ & 0 & C90^\circ \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}; \mathbf{Q}_Z = \begin{bmatrix} C90^\circ & -S90^\circ & 0 \\ S90^\circ & C90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.44b)$$

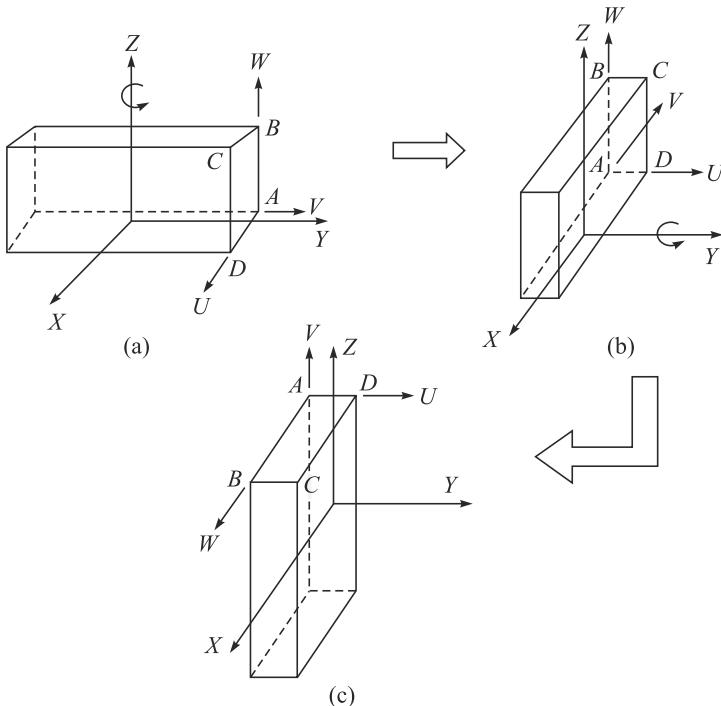


Fig. 5.20 Successive rotation of a box about Z and Y-axes

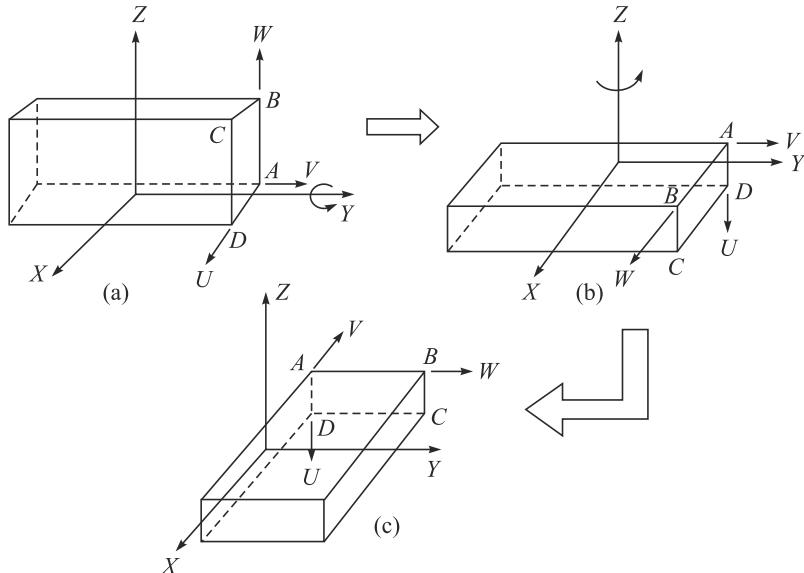


Fig. 5.21 Successive rotation of a box about Y and Z -axes

Note the order of multiplications in Eq. (5.44a) which are with respect to fixed axes, as done in Eq. (5.31d). The result of Eq. (5.44a) can be verified from Fig. 5.20(c) using the concept of direction cosine representation of a rotation matrix explained earlier. Similarly, the resultant rotation matrix for the rotations about Y and Z -axes, denoted by \mathbf{Q}_{YZ} , can be obtained as

$$\mathbf{Q}_{YZ} = \mathbf{Q}_Z \mathbf{Q}_Y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \quad (5.44c)$$

Comparing Eqs. (5.44a) and (5.44c), it is obvious that the two sequences of rotations are different. However, the same is not true for the vector representation of a point P with respect to O shown in Fig. 5.22. One can either add the vector \mathbf{b} to \mathbf{a} or \mathbf{a} to \mathbf{b} . Both additions will result the same vector \mathbf{p} .

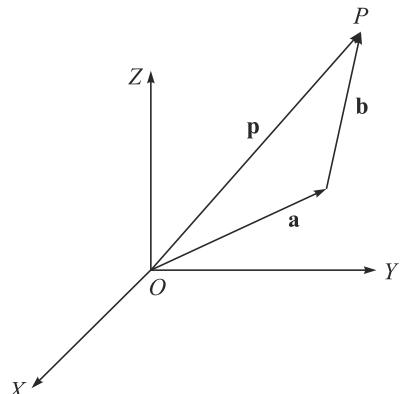


Fig. 5.22 Addition of two vectors

5.3 COORDINATE TRANSFORMATION

As pointed out in Section 5.2, the position of a rigid body in space is expressed in terms of the position of a suitable point on the body, say, O_M in Fig. 5.23, with respect to the origin O of the fixed frame F , while its orientation or rotation is expressed in terms of the unit vector components of the axes of the moving frame M attached to the body. Referring to Fig. 5.23, the relation between the two coordinate frames,

namely, F and M , are derived as follows: Consider an arbitrary point P on the rigid body to which the frame M is attached at O_M . Let \mathbf{p} and \mathbf{p}' be the vectors denoting the point P from the origin of the frames F and M , respectively. In addition, let \mathbf{o} be the position vector denoting the translation of the origin of the frame M , O_M , from that of the frame F , i.e., O . Thus,

$$\mathbf{p} = \mathbf{o} + \mathbf{p}' \quad (5.45)$$

Note that if \mathbf{p}' is known in the moving frame M , then it is nothing but the coordinates of the point P in the frame M , i.e., $[\mathbf{p}']_M$. Moreover, if \mathbf{Q} is the orientation of the frame M with respect to the frame F then the vector \mathbf{p}' in the frame F , i.e., $[\mathbf{p}']_F = \mathbf{Q}[\mathbf{p}']_M$. Hence, the vector, \mathbf{p} , in the fixed frame F , i.e., $[\mathbf{p}]_F$, can be obtained as

$$[\mathbf{p}]_F = [\mathbf{o}]_F + \mathbf{Q}[\mathbf{p}']_M \quad (5.46)$$

Equation (5.46) represents the coordinate transformation of the point P of the rigid body from the moving frame M to the fixed frame F , while both translation and rotation are involved.

5.3.1 Homogeneous Transformation

The coordinate transformation given by Eq. (5.46) can be rearranged as

$$\begin{bmatrix} [\mathbf{p}]_F \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{Q} & [\mathbf{o}]_F \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} [\mathbf{p}']_M \\ 1 \end{bmatrix} \quad (5.47)$$

where $\mathbf{0} \equiv [0, 0, 0]^T$ is the three-dimensional vector (column) of zeros. Accordingly, $\mathbf{0}^T \equiv [0, 0, 0]$. Equation (5.47) is written in a compact form as

$$[\bar{\mathbf{p}}]_F = \mathbf{T}[\bar{\mathbf{p}}']_M \quad (5.48)$$

where $[\bar{\mathbf{p}}]_F$ and $[\bar{\mathbf{p}}']_M$ are the four-dimensional vectors obtained by putting one at

the bottom of the original three-dimensional vectors, $[\mathbf{p}]_F$ and $[\mathbf{p}']_M$, respectively, as the fourth element, whereas the 4×4 matrix \mathbf{T} is called *Homogeneous Transformation Matrix* (HTM). Equation (5.48) is simple in a sense that the transformation of a vector, which includes

both translation and rotation, from the frame M to F is done by just multiplying one 4×4 matrix, instead of a matrix multiplication and vector addition, as in Eq. (5.46). However, from the view of computational complexity, i.e., the numbers of multiplications/divisions and additions/subtractions required in a computer program, Eq. (5.46) is economical compared to Eq. (5.47) or (5.48), as some unnecessary multiplications and additions with 1's and 0's, respectively, have to be performed.

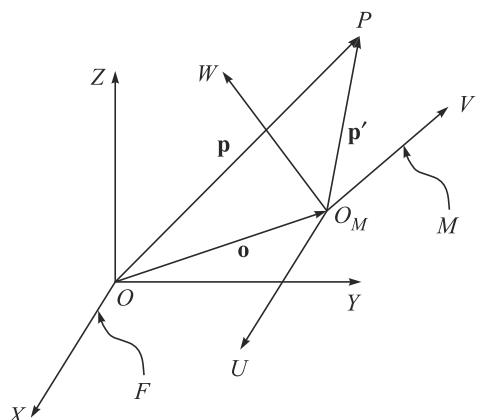


Fig. 5.23 Two coordinate frames

Why Homogenous?

Matrix \mathbf{T} of Eq. (5.48) takes care of both the translation and rotation of the frame attached to the body with respect to the fixed frame.

It is pointed out here that for the homogeneous transformation matrix \mathbf{T} , the orthogonality property does not hold good, i.e.,

$$\mathbf{T}^T \mathbf{T} \neq \mathbf{1} \quad \text{or} \quad \mathbf{T}^{-1} \neq \mathbf{T}^T \quad (5.49)$$

However, the inverse of the HTM \mathbf{T} can be obtained easily from Eq. (5.47) as

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{Q}^T & -\mathbf{Q}^T [\mathbf{o}]_F \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (5.50)$$

Example 5.12 Pure Translation

Referring to Fig. 5.24(a), consider the frame M which is obtained from the frame F by translating it two units along Y and one unit along Z . Their relation is represented by a homogeneous transformation matrix (HTM), namely,

$$\mathbf{T} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.51)$$

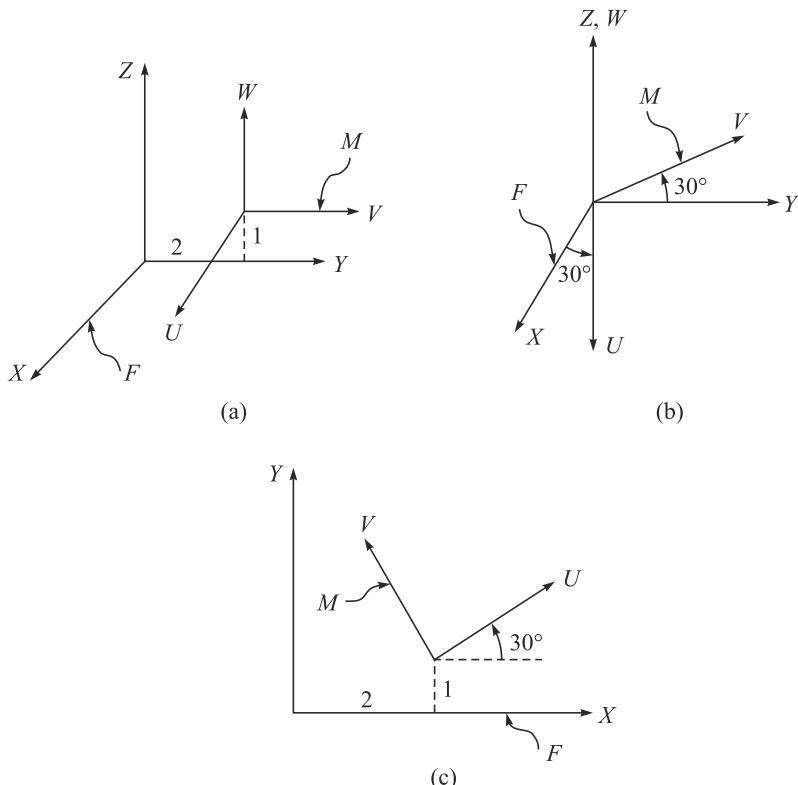


Fig. 5.24 Different motions

Example 5.13 Pure Rotation

Consider Fig. 5.24(b), where the frame M is obtained from the frame F by rotating it about its Z -axis by an angle of 30° . The HTM for the pure rotation is then given by

$$\mathbf{T} \equiv \begin{bmatrix} C30^\circ & -S30^\circ & 0 & 0 \\ S30^\circ & C30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.52)$$

Example 5.14 General Motion

As shown in Fig. 5.24(c), the frame M is obtained from the frame F by translating it along X and Y axes by 2 and 1 units, respectively, followed by a rotation of 30° about W . The corresponding homogeneous transformation matrix is given from the multiplication of the HTMs representing the pure translation and rotation, respectively, with respect to the current frames, i.e.,

$$\mathbf{T} \equiv \mathbf{T}_t \mathbf{T}_r \quad (5.53)$$

where subscripts t and r stand for *translation* and *rotation*, respectively. The resultant matrix \mathbf{T} is given by

$$\mathbf{T} \equiv \begin{bmatrix} C30^\circ & -S30^\circ & 0 & 2 \\ S30^\circ & C30^\circ & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 2 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.54)$$

Example 5.15 Transfer of a Point

Referring to Fig. 5.25, consider a point P , which is expressed in the frame F and denoted by the vector \mathbf{p}_1 . Let \mathbf{p}_2 denotes the new location of this point after rotating \mathbf{p}_1 by 30° about Z , and then translating 2 units along Y - and -1 unit along Z -axes. The relation between \mathbf{p}_1 and \mathbf{p}_2 can be described as

$$[\mathbf{p}_2]_F = \mathbf{T}[\mathbf{p}_1]_F, \text{ where } \mathbf{T} \equiv \mathbf{T}_t \mathbf{T}_r = \begin{bmatrix} C30^\circ & -S30^\circ & 0 & 0 \\ S30^\circ & C30^\circ & 0 & 2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.55a)$$

where \mathbf{T}_t and \mathbf{T}_r are corresponding to the pure translation and pure rotation, respectively, i.e.,

$$\mathbf{T}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{T}_r = \begin{bmatrix} C30^\circ & -S30^\circ & 0 & 0 \\ S30^\circ & C30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.55b)$$

Note that the order of multiplications in Eq. (5.55a) is similar to Eq. (5.31d), as the motions are specified with respect to the fixed frame F instead of the current frames.

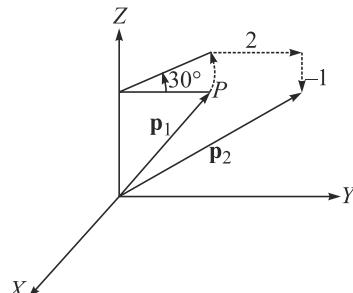


Fig. 5.25 Transfer of a point

Example 5.16 Product of two Homogeneous Transformation Matrices

Assume that \mathbf{T}_A and \mathbf{T}_B are given by

$$\mathbf{T}_A = \begin{bmatrix} C30^\circ & -S30^\circ & 0 & 2 \\ S30^\circ & C30^\circ & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{T}_B = \begin{bmatrix} C45^\circ & S45^\circ & 0 & 1 \\ -S45^\circ & C45^\circ & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.56)$$

which are graphically represented in Fig. 5.26. Note that each of the transformations, \mathbf{T}_A or \mathbf{T}_B , is a product of two elementary transformations representing translation and rotation with respect to the current frames. For example, the matrix \mathbf{T}_A can be written as,

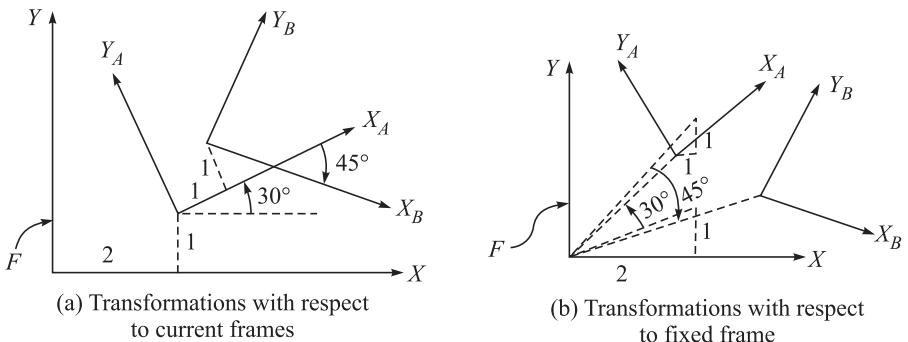


Fig. 5.26 Graphical interpretations of the products of two HTMs

$$\mathbf{T}_A = \mathbf{T}_{At} \mathbf{T}_{Ar} \quad (5.57a)$$

where the 4×4 homogeneous transformation matrices, \mathbf{T}_{At} and \mathbf{T}_{Ar} , are associated with the translation and rotation from the fixed frame F to the frame A , respectively, i.e.,

$$\mathbf{T}_{At} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{T}_{Ar} = \begin{bmatrix} C30^\circ & -S30^\circ & 0 & 0 \\ S30^\circ & C30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.57b)$$

The total transformation to reach the frame B from the frame F is now obtained as

$$\mathbf{T} = \mathbf{T}_A \mathbf{T}_B \quad (5.58a)$$

The 4×4 matrix \mathbf{T} is expressed as

$$\mathbf{T} = \begin{bmatrix} \frac{\sqrt{3}+1}{2\sqrt{2}} & \frac{\sqrt{3}-1}{2\sqrt{2}} & 0 & \frac{3+\sqrt{3}}{2} \\ -\frac{\sqrt{3}-1}{2\sqrt{2}} & \frac{\sqrt{3}+1}{2\sqrt{2}} & 0 & \frac{3+\sqrt{3}}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.58b)$$

Note that if the same transformations are performed with respect to the fixed frame F , as shown in Fig. 5.21(b), the order of multiplications has to be reversed, i.e.,

$$\mathbf{T} = [\mathbf{T}_B]_F [\mathbf{T}_A]_F \quad (5.59a)$$

where $[\mathbf{T}_A]_F$ and $[\mathbf{T}_B]_F$ are also obtained from the reverse-order multiplications of the translations and rotations, e.g., $[\mathbf{T}_A]_F = [\mathbf{T}_{Ar}]_F [\mathbf{T}_{At}]_F \cdots \mathbf{T}_{Ar}$ and \mathbf{T}_{At} being the 4×4 matrices given in Eq. (5.57b). Matrices $[\mathbf{T}_A]_F$, $[\mathbf{T}_B]_F$ and \mathbf{T} are given as follows:

$$[\mathbf{T}_A]_F = \begin{bmatrix} C30^\circ & -S30^\circ & 0 & \sqrt{3}-\frac{1}{2} \\ S30^\circ & C30^\circ & 0 & 1+\frac{\sqrt{3}}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, [\mathbf{T}_B]_F = \begin{bmatrix} C45^\circ & S45^\circ & 0 & \sqrt{2} \\ -S45^\circ & C45^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.59b)$$

$$\mathbf{T} = \begin{bmatrix} \frac{\sqrt{3}+1}{2\sqrt{2}} & \frac{\sqrt{3}-1}{2\sqrt{2}} & 0 & \frac{\sqrt{6}+2\sqrt{3}-1}{2} \\ -\frac{\sqrt{3}-1}{2\sqrt{2}} & \frac{\sqrt{3}+1}{2\sqrt{2}} & 0 & \frac{\sqrt{2}+2+\sqrt{3}}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.59c)$$

Comparing Eqs. (5.58b) and (5.59c), only the coordinates of the origin of frame B are different, as clear from Figs. 5.26(a-b).

Example 5.17 Verification of Eq. (5.59c) using MATLAB

In order to verify the equation, the following commands were used:

```
>> tam = [cos(pi/6), -sin(pi/6), 0, sqrt(3) - 1/2; sin(pi/6),
cos(pi/6), 0, 1 + sqrt(3)/2; 0, 0, 1, 0; 0, 0, 0, 1];
>> tbm = [cos(pi/4), sin(pi/4), 0, sqrt(2); -sin(pi/4), cos(pi/4),
0, 0; 0, 1, 0; 0, 0, 0, 1];>>
tm = tam*tbm
tm =
    0.9659    0.2588    0        2.4568
   -0.2588    0.9659    0        2.5731
    0          0        1.0000    0
    0          0        0        1.0000
```

where ‘tam’ and ‘tbm’ represent the matrices $[T_A]_F$ and $[T_B]_F$, respectively, whereas the resultant matrix $[T_A]_F$ is denoted by ‘tm.’ Equation (5.59c) can be easily verified to be the expression given by ‘tm.’

5.4 DENAVIT AND HARTENBERG (DH) PARAMETERS

A robot manipulator consists of several links connected by usually single degree-of-freedom (DOF) joints, say, a revolute or a prismatic joint. In order to control the end-effector with respect to the base, it is necessary to find the relation between the coordinate frames attached to the end-effector and the base. This can be obtained from the description of the coordinate transformations between the coordinate frames attached to all the links and forming the overall description in a recursive manner.

First appearance of DH parameters

The DH parameters were first appeared in Denavit and Hartenberg (1955) to represent a directed line which is nothing but the axis of a lower pair joint. The definitions used in this book are variants of the original definitions.

For this purpose, the material presented in the previous section for describing the position and orientation of the rigid body is useful for obtaining composition of coordinate transformations between the consecutive frames. As a first step, a systematic general method is to be derived to define the relative position and orientation of two consecutive links. The problem is to define two frames attached to two successive links and compute the coordinate transformation between them. In general, the frames are arbitrarily chosen as long as they are attached to the link they are referred to. Nevertheless, it is convenient to set some rules for the definition of the link frames. The convention adopted here for a serial chain robot shown in Fig. 5.27 is that it has $n + 1$ links, namely, link #0, ..., # n , coupled by n joints, i.e., joint 1, ..., n .

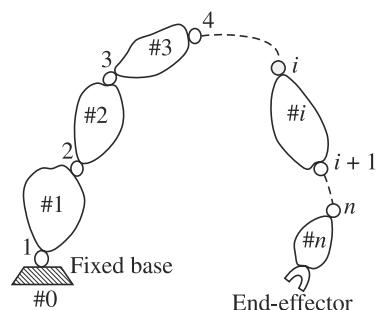
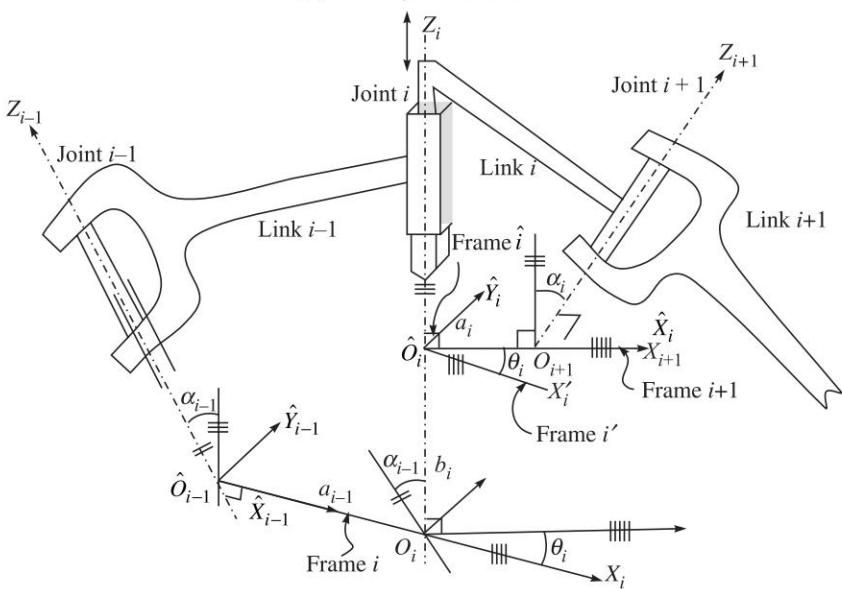
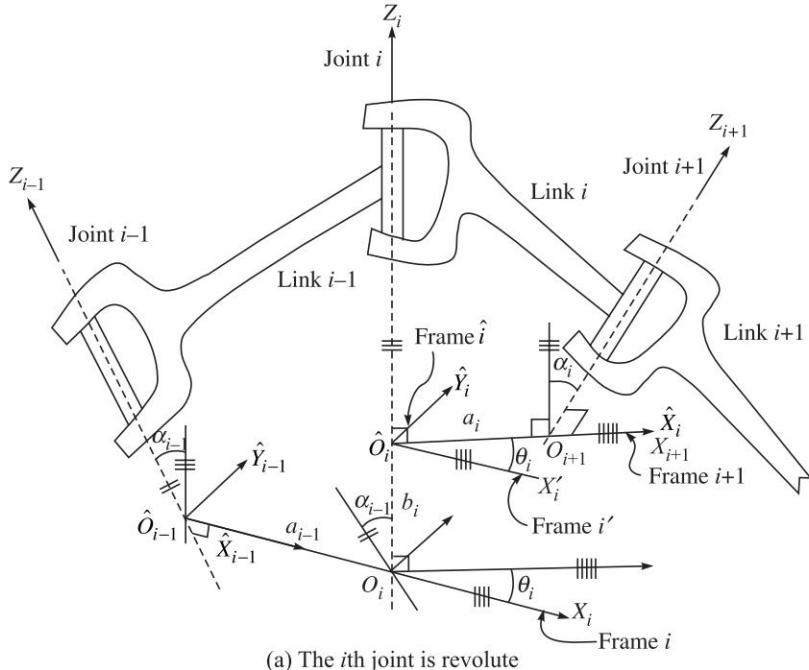


Fig. 5.27 Serial manipulator

Now, referring to Fig. 5.28,

- Let axis i denote the axis of the joint connecting the link $i - 1$ to the link i .
- A coordinate system X_i, Y_i, Z_i is attached to the end of the link $i - 1$ —not to the link i !—for $i = 1, \dots, n + 1$.



(b) The i th joint is prismatic

Fig. 5.28 Frame conventions and Denavit and Hartenberg (DH) parameters

- (c) Choose axis Z_i along the axis of joint i , whose positive direction can be taken towards either direction of the axis.
- (d) Locate the origin O_i at the intersection of the axis Z_i with the common normal to Z_{i-1} and Z_i . Also, locate \hat{O}_i on Z_i at the intersection of the common normal to Z_i and Z_{i+1} .
- (e) Choose the axis X_i along the common normal to axes Z_{i-1} and Z_i with the direction from the former to the later.
- (f) Choose the axis Y_i so as to complete a right-handed frame.

Note that the above conventions do not give a unique definition of the link frames in the following cases:

- For Frame 1 that is attached to the fixed base, i.e., link 0, only the direction of axes Z_1 is specified. Then O_1 and X_1 can be chosen arbitrarily.
- For the last frame $n + 1$, the foregoing conventions do not apply since there is no link $n + 1$. Thus, the frame $n + 1$ can be arbitrarily chosen.
- When two consecutive axes are parallel, the common normal between them is not uniquely defined.
- When two consecutive axes intersect, the direction of X_i is arbitrary.
- When the joint i is prismatic, only the direction of the axis Z_i is determined, whereas the location of O_i is arbitrary.

In all such cases, the indeterminacy can be exploited to simplify the procedure. For instance, the axes of the frame $n + 1$ can be made parallel to those of the frame n . Once the link frames have been established, the position and orientation of the frame i with respect to the frame $i - 1$ are completely specified by four parameters known as the Denavit and Hartenberg (DH) parameters. Hence, these frames are also referred as DH frames. The four DH parameters are defined as follows:

1. b_i (Joint Offset**)** Length of the intersections of the common normals on the joint axis Z_i , i.e., O_i and \hat{O}_i . It is the relative position of links $i - 1$ and i . This is measured as the distance between X_i and X_{i+1} along Z_i .

2. θ_i (Joint Angle**)** Angle between the orthogonal projections of the common normals, X_i and X_{i+1} , to a plane normal to the joint axis Z_i . Rotation is positive when it is made counter-clockwise. It is the relative angle between links $i - 1$ and i . This is measured as the angle between X_i and X_{i+1} about Z_i .

3. a_i (Link Length**)** Length between \hat{O}_i and O_{i+1} . This is measured as the distance along the common normal X_{i+1} to axes Z_i and Z_{i+1} .

4. α_i (Twist Angle**)** Angle between the orthogonal projections of joint axes Z_i and Z_{i+1} onto a plane normal to the common normal X_{i+1} . This is measured as the angle between the axes Z_i and Z_{i+1} about the axis X_{i+1} to be taken positive when rotation is made counter-clockwise.

Note that the above four parameters are defined sequentially as one moves from link $i - 1$ to link $i + 1$ through link i . Moreover, the first two parameters, namely, b_i and θ_i , define the relative position of links $i - 1$ and i , whereas the last two, a_i and α_i , describe the size and shape of link i that are always constant. Parameters b_i and θ_i are, however, variable depending on the type of joints in use. In particular,

- θ_i is variable if the joint i is revolute; and

- b_i is variable if the joint i is prismatic.

So, for a given type of joint, i.e., revolute or prismatic, one of the DH parameters is variable, which is called *joint variable*, whereas the other three remaining parameters are constant, called *link parameters*.

Example 5.18 DH Parameters of a Three-link Planar Arm

Figure 5.29 shows a three-link planar arm. The coordinate frames to define the DH parameters are shown in the figure. The DH parameters are tabulated in Table 5.3, where a_i and θ_i , for $i = 1, 2, 3$, are the link lengths and the joint angles, respectively.

Axis Z_i is perpendicular to the plane of the page and X_1 is chosen arbitrarily. Note that frame 1, i.e., X_1, Y_1 , and Z_1 , is fixed to the link denoted as #0. Since there is no link 4, frame 4 can be arbitrarily assigned so that its X_4 -axis is placed along the link, as done for frames 2 and 3.

Table 5.3 DH parameters of the three-link arm

Link	b_i	θ_i	a_i	α_i
1	0	θ_1 (JV)	a_1	0
2	0	θ_2 (JV)	a_2	0
3	0	θ_3 (JV)	a_3	0

JV: Joint Variable

Note that for a 2-link planar with both revolute joints, #3 is removed from Fig. 5.29. The corresponding DH parameters are those in Table 5.3 but without the third row.

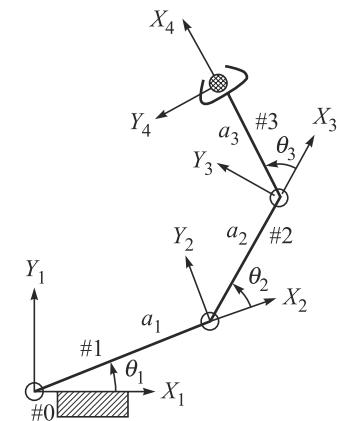
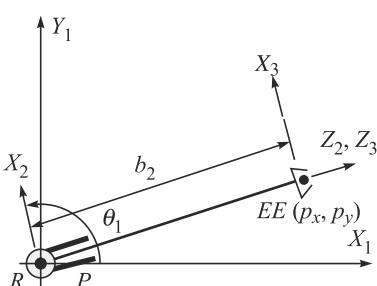


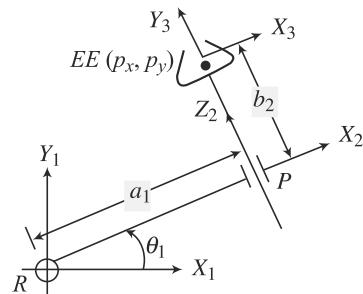
Fig. 5.29 A three-link planar arm

Example 5.19 DH Parameters of Revolute-Prismatic Planar Arms

Referring to two Revolute-Prismatic (RP) planar arms, Fig. 5.30, where the revolute and prismatic joints are indicated as R and P, respectively, the DH parameters are listed in Table 5.4.



(a) Intersecting joint axes



(b) Non-intersecting joint axes

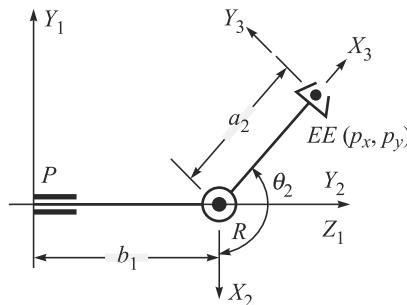
Fig. 5.30 Revolute-Prismatic planar arms

Table 5.4 DH parameters of the RP arm

Link	b_i	θ_i	a_i	α_i
(a) Intersecting joint axes				
1	0	θ_1 (JV)	0	$\pi/2$
2	b_2 (JV)	0	0	0
(b) Non-intersecting joint axes				
1	0	θ_1 (JV)	a_1	$-\pi/2$
2	b_2 (JV)	0	0	$\pi/2$

Example 5.20 DH Parameters of a Prismatic-Revolute Planar Arm

If the revolute and prismatic joints are interchanged, the result is a Prismatic-Revolute (PR) arm, as shown in Fig. 5.31. Its DH parameters are shown in Table 5.5.

**Fig. 5.31** Prismatic-Revolute planar arm**Table 5.5** DH parameters of the PR arm

Link	b_i	θ_i	a_i	α_i
1	b_2 (JV)	$-\pi/2$	0	$\pi/2$
2	0	θ_2 (JV)	a_2	0

Example 5.21 DH Parameters of a Spherical Arm

Referring to the spherical robot arm shown in Fig. 5.32, note that the first and second joint axes, namely, Z_1 and Z_2 , intersect. Hence, the first link length does not affect the end-effector motion due to the rotation of the first joint. So it is beneficial to put both the first and second frames at the intersection of the first two revolute axes, namely, at O_1 or O_2 . The DH parameters are tabulated in Table 5.6, where b_2 , b_3 , and θ_i , for $i = 1, 2, 3$, are indicated in Fig. 5.32.

Table 5.6 DH parameters of the spherical arm

Link	b_i	θ_i	a_i	α_i
1	0	θ_1 (JV)	0	$\pi/2$
2	b_2	θ_2 (JV)	0	$\pi/2$
3	b_3 (JV)	0	0	0

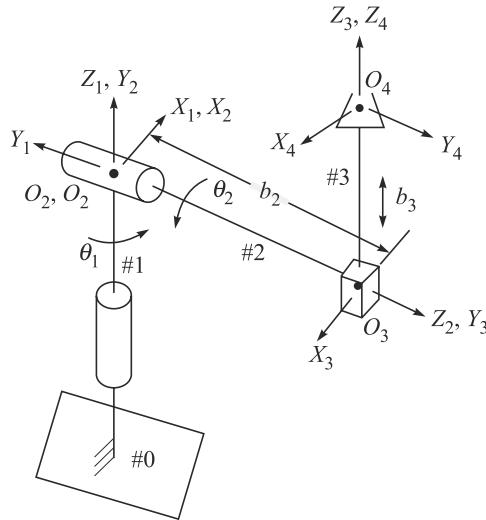


Fig. 5.32 A spherical arm

5.4.1 Transformation between DH Frames

At this point, it is possible to express the coordinate transformation between the DH frames i and $i + 1$, which are attached to links $i - 1$ and i , respectively. Referring to Fig. 5.28,

- (a) Translate frame i by b_i along axes Z_i . This brings the origin O_i of the frame, i , into coincidence with \hat{O}_i . Let \hat{O}_i be the origin of the displaced frame i' . The corresponding homogenous transformation matrix (HTM) is

$$\mathbf{T}_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.60a)$$

where subscript b on the left-hand side indicates the translation by amount b_i along the Z_i -axis.

- (b) The displaced frame, i.e., frame i' , is rotated by an angle θ_i about the axis Z_i , which brings the axis X'_i (parallel to X_i) to an alignment with the axis, X_{i+1} ($\equiv \hat{X}_i$). The new frame is denoted as frame \hat{i} , i.e., $\hat{O}_i - \hat{X}_i$ ($\equiv X_{i+1}$) $\hat{Y}_i Z_i$. The corresponding HTM is

$$\mathbf{T}_\theta = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.60b)$$

- (c) Translate the frame \hat{i} by a_i along the axis \hat{X}_i ($\equiv X_{i+1}$). This brings the origin of the frame \hat{i} at \hat{O}_i into coincidence with O_{i+1} .

The corresponding HTM is

$$\mathbf{T}_a = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.60c)$$

The new frame is denoted as frame i'' but not shown in the figures.

- (d) Finally, rotate the frame i'' by an angle α_i about the axis \hat{X}_i ($\equiv X_{i+1}$), which will now coincide with the frame $i + 1$. The corresponding HTM is

$$\mathbf{T}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.60d)$$

The resulting coordinate transformation between the frames connected to links i and $i-1$, namely, \mathbf{T}_i , is then obtained by post-multiplying the above four elementary transformations, i.e., HTMs given by Eqs. (5.60 a-d), as done in Eq. (5.32), i.e.,

$$\mathbf{T}_i = \mathbf{T}_b \mathbf{T}_\theta \mathbf{T}_a \mathbf{T}_\alpha \quad (5.61a)$$

The expression, \mathbf{T}_i , can also be read as the transformation matrix or the HTM of the frame attached to the link i , i.e., the frame $i + 1$, represented in the frame attached to the link $i - 1$, i.e., the frame i . Substituting the matrix expressions from Eqs. (5.60a-d) to Eq. (5.61a), the following expression is obtained:

$$\mathbf{T}_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.61b)$$

Notice that the transformation matrix from the frame i to the frame $i + 1$ is a function of only the i^{th} joint variable, i.e., θ_i for a revolute joint, and b_i for a prismatic joint, because the other three DH parameters are constants.

Example 5.22 Verify Eq. (5.61b) using MuPAD and Symbolic Commands of MATLAB

The commands in MuPAD to verify Eq. (5.61b) are shown in Fig. 5.33, whereas the same can be obtained using the symbolic commands of MATLAB as well. The symbolic commands and the corresponding output are shown below:

```
>> syms bi thi ai ali;
>> tbm = [1, 0, 0, 0; 0, 1, 0, 0; 0, 0, 1, bi; 0, 0, 0, 1];
>> tthm = [cos(thi), -sin(thi), 0, 0; sin(thi), cos(thi), 0, 0; 0, 1, 0; 0, 0, 0, 1];
>> tam = [1, 0, 0, ai; 0, 1, 0, 0; 0, 0, 1, 0; 0, 0, 0, 1];
>> talm = [1, 0, 0, 0; 0, cos(alii), -sin(alii), 0; 0, sin(alii), cos(alii), 0; 0, 0, 0, 1];
>> tim = tbm*tthm*tam*talm
```

where command ‘syms’ are used to define the symbolic variables, ‘bi,’ ‘thi,’ ‘ai,’ and ‘ali,’ for the DH parameters, b_i , θ_i , a_i , and α_i , respectively. The resulting output of the MATLAB commands is

```

tim =
[ cos(thi), -sin(thi)*cos(alii), sin(thi)*sin(alii), cos(thi)*ai]
[ sin(thi), cos(thi)*cos(alii), -cos(thi)*sin(alii), sin(thi)*ai]
[ 0, sin(alii), cos(alii), bi]
[ 0, 0, 0, 1]

**** This program in MuPAD computes Homogeneous Transformation Matrix ****
Define DH Parameters with subscript variables
[ bi := Symbol::subScript(b, i)
[ bi
[ thi := Symbol::subScript(`&theta;`, i)
[ theta_i
[ ai := Symbol::subScript(a, i)
[ ai
[ alii := Symbol::subScript(`&alpha;`, i)
[ alpha_i

Define Matrices
[ Tb:=matrix([[1,0,0,0],[0,1,0,0],[0,0,1,bi],[0,0,0,1]])
[ 
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & bi \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[ Tth:=matrix([[cos(thi),-sin(thi),0,0],[sin(thi),cos(thi),0,0],[0,0,1,0],[0,0,0,1]])
[ 
$$\begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[ Ta:=matrix([[1,0,ai],[0,1,0,0],[0,0,1,0],[0,0,0,1]])
[ 
$$\begin{pmatrix} 1 & 0 & 0 & ai \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[ Tal:=matrix([[1,0,0,0],[0,cos(alii),-sin(alii),0],[0,sin(alii),cos(alii),0],[0,0,0,1]])
[ 
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[ Ti:=Symbol::subScript(T, i)
[ T_i

Homogenous Transformation Matrix
[ T1:=Tb*Tth*Ta*Tal
[ 
$$\begin{pmatrix} \cos(\theta_i) & -\cos(\alpha_i) \sin(\theta_i) & \sin(\alpha_i) \sin(\theta_i) & a_i \cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i) \cos(\theta_i) & -\cos(\theta_i) \sin(\alpha_i) & a_i \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & b_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$


```

Fig. 5.33 MuPAD screenshot to find HTM in terms of DH parameters

Note the notations ‘tbm,’ ‘tthm,’ ‘tam,’ and ‘talm’ were used to represent the matrices \mathbf{T}_b , \mathbf{T}_θ , \mathbf{T}_a , and \mathbf{T}_α , given by Eqs. (5.60a-d), respectively. Finally, the resultant matrix \mathbf{T}_i , denoted as ‘tim’ in MATLAB environment, matches with Eq. (5.61b).

Example 5.23 Homogeneous Transformation Matrices of the Three-link Planar Arm

Referring to Fig. 5.29 and the DH parameters given in Table 5.3, the homogenous transformation matrices $\mathbf{T}_{i's}$, can be derived using Eq. (5.61b) as

$$\mathbf{T}_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_i C\theta_i \\ S\theta_i & C\theta_i & 0 & a_i S\theta_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ for } i = 1, 2, 3 \quad (5.62)$$

Example 5.24 Homogeneous Transformation Matrices of the RP Planar Arms

Referring to Fig. 5.30 and the DH parameters given in Table 5.4, the homogenous transformation matrices \mathbf{T}_i , for $i = 1, 2$, can be derived using Eq. (5.61b) as follows:

(a) For intersecting joint axes shown in Fig. 5.30(a)

$$\mathbf{T}_1 = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 0 \\ S\theta_1 & 0 & -C\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.63a)$$

(b) For non-intersecting joint axes shown in Fig. 5.30(b)

$$\mathbf{T}_1 = \begin{bmatrix} C\theta_1 & 0 & -S\theta_1 & a_1 C\theta_1 \\ S\theta_1 & 0 & C\theta_1 & a_1 S\theta_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.63b)$$

Example 5.25 Homogeneous Transformation Matrices of the PR Planar Arm

Referring to Fig. 5.31 and the DH parameters given in Table 5.5, the homogenous transformation matrices \mathbf{T}_i , for $i = 1, 2$, can be derived from Eq. (5.61b) as

$$\mathbf{T}_1 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{T}_2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_2 C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2 S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.64)$$

Example 5.26 Homogeneous Transformation Matrices of the Spherical Arm

Referring to Fig. 5.32 and the DH parameters given in Table 5.6, the homogenous transformation matrices \mathbf{T}_i , for $i = 1, 2, 3$, can be derived from Eq. (5.61b) as

$$\mathbf{T}_1 = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 0 \\ S\theta_1 & 0 & -C\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} C\theta_2 & 0 & S\theta_2 & 0 \\ S\theta_2 & 0 & -C\theta_2 & 0 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.65)$$

5.4.2 Visualization of DH Parameters using RoboAnalyzer¹

It is emphasized here that the concept of the DH frames and its associated DH parameters may not be conceptually very difficult but it is confusing. Moreover, keeping track of successive frames to define the associated DH parameters with appropriate indices is prone to making mistakes. Hence, it is useful if there is a real model to visualize the DH frames. With a real model, it is not always possible to visualize all DH frames as some lie inside the physical links. In order to overcome the above difficulty in visualizing the DH frames and parameters, a feature was created in one of the modules of RoboAnalyzer (RA) software. It is called “Visualize DH.” A screenshot of the RA showing DH frames and highlighting the parameters in the DH table is shown in Fig. 5.34.

5.5 A VARIANT OF DH PARAMETERS

In this section, a variant of the DH parameters defined above will be explained, which are used by many researchers and have appeared in many textbooks, e.g., in Ghosal (2006), Craig (2009), and others. Sometimes it is difficult for the readers who are acquainted with the DH parameters followed in this book to understand certain topics of other books which use different definitions. In reality, there are a few more definitions of DH parameters, e.g., in Khalil and Kleinfinger (1986), Shah et al. (2013), and others. Those used by Ghosal (2006) and Craig (2009) are nothing but a variant of the modified DH (MDH) parameters proposed by Khalil and Kleinfinger (1986). These are referred in this book as MDH-V parameters and will be correlated with the definitions given in Section 5.4. This will allow the users of MDH-V parameters to use RoboAnalyzer software easily to verify numerical results and understand the concepts in a more lucid manner.

1. RoboAnalyzer is a teaching/learning software developed at IIT Delhi. It is available free from <http://www.roboanalyzer.com>

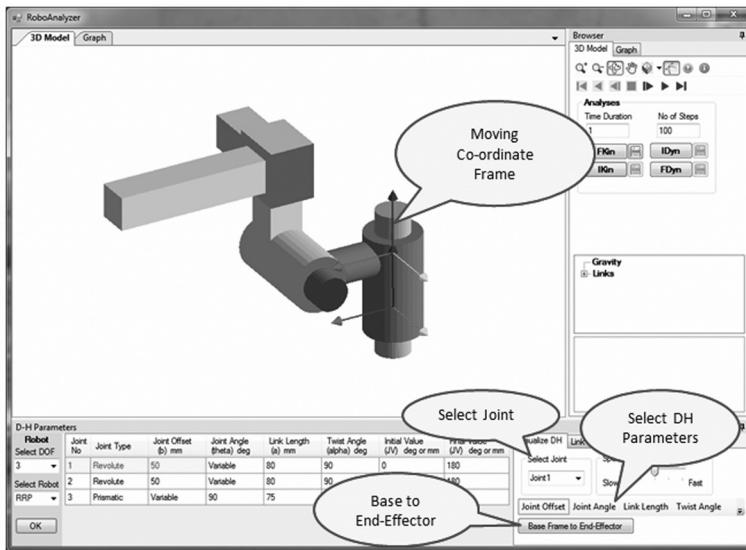


Fig. 5.34 RoboAnalyzer screenshot for “Visualize DH”

5.5.1 Definitions of MDH-V Parameters

Referring to Fig. 5.28, the coordinate frame $\hat{O}_i - \hat{X}_i\hat{Y}_iZ_i$, denoted here as the frame \hat{i} is attached to the link i , instead of link $(i-1)$. The common normal between Z_i and Z_{i+1} is then defined as the axis \hat{X}_i . Accordingly, the origin \hat{O}_i of the frame \hat{i} is located at a point where \hat{X}_i intersects Z_i . The four parameters, denoted here as MDH-V parameters, are now defined with reference to Figs. 5.28(a) or (b):

- (a) α_{i-1} (**Twist Angle**): Angle between Z_{i-1} and Z_i about \hat{X}_{i-1}
- (b) a_{i-1} (**Link Length**): Distance from Z_{i-1} to Z_i along \hat{X}_{i-1}
- (c) θ_i (**Joint Angle**): Angle between \hat{X}_{i-1} and \hat{X}_i about Z_i
- (d) b_i (**Joint Offset**): Distance from \hat{X}_{i-1} and \hat{X}_i along Z_i

Modified DH parameters

The concept of DH parameters given by Denavit and Hartenberg (1955) were later extended by Khalil and Kleinfinger (1986) to account for a link connecting to more than two links, as happens in a tree-type system, in an unambiguous manner. These parameters are referred in the literature as the modified DH parameters.

Similar to the definitions of the DH parameters given in Section 5.4, θ_i or b_i is also a *variable* in the definitions of MDH-V parameters depending on the type of joints, i.e., revolute or prismatic, respectively, whereas the other three are *constants*. The position and the orientation between any two frames, say, $\hat{i} - 1$ and \hat{i} , can then be specified using above four parameters denoted here as the MDH-V parameters. Two of the four corresponding homogeneous transformation matrices (HTMs) associated to the parameters α_{i-1} and a_{i-1} , denoted respectively as \hat{T}_α and \hat{T}_a , are given by

$$\hat{\mathbf{T}}_\alpha \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_{i-1} & -S\alpha_{i-1} & 0 \\ 0 & S\alpha_{i-1} & C\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \hat{\mathbf{T}}_a \equiv \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.66)$$

The other two HTMs associated to the joint offset b_i and the joint angle θ_i are same as in Eqs. (5.60a-b), respectively, because their definitions have not changed. Hence, the overall HTM to reach the frame \hat{i} from $\hat{i}-1$ is thought of the following sequence of four movements: (i) Rotation about \hat{X}_{i-1} ($\equiv X_i$) by an angle α_{i-1} ; (ii) Translation along the same axis by an amount a_{i-1} . This will take the frame $\hat{i}-1$ to an intermediate frame i , i.e., $O_i - X_i (\equiv \hat{X}_{i-1})Y_iZ_i$; (iii) Rotation by an angle θ_i about Z_i ; and (iv) Translation along the same axis by an amount b_i . This will lead to the final frame \hat{i} . The mathematical representation for the result of the above four sequential movements can be obtained from $\hat{\mathbf{T}}_i = \hat{\mathbf{T}}_\alpha \hat{\mathbf{T}}_a \mathbf{T}_\theta \mathbf{T}_b$, where the 4×4 HTM, $\hat{\mathbf{T}}_i$, has the following representation:

$$\hat{\mathbf{T}}_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -b_i S\alpha_{i-1} \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & b_i C\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.67)$$

It is clear from Eq. (5.67) that there is a mixture of subscripts i and $i-1$ which is sometimes confusing. This is not the case with the expression given by Eq. (5.61b). Hence, Eq. (5.61b) and the associated definitions of the DH parameters will be used throughout this book and have been adopted for the description of robots in RoboAnalyzer software.

5.5.2 Correlation between the DH and MDH-V Parameters

For an n -degree-of-freedom robot, one can find the correlation between all the $4n$ scalar DH parameters defined in Section 5.4 and the MDH-V parameters defined in Section 5.5.1. To do this, it is necessary to assign all the coordinate frames attached to the links based on the DH and MDH-V notations. Note that, as per the DH notation, there will be frames numbered from 1 or $n+1$, whereas there are frames 0 to n as per the MDH-V notation. To correlate the two sets of parameters, it is important to assign the frames numbered from 0 to $n+1$ using either of the conventions. Next, a six-dimensional vector \mathbf{d} is introduced as follows:

$$\mathbf{d} \equiv \begin{bmatrix} \mathbf{d}_{i-1}^c \\ \mathbf{d}_i^v \\ \mathbf{d}_i^c \end{bmatrix}, \text{ where } \mathbf{d}_{i-1}^c \equiv \begin{bmatrix} a_{i-1} \\ \alpha_{i-1} \end{bmatrix}, \mathbf{d}_i^v \equiv \begin{bmatrix} b_i \\ \theta_i \end{bmatrix}; \text{ and } \mathbf{d}_i^c \equiv \begin{bmatrix} a_i \\ \alpha_i \end{bmatrix} \quad (5.68)$$

In Eq. (5.68), the six scalar parameters are those defined either as per DH or MDH-V notations. Note from Fig. 5.28 that the above parameters are actually same but have different nomenclatures. The four DH parameters followed in this book are then nothing but the last four elements of the 6-dimensional vector \mathbf{d} , whereas the

MDH-V parameters are those first four elements of \mathbf{d} . Mathematically, they can be expressed as

$$\mathbf{d}_4 \equiv \mathbf{S}\mathbf{d}, \text{ and } \hat{\mathbf{d}}_4 \equiv \hat{\mathbf{S}}\mathbf{d} \quad (5.69a)$$

where \mathbf{d}_4 and $\hat{\mathbf{d}}_4$ correspond to four parameters according to the definitions of DH and MDH-V notations, respectively, whereas \mathbf{S} and $\hat{\mathbf{S}}$ are the associated 4×6 matrices to extract the set of four parameters from the 6-dimensional vector \mathbf{d} . Matrices \mathbf{S} and $\hat{\mathbf{S}}$ are defined below:

$$\mathbf{S} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{S}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \quad (5.69b)$$

In Eq. (5.69b), $\mathbf{1}$ and $\mathbf{0}$ are the 2×2 identity and zero matrices, respectively.

Example 5.27 Correlation between DH and MDH-V Parameters of PUMA 560 Robot

Referring to PUMA robot of Fig. 5.35, seven coordinate frames are attached as per the definitions of Sections 5.4 and 5.5. Then, the six-set parameters associated to those appearing in vector \mathbf{d} of Eq. (5.68) are tabulated in Table 5.7 as per the definitions given in Section 5.4 or 5.5.

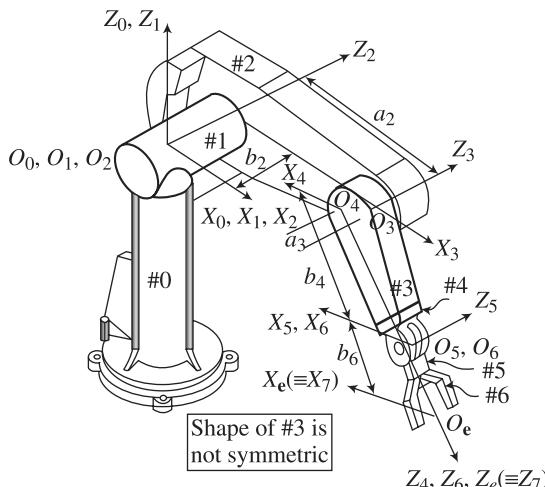


Fig. 5.35 PUMA 560 and its frames

One can now easily input the DH parameters from Table 5.7 in the RoboAnalyzer software to generate numerical results and visualize the robot's motion through animation, whereas MDH-V parameters are same as reported in the book of Ghosal (2006) and Craig (2009).

Table 5.7 DH and MDH-V parameters of PUMA 560

i	\mathbf{d}_{i-1}^c		\mathbf{d}_i^r		\mathbf{d}_i^e			
			DH parameters					
MDH-V parameters								
	a_{i-1}	α_{i-1}	b_i	θ_i	a_i	α_i		
1	0	0	0	θ_1	0	$-\pi/2$		
2	0	$-\pi/2$	0	θ_2	a_2	0		
3	a_2	0	b_3	θ_3	a_3	$-\pi/2$		
4	a_3	$-\pi/2$	b_4	θ_4	0	$\pi/2$		
5	0	$\pi/2$	0	θ_5	0	$-\pi/2$		
6	0	$-\pi/2$	0	θ_6	0	0		

5.6 DH PARAMETRIZATION OF EULER ANGLES

Although DH representation (DH or MDH-V) has been extensively used in modeling and analysis of robot motions and has become almost a standard practice, some researchers, e.g., Niku (2001) and others, find a fundamental flaw with this technique. The problem is that all motions are about X - and Z -axes. In its present form, as explained above, it cannot represent any motion about the Y -axis. Hence, the Euler angles which define rotations about all axes, including the Y -axis, cannot be typically represented using the DH parameters. Recently, Shah et al. (2012) have showed how all 12 sets of Euler angles can be represented using the DH parameters to take advantage of their inherent benefits, i.e., considering any general rotation as a combination of one DOF motions only. This process is referred as *DH parametrization of Euler angles*. Due to involved derivations and associated discussions required, they are not included in this book. However, their detailed treatments are available in Shah et al. (2012, 2013).

SUMMARY

In this chapter, first the rigid-body pose or configuration is defined. Their representations are explained. Several methods to represent a rotation are presented. It is followed by the definition of homogeneous transformation matrix (HTM) that takes care of both translation and rotation of a coordinate frame with respect to another one. Next, the Denavit and Hartenberg (DH) parameters and its one variant are explained. Associated HTMs are derived. A correlation between the DH and MDH-V parameters is also established which would allow the users of MDH-V parameters to effectively use RoboAnalyzer and similar software.

EXERCISES

- 5.1** Prove the following:
- $\det(\mathbf{Q}) = 1$ (5.70a)
 - $\mathbf{Q}^{-1} = \mathbf{Q}^T$ (5.70b)
 - $\mathbf{Q}' = \mathbf{Q}^T$ (5.70c)
 - For elementary rotation by α about any axis, X , Y , or Z , $\mathbf{Q}_k = \mathbf{Q}_k^T$, where $k = X, Y$, or Z .
- 5.2** What are the equivalent meanings of a rotation matrix?
- 5.3** Let \mathbf{Q}_A be the rotation of the fixed frame to a new frame A with respect to the frame F , and \mathbf{Q}_B be the rotation of the frame A to another new frame B with respect to the frame A . What is the resultant matrix representation of the frame B with respect to their original reference frame F , i.e., \mathbf{Q} ?
- 5.4** Let $[\mathbf{Q}_A]_F$ be the rotation of the fixed frame to a new frame A with respect to the frame F , and $[\mathbf{Q}_B]_F$ be the rotation of frame A to another new frame B , also with respect to the frame F . Find the resultant matrix representation of the frame B with respect to the original reference frame F , i.e., \mathbf{Q} ? Should the result be same as obtained in Exercise 5.3 above?
- 5.5** Assume that \mathbf{Q}_A and \mathbf{Q}_B , as defined in Exercise 5.3 above, are given by
- $$\mathbf{Q}_{C45^{\circ}A} = \begin{bmatrix} C30^{\circ} & -S30^{\circ} & 0 \\ S30^{\circ} & C30^{\circ} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{Q}_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C45^{\circ} & -S45^{\circ} \\ 0 & S45^{\circ} & C45^{\circ} \end{bmatrix} \quad (5.71)$$
- Illustrate the above two transformations graphically, and find $\mathbf{Q} = \mathbf{Q}_A \mathbf{Q}_B$.
- 5.6** What are the ZYZ Euler angles for the overall rotation representation by \mathbf{Q} of Exercise 5.5?
- 5.7** Repeat Exercise 5.6 to find ZYZ fixed-axes rotation angles.
- 5.8** If \mathbf{T}_A be the homogeneous matrix representing a transformation of the fixed frame to a new frame A with respect to the fixed frame F and \mathbf{T}_B be another transformation to a frame B with respect to the frame A , find the resultant transformation matrix \mathbf{T} .
- 5.9** If $[\mathbf{T}_A]_F$ be the homogeneous matrix representing a transformation of the fixed frame to a new frame A with respect to the frame F , and $[\mathbf{T}_B]_F$ be another transformation to a frame B , also with respect to the frame F , find the resultant transformation matrix. Comment on the results obtained in this question and in the previous question.
- 5.10** Find out the DH parameters of the SCARA robot shown in Fig. 5.36.

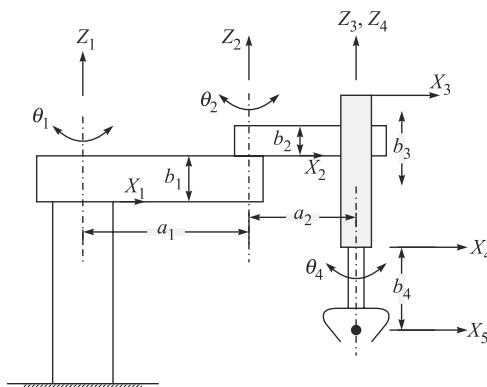


Fig. 5.36 A SCARA robot

- 5.11** Figure 5.37 shows an anthropomorphic articulated arm. Find its DH parameters.
- 5.12** Find the homogeneous transformation matrices for the robot architecture shown in Fig. 5.37.
- 5.13** Find the DH parameters of a spherical arm shown in Fig. 5.38.
- 5.14** What are the homogeneous transformation matrices for the robot architecture shown in Fig. 5.38.

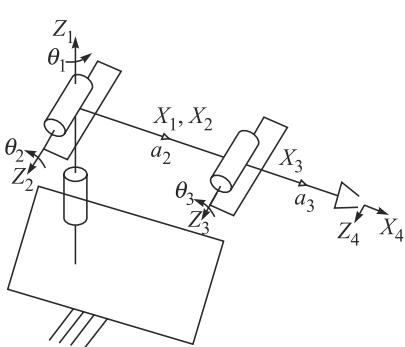


Fig. 5.37 An articulated arm

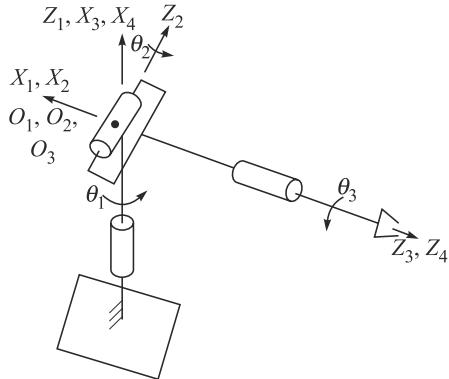


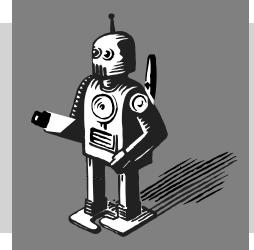
Fig. 5.38 A spherical wrist

MATLAB AND ROBOANALYZER BASED EXERCISES

- 5.15** Evaluate the numerical values of \mathbf{Q}_A and \mathbf{Q}_B given in Eq. (5.71).
- 5.16** Find the homogeneous transformation matrix (HTM) for the spherical arm using the DH parameters of Table 5.6.
- 5.17** Using the numerical values in Exercise 5.5 show the noncommutative property of the rotations, i.e., $\mathbf{Q}_A\mathbf{Q}_B \neq \mathbf{Q}_B\mathbf{Q}_A$.
- 5.18** Using the functions, ‘atan’ and ‘atan2,’ find the angle formed by the line joining the origin $(0, 0)$ and a point $(2, 1)$.
- 5.19** Repeat Exercise 5.18 for the points $(0, 0)$ and $(-2, -1)$. Comment on the results obtained using ‘atan’ and ‘atan2.’ Which one is the correct one?
- 5.20** Symbolically compute the 12 sets of Euler angles.
- 5.21** Using RoboAnalyzer (RA) software, generate the spherical robot arm using the DH parameters of Table 5.6.
- 5.22** Visualize DH parameters of the spherical robot arm using RA.
- 5.23** What are the default DH parameters of PUMA 560 in RA software?
- 5.24** Generate a 6R robot model in RA using the DH parameters of Exercise 5.23.
- 5.25** Note down individual HTM of the Stanford robot from “Link Config” of RA software. Compare those in MATLAB using the default DH parameters in RA screen and Eq. (5.61b).

6

Kinematics



For a robot to perform a specific task, the position and orientation of the end-effector, i.e., its pose or configuration, relative to the base should be established first. This is essential for positioning problem. When the above relations are differentiated once and twice then the problems of velocity and acceleration analyses arise, respectively, which are required for smooth motion control of the end-effector and dynamic analysis of the robot at hand. Since the configuration of the end-effector is determined by the six *Cartesian variables* that are controlled by the robot's joint motions, it is necessary to find the relations between the two sets. In position analysis, a relation between the Cartesian coordinates, i.e., the position of a point on the end-effector and its orientation, and the joint angles are found. Here, two types of problems exist, namely, *forward* or *direct*, and *inverse*, as illustrated in Fig. 6.1. In forward kinematics, the joint positions are given and the problem is to find the end-effector's configuration. In inverse kinematics, the reverse problem is solved, i.e., if the configuration of the end-effector is given, the problem is to find the joint angles.

Forward vs. Inverse Kinematics

Forward kinematics of a serial robot admits only one solution, whereas the inverse kinematics has multiple solutions.

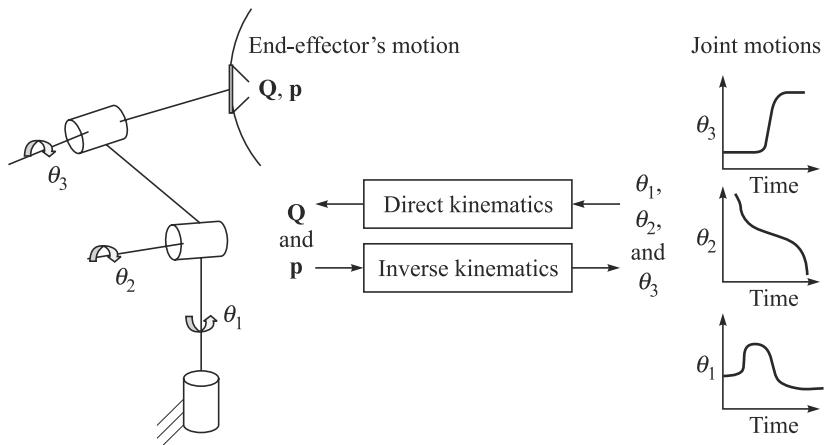


Fig. 6.1 Forward and inverse kinematics

6.1 FORWARD POSITION ANALYSIS

In forward kinematics for positions, the joint positions, i.e., the angles of revolute joints and the displacements of prismatic joints, are prescribed. The task is to find the end-effector's configuration or pose, i.e., its position and orientation. This can be obtained from the *closure equations*, as explained in the following steps:

- According to the rules given in Section 5.4, attach coordinate frames in each of the $n + 1$ links of the robot with frame 1 attached to the fixed link (#0 of Fig. 5.27), and frame $n + 1$ to the end-effector or the n th link (# n of Fig. 5.27).
- Define the Denavit and Hartenberg (DH) parameters, as presented in Section 5.4.
- Write the homogeneous transformation matrices (HTMs) from Eq. (5.61b) as $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_n$, where \mathbf{T}_i , for $i = 1, \dots, n$, represents the transformation of body i or frame $i + 1$ with respect to its previous link $i - 1$ or the frame attached to it, i.e., frame i .
- Similar to Section 5.4.1 to obtain the individual HTM from the four elementary transformations corresponding to the DH parameters, the HTM of the end-effector (frame $n + 1$) with respect to frame 1, i.e., \mathbf{T} , is now obtained from the post-multiplication of the above individual HTM, i.e., \mathbf{T}_i , for $i = 1, \dots, n$. The result is given by

$$\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2 \dots \mathbf{T}_n \quad (6.1)$$

Note that each transformation on the right-hand side of Eq. (6.1), \mathbf{T}_i , is expressed in the coordinate frame attached to link $i - 1$ or frame i , whereas the left-hand side of matrix \mathbf{T} is expressed in the fixed frame, i.e., frame 1. Hence, proper care must be taken by appropriately transferring the associated vectors and matrices to the successive frames before multiplying or adding them which are represented in the latter frames. Equation (6.1) is known as the closure equation of the robot at hand. If the expression for the HTM, \mathbf{T}_i of Eq. (5.61b) written in the form of Eq. (5.47), is substituted in Eq. (6.1), two distinct relations in terms of the orientation of the links and the positions of the origins of the frames attached to the links are obtained. They are

$$\mathbf{Q} = \mathbf{Q}_1 \mathbf{Q}_2 \dots \mathbf{Q}_n \quad (6.2)$$

$$\mathbf{p} = \mathbf{a}_1 + \mathbf{Q}_1 \mathbf{a}_2 + \dots + \mathbf{Q}_{n-1} \mathbf{a}_n \quad (6.3)$$

where matrix \mathbf{Q}_i is the orthogonal rotation matrix explained in Section 5.2.2, which represents the orientation of frame $i + 1$ with respect to frame i . Moreover, the vector \mathbf{a}_i is the position of the origin of frame $i + 1$ attached to link i from that of frame i attached to link $i - 1$. Furthermore, \mathbf{Q} is the orientation of the end-effector with respect to the fixed frame, namely, frame 1, and \mathbf{p} is the position of the origin of the frame attached to the end-effector, i.e., frame $n + 1$, from the origin of frame 1. Comparing both sides of Eq. (6.1) or Eqs. (6.2–6.3), it is noted that the former is easy to apprehend, whereas the latter equations are computationally efficient. Using Eq. (6.1), one has to compute 16 parameters of the 4×4 matrices, whereas using Eqs. (6.2–6.3), only 12 parameters, nine for the 3×3 matrices and three for the position vectors, are calculated. Note that in forward position analysis, the right-hand sides of Eq. (6.1) or Eqs. (6.2–6.3) are given as inputs and the left-hand sides are calculated as outputs.

Example 6.1 Forward Kinematics of a Two-link Planar Arm

The DH parameters of a two-link arm shown in Fig. 6.2 can be extracted from Table 5.3, as they are the first two rows. Its two homogeneous transformation matrices \mathbf{T}_i , $i = 1, 2$, are then given using Eq. (5.61b), i.e.,

$$\mathbf{T}_i \equiv \begin{bmatrix} c_i & -s_i & 0 & a_i c_i \\ s_i & c_i & 0 & a_i s_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.4)$$

where $s_i \equiv \sin \theta_i$ and $c_i \equiv \cos \theta_i$. The solution to the forward kinematic position analysis of the arm is then given as, $\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2$, where \mathbf{T} represents the end-effector's configuration with respect to its base, i.e., frame 1. The 4×4 matrix \mathbf{T} is derived as

$$\mathbf{T} \equiv \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.5)$$

where $s_{12} \equiv \sin \theta_{12}$; $c_{12} \equiv \cos \theta_{12}$; and $\theta_{12} \equiv \theta_1 + \theta_2 - \theta_i$ being the joint angle of the i th joint, as indicated in Fig. 6.2. Moreover, $a_1 c_1 + a_2 c_{12} \equiv p_x$ and $a_1 s_1 + a_2 s_{12} \equiv p_y$, where p_x and p_y are shown in Fig. 6.2. Knowing the numerical values of the DH parameters, namely, a_1 and a_2 , and the input values of θ_1 and θ_2 , one can easily calculate matrix \mathbf{T} , which will specify the position of the point P on the end-effector and the orientation of the end-effector, i.e., φ .

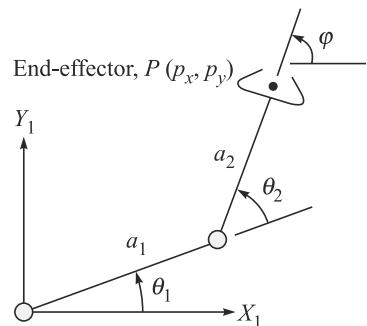


Fig. 6.2 Kinematics of a two-link planar arm

Example 6.2 Forward Kinematics of Revolute-Prismatic Planar Arms

Two Revolute-Prismatic (RP) planar arms are shown in Fig. 5.30, whose DH parameters are evaluated in Table 5.4. Its HTMs, \mathbf{T}_1 and \mathbf{T}_2 , are also given in Eq. (5.63a-b). Hence, the forward kinematics relation given by the 4×4 matrix, $\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2$, which can be expressed as follows:

(a) For intersecting joint axes shown in Fig. 5.30(a),

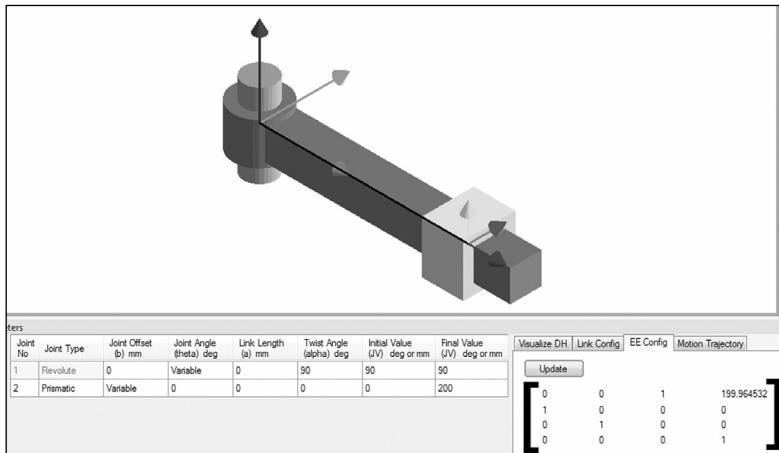
$$\mathbf{T} \equiv \begin{bmatrix} c_1 & 0 & s_1 & b_2 s_1 \\ s_1 & 0 & -c_1 & -b_2 c_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.6a)$$

(b) For the RP arm shown in Fig. 5.30(b) with non-intersecting joint axes, the axis of the prismatic joint is at a distance along the first link and orthogonal to it. Its DH parameters are shown in Table 5.4(b), whereas the individual HTMs are shown in Eq. (5.63b). The overall HTM of the end-effector with respect to its base is then given by

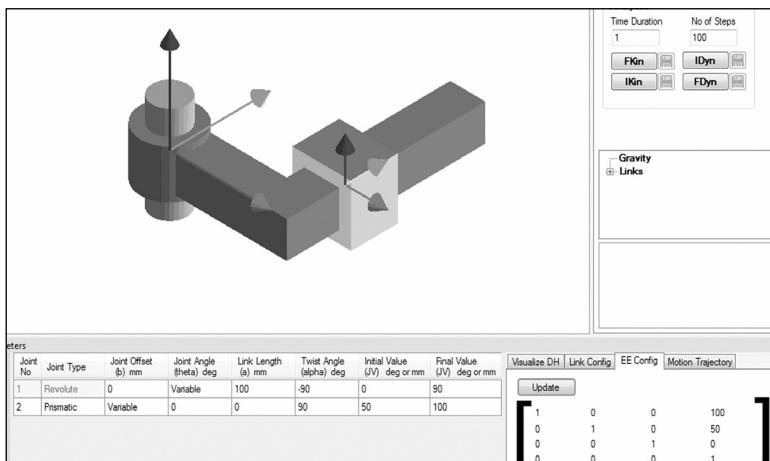
$$\mathbf{T} \equiv \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 - b_2s_1 \\ s_1 & c_1 & 0 & a_1s_1 + b_2c_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.6b)$$

Example 6.3 Forward Kinematics of Revolute-Prismatic Planar Arms in RoboAnalyzer

The motions of the RP planar arms, Figs. 5.30(a-b), are now displayed in the RoboAnalyzer environment whose screenshots are shown in Figs. 6.3(a-b). The numerical values of the DH parameters and the corresponding values of the end-effector's HTM, as per Eqs. 6.6(a-b), are appearing in Figs. 6.3(a-b).



(a) Intersecting joint axes



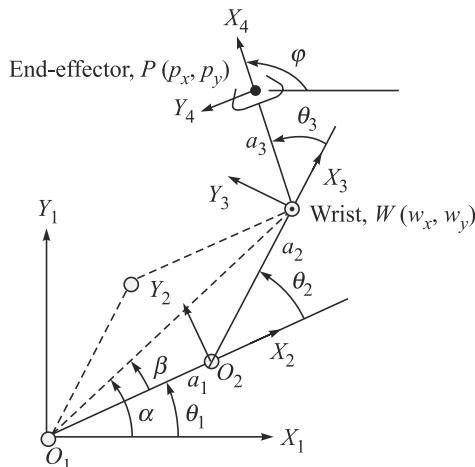
(b) Non-intersecting joint axes

Fig. 6.3 Screenshot of RoboAnalyzer for RP arms

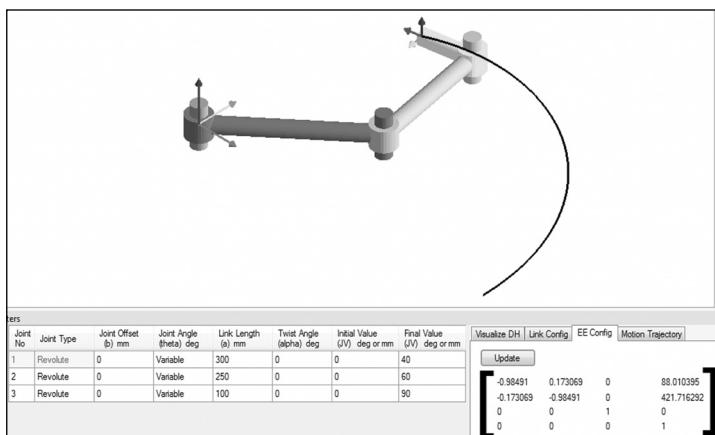
Example 6.4 Forward Kinematics of a Three-link Planar Arm

The DH parameters of the arm, as shown in Fig. 6.4 were obtained in Table 5.3. The three HTM, $\mathbf{T}_i, i = 1, 2, 3$, are given from Eq. (5.62). They are basically same as shown in Eq. (6.4). The result of the forward kinematic position analysis of the arm is then given by, $\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3$, where \mathbf{T} represents the end-effector's pose or configuration with respect to its base, i.e., frame 1. The 4×4 matrix \mathbf{T} is derived as

$$\mathbf{T} \equiv \begin{bmatrix} c_{123} & -s_{123} & 0 & a_1c_1 + a_2c_{12} + a_3c_{123} \\ s_{123} & c_{123} & 0 & a_1s_1 + a_2s_{12} + a_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.7)$$



(a) Three-link arm



(b) RoboAnalyzer screenshot

Fig. 6.4 Kinematics of a three-link planar arm

where s_{12} , c_{12} and θ_{12} are defined after Eq. (6.5). Moreover, $s_{123} \equiv \sin \theta_{123}$; $c_{123} \equiv \cos \theta_{123}$, and $\theta_{123} \equiv \theta_{12} + \theta_3 \equiv \varphi - \theta_i$ and φ being shown in Fig. 6.4. Furthermore, $a_1 c_1 + a_2 c_{12} + a_3 c_{123} \equiv p_x$ and $a_1 s_1 + a_2 s_{12} + a_3 s_{123} \equiv p_y$, where p_x and p_y are indicated in Fig. 6.4.

Note that the direct use of Eqs. (6.2) and (6.3) for the planar case also provides the expressions, $\varphi = \theta_{123}$, and above relations of p_x and p_y , respectively, which can be shown to be computationally more efficient. A RoboAnalyzer screenshot of the three-link arm is shown in Fig. 6.4(b).

Example 6.5 Forward Kinematics of a SCARA Robot

A SCARA robot is shown in Fig. 5.36. Based on its DH parameters found in Exercise 5.10, its homogeneous transformation matrices are obtained as follows:

$$\mathbf{T}_1 \equiv \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & b_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_2 \equiv \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.8a)$$

$$\mathbf{T}_3 \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_4 \equiv \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & -b_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.8b)$$

The resultant forward kinematics for position then can be obtained as, $\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3 \mathbf{T}_4$, which yields

$$\mathbf{T} \equiv \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{124} & c_{124} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & b_1 + b_2 + b_3 - b_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.9)$$

where $s_{124} \equiv \sin \theta_{124}$, $c_{124} \equiv \cos \theta_{124}$, and $\theta_{124} \equiv \theta_{12} + \theta_4$.

Example 6.6 Forward Kinematics of a Spherical Arm

The kinematic diagram of a spherical arm introduced in Fig. 2.13 is shown in Fig. 5.32. Its DH parameters are given in Table 5.6, whereas the HTM is given in Example 5.26. From Eq. (5.65), the forward kinematics relation is derived as

$$\mathbf{T} = \begin{bmatrix} c_1 c_2 & s_1 & c_1 s_2 & b_2 s_1 + b_3 c_1 s_2 \\ s_1 c_2 & -c_1 & s_1 s_2 & -b_2 c_1 + b_3 s_1 s_2 \\ s_2 & 0 & -c_2 & -b_3 c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.10)$$

Example 6.7 Forward Kinematics of an Anthropomorphic Articulated Arm

The DH parameters and the HTMs of the anthropomorphic articulated arm of Fig. 5.37 are obtained in Exercises 5.11 and 5.12, respectively. The three corresponding HTMs, i.e., $\mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3$, are evaluated as follows:

$$\mathbf{T}_1 \equiv \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_2 \equiv \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_3 \equiv \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.11a)$$

The resulting HTM is then obtained as $\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3$ representing the configuration of the end-effector represented in the fixed frame. It is given by

$$\mathbf{T} = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 & c_1 (a_2 c_2 + a_3 c_{23}) \\ s_1 c_{23} & -s_1 s_{23} & -c_1 & s_1 (a_2 c_2 + a_3 c_{23}) \\ s_{23} & c_{23} & 0 & a_2 s_2 + a_3 s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.11b)$$

The result of Eq. 6.11(b) will be used for its inverse kinematic solutions later.

Example 6.8 Forward Kinematics of a Wrist

The DH parameters and the HTM of the wrist shown in Fig. 5.38 are evaluated in Exercises 5.13 and 5.14, respectively. It has three degrees of freedom. The HTMs \mathbf{T}_1 , \mathbf{T}_2 and \mathbf{T}_3 are evaluated as

$$\mathbf{T}_1 \equiv \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_2 \equiv \begin{bmatrix} c_2 & 0 & -s_2 & 0 \\ s_2 & 0 & c_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_3 \equiv \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.12a)$$

The overall HTM is then given as $\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3$, where \mathbf{T}_i 's are appearing in Eq. 6.12(a). The forward kinematics result, matrix \mathbf{T} , is as follows:

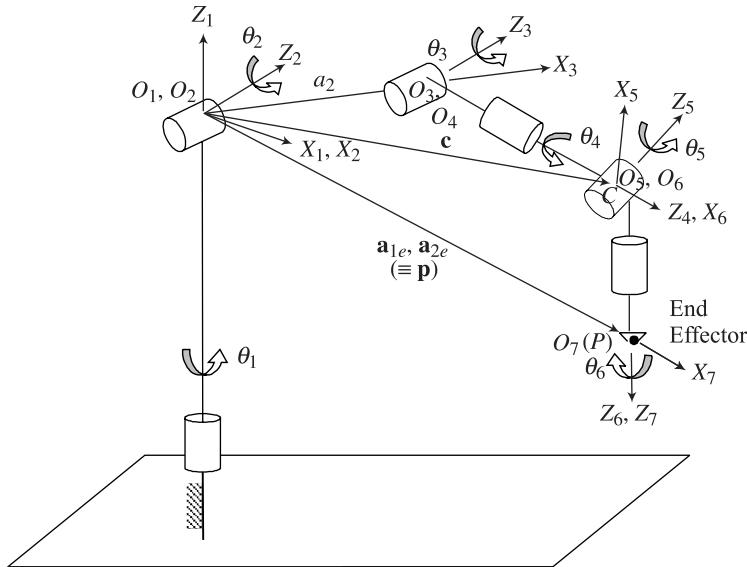
$$\mathbf{T} = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 c_2 s_3 - s_1 c_3 & -c_1 s_2 & 0 \\ s_1 c_2 c_3 + c_1 s_3 & -s_1 c_2 s_3 + c_1 c_3 & -s_1 s_2 & 0 \\ s_2 c_3 & -s_2 s_3 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.12b)$$

Example 6.9 Forward Kinematics of an Anthropomorphic Articulated Robot

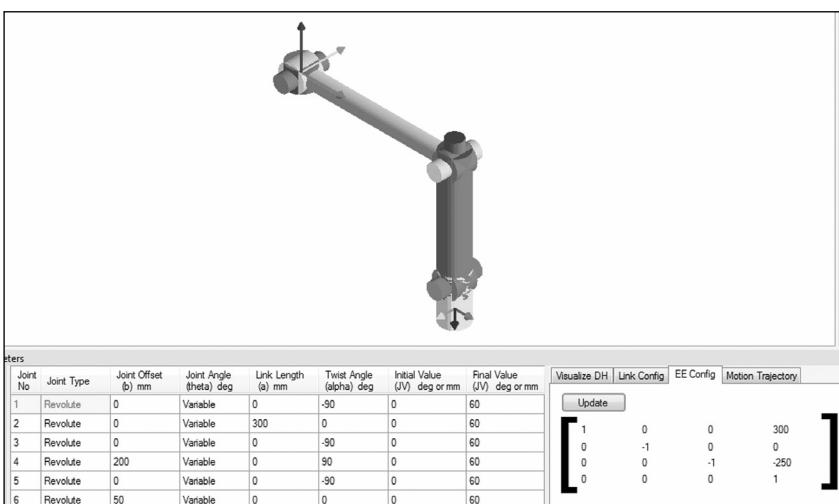
An anthropomorphic articulated robot is shown in Fig. 6.5, whose DH parameters are listed in Table 6.1. It is basically the arm of Fig. 5.37 with the wrist of Fig. 5.38 attached to the arm. It has six degrees of freedom (DOF). The configuration of the

end-effector is obtained as, $\mathbf{T} = \mathbf{T}_1 \dots \mathbf{T}_6$. The final expressions of the orientation matrix \mathbf{Q} and the position vector of the end-effector \mathbf{p} , which are block components of the 4×4 matrix \mathbf{T} are given by

$$\mathbf{Q} = \begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix}, \text{ and } \mathbf{p} \equiv \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \quad (6.13a)$$



(a) Schematic diagram



(b) RoboAnalyzer screenshot

Fig. 6.5 Anthropomorphic articulated robot

where

$$\begin{aligned}
 q_{11} &\equiv c_1 [c_{23}(c_4 c_5 c_6 - s_4 s_6) - s_{23} s_5 c_6] + s_1 (s_4 c_5 c_6 + c_4 s_6) \\
 q_{12} &\equiv c_1 [-c_{23}(c_4 c_5 s_6 + s_4 s_6) + s_{23} s_5 s_6] - s_1 (s_4 c_5 s_6 - c_4 c_6) \\
 q_{13} &\equiv -c_1 (c_{23} c_4 s_5 + s_{23} c_5) - s_1 s_4 s_5 \\
 q_{21} &\equiv s_1 [c_{23}(c_4 c_5 c_6 - s_4 s_6) - s_{23} s_5 c_6] - c_1 (s_4 c_5 c_6 + c_4 s_6) \\
 q_{22} &\equiv s_1 [-c_{23}(c_4 c_5 s_6 + s_4 c_6) + s_{23} s_5 s_6] + c_1 (s_4 c_5 s_6 - c_4 c_6) \\
 q_{23} &\equiv -s_1 (c_{23} c_4 s_5 + s_{23} c_5) + c_1 s_4 s_5 \\
 q_{31} &\equiv -c_{23} s_5 c_6 + s_{23} (s_4 s_6 - c_4 c_5 c_6) \\
 q_{32} &\equiv c_{23} s_5 s_6 + s_{23} (s_4 c_6 + c_4 c_5 s_6) \\
 q_{33} &\equiv -c_{23} c_5 + s_{23} c_4 s_5
 \end{aligned} \tag{6.13b}$$

and

$$\begin{aligned}
 p_1 &= a_2 c_1 c_2 - b_4 c_1 s_{23} - b_6 [c_1 (c_{23} c_4 s_5 + s_{23} c_5) + s_1 s_4 s_5] \\
 p_2 &= a_2 s_1 c_2 - b_4 s_1 s_{23} - b_6 [s_1 (c_{23} c_4 s_5 + s_{23} c_5) - c_1 s_4 s_5] \\
 p_3 &= -a_2 s_2 - b_4 c_{23} + b_6 (s_{23} c_4 s_5 - c_{23} c_5)
 \end{aligned} \tag{6.13c}$$

Table 6.1 The DH parameters of the 6-DOF articulated robot

i	b_i	θ_i	a_i	α_i
1	0	θ_1 (JV)	0	$-\pi/2$
2	0	θ_2 (JV)	a_2	0
3	0	θ_3 (JV)	0	$-\pi/2$
4	b_4	θ_4 (JV)	0	$\pi/2$
5	0	θ_5 (JV)	0	$-\pi/2$
6	b_6	θ_6 (JV)	0	0

JV: Joint variable

Example 6.10 Forward Kinematics of a PUMA Robot

The architecture of a PUMA robot is shown in Fig. 6.6 where coordinate frames are assigned in Figs. 6.6(a-b) according to the definitions of DH and MDH-V parameters. These parameters are shown in Table 6.2. The forward dynamics of the PUMA robot for a given set of initial joint angles, namely, θ_i , for $i = 1, \dots, 6$, that are given inside [] of Table 6.2 is performed as follows:

$$\mathbf{T} \equiv \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 1 & a_2 + b_4 + b_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.14a}$$

where the individual HMT $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_6$ are given as

$$\begin{aligned}
 \mathbf{T}_1 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -a_2 \\ 0 & 0 & 1 & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \mathbf{T}_4 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & b_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned} \tag{6.14b}$$

Comparing the expressions of Eq. (6.14a) with the orientation and position of the end-effector frame $X_e - Y_e - Z_e$ of Fig. 6.6 (a or b), one can easily verify that the results are correct.

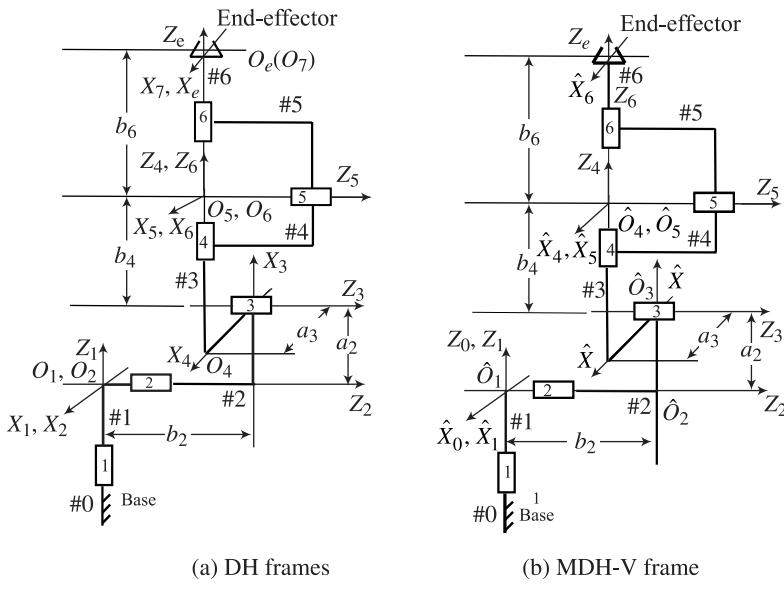


Fig. 6.6 A PUMA 560 robot

Table 6.2 DH and MDH-V parameters of PUMA 560 robot

i	a_{i-1}	α_{i-1}	b_i	θ_i	a_i	α_i
(a) DH parameters						
(b) MDH-V parameters						
1	0	0	0	$\theta_1(\text{JV})[0]$	0	$-\pi/2$
2	0	$-\pi/2$	b_2	$\theta_2(\text{JV})[-\pi/2]$	a_2	0
3	a_2	0	0	$\theta_3(\text{JV})[\pi/2]$	a_3	$\pi/2$
4	a_3	$\pi/2$	b_4	$\theta_4(\text{JV})[0]$	0	$-\pi/2$
5	0	$-\pi/2$	0	$\theta_5(\text{JV})[0]$	0	$\pi/2$
6	0	$\pi/2$	b_6	$\theta_6(\text{JV})[0]$	0	0

Example 6.11 Forward Kinematics of PUMA 560 using MDH-V Parameters

In order to test whether the parameters for the PUMA 560 robot using the MDH-V parameters explained in Section 5.5 are correct or not, the overall HMT relating the end-effector frame, i.e., $X_e - Y_e - Z_e$ with respect to the base-frame $\hat{X}_0 - \hat{Y}_0 - \hat{Z}_0$, as shown in Fig. 6.6(b), is obtained. For that, MDH-V parameters are obtained in Table 6.2(b). The result, $\hat{\mathbf{T}} = \hat{\mathbf{T}}_1 \cdots \hat{\mathbf{T}}_6$, is same as obtained in Eq. (6.14a), i.e., $\hat{\mathbf{T}} = \mathbf{T}$ in which $\hat{\mathbf{T}}_i$'s, for $i = 1, \dots, 6$, are given below using Eq. (5.67):

$$\begin{aligned}\hat{\mathbf{T}}_1 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \hat{\mathbf{T}}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \hat{\mathbf{T}}_3 = \begin{bmatrix} 0 & -1 & 0 & a_2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \hat{\mathbf{T}}_4 &= \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 0 & -1 & -b_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \hat{\mathbf{T}}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \hat{\mathbf{T}}_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -b_6 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned}\quad (6.15)$$

Example 6.12 Forward Kinematics of KUKA KR-5 Robot

Another robot similar to PUMA 560, namely, KUKA KR-5, is analyzed for its forward kinematics. The DH parameters are shown in Table 6.3. The corresponding HTM of the end-effector of the KUKA robot in its initial configuration, i.e., when all joint angles are zeros, is given in Eq. (6.16)

$$\mathbf{T} \equiv \begin{bmatrix} 1 & 0 & 0 & 900 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -335 \\ 0 & 0 & 0 & 1 \end{bmatrix}\quad (6.16)$$

Table 6.3 DH parameters of KUKA KR-5

i	b_i	θ_i	a_i	α_i
1	400	θ_1 (JV) [0]	180	$\pi/2$
2	135	θ_2 (JV) [0]	600	π
3	135	θ_3 (JV) [0]	120	$-\pi/2$
4	620	θ_4 (JV) [0]	0	$\pi/2$
5	0	θ_5 (JV) [0]	0	$-\pi/2$
6	115	θ_6 (JV) [0]	0	0

In order to verify the results, RoboAnalyzer software was used whose screenshot is shown in Fig. 6.7.

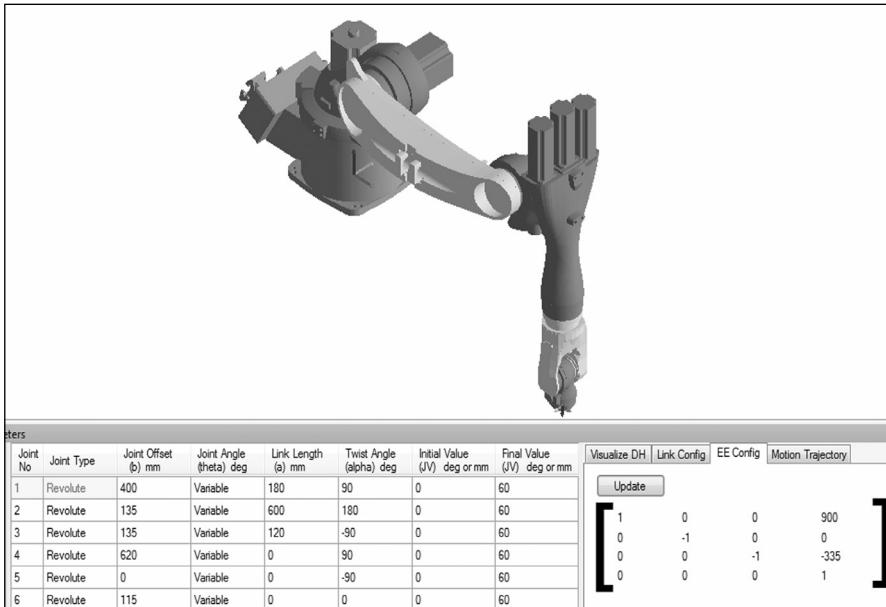


Fig. 6.7 RoboAnalyzer screenshot of KUKA KR-5 robot

Example 6.13 Forward Kinematics of the Stanford Arm

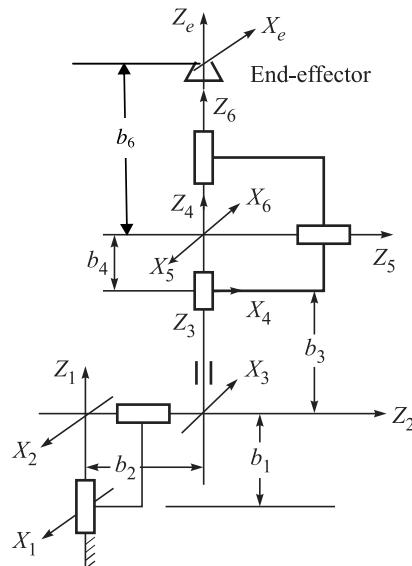
The architecture of the Stanford PUMA robot is shown in Fig. 6.8 whose DH parameters are given in Table 6.4. Similar to the PUMA and KUKA KR-5 robots, the overall HMT \mathbf{T} is obtained below:

$$\mathbf{T} \equiv \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & b_2 \\ 0 & 0 & 1 & b_1 + b_3 + b_4 + b_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.17)$$

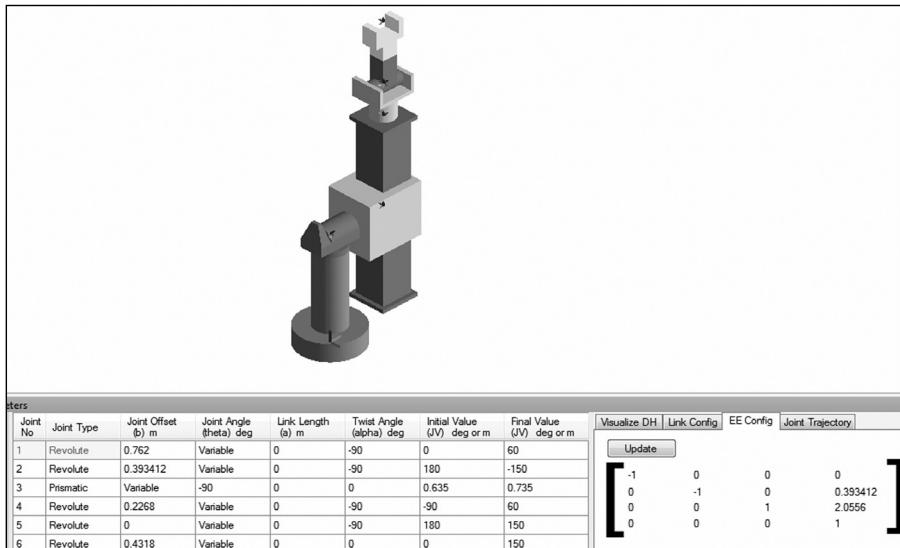
Table 6.4 DH parameters of the Stanford Arm

i	b_i	θ_i	a_i	α_i
1	b_1	θ_1 (JV) [0]	0	$-\pi/2$
2	b_2	θ_2 (JV) [π]	0	$-\pi/2$
3	b_3 (JV)	$-\pi/2$	0	0
4	b_4	θ_4 (JV) [- $\pi/2$]	0	$-\pi/2$
5	0	θ_5 (JV) [π]	0	$-\pi/2$
6	b_6	θ_6 (JV) [0]	0	0

Note that the arrangements of the DH frames in Fig. 6.8(a). Comparison of the HTM of Eq. (6.17) due to the initial joint values for the Stanford Arm given in Table 6.4 with the configuration of the end-effector shown in Fig. 6.8(a) shows that the result is correct. The same can be verified using RoboAnalyzer software also whose screenshot is shown in Fig. 6.8(b).



(a) Schematic diagram



(b) RoboAnalyzer screenshot

Fig. 6.8 The Stanford Robot

6.2 INVERSE POSITION ANALYSIS

The forward or direct kinematic equations, either in the form of Eq. (6.1) or Eqs. (6.2–6.3), established the functional relationship between the joint variables and the end-effector's orientation and position. The inverse kinematics problem consists of the determination of the joint variables corresponding to a given end-effector's orientation and position. The solution to this problem is of fundamental importance in order to transform the motion specifications assigned to the end-effector in the operational space into the corresponding joint space motions. As regards the forward kinematics, Eq. (6.1), the end-effector rotation matrix and position are computed in a unique manner when the joint variables are known. On the other hand, the inverse kinematics problem is much more complex for the following reasons:

- The algebraic equations to be solved are, in general, nonlinear in the joint variables, and thus it is not always possible to find a closed-form solution.
- Multiple solutions may exist.
- Infinite solutions may also exist, e.g., in the case of kinematically redundant robot manipulators.
- There might be no admissible solution in view of the manipulator architecture.

One approach to the inverse kinematics problem is to find a closed-form solution using algebra or geometry. Another approach is to find a numerical solution by some successive-approximation algorithm. Although the former approach is generally more desirable in applying the solution to real-time control of robots, it is not always possible to obtain the closed-form solutions for the manipulators with arbitrary architectures. Rather, the class of manipulators for which the closed-form solutions are guaranteed is very limited. Notice, however, that most of the manipulators in industrial use belong to this class. The algebraic approach to closed-form solution means to find the joint angles through algebraic transformations of Eqs. (6.1–6.3). The geometric approach means to find the joint angles using geometrical heuristics to take advantage of the special structure of the manipulators. It is sometimes advantageous to use both approaches together to solve a problem. Since it is difficult to find a general solution for a manipulator with arbitrary architecture, the inverse kinematic solutions for position are presented with respect to specific robot architectures, as explained next.

6.2.1 A Three-link Planar Arm

Consider the arm shown in Fig. 6.4 whose forward position kinematics was solved in Example 6.4. Here, it is desired to find the joint angles θ_1 , θ_2 and θ_3 corresponding to a given end-effector's position and orientation. For planar motion, the position and orientation of the end-effector can be specified by the origin of frame 4, i.e., $(p_x \ p_y)$, and the orientation of the frame attached to the end-effector with respect to the X_1 -axis, i.e., the angle φ , as shown in Fig. 6.4. Hence, they are specified as the input. The solutions are then obtained using two different approaches, as explained next.

1. Algebraic Solution First, the algebraic solution technique is illustrated. From the forward position kinematics analysis of the planar three-link arm, as done in Example 6.4,

$$\varphi = \theta_1 + \theta_2 + \theta_3 \quad (6.18a)$$

$$p_x = a_1 c_1 + a_2 c_{12} + a_3 c_{123} \quad (6.18b)$$

$$p_y = a_1 s_1 + a_2 s_{12} + a_3 s_{123} \quad (6.18c)$$

Note that these equations are nonlinear in joint angles, θ_1 , θ_2 , and θ_3 . Moreover, the inverse kinematic problem is simplified by subdividing the task as to position the wrist, W of Fig. 6.4(a), before the end-effector is oriented. The coordinates of W are w_x and w_y . This is equivalent to finding the inverse kinematic solution of the two-link planar arm shown in Fig. 6.2 whose end-effector point is (w_x, w_y) instead of (p_x, p_y) . The position relations of Eqs. (6.18b-c) are now recast as

$$w_x = p_x - a_3 c \varphi = a_1 c_1 + a_2 c_{12} \quad (6.19a)$$

$$w_y = p_y - a_3 s \varphi = a_1 s_1 + a_2 s_{12} \quad (6.19b)$$

Squaring the two sides of Eqs. (6.19a-b) and adding, one obtains

$$w_x^2 + w_y^2 = a_1^2 + a_2^2 + 2 a_1 a_2 c_2 \quad (6.20a)$$

which yields

$$c_2 = \frac{w_x^2 + w_y^2 - a_1^2 - a_2^2}{2 a_1 a_2} \quad (6.20b)$$

The existence of a solution for Eq. (6.20b) imposes the condition, $-1 \leq c_2 \leq 1$. Otherwise, the given point (w_x, w_y) is outside the reachable workspace of the arm. Then, set

$$s_2 = \pm \sqrt{1 - c_2^2} \quad (6.20c)$$

where the positive sign is relative to the elbow-up posture and the negative sign is to the elbow-down posture. Hence, the angle θ_2 is computed as

$$\theta_2 = \text{atan2}(s_2, c_2) \quad (6.21)$$

where ‘atan2’ in contrast to ‘atan’ is a function in any computer language like MATLAB, C, FORTAN, etc., which computes the value of “ $\tan^{-1}(.)$ ” in appropriate quadrant. If a point lies in the first and third quadrants, the function ‘atan’ gives the answer in both the cases the one that corresponds to the first quadrant only. This is so because the arguments, y/x and $-y/(-x)$, result in the same numerical value. While ‘atan2’ uses the argument (y, x) as input for y/x which provides the result corresponding to the first quadrant, the input argument $(-y, -x)$ for $-y/-x$ gives the result that lies in the third quadrant. Having determined θ_2 , the angle θ_1 is found by expanding c_{12} and s_{12} of Eq. (6.19) and rearranging them as

$$w_x = (a_1 + a_2 c_2)c_1 - a_2 s_1 s_2 \quad (6.22a)$$

$$w_y = (a_1 + a_2 c_2)s_1 + a_2 c_1 s_2 \quad (6.22b)$$

In order to evaluate θ_1 , Eq. (6.22a) is multiplied by $a_2 s_2$ and Eq. (6.22b) by $(a_1 + a_2 c_2)$, followed by the subtraction of the former from the latter. This yields in the value of s_1 as

$$s_1 = \frac{(a_1 + a_2 c_2)w_y - a_2 s_2 w_x}{\Delta} \quad (6.23a)$$

where $\Delta \equiv a_1^2 + a_2^2 + 2 a_1 a_2 c_2 = w_x^2 + w_y^2$. Similarly, c_1 is obtained as

$$c_1 = \frac{(a_1 + a_2 c_2)w_x + a_2 s_2 w_y}{\Delta} \quad (6.23b)$$

Following the analogy to Eq. (6.21), the solution for θ_1 is now obtained below

$$\theta_1 = \text{atan2}(s_1, c_1) \quad (6.23c)$$

Finally, the angle θ_3 is found from the expression of Eq. (6.18a) as

$$\theta_3 = \varphi - \theta_1 - \theta_2 \quad (6.24)$$

2. Geometric Solution A geometric solution technique is presented here. As above, the orientation angle is given in Eq. (6.18a), and the coordinates of the origin

Why Geometric?

Geometric solution has better pictorial understanding. Hence, it is useful to study.

of frame 3 in Fig. 6.4(a) are computed from Eqs. (6.18b–c). The application of the cosine theorem to the angle formed by the links, a_1 , a_2 , and the segment connecting points O_1 and W , Fig. 6.4(a), gives

$$w_x^2 + w_y^2 = a_1^2 + a_2^2 - 2 a_1 a_2 \cos(\pi - \theta_2) \quad (6.25)$$

The two admissible configurations of the triangle are shown in Fig. 6.4(a). Observing $\cos(\pi - \theta_2) = -\cos \theta_2 \equiv -c_2$, one immediately obtains Eq. (6.20a), whereas the existence of the triangle guarantees the following:

$$\sqrt{w_x^2 + w_y^2} \leq a_1 + a_2$$

This condition is not satisfied when the given point W is outside the arm's reachable workspace. However, based on the assumption of admissible solutions, the angle θ_2 is obtained as

$$\theta_2 = \cos^{-1}(c_2) \quad (6.26)$$

where the elbow-up posture is obtained when θ_2 is in between $-\pi$ and 0, and the elbow-down posture is obtained for θ_2 between 0 and π . To find θ_1 , consider the angles α and β in Fig. 6.4(a), which are computed from

$$\alpha = \text{atan2}(w_y, w_x) \text{ and } \sqrt{w_x^2 + w_y^2} \cos \beta = a_1 + a_2 c_2 \quad (6.27a)$$

where the determination of the angles, α and β , depends on the sign of w_x and w_y . Moreover, substituting the expression of c_2 , Eq. (6.20b), into Eq. (6.27a), yields the angle β , i.e.,

$$\beta = \cos^{-1} \frac{w_x^2 + w_y^2 + a_1^2 - a_2^2}{2 a_1 \sqrt{w_x^2 + w_y^2}} \quad (6.27b)$$

In Eq. (6.27b), β should be within 0 and π so as to preserve the existence of the triangle. Then,

$$\theta_1 = \alpha \pm \beta \quad (6.28)$$

in which the positive sign holds for $\theta_2 \in (-\pi, 0)$, and the negative sign for $\theta_2 \in (0, \pi)$. Finally, θ_3 is computed from Eq. (6.24).

Example 6.14 Inverse Kinematics of Two-link Planar Arm

For a two-link planar arm shown in Fig. 6.2 with unit link lengths, i.e., $a_1 = a_2 = 1$ unit, consider the input configuration of the end-effector given by the homogeneous transformation matrix (HTM) as follows:

$$\mathbf{T} \equiv \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 1 + \frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.29a)$$

Note that in Eq. (6.29a) only the first two elements of the last column will be used as inputs (p_x, p_y) to solve the inverse kinematics problem, i.e., to find θ_1 and θ_2 . This is because the arm has two degrees of freedom (DOF) and only variables, say, p_x and p_y , can be controlled using two input variables, i.e., θ_1 and θ_2 . Using Eqs. (6.20b-6.23c), where w_x and w_y are actually $p_x = 1 + \sqrt{3}/2$ and $p_y = 1/2$, the two inverse kinematics solutions are obtained as

$$\theta_1 = 0^\circ, \theta_2 = 30^\circ, \text{ and } \theta_1 = 30^\circ, \theta_2 = -30^\circ \quad (6.29b)$$

Using RoboAnalyzer's "ikin" command, the inverse kinematics solutions can also be obtained, which are shown in Fig. 6.9.

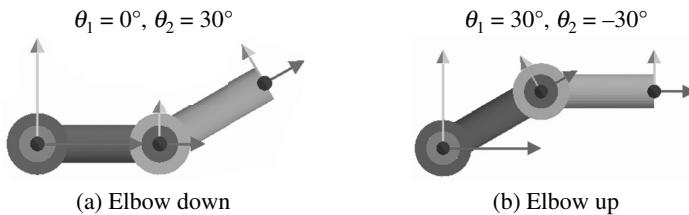


Fig. 6.9 Inverse kinematics solution of two-link planar arm

Example 6.15 Inverse Kinematics of Three-link Planar Arm

Referring to Fig. 6.4, the input homogeneous matrix, \mathbf{T} of Eq. (6.7), is given as

$$\mathbf{T} \equiv \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 & \sqrt{3} + \frac{5}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} + 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.30a)$$

where $\varphi = 60^\circ$, and the nonzero constant DH parameters of Table 5.3 are taken as, $a_1 = a_2 = 2$ units, and $a_3 = 1$ unit. Using Eqs. (6.20b-c), c_2 and s_2 are calculated as, $c_2 = 0.866$, and $s_2 = 0.5$, which yields, $\theta_2 = 30^\circ$. Next, s_1 and c_1 are evaluated from Eqs. (6.23a-b), as $s_1 = 0$, and $c_1 = 1$. The value of the joint angle θ_1 is obtained as $\theta_1 = 0^\circ$. Finally, θ_3 is obtained from Eq. (6.24) as $\theta_3 = 30^\circ$. The results are summarized below:

$$\theta_1 = 0^\circ, \theta_2 = 30^\circ, \text{ and } \theta_3 = 30^\circ \quad (6.30b)$$

Note that the positive value of s_2 was used in evaluating $\theta_2 = 30^\circ$. The use of negative value would result in the following set:

$$\theta_1 = 30^\circ, \theta_2 = -30^\circ, \text{ and } \theta_3 = 60^\circ \quad (6.30c)$$

The configurations corresponding to Eqs. (6.30b-c) were also verified using RoboAnalyzer's "ikin" command. They are shown in Fig. 6.10. Note that constant orientation of link 3 which carries the end-effector.

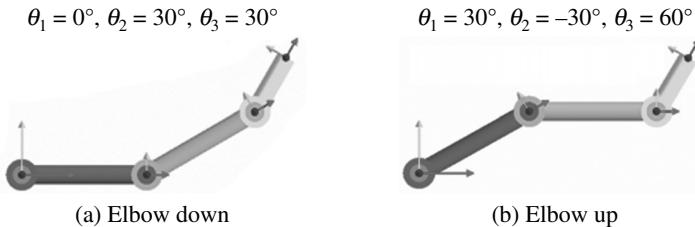


Fig. 6.10 Inverse kinematics solution of a three-link planar arm

Example 6.16 Inverse Kinematics of the Three-link Planar Arm Using MATLAB

In order to solve Example 6.15, a MATLAB program can be written, as shown in Fig. 6.11, which can be stored in a file, say, 'ch6ikin3.m' that can be run to yield the above results.

```
%Non-zero constant DH parameters
a1=2; a2=2;a3=1;

%Input
phi=pi/3; px=2.5+sqrt(3); py=1+sqrt(3)/2;

%Intermediate calculations
wx=px-a3*cos(phi); wy=py-a3*sin(phi); del=wx*wx+wy*wy;

%Calculations for theta_2
c2=(del-a1*a1-a2*a2)/(2*a1*a2); s2=sqrt(1-c2*c2);
th21=atan2(s2,c2); th22=atan2(-s2,c2);

%Calculation for finding theta_1
s11=((a1+a2*cos(th21))*wy-a2*sin(th21)*wx)/del; c11=((a1+a2*cos(th21))
)*wx+a2*sin(th21)*wy)/del;
s12=((a1+a2*cos(th22))*wy-a2*sin(th22)*wx)/del; c12=((a1+a2*cos(th22))
)*wx+a2*sin(th22)*wy)/del;
th11=atan2(s11,c11); th12=atan2(s12,c12);

%Calculation for theta_3
th31=phi-th11-th21; th32=phi-th12-th22;

%Angles in degree
r2d=180/pi;
th11d=th11*r2d, th12d=th12*r2d, th21d=th21*r2d, th22d=th22*r2d,
th31d=th31*r2d, th32d=th32*r2d
```

Fig. 6.11 Texts in file 'ch6ikin3.m' for inverse kinematics of a three-link arm

6.2.2 Inverse Kinematics of Revolute-Prismatic (RP) Planar Arm

Referring to Fig. 5.30(b) and Eq. (6.6b), it is obvious that

$$p_x = a_1 c_1 - b_2 s_1 \quad (6.31a)$$

$$p_y = a_1 s_1 + b_2 c_1 \quad (6.31b)$$

Multiplication of Eq. (6.31a) with c_1 and Eq. (6.31b) with s_1 , and addition of the resultant equations yield the following:

$$p_x c_1 + p_y s_1 = a_1 \quad (6.32)$$

Now, the following identities are used to convert Eq. (6.32) into a second order polynomial:

$$s_1 = \frac{2z_1}{1+z_1^2} \text{ and } c_1 = \frac{1-z_1^2}{1+z_1^2}, \text{ where } z_1 \equiv \tan \frac{\theta_1}{2} \quad (6.33)$$

The resultant second order polynomial is given by

$$Az_1^2 + Bz_1 + C = 0, \text{ where } A \equiv a_1 + p_x; B \equiv -2p_y; C \equiv a_1 - p_x \quad (6.34)$$

Two solutions of Eq. (6.34) then immediately follow as

$$z_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \quad (6.35)$$

Accordingly, the two joint-angle solutions, namely, $\theta_1^{(i)}$, for $i = 1, 2$, are evaluated as

$$\theta_1^{(i)} = 2 \tan^{-1}(z_i), \text{ for } i = 1, 2 \quad (6.36a)$$

Substituting Eq. (6.36a) into Eqs. (6.31a-b), one immediately finds the solution for b_2 by multiplying Eq. (6.31a) with s_1 and Eq. (6.31b) with c_1 , followed by subtracting the former from the latter one, i.e.,

$$b_2^{(i)} = p_y c_1^{(i)} - p_x s_1^{(i)} \quad (6.36b)$$

Equation (6.36) is the required inverse kinematic solutions for the RP planar arm at hand. Note that this arm has also two solutions for a given end-effector position.

Example 6.17 Inverse Kinematics of Revolute-Prismatic (RP) Planar Arm

For a two-link planar arm shown in Fig. 5.30(b) with $a_1 = 1$ unit, consider the input configuration of the end-effector given by the homogeneous transformation matrix (HTM) as follows:

$$\mathbf{T} \equiv \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & \frac{\sqrt{3}-1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & \frac{\sqrt{3}+1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.37a)$$

Note that similar to the two revolute jointed planar arms, Example 6.14, only the first two elements of the last column will be required to solve the inverse kinematics problem, i.e., to find θ_1 and b_2 . Using Eq. (6.36), two sets of solutions are obtained as follows:

$$\theta_1 = 30^\circ, b_2 = 1.0, \text{ and } \theta_1 = 120^\circ, b_2 = -1.0 \quad (6.37b)$$

The above solutions can be visualized using RoboAnalyzer's "FKin" command, as shown in Fig. 6.12. This is an alternate way to verify the inverse kinematics results in RoboAnalyzer if a model of the same configuration does not exist in "IKin" database. Note also in Fig. 6.12(b) that link 2 is out of the assembly from link 1, which is not practically possible. Hence, one needs to discard Solution 2 during the actual control of such arm.

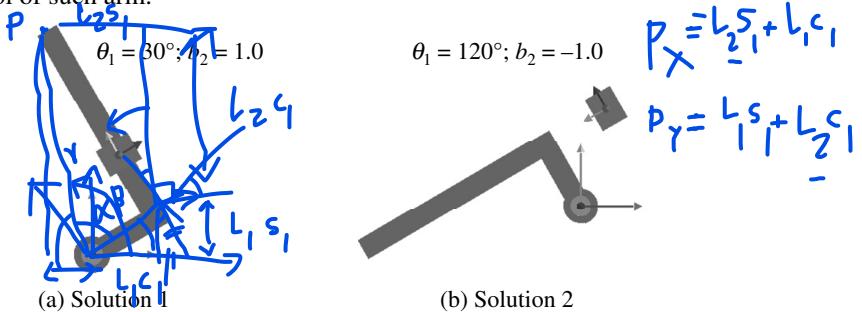


Fig. 6.12 Inverse kinematics solutions of RP arm

6.2.3 An Articulated Arm

Consider the articulated arm shown in Fig. 6.13. It is desired to find the joint variables, θ_1 , θ_2 , and θ_3 , corresponding to a given end-effector position, \mathbf{p}_w . Note that the kinematic relation for \mathbf{p}_w is expressed by the 4th column of matrix \mathbf{T} in Eq. (6.11b) from which

$$\theta_1 = \text{atan}2(p_y, p_x) \quad (6.38a)$$

where inputs p_x and p_y are equal to the (1,4) and (2,4) elements of the matrix \mathbf{T} , i.e.,

$$p_x = c_1(a_2c_2 + a_3c_{23}),$$

$$\text{and } p_y = s_1(a_2c_2 + a_3c_{23})$$

Observe that another admissible solution for θ_1 is

$$\theta_1 = \pi + \text{atan}2(p_y, p_x) \quad (6.38b)$$

when θ_2 is equal to $\pi - \theta_2^{(1)}$, where $\theta_2^{(1)}$ is one of the solutions corresponding to θ_1 given by Eq. (6.38a), as indicated in Fig. 6.13.

Once θ_1 is known, the remaining architecture is planar with regard to the variables θ_2 and θ_3 . Hence, exploiting the solution for the wrist of the three-link planar arm in Section 6.2.1, one gets

$$\theta_3 = \text{atan}2(s_3, c_3) \quad (6.39)$$

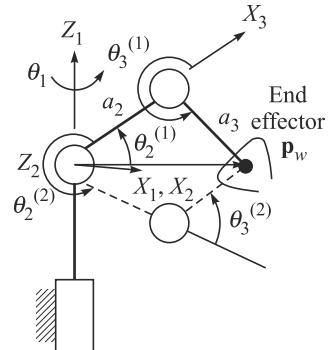


Fig. 6.13 Two admissible solutions

in which

$$c_3 \equiv \frac{p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2}{2a_2a_3}; \quad s_3 \equiv \pm \sqrt{1 - c_3^2}$$

and

$$\theta_2 = \text{atan}2(s_2, c_2) \quad (6.40)$$

In Eq. (6.40), the arguments, s_2 and c_2 , are evaluated as

$$s_2 \equiv \frac{(a_2 + a_3c_3)p_z - a_3s_3\sqrt{p_x^2 + p_y^2}}{\Delta}; \quad c_2 \equiv \frac{(a_2 + a_3c_3)\sqrt{p_x^2 + p_y^2} + a_3s_3p_z}{\Delta}$$

The denominator Δ being equal to $p_x^2 + p_y^2 + p_z^2$. The inverse problem of this arm admits four solutions. Two of them are depicted in Fig. 6.13, whereas one of the other two is the mirror image of the figure about Z_1 -axis, followed by rotation about Z_2 -axis to reach the point P_W . The fourth solution is the mirror image of the last two links about X_1 from its latest configuration. For a robot arm of more generic nature where the link offset $b_2 \neq 0$, it can also be recognized that there exists four solutions, as depicted in Fig. 6.14. According to the values of θ_1 , θ_2 and θ_3 , they are referred as

- Shoulder-Front/Elbow-Up (SFEU), and Shoulder-Front/Elbow-Down (SFED), as shown by the solid lines; and
- Shoulder-Back/Elbow-Up (SBEU) and Shoulder-Back/Elbow-Down (SBED), as shown by the dotted lines.

Note that, for shoulder-front configuration, elbow-up and elbow-down configurations are shown in Fig. 6.14(b). Obviously, the forearm orientation is different for the two pairs of solutions. Notice also that it is possible to find the solutions only if $p_x \neq 0$ and $p_y \neq 0$. In case $p_x = p_y = 0$, Eq. (6.38a) does not yield any θ_1 . Moreover, an infinity of solutions is obtained, and it is possible to determine the joint variables, θ_2 and θ_3 , independent of the value of θ_1 . In the following, it will be seen that the arm in such configuration is kinematically *singular*, more specifically, it is called *shoulder singularity*.

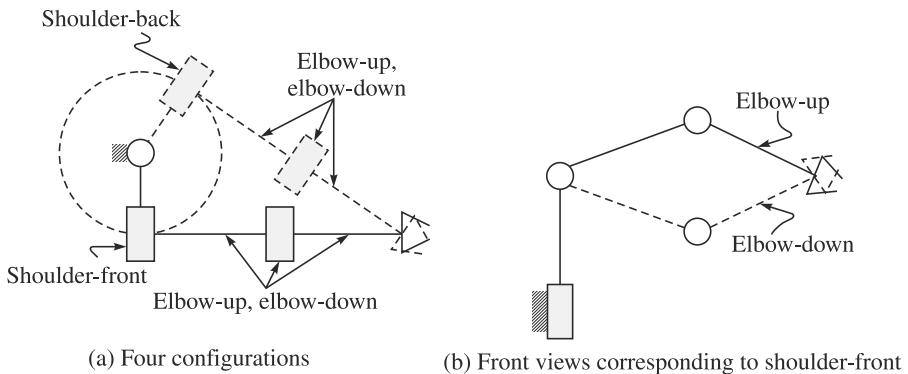


Fig. 6.14 Configurations of an articulated arm with nonzero offset ($b_2 \neq 0$)

Example 6.18 Inverse Kinematics of the Articulated Arm

In order to find the four admissible solutions of the arm shown in Fig. 6.13, the following numerical values are considered: $a_2 = a_3 = 1$ unit; and $p_x = 1$ units, $p_y = p_z = 0$ unit. The solutions are then evaluated as

1. $\theta_1^{(1)} = 0^\circ$; $\theta_2^{(1)} = 60^\circ$; $\theta_3^{(1)} = -120^\circ$; SFEU configuration
2. $\theta_1^{(1)} = 0^\circ$; $\theta_2^{(2)} = -60^\circ$; $\theta_3^{(2)} = 120^\circ$; SFED configuration
3. $\theta_1^{(1)} = 180^\circ$; $\theta_2^{(1)} = 240^\circ$; $\theta_3^{(1)} = -120^\circ$; SBEU configuration
4. $\theta_1^{(1)} = 180^\circ$; $\theta_2^{(2)} = 120^\circ$; $\theta_3^{(2)} = 120^\circ$; SBED configuration

Note that the values of p_x , p_y , and p_z are taken in such a manner that the above four solutions can be easily verified using Fig. 6.13. Besides, “IKin” command of RoboAnalyzer can be used to verify the results whose output is shown in Fig. 6.15, where “Solution 1” corresponds to (1) above, “Solution 2” and “Solution 3” geometrically correspond to (3) and (4), respectively, and “Solution 4” is same as (2) above. Note that in the input parameters of the articulated arm in Fig. 6.15 “Link Length” of 1 has some small value, namely, 0.0001 mm, even though in the inverse kinematics analysis given in Eqs. (6.38–6.40), it was considered zero. This was due to general implementation of a 3-DOF articulated arm based on the derivations reported in Bahuguna (2012). Without a small value, no inverse kinematics result can be found as one of the associated expressions become zero in the denominator resulting in “NaN” value by the computer.

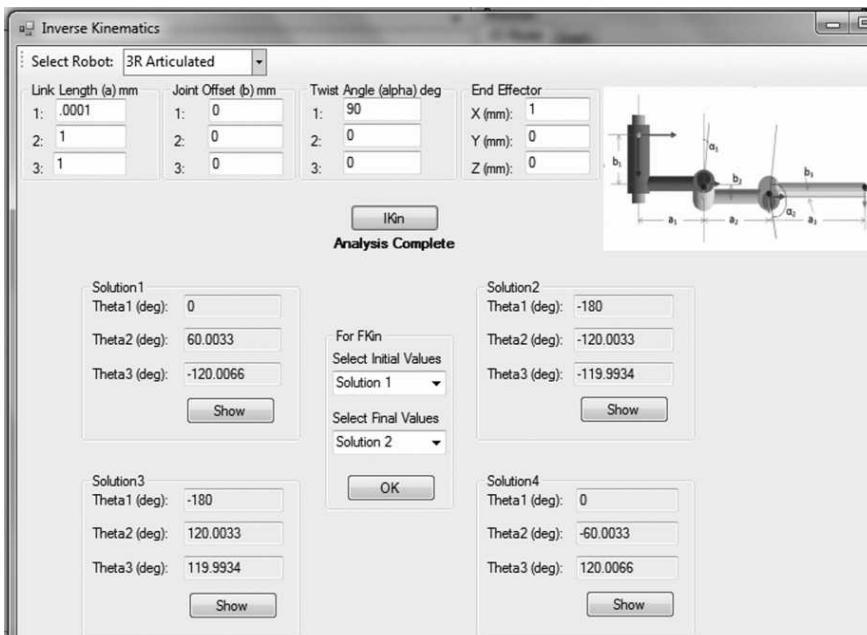


Fig. 6.15 Inverse kinematics of articulated arm in RoboAnalyzer

Example 6.19 Inverse Kinematics of the Articulated Arm Using MATLAB

In order to solve Example 6.18, a MATLAB program can be written, as shown in Fig. 6.16, which can be stored in a file, say, ‘ch6ikin3aa.m’ that can be run to yield the above results. Moreover, by changing the values of “a2” “a3,” and “px,” “py,” “pz” that correspond to the robot’s DH parameters and the end-effector position, respectively, many other solutions can be generated.

```
%Non-zero constant DH parameters
a2=1; a3=1;

%Input
px=1; py=0; pz=0;

%Intermediate calculation
delxy = px*px+py*py; del=delxy+pz*pz;

%Calculations for theta_1
th11=atan2(py,px);
th12=pi+atan2(py,px);

%Calculations for theta_3
c3=(del-a2*a2-a3*a3)/(2*a2*a3); s3=sqrt(1-c3*c3);
th31=atan2(s3,c3); th32=atan2(-s3,c3);

%Calculation for finding theta_2
s21=(-(a2+a3*cos(th31))*pz-a3*s3*delxy)/del; c21=((a2+a3*cos(th31))
*delxy+a3*s3*pz)/del;
s22 = ( - ( a 2 + a 3 * c o s ( t h 3 1 ) ) * p z + a 3 * s 3 * d e l x y ) / d e l ;
c22=(a2+a3*cos(th31))*delxy-a3*s3*pz)/del;
th21=atan2(s21,c21); th22=atan2(s22,c22);
th23=pi-th21; th24=pi-th22;

%Angles in degree
r2d=180/pi;
th11d=th11*r2d, th12d=th12*r2d, th21d=th21*r2d, th22d=th22*r2d,
th23d = th23*r2d, th24d = th24*r2d, th31d=th31*r2d, th32d=th32*r2d
```

Fig. 6.16 Texts of file ‘ch6ikin3aa.m’ for inverse kinematics of articulated arm

6.2.4 A Wrist

Consider the wrist architecture shown in Figs. 5.38 and 6.16, whose kinematic relations are given in Eq. (6.12b). It is desired to find the joint variable, θ_1 , θ_2 and θ_3 , corresponding to a given end-effector orientation, \mathbf{Q} , with the following form:

$$\mathbf{Q} \equiv \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \quad (6.41)$$

where q_{ij} , for $i, j = 1, \dots, 3$, are the elements of the 3×3 matrix \mathbf{Q} of the homogeneous transformation matrix \mathbf{T} , Eq. (6.12b). It is now possible to compute the solutions of the joint angles directly as

$$\theta_1 = \text{atan2}(q_{23}, q_{13}) \quad (6.42a)$$

$$\theta_2 = \text{atan2}(\sqrt{q_{13}^2 + q_{23}^2}, q_{33}) \quad (6.42b)$$

$$\theta_3 = \text{atan2}(-q_{32}, q_{31}) \quad (6.42c)$$

for θ_2 between 0 and π . Note that if each joint is allowed to rotate 360°, there are two possible solutions for the last three joint displacements. Indeed, since Eqs. (6.42a–c) involve the arctangent function, the angle for θ_1 may have two values, which are 180° apart. The two configurations corresponding to the two solutions are illustrated in Fig. 6.17.

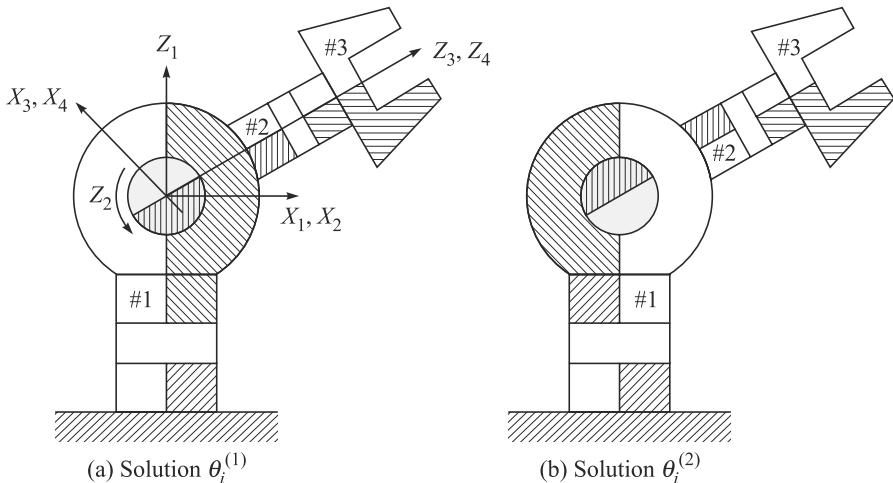


Fig. 6.17 Two solutions of the wrist [Courtesy: Asada and Slotine (1986)]

Let $\theta_i^{(1)}$ and $\theta_i^{(2)}$ be the first and second solutions with $\theta_i^{(1)} \leq \theta_i^{(2)}$. They are related by

$$\theta_1^{(2)} = \theta_1^{(1)} + \pi, \theta_2^{(2)} = -\theta_2^{(1)}, \text{ and } \theta_3^{(2)} = \theta_3^{(1)} + \pi$$

Hence, for θ_2 between $-\pi$ and 0,

$$\theta_1 = \text{atan2}(-q_{23}, -q_{13}) \quad (6.43a)$$

$$\theta_2 = \text{atan2}(-\sqrt{q_{13}^2 + q_{23}^2}, q_{33}) \quad (6.43b)$$

$$\theta_3 = \text{atan2}(-q_{32}, q_{31}) \quad (6.43c)$$

It is noted here that the joint angles should always be found using the “atan2” values instead of “sin” or “cos” values due to the reason explained after Eq. (6.21). Moreover, note that a 6-DOF wrist-partitioned robot has a total of eight inverse kinematics solutions for the position analysis.

Example 6.20 Inverse Kinematics of the Wrist

In order to find the two solutions of the wrist shown in Fig. 6.17, the following numerical values for the orientation matrix \mathbf{Q} are considered:

$$\mathbf{Q} \equiv \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad (6.44)$$

which leads to the following two sets of results:

1. $\theta_1^{(1)} = 0^\circ; \theta_2^{(1)} = 90^\circ; \theta_3^{(1)} = 0^\circ;$
2. $\theta_1^{(1)} = 180^\circ; \theta_2^{(2)} = -90^\circ; \theta_3^{(2)} = 180^\circ$

The above values can easily be verified from the coordinate frames attached in Fig. 6.17.

6.2.5 A Wrist-partitioned Spatial Robot¹

Commercial robots used in the industries are mostly serial in nature whose wrists are equipped with three intersecting joint axes. Hence, their inverse kinematics problem can be separated from the same needed for the position of the arm portion of the robot, namely, the center of intersection of the wrist axes. Hence, such robots are called *wrist-partitioned (WP)* or *decoupled robots*, as shown in Fig. 6.5(a). In such cases, the DH parameters associated to the joint axes 4-6 satisfy $a_4 = a_5 = b_5 = 0$, and thus, the origins of frames 5 and 6 are coincident. All other DH parameters can assume arbitrary values. Inverse kinematics a 6-axis serial WP robot manipulator can be obtained based on the knowledge gained from the solutions of the positioning and orienting problems explained in Sections 6.2.3 and 6.2.4. The detailed general derivations are given next, which are adopted from Angeles (2003) and implemented in the RoboAnalyzer software by Bahuguna (2012).

1. Position Problem Let C be the point of intersection of the last three axes, namely, 4, 5, and 6 of the spherical wrist. If \mathbf{c} denotes the position vector of two coincident points, namely, O_5 and O_6 , which is nothing but the wrist point shown in Fig. 6.5(a), it can be seen that its expression do not contain the joint angles θ_4 , θ_5 and θ_6 , but only the first three joints angles, namely, θ_1 , θ_2 and θ_3 . Vector \mathbf{c} can be expressed from Fig. 6.5(a) as

$$\mathbf{a}_1 + \mathbf{Q}_1 \mathbf{a}_2 + \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{a}_3 + \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3 \mathbf{a}_4 = \mathbf{c} \quad (6.45a)$$

Both sides of Eq. (6.45a) are pre-multiplied by $\mathbf{Q}_1^{-1} = \mathbf{Q}_1^T$ to yield the following:

$$\mathbf{a}_2 + \mathbf{Q}_2 \mathbf{a}_3 + \mathbf{Q}_2 \mathbf{Q}_3 \mathbf{a}_4 = \mathbf{Q}_1^T (\mathbf{c} - \mathbf{a}_1) \quad (6.45b)$$

where the 3×3 orientation matrix \mathbf{Q}_i represents the orientation of the frame $i+1$ with respect to (w.r.t) frame i , and the 3-dimensional position vector \mathbf{a}_i represents the origin of the frame $i+1$, i.e., O_{i+1} , w.r.t. that of the frame i , i.e., O_i . They are extracted from the representation of the homogenous transformation matrix (HTM) given in Eq. (5.61b) as

$$\mathbf{Q}_i \equiv \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i \\ 0 & S\alpha_i & C\alpha_i \end{bmatrix} \text{ and } \mathbf{a}_i \equiv \begin{bmatrix} a_i C\theta_i \\ a_i S\theta \\ b_i \end{bmatrix} \quad (6.45c)$$

¹ This section is based on the methodology explained in Angeles (2003). It is included here for the completeness of a general methodology for the inverse kinematics of industrial robots that has been implemented in the “IKin” module of RoboAnalyzer software developed at IIT Delhi.

In Eq. (6.45b), the vector \mathbf{c} should be input to the inverse kinematics problem, which needs to be calculated from the given end-effector's position \mathbf{p} or \mathbf{a}_{1e} , and its orientation, i.e., matrix \mathbf{Q} . Moreover, the vectors \mathbf{a}_i 's are represented as $\mathbf{a}_i = \mathbf{Q}_i \mathbf{b}_i$ where $\mathbf{b}_i \equiv [a_i \ b_i S\alpha_i \ b_i C\alpha_i]^T$. Hence, Eq. (6.45b) is re-written as

$$\mathbf{Q}_2(\mathbf{b}_2 + \mathbf{Q}_3 \mathbf{b}_3 + \mathbf{Q}_3 \mathbf{Q}_4 \mathbf{b}_4) = \mathbf{Q}_1^T \mathbf{c} - \mathbf{b}_1 \quad (6.45d)$$

For the WP or decoupled robot of Fig. 6.5(a), the following holds true:

$$\mathbf{a}_4 \equiv \mathbf{Q}_4 \mathbf{b}_4 = \begin{bmatrix} 0 \\ 0 \\ b_4 \end{bmatrix} = b_4 \mathbf{e}, \text{ where } \mathbf{e} \equiv [0 \ 0 \ 1]^T \quad (6.45e)$$

The term b_4 is constant, whereas \mathbf{e} is a unit vector. The product $\mathbf{Q}_3 \mathbf{Q}_4 \mathbf{b}_4$ is then computed as

$$\mathbf{Q}_3 \mathbf{Q}_4 \mathbf{b}_4 = \mathbf{b}_4 \mathbf{Q}_3 \mathbf{e} \equiv b_4 \mathbf{u}_3, \text{ where } \mathbf{u}_3 \equiv \mathbf{Q}_3 \mathbf{e} \quad (6.45f)$$

Upon substitution of Eq. (6.45f) into Eq. (6.45d), one gets

$$\mathbf{Q}_2(\mathbf{b}_2 + \mathbf{Q}_3 \mathbf{b}_3 + b_4 \mathbf{u}_3) = \mathbf{Q}_1^T \mathbf{c} - \mathbf{b}_1 \quad (6.45g)$$

where the input vector \mathbf{c} can be represented as

$$\mathbf{c} = \mathbf{p} - \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3 \mathbf{Q}_4 \mathbf{a}_5 - \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3 \mathbf{Q}_4 \mathbf{Q}_5 \mathbf{a}_6 \quad (6.45h)$$

Since for the WP robot, $a_5 = b_5 = 0$, from Fig. 6.5(a) it can be seen that vector \mathbf{a}_5 vanishes because the same is represented in its own frame as $[\mathbf{a}_5]_5 \equiv [a_5 C\theta_5 \ a_5 S\theta_5 \ b_5]^T$, which leads to the following:

$$\mathbf{c} = \mathbf{p} - \mathbf{Q} \mathbf{Q}_6^T \mathbf{a}_6 \equiv \mathbf{p} - \mathbf{Q} \mathbf{b}_6, \text{ where } \mathbf{Q} \equiv \mathbf{Q}_1 \dots \mathbf{Q}_6 \quad (6.45i)$$

Next, let the points O_7 and O_5 or O_6 be represented in the fixed first frame 1 as

$$[\mathbf{p}]_1 \equiv \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{and} \quad [\mathbf{c}]_1 \equiv \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad (6.45j)$$

Expansion of Eq. (6.45i) is then compared with Eq. (6.45a) to provide the following:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} x - (q_{11}a_6 + q_{12}b_6 S\alpha_6 + q_{13}b_6 C\alpha_6) \\ y - (q_{21}a_6 + q_{22}b_6 S\alpha_6 + q_{23}b_6 C\alpha_6) \\ z - (q_{31}a_6 + q_{32}b_6 S\alpha_6 + q_{33}b_6 C\alpha_6) \end{bmatrix} \quad (6.45k)$$

where q_{ij} is the (i, j) entry of the input matrix \mathbf{Q} . The positioning problem is then nothing but to find the first three joint angles θ_1 , θ_2 and θ_3 required to position the end-effector's point O_5 or O_6 given by its coordinates x_c , y_c and z_c . The problem is now similar to the one explained in Section 6.2.3 with $b_1 \neq 0$. Note that Eq. (6.45g) represents three scalar equations in three unknowns. Let the Euclidean norm of the left-hand side of Eq. (6.45g) be l and that of its right-hand side be r . Then

$$l^2 \equiv a_2^2 + b_2^2 + a_3^2 + b_3^2 + b_4^2 + 2\mathbf{b}_2^T \mathbf{Q}_3 \mathbf{b}_3 + 2b_4 \mathbf{b}_2^T \mathbf{u}_3 + 2C\alpha_3 b_3 b_4 \quad (6.46a)$$

$$r^2 \equiv \|\mathbf{c}\|^2 + \|\mathbf{b}_1\|^2 - 2\mathbf{b}_1^T \mathbf{Q}_1^T \mathbf{c} \quad (6.46b)$$

In Eq. (6.46a), a_i and b_i , $i = 2, 3, 4$ are the DH parameters of the robot shown in Fig. 6.5(a), whereas Eqs. (6.46a-b) are rewritten as

$$Ac_1 + Bs_1 + Cc_3 + Ds_3 + E = 0 \quad (6.46c)$$

whose coefficients do not contain any unknown, i.e.,

$$A = 2a_1x_c; B = 2a_1y_c; C = 2a_2a_3 - 2b_2b_4S\alpha_2S\alpha_3; D = 2a_3b_2S\alpha_2 + 2a_2b_4S\alpha_3 \quad (6.46d)$$

$$E = a_2^2 + a_3^2 + b_2^2 + b_3^2 + b_4^2 - a_1^2 - x_c^2 - y_c^2 - (z_c - b_1)^2 \\ + 2b_2b_3C\alpha_2 + 2(b_2C\alpha_2 + b_3)b_4C\alpha_3 \quad (6.46e)$$

In addition, a third scalar equation of Eq. (6.45g) can be expressed as

$$Fc_1 + Gs_1 + Hc_3 + Is_3 + J = 0 \quad (6.46f)$$

where

$$F = y_cS\alpha_1; G = -x_cS\alpha_1; H = -b_4S\alpha_2S\alpha_3; I = a_3S\alpha_2; \quad (6.46g)$$

$$J = b_2 + b_3C\alpha_2 + b_4C\alpha_2C\alpha_3 - (z_c - b_1)C\alpha_1 \quad (6.46h)$$

Equations (6.46c) and (6.46f) define two nonlinear equations in θ_1 and θ_3 but linear in terms of c_1 , s_1 , c_3 and s_3 . Each of these equations thus defines a contour in the $\theta_1 - \theta_3$ plane. Their intersections will determine all real solutions to the problem at hand. For that c_i and s_i are substituted for their tangent half-angles, i.e., $\tan(\theta_i/2)$, for $i = 1, 3$. This leads to two bi-quadratic polynomials equations in terms of $\tan(\theta_1/2)$ and $\tan(\theta_3/2)$. One can then eliminate one of these variables to reduce to a single quadratic polynomial equation in the other variable, which is termed the *characteristic equation* of the positioning problem at hand (Angeles, 2003). The steps are shown by first solving c_1 and s_1 in terms of c_3 and s_3 , i.e.,

$$c_1 = \frac{-G(Cc_3 + Ds_3 + E) + B(Hc_3 + Is_3 + J)}{\Delta_1} \quad (6.47a)$$

$$s_1 = \frac{F(Cc_3 + Ds_3 + E) - A(Hc_3 + Is_3 + J)}{\Delta_1} \quad (6.47b)$$

where

$$\Delta_1 = AG - FB = -2a_1S\alpha_1(x_c^2 + y_c^2) \quad (6.47c)$$

Using the identity $c_1^2 + s_1^2 = 1$, a quadratic equation in c_3 and s_3 is then obtained as

$$Kc_3^2 + Ls_3^2 + Mc_3s_3 + Nc_3 + Ps_3 + Q = 0 \quad (6.48a)$$

whose coefficients are as follows:

$$K = 4a_1^2H^2 + S^2\alpha_1C^2; \quad L = 4a_1^2I^2 + S^2\alpha_1D^2; \quad M = 2(4a_1^2HI + S^2\alpha_1CD) \quad (6.48b)$$

$$N = 2(4a_1^2HJ + S^2\alpha_1CE); \quad P = 2(4a_1^2IJ + S^2\alpha_1DE) \quad (6.48c)$$

$$Q = 4a_1^2J^2 + S^2\alpha_1E^2 - 4a_1^2S^2\alpha_1\rho^2; \quad \rho^2 = x_c^2 + y_c^2 \quad (6.48d)$$

Now, two well-known trigonometric identities are introduced, namely,

$$c_3 \equiv \frac{1 - z_3^2}{1 + z_3^2} \text{ and } s_3 \equiv \frac{2z_3}{1 + z_3^2}, \text{ where } z_3 = \tan \frac{\theta_3}{2} \quad (6.49)$$

Upon substitution of the foregoing identities into Eq. (6.48a), a quartic equation in z_3 is obtained, namely,

$$Rz_3^4 + Sz_3^3 + Tz_3^2 + Uz_3 + V = 0 \quad (6.50a)$$

whose coefficients are computed from the input data as

$$R = 4a_1^2(J - H)^2 + S^2\alpha_1(E - C)^2 - 4a_1^2S^2\alpha_1\rho^2 \quad (6.50b)$$

$$S = 4[4a_1^2I(J - H) + S^2\alpha_1D(E - C)] \quad (6.50c)$$

$$T = 2[4a_1^2(J^2 - H^2 + 2I^2) + S^2\alpha_1(E^2 - C^2 + 2D^2) - 4a_1^2S^2\alpha_1\rho^2] \quad (6.50d)$$

$$U = 4[4a_1^2I(J + H) + S^2\alpha_1D(E + C)] \quad (6.50e)$$

$$V = 4a_1^2(J + H)^2 + S^2\alpha_1(E + C)^2 - 4a_1^2S^2\alpha_1\rho^2 \quad (6.50f)$$

Equation (6.50a) provides four solutions for θ_3 , which then can be used to obtain four different values of θ_1 using Eqs. (6.47a-b). For each set of θ_1 and θ_3 , it is then a simple matter to find the value of θ_2 using the first two scalar equations of Eq. (6.45g), which are displayed as

$$A_{11}c_2 + A_{12}s_2 = x_c c_1 + y_c s_1 - a_1 \quad (6.51a)$$

$$-A_{12}c_2 + A_{11}s_2 = -x_c s_1 C\alpha_1 + y_c C\alpha_1 c_1 + (z_c - b_1)S\alpha_1 \quad (6.51b)$$

where

$$A_{11} = a_2 + a_3c_3 + b_4s_3S\alpha_3; A_{12} \equiv -a_3C\alpha_2s_3 + b_3S\alpha_2 + b_4C\alpha_2S\alpha_3c_3 + b_4C\alpha_3S\alpha_2 \quad (6.51c)$$

In Eq. (6.51b), if A_{11} and A_{12} do not vanish simultaneously, angle θ_2 is available in terms of θ_1 and θ_3 as

$$c_2 = \frac{1}{\Delta_2} [A_{11}(x_c c_1 + y_c s_1 - a_1) - A_{12}(-x_c s_1 C\alpha_1 + y_c C\alpha_1 c_1 + (z_c - b_1)S\alpha_1)] \quad (6.52a)$$

$$s_2 = \frac{1}{\Delta_2} [A_{12}(x_c c_1 + y_c s_1 - a_1) + A_{11}(-x_c s_1 C\alpha_1 + y_c C\alpha_1 c_1 + (z_c - b_1)S\alpha_1)] \quad (6.52b)$$

where Δ_2 is defined as $\Delta_2 \equiv A_{11}^2 + A_{12}^2$, in which A_{11} and A_{12} are given in Eq. (6.51c).

2. Orientation Problem This orientation problem involved the determination of the wrist angles θ_4 , θ_5 and θ_6 that will produce a given orientation of the end-effector denoted with \mathbf{Q} . An orientation can be specified using any of the representations explained in Section 5.2.2. Irrespective of the type of orientation representation used, the input to the inverse kinematics orientation problem will be taken as the 3×3 rotation matrix \mathbf{Q} . Moreover, the positioning problem is assumed to be solved first. Hence, the joint angles θ_1 , θ_2 , and θ_3 , and accordingly, matrices \mathbf{Q}_1 , \mathbf{Q}_2 and \mathbf{Q}_3 , are known. Note that the matrix \mathbf{Q} represents the orientation of the end-effector. Hence, the components of vector \mathbf{e}_6 in the fixed frame is nothing but the 3rd column of \mathbf{Q} , i.e., $[\mathbf{e}_6]_1 = \mathbf{Q}[\mathbf{e}_6]_6$, where $[\mathbf{e}_6]_6 \equiv [0 \ 0 \ 1]$. For the purpose of analysis here, let \mathbf{e}_6 be known in frame 4 which is denoted as

$$[\mathbf{e}_6]_4 = [\xi \ \eta \ \zeta]^T \quad (6.53)$$

Moreover, the components of vector \mathbf{e}_5 in Frame 4 are nothing but the entries of the third column of matrix \mathbf{Q}_4 , i.e.,

$$[\mathbf{e}_5]_4 = [S\alpha_4 S_4 \quad -S\alpha_4 C_4 \quad C\alpha_4]^T \quad (6.54)$$

Furthermore, as per the definition of DH parameters, the joint axes Z_5 and Z_6 , i.e., vectors \mathbf{e}_5 and \mathbf{e}_6 , make an angle α_5 , as indicated in Fig. 6.18. Hence, from the definition of dot-product between two vectors, the following is obtained:

$$\mathbf{e}_6^T \mathbf{e}_5 = C\alpha_5 \quad (6.55a)$$

Upon substitution of Eqs. (6.53–6.54) into Eq. (6.55a) yields

$$\xi S\alpha_4 S_4 - \eta S\alpha_4 C_4 + \zeta C\alpha_4 = C\alpha_5 \quad (6.55b)$$

Equation (6.55b) is transformed using tan half-angle, $z_4 = \tan(\theta_4/2)$ as

$$(C\alpha_5 - \eta S\alpha_4 - \xi S\alpha_4)z_4^2 - 2\xi S\alpha_4 z_4 + (C\alpha_5 + \eta S\alpha_4 - \xi S\alpha_4) = 0 \quad (6.56a)$$

which is a quadratic equation in z_4 leading to two solutions, i.e.,

$$z_4 = \frac{\xi S\alpha_4 \pm \sqrt{(\xi^2 + \eta^2)S^2\alpha_4^2 - (C\alpha_5 - \zeta C\alpha_4)^2}}{C\alpha_5 - \zeta C\alpha_4 - \eta S\alpha_4} \quad (6.56b)$$

Once θ_4 is calculated as $\theta_4 = 2 \tan^{-1}(z_4)$ from the above two foregoing values of z_4 , angle θ_5 can be obtained uniquely for each value of θ_4 . The methodology is explained as follows: Note that for the 6-axis robot $\mathbf{Q} = \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3 \mathbf{Q}_4 \mathbf{Q}_5 \mathbf{Q}_6$, which is rewritten as

$$\mathbf{Q}_4 \mathbf{Q}_5 \mathbf{Q}_6 = \mathbf{R}, \text{ where } \mathbf{R} = \mathbf{Q}_3^T \mathbf{Q}_2^T \mathbf{Q}_1^T \mathbf{Q} \quad (6.57a)$$

whose entries in frame 4 be given as

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad (6.57b)$$

Expressions for θ_5 and θ_6 are next derived by solving first for \mathbf{Q}_5 from Eq. (6.57a) as

$$\mathbf{Q}_5 = \mathbf{Q}_4^T \mathbf{R} \mathbf{Q}_6^T \quad (6.58a)$$

Now, from the expression of the matrix \mathbf{Q}_i of Eq. (6.45c), it is clear that the third row of \mathbf{Q}_i does not contain θ_i . Hence, the third column of the matrix product given by Eq. (6.58a) is independent of θ_6 . Hence, two equations for θ_5 are obtained by equating the top two elements of the third column of that equation, i.e.,

$$S\alpha_5 s_5 = (S\alpha_6 r_{12} + C\alpha_6 r_{13})c_4 + (S\alpha_6 r_{22} + C\alpha_6 r_{23})s_4 \quad (6.58b)$$

$$-S\alpha_5 c_5 = -C\alpha_4 (S\alpha_6 r_{12} + C\alpha_6 r_{13})s_4 + C\alpha_4 (S\alpha_6 r_{22} + C\alpha_6 r_{23})c_4 + S\alpha_4 (S\alpha_6 r_{32} + C\alpha_6 r_{33}) \quad (6.58c)$$

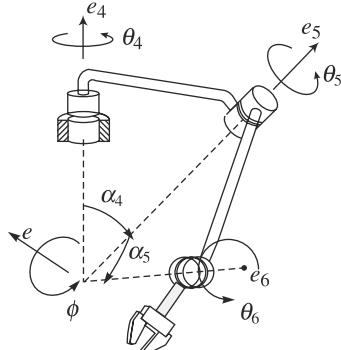


Fig. 6.18 Architecture of a spherical wrist [Courtesy: Angeles (2003)]

Equations (6.58b-c) yield a unique solution for θ_5 for every value of θ_4 . With the angles θ_4 and θ_5 known, it is a simple matter to calculate θ_6 , which is done by obtaining an expression for \mathbf{Q}_6 from Eq. (6.57a) as

$$\mathbf{Q}_6 = \mathbf{Q}_5^T \mathbf{Q}_4^T \mathbf{R} \quad (6.59a)$$

Now rewriting \mathbf{Q}_i matrix column wise

$$\mathbf{Q}_i = [\mathbf{q}_{1i} \quad \mathbf{q}_{2i} \quad \mathbf{q}_{3i}] \quad (6.59b)$$

Vector \mathbf{q}_{1i} defined in Eq. (6.59a) can now be rearranged for $i = 6$ in the form given below:

$$\mathbf{q}_{1,6} = \mathbf{Q}_5^T \mathbf{Q}_4^T \mathbf{r}_1 \quad (6.59c)$$

where \mathbf{r}_1 is the first column of \mathbf{R} . Let \mathbf{w} denotes the product $\mathbf{Q}_4^T \mathbf{r}_1$, i.e.,

$$\mathbf{w} \equiv \mathbf{Q}_4^T \mathbf{r}_1 = \begin{bmatrix} r_{11}c_4 + r_{21}s_4 \\ -C\alpha_4(r_{11}s_4 - r_{21}c_4) + S\alpha_4r_{31} \\ S\alpha_4(r_{11}s_4 - r_{21}c_4) + C\alpha_4r_{31} \end{bmatrix} \quad (6.59d)$$

Now substituting the value of $\mathbf{Q}_4^T \mathbf{r}_1$ from Eq. (6.59d) into Eq. (6.59c), one obtains

$$\mathbf{Q}_5^T \mathbf{Q}_4^T \mathbf{r}_1 = \begin{bmatrix} w_1c_5 + w_2s_5 \\ C\alpha_5(-w_1s_5 + w_2c_5) + S\alpha_5w_3 \\ S\alpha_5(w_1s_5 - w_2c_5) + C\alpha_5w_3 \end{bmatrix} \quad (6.59e)$$

in which w_i denotes the i^{th} component of \mathbf{w} . Hence, c_6 and s_6 are determined from the first two scalar equations of Eq. (6.59c), i.e.,

$$c_6 = w_1c_5 + w_2s_5 \quad (6.60a)$$

$$s_6 = C\alpha_5(-w_1s_5 + w_2c_5) + S\alpha_5w_3 \quad (6.60b)$$

Equations (6.60a-b) provide a unique value of θ_6 for every pair of values (θ_4 , θ_5). Note that two values of θ_4 have been determined, and for each value, one set of θ_5 and θ_6 has been obtained. Therefore, there exist two sets of solutions for the orientation problem which correspond to two wrist postures, as indicated in Fig. 6.19 for the inverse kinematic solution of the 3-DOF wrist. When combined with the four configurations of the positioning problem, a total of eight solutions are possible for a WP or decoupled robot, as also pointed out at the end of Section 6.2.4.

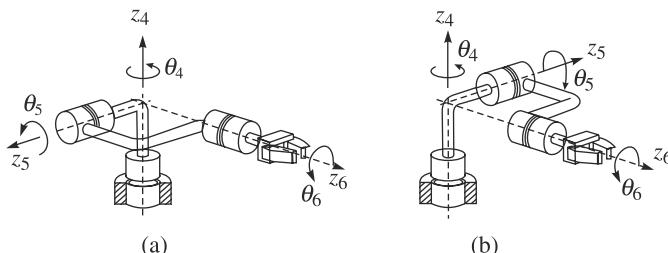


Fig. 6.19 Two configurations of the spherical wrist [Courtesy: Angeles (2003)]

Example 6.21 Inverse Kinematics of KUKA KR-5 using RoboAnalyzer

For the KUKA KR-5 whose DH parameters, shown in Table 6.3, are analysed for its inverse kinematics using RoboAnalyzer software. The end-effector configuration, denoted with the HTM \mathbf{T} was taken as follows:

$$\mathbf{T} = \begin{bmatrix} -0.8086 & 0 & 0.5883 & 80 \\ 0 & 1 & 0 & 100 \\ -0.5883 & 0 & -0.8086 & 1200 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.61)$$

The corresponding eight solutions are listed in Table 6.5 whose eight sets of joint angles can be easily verified by inserting them in RoboAnalyzer's forward kinematics module by clicking "FKin" button. The "EE Config" immediately shows the input matrix \mathbf{T} shown in Eq. (6.61).

Table 6.5 Inverse kinematics solutions of KUKA KR-5 robot

Solution /Angle	1 (deg)	2 (deg)	3 (deg)	4 (deg)	5 (deg)	6 (deg)	7 (deg)	8 (deg)
θ_1	82.96	82.96	-97.04	-97.04	-97.04	-97.04	82.96	82.96
θ_2	50.39	50.39	65.66	65.66	149.25	149.25	139.75	139.75
θ_3	-165.65	-165.65	-160.13	-160.13	2.04	2.04	7.56	7.56
θ_4	234.44	-305.56	137.17	-42.83	57.13	-122.87	317.96	-222.04
θ_5	134.13	-134.13	120.81	-120.81	135.96	-135.96	119.32	-119.32
θ_6	125.56	-54.44	-124.08	55.92	129.37	-50.64	-122.51	57.49

6.2.6 Comments on the Inverse Kinematics Problem

It has been observed in the previous sections that the inverse kinematics of several planar and spatial robotic manipulators are based on the kinematic equations that relate the joint variables with the end-effector's configuration. These equations are transcendental in nature, which are generally converted to polynomials using tangent half-angles. A typical equation which is linear in $\sin \theta$ and $\cos \theta$ will get transformed into a quadratic equation in z if $z = \tan(\theta/2)$ is used. For example, see Eqs. (6.33–6.34). For multiple degrees-of-freedom, the intention is to eliminate one variable at a time from the kinematic equations in order to convert to higher-order polynomials in terms of the other joint variables. There is, however, no general methodology to decide which variable is to be eliminated first, and so on, as it is very much dependent on the architectures of a robot manipulator and the types of joints used, e.g., revolute or prismatic. With the presence of a prismatic joint, the problem of inverse kinematics simplifies, as the joint variable does not contain "sine" or "cosine" terms.

Another important aspect which simplifies the inverse kinematics problem is the decoupling of wrist from the arm, as pointed out in Section 6.2.5, which is mostly seen in almost all industrial robots including PUMA-560, KUKA KR-5 and others. If the last three axes did not intersect or no special feature is noticed, the inverse kinematic solutions are more complex. For example, Raghavan and Roth (1993) have

shown that an arbitrary six-revolute-jointed robot manipulator leads to 16th degree polynomial for its inverse kinematics solutions. It is not necessary that all roots of the polynomial need to be real. In case of complex solutions, the robot cannot reach those solutions. This is very difficult to predict as no close-form solution exists for the 16th degree polynomial. Hence, one needs to resort to numerical techniques. These challenges make them still an active area of research.

6.3 VELOCITY ANALYSIS: THE JACOBIAN MATRIX

In Section 6.2, relationships between the joint variables and the end-effector orientation and position were presented. In this section, the relationships between the joint rates and the corresponding end-effector's angular and linear velocities are presented. This mapping or the relationship is described by a matrix called *Jacobian*, which depends on the robot's configuration. The Jacobian constitutes one of the most important tools for robot characterization. In fact, it is useful for

- finding singular configurations,
- analyzing redundancy,
- determining inverse kinematics algorithms for velocity analysis,
- describing the relationship between the forces applied at the end-effector and the resulting torques at the joints, and
- deriving dynamics algorithms.

Jacobian
It is the term used in mathematics, which for the m -dimensional vector function $\mathbf{f}(\mathbf{x})$ is defined as, $\mathbf{J} \equiv \partial \mathbf{f} / \partial \mathbf{x}$, where \mathbf{J} is called the Jacobian matrix of size $m \times n$ for the n -dimensional independent variable \mathbf{x} .

Note that the position kinematics equations given by Eqs. (6.1)–(6.3) have both the rotation matrix \mathbf{Q} , and the position vector \mathbf{p} or \mathbf{a}_{1e} of Fig. 6.5(a) which are functions of the joint variables, namely, $\boldsymbol{\theta} \equiv [\theta_1, \dots, \theta_n]^T$. Both the end-effector's orientation and position vary as $\boldsymbol{\theta}$ varies. The goal here is to express the end-effector's angular velocity, $\boldsymbol{\omega}_e$ and linear velocity or simply the velocity \mathbf{v}_e ($\equiv \dot{\mathbf{p}}$ or $\dot{\mathbf{a}}_{1e}$) as a function of the joint velocities $\dot{\boldsymbol{\theta}}$ as

$$\boldsymbol{\omega}_e = \mathbf{J}_\omega \dot{\boldsymbol{\theta}} \quad (6.62a)$$

$$\mathbf{v}_e = \mathbf{J}_v \dot{\boldsymbol{\theta}} \quad (6.62b)$$

where \mathbf{J}_ω and \mathbf{J}_v are the $3 \times n$ matrices relating the contribution of the joint velocities or rates $\dot{\boldsymbol{\theta}}$ to the end-effector angular velocity $\boldsymbol{\omega}_e$ and velocity \mathbf{v}_e respectively. Matrices \mathbf{J}_ω and \mathbf{J}_v are also the functions of the joint variables $\boldsymbol{\theta}$. Equations (6.62a-b) can be written in compact form as

$$\mathbf{t}_e = \mathbf{J} \dot{\boldsymbol{\theta}}, \text{ where } \mathbf{J} \equiv \begin{bmatrix} \mathbf{J}_\omega \\ \mathbf{J}_v \end{bmatrix} \quad (6.62c)$$

and $\mathbf{t}_e \equiv [\boldsymbol{\omega}_e^T \ \mathbf{v}_e^T]^T$ is the 6-dimensional vector, which is referred here as the “twist” of the end-effector (Angeles, 2003), where the $6 \times n$ Jacobian matrix \mathbf{J} is a function of the joint variables $\boldsymbol{\theta}$ only. In Eqs. (6.62a-c), the velocity \mathbf{v}_e is specified as the time derivative of the end-effector position \mathbf{a}_{1e} , i.e., $\mathbf{v}_e \equiv \dot{\mathbf{a}}_{1e}$, whereas nothing was

mentioned about the angular velocity $\boldsymbol{\omega}_e$, and orientation representation matrix, \mathbf{Q} . This is derived as follows:

1. Consider a time-varying rotation matrix \mathbf{Q} . Since the matrix \mathbf{Q} is orthogonal, it has the following property:

$$\mathbf{Q} \mathbf{Q}^T = \mathbf{Q}^T \mathbf{Q} = \mathbf{I} \quad (6.63a)$$

where \mathbf{I} is the 3×3 identity matrix. The differentiation of Eq. (6.63a) with respect to time yields

$$\dot{\mathbf{Q}} \mathbf{Q}^T + \mathbf{Q} \dot{\mathbf{Q}}^T = \mathbf{0} \quad (6.63b)$$

2. Now introduce

$$\boldsymbol{\Omega} = \dot{\mathbf{Q}} \mathbf{Q}^T \quad (6.64a)$$

where $\boldsymbol{\Omega}$ is the 3×3 “skew-symmetric” matrix due to Eq. (6.63b), i.e.,

$$\boldsymbol{\Omega} + \boldsymbol{\Omega}^T = \mathbf{0} \quad (6.64b)$$

3. Post-multiplying both sides of Eq. (6.64a) by \mathbf{Q} gives

$$\boldsymbol{\Omega} \mathbf{Q} = \dot{\mathbf{Q}} \quad (6.65)$$

that allows us to express the time derivative of the rotation matrix \mathbf{Q} , i.e., $\dot{\mathbf{Q}}$, as a function of $\boldsymbol{\Omega}$.

4. Now consider an arbitrary vector \mathbf{x} fixed to a moving body. Representations of the vector \mathbf{x} in the fixed frame F , and the moving frame attached to the body M , are $[\mathbf{x}]_F$ and $[\mathbf{x}]_M$, respectively. If \mathbf{Q} represents the orientation of frame M with respect to frame F then, $[\mathbf{x}]_F = \mathbf{Q} [\mathbf{x}]_M$, whose time derivative is given by

$$[\dot{\mathbf{x}}]_F = \dot{\mathbf{Q}} [\mathbf{x}]_M + \mathbf{Q} [\dot{\mathbf{x}}]_M \quad (6.66a)$$

Since vector $[\mathbf{x}]_M$ is the representation of vector \mathbf{x} which is fixed in the moving frame M , its expression in frame M does not change as the body moves. Hence, $[\dot{\mathbf{x}}]_M = 0$. As a result Eq. (6.66a) becomes

$$[\dot{\mathbf{x}}]_F = \dot{\mathbf{Q}} [\mathbf{x}]_M \quad (6.66b)$$

Using the expression of $\dot{\mathbf{Q}}$, i.e., Eq. (6.65), Eq. (6.66b) is rewritten as

$$[\dot{\mathbf{x}}]_F = \boldsymbol{\Omega} \mathbf{Q} [\mathbf{x}]_M \quad (6.66c)$$

Alternatively, if $\boldsymbol{\omega}$ denotes the angular velocity of the rigid body with which the frame M is attached then it is known from the fundamentals of mechanics (Ghosh and Mallik, 1998) that

$$[\dot{\mathbf{x}}]_F = [\boldsymbol{\omega} \times \mathbf{x}]_F, \text{ or } [\dot{\mathbf{x}}]_F = [\boldsymbol{\omega}]_F \times \mathbf{Q} [\mathbf{x}]_M \quad (6.67)$$

where ‘ \times ’ denotes the *cross-product* between two 3-dimentional Cartesian vectors. Comparing Eqs. (6.66c) and (6.67), it is obvious that the skew-symmetric matrix $\boldsymbol{\Omega}$ denotes the vector cross-product operator between the vector $\boldsymbol{\omega}$ and the vector $\mathbf{Q}[\mathbf{x}]_M$. The matrix $\boldsymbol{\Omega}$ is so that its symmetric elements with respect to the zero diagonal elements represent the components of the vector $\boldsymbol{\omega}$, i.e., $\boldsymbol{\omega} \equiv [\omega_x \ \omega_y \ \omega_z]^T$, in the form

$$\boldsymbol{\Omega} \equiv \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (6.68)$$

which justifies that $\boldsymbol{\Omega}$ is a function of $\boldsymbol{\omega}$.

6.3.1 Geometric Interpretation of Angular Velocity

One can also obtain the angular velocity of a coordinate frame given by $\boldsymbol{\omega}$ of Eq. (6.68), whose components are the elements of the skew-symmetric matrix $\boldsymbol{\Omega}$ geometrically. If $\mathbf{Q}(t)$ and $\mathbf{Q}(t + \Delta t)$ denote the orientation of the coordinate frame M with respect to the frame F at times t and $(t + \Delta t)$, respectively, the time derivative of \mathbf{Q} , denoted with $\dot{\mathbf{Q}}$, can be written from the knowledge of differential calculus as

$$\dot{\mathbf{Q}} = \lim_{\Delta t \rightarrow 0} \frac{\mathbf{Q}(t + \Delta t) - \mathbf{Q}(t)}{\Delta t} \quad (6.69)$$

where matrix $\mathbf{Q}(t + \Delta t)$ can be thought of as a composition of two successive rotations, namely, the rotation given by $\mathbf{Q}(t)$ followed by a small rotation represented, say, using the single-axis rotation presented in Chapter 5. If $\mathbf{Q}(\mathbf{e}, \Delta\theta)$ denotes the small rotation about the axis \mathbf{e} by a small angle $\Delta\theta$ and noting that both the rotations are denoted with respect to the fixed frame, the resulting expression can be obtained by pre-multiplication of $\mathbf{Q}(t)$ as,

$$\mathbf{Q}(t + \Delta\theta) = \mathbf{Q}(\mathbf{e}, \Delta\theta)\mathbf{Q}(t), \text{ where } \mathbf{Q}(\mathbf{e}, \Delta\theta) = \begin{bmatrix} 1 & -e_z \Delta\theta & e_y \Delta\theta \\ e_z \Delta\theta & 1 & -e_x \Delta\theta \\ -e_y \Delta\theta & e_x \Delta\theta & 1 \end{bmatrix} \quad (6.70)$$

Substituting Eq. (6.70) into Eq. (6.69) yields

$$\dot{\mathbf{Q}} = \lim_{\Delta t \rightarrow 0} \left[\frac{\mathbf{Q}(\mathbf{e}, \Delta\theta) - \mathbf{1}}{\Delta t} \right] \mathbf{Q}(t), \text{ where } \mathbf{Q}(\mathbf{e}, \Delta\theta) - \mathbf{1} = \begin{bmatrix} 0 & -e_z \Delta\theta & e_y \Delta\theta \\ e_z \Delta\theta & 0 & -e_x \Delta\theta \\ -e_y \Delta\theta & e_x \Delta\theta & 0 \end{bmatrix} \quad (6.71)$$

Note the symbol $\mathbf{1}$ in Eq. (6.71) denotes the 3×3 identity matrix. Now, dividing the matrix elements of $\mathbf{Q}(\mathbf{e}, \Delta\theta) - \mathbf{1}$ by Δt and taking the limit, one can write the following:

$$\dot{\mathbf{Q}} = \begin{bmatrix} 0 & -e_z \dot{\theta} & e_y \dot{\theta} \\ e_z \dot{\theta} & 0 & -e_x \dot{\theta} \\ -e_y \dot{\theta} & e_x \dot{\theta} & 0 \end{bmatrix} \mathbf{Q}(t), \text{ where } \dot{\theta} \equiv \lim_{\Delta t \rightarrow 0} \frac{\Delta\theta}{\Delta t} \quad (6.72)$$

The term $\dot{\theta}$ is nothing but the angular speed about the unit vector \mathbf{e} parallel to the axis of rotation. Comparing Eqs. (6.65), (6.68) and (6.72), one can then immediately observe that the angular velocity of the frame M with respect to F is given by

$$[\boldsymbol{\omega}]_F \equiv \dot{\theta} \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}^T \quad (6.73)$$

where e_x , e_y , and e_z are the components of the unit vector \mathbf{e} , which is parallel to the axis of rotation or the instantaneous axis of rotation.

6.3.2 Angular Velocity vs. Euler Angles

Since the Euler angles or the fixed-axes rotations are frequently used in the literature, it is desirable to have a relation between the angular velocity of a link or the coordinate frame attached to it and the time rates of the Euler angles used to represent a rotation.

Using the time derivatives of the composition of the matrix \mathbf{Q} in Eq. (5.34d), Eq. (6.64a) is written as

$$\begin{aligned}\Omega &= (\dot{\mathbf{Q}}_Z \mathbf{Q}_{Y'} \mathbf{Q}_{Z''} + \mathbf{Q}_Z \dot{\mathbf{Q}}_{Y'} \mathbf{Q}_{Z''} + \mathbf{Q}_Z \mathbf{Q}_{Y'} \dot{\mathbf{Q}}_{Z''}) \mathbf{Q}^T \\ &= (\dot{\mathbf{Q}}_Z + \mathbf{Q}_Z \dot{\mathbf{Q}}_{Y'} \mathbf{Q}_{Y'}^T + \mathbf{Q}_Z \mathbf{Q}_{Y'} \dot{\mathbf{Q}}_{Z''} \mathbf{Q}_{Z''}^T \mathbf{Q}_{Y'}^T) \mathbf{Q}^T\end{aligned}\quad (6.74)$$

Substituting the time derivatives of the elements of the matrices \mathbf{Q}_Z , $\mathbf{Q}_{Y'}$, and $\mathbf{Q}_{Z''}$, given by Eqs. (5.34a-c), and comparing Eq. (6.68) with Eq. (6.74), one can write the components of the angular velocity vector represented in the fixed-frame F , i.e., $[\dot{\boldsymbol{\omega}}]_F$, as a linear transformation of the three-dimensional array containing the time-rates of the ZYZ Euler angles, namely, ϕ , θ , and ψ , i.e.,

$$[\dot{\boldsymbol{\omega}}]_F = \mathbf{L}_{ZYZ} \dot{\boldsymbol{\theta}}_{ZYZ}, \text{ where } \mathbf{L}_{ZYZ} \equiv \begin{bmatrix} 0 & -S\phi & C\phi S\theta \\ 0 & C\phi & S\phi S\theta \\ 1 & 0 & C\theta \end{bmatrix} \text{ and } \dot{\boldsymbol{\theta}}_{ZYZ} \equiv \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (6.75)$$

In Eq. (6.75), $\dot{\boldsymbol{\theta}}_{ZYZ}$ denotes the three-dimensional array² of the ZYZ Euler angles, whereas \mathbf{L}_{ZYZ} is the 3×3 matrix. The derivation of Eq. (6.75) is quite useful as similar expressions can be obtained while Euler angles or fixed-axes rotations are used to denote the orientation of a rigid body with respect to another one.

Example 6.22 Rotation Matrix and Angular Velocity

Consider the elementary rotation matrix about the axis Z , given in Eq. (5.21). If the angle of rotation is α , which is a function of time then the rotation matrix \mathbf{Q} is as follows

$$\mathbf{Q} \equiv \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.76)$$

Its time derivative $\dot{\mathbf{Q}}$ is then given by

$$\dot{\mathbf{Q}} = \begin{bmatrix} -\dot{\alpha} \sin \alpha & -\dot{\alpha} \cos \alpha & 0 \\ \dot{\alpha} \cos \alpha & -\dot{\alpha} \sin \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (6.77a)$$

Hence, the cross-product matrix Ω associated with the angular velocity vector $\boldsymbol{\omega}$ is calculated as

$$\Omega \equiv \dot{\mathbf{Q}} \mathbf{Q}^T = \begin{bmatrix} 0 & -\dot{\alpha} & 0 \\ \dot{\alpha} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (6.77b)$$

Comparing Eqs.(6.68) and (6.77b), the angular velocity $\boldsymbol{\omega}$ is given by

$$\boldsymbol{\omega} \equiv [\omega_x, \omega_y, \omega_z]^T = [0 \ 0 \ \dot{\alpha}]^T$$

which is the angular velocity of the frame about the axis Z .

² It is not a vector in true sense as it does not satisfy all the properties of a vector space (Strang, 1980). However, such arrays are frequently referred as vectors for the purposes of mathematical operations.

Example 6.23 Euler Angles and Angular Velocity

If ZYZ Euler angles are $\varphi = 90^\circ$, $\theta = 0^\circ$ and $\psi = 0^\circ$, then

$$\mathbf{L}_{ZYZ} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad (6.78)$$

Using Eq. (6.78), the components of angular velocities in terms of the rates of the ZYZ Euler angles, i.e., $\dot{\varphi}, \dot{\theta}$, and $\dot{\psi}$ are $\omega_x = \dot{\theta}$, $\omega_y = 0$ and $\omega_z = \dot{\varphi} + \dot{\psi}$

6.4 LINK VELOCITIES

Consider a generic link i of a robot manipulator. According to the Denavit–Hartenberg (DH) convention presented in Section 5.4, link i connects joints i and $i + 1$, whereas link $i - 1$ connects the joints $i - 1$ and i . Note that frame i is attached to link $i - 1$ with its origin on the axis of the joint i , as shown in Fig. 6.20. Let \mathbf{o}_i and \mathbf{o}_{i-1} be the position vectors of the origins of the links i and $i - 1$, i.e., O_i and O_{i-1} , respectively. Let \mathbf{a}_{i-1} denotes the position of the origin of link i with respect to link $i - 1$. According to the vector summation rules, one can write

$$\mathbf{o}_i = \mathbf{o}_{i-1} + \mathbf{a}_{i-1} \quad (6.79a)$$

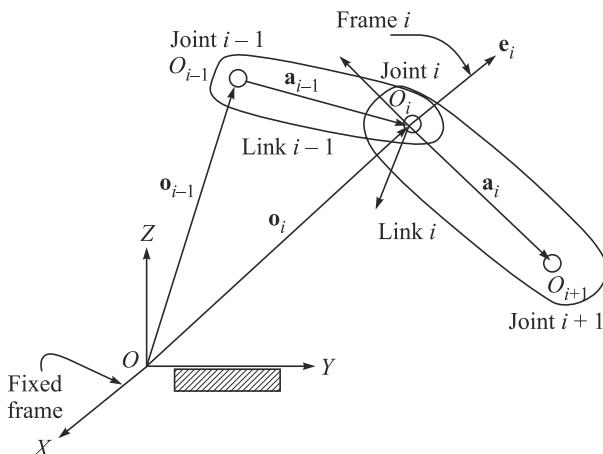


Fig. 6.20 Coupled links of a robot

Differentiating the above expression, velocity of the origin of link i , namely, O_i , is obtained as

$$\dot{\mathbf{o}}_i = \dot{\mathbf{o}}_{i-1} + \dot{\mathbf{a}}_{i-1} \quad (6.79b)$$

where $\dot{\mathbf{a}}_{i-1} = \boldsymbol{\omega}_{i-1} \times \mathbf{a}_{i-1}$, in which $\boldsymbol{\omega}_{i-1}$ is the angular velocity of link $i - 1$. Since \mathbf{o}_i and \mathbf{o}_{i-1} are also the position vectors of the points on links i and $i - 1$, respectively, Eq. (6.79b) gives the expression of linear velocity of link i as a function of the velocity and the angular velocity of link $i - 1$. For the angular velocity expression of link i , $\boldsymbol{\omega}_i$, it can be obtained as

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \boldsymbol{\omega}_{i,i-1} \quad (6.80)$$

where $\boldsymbol{\omega}_{i-1}$ is the angular velocity of link $i-1$, and $\boldsymbol{\omega}_{i,i-1}$ is the relative angular velocity of link i with respect to link $i-1$. Note that Eqs. (6.79a-b) attain different expressions depending on the type of the joint i , i.e., revolute or prismatic.

1. Revolute Joint Since the revolute joint provides angular motion to a link with respect to its previous one, the joint angle θ_i , as defined in Section 5.4, is variable. If \mathbf{e}_i denotes the unit vector parallel to the axis of the revolute joint then, $\boldsymbol{\omega}_{i,i-1} = \dot{\theta}_i \mathbf{e}_i$. Hence, $\boldsymbol{\omega}_i$ of Eq. (6.80) can be rewritten as

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i \quad (6.81a)$$

The corresponding velocity expression, Eq. (6.79b) is given by

$$\dot{\mathbf{o}}_i = \dot{\mathbf{o}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{a}_{i-1} \quad (6.81b)$$

2. Prismatic Joint A prismatic joint allows only relative translational motion, i.e., joint length b_i is variable, and there is no relative angular motion between the links $i-1$ and i . Hence,

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} \quad (6.82a)$$

and the velocity expression, Eq. (6.79b), is as follows:

$$\dot{\mathbf{o}}_i = \dot{\mathbf{o}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{a}_{i-1} + \dot{b}_i \mathbf{e}_i \quad (6.82b)$$

where \mathbf{e}_i is the unit vector parallel to the translational direction of link i with respect to link $i-1$, and $\dot{b}_i \mathbf{e}_i$ is the relative linear velocity.

6.5 JACOBIAN COMPUTATION

Here, the general expression for the Jacobian matrix, \mathbf{J} of Eq. (6.62c), will be derived. Since the end-effector is the n th body of the serial-chain, the angular velocity and velocity of the end-effector, $\boldsymbol{\omega}_e$ and \mathbf{v}_e , respectively, are obtained from the velocities of the n th link, namely, $\boldsymbol{\omega}_n$ and $\dot{\mathbf{o}}_n$, as

$$\boldsymbol{\omega}_e \equiv \boldsymbol{\omega}_n, \text{ and } \mathbf{v}_e = \dot{\mathbf{o}}_n + \boldsymbol{\omega}_n \times \mathbf{a}_{ne} \quad (6.83)$$

where \mathbf{a}_{ne} is the three-dimensional position vector of the end-effector with respect to the origin of the n th link, O_n . The velocities of the links are now computed starting from the fixed body, i.e., link 0, and following the recursive relations given in Eqs. (6.81a-b) or Eqs. (6.82a-b) depending on the type of joints used. For example, if all joints are revolute then

$$\boldsymbol{\omega}_0 = 0 \quad (6.84a)$$

$$\boldsymbol{\omega}_1 = \dot{\theta}_1 \mathbf{e}_1 \quad (6.84b)$$

⋮

$$\boldsymbol{\omega}_n = \dot{\theta}_1 \mathbf{e}_1 + \dot{\theta}_2 \mathbf{e}_2 + \cdots + \dot{\theta}_n \mathbf{e}_n \quad (6.84c)$$

and

$$\dot{\mathbf{o}}_1 = 0 \quad (6.85a)$$

$$\dot{\mathbf{o}}_2 = \dot{\mathbf{o}}_1 + \boldsymbol{\omega}_1 \times \mathbf{a}_1 = \dot{\theta}_1 \mathbf{e}_1 \times \mathbf{a}_2 \quad (6.85b)$$

\vdots

$$\dot{\mathbf{o}}_n = \dot{\theta}_1 \mathbf{e}_1 \times \mathbf{a}_{1,n} + \dot{\theta}_2 \mathbf{e}_2 \times \mathbf{a}_{2,n} + \cdots + \dot{\theta}_{n-1} \mathbf{e}_{n-1} \times \mathbf{a}_{n-1,n} \quad (6.85c)$$

where $\mathbf{a}_{i,j} \equiv \mathbf{a}_i + \mathbf{a}_{i+1} + \cdots + \mathbf{a}_{j-1}$ is the vector joining the origin of link i to the origin of link n , i.e., O_n . From Eqs. (6.84)-(6.85), the Jacobian matrix, \mathbf{J} of Eq. (6.62c), can be extracted as

$$\mathbf{J} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \\ \mathbf{e}_1 \times \mathbf{a}_{1e} & \mathbf{e}_2 \times \mathbf{a}_{2e} & \cdots & \mathbf{e}_n \times \mathbf{a}_{ne} \end{bmatrix} \quad (6.86)$$

in which $\mathbf{a}_{i,e} \equiv \mathbf{a}_{i,n} + \mathbf{a}_{n,e}$ is the vector shown in Fig. 6.5(a), for $i = 1, 2$, and \mathbf{J} is the $6 \times n$ matrix. From Eq. (6.86), the i th column of the matrix \mathbf{J} , denoted as \mathbf{j}_i , can be written as

$$\mathbf{j}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_i \times \mathbf{a}_{ie} \end{bmatrix}, \text{ if Joint } i \text{ is revolute} \quad (6.87a)$$

$$\text{and} \quad \mathbf{j}_i \equiv \begin{bmatrix} 0 \\ \mathbf{e}_i \end{bmatrix}, \text{ if Joint } i \text{ is prismatic} \quad (6.87b)$$

Example 6.24 Jacobian of the Two-link Revolute Jointed Arm

Since there are two revolute joints which can be used only for the positioning of the end-effector, the top block row of Eq. (6.86) that corresponds to the orientation of the end-effector is not relevant. Hence, the Jacobian of the two-link manipulator can be expressed as

$$\mathbf{J} = \begin{bmatrix} \mathbf{e}_1 \times \mathbf{a}_{1e} & \mathbf{e}_2 \times \mathbf{a}_{2e} \end{bmatrix} \quad (6.88a)$$

where, $\mathbf{e}_1 \equiv \mathbf{e}_2 \equiv [0 \ 0 \ 1]^T$. Moreover,

$$\mathbf{a}_{1e} \equiv \mathbf{a}_1 + \mathbf{a}_2 \equiv [a_1 c_1 + a_2 c_{12} \ a_1 s_1 + a_2 s_{12} \ 0]^T$$

$$\mathbf{a}_{2e} \equiv \mathbf{a}_2 \equiv [a_2 c_{12} \ a_2 s_{12} \ 0]^T$$

in which

$$s_1 \equiv \sin \theta_1; \ c_1 \equiv \cos \theta_1; \ s_{12} \equiv \sin(\theta_1 + \theta_2); \ c_{12} \equiv \cos(\theta_1 + \theta_2);$$

Furthermore, if the nonzero elements are extracted from Eq. (6.88a), the resulting Jacobian matrix is the 2×2 matrix given by

$$\mathbf{J} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \quad (6.88b)$$

Example 6.25 Jacobian of the Three-link Arm

Since there are three revolute joints, Eq. (6.86) gives

$$\mathbf{J} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ \mathbf{e}_1 \times \mathbf{a}_{1e} & \mathbf{e}_2 \times \mathbf{a}_{2e} & \mathbf{e}_3 \times \mathbf{a}_{3e} \end{bmatrix} \quad (6.89)$$

where, $\mathbf{e}_1 \equiv \mathbf{e}_2 \equiv \mathbf{e}_3 \equiv [0 \ 0 \ 1]^T$. Moreover,

$$\mathbf{a}_{1e} \equiv \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3 \equiv [a_1 c_1 + a_2 c_{12} + a_3 c_{123} \quad a_1 s_1 + a_2 s_{12} + a_3 s_{123} \quad 0]^T$$

$$\mathbf{a}_{T2e} \equiv \mathbf{a}_2 + \mathbf{a}_3 \equiv [a_2 c_{12} + a_3 c_{123} \quad a_2 s_{12} + a_3 s_{123} \quad 0]^T$$

$$\mathbf{a}_{T3e} \equiv \mathbf{a}_3 \equiv [a_3 c_{123} \quad a_3 s_{123} \quad 0]^T$$

in which s_1 , c_1 , s_{12} and c_{12} are defined after Eq. (6.89), whereas s_{123} and c_{123} are defined as follows:

$$s_{123} \equiv \sin(\theta_1 + \theta_2 + \theta_3); \text{ and } c_{123} \equiv \cos(\theta_1 + \theta_2 + \theta_3).$$

Once the elements of the vectors, \mathbf{e}_i and \mathbf{a}_{ie} , for $i = 1, 2, 3$, are substituted in Eq. (6.89), there will be three nonzero rows corresponding to the planar three degrees-of-freedom motion of the manipulator, which are relevant, i.e.,

$$\mathbf{J} = \begin{bmatrix} 1 & 1 & 1 \\ -a_1 s_1 - a_2 s_{12} - a_3 s_{123} & -a_2 s_{12} - a_3 s_{123} & -a_3 s_{123} \\ a_1 c_1 + a_2 c_{12} + a_3 c_{123} & a_2 c_{12} + a_3 s_{123} & a_3 c_{123} \end{bmatrix} \quad (6.90)$$

The above three rows refer to the components of the scalar value of angular velocity about the Z_1 -axis, and the two values for the linear velocity along axes X_1 , Y_1 , respectively.

Example 6.26 Anthropomorphic Articulated Arm

The Jacobian matrix of the anthropomorphic articulated arm shown in Fig. 6.13 is obtained from Eq. (6.86). The elements of the vectors are obtained with the help of the transformation matrices given in Example 6.7, i.e.,

$$[\mathbf{e}_1]_l \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; [\mathbf{e}_2]_l \equiv [\mathbf{e}_3]_l \equiv \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \quad (6.91a)$$

and

$$[\mathbf{a}_{1e}]_l \equiv [\mathbf{a}_{2e}]_l \equiv \begin{bmatrix} c_1(a_2 c_2 + a_3 c_{23}) \\ s_1(a_2 c_2 + a_3 c_{23}) \\ a_2 s_2 + a_3 s_{23} \end{bmatrix}; [\mathbf{a}_{3e}]_l \equiv [\mathbf{a}_3]_l \equiv \begin{bmatrix} a_3 c_1 c_{23} \\ a_3 s_1 c_{23} \\ a_3 s_{23} \end{bmatrix} \quad (6.91b)$$

Hence, the 6×3 Jacobian matrix is given by

$$\mathbf{J} = \begin{bmatrix} 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \\ -s_1(a_2 c_2 + a_3 c_{23}) & -c_1(a_2 s_2 + a_3 s_{23}) & -a_3 c_1 s_{23} \\ c_1(a_2 c_2 + a_3 c_{23}) & -s_1(a_2 s_2 + a_3 s_{23}) & -a_1 s_1 s_{23} \\ 0 & a_2 c_2 + a_3 c_{23} & a_3 c_{23} \end{bmatrix} \quad (6.91c)$$

Example 6.27 Jacobian of a PUMA Robot using MATLAB

Since evaluating the Jacobian expression for the PUMA robot is difficult, a MATLAB program is written in Fig. 6.21 for the configuration shown in Fig. 6.6.

```

syms a1 a2 a3 a4 a5 a6 b1 b2 b3 b4 b5 b6 al1 al2 al3 al4 al5 al6 th1 th2
th3 th4 th5 th6;

%For PUMA Robot DH parameters
al1=-pi/2;al2=0;al3=pi/2;al4=-pi/2;al5=pi/2;al6=0; b1=0; b3=0; b5=0;
sal1 = -1; cal1 = 0; sal2 = 0; cal2 = 1; sal3 = 1; cal3 = 0; sal4 = -1;
cal4 = 0; sal5 = 1;
cal5 = 0; sal6 = 0; cal6 = 1;
th1=0;th2=-pi/2;th3=pi/2;th4=0;th5=0;th6=0;
sth1 = 0; cth1 = 1; sth2 = -1; cth2 = 0; sth3 = 1; cth3 = 0; sth4 =
0; cth4 = 1; sth5 = 0;
sth5 = 1; sth6 = 0; cth6 = 1;

t1m = [cth1, -sth1*call, sth1*sall, al1*cth1; sth1, cth1*call, -cth1*sall,
al1*sth1; 0,sall,call,b1; 0,0,0,1];
t2m = [cth2, -sth2*cal2, sth2*sall, a2*cth2; sth2, cth2*cal2, -cth2*sall,
a2*sth2; 0,sall,cal2,b2; 0,0,0,1];
t3m = [cth3, -sth3*cal3, sth3*sall, a3*cth3; sth3, cth3*cal3, -cth3*sall,
a3*sth3; 0,sall,cal3,b3; 0,0,0,1];
t4m = [cth4, -sth4*cal4, sth4*sall, a4*cth4; sth4, cth4*cal4, -cth4*sall,
a4*sth4; 0,sall,cal4,b4; 0,0,0,1];
t5m = [cth5, -sth5*cal5, sth5*sall, a5*cth5; sth5, cth5*cal5, -cth5*sall,
a5*sth5; 0,sall,cal5,b5; 0,0,0,1];
t6m = [cth6, -sth6*cal6, sth6*sall, a6*cth6; sth6, cth6*cal6, -cth6*sall,
a6*sth6; 0,sall,cal6,b6; 0,0,0,1];

%To extract a_ie vectors in their own frames
t56m=t5m*t6m; t46m=t4m*t56m; t36m=t3m*t46m; t26m=t2m*t36m;
t16m=t1m*t26m;
a6v=t6m(1:3,4);a56v=t56m(1:3,4);a46v=t46m(1:3,4);a36v=t36m(1:3,4);a26v
=t26m(1:3,4);
a16v=t16m(1:3,4);

%To form rotation matrices for coordinate transformations
q1m=t1m(1:3,1:3); q12m=q1m*t2m(1:3,1:3); q13m=q12m*t3m(1:3,1:3);
q14m=q13m*t4m(1:3,1:3);
q15m=q14m*t5m(1:3,1:3);

%e_i and e_i x a_ie vectors in the 1st frame
e6v_1=simple(q15m(1:3,3)), e6ca6v_1=simple(q15m*[-a6v(2);a6v(1);0])
e5v_1=simple(q14m(1:3,3)), e5ca56v_1=simple(q14m*[-a56v(2);a56v(1);0])
e4v_1=simple(q13m(1:3,3)), e4ca46v_1=simple(q13m*[-a46v(2);a46v(1);0])
e3v_1=simple(q12m(1:3,3)), e3ca36v_1=simple(q12m*[-a36v(2);a36v(1);0])
e2v_1=simple(q1m(1:3,3)), e2ca26v_1=simple(q1m*[-a26v(2);a26v(1);0])
e1v_1=[0;0;1], e1ca16v_1=[-a16v(2);a16v(1);0]

```

Fig. 6.21 Texts of file ‘ch6jac.m’ to Jacobian of PUMA

Example 6.28 Jacobain for the PUMA Configuration shown in Fig. 6.6

The Jacobian matrix of the PUMA robot for the configuration shown in Fig. 6.6 is obtained from the MATLAB program given in Fig. 6.20 as

$$\mathbf{J} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ -b_2 & a_2 + b_4 + b_6 & b_4 + b_6 & 0 & b_6 & 0 \\ a_1 + a_{3456} & 0 & 0 & a_{456} & 0 & a_6 \\ 0 & -a_{3456} & -a_{3456} & 0 & -a_{56} & 0 \end{bmatrix} \quad (6.92)$$

where $a_{56} \equiv a_5 + a_6$; $a_{456} \equiv a_4 + a_{56}$; and $a_{3456} \equiv a_3 + a_{456}$. From Eq. (6.92), at least the first three rows can be verified to be true from Fig. 6.6 as they are the components of the unit vectors representing the axes of the revolute joints of the PUMA robot represented in the fixed frame, i.e., frame 1.

6.6 JACOBIAN USING THE DECOUPLED NATURAL ORTHOGONAL COMPLEMENT (DeNOC)

In this section, the concept of the Decoupled Natural Orthogonal Complement (DeNOC) matrices (Saha, 1999), originally introduced for the dynamic modeling of serial robots and later used for tree-type, parallel and other closed-loop systems, is first derived.

Next, it is shown how the DeNOC can be used to derive the Jacobian matrix using the DeNOC.

6.6.1 Definition of the DeNOC

First, the twist of a rigid body is defined as

$$\mathbf{t}_i \equiv \begin{bmatrix} \boldsymbol{\omega}_i \\ \dot{\mathbf{o}}_i \end{bmatrix} \quad (6.93)$$

where $\boldsymbol{\omega}_i$ and $\dot{\mathbf{o}}_i$ are the three-dimensional vectors of angular velocity and velocity of the origin of the i^{th} body, namely, O_i . Using the definition of twist given in Eq. (6.93), Eqs. (6.81a-b) are now re-written as (Bhangale, 2004)

$$\mathbf{t}_i = \mathbf{A}_{i,i-1} \mathbf{t}_{i-1} + \mathbf{p}_i \dot{\theta}_i \quad (6.94a)$$

where $\dot{\theta}_i$ is the joint rate of the i^{th} revolute joint, \mathbf{e}_i is the unit vector along the axis of the i^{th} joint, and \mathbf{a}_{i-1} is the vector denoting the point O_i with respect to O_{i-1} , as shown in Fig. 6.20 for \mathbf{a}_i . Moreover, \mathbf{t}_{i-1} is the 6-dimensional twist vector of the $(i-1)^{\text{st}}$ body. Furthermore, the 6×6 matrix $\mathbf{A}_{i,i-1}$ and the 6-dimensional vector \mathbf{p}_i are defined by

Origin of the DeNOC

It has first appeared in 1995 when it was presented in the International Conference on Robotics and Automation held in Nagoya, Japan.

$$\mathbf{A}_{i,i-1} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{a}_{i,i-1} \times \mathbf{1} & \mathbf{1} \end{bmatrix}; \text{ and } \mathbf{p}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{0} \end{bmatrix} \quad (6.94b)$$

In Eq. (6.94b), $\mathbf{a}_{i,i-1} \equiv -\mathbf{a}_{i-1,i} = -\mathbf{a}_{i-1}$, is the vector from the origin of the i^{th} body to the $(i-1)^{\text{st}}$ one, $\mathbf{1}$ and $\mathbf{0}$ are the 3×3 identity and zero matrices, respectively, $\mathbf{0}$ is the three-dimensional vector of zeros, and $\mathbf{a}_{i,i-1} \times \mathbf{1}$ is the 3×3 cross-product tensor associated with vector $\mathbf{a}_{i,i-1}$, which is defined as $(\mathbf{a}_{i,i-1} \times \mathbf{1})\mathbf{x} = \mathbf{a}_{i,i-1} \times \mathbf{x}$, for any three-dimensional Cartesian vector \mathbf{x} . Matrix $\mathbf{A}_{i,i-1}$ and vector \mathbf{p}_i have the following physical interpretations:

1. If $\dot{\theta}_i = 0$, i.e., when the two links $(i-1)$ and i are rigidly connected, $\mathbf{A}_{i,i-1} \mathbf{t}_{i-1}$ propagates the twist \mathbf{t}_{i-1} to \mathbf{t}_i . That is the angular velocity of link i remains same as that of $(i-1)$, namely, $\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1}$, and the linear velocity of the point O_i , $\dot{\mathbf{o}}_i$, is obtained from the velocity of the point O_{i-1} , $\dot{\mathbf{o}}_{i-1}$, and the angular velocity of link $(i-1)$, $\boldsymbol{\omega}_{i-1}$, namely, $\dot{\mathbf{o}}_i = \dot{\mathbf{o}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{a}_{i-1}$. The 6×6 matrix $\mathbf{A}_{i,i-1}$ is termed here as the *twist propagation* matrix (Saha, 1999) that has the following property:

$$\mathbf{A}_{i-1,i} \mathbf{A}_{i,i+1} = \mathbf{A}_{i-1,i+1} \quad (6.95)$$

2. When $\dot{\theta}_i \neq 0$, i.e., when the i^{th} joint motion is allowed, $\mathbf{p}_i \dot{\theta}_i$ adds to the components of $\mathbf{A}_{i,i-1} \mathbf{t}_{i-1}$, thus, giving rise to the actual twist of the i^{th} body \mathbf{t}_i . Hence, the six-dimensional vector \mathbf{p}_i is referred as the *joint-motion propagation* vector of the i^{th} joint. Note here that if the i^{th} joint is prismatic, the expression of \mathbf{p}_i in Eq. (6.94a) changes to

$$\mathbf{p}_i \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_i \end{bmatrix} \quad (6.96)$$

and no change occurs in the expression of the matrix $\mathbf{A}_{i,i-1}$. Now, for n bodies of the serial manipulator, Eq. (6.94a) are given by

$$\mathbf{t}_1 = \mathbf{p}_1 \dot{\theta}_2 \quad (6.97a)$$

$$\mathbf{t}_2 = \mathbf{A}_{21} \mathbf{t}_1 + \mathbf{p}_2 \dot{\theta}_2 \quad (6.97b)$$

⋮

$$\mathbf{t}_n = \mathbf{A}_{n,n-1} \mathbf{t}_{n-1} + \mathbf{p}_n \dot{\theta}_n \quad (6.97c)$$

Equations (6.97a-c) are written in a compact form as

$$\mathbf{t} = \mathbf{At} + \mathbf{N}_d \dot{\theta} \quad (6.98a)$$

where the $6n$ -dimensional generalized twist vector \mathbf{t} , the $6n \times 6n$ lower block bi-diagonal matrix \mathbf{A} , the $6n \times n$ block diagonal matrix \mathbf{N}_d and the n -dimensional joint-rate vector $\dot{\theta}$ are given by

$$\mathbf{t} \equiv \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_n \end{bmatrix}; \mathbf{A} \equiv \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{A}_{n,n-1} & \mathbf{0} \end{bmatrix}; \mathbf{N}_d \equiv \begin{bmatrix} \mathbf{p}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \cdots & & \mathbf{p}_n \end{bmatrix}; \dot{\theta} \equiv \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} \quad (6.98b)$$

Rewriting Eq. (6.98a) as

$$(\mathbf{1} - \mathbf{A}) \mathbf{t} = \mathbf{N}_d \dot{\boldsymbol{\theta}} \quad (6.99)$$

one can easily find the generalized twist, \mathbf{t} , defined as, $\mathbf{t} \equiv [\mathbf{t}_1^T \ \cdots \ \mathbf{t}_n^T]^T$, in terms of the joint-rate vector, $\dot{\boldsymbol{\theta}}$, i.e.,

$$\mathbf{t} = \mathbf{N} \dot{\boldsymbol{\theta}}, \text{ where } \mathbf{N} \equiv \mathbf{N}_l \mathbf{N}_d \quad (6.100)$$

The matrix $\mathbf{N}_l \equiv (\mathbf{1} - \mathbf{A})^{-1}$ can be obtained as

$$\mathbf{N}_l \equiv (\mathbf{1} - \mathbf{A})^{-1} \equiv \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \mathbf{A}_{21} & 1 & 0 & \cdots & 0 \\ \mathbf{A}_{31} & \mathbf{A}_{32} & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ \mathbf{A}_{n,1} & \cdots & \cdots & \mathbf{A}_{n,n-1} & 1 \end{bmatrix} \quad (6.101)$$

In Eq. (6.100), the result of \mathbf{N} is referred as the Natural Orthogonal Compliment (NOC) matrix, originally proposed by Angeles and Lee (1988), whereas the decoupled form given by Eq. (6.100) called the Decoupled NOC (DeNOC) matrices appeared in Saha (1997). It will be shown below how the DeNOC matrices can be used to derive the Jacobian matrix of a robot manipulator.

6.6.2 Derivation of Jacobian Matrix

The Jacobian matrix is derived here from the definitions of the DeNOC matrices, appearing in Eq. (6.100). Note that the end-effector of a robot manipulator is nothing but a part of the n^{th} body with its position different from the origin O_n . It is located at a point denoted by \mathbf{a}_{ne} from O_n . Thus, a twist-propagation matrix, say, \mathbf{A}_{en} , as defined after Eq. (6.94b), is introduced to find the end-effector twist \mathbf{t}_e from \mathbf{t}_n , namely,

$$\mathbf{t}_e = \mathbf{A}_{en} \mathbf{t}_n \quad (6.102a)$$

where the 6×6 matrix \mathbf{A}_{en} is given by

$$\mathbf{A}_{en} \equiv \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{a}_{en} \times \mathbf{1} & \mathbf{1} \end{bmatrix} \quad (6.102b)$$

Vector $\mathbf{a}_{en} = -\mathbf{a}_{ne}$ is defined similar to the vector $\mathbf{a}_{i+1,i} \equiv -\mathbf{a}_{i,i+1}$ of Fig. 6.20. Substituting the expression of \mathbf{t}_n from the generalized twist expression, \mathbf{t} of Eq. (6.98b), Eq. (6.102a) can be re-written as

$$\mathbf{t}_e = \mathbf{A}_{en} \mathbf{N}_{ln} \mathbf{N}_d \dot{\boldsymbol{\theta}} \quad (6.103)$$

where the $6 \times 6n$ matrix \mathbf{N}_{ln} is as follows:

$$\mathbf{N}_{ln} \equiv [\mathbf{A}_{n1} \ \mathbf{A}_{n2} \ \dots \ \mathbf{A}_{n,n-1} \ \mathbf{1}] \quad (6.104)$$

in which \mathbf{N}_d and $\dot{\boldsymbol{\theta}}$ are given in Eq. (6.98b). Substituting the expressions of \mathbf{N}_{ln} , Eq. (6.104), and \mathbf{N}_d , Eq. (6.98b), into Eq. (6.103), one obtains

$$\mathbf{t}_e = \mathbf{A}_{en} [\mathbf{A}_{n1} \mathbf{p}_1 \ \mathbf{A}_{n2} \mathbf{p}_2 \ \dots \ \mathbf{A}_{n,n-1} \mathbf{p}_n \ \mathbf{p}_n] \dot{\boldsymbol{\theta}}_i \quad (6.105a)$$

Using the twist-propagation matrix property, Eq. (6.95), Eq. (6.105a) can be written in the form of Eq. (6.62c), where

$$\mathbf{J} \equiv [\mathbf{A}_{e1} \mathbf{p}_1 \ \mathbf{A}_{e2} \mathbf{p}_2 \ \dots \ \mathbf{A}_{en} \mathbf{p}_n] \quad (6.105b)$$

In Eq. (6.105b), \mathbf{J} is the $6 \times n$ Jacobian matrix. If the expressions for the matrix, \mathbf{A}_{ei} , and the vector, \mathbf{p}_i , for $i = 1, \dots, n$, as defined in Eq. (6.94b), are substituted in Eq. (6.105b), the familiar expression of \mathbf{J} , as in Eq. (6.86), results where $\mathbf{a}_{ei} = -\mathbf{a}_{ie} - \cdots - \mathbf{a}_{ie}$ being the vector joining the origin of the i^{th} link, O_i , with a point on the end-effector. Depending on the requirement, any of the expressions for the Jacobian matrix, i.e., Eq. (6.86) or Eq. (6.105b), may be used to the user's advantage, e.g., to achieve efficiency or to get better physical interpretations, etc.

Example 6.29 Jacobian for Anthropomorphic Arm using the DeNOC

Referring to Eq. (6.105a-b), the Jacobian for the anthropomorphic arm is given by

$$\mathbf{J} \equiv [\mathbf{A}_{e1}\mathbf{p}_1 \ \mathbf{A}_{e2}\mathbf{p}_2 \ \mathbf{A}_{e3}\mathbf{p}_3] \quad (6.106a)$$

where the 6×6 matrices $\mathbf{A}_{e,i}$, and the 6-dimensional vectors \mathbf{p}_i , for $i = 1, 2, 3$, are shown below:

$$\mathbf{A}_{21} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{a}_{ei} \times \mathbf{1} & \mathbf{1} \end{bmatrix}, \text{ and } \mathbf{p}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{0} \end{bmatrix} \quad (6.106b)$$

Expansion of the expression in each column of \mathbf{J} given by Eq. (6.106a) yields the following:

$$\mathbf{A}_{ei}\mathbf{p}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{a}_{ei} \times \mathbf{e}_i \end{bmatrix} = \begin{bmatrix} \mathbf{e}_i \\ -\mathbf{e}_i \times \mathbf{a}_{ei} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_i \times \mathbf{a}_{ie} \end{bmatrix} \quad (6.106c)$$

where $\mathbf{a}_{ei} = -\mathbf{a}_{ie}$ was used. Finally, putting $i = 1, 2, 3$, in Eq. (6.106c) and substituting them into Eq. (6.106a), one gets

$$\mathbf{J} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ \mathbf{e}_1 \times \mathbf{a}_{1e} & \mathbf{e}_2 \times \mathbf{a}_{2e} & \mathbf{e}_3 \times \mathbf{a}_{3e} \end{bmatrix} \quad (6.106d)$$

which is the same as obtained in Eq. (6.89) but the vectors are spatial in this example given by Eq. (6.91), whereas the same were planar in Eq. (6.89) associated to the three-DOF planar manipulator.

6.7 FORWARD AND INVERSE VELOCITY ANALYSES

Unlike forward and inverse position analyses, the same for velocity analyses are relatively straightforward. For example, in forward velocity analysis, given a set of joint rates denoted with $\dot{\theta}$ and, of course, the kinematic parameters of the robots, i.e., the DH parameters, one can easily compute the end-effector's velocities using the recursive relations of Eqs. (6.84-6.85) or the twists of Eq. (6.97). One can also find the Jacobian matrix of the robot, i.e., \mathbf{J} , using Eq. (6.86) or Eq. (6.105b), explicitly and multiply it with the joint rate vector $\dot{\theta}$ using Eq. (6.62c) or Eq. (6.103). In either way, only simple multiplications and additions are required.

In inverse velocity analysis, as illustrated in Fig. 6.1, however, one has to compute the Jacobian matrix, i.e., \mathbf{J} of Eq. (6.86) or (6.105b), explicitly before it is used for the solution of the joint rates $\dot{\theta}$ using the set of linear algebraic equations given by Eq. (6.62c) or Eq. (6.103).

The joint-rates $\dot{\theta}$ can be solved as

$$\dot{\theta} = \mathbf{J}^{-1} \mathbf{t}_e \quad (6.107)$$

Equation (6.107) is only symbolic, and the explicit inversion of matrix \mathbf{J} is never recommended as it is numerically expensive requiring order (n^3) , i.e., $O(n^3)$, computations, namely, multiplications/divisions (M) and additions/subtractions (A), for calculating the inverse of \mathbf{J} , and $O(n^2)$ to multiply \mathbf{J}^{-1} with \mathbf{t}_e . Instead, *LU decomposition* of \mathbf{J} , followed by forward and backward substitutions (Strang, 1980) should be performed to solve for the joint rates, $\dot{\theta}$. This is outlined as follows:

$$\mathbf{J} = \mathbf{L}\mathbf{U} \quad (6.108a)$$

where the $n \times n$ matrices \mathbf{L} and \mathbf{U} have the following forms:

$$\mathbf{L} \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \text{ and } \mathbf{U} \equiv \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \quad (6.108b)$$

Substituting Eqs. (6.108a–b) in Eq. (6.86) or Eq. (6.105), one can solve for the unknown vector of joint rates from the two triangular systems of linear equations, namely,

$$\mathbf{Ly} = \mathbf{t}_e \text{ and } \mathbf{U}\dot{\theta} = \mathbf{y} \quad (6.109)$$

In Eq. (6.109), \mathbf{y} is first solved and then $\dot{\theta}$ by applying only forward and backward substitutions, respectively. Note that *LU* decomposition requires only $O(n^3/3)$ arithmetic operations, followed by $O(n^2/2)$ operations for each of the forward and backward substitutions. Hence, the overall complexity for inverse velocity analysis using Eqs. (6.108–6.109) is less than that of using Eq. (6.107).

6.8 ACCELERATION ANALYSIS

Typically, the acceleration analysis is required in the dynamic calculations where the inertia forces are equated with the external forces. Acceleration expressions can be obtained by differentiating the velocity expressions. For example, the end-effector accelerations of a robot are obtained by differentiating Eq. (6.62c) or Eq. (6.105a) as

$$\dot{\mathbf{t}}_e = \mathbf{J}\ddot{\theta} + \dot{\mathbf{J}}\dot{\theta} \quad (6.110a)$$

where, $\dot{\mathbf{t}}_e \equiv [\dot{\boldsymbol{\omega}}_e^T \ \dot{\mathbf{v}}_e^T]^T$, $\dot{\boldsymbol{\omega}}_e$ and $\dot{\mathbf{v}}_e$ being the angular and linear accelerations of the end-effector, respectively, and $\dot{\theta} \equiv [\dot{\theta}_1, \dots, \dot{\theta}_n]^T$, in which $\dot{\theta}_i$ represents the joint acceleration for the i^{th} joint. The i^{th} column of the time derivative of the Jacobian \mathbf{J} , denoted as $\dot{\mathbf{J}}$, is given from Eq. (6.87a) by

$$\frac{d\mathbf{j}_i}{dt} \equiv \begin{bmatrix} \dot{\mathbf{e}}_i \\ \dot{\mathbf{e}}_i \times \mathbf{a}_{ie} + \mathbf{e}_i \times \dot{\mathbf{a}}_{ie} \end{bmatrix}, \text{if the joint } i \text{ is revolute} \quad (6.110b)$$

$$\text{and } \frac{d\mathbf{j}_i}{dt} \equiv \begin{bmatrix} \mathbf{0} \\ \dot{\mathbf{e}}_i \times \mathbf{a}_{ie} + \mathbf{e}_i \times \dot{\mathbf{a}}_{ie} \end{bmatrix}, \text{if the joint } i \text{ is prismatic} \quad (6.110c)$$

Vectors $\dot{\mathbf{e}}_i$ and $\ddot{\mathbf{a}}_{ie}$ are the time derivatives of the vectors \mathbf{e}_i and \mathbf{a}_{ie} , respectively. The expressions for three-dimensional vectors $\dot{\mathbf{e}}_i$ and $\ddot{\mathbf{a}}_{ie}$ can be obtained as

$$\dot{\mathbf{e}}_i \equiv \boldsymbol{\omega}_i \times \mathbf{e}_i \text{ and } \ddot{\mathbf{a}}_{ie} \equiv \ddot{\mathbf{a}}_i + \dot{\mathbf{a}}_{i+1} + \cdots + \dot{\mathbf{a}}_n \quad (6.110d)$$

where $\dot{\mathbf{e}}_1 = \mathbf{0}$ as the first joint axis is attached to the fixed-link, and $\dot{\mathbf{a}}_i \equiv \boldsymbol{\omega}_i \times \mathbf{a}_i$, if joint i is revolute or $\dot{\mathbf{a}}_i \equiv \boldsymbol{\omega}_i \times \mathbf{a}_i + \dot{b}\mathbf{e}_i$, if the joint i is prismatic. The foregoing expressions are invariant and hence, valid in any coordinate frame. However, they are going to be incorporated into the matrix $\dot{\mathbf{J}}$, and then the latter is to be multiplied by vector $\dot{\boldsymbol{\theta}}$, as indicated in Eq. (6.110a). Thus, eventually all columns of $\dot{\mathbf{J}}$, i.e., three-dimensional vectors of Eq. (6.110c), will have to be represented in the same frame.

Once the values of $\dot{\mathbf{J}}$ are obtained, it is a simple matter to perform the forward and inverse acceleration analysis, i.e., add $\dot{\mathbf{J}}\dot{\boldsymbol{\theta}}$ to $\mathbf{J}\ddot{\boldsymbol{\theta}}$ which is obtained in a way $\mathbf{J}\ddot{\boldsymbol{\theta}}$ was obtained during the forward velocity analyses using Eq. (6.86) or Eq. (6.105a). This will give the end-effector's accelerations, i.e., $\dot{\mathbf{t}}_e$. For the inverse acceleration analyses, i.e., to find $\ddot{\boldsymbol{\theta}} = \mathbf{J}^{-1}(\dot{\mathbf{t}}_e - \mathbf{J}\dot{\boldsymbol{\theta}})$, one needs to follow exactly the same steps as for the inverse velocity analysis given by Eqs. (6.109), where \mathbf{t}_e of Eq. (6.107) is to be replaced with $(\dot{\mathbf{t}}_e - \mathbf{J}\dot{\boldsymbol{\theta}})$. Rest of the operations is same as given in Eqs. (6.108-6.109).

Example 6.30 Acceleration Analysis of the Two-link Revolute Jointed Arm

Using Eq. (6.88b) or Eq. (6.110b), the time rate of the Jacobian matrix $\dot{\mathbf{J}}$ for the two-link revolute jointed arm can be given as

$$\dot{\mathbf{J}} \equiv \begin{bmatrix} -a_1 c_1 \dot{\theta}_1 - a_2 c_{12} (\dot{\theta}_1 + \dot{\theta}_2) & a_2 c_{12} (\dot{\theta}_1 + \dot{\theta}_2) \\ -a_1 s_1 \dot{\theta}_1 - a_2 s_{12} (\dot{\theta}_1 + \dot{\theta}_2) & -a_2 s_{12} (\dot{\theta}_1 + \dot{\theta}_2) \end{bmatrix} \quad (6.111)$$

The acceleration of the end-effector $\dot{\mathbf{t}}_e$ can then be evaluated using the expression for \mathbf{J} and $\dot{\mathbf{J}}$ given by Eqs. (6.88b) and (6.111), respectively.

Example 6.31 Acceleration Analysis of the Anthropomorphic Arm

Similar to the planar case in Example 6.30, the time rate of the Jacobian matrix, $\dot{\mathbf{J}}$, for the anthropomorphic arm is given by

$$\dot{\mathbf{J}} = \begin{bmatrix} 0 & c_1 \dot{\theta}_1 & c_1 \dot{\theta}_1 \\ 0 & s_1 \dot{\theta}_1 & s_1 \dot{\theta}_1 \\ 0 & 0 & 0 \\ -c_1 \dot{\theta}_1 (a_{2c} + a_{3c}) + s_1 (a_{2s} \dot{\theta}_2 + a_{3s} \dot{\theta}_{23}) & s_1 \dot{\theta}_1 (a_{2s} + a_{3s}) - c_1 (a_{2c} \dot{\theta}_2 + a_{3c} \dot{\theta}_{23}) & a_{3s} s_1 \dot{\theta}_1 - a_{3c} c_1 \dot{\theta}_{23} \\ -s_1 \dot{\theta}_1 (a_{2c} + a_{3c}) - c_1 (a_{2s} \dot{\theta}_2 + a_{3s} \dot{\theta}_{23}) & -c_1 \dot{\theta}_1 (a_{2s} + a_{3s}) - s_1 (a_{2c} \dot{\theta}_2 + a_{3c} \dot{\theta}_{23}) & -a_{3s} c_1 \dot{\theta}_1 - a_{3c} s_1 \dot{\theta}_{23} \\ 0 & -(a_{2s} \dot{\theta}_2 + a_{3s} \dot{\theta}_{23}) & -a_{3s} c_1 \dot{\theta}_{23} \end{bmatrix} \quad (6.112)$$

where $a_{2s} \equiv a_2 s_2$; $a_{2c} \equiv a_2 c_2$; $a_{3s} \equiv a_3 s_{23}$; $a_{3c} \equiv a_3 c_{23}$.

SUMMARY

In this chapter, first, the rigid body kinematics is introduced. Position and orientation representations are given. Based on the above concepts, position, velocity, and acceleration analyses of a serial robot are carried out. Emphasis is given on forward and inverse position analyses and the Jacobian matrix that appear in velocity analysis. The concept of the DeNOC matrices is also introduced for the derivation of the Jacobian matrix. How to obtain forward and inverses kinematics for the velocity and acceleration are also outlined. Several examples are provided for understanding of the concepts introduced in this chapter.

EXERCISES

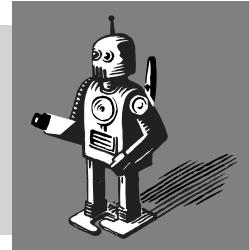
- 6.1** Define forward and inverse kinematics.
- 6.2** If the two links of a two-link planar manipulator have equal lengths, find out the expression for the homogeneous transformation matrix.
- 6.3** Find the overall transformation matrix for the prismatic-revolute planar arm robot shown in Fig. 5.31.
- 6.4** Find out the individual and overall homogeneous transformation matrices for the spherical arm shown in Fig. 5.32.
- 6.5** What is the purpose of a wrist? What is its typical degrees-of-freedom (DOF) in an industrial robot?
- 6.6** Can a position arm be used as a wrist even it may have same DOF?
- 6.7** Is angular velocity the time derivative Euler angles? If no, how are they related?
- 6.8** What is Jacobian of a robot system?
- 6.9** Calculate the Jacobian of a two-link planar arm when $\theta_1 = 45^\circ$ and $\theta_2 = 20^\circ$.
- 6.10** Find the expressions for t_e and \dot{t}_e and corresponding $\dot{\theta}$ and $\ddot{\theta}$ for the two-link planar arm when $\theta_1 = 45^\circ$ and $\theta_2 = 0^\circ$. Comment if there is any difficulty.
- 6.11** Derive the Jacobian matrix for Exercise 6.2 above.
- 6.12** Why is acceleration analysis important?

MATLAB AND ROBOANALYZER BASED EXERCISES

- 6.13** Find the overall transformation matrix for the SCARA robot shown in Fig. 5.36.
- 6.14** Express the Jacobian matrix for the SCARA robot shown in Fig. 5.36.
- 6.15** Repeat Exercises 6.11 to 6.13 for the architecture of articulated arm shown in Fig. 5.37.
- 6.16** Repeat Exercises 6.11 to 6.13 for the architecture of the spherical wrist shown in Fig. 5.38.
- 6.17** Perform forward and inverse kinematics of KUKA KR-5 robot using Roboanalyzer software.
- 6.18** Generate the plots of the joint angles given as input to the analysis of KUKA KR-5 [Exercise 6.17].
- 6.19** Perform inverse kinematics of a spatial revolute-revolute-prismatic (RRP) jointed robot using Roboanalyzer. Visualize their configurations.
- 6.20** Move the RRP from Solution 1 to Solution 2 of the inverse kinematics problem using the “FKin” command of the RoboAnalyzer.

7

Statics and Manipulator Design



When a robot manipulator performs a given task, such as lifting up a workpiece from a machine, its end-effector exerts moment and force to the external environment at the point of contact. The force and moment are generated by the actuators installed at various joints. For a serial robot manipulator, actuator moments and forces are transmitted through an open-loop chain to the point of contact. In statics, relationships between the joint torques and forces, and the Cartesian moments and forces at the end-effector are sought. It is of practical importance in determining the quality of moment and force transmission through various joints of a robot manipulator. It serves as a basis for sizing the links and bearings of the manipulator and for selecting appropriate actuators. The results can also be used for compliance control. The statics of spatial mechanism can be treated by various methods, for example, the vector method, the principle of virtual work, and others.

In this chapter, methods for representing static forces and moments acting on a robot manipulator and their transformation between different coordinate systems are developed. The transformation of moments and forces between the actuator space and the end-effector space is the focal point of this study. It is shown that the actuator input moments and forces are related to the output forces at the end-effector by the transpose of the robot's Jacobian matrix derived in Chapter 6. In addition, the free-body diagrams are introduced for the derivation of reactions generated at various joints. A thorough understanding of the joint reactions is important for proper sizing of links and the actuators at the design stage. First, the basic equations governing the static balance of a link are derived. Then these equations are applied for the static analysis of serial manipulators. The concept of equivalent joint torques and transformation between the end-effector forces and equivalent joint torque are described. Later, some design steps in selecting appropriate link lengths, robot architecture, etc., are explained.

What is Statics?

Statics is the branch of applied physics concerned with the analysis of moments and forces acting on a system that are in static equilibrium.

Static Equilibrium

In static equilibrium, a system is either at rest, or moving at constant velocity through its center of mass.

7.1 FORCES AND MOMENTS BALANCE

In a serial robot manipulator, each link is connected to one or two other links by various joints. Figure 7.1 depicts the forces and moments acting on a typical link i or $\#i$ that is connected to link $i - 1$ by joint i and to link $i + 1$ by joint $i + 1$. The forces acting on link i by link $i - 1$ through joint i can be reduced to a resultant force $\mathbf{f}_{i-1,i}$ and a resultant moment $\mathbf{n}_{i-1,i}$ about the origin O_i of the i th coordinate frame attached to the $(i - 1)$ st link. Similarly, the forces acting on link $i + 1$ by link i at the $(i + 1)$ st joint can be reduced to a resultant force $\mathbf{f}_{i,i+1}$ and a moment $\mathbf{n}_{i,i+1}$ about the origin O_{i+1} of the $(i + 1)$ st coordinate frame attached to the i th link. The following notations are then defined:

- $\mathbf{f}_{i-1,i}$: Three-dimensional vector of resulting force exerted on link i ($\#i$) by link $i - 1$ at O_i
- $\mathbf{n}_{i-1,i}$: Three-dimensional vector of resulting moment exerted on link i ($\#i$) by link $i - 1$ at O_i
- $\mathbf{f}_{i+1,i}$: Three-dimensional vector of resulting force exerted on link i ($\#i$) by link $i + 1$ at O_{i+1} . Note that, $\mathbf{f}_{i+1,i} = -\mathbf{f}_{i,i+1}$
- $\mathbf{n}_{i+1,i}$: Three-dimensional vector of resulting moment exerted on link i ($\#i$) by link $i + 1$ at O_{i+1} , such that $\mathbf{n}_{i+1,i} = -\mathbf{n}_{i,i+1}$.
- \mathbf{g} : Three-dimensional vector of acceleration due to gravity
- \mathbf{d}_i : Three-dimensional vector denoting the position of the mass center of the i th link ($\#i$) C_i relative to the origin of the i^{th} frame, i.e., O_i , as indicated in Fig. 7.1
- \mathbf{r}_i : Three-dimensional vector denoting the position of the origin of the $(i + 1)$ st frame, i.e., O_{i+1} , relative to the center of mass of Link i ($\#i$) C_i .
- \mathbf{a}_i : Three-dimensional position vector of O_{i+1} with respect to O_i such that $\mathbf{a}_i = \mathbf{d}_i + \mathbf{r}_i$.

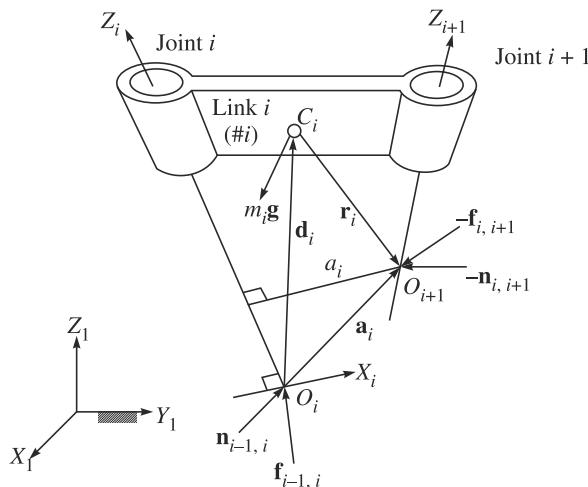


Fig. 7.1 Forces and moments acting on link i

Consider now the force and moment balance in Fig. 7.1. There are three forces exerted on link i , namely, $\mathbf{f}_{i-1,i}$, $\mathbf{f}_{i+1,i}$ ($= -\mathbf{f}_{i,i+1}$), and $m_i \mathbf{g}$. The force balance equation is then written as

$$\mathbf{f}_{i-1,i} - \mathbf{f}_{i,i+1} + m_i \mathbf{g} = \mathbf{0} \quad (7.1)$$

Next, consider the balance of moments about O_i of link i , i.e., the point where link $i-1$ connects to link i . There are two moments acting on link i , i.e., $\mathbf{n}_{i-1,i}$ and $\mathbf{n}_{i,i+1}$ ($= -\mathbf{n}_{i,i+1}$). In addition, the two forces $m_i \mathbf{g}$ and $-\mathbf{f}_{i,i+1}$ produce moments about O_i . Summing these moments together, one obtains

$$\mathbf{n}_{i-1,i} - \mathbf{n}_{i,i+1} - \mathbf{a}_i \times \mathbf{f}_{i,i+1} + \mathbf{d}_i \times m_i \mathbf{g} = \mathbf{0} \quad (7.2)$$

where $\mathbf{f}_{i-1,i}$ and $\mathbf{n}_{i-1,i}$ are the reaction force and moment between link $i-1$ and link i . For $i = 1$, \mathbf{f}_{01} and \mathbf{n}_{01} represent the force and moment exerted on the first moving link by the fixed base. For $i = n$, $\mathbf{f}_{n,n+1}$ and $\mathbf{n}_{n,n+1}$ represent the force and moment exerted on the environment by the end-effector. In this regard, the environment is treated as an additional link, numbered $n+1$. Equations (7.1) and (7.2) are written for every moving link, $i = 1, \dots, n$, to yield $6n$ equations in $6(n+1)$ number of reaction forces and moments. Therefore, to yield a unique solution, six of the reaction forces and moments should be specified. When a manipulator performs a given task, such as insertion or grinding, the end-effector exerts some force and/or moment on its environment. On the other hand, when the manipulator carries an object, the weight of the object becomes a load on the end-effector. Hence, considering the end-effector output force and moment $\mathbf{f}_{n,n+1}$ and $\mathbf{n}_{n,n+1}$ as known quantity, Eqs. (7.1–7.2) can be solved for the remaining reaction forces and moments using the following equations:

$$\begin{bmatrix} 1 & \mathbf{0} & -1 & -\mathbf{a}_1 \times \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & -1 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & & & \ddots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \vdots & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \vdots & \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{n}_{01} \\ \mathbf{f}_{01} \\ \mathbf{n}_{12} \\ \mathbf{f}_{12} \\ \vdots \\ \mathbf{n}_{n-1,n} \\ \mathbf{f}_{n-1,n} \end{bmatrix} = \begin{bmatrix} -\mathbf{d}_1 \times m_1 \mathbf{g} \\ -m_1 \mathbf{g} \\ -\mathbf{d}_2 \times m_2 \mathbf{g} \\ -m_2 \mathbf{g} \\ \vdots \\ -\mathbf{d}_n \times m_n \mathbf{g} + \mathbf{a}_n \times \mathbf{f}_{n,n+1} \\ -m_n \mathbf{g} + \mathbf{f}_{n,n+1} \end{bmatrix} \quad (7.3)$$

Note that Eq. (7.3) is arranged by writing Eq. (7.2) first, followed by Eq. (7.1). This is done to represent Eq. (7.3) in a compact form using the definition of wrench, which is a six-dimensional vector, namely,

$$\mathbf{w}_{i-1,i} \equiv \begin{bmatrix} \mathbf{n}_{i-1,i} \\ \mathbf{f}_{i-1,i} \end{bmatrix} \quad (7.4)$$

where the three-dimensional vectors $\mathbf{f}_{i-1,i}$ and $\mathbf{n}_{i-1,i}$ are the reaction force on link i by link $i-1$ at O_i and the moment on link i by link $i-1$ about O_i , respectively. Using Eq. (7.4), Eq. (7.3) can then be re-written as

$$\begin{bmatrix} \mathbf{1} & -\mathbf{A}'_{12} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & -\mathbf{A}'_{n-1,n} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{01} \\ \vdots \\ \mathbf{w}_{n-2,n-1} \\ \mathbf{w}_{n-1,n} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^g \\ \vdots \\ \mathbf{w}_{n-1}^g \\ \mathbf{w}_n^g + \mathbf{w}_n^e \end{bmatrix} \quad (7.5a)$$

where the 6×6 matrix, $\mathbf{A}'_{i,i+1}$, is defined as

$$\mathbf{A}'_{i,i+1} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{a}_{i,i+1} \times \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \quad (7.5b)$$

in which $\mathbf{a}_{i,i+1}$ denotes the three-dimensional vector from the point O_i to O_{i+1} , i.e., $\mathbf{a}_{i,i+1} \equiv \mathbf{a}_i$. Matrix $\mathbf{A}'_{i,i+1}$ is termed the *wrench propagation matrix* in Chaudhary and Saha (2009), which is nothing but the transpose of the *twist propagation matrix* between the links i and $i + 1$, as introduced in Chapter 6. Moreover, $\mathbf{w}_i^g \equiv -m_i[(\mathbf{d}_i \times \mathbf{g})^T, \mathbf{g}^T]^T$ and $\mathbf{w}_n^g \equiv [(\mathbf{a}_n \times \mathbf{f}_{n,n+1})^T, \mathbf{f}_{n,n+1}^T]^T$ are the 6-dimesional external wrench vectors due to gravity and the known external force on the link n , respectively. Furthermore, using the definitions, $\mathbf{w}^J \equiv [\mathbf{w}_{01}^T \ \mathbf{w}_{12}^T \ \dots \ \mathbf{w}_{n-1,n}^T]^T$, and $\mathbf{w}' \equiv [\mathbf{w}_1^{gT} \ \mathbf{w}_2^{gT} \ \dots \ (\mathbf{w}_n^g + \mathbf{w}_n^e)^T]^T$, Eq. (7.5) can be solved for \mathbf{w}^J , as

$$\mathbf{w}^J = \mathbf{N}_u \mathbf{w}' \quad (7.6a)$$

where

$$\mathbf{N}_u \equiv \begin{bmatrix} \mathbf{1} & \mathbf{A}'_{12} & \cdots & \mathbf{A}'_{1n} \\ \mathbf{0} & \mathbf{1} & \cdots & \mathbf{A}'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad (7.6b)$$

Note that \mathbf{N}_u is nothing but the transpose of the matrix \mathbf{N}_l given by Eq. (6.101).

7.2 RECURSIVE CALCULATIONS

In this section, a recursive method for the static analysis of a serial manipulator is developed. The recursive method solves joint reaction forces and moments of one link at a time without the need for solving $6n$ equations, Eq. (7.3) or (7.5) simultaneously. To facilitate the analysis based on the definition of wrench given in Eq. (7.4), Eqs. (7.1-7.2) are written recursively as

$$\mathbf{f}_{i-1, i} = \mathbf{f}_{i,i+1} - m_i \mathbf{g} \quad (7.7a)$$

$$\mathbf{n}_{i-1, i} = \mathbf{n}_{i,i+1} + \mathbf{a}_i \times \mathbf{f}_{i,i+1} - \mathbf{d}_i \times m_i \mathbf{g} \quad (7.7b)$$

Note that the vectors in Eqs. (7.7a-b) are not yet expressed in any coordinate frame. However, the position vector \mathbf{r}_i is often specified in the $(i + 1)$ st frame where it has constant representation. On the other hand, the vector \mathbf{a}_i can be conveniently expressed in the i th frame in terms of the DH parameters as

$$[\mathbf{a}_i]_i \equiv \begin{bmatrix} a_i c_i \\ a_i s_i \\ b_i \end{bmatrix} \quad (7.8a)$$

where s_i and c_i respectively represent $\sin \theta_i$ and $\cos \theta_i$. Vectors \mathbf{d}_i can then be obtained as

$$[\mathbf{d}_i]_i \equiv [\mathbf{a}_i]_i - [\mathbf{r}_i]_i = \begin{bmatrix} a_i c_i \\ a_i s_i \\ b_i \end{bmatrix} - \mathbf{Q}_i \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix}; \text{ where } [\mathbf{r}_i]_{i+1} \equiv \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix} \quad (7.8b)$$

In Eq. (7.8b) r_{ix} , r_{iy} , and r_{iz} are the components of the vector \mathbf{r}_i along X_{i+1} -, Y_{i+1} -, and Z_{i+1} -axes of the $(i+1)$ st frame, and \mathbf{Q}_i is the rotation matrix transforming the vector representation in the $(i+1)$ st frame to the i th frame. Using Eqs. (7.7a-b), the joint reaction moments and forces can be computed recursively. The process starts with the last link where the end-effector is connected. It continues with one link at a time and ends at the base link. For $i = n$, the end-effector moment and force, $\mathbf{n}_{n+1,n}$ ($= -\mathbf{n}_{n,n+1}$) and $\mathbf{f}_{n+1,n}$ ($= -\mathbf{f}_{n,n+1}$) are considered known in the frame attached to it, i.e., $-[\mathbf{n}_{n,n+1}]_{n+1}$ and $-[\mathbf{f}_{n,n+1}]_{n+1}$ are inputs. Hence, Eqs. (7.7a-b) give the reaction moment and force, $\mathbf{n}_{n-1,n}$ and $\mathbf{f}_{n-1,n}$ at the n th joint. The values can be evaluated suitably either in the n th or $(n-1)$ st coordinate frame. The process is repeated for $i = n-1, \dots, 1$ until all the reaction moments and forces are found. The computation is now shown. Equations (7.7a-b) for the i th link are first written in the $(i+1)$ st frame which is attached to it, i.e.,

$$[\mathbf{f}_{i-1,i}]_{i+1} = [\mathbf{f}_{i,i+1} - m_i \mathbf{g}]_{i+1} \quad (7.9a)$$

$$[\mathbf{n}_{i-1,i}]_{i+1} = [\mathbf{n}_{i,i+1} + \mathbf{a}_i \times \mathbf{f}_{i,i+1} + \mathbf{d}_i \times m_i \mathbf{g}]_{i+1} \quad (7.9b)$$

Once the reaction forces and moments are computed in the $(i+1)$ st frame, they are converted into the i th frame by the following transformations:

$$[\mathbf{f}_{i-1,i}]_i = \mathbf{Q}_i [\mathbf{f}_{i-1,i}]_{i+1}, \text{ and } [\mathbf{n}_{i-1,i}]_i = \mathbf{Q}_i [\mathbf{n}_{i-1,i}]_{i+1} \quad (7.10)$$

Note that the term \mathbf{g} in Eqs. (7.9a-b) denotes the acceleration due to gravity expressed in the $(i+1)$ st frame. Since \mathbf{g} is usually specified in the fixed frame, i.e., frame 1, it should be converted into the frame attached to the link before it is substituted into Eqs. (7.9a-b). This can be accomplished by the following recursive formula: For $i = 1, \dots, n$,

$$[\mathbf{g}]_{i+1} = \mathbf{Q}_i^T [\mathbf{g}]_i \quad (7.11)$$

Moreover, if the end-effector's output moment and force are specified in the fixed frame, they should also be transformed into the end-effector's frame in the same manner, i.e.,

$$[\mathbf{f}_{n,n+1}]_{n+1} = \mathbf{Q}^T [\mathbf{f}_{n,n+1}]_1 \text{ and } [\mathbf{n}_{n,n+1}]_{n+1} = \mathbf{Q}^T [\mathbf{n}_{n,n+1}]_1 \quad (7.12)$$

where, $\mathbf{Q} \equiv \mathbf{Q}_1 \dots \mathbf{Q}_n \dots \mathbf{Q}_i$, for $i = 1, \dots, n$, being the rotation matrix between the i th and $(i+1)$ st frames, i.e., the first 3×3 matrix of the homogeneous transformation matrix \mathbf{T}_i derived in Eq. (5.61b).

7.3 EQUIVALENT JOINT TORQUES

Once the reaction forces and moments in the joints are known, the actuator forces and/or torques can be determined. For a serial manipulator, each joint is driven by an actuator that exerts either a force or a torque between the two adjacent links.

These actuating forces and/or torques can be found by projecting the reaction forces onto their corresponding joint axes. For a revolute joint, the actuator exerts a torque about the i th joint axis. Assuming that frictional torque at the joint is negligible, the actuator force τ_i is given by

$$\tau_i = \mathbf{e}_i^T \mathbf{n}_{i-1, i} \quad (7.13)$$

where \mathbf{e}_i is the unit vector pointing along the positive i th joint axis, i.e., Z_i of Fig. 7.1, about which the relative motions between the two neighboring links take place. The actuator torque has that component of $\mathbf{n}_{i-1, i}$ which is corresponding to the direction of the joint axis. Its other two components are reactions that are to be supported by the joint bearing. The term τ_i is called the joint torque. For a prismatic joint, the actuator force is similarly obtained. It is the force exerted along the i th joint axis. Again, assuming that frictional force at the joint is negligible, the actuator force denoted with the same letter τ_i , is expressed as

$$\tau_i = \mathbf{e}_i^T \mathbf{f}_{i-1, i} \quad (7.14)$$

where \mathbf{e}_i is the unit vector pointing along the positive i th joint axis along which the two neighboring links translate. Equation (7.14) implies that the actuator force only bears the component of $\mathbf{f}_{i-1, i}$ along the direction of the joint axis, while its other two components are supported by the joint bearings. Using the 6-dimensional wrench notation $\mathbf{w}_{i-1, i}$ the joint force or torque τ_i can be written as

$$\tau_i = \mathbf{p}_i^T \mathbf{w}_{i-1, i} \quad (7.15)$$

where \mathbf{p}_i is the 6-dimensional *joint-motion propagation vector* for the revolute or prismatic joint, as the case may be which are given in Eqs. (6.94b) or (6.96), respectively.

Example 7.1 Statics of a Two-link Planar Arm

The two-link planar manipulator arm is applying a force \mathbf{f} on the environment, say, a wall, with its end-effector. Assume that the force \mathbf{f} is known in the end-effector frame, i.e., in frame 3. Hence, $[\mathbf{f}_{23}]_3 \equiv [f_x, f_y, 0]^T$. The required joint torques are found as a function of the arm configurations and the applied force components. Note that the gravity does not play any role when the manipulator is lying on the horizontal plane. Referring to Fig. 7.2 and Eqs. (7.9–7.10),

$$[\mathbf{f}_{23}]_3 \equiv \begin{bmatrix} f_x \\ f_y \\ 0 \end{bmatrix}; \text{ and } [\mathbf{f}_{12}]_2 = \mathbf{Q}_2 [\mathbf{f}_{23}]_3 = \begin{bmatrix} f_x c_2 - f_y s_2 \\ f_x s_2 + f_y c_2 \\ 0 \end{bmatrix} \quad (7.16a)$$

where the orientation matrix \mathbf{Q}_2 is given by

$$\mathbf{Q}_2 \equiv \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.16b)$$

In Eqs. (7.16a-b), s_2 and c_2 respectively represent $\sin \theta_2$ and $\cos \theta_2$.

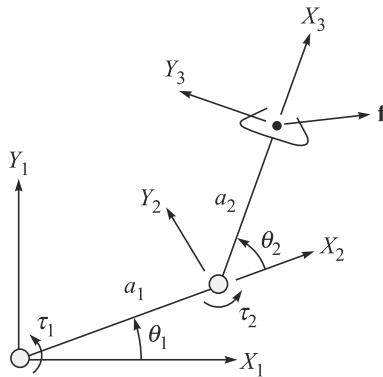


Fig. 7.2 A two-link planar arm applying a force

Next, since no external moment is exerted by the end-effector, i.e., $\mathbf{n}_{23} = \mathbf{0}$, the moment $[\mathbf{n}_{12}]_2$ is obtained as

$$[\mathbf{n}_{12}]_2 \equiv [\mathbf{a}_2]_2 \times [\mathbf{f}_{23}]_2 = [\mathbf{a}_2 \times \mathbf{f}_{12}]_2 = \begin{bmatrix} 0 \\ 0 \\ a_2 f_y \end{bmatrix}, \text{ where } [\mathbf{a}_2]_2 \equiv \begin{bmatrix} a_2 c_2 \\ a_2 s_2 \\ 0 \end{bmatrix} \quad (7.16c)$$

The force $[\mathbf{f}_{01}]_1$ and moment $[\mathbf{n}_{01}]_1$ are then evaluated as

$$[\mathbf{f}_{01}]_1 = \mathbf{Q}_1 [\mathbf{f}_{12}]_2 = \begin{bmatrix} f_x c_{12} - f_y s_{12} \\ f_x s_{12} + f_y c_{12} \\ 0 \end{bmatrix} \quad (7.17a)$$

$$[\mathbf{n}_{01}]_1 \equiv \mathbf{Q}_1 [\mathbf{n}_{12} + \mathbf{a}_1 \times \mathbf{f}_{12}]_2 = \mathbf{Q}_1 [\mathbf{n}_{12}]_2 + [\mathbf{a}_1 \times \mathbf{f}_{01}]_1 = \begin{bmatrix} 0 \\ 0 \\ a_2 f_y + a_1 f_x s_2 + a_1 f_y c_2 \end{bmatrix};$$

$$[\mathbf{a}_1]_1 \equiv \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix} \quad (7.17b)$$

where $[\mathbf{n}_{12}]_2$ will have the same expression as $[\mathbf{n}_{12}]_2$ of Eq. (7.16c). Finally, the joint torques to generate the force \mathbf{f} at the end-effector are given by

$$\tau_1 = a_1 f_x s_2 + (a_2 + a_1 c_2) f_y; \text{ and } \tau_2 = a_2 f_y \quad (7.18)$$

Example 7.2 Statics of a Revolute-Prismatic Planar Arm

A Revolute-Prismatic (RP) planar arm is shown in Fig. 7.3 whose end-effector is applying force \mathbf{f} . Its statics problem is solved here, i.e., joint torque and force are evaluated to exert force \mathbf{f} by the end-effector.

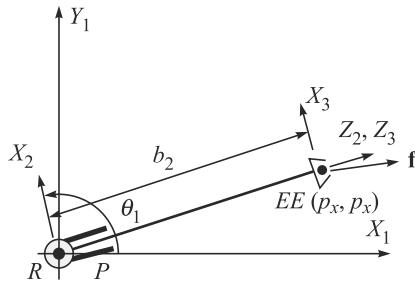


Fig. 7.3 A two-link revolute-prismatic arm applying a force

First, the forces at Joint 2 are evaluated from the given end-effector force, $\mathbf{f} \equiv [\mathbf{f}_{23}]_3$, i.e.,

$$[\mathbf{f}_{23}]_3 \equiv \begin{bmatrix} f_x \\ 0 \\ f_z \end{bmatrix}; \text{ and } [\mathbf{f}_{12}]_2 = \mathbf{Q}_2 [\mathbf{f}_{23}]_3 = \begin{bmatrix} f_x \\ 0 \\ f_z \end{bmatrix} \quad (7.19a)$$

where the orientation matrix \mathbf{Q}_2 is an identity matrix, as the frames 2 and 3 are parallel. Since no external moment is exerted by the end-effector, i.e., $\mathbf{n}_{23} = \mathbf{0}$, the moment $[\mathbf{n}_{12}]_2$, is obtained as

$$[\mathbf{n}_{12}]_2 \equiv \mathbf{Q}_2 [\mathbf{a}_2 \times \mathbf{f}_{23}]_3 = [\mathbf{a}_2]_2 \times [\mathbf{f}_{23}]_2 = [\mathbf{a}_2 \times \mathbf{f}_{12}]_2 = \begin{bmatrix} 0 \\ b_2 f_x \\ 0 \end{bmatrix}, \text{ where } [\mathbf{a}_2]_2 \equiv \begin{bmatrix} 0 \\ 0 \\ b_2 \end{bmatrix} \quad (7.19b)$$

The force $[\mathbf{f}_{01}]_1$ and moment $[\mathbf{n}_{01}]_1$ are then evaluated as

$$[\mathbf{f}_{01}]_1 = \mathbf{Q}_1 [\mathbf{f}_{12}]_2 = \begin{bmatrix} f_x c_1 + f_z s_1 \\ f_x s_1 - f_z c_1 \\ 0 \end{bmatrix} \quad (7.20a)$$

$$[\mathbf{n}_{01}]_1 \equiv [\mathbf{n}_{12}]_1 + [\mathbf{a}_1]_1 \times [\mathbf{f}_{12}]_1 = \mathbf{Q}_1 [\mathbf{n}_{12}]_2 + [\mathbf{a}_1 \times \mathbf{f}_{01}]_1 = \begin{bmatrix} 0 \\ 0 \\ b_2 f_x \end{bmatrix} \quad (7.20b)$$

where $[\mathbf{a}_1]_1 = \mathbf{0}$ because the DH parameter, $a_1 = 0$, and the matrix \mathbf{Q}_1 is calculated using the DH parameters given in Table 5.4(a) and Eq. (5.61b) as

$$\mathbf{Q}_1 \equiv \begin{bmatrix} c_1 & 0 & s_1 \\ s_1 & 0 & -c_1 \\ 0 & 1 & 0 \end{bmatrix} \quad (7.20c)$$

Finally, the joint torque in the revolute joint τ_1 and the force at the prismatic joint τ_2 are given by

$$\tau_1 = b_2 f_x; \text{ and } \tau_2 = f_z \quad (7.21)$$

Example 7.3 Statics of a Planar 3-DOF Planar Arm

Figure 7.4 shows a planar 3-link manipulator having three degrees of freedom (DOF) whose all joints are revolute. Four coordinate frames with all the Z-axes pointing out of the paper are defined for each link according to the DH convention defined in Chapter 5. Let the force and moment on the environment be given by $[\mathbf{f}_{34}]_4 = [f_x, f_y, 0]^T$ and $[\mathbf{n}_{34}]_4 = [0, 0, n_z]^T$, respectively. Also the acceleration due to gravity \mathbf{g} be pointing along the negative Y_1 , and the center of mass be located at the midpoint of each link. The joint reaction moments and forces are then obtained next. Note that the DH parameters of the manipulator are given in Table 5.3, where the transformation matrices needed for the kinematics analyses are derived in Example 6.4. Moreover, the vectors, \mathbf{a}_i , \mathbf{r}_i , and \mathbf{d}_i , for $i = 1, 2$ and 3 , are given as

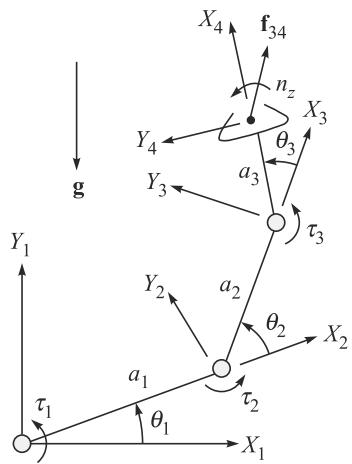


Fig. 7.4 A 3-link, 3-DOF planar manipulator

$$[\mathbf{a}_i]_i \equiv \begin{bmatrix} a_i c_i \\ a_i s_i \\ 0 \end{bmatrix}, [\mathbf{r}_i]_{i+1} \equiv \begin{bmatrix} \frac{1}{2} a_i \\ 0 \\ 0 \end{bmatrix} \text{ and } [\mathbf{d}_i]_i = [\mathbf{a}_i]_i - \mathbf{Q}_i [\mathbf{r}_i]_{i+1} = \begin{bmatrix} \frac{1}{2} a_i c_i \\ \frac{1}{2} a_i s_i \\ 0 \end{bmatrix} \quad (7.22)$$

where a_i , for $i = 1, 2, 3$, is the length of the i th link. Equation (7.22) is written in frame 1 using the rotation matrices, \mathbf{Q}_i , for $i = 1, 2, 3$, as

$$[\mathbf{a}_1]_1 = a_1 \begin{bmatrix} c_1 \\ s_1 \\ 0 \end{bmatrix}; \quad [\mathbf{d}_1]_1 = \frac{a_1}{2} \begin{bmatrix} c_1 \\ s_1 \\ 0 \end{bmatrix} \quad (7.23a)$$

$$[\mathbf{a}_2]_1 = \mathbf{Q}_1 [\mathbf{a}_2]_2 = a_2 \begin{bmatrix} c_{12} \\ s_{12} \\ 0 \end{bmatrix}; \quad [\mathbf{d}_2]_1 = \mathbf{Q}_1 [\mathbf{d}_2]_2 = \frac{a_2}{2} \begin{bmatrix} c_{12} \\ s_{12} \\ 0 \end{bmatrix} \quad (7.23b)$$

$$[\mathbf{a}_3]_1 = \mathbf{Q}_{12} [\mathbf{a}_3]_3 = a_3 \begin{bmatrix} c_{13} \\ s_{13} \\ 0 \end{bmatrix}; \quad [\mathbf{d}_3]_1 = \mathbf{Q}_{12} [\mathbf{d}_3]_3 = \frac{a_3}{2} \begin{bmatrix} c_{13} \\ s_{13} \\ 0 \end{bmatrix} \quad (7.23c)$$

where $\mathbf{Q}_{12} \equiv \mathbf{Q}_1 \mathbf{Q}_2$, and $s_{12} \equiv \sin \theta_{12}$, $c_{12} \equiv \cos \theta_{12}$, for $\theta_{12} \equiv \theta_1 + \theta_2$. Similarly, s_{13} and c_{13} are defined for $\theta_{13} \equiv \theta_{12} + \theta_3$. Equations (7.7a-b) are now applied to compute the reaction forces exerted on link 3. Then, proceed to links 2 and 1, in sequence.

For, $i = 3$, substitution of \mathbf{a}_3 , \mathbf{d}_3 , \mathbf{f}_{34} , and \mathbf{n}_{34} into Eqs. (7.7a-b) yields

$$[\mathbf{f}_{23}]_3 = \mathbf{Q}_3[\mathbf{f}_{34}]_4 - m_3 \mathbf{Q}_{12}^T [\mathbf{g}]_1 = \begin{bmatrix} f_x c_3 - f_y s_3 + m_3 g s_{12} \\ f_x s_3 + f_y c_3 + m_3 g c_{12} \\ 0 \end{bmatrix} \quad (7.24a)$$

$$[\mathbf{n}_{23}]_3 = \mathbf{Q}_3[\mathbf{n}_{34}]_4 + [\mathbf{a}_3]_3 \times \mathbf{Q}_3[\mathbf{f}_{34}]_4 - [\mathbf{d}_3]_3 \times m_3 \mathbf{Q}_{12}^T [\mathbf{g}]_1 \equiv \begin{bmatrix} 0 \\ 0 \\ n_{23z} \end{bmatrix} \quad (7.24b)$$

where $[\mathbf{g}]_1 \equiv [0, -g, 0]^T$ — g being the acceleration due to gravity and equals 9.81 m/s^2 . Moreover,

$$n_{23z} \equiv n_z + a_3 \left(f_y - \frac{1}{2} m_3 g c_{13} \right) \quad (7.24c)$$

For, $i = 2$, \mathbf{f}_{23} and \mathbf{n}_{23} are then used to obtain the following:

$$[\mathbf{f}_{12}]_2 = [\mathbf{f}_{23} - m_2 \mathbf{g}]_2 = \mathbf{Q}_2[\mathbf{f}_{23}]_3 - m_2 \mathbf{Q}_1^T [\mathbf{g}]_1 = \begin{bmatrix} f_x c_{23} - f_y s_{23} + m_{23} g s_1 \\ f_x s_{23} + f_y c_{23} + m_{23} g c_1 \\ 0 \end{bmatrix} \quad (7.25a)$$

$$[\mathbf{n}_{12}]_2 = \mathbf{Q}_2[\mathbf{n}_{23}]_3 + [\mathbf{a}_2]_2 \times \mathbf{Q}_2[\mathbf{f}_{23}]_3 - [\mathbf{d}_2]_2 \times m_2 \mathbf{Q}_1^T [\mathbf{g}]_1 \equiv \begin{bmatrix} 0 \\ 0 \\ n_{12z} \end{bmatrix} \quad (7.25b)$$

where $m_{23} \equiv m_2 + m_3$, and the term n_{21z} is given by

$$n_{12z} \equiv n_z - a_3 \left(f_y - \frac{1}{2} m_3 g c_{13} \right) + a_2 \left[f_x s_3 + f_y c_3 + \left(m_3 + \frac{1}{2} m_2 \right) g c_{12} \right] \quad (7.25c)$$

Similarly, for $i = 1$, the following are obtained:

$$[\mathbf{f}_{01}]_1 = \mathbf{Q}_1[\mathbf{f}_{12}]_2 - m_1 [\mathbf{g}]_1 = \begin{bmatrix} f_x c_{13} - f_y s_{13} \\ f_x s_{13} + f_y c_{13} + m_{13} g \\ 0 \end{bmatrix} \quad (7.26a)$$

$$[\mathbf{n}_{01}]_1 = \mathbf{Q}_1[\mathbf{n}_{12}]_2 + [\mathbf{a}_1]_1 \times \mathbf{Q}_1[\mathbf{f}_{12}]_2 - [\mathbf{d}_1]_1 \times m_1 [\mathbf{g}]_1 \equiv \begin{bmatrix} 0 \\ 0 \\ n_{01z} \end{bmatrix} \quad (7.26b)$$

where $m_{13} \equiv m_1 + m_2 + m_3$, and the term n_{01z} is given by

$$\begin{aligned} n_{01z} \equiv & n_z + a_3 \left(f_y - \frac{1}{2} m_3 g c_{13} \right) + a_2 \left[f_x s_3 + f_y c_3 + \left(m_3 + \frac{1}{2} m_2 \right) g c_{12} \right] \\ & + a_1 \left[f_x s_{23} + f_y c_{23} + \left(m_{23} + \frac{1}{2} m_1 \right) g c_1 \right] \end{aligned} \quad (7.26c)$$

Finally, Eq. (7.13) is applied to compute the joint torques, which are obtained as

$$\tau_1 = [\mathbf{e}_1]_1^T [\mathbf{n}_{01}]_1 = n_{01z}; \tau_2 = [\mathbf{e}_2]_2^T [\mathbf{n}_{12}]_2 = n_{12z}; \text{ and } \tau_3 = [\mathbf{e}_3]_3^T [\mathbf{n}_{23}]_3 = n_{23z} \quad (7.27)$$

where $[\mathbf{e}_i]_i \equiv [0, 0, 1]^T$, for $i = 1, 2, 3$, and n_{23z} , n_{12z} , n_{01z} are given in Eqs. (7.24c), (7.25c) and (7.26c), respectively.

7.4 ROLE OF JACOBIAN IN STATICS

The joint torques and forces obtained in the static situation balance the forces and moments acting on the end-effector. The relationship between these two sets of generalized forces, i.e., moments and forces on the end-effector in the Cartesian space and the joint torques and forces in the joint space, can be derived using the

Virtual Work

Virtual work on a system is the work resulting from the real forces acting through a virtual displacement.

Virtual Displacement

A virtual displacement is an assumed infinitesimal change of system coordinates occurring while time is held constant. It is called virtual rather than real since no actual displacement can take place without the passage of time.

principle of virtual work. Let $\delta\mathbf{x}$ and $\delta\boldsymbol{\theta}$ represent virtual displacements of the end-effector that include both linear and rotational components, and the manipulator joints, respectively. These displacements are consistent with any

constraints imposed on the system. For example, if the end-effector is in contact with a rigid wall then the virtual displacements in position are tangent to the wall.

Now, if \mathbf{w}_e denotes the wrench on the end-effector then the virtual work done by it is given by $\mathbf{w}_e^T \delta\mathbf{x}$. Similarly, if $\boldsymbol{\tau}$ denotes the vector of all the joint torques and forces acting at the joints to produce the wrench \mathbf{w}_e then the virtual work defined in the joint space is given by $\boldsymbol{\tau}^T \delta\boldsymbol{\theta}$. Note that when the robot manipulator is in equilibrium, the principle of virtual work states that

$$\mathbf{w}_e^T \delta\mathbf{x} = \boldsymbol{\tau}^T \delta\boldsymbol{\theta} \quad (7.28)$$

where the virtual displacements $\delta\mathbf{x}$ and $\delta\boldsymbol{\theta}$ can be related from the definition of Jacobian introduced in Chapter 6, which is

$$\delta\mathbf{x} = \mathbf{J}\delta\boldsymbol{\theta} \quad (7.29)$$

where \mathbf{J} is the Jacobian of the robot manipulator at hand. Substitution of Eq. (7.29) into Eq. (7.28) yields

$$\mathbf{w}_e^T \mathbf{J} \delta\boldsymbol{\theta} = \boldsymbol{\tau}^T \delta\boldsymbol{\theta} \quad (7.30)$$

The above equation holds true for all $\delta\boldsymbol{\theta}$, hence, the following is true:

$$\mathbf{w}_e^T \mathbf{J} = \boldsymbol{\tau}^T \quad (7.31)$$

Transposing both sides of Eq. (7.31) yields the relationship between the end-effector wrench and the joint torques and forces, i.e.,

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{w}_e \quad (7.32)$$

It is pointed out here that if the Jacobian matrix \mathbf{J} of Eq. (7.31) or (7.32), is singular then the end-effector cannot exert static forces as desired.

Example 7.4 Jacobian in Statics of Two-link Planar Arm

Two relations for the joint torques of the two-link planar manipulator arm presented in Example 7.1 were derived in Eq. (7.18). They are now rearranged as

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{f}, \text{ where } \boldsymbol{\tau} \equiv [\tau_1, \tau_2]^T, \mathbf{f} \equiv [f_x, f_y, 0]^T \quad (7.33a)$$

where the 3×2 matrix \mathbf{J} is given by

$$\mathbf{J} \equiv \begin{bmatrix} a_1 s_2 & 0 \\ a_1 c_2 + a_2 & a_2 \\ 0 & 0 \end{bmatrix} \quad (7.33b)$$

The nonzero rows of the matrix \mathbf{J} of Eq. (7.33b) is, however, not same as the Jacobian matrix for the two-link planar arm derived in Eq. (6.88b). This is due to the following reason: Matrix \mathbf{J} of Eq. (7.33b) is represented in Frame 3, as the values of the end-effector force \mathbf{f} is given in frame 3, whereas \mathbf{J} of Eq. (6.88b) is represented in the fixed frame, i.e., in frame 1. In fact, $[\mathbf{J}]_1 = \mathbf{Q}_1 \mathbf{Q}_2 [\mathbf{J}]_3$, i.e.,

$$[\mathbf{J}]_1 \equiv \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 s_2 & 0 \\ a_1 c_2 + a_2 & a_2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix} \quad (7.34)$$

where \mathbf{Q}_1 and \mathbf{Q}_2 are respectively the 3×3 rotation matrices, whereas $[\mathbf{J}]_1$ and $[\mathbf{J}]_3$ are respectively the representations of the same Jacobian in frames 1 and 3. Now, the nonzero rows of Eq. (7.34) match with the matrix given in Eq. (6.88b).

Example 7.5 Jacobian in Statics of the 3-DOF Planar Arm

Note that the expressions for the joint torques, Eqs. (7.26–7.27), can be arranged as follows:

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{w}_e + \mathbf{A}_g \mathbf{m}_g \quad (7.35a)$$

where the 3-, 6-, and 3-dimensional vectors $\boldsymbol{\tau}$, \mathbf{w}_e and \mathbf{m}_g , are defined as

$$\boldsymbol{\tau} \equiv [\tau_1, \tau_2, \tau_3]^T; \mathbf{w}_e \equiv [0, 0, n_z, f_x, f_y, 0]^T; \mathbf{m}_g \equiv [m_{1g}, m_{2g}, m_{3g}]^T \quad (7.35b)$$

whereas the 6×3 matrix \mathbf{J} and the 3×3 matrix \mathbf{A}_g are given by

$$\mathbf{J} \equiv \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ a_2 s_3 + a_1 s_{23} & a_2 s_3 & 0 \\ a_3 + a_2 c_3 + a_1 c_{23} & a_3 + a_2 c_3 & a_3 \\ 0 & 0 & 0 \end{bmatrix} \quad (7.35c)$$

and

$$\mathbf{A}_g \equiv \begin{bmatrix} \frac{1}{2}a_1c_1 & a_1c_1 + \frac{1}{2}a_2c_{12} & a_1c_1 + a_2c_{12} + \frac{1}{2}a_3c_{13} \\ 0 & \frac{1}{2}a_2c_{12} & a_2c_{12} + \frac{1}{2}a_3c_{13} \\ 0 & 0 & \frac{1}{2}a_3c_{13} \end{bmatrix} \quad (7.35d)$$

Two points are to be noted in Eq. (7.35a). First, unlike Eq. (7.32) or (7.33a), the expression for the joint torques contains an additional term, $\mathbf{A}_g \mathbf{m}_g$, in addition to $\mathbf{J}^T \mathbf{w}_e$. Second, the nonzero rows of the Jacobian matrix, \mathbf{J} of Eq. (7.35c), are not the same as obtained in Eq. (6.90). Actually, Eq. (7.32) has been derived without considering the link weights. Hence, no term associated with the link weights appeared. However, one may convert the link weights to equivalent end-effector forces and moments, and can re-write Eq. (7.32), where one would obtain an additional term like in Eq. (7.35a). Secondly, similar to the point mentioned in Example 7.4, the expression for the Jacobian matrix Eq. (7.35c) is represented in frame 4, as the wrench \mathbf{w}_e is known in frame 4, whereas the expression of \mathbf{J} in Eq. (6.90) is the representation in frame 1. In fact,

$$[\mathbf{J}]_1 = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_1 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_2 & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_3 & \mathbf{O} \\ \mathbf{O} & \mathbf{Q}_3 \end{bmatrix} [\mathbf{J}]_4 \quad (7.36)$$

where $\mathbf{Q}_1, \mathbf{Q}_2$ and \mathbf{Q}_3 are the 3×3 rotation matrices, and \mathbf{O} represents the 3×3 zero matrix. Moreover, $[\mathbf{J}]_1$ and $[\mathbf{J}]_4$ are respectively the representations of the same Jacobian in frames 1 and 4. Now, it can be shown that the nonzero rows of $[\mathbf{J}]_1$ are exactly same as those in Eq. (6.90).

Example 7.6 Static Torques in the Anthropomorphic Articulated Arm

An anthropomorphic articulated arm shown in Fig. 6.13 for which the Jacobian matrix is obtained in Eq. (6.91c). If moment \mathbf{n}_e and force \mathbf{f}_e are acting on the end-effector which are known in the end-effector frame itself then the torques required at the three joints of the arm can be calculated using Eq. (7.32), where no link weights are accounted for, as

$$\boldsymbol{\tau} = \mathbf{J}^T \hat{\mathbf{Q}} \mathbf{w}_e, \text{ where } \boldsymbol{\tau} \equiv [\tau_1, \tau_2, \tau_3]^T; \mathbf{w}_e \equiv [\mathbf{w}_{34}]_4 \equiv [n_x, n_y, n_z, f_x, f_y, f_z]^T \quad (7.37a)$$

and the 6×6 matrix, $\hat{\mathbf{Q}} \equiv \text{diag}[\mathbf{Q}, \mathbf{Q}]$, in which, $\mathbf{Q} = \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3$, transforms the representation of a vector from the end-effector frame, i.e., frame 4, to the fixed-frame, i.e., frame 1. This can be extracted from Eq. (6.11b) as

$$\mathbf{Q} \equiv \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 \\ s_1c_{23} & -s_1s_{23} & c_1 \\ -s_{23} & -c_{23} & 0 \end{bmatrix} \quad (7.37b)$$

```

syms a1 a2 a3 th1 th2 th3 nx ny nz fx fy fz;

%For Articulated 3-DOF Arm, initial values are
th1=0; th2=0; th3=0;
sth1=0;cth1=1;sth2=0;cth2=1;sth3=0;cth3=1;
sth23=0;cth23=1;
qm = [cth1*cth23, -cth1*sth23, -sth1; sth1*cth23, -sth1*sth23,
      cth1; -sth23, -cth23, 0];
ne_1 = qm*[nx;ny;nz];
fe_1 = qm*[fx;fy;fz];
jacm = [0,-sth1,-sth1;0,cth1,cth1;1,0,0;
        -sth1*(a2*cth2+a3*cth23),-cth1*(a2*sth2+a3*sth23),-
a3*cth1*sth23;
        cth1*(a2*cth2+a3*cth23), -sth1*(a2*sth2+a3*sth23),-
a1*sth1*sth23;
        0,-(a2*cth2+a3*cth23),-a3*cth23];
%Joint torques
tau=jacm.'*[ne_1;fe_1]

```

Fig. 7.5 Texts of file ‘ch7tor3aa.m’

The result of Eq. (7.37a) for the joint torques can be obtained using a MATLAB program given in Fig. 7.5. For the initial configuration, i.e., $\theta_1 = \theta_2 = \theta_3 = 0$, the output from the MATLAB can be checked as follows:

$$\boldsymbol{\tau} \equiv \begin{bmatrix} -n_y + (a_2 + a_3)f_z \\ -n_z + (a_2 + a_3)f_y \\ n_z + a_3f_y \end{bmatrix} \quad (7.37c)$$

which can be quickly checked from the configuration of the arm in Fig. 6.13.

7.5 MANIPULATOR DESIGN

In this section, design of mechanical structure of the robot manipulator consisting of linkages and joints capable of various movements are considered even though the design of the robot must include the design of its controllers, control hardware, etc. Some of the latter issues will be mentioned in Chapters 10–13, which are actually considered to be relatively developed and economically available compared to the mechanical design of the manipulator. In fact, very often the reason for high investment in robot usage is attributed to the low technological level of the mechanical structure of the manipulator.

Note that a human being with the total mass of 68 to 90 kg can handle loads up to 15 to 25 kg, i.e., $1/5^{\text{th}}$ to $1/4^{\text{th}}$ of its overall mass, whereas a manipulator can handle only about $1/20^{\text{th}}$ to $1/5^{\text{th}}$ of their total mass. Hence, a manipulator is about $1/5^{\text{th}}$ less effective than a human being. Another aspect to the design of a manipulator is that they are very different from any other structures used as machine tools or material-handling equipment. A better mechanical structure for a manipulator will lead to lower weight-to-payload ratio and higher stiffness, and natural frequency leading to

better accuracy and repeatability of the robot. Hence, some of the parameters that should be considered for the design of a robot manipulator are the following:

Functional Requirements Payload; Mobility or the degrees of freedom (DOF); Agility or effective speeds of execution of prescribed motions; Accuracy and repeatability

Kinetostatic Measures Workspace, its volume, shape, and how well a manipulator performs in these regions, etc.

Structural Design and Dynamics Structural stiffness or compliances, masses, damping coefficients and natural frequencies

Economics Cost, reliability, maintainability, etc.

The first three sets of parameters are discussed below to help a robot designer take appropriate decisions. Since economic aspects depend on so many varied factors, they are almost impossible to put under set rules. For example, the cost of robot installation depends on the land price, availability and cost of skilled manpower that can program and maintain the robots, price of electricity and other raw materials, etc. These can vary significantly from country to country or region to region. Hence, they are avoided in this book except that some related issues were discussed in Section 1.3.3. However, it is assumed that during the design process of a robot, the selection of link materials, actuators, sensors, and other components would be done based on the above economic considerations, namely, the ease of availability, reliability, maintainability, etc.

Note that many of the above parameters are closely interrelated. For example, depending on the point in the workspace, the payload of a robot can vary, and accordingly the speed, accelerations, repeatability, etc. If a load has to be carried by the robot arm while it is almost stretched, it will not be able carry much load compared to some other comfortable postures. This is similar to our human arms. Since the stretched arm acts as a long cantilever beam, its deflection at the gripper location will be large. Hence, the accuracy at that point of the workspace will be certainly less. Due to such variability, comprehensive guidelines for the design of a robot manipulator is extremely difficult and almost impossible. Depending on the applications and requirements, one needs to apply several criteria and possibly perform some kind of optimization before settling on a design. It is generally affected by the experience of the designer, as it is the case in any design.

7.6 FUNCTIONAL REQUIREMENTS OF A ROBOT

In this section, aspects of payload, mobility, speed, accuracy and repeatability are highlighted as these form the specifications of a robot manipulator. Anybody interested to use a robot for a particular application first look for these specifications.

7.6.1 Payload

It is the maximum mass which a robot can carry in any configuration and any location of its workspace. The load-carrying ability depends on the actuators' capacity. Note that depending on configuration or workspace, the mass-carrying capacity may change. If the working zone of a robot is chosen to be its comfortable zone, which

can be identified from its manipulability measures explained in Section 7.7.3, the load-carrying capacity can be higher.

7.6.2 Mobility

Mobility or degrees of freedom (DOF) of a robot depends on the task at hand. The DOF of a robot should match the DOF of the task. A typical pick-and-place operation of a rigid body of arbitrary shape requires six DOF of the robot. However, in special cases, all six may not be required. For example, if a robot is used to weld a cylindrical surface or it is carrying a cylindrical tool like a grinder for a polishing task, five DOF would be sufficient. In many instances, a robot with lesser DOF can perform a task of higher DOF where the workpiece is held in a positioning device, as indicated in Fig. 7.6.

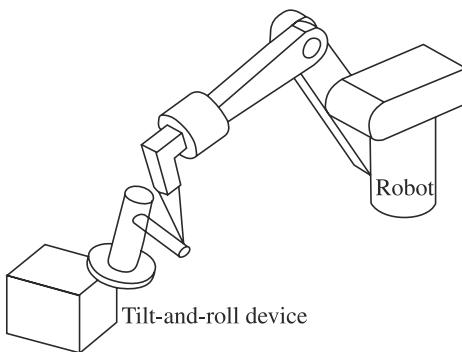


Fig. 7.6 A tilt-and-roll device provides additional DOF to the robot system

In many applications, for example, in space and in which the robot has to reach a normally not-reachable zone, redundant robots with more DOF than required by the task are used. The additional DOF can be used in case one or more of the actuators fail during operation in order to maintain uninterrupted action or to avoid what is called singularity (see Section 7.7.2) in order to reach the not-reachable zones. To provide more structural flexibility in a robot, highly redundant ones like spine or snake robots that consist of small chain like segments are also gaining popularity in industries.

7.6.3 Configuration

As explained in Chapter 2, a robot manipulator can have Cartesian, cylindrical, spherical, and articulated configurations. Advantages and disadvantages of these robot arms are mentioned in Table 2.3. Amongst the four, a Cartesian robot, mainly, the one used as a gantry robot has much better utilization of shop-floor space and safety assurance since the workspace is clearly indicated by the supporting columns. This is indicated in Fig. 2.15. Such robots have, however, disadvantages of higher effective inertia (Rivin, 1988) due to rectilinear frames. Such robots require larger motors leading to higher energy consumption.

In cylindrical robots, Fig. 2.12, there are two translational motions, namely, elevation and reach. It is possible to extend the telescopic radial motion by increasing the telescoping reach. Hence, they require a relatively small space, and layout planning is simplified. The telescopic structure can also reduce dynamic loads and allow faster transients since initial rotation can be executed with the folded arm, thus greatly reducing the effective moment of inertia. One disadvantage is their reduced work zone, and the arm cannot go below the bed of the structure.

A spherical robot, Fig. 2.13, is considered to have more complicated control system but has a larger workspace.

Articulated or jointed robots, as shown in Fig. 2.14, have the advantages of small size for their hardware and the floor space for their installations. These robots show a high degree of accessibility, and have only rotary joints which are easy to seal. Such robots require lower energy consumption. Shortcomings of articulated robots are their degeneration near the workspace boundaries and low stiffness in all directions.

7.6.4 Speed, Accuracy, and Repeatability

One of the reasons a robot is used in factories is to enhance the speed of an operation. Sometimes, however, the process itself limits the speed of the robot's movement. For example, the quality of weld may degrade with higher speed of operation. Hence, a judicious selection of a robot is important for a given set of applications.

High accuracy and repeatability are desirable in any robot manipulator. They are defined in Section 4.6, which are expensive to achieve. Sometimes such high accuracy is not required. For example, in paint spraying, the tolerance of the spray spot diameter may be more than the tolerance. To a large extent, accuracy of a particular robot model depends upon the precision of the manufacturing and assembly processes rather than the design. Additionally, these can be influenced by friction, hysteresis, backlash, compliance in links, joints and drives, etc., or in other words, by design and kinematics of the linkage, as well as by its rated payload, types of drives, etc.

7.6.5 Actuators and Sensors

Actuators and sensors are covered in Chapters 3 and 4, respectively. A transmission system, many times considered an integral part of an actuator, is explained in Chapter 2. For the choice of an actuator, it is important to know the applications, required payload, speed, repeatability, accuracy, etc., as the structural stiffness of an actuator, say, an electric motor with a gear box or a pneumatic system, influence such functional requirements of a robot. Section 3.4 provides guidelines to choose an actuator which should help any designer to appropriately select an actuator for an application.

On the other hand, selection of a sensor is relatively a complex phenomenon, particularly when many sensors like position, forces, vision, etc., have to be integrated for the robot to take a decision and act accordingly. In such a situation, the associated electronics which act as transducers, amplifiers, etc., would also influence the performance of the robot. For example, the speed of a robot may drastically hamper if the processing of the sensed signals is itself very slow. In any case, Section 4.6 define different sensor characteristics which will help a designer choose a set of appropriate sensors for an application.

7.7 KINEMATIC AND KINETOSTATIC MEASURES

One of the kinematic measures could be the measure of size and shape of a workspace resulted from a given manipulator architecture, as seen in Figs. 2.11–2.14, whereas kinetostatic measures are those which provide a clue as to how well a manipulator transmits motion and force while performing some tasks inside its workspace. These are important design criteria for robot manipulators, which are outlined in the following sections.

7.7.1 Workspace

Workspace of a robot manipulator, as defined in Chapter 2, is a space composed of all points which can be reached by its end-effector. Many times in the literature, a distinction is made between a *reachable* workspace from a *dexterous* one. While the former is defined as the volume or area that a manipulator can reach in at least one orientation of the end-effector, the latter is the volume or area that the end-effector can reach with all orientations. Depending on the configuration of a robot arm, say, Cartesian or cylindrical, etc., the shape and size of a workspace can vary significantly. Figures 2.11–2.14 show different shapes of the workspaces for Cartesian, cylindrical, spherical, and articulated robot arms. For the simple 2-DOF planar RP manipulator shown in Fig. 7.3, also referred as polar manipulator, the workspace can be obtained as an annular area shown in Fig. 7.7 where $b_{\min} \leq b \leq b_{\max}$, for $0^\circ \leq \theta \leq 360^\circ$.

In many instances, the size of a workspace may be big but the robot footprint is also large. Such situations may not lead to economic implementation of robots. In such situations, measures like *service index* or *Structural Length Index (SLI)*, as reported in Rivin (1988) and Craig (2009), respectively, are used to measure the size of a robot workspace with respect to its overall dimensions. They are defined below.

1. Service Index (SI), S It is defined as the ratio of the solid angle of a spatial robot enveloping all possible allowed configurations of the last (orienting) link at a given point of the workspace, θ , to the solid angle of the complete sphere, i.e., 4π steradians, which is given by

$$S = \frac{\theta}{4\pi} \quad (7.38a)$$

For a planar robot manipulator, Eq. (7.38a) is modified as

$$S_p = \frac{\theta}{2\pi} \quad (7.38b)$$

where 2π radians is the angle that can be covered by the last link of a planar manipulator. For the planar RP manipulator of Fig. 7.7, if a third link of length a_3 is added as wrist such that $a_3 < (b_{\max} - b_{\min})/2$ then the annular workspace within the boundary circles of radii $b_{\min} + a_3 \leq b_a \leq b_{\max} - a_3$ --- b_a being the radius of the point P --- will be characterized by $S_p = 1$. However, if one needs to calculate the service index S_p of the other reachable area, namely, near the outer and the inner circles, the same can be calculated as

$$S_p = \frac{\beta_1}{\pi} = \frac{1}{\pi} \cos^{-1} \frac{b_a^2 + a_3^2 - b_{\max}^2}{2b_a a_3} \text{ (outer)} \text{ or } \frac{\beta_2}{\pi} = \frac{1}{\pi} \cos^{-1} \frac{b_{\min}^2 - b_a^2 - a_3^2}{2b_a a_3} \text{ (inner)} \quad (7.39)$$

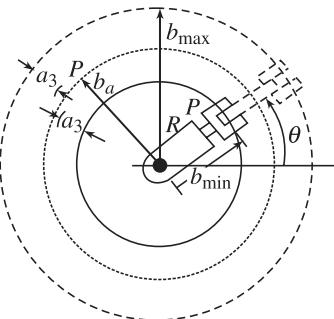


Fig. 7.7 Workspace of a 2-DOF RP planar manipulator

The plot of the workspace along with its SI (Rivin, 1988) is shown in Fig. 7.8.

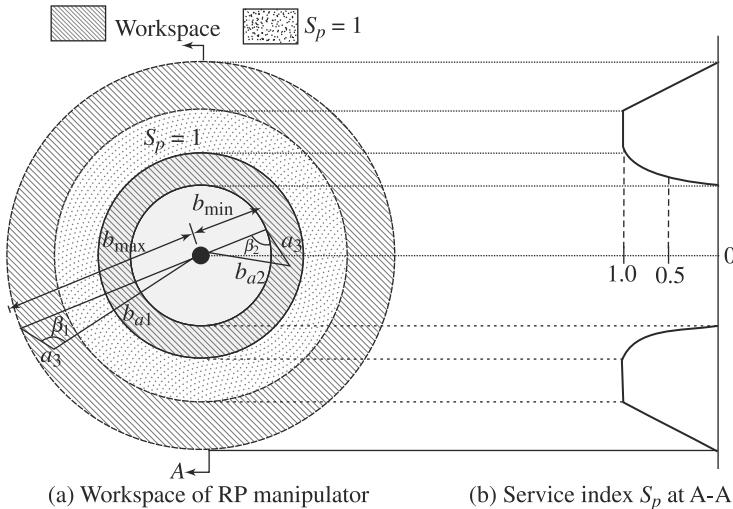


Fig. 7.8 Service index of RP robot

2. Structural Length Index (SLI), Q Another measure of the workspace with respect to the overall dimensions of the robot can be expressed using SLI Q which can be calculated using a formula given in Craig (2009), i.e.,

$$Q = \frac{L}{\sqrt[3]{W}}, \text{ where } L \equiv \sum_1^n (a_i + b_i) \quad (7.40)$$

The term L is the length sum of a manipulator in which a_i and b_i are the link length and joint offset, i.e., lengths related to Denavit–Hartenberg (DH) parameters defined in Chapter 5. For a prismatic joint, b_i should be interpreted as the maximum traverse by the joint. Moreover, W is the workspace volume of the manipulator. Referring to the Cartesian robot arm shown in Fig. 2.11, the value of Q can be calculated as 3, whereas for an articulated arm of Fig. 2.14, the value of Q is 0.62. The lower value of SLI for the articulated arm can be interpreted as that it requires less material compared to a Cartesian robot for the same workspace. Hence, in the literature, it is mentioned that articulated manipulators are superior to other configurations because they have minimal intrusion into their own workspace.

7.7.2 Singularity

Mathematically, if a square matrix is not invertible, one calls it singular. For a robot manipulator, if the Jacobian matrix relating the joint rates with the Cartesian velocities of the end-effector, as derived in Chapter 6, is not invertible then the robot is in singularity. One of the ways singularity can be detected is by finding the joint values of the manipulator corresponding to

Singular Matrix

It is a term from Linear Algebra. For a determined system of linear algebraic equations, $\mathbf{Ax} = \mathbf{b}$, i.e., where the number of equations are same as the number of unknowns, the singular matrix \mathbf{A} implies no solution for unknown \mathbf{x} . This happens when the determinant of the matrix \mathbf{A} vanishes.

$$\det[\mathbf{J}(\boldsymbol{\theta})] = 0 \quad (7.41)$$

Note in Eq. (7.41) that the matrix $\mathbf{J}(\boldsymbol{\theta})$ is a function of the manipulator configurations only, i.e., the joint angles denoted with $\boldsymbol{\theta}$. In robotics, the question of whether the matrix \mathbf{J} , as in Eq. (6.86), is singular or not is frequently asked. For nonsingular \mathbf{J} , one calculates the necessary joint rates at each instant of time along the path of end-effector's motion. The real question is: Is the Jacobian nonsingular or invertible for all values of the joint positions, i.e., $\boldsymbol{\theta}$? Most robots have values of $\boldsymbol{\theta}$ where the Jacobian becomes singular. Such locations are called singularities of the robot. All robot manipulators have singularities at the boundary of their workspace, and most have loci of singularities inside their workspace. An in-depth study of the classification of singularities is beyond the scope of this book. However, singularities are typically classified as

Workspace Boundary Singularities Such singularities occur when the manipulator is fully stretched out or folded back on itself such that the end-effector is near or at the boundary of the workspace.

Workspace Interior Singularities Such singularities occur away from the workspace boundary and generally are caused by two or more joint axes lining up.

Note that when a manipulator is in singular configuration, it has lost one or more degrees of freedom as viewed from the Cartesian space. This means that there is some direction in Cartesian space along which it is impossible to move the robot's end-effector no matter which joint rates are selected. It is obvious that such a situation happens at the workspace boundaries. Alternatively, the joint rates approach infinity as the singularity is approached. In practical situations, the controller will demand more power from the actuators to meet the infinity joint rate requirement. As a result, the actuator will blow off.

Example 7.7 Singularity of a Two-link Planar Arm

Referring to Fig. 7.9, the singularities can be obtained by equating the determinant of the associated Jacobian to zero, i.e., $\det(\mathbf{J}) = 0$. The 2×2 Jacobian matrix \mathbf{J} is given as

$$\mathbf{J} \equiv \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \quad (7.42)$$

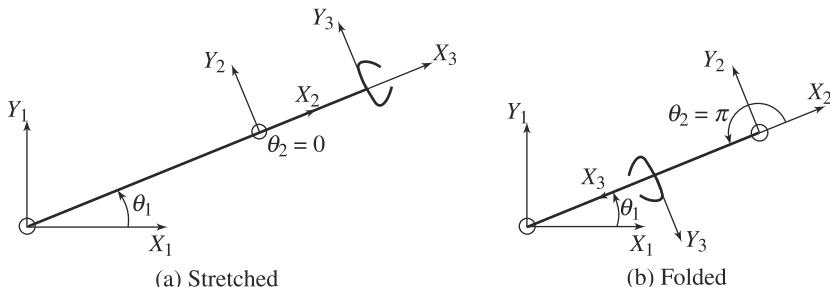


Fig. 7.9 Singular configurations of a two-link planar arm

where $\det(\mathbf{J}) = a_1 a_2 s_2$, which implies that the manipulator is singular when $\theta_2 = 0$ or π . Physically, when $\theta_2 = 0$, the arm is stretched straight out, as shown in Fig. 7.9(a). In this configuration, motion of the end-effector is possible only along the perpendicular to the stretched manipulator arm. Likewise, when $\theta_2 = \pi$, the arm is folded completely back on itself, as shown in Fig. 7.9(b). In this configuration, the motion of the manipulator is again only possible perpendicular to the links. In a way, the motion is possible along one Cartesian direction only instead of two. These are called *boundary singularities*.

Example 7.8 Singularity of the Anthropomorphic Arm

Referring to Fig. 6.13 and Example 6.26, the singularities of the arm corresponding to the linear motions can be obtained from the determinant of the associated Jacobian, i.e., using the last three rows of Eq. (6.91c). This is denoted as \mathbf{J}_v , which is given by

$$\mathbf{J}_v = \begin{bmatrix} -s_1(a_2 c_2 + a_3 c_{23}) & -c_1(a_2 s_2 + a_3 s_{23}) & -a_3 c_1 s_{23} \\ c_1(a_2 c_2 + a_3 c_{23}) & -s_1(a_2 s_2 + a_3 s_{23}) & -a_3 s_3 s_{23} \\ 0 & -a_2 c_2 + a_3 c_{23} & a_3 c_{23} \end{bmatrix} \quad (7.43a)$$

whose determinant can be obtained as

$$\det(\mathbf{J}_v) = -a_2 a_3 s_3 (a_2 c_2 + a_3 c_{23}) \quad (7.43b)$$

From Eq. (7.43b), it is clear that one of the singular configuration is when $\theta_3 = 0$ or π , i.e., links 2 and 3 are in one line, i.e., stretched or folded, as in the case of a two-link planar arm.

7.7.3 Dexterity and Manipulability

As seen in Section 7.7.2, if the determinant of the Jacobian matrix \mathbf{J} is zero then the manipulator is in singularity. However, during actual operation of a robot in an application, one does not even want the robot to go close to singularity. This led to several definitions like dexterity, manipulability, and others. *Dexterity* of a manipulator is defined as the determinant of the Jacobian \mathbf{J} given by Eq. (6.86), whereas *manipulability* is defined as the square root of the determinant of the product of the manipulator Jacobian by its transpose. If the dexterity and manipulability are denoted with w_d and w_m , respectively, they can be expressed as follows:

$$w_d = \det(\mathbf{J}) \text{ and } w_m = \sqrt{\det(\mathbf{J}\mathbf{J}^T)} \quad (7.44)$$

For a nonredundant manipulator, the matrix \mathbf{J} is square. Hence, $w_m = |\det(\mathbf{J})|$, and $w_d = w_m$. From the kinetostatic point of view, a dexterous robot with good manipulability value would mean that the manipulator can reach all points within the

workspace with all possible orientations of its end-effector, and can transmit motion and force comfortably. The latter aspect can be better understood from the concept of velocity and force ellipsoids, as explained next. In other words, the

Condition Number

It is formally defined as the ratio of the maximum to minimum singular values of a matrix, and can measure a singularity or other kinetostatic indices.

force felt by the end-effector is distributed in contrast to a very high force in some directions whereas there is almost no force along other directions. The latter happens when the manipulator is in singularity.

7.7.4 Velocity and Force Ellipsoids

The kinematic relationship given by Eq. (6.62c) or (6.105a), combined with the statics relationship given by Eq. (7.32), points out a property of kinetostatics duality. The transformation characteristics of the joint rates required to produce a unity end-effector velocity in all possible directions can be obtained from the following condition:

$$\mathbf{t}_e^T \mathbf{t}_e = 1 \quad (7.45)$$

where \mathbf{t}_e is the twist of the end-effector comprising of its angular and linear velocities, as defined in Eq. (6.93). Substituting Eq. (6.62c), $\mathbf{t}_e = \mathbf{J}\dot{\theta}$, in Eq. (7.45), one obtains

$$\dot{\theta}^T \mathbf{J}^T \mathbf{J} \dot{\theta} = 1 \quad (7.46)$$

Equation (7.46) represents an ellipsoid in the n -dimensional joint space. This is illustrated in Fig. 7.10 with respect to a 2-DOF manipulator arm shown in Fig. 7.9.

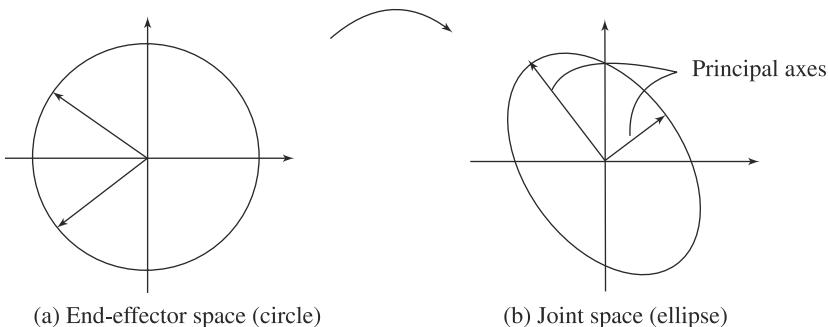


Fig. 7.10 Velocity ellipsoid

Since $\mathbf{J}^T \mathbf{J}$ is symmetric positive semi-definite, its eigenvectors are orthogonal. The principal axes of the ellipsoid coincide with the eigenvectors of $\mathbf{J}^T \mathbf{J}$, and the lengths of its principal axes are equal to the reciprocals of the square roots of the eigenvalues of $\mathbf{J}^T \mathbf{J}$. Since the Jacobian matrix is configuration dependent, the ellipsoid is also dependent on the configuration of the manipulator under study. As the end-effector moves from one configuration to another, the shape and orientation of the ellipsoid will also change. The closer the velocity ellipsoid to a sphere, the better the transformation characteristics are. If the lengths of principal axes are equal, the transformation is called isotropic, all joint-rates equally contribute to the equally distributed Cartesian velocities. At a singular point, lengths of one or more principal axes become infinitely long. As a result, the ellipsoid degenerates into a cylinder, and the end-effector will not be able to move in some directions.

Similar to the transformation of velocities, the transformation of forces for robots with only one type of joints and for one type of tasks can be characterized by a comparison of the end-effector force produced by a unit joint torque. Substituting Eq. (7.32) into the expression of $\boldsymbol{\tau}^T \boldsymbol{\tau} = 1$, one gets the following:

$$\mathbf{w}_e^T \mathbf{J} \mathbf{J}^T \mathbf{w}_e = 1 \quad (7.47)$$

At a given robot configuration, Eq. (7.47) represents the 6-dimensional ellipsoid for the 6-DOF manipulator, similar to Fig. 7.10 for the 2-DOF planar arm. As mentioned for the velocity ellipsoid, the closer the ellipsoid to a sphere, the better the transmission characteristics. This transformation is said to be isotropic when the

Isotropic

It is a word from Linear Algebra used to identify the condition number of a matrix as one.

principal axes are of equal lengths. At an isotropic point, an n -dimensional unit sphere in the joint torque space maps onto an 6-dimensional sphere in the end-effector force space. On the other hand, at a singular point, an n -dimensional unit

sphere in the joint torque space maps onto an 6-dimensional cylinder in the end-effector force space. Thus, the mechanical advantage of the manipulator becomes infinitely large in some direction.

7.8 STRUCTURAL MEASURES

Strength and stiffness of an individual link of a manipulator or the overall assembly play an important role in terms of accuracy, repeatability, vibration characteristics, and other phenomena of a robot manipulator. These are explained below.

7.8.1 Strength

Once the forces and moments acting on an individual link are obtained as reactions at the joints from the analyses of statics and dynamics, they can act as inputs to the design formulas for the determination of the link shapes and cross sections based on the choice of a material. While statics is covered in this chapter, dynamics will be taken up in Chapters 8 and 9. These are the topics of design of machine elements provided in the textbooks authored by Norton (2001), Shigley (2008), Bhandari (2010), and others. Some key points to remember are as follows:

- Robot links many times have to comply with contradictory constraints. For example, the links should be hollow to provide conduits for electric power and communication cables, hoses, power-transmitting elements like belts, chains, etc. At the same time, their external dimensions are limited in order to maximize usable workspace.
- Links should be lightweight to reduce inertia forces and allow for the largest payload per given size of actuators.
- Materials should be easily available.

7.8.2 Manipulator Stiffness

Choice of a link cross section is generally governed by its bending stiffness, as the links of a serial manipulator behave like a cantilever beam where the joints are considered as fixed supports. Since the links are driven by motors coupled with either gears or belts, chains, or cables, their flexibility substantially influence the overall stiffness of the assembled manipulator. In order to take care of such stiffness contributions, it is important to model the sub-assemblies either connected in series or parallel. If a shaft is connected to a motor through a gearbox, as depicted in Fig. 7.11, the overall stiffness of the subassembly is obtained as follows:

$$\frac{1}{k_e} = \frac{1}{\eta^2 k_1} + \frac{1}{k_2} \quad (7.48)$$

where k_e is the equivalent stiffness of the sub-assembly, η is the gear ratio defined as the ratio of the smaller to larger gear diameters, and k_1 and k_2 are the torsional stiffnesses of the input and output shafts, respectively. In case one knows the material, diameter and length of the shafts, k_1 and k_2 can be evaluated from the text books on Mechanical Design, e.g., Bhandari (2010) and others, as

$$k_i = \frac{G\pi d_i^4}{32l_i} \quad (7.49)$$

where G is the shear modulus of elasticity which is a material property. The value of G for steel is about 7.5×10^{10} N/m², whereas it is one-third as much for aluminum. The parameters d_i and l_i are the diameter and length of the i th shaft, respectively. For a given weight and desired accuracy, the bending and torsional stiffnesses of the links should be as high as possible.

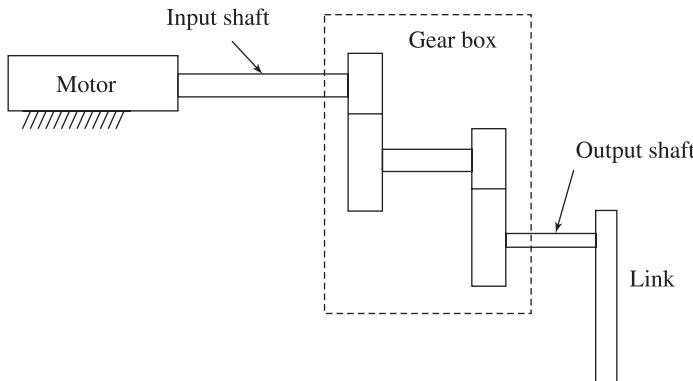


Fig. 7.11 Shaft assembly of a link

Example 7.9 Stiffness of a Drive System

Assume a steel input shaft of 25 mm diameter and 250 mm length connected to an output shaft of 10 mm diameter and 100 mm length. If the gear reduction ratio is 5, the net stiffness of the drive system is calculated as

$$\frac{1}{k_e} = \frac{1}{\eta^2 k_1} + \frac{1}{k_2} \quad (7.50a)$$

where $\eta = 5$, k_1 and k_2 are obtained from Eq. (7.49) as

$$k_1 = \frac{7.5 \times 10^{10} \pi (0.025)^4}{32(0.25)} = 11504.85 \frac{\text{Nm}}{\text{rad}} \quad (7.50b)$$

$$k_2 = \frac{7.5 \times 10^{10} \pi (0.01)^4}{32(0.1)} = 736.31 \frac{\text{Nm}}{\text{rad}} \quad (7.50c)$$

The effective stiffness k_e is then calculated as

$$k_e = \frac{\eta^2 k_1 + k_2}{\eta^2 k_1 k_2} = \frac{(25(11504.85) + 736.31)}{(25(11504.85)(736.31))} = 287621.25 \text{ Nm} \quad (7.50d)$$

7.9 DYNAMICS AND CONTROL MEASURES

In most industrial applications of robots, e.g., those explained in Chapter 2, the speed is not that sufficiently high to be affected by the robot's dynamics, i.e., mass, moment of inertia of the links, the payload, and other moving components. However, modern high-speed robots, e.g., those used in assembling PCBs, etc., or those using very thin lightweight links for space applications must consider the robot dynamics for their control.

Note that the robot elements like shafts, gears, driven links, etc., are often assumed rigid, whereas they have finite stiffnesses. If their flexibility is modeled the order of the equations of motion representing the dynamics of the robot would increase, thereby, complicating the control aspects of the robot at hand (see Chapters 10 and 11). If the robot links are assumed sufficiently rigid, the natural frequencies of the unmodelled resonances are very high that can be neglected compared to the influence of the dominant second-order poles. If the structural flexibilities are not modeled then one must be careful not to excite those frequencies through control gains. A rule of thumb is

$$\omega_n \leq \frac{1}{2} \omega_r \quad (7.51)$$

where ω_n is the closed-loop natural frequency, whereas ω_r is the lowest structural resonant frequency which can be evaluated from the knowledge of the stiffness k and the mass m of the system at hand, i.e., $\omega_r = \sqrt{k/m}$ for a simple mass-spring system. Note that for a rotational system, k and m are to be interpreted as torsional stiffness and moment of inertia, respectively.

Example 7.10 Closed-loop Natural Frequency

If a shaft carrying a mass is modeled as a massless spring attached to a mass of 10 kg with the radius of gyration equal to 250 mm, the resonant frequency ω_r is calculated as

$$\omega_r = \sqrt{\frac{736.31}{10 \times (0.25)^2}} = 34.32 \frac{\text{rad}}{\text{s}} = 215.66 \text{ Hz} \quad (7.52)$$

In Eq. (7.52), the value of torsional stiffness is taken from k_2 of Eq. (7.50c) and the moment of inertia is calculated as the multiplication of the mass and the square of radius of gyration. Now the closed-loop natural frequency should be as follows:

$$\omega_n < 107.83 \quad (7.53)$$

SUMMARY

Torques required to exert a force and a moment by the end-effector are derived in this chapter while the robot is in static equilibrium. It is shown how the velocity Jacobian comes into picture in static analysis. Physical interpretations of the Jacobian matrix in velocity and force domains were also provided besides different aspects of mechanical design of robot links.

EXERCISES

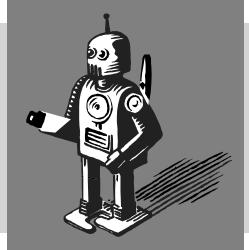
- 7.1** What is static equilibrium in robotics?
- 7.2** How are wrench and twist propagation matrices related?
- 7.3** Define virtual displacement and work.
- 7.4** How does velocity Jacobian matrix comes into picture in static analysis?
- 7.5** What would happen to the end-effector forces when certain joint torques are applied in singular configuration of the robot?
- 7.6** Extract the Jacobian matrix for the revolute-prismatic (RP) planar arm in Example 7.2.
- 7.7** Find the expression of the Jacobian matrix in the fixed frame from the static relations.
- 7.8** Why is the concept of velocity and force ellipsoids important in the study of robot performance?
- 7.9** What happens if one of the principal axes of the force ellipsoid vanishes?
- 7.10** Derive the torque and force expressions for the prismatic-revolute (PR) planar arm shown in Fig. 5.31 while the end-effector force \mathbf{f}_e is applied and no link weights are considered.
- 7.11** What are the three broad categories under whom the design parameters of a robot manipulator can be classified?
- 7.12** Define *dexterity* and *manipulability*.
- 7.13** Define *service index* and *structural length index*.
- 7.14** Justify the cantilever beam model of each link of a serial robot.
- 7.15** If the flexibility of robot links is ignored, what should be the requirement of closed-loop natural frequency?

MATLAB BASED EXERCISES

- 7.16** Verify the results of Exercise 7.6 using the symbolic toolbox of MATLAB.
- 7.17** Verify the results of Exercise 7.10.
- 7.18** Find the static torque expressions for the SCARA robot shown in Fig. 5.36.
- 7.19** Find out the torques for the anthropomorphic arm of Example 7.6 while $\theta_1 = \theta_2 = 0$ and $\theta_3 = \pi/2$.
- 7.20** What are the torque values for the spherical wrist shown in Fig. 5.38 while only a moment \mathbf{n}_e is applied at its end-effector?

8

Dynamics



In this chapter, the equations of motion for a robot, i.e., the way the motion of the robot arises due to torques and forces applied at the joints by the actuators, will be studied. A set of equations that describe the dynamical behavior of a robot, also referred as the dynamic model of the robot, will be developed. This development is important in several ways:

Dynamics

It is the study of forces and moments causing the motion in a system.

- (i) A dynamic model can be used to develop suitable control strategies. A sophisticated controller requires the use of a realistic dynamic model to achieve optimal performance of the robot under high-speed operations. Some control schemes rely directly on a dynamic model to compute actuator torques and forces required to follow a desired trajectory.
- (ii) The derived dynamic model can be used for computer simulation of a robotic system. By examining the behavior of the model under various operating conditions, it is possible to predict how a robot will behave when it will be built.
- (iii) The dynamic analysis of a robot gives all the joint reaction forces and moments needed for the design and sizing of the links, bearings, and actuators.

Given a robot, i.e., knowing its physical parameters, one normally wishes to solve two problems related to its dynamics. They are *inverse dynamics* and *forward* or *direct* dynamics. The inverse dynamics problem is to find the actuator torques and/or forces required to generate a desired trajectory of the robot's end-effector. This formulation of dynamics is useful for the problem of controlling the robot. The second problem, the forward dynamics, is required to find the response of the robot arm corresponding to some applied torques and/or forces at the joints. That is, given the joint torques and/or forces, compute the resulting motion of the robot as a function of time. This is not as critical as that for inverse dynamics since it is used primarily for computer simulation of a robot, which just shows how a robot will perform when it is built. On the other hand, an efficient inverse dynamics model becomes extremely important for the real-time control of robots.

The dynamic equations of motion can be formulated by several methods. One approach is to apply the Euler–Lagrange equations of motion. The advantage of employing this approach is that it eliminates the forces of constraints from the

dynamic equations of motion if the generalized coordinates are independently chosen. The elimination makes it suitable for motion control and simulation. However, these eliminated constraint forces can be recovered using Lagrange multiplies, if they are to be used for the purpose of design.

Another approach is the application of Newton and Euler laws for linear and rotational motions, respectively. Newton's and Euler's equations of motion for each link of the robot results in a system of equations that contain both the applied forces and the forces of constraints. These equations can then be solved simultaneously for all the forces, including those due to constraints which do not contribute to the motion of the links but required for the link design. If one is interested only with the motion simulation, the constraint forces can be eliminated using the geometric and kinematic relations describing the nature of constraints. There exist alternate methodologies to solve robot dynamics as well, e.g., D'Alembart principle, Kane's equations of motion, and those based on the orthogonal complements of the velocity-constraint matrices. One such orthogonal-complement-based method, namely, using the Decoupled Natural Orthogonal Complement (DeNOC) matrices (Saha, 1999), will be presented in Chapter 9.

Vector vs. Energy Approaches

Newton–Euler equations are known as vector approach, whereas the Euler–Lagrange equations are based on energy approach.

8.1 INERTIA PROPERTIES

In the study of dynamics, inertia properties like center of mass, moment of inertia, inertia tensor, etc., associated with the mass or inertia of individual links appearing in the dynamic equations of motion of a robot affect their behavior. It is important to know these different inertia properties for the dynamics study of a robot.

8.1.1 Center of Mass

Mass is a quantity of matter that forms the body of a certain shape and size. Referring to Fig. 8.1, F is a Cartesian coordinate frame, dV is a differential volume of the material body B , ρ is the material density, and \mathbf{p} is the position vector of the differential mass, ρdV , from the origin of frame F , i.e., O . The *center of mass* of such a material body B is then defined as the point C such that its position vector denoted by \mathbf{c} satisfies

$$\mathbf{c} \equiv \frac{1}{m} \int_V \mathbf{p} \rho dV \quad (8.1a)$$

where $m \equiv \int_V \rho dV$ is the total mass of the material body B . Moreover, if $\mathbf{p} = \mathbf{c} + \mathbf{r}$, as shown in Fig. 8.1, is substituted into Eq. (8.1a), one obtains the following:

$$\mathbf{c} \equiv \frac{1}{m} \int_V \mathbf{c} \rho dV + \frac{1}{m} \int_V \mathbf{r} \rho dV \quad (8.1b)$$

Mass vs. Moment of Inertia

Mass influences a linear motion directly, whereas it affects the rotational motion indirectly through the *moment of inertia*. Moment of inertia of an object is the property of its mass and geometric shape.

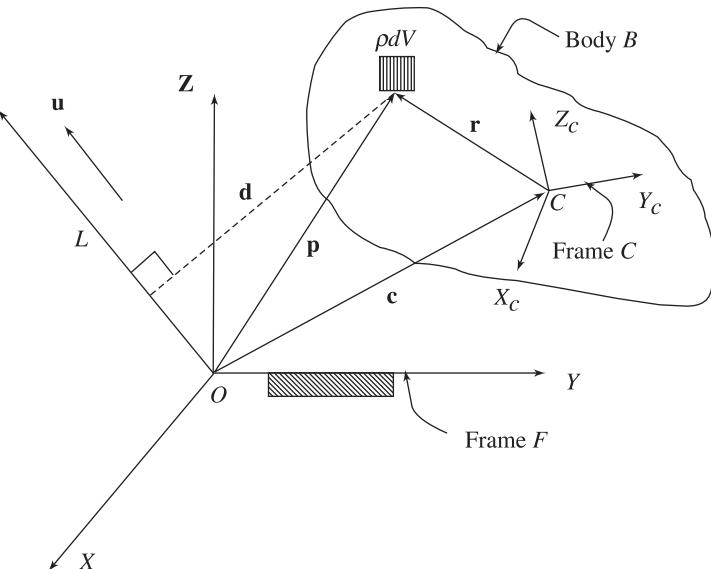


Fig. 8.1 Moments of mass

Since the vector \mathbf{c} is independent of the volume V of body B , it can be taken out of the integral sign. Furthermore, $\int_V \rho dV = m$. Hence, Eq. (8.1b) can be rewritten as

$$\mathbf{c} \equiv \mathbf{c} + \frac{1}{m} \int_V \mathbf{r} \rho dV \quad (8.1c)$$

which immediately gives

$$\int_V \mathbf{r} \rho dV = 0 \quad (8.1d)$$

The above property associated with the mass center of a rigid body will be used later in Section 8.3.2 for the derivation of angular momentum.

8.1.2 Moment of Inertia and Inertia Tensor

Referring to Fig. 8.1, let \mathbf{u} be the unit vector parallel to the line L passing through a reference point, namely, the origin O of the frame F . Moreover, if d is the distance of the differential mass ρdV of the body B from the line, L , as shown in Fig. 8.1 then the mass moment of inertia of the body B with respect to line L is given by a positive scalar, namely,

$$I_{uu} = \int_V d^2 \rho dV = \int_V \mathbf{d}^T \mathbf{d} \rho dV \quad (8.2)$$

where the vector \mathbf{d} is orthogonal to the unit vector \mathbf{u} indicated in Fig. 8.1, and d is its magnitude, which is equal to

$$d = |\mathbf{p} \times \mathbf{u}| \quad (8.3)$$

Substitution of Eq. (8.3) into Eq. (8.2) yields

$$\mathbf{I}_{uu} = \int_V |\mathbf{p} \times \mathbf{u}|^2 \rho dV = \int_V (\mathbf{p} \times \mathbf{u})^T (\mathbf{p} \times \mathbf{u}) \rho dV = \mathbf{u}^T \mathbf{I} \mathbf{u} \quad (8.4a)$$

where

$$(\mathbf{p} \times \mathbf{u})^T (\mathbf{p} \times \mathbf{u}) = (\mathbf{p}^T \mathbf{p})(\mathbf{u}^T \mathbf{u}) - (\mathbf{p}^T \mathbf{u})^2 = \mathbf{u}^T [(\mathbf{p}^T \mathbf{p}) \mathbf{1} - \mathbf{p} \mathbf{p}^T] \mathbf{u} \quad (8.4b)$$

$\mathbf{1}$ being the 3×3 identity matrix. Hence, the 3×3 matrix, \mathbf{I} of Eq. (8.4a), is given by

$$\mathbf{I} \equiv \int_V [(\mathbf{p}^T \mathbf{p}) \mathbf{1} - \mathbf{p} \mathbf{p}^T] \rho dV \quad (8.5)$$

The term \mathbf{I} is called *inertia tensor* or *inertia matrix* of the body B about the point O . The inertia tensor \mathbf{I} is symmetric and positive definite that can also be written as

$$\mathbf{I} \equiv \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \quad (8.6)$$

In Eq. (8.6), the expressions for the elements of \mathbf{I} are given by

$$I_{xx} \equiv \int_V (y^2 + z^2) \rho dV, I_{yy} \equiv \int_V (z^2 + x^2) \rho dV, I_{zz} \equiv \int_V (x^2 + y^2) \rho dV \quad (8.7a)$$

$$I_{xy} = I_{yx} \equiv - \int_V xy \rho dV, I_{yz} = I_{zy} \equiv - \int_V yz \rho dV, I_{xz} = I_{zx} \equiv - \int_V xz \rho dV \quad (8.7b)$$

where x , y , and z represent the coordinates of the differential mass ρdV with respect to the origin of frame F , i.e., O . Moreover, the subscripts, xx , yy , and zz denote the coordinate axes, X , Y , and Z of the frame F whose origin is located at O , respectively, with respect to which the moment of inertias, I_{xx} , I_{yy} , I_{zz} , and the product of inertias, I_{xy} , I_{yz} , I_{xz} , are calculated. The numerical values of these elements depend on the choice of a reference point and the coordinate frame at this point with respect to which they are calculated. While for a rigid body of simple geometry, the inertia terms are computed using the volumetric integration given by Eqs. (8.7a-b), the same need to be determined experimentally for the objects of irregular shapes.

Finally, note that in Eq. (8.2), the distance d or vector \mathbf{d} is independent of the volume V of the body B , as seen for the vector \mathbf{c} after Eq. (8.1b). Hence, the term d can be taken out of the integral sign. As a result, Eq. (8.2) is rewritten as follows:

$$I_{uu} = d^2 \int_V \rho dV = md^2 \quad (8.8)$$

where the scalar d has an expression in terms of vectors \mathbf{p} and \mathbf{u} as $\mathbf{d} \equiv |\mathbf{p} \times \mathbf{u}|$, which is a non-negative real quantity. The scalar d is referred as the *radius of gyration* of the body B with respect to the line L .

Tensor and Its Orders

In physics, tensors characterize the properties of a physical system. For example, a tensor of order zero is a *scalar* representing, say, mass of a body. A tensor of order one is a *vector* that may represent the mass center of the body, whereas a second-order tensor is represented by a 3×3 matrix denoting, for example, the so-called inertia matrix (or tensor) of the body.

8.1.3 Parallel-Axis Theorem

Let the frame C be the Cartesian coordinate frame attached to the center of mass C (x_c, y_c, z_c) of a rigid body B with its coordinate axes parallel to those of F , as shown in Fig. 8.1. Then it can be shown that

$$I_{xx} = I_{xx}^c + m(y_c^2 + z_c^2), \quad I_{yy} = I_{yy}^c + m(z_c^2 + x_c^2), \quad I_{zz} = I_{zz}^c + m(x_c^2 + y_c^2) \quad (8.9a)$$

$$I_{xy} = I_{xy}^c - mx_c y_c, \quad I_{yz} = I_{yz}^c - my_c z_c, \quad I_{zx} = I_{zx}^c - mz_c x_c \quad (8.9b)$$

where x_c, y_c , and z_c are the coordinates of the center of mass in the frame F . Equations (8.9a-b) are called the parallel-axis theorem.

8.1.4 Perpendicular-Axis Theorem

For a planar object, say, the thin disk shown in Fig. 8.2, the moment of inertia about an axis perpendicular to the plane, I_{zz} , is the sum of the moments of inertia of two perpendicular axes through the same point in the plane of the object, I_{xx} and I_{yy} , i.e.,

$$I_{zz} = I_{xx} + I_{yy} \quad (8.10)$$

The utility of this theorem goes beyond that of calculation of moments of strictly planar objects. It is a valuable tool in the building up of the moments of inertia of three-dimensional objects such as cylinders by breaking them up into planar disks and summing the moments of inertia of the composite disks, as done in Example 8.3.

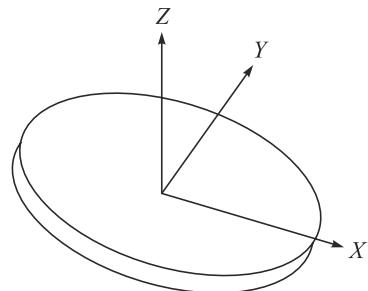


Fig. 8.2 A thin disk

8.1.5 Principal Moments of Inertia

It is shown that the inertia matrix depends on the choice of a reference point and the orientation of the reference frame. It turns out that for a certain orientation of a reference frame, the products of inertia will vanish. These special coordinate axes are called the *principal axes*, and the corresponding moments of inertia are called the *principal moments of inertia*. Let \mathbf{I} be the inertia tensor of a rigid body B about a point O expressed in the reference frame F . Also, let L be the principal axis that passes through the origin O and points in the direction of \mathbf{u} . By definition, \mathbf{u} is parallel to the vector of the second moment of B about L . That is,

$$\mathbf{I}\mathbf{u} = \lambda\mathbf{u} \quad (8.11)$$

Equation (8.11) contains three linear homogeneous equations in three unknowns, namely, u_x , u_y , and u_z . The condition for existence of nontrivial solutions is

$$\begin{vmatrix} I_{xx} - \lambda & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} - \lambda & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} - \lambda \end{vmatrix} = 0 \quad (8.12)$$

Hence, the eigenvalues and eigenvectors of the inertia matrix \mathbf{I} correspond to the principal moments of inertia, and the directions of principal axes, respectively. In general, corresponding to each reference point, there exists at least one set of three

mutually perpendicular principal axes of inertia. Additional facts about inertia matrix are as follows:

- (i) If two axes of the reference frame form a plane of symmetry for the mass distribution of the body, the products of inertia having an index as the coordinate which is normal to the plane of symmetry will be zero.
- (ii) Moments of inertia must always be positive. Products of inertia may have either sign.
- (iii) The sum of the three moments of inertia are invariant under orientation changes in the reference frame.
- (iv) The sum of any two moment of inertia is greater than the third one, i.e., $I_{xx} + I_{yy} > I_{zz}$.

Example 8.1 Inertia Tensor of a Box

The inertia tensor for the rectangular body of uniform density ρ with respect to the rectangular coordinate system at one corner, as shown in Fig. 8.3, is obtained as follows: First, I_{xx} is computed. Using the volume element at (x, y, z) , $dV = dx dy dz$, one gets

$$\begin{aligned} I_{xx} &= \int_0^h \int_0^l \int_0^w (y^2 + z^2) \rho dx dy dz = \int_0^h \int_0^l (y^2 + z^2) w \rho dy dz = \int_0^h \left(\frac{l^3}{3} + z^2 l \right) w \rho dz \\ &= \left(\frac{hl^3 w}{3} + \frac{h^3 lw}{3} \right) \rho = \frac{hlw\rho}{3} (l^2 + h^2) = \frac{m}{3} (l^2 + h^2) \end{aligned} \quad (8.13)$$

where $m \equiv hlw\rho$ is the total mass of the body. Similarly, one can get I_{yy} and I_{zz} by inspection, i.e.,

$$I_{yy} = \frac{m}{3} (w^2 + h^2); \text{ and } I_{zz} = \frac{m}{3} (l^2 + w^2) \quad (8.14)$$

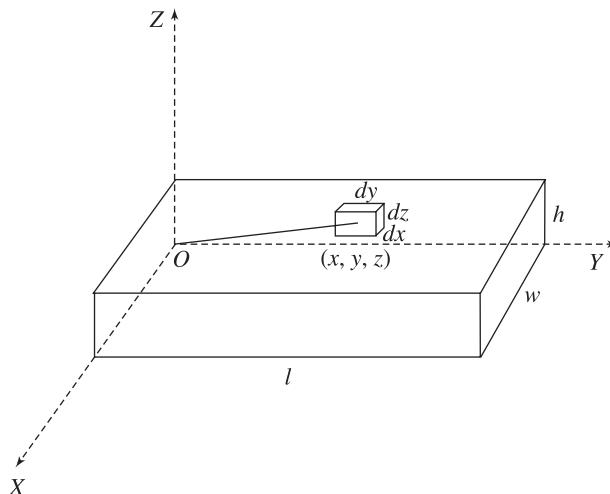


Fig. 8.3 A body of uniform density

Next, I_{xy} is computed as

$$I_{xy} = - \int_0^h \int_0^l \int_0^w xy \rho dx dy dz = - \int_0^h \int_0^l \frac{w^2}{2} y \rho dy dz = - \int_0^h \frac{w^2 l^2}{4} \rho dz = - \frac{m}{4} wl \quad (8.15)$$

Permuting the terms, one gets

$$I_{xz} = - \frac{m}{4} hw; \text{ and } I_{yz} = - \frac{m}{4} hl \quad (8.16)$$

Hence, the inertia tensor for this object is

$$\mathbf{I} = \begin{bmatrix} \frac{m}{3}(l^2 + h^2) & -\frac{m}{4}wl & -\frac{m}{4}hw \\ -\frac{m}{4}wl & \frac{m}{3}(w^2 + h^2) & -\frac{m}{4}hl \\ -\frac{m}{4}hw & -\frac{m}{4}hw & \frac{m}{3}(l^2 + w^2) \end{bmatrix} \quad (8.17)$$

Note that the 3×3 tensor, represented by matrix \mathbf{I} of Eq. (8.17), is symmetric.

Example 8.2 Inertia Tensor of a Box about its Mass Center

The inertia tensor for the same solid body described in Example 8.1 is obtained in a coordinate system whose origin is at the body's center of mass C . Note here that one can apply the parallel-axis theorem in Eq. (8.9), where $x_c = w/2$, $y_c = l/2$, and $z_c = h/2$. Then,

$$I_{zz}^c = \frac{m}{12} (w^2 + l^2), \text{ and } I_{xy}^c = 0 \quad (8.18a)$$

The other elements are found by symmetry. The resulting inertia tensor written in the frame located at the center of mass C is given by

$$\mathbf{I}^c = \frac{m}{12} \begin{bmatrix} h^2 + l^2 & 0 & 0 \\ 0 & w^2 + h^2 & 0 \\ 0 & 0 & l^2 + w^2 \end{bmatrix} \quad (8.18b)$$

Since the result is diagonal, the frame at C must represent the principal axes of this body. For a slender rod, w and h are much smaller than l . Hence, its inertia tensor can be approximated by

$$\mathbf{I}^c = \frac{ml^2}{12} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8.19)$$

Example 8.3 Inertia Tensor of a Cylinder

A cylinder of diameter d and length l shown in Fig. 8.4(a) is a very common shape used as robot links. Similar to Eq. (8.7a), the I_{xx} component of the cylinder is computed as

$$I_{xx} = \int_V r^2 \rho dV = \int_0^{d/2} r^2 \rho l 2\pi r dr = \frac{d^4}{32} \rho l \pi = \frac{m d^2}{8} \quad (8.20)$$

where r is the radial distance from the axis of the cylinder, i.e., X -axis, where the elemental volume of the cylinder lies. Moreover, $m \equiv \rho l \pi d^2/4$ is the total mass of the cylinder. The expression for the moment of inertia of a cylinder about Y - or Z -axis, i.e., a diameter at its end, can be obtained using both the parallel- and perpendicular-axis theorems. The approach involves finding an expression for a thin disk at distance x from the axis Y or Z , as shown in Fig. 8.4(b), and summing over all such disks. For an infinitesimally thin disk of thickness dx , the moment of inertia about the central axis, i.e., X -axis, is given using Eq. (8.20) as

$$dI_{xx} = \frac{1}{8} d^2 dm \quad (8.21a)$$

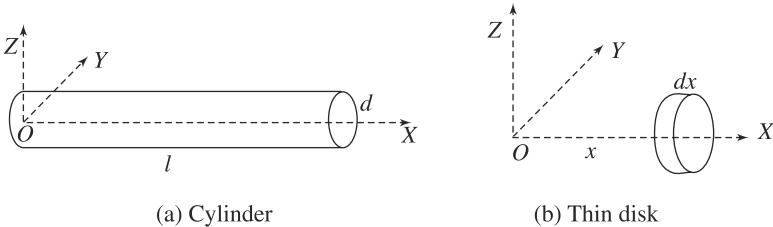


Fig. 8.4 A cylindrical object

But by the perpendicular theorem,

$$dI_{xx} = dI_{yy} + dI_{zz} \quad (8.21b)$$

Since the moment of inertia about Y - and Z -axis must be equal, by symmetry it follows that

$$dI_{yy} = \frac{1}{2} dI_{xx} = \frac{1}{16} d^2 dm \quad (8.21c)$$

Obtaining the moment of inertia of the full cylinder about a diameter at its end involves summing over an infinite number of thin disks at different distances from that axis, i.e., Y or Z . This involves an integral from $x = 0$ to $x = l$. For any given disk at distance x from say Y -axis, using the parallel-axis theorem, gives the moment of inertia about the Y -axis, i.e.,

$$dI_{yy} = \frac{1}{16} d^2 dm + x^2 dm \quad (8.21d)$$

Now expressing the mass element dm in terms of dx , one can integrate over the length of the cylinder, namely,

$$I_{yy} = \int_0^l dI_{yy} = \frac{d^2 \rho A}{16} \int_0^l dx + \rho A \int_0^l x^2 dx = \frac{1}{16} m d^2 + \frac{1}{3} m l^2 \quad (8.21e)$$

This above can be seen to be plausible if one notices that Eq. (8.21e) is the sum of the expressions for a thin disk about a diameter plus the expression for a thin rod about its end. The 3×3 inertia tensor \mathbf{I} for the cylinder is given by

$$\mathbf{I} = \begin{bmatrix} \frac{md^2}{16} & 0 & 0 \\ 0 & \frac{md^2}{16} + \frac{ml^2}{3} & 0 \\ 0 & 0 & \frac{md^2}{16} + \frac{ml^2}{3} \end{bmatrix} \quad (8.22a)$$

where the product of inertia terms I_{xy} , I_{xz} , and I_{yz} can be proven to be zeros as the cylinder is symmetrically placed about Y and Z axes. In the limiting case when $d \approx 0$, one gets the thin rod expression, and when $l \approx 0$, the expression for the disk is obtained. Matrix \mathbf{I} of Eq. (8.22a) is then simplified as

$$\mathbf{I} = \frac{ml^2}{3} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ for a thin cylindrical rod} \quad (8.22b)$$

$$\mathbf{I} = \frac{md^2}{16} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ for a thin disk} \quad (8.22c)$$

8.2 EULER–LAGRANGE FORMULATION

The dynamic model of a robot can be derived in a systematic way using the concept of generalized coordinates and a scalar function called *Lagrangian*. The *Lagrangian* is defined as the difference between the kinetic and potential energies of the mechanical system under study, i.e.,

$$L = T - U \quad (8.23)$$

where L denotes the Lagrangian, and T and U are respectively the total kinetic and potential energies of the system at hand. Note that the kinetic energy depends on both configuration, i.e., position and orientation, and the velocity of the links of a robotic system, whereas the potential energy depends only on the configuration of the links. Euler–Lagrange equations of motion are then given by

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \phi_i, \quad \text{for } i = 1, \dots, n \quad (8.24)$$

where n is the number of independent generalized coordinates used to define the system's configuration, and q_i 's and ϕ_i 's are the *generalized coordinates* and *generalized forces* due to applied forces corresponding to the generalized coordinates, respectively.

Note that Eq. (8.24) can be obtained from several fundamental principles. For example, *Principle of Stationary Action* was used in Meirovitch (1970), whereas Spong and Vidyasagar (2004) used the *Principle of Virtual Work* to derive the same. The concepts of virtual displacement and virtual work were introduced in the previous chapter to correlate static forces and moments. Here, in dynamics, it can be used in association with D'Alembert's principle which states that if a fictitious additional

force (also called as inertia force in the literature) equals the negative of the product of mass (or inertia) and acceleration then the system will be in equilibrium.

8.2.1 Generalized Coordinates

The coordinates that specify the configuration, i.e., the position and orientation, of all the bodies or links of a mechanical system completely are called generalized coordinates. Assume that the configuration of a mechanical system is known completely. That is the position and orientation of all links in the system are known. Since a rigid link has six degrees of freedom (DOF), a mechanical system with m moving links requires $6m$ coordinates to specify its configuration completely in the three-dimensional Cartesian space. These coordinates are referred the generalized coordinates of a mechanical system, e.g., a robot manipulator. However, these links cannot move freely as the joints put restrictions. Thus, the links are subjected to constraints imposed by the joints. As a result, the $6m$ coordinates are no longer independent. If there exist c constraints then $n = 6m - c$ coordinates can be specified independently, which are called independent generalized coordinates, and the system has n DOF. Hence, the number of independent generalized coordinates is equal to the number of DOF of the robot.

It is pointed out here that the generalized coordinates can have several representations. For example, consider the two-link planar robot arm shown in Fig. 8.5. Since a rigid link on a plane has 3-DOF, two links require six coordinates, namely, (x_1, y_1, θ_1) and (x_2, y_2, θ_2) , which are not independent as there are two revolute joints that restrict the motion of the two bodies. Note that the coordinates (x_1, y_1) and (x_2, y_2) define the mass center positions of the links, whereas θ_1 and θ_2 denote the orientation of the links. Moreover, d_1 and d_2 are the mass center locations from the origin of the frames. Furthermore, for the first set of six coordinates there are four constraints, namely,

$$x_1 = d_1 \cos \theta_1; y_1 = d_1 \sin \theta_1 \quad (8.25a)$$

$$x_2 = a_1 \cos \theta_1 + d_2 \cos \theta_{12}; y_2 = a_1 \sin \theta_1 + d_2 \sin \theta_{12} \quad (8.25b)$$

where $\theta_{12} \equiv \theta_1 + \theta_2$. Hence, using Eqs. (8.25a-b), the system has $6 - 4 = 2$ DOF. Then, the independent set of generalized coordinates can be θ_1 and θ_2 . One can of course choose any other two independent generalized coordinates. However, for the above example, θ_1 and θ_2 are the two most convenient and popular coordinates used in robotics literature for robot kinematic and dynamic analyses, as they represent the controlling motions driven typically by electric motors.

8.2.2 Kinetic Energy

Consider a robot consisting of n rigid links, as shown in Fig. 8.6(a). Then, the kinetic energy of a typical link i shown in Fig. 8.6(b), denoted as T_i , is given by

$$T_i = \frac{1}{2} m_i \dot{\mathbf{c}}_i^T \dot{\mathbf{c}}_i + \frac{1}{2} \boldsymbol{\omega}_i^T \mathbf{I}_i \boldsymbol{\omega}_i \quad (8.26)$$

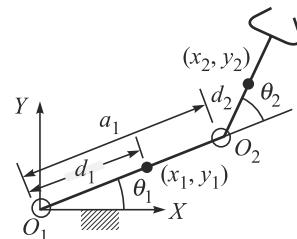


Fig. 8.5 A two-link robot arm

where

$\dot{\mathbf{c}}_i$: Three-dimensional velocity (linear) vector of the mass center C_i of the i th link

$\boldsymbol{\omega}_i$: Three-dimensional angular velocity vector of the i th link

m_i : Mass of the i th link (a scalar quantity)

\mathbf{I}_i : The 3×3 inertia tensor or matrix of the i th link about C_i , as defined in Section 8.1.

In Eq. (8.26), the inertia tensor \mathbf{I}_i for the i th body is time invariant, i.e., constant. However, in reality, its representation varies with the robot's configurations. In a different reference frame, say, the fixed frame, the tensor \mathbf{I}_i can be obtained as

$$[\mathbf{I}_i]_F = \mathbf{Q}_i [\mathbf{I}_i]_{i+1} \mathbf{Q}_i^T \quad (8.27)$$

where $[\mathbf{I}_i]_{i+1}$ is the inertia matrix of the i th link about C_i but represented in the moving link frame $i + 1$ that is attached to the i th body, and \mathbf{Q}_i is the 3×3 rotation matrix of link i or frame $i + 1$ with respect to the fixed frame 1. The total kinetic energy T is now given by the sum of the contributions of each rigid link due to the relative motions, i.e.,

$$T = \sum_{i=1}^n T_i = \sum_{i=1}^n \frac{1}{2} (m_i \dot{\mathbf{c}}_i^T \dot{\mathbf{c}}_i + \boldsymbol{\omega}_i^T \mathbf{I}_i \boldsymbol{\omega}_i) \quad (8.28)$$

At this point, it is necessary to express the kinetic energy as a function of the generalized coordinates of the system. For a robot, the joint variables, i.e., the angles of the revolute joints and the displacements of the prismatic joints, completely specify a robot's configuration. Hence, the joint angles and displacements are taken as the generalized coordinates. For the i th link, the angular velocity and the linear velocity are calculated from the first link of the serial-chain, Fig. 8.6(a), as

$$\boldsymbol{\omega}_1 = \dot{\theta}_1 \mathbf{e}_1 \quad (8.29a)$$

$$\dot{\mathbf{c}}_1 = \boldsymbol{\omega}_1 \times \mathbf{d}_1 = \mathbf{e}_1 \times \mathbf{d}_1 \dot{\theta}_1 \quad (8.29b)$$

$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_1 + \dot{\theta}_2 \mathbf{e}_2 = \mathbf{e}_1 \dot{\theta}_1 + \mathbf{e}_2 \dot{\theta}_2 \quad (8.29c)$$

$$\dot{\mathbf{c}}_2 = \dot{\mathbf{c}}_1 + \boldsymbol{\omega}_1 \times \mathbf{r}_1 + \boldsymbol{\omega}_2 \times \mathbf{d}_2 = \mathbf{e}_1 \times (\mathbf{a}_1 + \mathbf{d}_2) \dot{\theta}_1 + \mathbf{e}_2 \times \mathbf{d}_2 \dot{\theta}_2 \quad (8.29d)$$

$$\vdots \quad \vdots$$

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i = \mathbf{e}_1 \dot{\theta}_1 + \mathbf{e}_2 \dot{\theta}_2 + \cdots + \mathbf{e}_i \dot{\theta}_i \quad (8.29e)$$

$$\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{d}_i = \mathbf{e}_1 \times \mathbf{p}_{1i} \dot{\theta}_1 + \cdots + \mathbf{e}_i \times \mathbf{p}_{ii} \dot{\theta}_i \quad (8.29f)$$

Note that in writing Eqs. (8.29a-f), all joints were assumed revolute. Accordingly, the vector \mathbf{e}_i is the unit vector parallel to the axis of the i th revolute joint, whereas \mathbf{a}_i , \mathbf{d}_i , and \mathbf{r}_i are shown in Fig. 8.6(b). Moreover, \mathbf{p}_{ij} is the vector joining the origin of the i th joint O_i to the j th mass center C_j . For $i = j$, $\mathbf{p}_{ii} \equiv \mathbf{d}_i$. Using Eqs. (8.29e-f), the velocities of the i th link can now be expressed in terms of all the n joint rates as

$$\boldsymbol{\omega}_i = \mathbf{J}_{\omega,i} \dot{\boldsymbol{\theta}}, \text{ where } \mathbf{J}_{\omega,i} \equiv [\mathbf{j}_{\omega,i}^{(1)} \mathbf{j}_{\omega,i}^{(2)} \cdots \mathbf{j}_{\omega,i}^{(j)} \mathbf{0} \cdots \mathbf{0}] \quad (8.30a)$$

$$\dot{\mathbf{c}}_i = \mathbf{J}_{c,i} \dot{\boldsymbol{\theta}}, \text{ where } \mathbf{J}_{c,i} \equiv [\mathbf{j}_{c,i}^{(1)} \mathbf{j}_{c,i}^{(2)} \cdots \mathbf{j}_{c,i}^{(j)} \mathbf{0} \cdots \mathbf{0}] \quad (8.30b)$$

In Eqs. (8.30a-b), $\dot{\boldsymbol{\theta}} \equiv [\dot{\theta}_1, \dot{\theta}_2, \dots, \dot{\theta}_n]^T$ is the n -dimensional joint-rate vector. Moreover, $\mathbf{J}_{\omega,i}$ and $\mathbf{J}_{c,i}$ are the $3 \times n$ matrices. Correspondingly, $\mathbf{j}_{\omega,i}^{(j)}$ and $\mathbf{j}_{c,i}^{(j)}$ are the 3-dimensional vectors which can be written from Eqs. (8.29e-f) as

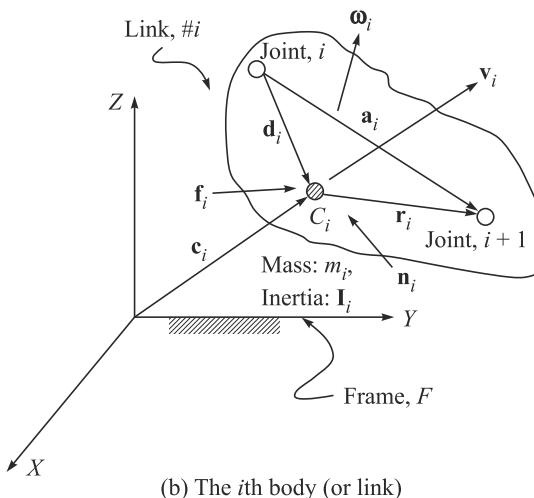
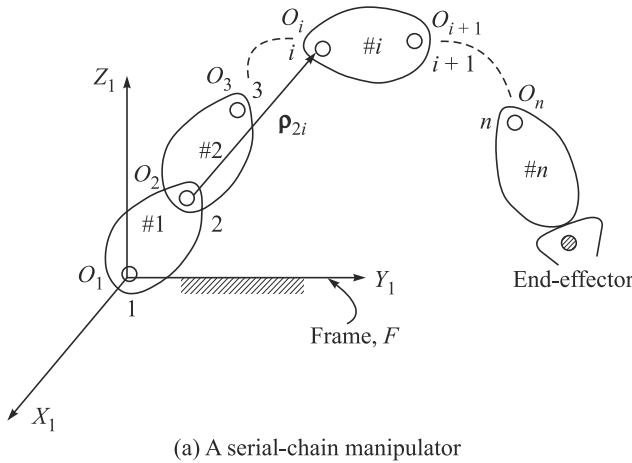


Fig. 8.6 A serial-chain robot

$$\mathbf{j}_{\omega,i}^{(j)} \equiv \mathbf{e}_j; \text{ and } \mathbf{j}_{c,i}^{(j)} \equiv \mathbf{e}_j \times \boldsymbol{\rho}_{ij} \quad (8.31a)$$

Note, for a prismatic joint, Eq. (8.31a) is modified as

$$\mathbf{j}_{\omega,i}^{(j)} \equiv \mathbf{0}; \text{ and } \mathbf{j}_{c,i}^{(j)} \equiv \mathbf{e}_j \quad (8.31b)$$

where $\mathbf{0}$ represents the 3-dimensional vector of zeros, and \mathbf{e}_j is the unit vector along the translation of the j th prismatic joint. Substituting Eqs. (8.31a-b) into Eq. (8.28), and then summing over all links, the expression for the resulting kinetic energy of the system is obtained as

$$T = \frac{1}{2} \sum_{i=1}^n (m_i \dot{\mathbf{c}}_i^T \dot{\mathbf{c}}_i + \boldsymbol{\omega}_i^T \mathbf{I}_i \boldsymbol{\omega}_i) = \frac{1}{2} \sum_1^n \dot{\boldsymbol{\theta}}^T \bar{\mathbf{I}}_i \dot{\boldsymbol{\theta}} \quad (8.32a)$$

where the $n \times n$ matrix $\bar{\mathbf{I}}_i$ is given by

$$\bar{\mathbf{I}}_i = m_i \mathbf{J}_{c,i}^T \mathbf{J}_{c,i} + \mathbf{J}_{\omega,i}^T \mathbf{I}_i \mathbf{J}_{\omega,i} \quad (8.32b)$$

In Eq. (8.32b), the expressions $\mathbf{J}_{c,i}^T \mathbf{J}_{c,i}$ and $\mathbf{J}_{\omega,i}^T \mathbf{I}_i \mathbf{J}_{\omega,i}$ are the $n \times n$ matrices. Moreover, if the $n \times n$ matrix \mathbf{I} is defined as

$$\mathbf{I} = \sum_{i=1}^n \bar{\mathbf{I}}_i \quad (8.33)$$

then the total kinetic energy can be rewritten from Eq. (8.32a) as

$$T = \frac{1}{2} \dot{\boldsymbol{\theta}}^T \mathbf{I} \dot{\boldsymbol{\theta}} \quad (8.34)$$

where the matrix \mathbf{I} is called the *Generalized Inertia Matrix* (GIM) of the robot at hand. It is pointed out here that the GIM of the robot, \mathbf{I} of Eq. (8.33), involves the matrices $\bar{\mathbf{I}}_i$ which in turn is a function of the matrices $\mathbf{J}_{c,i}$ and $\mathbf{J}_{\omega,i}$, as evident from Eq. (8.32b). Therefore, the robot manipulator's GIM is configuration dependent, i.e., the function of $\boldsymbol{\theta}$ only. Also, similar to the inertia of a rigid body \mathbf{I}_i , as defined in Eqs. (8.5) and (8.6), the manipulator GIM \mathbf{I} is also symmetric and positive-definite. This is obvious from the quadratic form of Eq. (8.34), which indicates that the kinetic energy of the system is always positive unless it is at rest.

8.2.3 Potential Energy

As done for the kinetic energy, the total potential energy stored in a robot is given by the sum of the contributions of each link. On the assumption of rigid links, the potential energy stored in link i is defined as the amount of work required to raise the center of mass of link i from a horizontal reference plane to its present position under the influence of gravity. With reference to the inertia frame, the work required to displace link i to the position C_i is given by $-m_i \mathbf{c}_i^T \mathbf{g}$, where \mathbf{g} is the vector due to gravity acceleration. Hence, the total potential energy stored in a robot manipulator is given by

$$U = - \sum_{i=1}^n m_i \mathbf{c}_i^T \mathbf{g} \quad (8.35)$$

In Eq. (8.35), the vector \mathbf{c}_i is a function of joint variables, i.e., θ_i 's of the robot. Hence, the total potential energy is a function of joint variables $\boldsymbol{\theta}$ only, and not of the joint velocities $\dot{\boldsymbol{\theta}}$. Moreover, \mathbf{c}_i is a nonlinear function of $\boldsymbol{\theta}$, and, therefore, U cannot be explicitly conveniently expressed in terms of $\boldsymbol{\theta}$.

8.2.4 Equations of Motion

Having computed the total kinetic and potential energies of the robot at hand, Eqs. (8.34) and (8.35), respectively, the Lagrangian of Eq. (8.23) can be written as

$$L = T - U = \sum_{i=1}^n \left[\frac{1}{2} \dot{\boldsymbol{\theta}}^T \bar{\mathbf{I}}_i \dot{\boldsymbol{\theta}} + m_i \mathbf{c}_i^T \mathbf{g} \right] \quad (8.36)$$

Next, the Lagrangian function is differentiated with respect to θ_i , $\dot{\theta}_i$, and t to obtain the dynamic equations of motion. To facilitate the derivations, the kinetic energy term is expanded as a sum of scalar terms. Let i_{ij} be the (i,j) element of the robot's GIM \mathbf{I} then Eq. (8.36) can be written as

$$L = \sum_{i=1}^n \left[\sum_{j=1}^n \frac{1}{2} i_{ij} \dot{\theta}_i \dot{\theta}_j + m_i \mathbf{c}_i^T \mathbf{g} \right] \quad (8.37)$$

Since the potential energy does not depend on $\dot{\theta}_i$, taking the partial derivative of L given by Eq. (8.37) with respect to $\dot{\theta}_i$ yields,

$$\frac{\partial L}{\partial \dot{\theta}_i} = \frac{\partial T}{\partial \dot{\theta}_i} = \sum_{j=1}^n i_{ij} \dot{\theta}_j \quad (8.38a)$$

for $i = 1, \dots, n$. Note that using the definition of $\dot{\theta}$ after Eq. (8.30b) and the GIM \mathbf{I} of Eq. (8.33), Eq. (8.38a) can be rewritten in a compact form as

$$\frac{\partial L}{\partial \dot{\theta}} = \mathbf{I} \dot{\theta} \quad (8.38b)$$

Equation (8.38a) is differentiated next with respect to time t as

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_i} \right) = \sum_{j=1}^n \left[i_{ij} \ddot{\theta}_j + \left(\frac{di_{ij}}{dt} \right) \dot{\theta}_j \right] = \sum_{j=1}^n \left[i_{ij} \ddot{\theta}_j + \sum_{k=1}^n \frac{\partial i_{ij}}{\partial \theta_k} \dot{\theta}_j \dot{\theta}_k \right] \quad (8.39a)$$

In a matrix-vector form, one can express Eq. (8.39a) from the time derivative of Eq. (8.38b) as

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) = \mathbf{I} \ddot{\theta} + \dot{\mathbf{I}} \dot{\theta}, \quad \text{where } \dot{\mathbf{I}} \equiv \frac{\partial \mathbf{I}}{\partial \theta_1} \dot{\theta}_1 + \dots + \frac{\partial \mathbf{I}}{\partial \theta_n} \dot{\theta}_n \quad (8.39b)$$

In Eq. (8.39b), $\ddot{\theta}$ is the time-derivative of $\dot{\theta}$, or double time derivative of θ , which is defined as the vector of joint accelerations. Moreover, $\dot{\mathbf{I}}$ is the time-derivative of the GIM \mathbf{I} . Taking the partial derivative of Eq. (8.37) with respect to θ_i gives

$$\frac{\partial L}{\partial \theta_i} = \frac{1}{2} \frac{\partial}{\partial \theta_i} \left(\sum_{j=1}^n \sum_{k=1}^n i_{ij} \dot{\theta}_j \dot{\theta}_k \right) + \sum_{j=1}^n m_j \left(\frac{\partial \mathbf{c}_j}{\partial \theta_i} \right)^T \mathbf{g} \quad (8.40a)$$

Since the partial derivative of \mathbf{c}_j with respect to θ_i , i.e., $\partial \mathbf{c}_j / \partial \theta_i$, is nothing but the i th column vector of the Jacobian $\mathbf{J}_{c,j}$, which is similar to Eq. (8.30b) and denoted with $\mathbf{j}_{c,j}^{(i)}$, Eq. (8.40a) can be written as

$$\frac{\partial L}{\partial \theta_i} = \frac{1}{2} \frac{\partial}{\partial \theta_i} \left(\sum_{j=1}^n \sum_{k=1}^n i_{ij} \dot{\theta}_j \dot{\theta}_k \right) + \sum_{j=1}^n m_j [\mathbf{j}_{c,j}^{(i)}]^T \mathbf{g} \quad (8.40b)$$

Similar to the derivation of Eqs. (8.38b) and (8.39b), Eq. (8.40b) can also be expressed in a compact form as

$$\frac{\partial L}{\partial \theta} = \frac{1}{2} \begin{bmatrix} \dot{\theta}^T (\partial \mathbf{I} / \partial \theta_1) \dot{\theta} \\ \vdots \\ \dot{\theta}^T (\partial \mathbf{I} / \partial \theta_n) \dot{\theta} \end{bmatrix} + \mathbf{J}_c^T \mathbf{m}, \quad \text{where } \mathbf{J}_c \equiv \begin{bmatrix} \mathbf{J}_{c,1} \\ \vdots \\ \mathbf{J}_{c,n} \end{bmatrix} \text{ and } \mathbf{m} \equiv \begin{bmatrix} m_1 \mathbf{g} \\ \vdots \\ m_n \mathbf{g} \end{bmatrix} \quad (8.41)$$

In Eq. (8.41), \mathbf{J}_c and \mathbf{m} are the $3n \times n$ matrix and $3n$ -dimensional vector, respectively. Combining Eqs. (8.38a), (8.39a) and (8.40b), the dynamic equations of motion are obtained as

$$\sum_{j=1}^n i_{ij} \dot{\theta}_j + h_i + \gamma_i = \tau_i \quad (8.42)$$

for $i = 1, \dots, n$, where

$$h_i \equiv \sum_{j=1}^n \sum_{k=1}^n \left(\frac{\partial i_{ij}}{\partial \theta_k} - \frac{1}{2} \frac{\partial i_{jk}}{\partial \theta_i} \right) \dot{\theta}_j \dot{\theta}_k \quad (8.43a)$$

and

$$\gamma_i \equiv - \sum_{j=1}^n [\mathbf{j}_{c,j}^{(i)}]^T (\mathbf{m}_j \mathbf{g}) \quad (8.43b)$$

Writing Eq. (8.42) for all the n generalized coordinates, the equations of motion can be written in a compact form as

$$\mathbf{I}\ddot{\boldsymbol{\theta}} + \mathbf{h} + \boldsymbol{\gamma} = \boldsymbol{\tau} \quad (8.44a)$$

where the $n \times n$ generalized inertia matrix \mathbf{I} is defined in Eq. (8.33) and the n -dimensional vectors, \mathbf{h} , $\boldsymbol{\gamma}$, and $\boldsymbol{\tau}$ are defined below:

$\mathbf{h} \equiv [h_1, \dots, h_n]^T$: the n -dimensional vector of centrifugal and Coriolis forces. Using the time-derivative of the GIM, i.e., $\dot{\mathbf{I}}$ of Eq. (8.39b), and taking the first term of the right hand side of Eq. (8.41), the vector \mathbf{h} can be expressed as

$$\mathbf{h} = \left(\frac{\partial \mathbf{I}}{\partial \theta_1} \dot{\theta}_1 + \dots + \frac{\partial \mathbf{I}}{\partial \theta_n} \dot{\theta}_n \right) \dot{\boldsymbol{\theta}} - \frac{1}{2} \begin{bmatrix} \dot{\boldsymbol{\theta}}^T (\partial \mathbf{I} / \partial \theta_1) \dot{\boldsymbol{\theta}} \\ \vdots \\ \dot{\boldsymbol{\theta}}^T (\partial \mathbf{I} / \partial \theta_n) \dot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\theta}}^T \mathbf{C}_1 \dot{\boldsymbol{\theta}} \\ \vdots \\ \dot{\boldsymbol{\theta}}^T \mathbf{C}_n \dot{\boldsymbol{\theta}} \end{bmatrix} \quad (8.44b)$$

where the $n \times n$ matrix \mathbf{C}_i , for $i = 1, \dots, n$, is given by

$$\mathbf{C}_i \equiv \begin{bmatrix} c_{i,11} & c_{i,12} & \dots & c_{i,1n} \\ c_{i,21} & c_{i,22} & \dots & c_{i,2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{i,n1} & c_{i,n2} & \dots & c_{i,nn} \end{bmatrix} \quad (8.44c)$$

in which the term c_{ijk} is known as the *Christoffel symbols* of the first type and given as

$$c_{ijk} \equiv \frac{1}{2} \left(\frac{\partial i_{ij}}{\partial \theta_k} + \frac{\partial i_{ik}}{\partial \theta_j} - \frac{\partial i_{jk}}{\partial \theta_i} \right) \quad (8.44d)$$

Notice in Eq. (8.44d) that $c_{ijk} = c_{ikj}$ because the GIM \mathbf{I} is symmetric. Moreover, $\boldsymbol{\gamma} \equiv [\gamma_1, \dots, \gamma_n]^T$: the n -dimensional vector of gravitational accelerations. Its compact representation appears in Eq. (8.41) as, $\boldsymbol{\gamma} \equiv \mathbf{J}_c^T \mathbf{m}$; $\boldsymbol{\tau} \equiv [\tau_1, \dots, \tau_n]^T$: the n -dimensional vector of generalized forces.

It is interesting to note that Eq. (8.44a) represents a set of second-order differential equations in terms of four major parts, namely, those due to joint accelerations, denoted by $\mathbf{I}\ddot{\boldsymbol{\theta}}$, the centrifugal and Coriolis

Coriolis Force in Dynamics!

If one walks along the radius of a rotating merry-go-round, he or she will experience a force sideways, which is called coriolis force. This is due to the force resulting from the coriolis acceleration.

forces denoted with \mathbf{h} , which are quadratic in joint rates, i.e., $\dot{\theta}$, gravity forces denoted with γ , and the external forces and torques denoted with τ . The centrifugal and Coriolis components arise because of the non-inertial frames attached to the moving links. They can easily be identified as the Coriolis forces from the terms associated with $\dot{\theta}_i \dot{\theta}_j$ when $i \neq j$, whereas the terms associated with $\dot{\theta}_i^2$ are the centrifugal forces. It is important to note here that once the expression for the GIM is available from the kinetic-energy expression of the robot given by Eq. (8.34), one can easily compute the vector \mathbf{h} . In fact, once the kinetic energy expression is available, all the ingredients for the dynamic equations of motion given by either Eq. (8.42) or Eq. (8.44a) can be computed.

Another observation which can be made from the scalar expressions of the equations of motion given by Eq. (8.44a) is that the dynamic model is linear with respect to the dynamic properties of the links, namely, the masses, link lengths, elements of the inertia tensors, etc. Such representation is very useful for dynamic identification, as briefed in Siciliano et al. (2011), and essential in adaptive control of a robotic system, as explained in Chapter 11. The concept is explained with the help of an example given in Example 8.12.

Example 8.4 Alternative Expression for the Elements of Vector \mathbf{h} in Eq. (8.44a)

From the expression of h_i given in Eq. (8.43a), one can split the first term inside the summation, and appropriately interchange the summation symbols j and k appearing in the numerator and denominator of the second split term, i.e.,

$$\sum_{j=1}^n \sum_{k=1}^n \left(\frac{\partial i_{ij}}{\partial \theta_k} \right) \dot{\theta}_j \dot{\theta}_k = \frac{1}{2} \sum_{j=1}^n \sum_{k=1}^n \left(\frac{\partial i_{ij}}{\partial \theta_k} + \frac{\partial i_{ik}}{\partial \theta_j} \right) \dot{\theta}_j \dot{\theta}_k \quad (8.45)$$

Hence,

$$h_i \equiv \sum_{j=1}^n \sum_{k=1}^n \left(\frac{\partial i_{ij}}{\partial \theta_k} - \frac{1}{2} \frac{\partial i_{jk}}{\partial \theta_i} \right) \dot{\theta}_j \dot{\theta}_k = \frac{1}{2} \sum_{j=1}^n \sum_{k=1}^n \left(\frac{\partial i_{ij}}{\partial \theta_k} + \frac{\partial i_{ik}}{\partial \theta_j} - \frac{\partial i_{jk}}{\partial \theta_i} \right) \dot{\theta}_j \dot{\theta}_k \quad (8.46)$$

As a result, the element c_{ij} is extracted from Eq. (8.46) as

$$c_{ij} \equiv \sum_{k=1}^n c_{ijk} \dot{\theta}_k = \mathbf{c}_{ij}^T \dot{\theta}, \text{ where } \mathbf{c}_{ij} \equiv \begin{bmatrix} c_{ij1} & c_{ij2} & \dots & c_{ijn} \end{bmatrix}^T \quad (8.47)$$

In Eq. (8.47), the term c_{ijk} is the *Christoffel symbol* defined in Eq. (8.44d). Due to the derivation of Eq. (8.47), the vector \mathbf{h} of Eq. (8.44a) can also be expressed as $\mathbf{h} = \mathbf{C} \dot{\theta}$.

Example 8.5 Skew-symmetric Matrix, $\mathbf{I} - 2\mathbf{C}$

First, the time derivative on an element of the inertia matrix, \mathbf{I} , i.e., di_{ij}/dt , is obtained as

$$\frac{di_{ij}}{dt} = \sum_{k=1}^n \frac{di_{ij}}{d\theta_k} \dot{\theta}_k \quad (8.48)$$

Therefore, the (i,j) element of the matrix, $\dot{\mathbf{I}} - 2\mathbf{C}$, can be obtained using Eqs. (8.44)–(8.45) as

$$\frac{di_{ij}}{dt} - 2c_{ij} = \sum_{k=1}^n \frac{di_{ij}}{d\theta_k} \dot{\theta}_k - \sum_{k=1}^n \left(\frac{\partial i_{ij}}{\partial \theta_k} + \frac{\partial i_{ik}}{\partial \theta_j} - \frac{\partial i_{jk}}{\partial \theta_i} \right) \dot{\theta}_k = \sum_{k=1}^n \left(\frac{\partial i_{ik}}{\partial \theta_j} - \frac{\partial i_{jk}}{\partial \theta_i} \right) \dot{\theta}_k \quad (8.49)$$

Since the inertia matrix \mathbf{I} is symmetric, i.e., $i_{ij} = i_{ji}$, it can be shown by interchanging the indices in Eq. (8.49) that

$$\frac{di_{ji}}{dt} - 2c_{ji} = - \left(\frac{di_{ij}}{dt} - 2c_{ij} \right) \quad (8.50)$$

which implies that the matrix $\dot{\mathbf{I}} - 2\mathbf{C}$ is skew-symmetric. As a result, the following holds true:

$$\mathbf{x}^T (\dot{\mathbf{I}} - 2\mathbf{C}) \mathbf{x} = 0 \quad (8.51)$$

where \mathbf{x} is any arbitrary n -dimensional non-zero vector. The above property will be used in Chapter 11 to prove the stability of a robotic system.

Example 8.6 Proof of $\dot{\mathbf{I}} - \mathbf{C} + \mathbf{C}^T$

Since the matrix $\dot{\mathbf{I}} - 2\mathbf{C}$ is skew-symmetric, as proven in Example 8.5,

$$(\dot{\mathbf{I}} - 2\mathbf{C})^T = -(\dot{\mathbf{I}} - 2\mathbf{C}) = 2\mathbf{C} - \dot{\mathbf{I}} \quad (8.52a)$$

Expanding the left-hand side of Eq. (8.52a), one gets

$$\dot{\mathbf{I}}^T - 2\mathbf{C}^T = 2\mathbf{C} - \dot{\mathbf{I}} \quad (8.52b)$$

Due to the symmetric property of the GIM \mathbf{I} , its time-derivative, i.e., $\dot{\mathbf{I}}$, is also symmetric. This implies that $\dot{\mathbf{I}} = \dot{\mathbf{I}}^T$. Hence, Eq. (8.52b) is modified as

$$\dot{\mathbf{I}} - 2\mathbf{C}^T = 2\mathbf{C} - \dot{\mathbf{I}} \quad \Rightarrow \dot{\mathbf{I}} = \mathbf{C} + \mathbf{C}^T \quad (\text{proved}) \quad (8.52c)$$

Example 8.7 Dynamics of an One-link Arm

The dynamic equation of motion of an one-link one-DOF arm shown in Fig. 8.7 is derived using the Euler-Lagrange (EL) formulation. Two coordinate frames are attached as per the DH-parameter convention given in Chapter 5. Note that in Fig. 8.7, no subscript is used in the link parameters because there exists only one link. Using the EL formulation, the generalized coordinate is θ , whereas $a/2$ is the distance of the mass center of the link from its joint origin O . Moreover, let the mass of the link is m , and its inertia tensor about the mass center is denoted by \mathbf{I} . With the chosen coordinate frame, i.e., the fixed frame X_1-Y_1 , the Jacobians, $\mathbf{J}_{\omega,i}$ and $\mathbf{J}_{c,i}$ for $i = 1$, Eqs. (8.30a-b) yield

$$\mathbf{J}_{\omega,1} \equiv [0 \ 0 \ 1]^T; \quad \mathbf{J}_{c,1} \equiv \left[-\frac{a}{2}s \ \frac{a}{2}c \ 0 \right]^T \quad (8.53a)$$

where s and c represent $\sin \theta$ and $\cos \theta$, respectively. From Eq. (8.33), the scalar inertia term I is given by

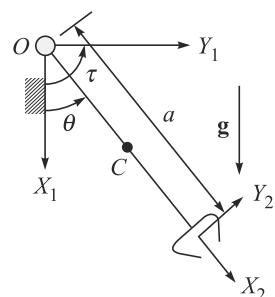


Fig. 8.7 One-link arm

$$I \equiv \frac{m}{4}a^2 + I_{zz} = \frac{m}{3}a^2 \quad (8.53b)$$

Note here that even though the inertia matrix \mathbf{I} of the link may have all nonzero elements, due to the structure of \mathbf{J}_{ω_1} only the (3, 3) element of \mathbf{I} , namely, I_{zz} , contributes to the generalized inertia term. Using Eq. (8.43), the elements of the vectors \mathbf{h} and $\boldsymbol{\gamma}$, namely, the scalars h and γ , respectively, are obtained as follows:

$$h = 0; \text{ and } \gamma \equiv -mg^T \mathbf{j}_{c,1}^{(1)} = \frac{a}{2}mgs \quad (8.54)$$

where $\mathbf{g} \equiv [g \ 0 \ 0]^T$ — g being the acceleration due to gravity acting along positive X_1 -axis. Using Eqs. (8.53b) and (8.54), the equation of motion in the form of Eq. (8.44a) is given as follows:

$$\frac{1}{3}ma^2\ddot{\theta} + \frac{1}{2}mgas = \tau \quad (8.55)$$

It is pointed out here that for a simple system like the one-link arm, it is advisable to obtain the equation of motion, Eq. (8.55), directly from Eqs. (8.24) and (8.36). This is shown below:

$$T \equiv \frac{1}{2}m\left(\frac{a}{2}\dot{\theta}\right)^2 + \frac{1}{2}\frac{ma^2}{12}\dot{\theta}^2; \ U = -mg\frac{a}{2}c$$

and $L = T - U \equiv \frac{ma^2}{6}\dot{\theta}^2 + mg\frac{a}{2}c$ (8.56a)

For the only independent generalized coordinate θ , Eq. (8.24) then yields

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = \frac{1}{3}ma^2\ddot{\theta}; \frac{\partial L}{\partial \theta} = -\frac{1}{2}mgas; \text{ and } \frac{1}{3}ma^2\ddot{\theta} + \frac{1}{2}mgas = \tau \quad (8.56b)$$

which is the same as obtained in Eq. (8.55).

Example 8.8 Dynamics of an one-link Arm using MATLAB and MuPAD

In order to obtain the result of Eq. (8.56) using Eq. (8.24), it is not possible to program in MATLAB or MuPAD exactly the way it is. The chain-rule must be used to evaluate the first integral of the EL equation, Eq. (8.24). For the single-DOF one-link arm with θ as the independent generalized coordinate, it can be obtained as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = \frac{\partial}{\partial \theta}\left(\frac{\partial L}{\partial \dot{\theta}}\right)\dot{\theta} + \frac{\partial}{\partial \dot{\theta}}\left(\frac{\partial L}{\partial \dot{\theta}}\right)\ddot{\theta} \quad (8.57)$$

Using Eqs. (8.24) and (8.57), the symbolic program in MATLAB can be written as

```
>> %MATLAB Symbolic Program to generate Euler-Lagrange
>> %equation of motion for one-link arm
>> syms m a g th thd thdd;
>> L=1/6*m*a*a*thd^2-1/2*m*g*a*(-cos(th));
>> pLpthd=diff(L,thd);
```

```
>> ddtpLpthd=diff (pLpthd, th) *thd+diff (pLpthd, thd) *thdd;
>> pLpth=diff (L, th);
>> tau=ddtpLpthd-pLpth;
>> disp ('tau=')
>> pretty(tau)
```

The output of the above program as visible in the MATLAB window is shown below:

$$\begin{aligned} \text{tau=} \\ & \frac{m \text{ thdd } a - g m \sin(\text{th}) a}{3 + \frac{2}{3}} \end{aligned}$$

The same result can also be verified using the MuPAD notebook of MATLAB. Figure 8.8 shows how to write the commands in MuPAD to obtain the Euler-Lagrange equation of motion for the one-link arm.

Define abbreviations for symbols used in the equations later

```
[th:=&theta; :thd:=Symbol::accentDot(th):
[thdd:=Symbol::accentDoubleDot(th):tau:='&tau; `:
[T:=1/2*m*((a/2)^2+a*a/12)*thd*thd
[θ² a² m
[ 6
[U:=-m*g*a/2*cos(th)
[- a g m cos(θ)
[ 2
[L:=T-U
[m θ² a² + g m cos(θ) a
[ 6 2
[pLpthd:=diff(L,thd)
[θ a² m
[ 3
[ddtpLpthd:=diff(pLpthd,th)*thd+diff(pLpthd,thd)*thdd
[θ a² m
[ 3
[pLpth:=diff(L,th)
[- a g m sin(θ)
[ 2
[tau:=ddtpLpthd-pLpth
[θ m a² + g m sin(θ) a
[ 3 2
```

Final EL equation of motion of one-link system is

```
[print(`&tau; `=tau)
[τ = θ m a² + g m sin(θ) a
[ 3 2
```

Fig. 8.8 MuPAD window to obtain EL equation of one-link arm

Example 8.9 Dynamics of a two-link Robot Arm

The equations of motion of a two-link two-DOF robot arm shown in Fig. 8.5 are derived here based on the Euler–Lagrange equations of motion, Eq. (8.24). The vector of independent generalized coordinates is $\boldsymbol{\theta} \equiv [\theta_1, \theta_2]^T$, whereas $a_1/2$ and $a_2/2$, are the distances of the center of the masses of the two links from their respective joint locations, namely, O_1 and O_2 of Fig. 8.5. Moreover, let the masses of the two links be m_1 and m_2 , and their inertia tensors about the mass centers are \mathbf{I}_1 and \mathbf{I}_2 . With the chosen coordinates frames, i.e., the fixed frame F , the Jacobians $\mathbf{J}_{\omega,i}$ and $\mathbf{J}_{c,i}$, for $i = 1, 2$ from Eqs. (8.30a-b) yield

$$\mathbf{J}_{\omega,1} \equiv \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}; \quad \mathbf{J}_{\omega,2} \equiv \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \quad (8.58a)$$

and

$$\mathbf{J}_{c,1} \equiv \begin{bmatrix} -\frac{a_1}{2}s_1 & 0 \\ \frac{a_1}{2}c_1 & 0 \\ 0 & 0 \end{bmatrix}; \quad \mathbf{J}_{c,2} \equiv \begin{bmatrix} -a_1s_1 - \frac{a_2}{2}s_{12} & -\frac{a_2}{2}s_{12} \\ a_1c_1 + \frac{a_2}{2}c_{12} & \frac{a_2}{2}c_{12} \\ 0 & 0 \end{bmatrix} \quad (8.58b)$$

From Eq. (8.33), the 2×2 generalized inertia matrix \mathbf{I} is given by

$$\mathbf{I} \equiv \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix} \quad (8.59a)$$

whose elements are as follows:

$$i_{11} = \frac{m_1}{4}a_1^2 + I_{1,zz} + m_2 \left(a_1^2 + \frac{a_2^2}{4} + a_1 a_2 c_2 \right) + I_{2,zz} \quad (8.59b)$$

$$i_{12} = i_{21} \equiv m_2 \left(\frac{a_2^2}{4} + \frac{a_1 a_2 c_2}{2} \right) + I_{2,zz} \quad (8.59c)$$

$$i_{22} = m_2 \frac{a_2^2}{4} + I_{2,zz} \quad (8.59d)$$

where, $s_i \equiv \sin \theta_i$ and $c_i \equiv \cos \theta_i$, for $i = 1, 2$. Note here that even though the inertia matrices \mathbf{I}_1 and \mathbf{I}_2 of links 1 and 2, respectively, may have all nonzero elements due to the structures of $\mathbf{J}_{\omega,1}$ and $\mathbf{J}_{\omega,2}$ only the (3, 3) elements of \mathbf{I}_1 and \mathbf{I}_2 , namely, $I_{1,zz}$ and $I_{2,zz}$, contribute to the generalized inertia matrix expressions, Eqs. (8.59a-d). Using Eqs. (8.43a-b), the elements of the vectors \mathbf{h} and $\boldsymbol{\gamma}$, i.e., h_i and γ_i , respectively, for $i = 1, 2$, are obtained as follows:

$$h_1 \equiv \frac{1}{2} \frac{\partial i_{11}}{\partial \theta_1} \dot{\theta}_1^2 + \frac{\partial i_{11}}{\partial \theta_2} \dot{\theta}_1 \dot{\theta}_2 + \left(\frac{\partial i_{12}}{\partial \theta_2} - \frac{1}{2} \frac{\partial i_{22}}{\partial \theta_1} \right) \dot{\theta}_2^2 = -m_2 a_1 a_2 s_2 \dot{\theta}_2 \left(\dot{\theta}_1 + \frac{1}{2} \dot{\theta}_2 \right) \quad (8.60a)$$

$$h_2 \equiv \left(\frac{\partial i_{21}}{\partial \theta_1} - \frac{1}{2} \frac{\partial i_{11}}{\partial \theta_2} \right) \dot{\theta}_1^2 + \frac{\partial i_{22}}{\partial \theta_1} \dot{\theta}_1 \dot{\theta}_2 + \frac{1}{2} \frac{\partial i_{22}}{\partial \theta_2} \dot{\theta}_2^2 = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1^2 \quad (8.60b)$$

and

$$\gamma_1 \equiv -m_1 \mathbf{g}^T \mathbf{j}_{c,1}^{(1)} - m_2 \mathbf{g}^T \mathbf{j}_{c,2}^{(1)} = m_1 g \frac{a_1}{2} c_1 + m_2 g \left(a_1 c_1 + \frac{a_2}{2} c_{12} \right) \quad (8.61a)$$

$$\gamma_2 \equiv -m_1 \mathbf{g}^T \mathbf{j}_{c,1}^{(2)} - m_2 \mathbf{g}^T \mathbf{j}_{c,2}^{(2)} = m_2 g \frac{a_2}{2} c_{12} \quad (8.61b)$$

where, $\mathbf{g} \equiv [0 \quad -g \quad 0]^T$ — g being the acceleration due to gravity.

Example 8.10 Use of Christoffel Symbols to Find Vector \mathbf{h} for a Two-link Robot Arm

For the 2-link 2-DOF arm, there will be a total of eight Christoffel symbols, namely, $c_{111}, c_{112}, c_{122}, c_{121}, c_{211}, c_{212}, c_{221}, c_{222}$. They are evaluated using Eq. (8.44d) as

$$\begin{aligned} c_{111} &\equiv \frac{1}{2} \left(\frac{\partial i_{11}}{\partial \theta_1} + \frac{\partial i_{11}}{\partial \theta_1} - \frac{\partial i_{11}}{\partial \theta_1} \right) = 0 \\ c_{112} &\equiv \frac{1}{2} \left(\frac{\partial i_{11}}{\partial \theta_2} + \frac{\partial i_{12}}{\partial \theta_1} - \frac{\partial i_{12}}{\partial \theta_1} \right) = -\frac{1}{2} m_2 a_1 a_2 s_2 \end{aligned} \quad (8.62a)$$

$$\begin{aligned} c_{121} &\equiv \frac{1}{2} \left(\frac{\partial i_{12}}{\partial \theta_1} + \frac{\partial i_{11}}{\partial \theta_2} - \frac{\partial i_{21}}{\partial \theta_1} \right) = -\frac{1}{2} m_2 a_1 a_2 s_2 \\ c_{122} &\equiv \frac{1}{2} \left(\frac{\partial i_{12}}{\partial \theta_2} + \frac{\partial i_{12}}{\partial \theta_2} - \frac{\partial i_{22}}{\partial \theta_1} \right) = -\frac{1}{2} m_2 a_1 a_2 s_2 \end{aligned} \quad (8.62b)$$

$$\begin{aligned} c_{211} &\equiv \frac{1}{2} \left(\frac{\partial i_{21}}{\partial \theta_1} + \frac{\partial i_{21}}{\partial \theta_1} - \frac{\partial i_{11}}{\partial \theta_2} \right) = \frac{1}{2} m_2 a_1 a_2 s_2 \\ c_{212} &\equiv \frac{1}{2} \left(\frac{\partial i_{21}}{\partial \theta_2} + \frac{\partial i_{22}}{\partial \theta_1} - \frac{\partial i_{12}}{\partial \theta_2} \right) = 0 \end{aligned} \quad (8.62c)$$

$$\begin{aligned} c_{221} &\equiv \frac{1}{2} \left(\frac{\partial i_{22}}{\partial \theta_1} + \frac{\partial i_{21}}{\partial \theta_2} - \frac{\partial i_{21}}{\partial \theta_2} \right) = 0 \\ c_{222} &\equiv \frac{1}{2} \left(\frac{\partial i_{22}}{\partial \theta_2} + \frac{\partial i_{22}}{\partial \theta_2} - \frac{\partial i_{22}}{\partial \theta_2} \right) = 0 \end{aligned} \quad (8.62d)$$

Using Eq. (8.47), the elements of 2×2 matrix \mathbf{C} are then computed as

$$\begin{aligned} c_{11} &\equiv c_{111} \dot{\theta}_1 + c_{112} \dot{\theta}_2 = -\frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_2 \\ c_{12} &\equiv c_{121} \dot{\theta}_1 + c_{122} \dot{\theta}_2 = -\frac{1}{2} m_2 a_1 a_2 s_2 (\dot{\theta}_1 + \dot{\theta}_2) \end{aligned} \quad (8.63a)$$

$$\begin{aligned} c_{21} &\equiv c_{211} \dot{\theta}_1 + c_{212} \dot{\theta}_2 = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1 \\ c_{22} &\equiv c_{221} \dot{\theta}_1 + c_{222} \dot{\theta}_2 = 0 \end{aligned} \quad (8.63b)$$

The two-dimensional vector \mathbf{h} is then evaluated as $\mathbf{h} = \mathbf{C}\dot{\theta}$, which yields

$$\begin{aligned} h_1 &\equiv c_{11}\dot{\theta}_1 + c_{12}\dot{\theta}_2 = -m_2 a_1 a_2 s_2 \dot{\theta}_2 \left(\dot{\theta}_1 + \frac{1}{2} \dot{\theta}_2 \right) \\ h_2 &\equiv c_{21}\dot{\theta}_1 + c_{22}\dot{\theta}_2 = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1^2 \end{aligned} \quad (8.64)$$

The expressions in Eq. (8.64) are exactly same as obtained in Eqs. (8.60a-b).

Note that Eq. (8.64) can also be explicitly written in terms of centrifugal and Coriolis terms, namely, $\mathbf{h} = \mathbf{C}_{cf}\dot{\theta}_{ii} + \mathbf{C}_{cl}\dot{\theta}_{ij}$, where \mathbf{C}_{cf} and $\dot{\theta}_{ii}$ are the coefficient matrix and the vector associated with the squares of the joint rates, i.e., $\dot{\theta}_1^2$, $\dot{\theta}_2^2$, etc., respectively, whereas \mathbf{C}_{cl} and $\dot{\theta}_{ij}$ are the coefficient matrix and the vector associated with the cross-multiplicative terms, i.e., $\dot{\theta}_1\dot{\theta}_2$, $\dot{\theta}_2\dot{\theta}_3$, etc., respectively. For the two-link arm they are as follows:

$$\mathbf{C}_{cf} \equiv \begin{bmatrix} c_{1,11} & c_{1,22} \\ c_{2,11} & c_{2,22} \end{bmatrix} = \frac{1}{2} m_2 a_1 a_2 s_2 \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}; \quad \dot{\theta}_{ii} \equiv \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} \quad (8.65a)$$

$$\mathbf{C}_{cl} \equiv \begin{bmatrix} 2c_{1,12} \\ 2c_{2,21} \end{bmatrix} = \frac{1}{2} m_2 a_1 a_2 s_2 \begin{bmatrix} -2 \\ 0 \end{bmatrix}; \quad \dot{\theta}_{ij} \equiv \begin{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \\ \dot{\theta}_2 \dot{\theta}_3 \end{bmatrix} \quad (8.65b)$$

For an n -DOF manipulator, \mathbf{C}_{cf} and \mathbf{C}_{cl} are respectively $n \times n$ and $n \times (n - 1)/2$ matrices, whereas $\dot{\theta}_{ii}$ and $\dot{\theta}_{ij}$ are correspondingly n - and $n(n - 1)/2$ -dimensional vectors, respectively.

Example 8.11 Proof of Symmetric Matrices $\dot{\mathbf{I}} - 2\mathbf{C}$ and $\dot{\mathbf{I}} = \mathbf{C} + \mathbf{C}^T$ for the 2-link Robot Arm

From the expressions of the elements of the GIM for the two-link robot arm given by Eqs. (8.59a-d), its time-derivative and the matrix \mathbf{C} from Eqs. (8.63a-b), are written as follows:

$$\dot{\mathbf{I}} \equiv \frac{1}{2} m_2 a_1 a_2 s_2 \begin{bmatrix} -2\dot{\theta}_2 & -\dot{\theta}_2 \\ -\dot{\theta}_2 & 0 \end{bmatrix}; \quad \mathbf{C} \equiv \frac{1}{2} m_2 a_1 a_2 s_2 \begin{bmatrix} -\dot{\theta}_2 & -\dot{\theta}_1 - \dot{\theta}_2 \\ \dot{\theta}_1 & 0 \end{bmatrix} \quad (8.66a)$$

The expression $\dot{\mathbf{I}} - 2\mathbf{C}$ is then obtained as

$$\dot{\mathbf{I}} - 2\mathbf{C} = \frac{1}{2} m_2 a_1 a_2 s_2 \begin{bmatrix} 0 & 2\dot{\theta}_1 + \dot{\theta}_2 \\ -2\dot{\theta}_1 - \dot{\theta}_2 & 0 \end{bmatrix} \quad (8.66b)$$

which is clearly skew-symmetric as $(\dot{\mathbf{I}} - 2\mathbf{C})^T = -(\dot{\mathbf{I}} - 2\mathbf{C})$. Moreover, it is a simple matter to verify from the elements of the expression $\mathbf{C} + \mathbf{C}^T$ that its elements are same as those appearing in $\dot{\mathbf{I}}$ of Eq. (8.66a).

Example 8.12 Linear Dynamic Model of a Two-link Robot Arm

As mentioned at the end of Section 8.2.4, the linear form of dynamic equations given by Eq. (8.44a) is essential for the adaptive control of a robotic system (see Section 11.9). For the two-link robot arm, Eq. (8.44a) can be represented as

$$\ddot{\mathbf{I}}\dot{\theta} + \mathbf{h} + \boldsymbol{\gamma} = \mathbf{Y}\hat{\mathbf{p}} = \boldsymbol{\tau} \quad (8.67)$$

where the matrix \mathbf{Y} is generally referred as *regressor* whose dimension depends on the choice of the constant physical parameters, i.e., the vector $\hat{\mathbf{p}}$. In this example, consider the vector $\hat{\mathbf{p}}$ as a 5-dimensional vector containing the mass, the mass center location and the inertia of the two links, namely,

$$\hat{\mathbf{p}} \equiv [m_1 a_1 / 2 \quad I_{1,ZZ} \quad m_2 \quad m_2 a_2 / 2 \quad I_{2,ZZ}]^T \quad (8.68a)$$

For the definition of vector $\hat{\mathbf{p}}$, corresponding to 2×5 matrix is given as

$$\mathbf{Y} \equiv \begin{bmatrix} \frac{a_1}{2} \ddot{\theta}_1 + g c_1 & \ddot{\theta}_1 & a_1^2 \ddot{\theta}_1 + g a_1 c_1 & a_1 c_2 (2\ddot{\theta}_1 + \ddot{\theta}_2) + \frac{a_2}{2} \ddot{\theta}_{12} + \tilde{h}_1 + g c_{12} & \ddot{\theta}_{12} \\ 0 & 0 & 0 & a_1 c_2 \ddot{\theta}_1 + \frac{a_2}{2} \ddot{\theta}_{12} + \tilde{h}_2 + g c_{12} & \ddot{\theta}_{12} \end{bmatrix} \quad (8.68b)$$

where $\dot{\theta}_{12} \equiv \dot{\theta}_1 + \dot{\theta}_2$, and $\tilde{h}_1 \equiv -a_1 s_2 (2\dot{\theta}_1 + \dot{\theta}_2) \dot{\theta}_2$ and $\tilde{h}_2 \equiv a_1 s_2 \dot{\theta}_1^2$. Note that the expression of the matrix \mathbf{Y} is not unique. For example, the term $a_2 \ddot{\theta}_{12} / 2$ in (1,4) element can be removed and added to (1,3) element as $a_2^2 \ddot{\theta}_{12} / 4$, thereby, changing the representation. Besides, the choice of the vector $\hat{\mathbf{p}}$ could also change the representation of matrix \mathbf{Y} , as shown in Spong and Vidyasagar (2004), Siciliano et al. (2011), and others.

8.3 NEWTON–EULER FORMULATION

In this section, first, the linear and angular momenta of a rigid body are defined which will be used for the derivation of Newton–Euler (NE) equations of the motion of a robotic system at hand. Based on the NE equations, a recursive method for the inverse dynamics analysis is presented that will be useful for robot control.

8.3.1 Linear Momentum

The linear momentum of a mass element ρdV with respect to a point O , as shown in Fig. 8.1, is denoted with $d\mathbf{m}$ —note that the boldfaced \mathbf{m} is different from the italics m representing mass—and is defined by

$$d\mathbf{m} = \frac{d\mathbf{p}}{dt} \rho dV \quad (8.69)$$

where \mathbf{p} is the position vector of the mass element shown in Fig. 8.1, and t is time. Hence, the total linear momentum of the material body B about O is given by

$$\mathbf{m} = \int_V \frac{d\mathbf{p}}{dt} \rho dV = \frac{d}{dt} \int_V \mathbf{p} \rho dV \quad (8.70)$$

Using the definition of the mass center given by Eq. (8.1a), Eq. (8.70) can be rewritten as

$$\mathbf{m} = \frac{d}{dt} (m\mathbf{c}) = m\dot{\mathbf{c}} \quad (8.71)$$

In deriving Eq. (8.71), the time derivative of the mass m is zero as for a rigid body it does not change with time. Moreover, $\dot{\mathbf{c}} \equiv d\mathbf{c}/dt$ denotes the velocity

Can the mass of an object change?

Of course, yes. Imagine an aeroplane or a car whose total mass reduces as the fuel is consumed due to their running.

of the center of mass with respect to the reference frame F . Equation (8.71) implies that the total linear momentum of a rigid body is equal to the linear momentum of a point mass m located at the center of mass, C .

8.3.2 Angular Momentum

Referring to Fig. 8.1, the angular momentum $d\tilde{\mathbf{m}}^O$ of a mass element ρdV about a reference point O is defined as the moment of its linear momentum, Eq. (8.71), about O , i.e.,

$$d\tilde{\mathbf{m}}^O = \mathbf{p} \times \frac{d\mathbf{p}}{dt} \rho dV \quad (8.72a)$$

Therefore, the total angular momentum of B about O is given by

$$\tilde{\mathbf{m}}^O = \int_V \mathbf{p} \times \frac{d\mathbf{p}}{dt} \rho dV \quad (8.72b)$$

Substituting $\mathbf{p} = \mathbf{c} + \mathbf{r}$, as shown in Fig. 8.1, into Eq. (8.72b) one obtains

$$\tilde{\mathbf{m}}^O = \int_V \mathbf{c} \times \left(\frac{d\mathbf{c}}{dt} \rho dV \right) + \int_V \mathbf{c} \times \left(\frac{d\mathbf{r}}{dt} \rho dV \right) + \int_V \mathbf{r} \times \left(\frac{d\mathbf{c}}{dt} \rho dV \right) + \int_V \mathbf{r} \times \left(\frac{d\mathbf{r}}{dt} \rho dV \right) \quad (8.73)$$

Noting that

$$\frac{d\mathbf{r}}{dt} = \boldsymbol{\omega} \times \mathbf{r}$$

where $\boldsymbol{\omega}$ is the angular velocity of body B . Equation (8.73) can be rearranged as

$$\tilde{\mathbf{m}}^O = \mathbf{c} \times \dot{\mathbf{c}} \int_V \rho dV + \mathbf{c} \times \left(\boldsymbol{\omega} \times \int_V \mathbf{r} \rho dV \right) + \left(\int_V \mathbf{r} \rho dV \right) \times \dot{\mathbf{c}} + \int_V \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) \rho dV \quad (8.74)$$

in which, $\int_V \rho dV = m$, and from Eq. (8.1d), $\int_V \mathbf{r} \rho dV = 0$. Hence, the second and third terms of Eq. (8.74) vanish yielding the total angular momentum about O , i.e., $\tilde{\mathbf{m}}^O$, as

$$\tilde{\mathbf{m}}^O = \mathbf{c} \times (m\dot{\mathbf{c}}) + \tilde{\mathbf{m}} \quad (8.75a)$$

where

$$\tilde{\mathbf{m}} \equiv \int_V \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) \rho dV \quad (8.75b)$$

The term, $\tilde{\mathbf{m}}$ of Eq. (8.75b), denotes the angular momentum of the rigid body about its center of mass C . Moreover, the vector $\dot{\mathbf{c}}$ of Eq. (8.75a) is the velocity of the center of mass C with respect to the reference frame F . Furthermore, Eq. (8.75a) is the total angular momentum of B about the origin O that is equal to the angular momentum of a point mass with mass m concentrated at the center of mass, plus the angular momentum of rotation about its center of mass. Now, using the definition of triple cross-product, namely, $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a}^T \mathbf{c})\mathbf{b} - (\mathbf{a}^T \mathbf{b})\mathbf{c}$ --- \mathbf{a} , \mathbf{b} , and \mathbf{c} being the three-dimensional Cartesian vectors, the expression $\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})$ of Eq. (8.75b) can be expressed as

$$\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) = (\mathbf{r}^T \mathbf{r})\boldsymbol{\omega} - (\mathbf{r}^T \boldsymbol{\omega})\mathbf{r} = [(\mathbf{r}^T \mathbf{r})\mathbf{1} - \mathbf{r}\mathbf{r}^T]\boldsymbol{\omega} \quad (8.76)$$

Comparing Eqs. (8.5) and (8.76), it follows that

$$\tilde{\mathbf{m}} \equiv \mathbf{I}\boldsymbol{\omega}, \text{ where } \mathbf{I} \equiv \int_V [(\mathbf{r}^T \mathbf{r})\mathbf{1} - \mathbf{r}\mathbf{r}^T] \rho dV \quad (8.77)$$

\mathbf{I} being the inertia tensor of the body B about its center of mass C . It is pointed out here that the angular momentum given by Eq. (8.77) can be expressed in any reference frame. Expressing Eq. (8.77) in the fixed reference frame F , one obtains

$$[\tilde{\mathbf{m}}]_F = [\mathbf{I}]_F [\boldsymbol{\omega}]_F \quad (8.78)$$

in which $[.]_F$ denotes the vector or matrix quantity expressed in the frame F . Expressing Eq. (8.78) in a body-fixed center of mass frame C , one obtains

$$[\tilde{\mathbf{m}}]_C = [\mathbf{I}]_C [\boldsymbol{\omega}]_C \quad (8.79)$$

where $[\boldsymbol{\omega}]_C$ is the angular velocity of the body B relative to the fixed frame F but expressed in the body frame C , i.e., $[\boldsymbol{\omega}]_C = \mathbf{Q}^T [\boldsymbol{\omega}]_F$. The 3×3 matrix \mathbf{Q} is the rotation matrix describing the orientation of the frame C with respect to frame F . Since $\tilde{\mathbf{m}}$ is a vector quantity, its transformation follows that of a vector, namely,

$$[\tilde{\mathbf{m}}]_F = \mathbf{Q}[\tilde{\mathbf{m}}]_C \quad (8.80)$$

Substituting Eqs. (8.78) and (8.79) into Eq. (8.80) the following is obtained:

$$[\mathbf{I}]_F [\boldsymbol{\omega}]_F = \mathbf{Q}[\mathbf{I}]_C [\boldsymbol{\omega}]_C \quad (8.81)$$

Using $[\boldsymbol{\omega}]_C = \mathbf{Q}^T [\boldsymbol{\omega}]_F$, Eq. (8.81) gives

$$[\mathbf{I}]_F = \mathbf{Q}[\mathbf{I}]_C \mathbf{Q}^T \quad (8.82)$$

Equation (8.82) transforms an inertia matrix expressed in one reference frame into another. Both inertia matrices are computed about the center of mass C , where the elements of $[\mathbf{I}]_C$ are constants because they are calculated in the body-fixed frame C attached to the body or link. On the other hand, the elements of $[\mathbf{I}]_F$ are variable according to Eq. (8.82), as it is a function of the rotation matrix \mathbf{Q} , which defines the orientation of frame C with respect to F .

8.3.3 Equations of Motion

It is assumed that there exists a fixed inertia frame with respect to which the Newton–Euler (NE) equations of motion will be derived. As shown in Fig. 8.9, let F be the fixed frame. Moreover, the vector \mathbf{m} is the linear momentum of the rigid link or the body B expressed in the frame F . The corresponding angular momentum is represented by the vector $\tilde{\mathbf{m}}$. Also, let vectors \mathbf{f}^O and \mathbf{n}^O be the resultant forces and moments exerted on B at and about the origin O , respectively. Then, the Newton’s linear equations of the motion can be stated as the time-derivative of the linear momentum \mathbf{m} equals to the external forces acting on it, i.e.,

$$\mathbf{f}^O = \frac{d\mathbf{m}}{dt} \quad (8.83)$$

whereas the Euler’s equations of rotational motion are given as the time rate of change of angular momentum $\tilde{\mathbf{m}}^O$ equals to the external moment acting on it (remember the point about which the angular momentum and external moment are taken, which should be same!), i.e.,

$$\mathbf{n}^O = \frac{d\tilde{\mathbf{m}}^O}{dt} \quad (8.84)$$

If an arbitrary point is chosen as the reference point, it may be inconvenient to apply the basic NE equations of motion, Eqs. (8.83)–(8.84). When the center of mass C is used as the reference point, the motion of the rigid body can be naturally split into two parts; linear motion of the center of mass C , plus a rotational motion of the rigid body about the center of mass C .

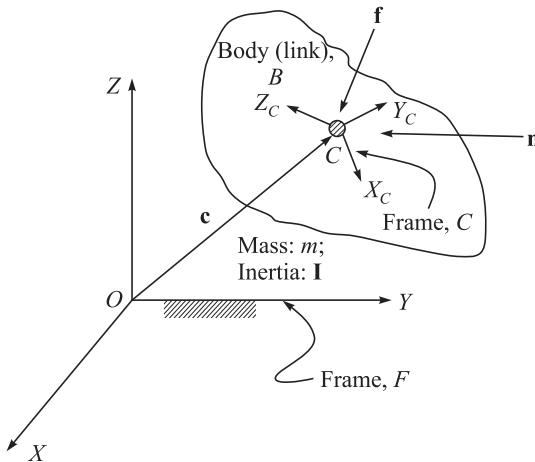


Fig. 8.9 Resultant force and moment acting on a rigid body (link)

Referring to Fig. 8.9, the center of mass of B is C , denoted with the vector \mathbf{c} . First, Newton's law is applied. Substituting Eq. (8.71) into Eq. (8.83) gives

$$\mathbf{f}^O = \frac{d(m\dot{\mathbf{c}})}{dt} \quad (8.85)$$

For a body of constant mass, Eq. (8.85) reduces to

$$\mathbf{f}^O = m \frac{d\dot{\mathbf{c}}}{dt} = m\ddot{\mathbf{c}} \quad (8.86)$$

Equation (8.86) is called *Newton's equation of motion for the center of mass*. Next, the rotational motion of B is considered. Differentiation of Eq. (8.84) with respect to time yields

$$\frac{d\tilde{\mathbf{m}}^O}{dt} = \mathbf{c} \times m \frac{d\dot{\mathbf{c}}}{dt} + \frac{d\tilde{\mathbf{m}}}{dt} = \mathbf{c} \times m\ddot{\mathbf{c}} + \frac{d\tilde{\mathbf{m}}}{dt} \quad (8.87)$$

Let \mathbf{f} and \mathbf{n} be the resultant of forces and moments exerted at the center of mass C , as shown in Fig. 8.9. Then, the following are true:

$$\mathbf{f}^O = \mathbf{f} \quad (8.88a)$$

$$\mathbf{n}^O = \mathbf{n} + \mathbf{c} \times \mathbf{f} \quad (8.88b)$$

Comparing Eqs. (8.86) and Eq. (8.88a), one gets

$$\mathbf{f} = m\ddot{\mathbf{c}} \quad (8.89)$$

Moreover, substituting Eqs. (8.87) and (8.88b) into (8.84), one obtains

$$\mathbf{n} + \mathbf{c} \times \mathbf{f} = \frac{d\tilde{\mathbf{m}}}{dt} + \mathbf{c} \times m\ddot{\mathbf{c}} \quad (8.90a)$$

Substituting Eq. (8.89) into Eq. (8.90a) and comparing its both sides, one obtains the following:

$$\mathbf{n} = \frac{d\tilde{\mathbf{m}}}{dt} \quad (8.90b)$$

In words, the rate of change of angular momentum of B about its center of mass C is equal to the resulting moment exerted at the same point. The derivative of $\tilde{\mathbf{m}}$ can be developed most conveniently in the body-fixed center of mass frame C shown in Fig. 8.9, because the inertia tensor components of B are constant in the frame C . Substituting Eq. (8.77) into Eq. (8.90), and expressing the resultant equation in the frame C yields

$$[\mathbf{n}]_C = \frac{d([\mathbf{I}]_C [\boldsymbol{\omega}]_C)}{dt} \quad (8.91)$$

Note that the differentiation of $\tilde{\mathbf{m}}$ in Eq. (8.90) or Eq. (8.91) means that it is with respect to the inertia frame F even though the quantities are represented in frame C . Applying the differentiation rule, Eq. (8.91) results in the following:

$$[\mathbf{n}]_C = [\mathbf{I}]_C [\dot{\boldsymbol{\omega}}]_C + [\boldsymbol{\omega}]_C \times ([\mathbf{I}]_C [\boldsymbol{\omega}]_C) \quad (8.92)$$

Equation (8.92) is called *Euler's equations of rotational motion for the center of mass coordinate frame*. Euler's equations of motion can also be written in the fixed frame F . To do this, both sides of Eq. (8.92) are to be multiplied by the rotation matrix \mathbf{Q} and the relationships, $[\boldsymbol{\omega}]_C = \mathbf{Q}^T [\boldsymbol{\omega}]_F$, and $[\mathbf{n}]_C = \mathbf{Q}^T [\mathbf{n}]_F$, need to be used, i.e.,

$$[\mathbf{n}]_F = \mathbf{Q}[\mathbf{I}]_C \mathbf{Q}^T [\boldsymbol{\omega}]_F + [\boldsymbol{\omega}]_F \times (\mathbf{Q}[\mathbf{I}]_C \mathbf{Q}^T [\boldsymbol{\omega}]_F) \quad (8.93a)$$

or simply

$$[\mathbf{n}]_F = [\mathbf{I}]_F [\dot{\boldsymbol{\omega}}]_F + [\boldsymbol{\omega}]_F \times ([\mathbf{I}]_F [\boldsymbol{\omega}]_F), \text{ where } [\mathbf{I}]_F \equiv \mathbf{Q}[\mathbf{I}]_C \mathbf{Q}^T \quad (8.93b)$$

Equation (8.93b) is called *Euler's equations of rotational motion for a nonbody* fixed frame, say, the fixed frame F . Although Eqs. (8.92) and (8.93b) have similar forms, they are fundamentally different. The inertia tensor elements in Eq. (8.92) are constants, whereas those in Eq. (8.93b) are time dependent. Hence, Eq. (8.92) is preferred over Eq. (8.93b).

8.4 RECURSIVE NEWTON-EULER ALGORITHM

Here, a recursive Newton-Euler (NE) algorithm for the dynamic analysis of a serial robot, namely, the inverse dynamics, is presented that incorporates all the moments and forces acting on the individual link of the robot manipulator. Here, the resulting equations include all the constraint moments and forces due to a joint between the two adjacent links. These constraint moments and forces are useful for sizing the links and bearings during the design stage. However, they are not required for motion analysis, be it inverse or forward

Inverse vs. Forward Dynamics

Inverse dynamics deals with the evaluation of joint torques and forces for a set of joint motions, whereas forward dynamics computes the joint motions for a given set of joint torques and forces.

dynamics. The method consists of a forward computation of the velocities and accelerations of each link, followed by the backward computation of the moments and forces at each joint. The moments and forces acting on a typical link i of a serial manipulator are shown in the Fig. 8.10. For the purpose of analysis, the following notations are employed:

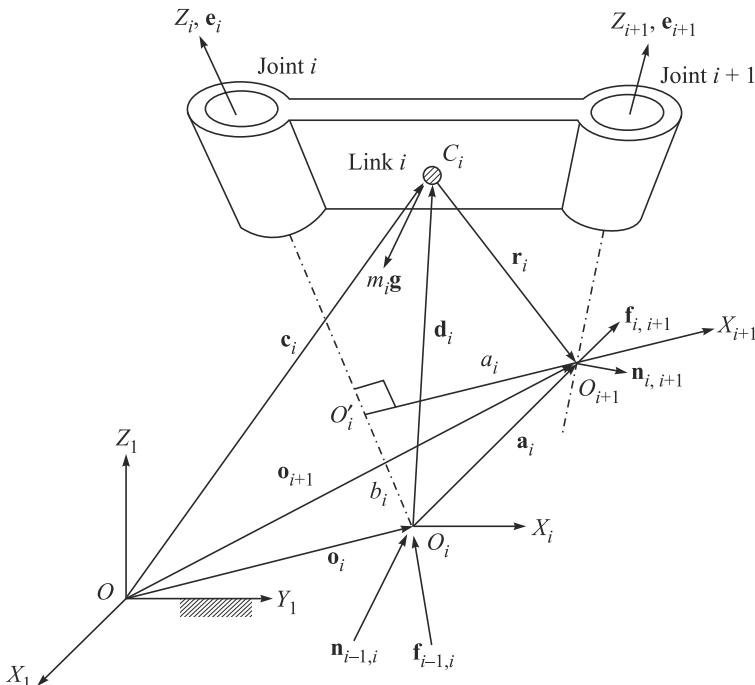


Fig. 8.10 Moments and forces exerted on link i

- $\mathbf{f}_{i-1,i}$: Three-dimensional vector of resulting force exerted on link i by link $i - 1$ at O_i
- $\mathbf{n}_{i-1,i}$: Three-dimensional vector of resulting moment exerted on link i by link $i - 1$ at O_i
- $\mathbf{f}_{i+1,i}$: Three-dimensional vector of resulting force exerted on link i by link $i + 1$ at O_{i+1} such that $\mathbf{f}_{i+1,i} = -\mathbf{f}_{i,i+1}$
- $\mathbf{n}_{i+1,i}$: Three-dimensional vector of resulting moment exerted on link i by link $i + 1$ at O_{i+1} such that $\mathbf{n}_{i+1,i} = -\mathbf{n}_{i,i+1}$
- \mathbf{g} : Three-dimensional vector of acceleration due to gravity
- \mathbf{n}_i : Resultant moment exerted about the center of mass of link i
- \mathbf{f}_i : Resultant force exerted at the center of mass of link i
- \mathbf{I}_i : Inertia matrix of link i about its center of mass C_i
- \mathbf{o}_i : Position vector of the origin of the i th link, as shown in Fig. 8.10
- \mathbf{e}_i : Position vector of the center of mass of the i th link

- \mathbf{r}_i : Position vector of the origin of link $i + 1$ from the center of mass of link i , C_i
- \mathbf{a}_i : Position vector of the origin of the link $i + 1$ from the origin of link i , O_i
- $\dot{\mathbf{o}}_i, \ddot{\mathbf{o}}_i$: Linear velocity and acceleration of the origin O_i
- $\dot{\mathbf{c}}_i, \ddot{\mathbf{c}}_i$: Linear velocity and acceleration of the center of mass of link i
- $\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i$: Angular velocity and acceleration of link i
- \mathbf{e}_i : Unit vector pointing along the Z_i -axis

The forward and backward computations are given next.

8.4.1 Forward Computations

First, the angular velocity, acceleration, and linear velocity and acceleration of each link in terms of its preceding link are evaluated. Those velocities can be computed in a recursive manner, starting at the first moving link and ending at the end-effector link. Since the base link is stationary, its velocities are $\mathbf{v}_0 = \dot{\mathbf{v}}_0 = \boldsymbol{\omega}_0 = \dot{\boldsymbol{\omega}}_0 = \mathbf{0}$.

1. Angular Velocity Propagation Due to the serial construction of the robot manipulator, the angular velocity of link i relative to link $i - 1$ is equal to $\dot{\theta}_i \mathbf{e}_i$ for a revolute joint and $\mathbf{0}$ for a prismatic joint, where \mathbf{e}_i denotes the unit vector pointing along the i th joint axis, Fig. 8.10. Hence, the angular velocity of the link i can be written as

$$\boldsymbol{\omega}_i = \begin{cases} \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i & : \text{for a revolute joint} \\ \boldsymbol{\omega}_{i-1} & : \text{for a prismatic joint} \end{cases} \quad (8.94a)$$

Expressing Eq. (8.94a) in the i th link frame one obtains

$$[\boldsymbol{\omega}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T [\boldsymbol{\omega}_{i-1}]_{i-1} + \dot{\theta}_i [\mathbf{e}_i]_i & : \text{for a revolute joint} \\ \mathbf{Q}_{i-1}^T [\boldsymbol{\omega}_{i-1}]_{i-1} & : \text{for a prismatic joint} \end{cases} \quad (8.94b)$$

where the 3×3 rotation matrix \mathbf{Q}_{i-1}^T represents the orientation of frame i attached to link $i - 1$ with respect to that of $i + 1$ attached to Link i , which is given by

$$\mathbf{Q}_{i-1}^T \equiv \begin{bmatrix} c\theta_{i-1} & s\theta_{i-1} & 0 \\ -s\theta_{i-1}c\alpha_{i-1} & c\theta_{i-1}c\alpha_{i-1} & s\alpha_{i-1} \\ s\theta_{i-1}s\alpha_{i-1} & -c\theta_{i-1}s\alpha_{i-1} & c\alpha_{i-1} \end{bmatrix} \quad (8.95)$$

where $\theta_{i-1}, \alpha_{i-1}$ are the joint and twist angles of the DH parameters defined in Chapter 5, whereas the unit vector $[\mathbf{e}_i]_i = [0, 0, 1]^T$ that points along the i th joint axis is expressed in its coordinate frame. Moreover, $s\theta_{i-1} \equiv \sin \theta_{i-1}$, $c\theta_{i-1} \equiv \cos \theta_{i-1}$, $s\alpha_{i-1} \equiv \sin \alpha_{i-1}$ and $c\alpha_{i-1} \equiv \cos \alpha_{i-1}$.

2. Angular Acceleration Propagation The angular acceleration of link i is obtained by differentiating Eq. (8.94a) with respect to time, i.e.,

$$\dot{\boldsymbol{\omega}}_i = \begin{cases} \dot{\boldsymbol{\omega}}_{i-1} + \ddot{\theta}_i \mathbf{e}_i + \dot{\theta}_i \boldsymbol{\omega}_i \times \mathbf{e}_i & : \text{for a revolute joint} \\ \ddot{\boldsymbol{\omega}}_{i-1} & : \text{for a prismatic joint} \end{cases} \quad (8.96a)$$

Expressing Eq. (8.96a) in the i th link frame, one obtains

$$[\dot{\omega}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T [\dot{\omega}_{i-1}]_{i-1} + \ddot{\theta}_i [\mathbf{e}_i]_i + \dot{\theta}_i [\omega_i]_i \times [\mathbf{e}_i]_i & : \text{for a revolute joint} \\ \mathbf{Q}_{i-1}^T [\dot{\omega}_{i-1}]_{i-1} & : \text{for a prismatic joint} \end{cases} \quad (8.96b)$$

Equation (8.96b) provides a recursive formula for computing the angular acceleration of link i in terms of link $i - 1$.

3. Linear Velocity Propagation Referring to Fig. 8.10, it can be observed that (1) if the i th joint is revolute, link i does not translate along the i th joint axis, and (2) if the i th joint is prismatic, there is a transnational velocity along the i th joint axis. Hence, the velocity of the mass center C_i can be written in terms of the velocity of C_{i-1} as

$$\dot{\mathbf{c}}_i = \begin{cases} \dot{\mathbf{c}}_{i-1} + \omega_{i-1} \times \mathbf{r}_{i-1} + \omega_i \times \mathbf{d}_i & : \text{for a revolute joint} \\ \dot{\mathbf{c}}_{i-1} + \omega_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i) + \dot{b}_i \mathbf{e}_i & : \text{for a prismatic joint} \end{cases} \quad (8.97a)$$

Equation (8.97a) can be written in the i th frame as

$$[\dot{\mathbf{c}}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T ([\dot{\mathbf{c}}_{i-1}]_{i-1} + [\omega_{i-1}]_{i-1} \times [\mathbf{r}_{i-1}]_{i-1}) + [\omega_i]_i \times [\mathbf{d}_i]_i & : \text{for a revolute joint} \\ \mathbf{Q}_{i-1}^T ([\dot{\mathbf{c}}_{i-1}]_{i-1} + [\omega_{i-1}]_{i-1} \times [\mathbf{r}_{i-1}]_{i-1}) + [\omega_i]_i \times [\mathbf{d}_i]_i + \dot{b}_i [\mathbf{e}_i]_i & : \text{for a prismatic joint} \end{cases} \quad (8.97b)$$

where $[\omega_i]_i = \mathbf{Q}_{i-1}^T [\omega_{i-1}]_{i-1}$ for the prismatic joint. Moreover, $[\mathbf{d}_i]_i = [\mathbf{a}_i]_i - [\mathbf{r}_i]_i$ in which

$$[\mathbf{a}_i]_i \equiv \begin{bmatrix} a_i c \theta_i \\ a_i s \theta_i \\ b_i \end{bmatrix}; \quad \text{and} \quad [\mathbf{r}_i]_{i+1} \equiv \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix} \quad (8.97c)$$

r_{ix} , r_{iy} , and r_{iz} being the constant components of the vector \mathbf{r}_i in frame $i + 1$ which is attached to link i . Equation (8.97b) is the recursive formula for computing the linear velocity of link i in terms of that of link $i - 1$.

4. Linear Acceleration Propagation Linear acceleration of the mass center of link i can be obtained by differentiating Eq. (8.97a) with respect to time as

$$\ddot{\mathbf{c}}_i = \begin{cases} \ddot{\mathbf{c}}_{i-1} + \dot{\omega}_{i-1} \times \mathbf{r}_{i-1} + \omega_{i-1} \times (\omega_{i-1} \times \mathbf{r}_{i-1}) + \dot{\omega}_i \times \mathbf{d}_i + \omega_i \times (\omega_i \times \mathbf{d}_i) & : \text{Revolute} \\ \ddot{\mathbf{c}}_{i-1} + \dot{\omega}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i) + \omega_{i-1} \times [\omega_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i)] + \ddot{b}_i \mathbf{e}_i + 2\dot{b}_i \omega_i \times \mathbf{e}_i & : \text{Prismatic} \end{cases} \quad (8.98a)$$

where in deriving Eq. (8.98a) for the prismatic joint, $\omega_i = \omega_{i-1}$ and $\dot{\omega}_i = \dot{\omega}_{i-1}$ are used. Expressing Eq. (8.98a) in the i th frame one gets

$$[\ddot{\mathbf{c}}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T [\ddot{\mathbf{c}}_{i-1} + \dot{\omega}_{i-1} \times \mathbf{r}_{i-1} + \omega_{i-1} \times (\omega_{i-1} \times \mathbf{r}_{i-1})]_{i-1} + [\dot{\omega}_i \times \mathbf{d}_i + \omega_i \times (\omega_i \times \mathbf{d}_i)] & : \text{Revolute} \\ \mathbf{Q}_{i-1}^T [\ddot{\mathbf{c}}_{i-1} + \dot{\omega}_{i-1} \times \mathbf{r}_{i-1} + \omega_{i-1} \times (\omega_{i-1} \times \mathbf{r}_{i-1})]_{i-1} + [\dot{\omega}_i \times \mathbf{d}_i + \omega_i \times (\omega_i \times \mathbf{d}_i) + \ddot{b}_i \mathbf{e}_i + 2\dot{b}_i \omega_i \times \mathbf{e}_i]_i & : \text{Prismatic} \end{cases} \quad (8.98b)$$

where all the vector quantities inside $[...]$ are represented in the frame j , for $j = i-1$ and i . Equation (8.98b) is the recursive formula for computing the linear acceleration of link i in terms of link $i-1$.

5. Acceleration due to Gravity Finally, the acceleration due to gravity from the $(i-1)$ st frame to the i th frame can be transformed as follows:

$$[\mathbf{g}]_i = \mathbf{Q}_{i-1}^T [\mathbf{g}]_{i-1} \quad (8.99)$$

8.4.2 Backward Computations

Once the velocities and accelerations of all the links are found using the forward computations of Section 8.4.1, the joint torques and forces can be computed one at a time starting from the end-effector link and ending at the base link. First, apply Eqs. (8.89) and (8.92) to compute the required force and moment to be exerted at and about the center of mass of link i , respectively. They are as follows:

$$[\mathbf{f}_i]_i = m_i [\ddot{\mathbf{c}}_i]_i \quad (8.100a)$$

$$[\mathbf{n}_i]_i = [\mathbf{I}_i]_i [\dot{\boldsymbol{\omega}}_i]_i + [\boldsymbol{\omega}_i]_i \times [\mathbf{I}_i]_i [\boldsymbol{\omega}_i]_i \quad (8.100b)$$

Next, the force and moment balance equations about the center of mass of link i are written. Referring to Fig. 8.10,

$$[\mathbf{f}_i]_i = [\mathbf{f}_{i-1,i}]_i - [\mathbf{f}_{i,i+1}]_i + m_i [\mathbf{g}]_i \quad (8.101a)$$

$$[\mathbf{n}_i]_i = [\mathbf{n}_{i-1,i}]_i - [\mathbf{n}_{i,i+1}]_i - [\mathbf{d}_i]_i \times [\mathbf{f}_{i-1,i}]_i - [\mathbf{r}_i]_i \times [\mathbf{f}_{i,i+1}]_i \quad (8.101b)$$

Rearranging Eqs. (8.101a-b) in recursive forms, one gets

$$[\mathbf{f}_{i-1,i}]_i = [\mathbf{f}_i]_i + [\mathbf{f}_{i,i+1}]_i - m_i [\mathbf{g}]_i \quad (8.101c)$$

$$[\mathbf{n}_{i-1,i}]_i = [\mathbf{n}_i]_i + [\mathbf{n}_{i,i+1}]_i + [\mathbf{d}_i]_i \times [\mathbf{f}_{i-1,i}]_i + [\mathbf{r}_i]_i \times [\mathbf{f}_{i,i+1}]_i \quad (8.101d)$$

Equations (8.101c-d) can be used to solve for $[\mathbf{f}_{i-1,i}]_i$ and $[\mathbf{n}_{i-1,i}]_i$ recursively, starting from the end-effector link. For the end-effector link, the moment and force, $[\mathbf{n}_{n,n+1}]_{n+1}$ and $[\mathbf{f}_{n,n+1}]_{n+1}$, respectively, represent the external moment and force applied by the end-effector on the environment, as done in Chapter 7. They are assumed to be known.

8.4.3 Joint Torque or Force Expressions

Actuator torque or force, denoted with τ_i , is obtained by projecting the moment or force of constraint obtained in Eqs. (8.101c-d) onto their corresponding joint axes, i.e.,

$$\tau_i = \begin{cases} [\mathbf{e}_i]^T [\mathbf{n}_{i-1,i}]_i : & \text{for a revolute joint} \\ [\mathbf{e}_i]^T [\mathbf{f}_{i-1,i}]_i : & \text{for a prismatic joint} \end{cases} \quad (8.102)$$

Equation (8.102) states that τ_i is nothing but the last element of the vector $[\mathbf{n}_{i-1,i}]_i$ or $[\mathbf{f}_{i-1,i}]_i$.

Meaning of Recursive

Link velocities, forces, etc., are computed for a link after knowing the same for its previous or proceeding link.

Example 8.13 NE Equations for One-link One-DOF Planar Arm

Consider the planar one-DOF manipulator shown in Fig. 8.7 whose free-body diagram is shown in Fig. 8.11. The rotation matrix representing the orientation of frame 2 with respect to frame 1 is

$$\mathbf{Q} \equiv \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8.103)$$

where $s \equiv \sin \theta$ and $c \equiv \cos \theta$. Moreover, the vector \mathbf{a}_1 in its own frame is as follows:

$$[\mathbf{a}]_1 \equiv [ac \quad as \quad 0]^T \quad (8.104)$$

Next, assuming the links are homogeneous, the vectors, $[\mathbf{d}]_1$ and $[\mathbf{r}]_1$, are given as

$$[\mathbf{d}]_1 \equiv \left[\frac{1}{2}ac \quad \frac{1}{2}as \quad 0 \right]^T; \text{ and } [\mathbf{r}]_1 = \left[\frac{a}{2} \quad 0 \quad 0 \right]^T \quad (8.105)$$

Let the link be a square beam of relatively small cross-sectional area. Then, the inertia matrix of the link about its center of mass C represented in a frame attached to it, i.e., Frame 2, is given by

$$\mathbf{I} = \frac{ma^2}{12} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8.106)$$

Applying the Newton–Euler recursive algorithm to calculate the link velocities and accelerations, and then forces and moments, recursively, one gets the following:

(a) *Forward Computations:* First, the velocities and accelerations of the link are computed by taking into the fact that it is attached to the fixed base whose angular and linear velocities are zeros, i.e.,

$$[\boldsymbol{\omega}]_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}; \text{ and } [\dot{\boldsymbol{\omega}}]_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix} \quad (8.107a)$$

$$[\dot{\mathbf{c}}]_1 = \dot{\theta} \frac{a}{2} \begin{bmatrix} -s \\ c \\ 0 \end{bmatrix}; \text{ and } [\ddot{\mathbf{c}}]_1 = \ddot{\theta} \frac{a}{2} \begin{bmatrix} -s \\ c \\ 0 \end{bmatrix} - \dot{\theta}^2 \frac{a}{2} \begin{bmatrix} c \\ s \\ 0 \end{bmatrix} \quad (8.107b)$$

It is assumed that the acceleration due to gravity points in the X_1 -direction of the first frame, i.e., $[\mathbf{g}]_1 \equiv [g \quad 0 \quad 0]^T$.

(b) *Backward Computations:* For backward computation, first the forces exerted on the link are obtained. Assuming that there are no externally applied forces, i.e., $[\mathbf{f}_{12}]_2 = [\mathbf{n}_{12}]_2 = \mathbf{0}$, the forward calculations for $i = 2, 1$ are as follows:

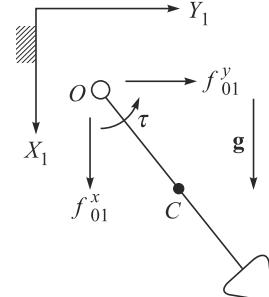


Fig. 8.11 Free-body diagram of one-link arm

$$[\mathbf{f}]_l = m \frac{a}{2} \begin{pmatrix} \ddot{\theta} \begin{bmatrix} -s \\ c \\ 0 \end{bmatrix} - \dot{\theta}^2 \begin{bmatrix} c \\ s \\ 0 \end{bmatrix} \end{pmatrix}; [\mathbf{n}]_l = m \frac{a^2}{12} \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix} \quad (8.108)$$

$$[\mathbf{f}_{01}]_l = \begin{bmatrix} -m \frac{a}{2} (\ddot{\theta}s + \dot{\theta}^2 c) - mg \\ m \frac{a}{2} (\ddot{\theta}c - \dot{\theta}^2 s) \\ 0 \end{bmatrix}; [\mathbf{n}_{01}]_l = \begin{bmatrix} 0 \\ 0 \\ \frac{ma^2}{12} \ddot{\theta} + \frac{ma^2}{4} \ddot{\theta} + \frac{1}{2} mgas \end{bmatrix} \quad (8.109)$$

(c) *Joint Torque Computation:* Finally, applying Eq. (8.102) to compute the required joint torque, the only dynamical equation is obtained as

$$\tau = \frac{1}{3} ma^2 \ddot{\theta} + \frac{1}{2} mgas \quad (8.110)$$

which is again same as in Eq. (8.55) or (8.56b).

Example 8.14 NE Equations for Two-link Robot Arm

Consider the planar two-DOF robot arm shown in Fig. 8.5. The associated rotation matrices are

$$\mathbf{Q}_1 \equiv \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \text{ and } \mathbf{Q}_2 \equiv \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8.111)$$

Vectors \mathbf{a}_1 and \mathbf{a}_2 in their own frames are as follows:

$$[\mathbf{a}_1]_1 \equiv [a_1 c_1 \quad a_1 s_1 \quad 0]^T; \text{ and } [\mathbf{a}_2]_2 \equiv [a_2 c_2 \quad a_2 s_2 \quad 0]^T \quad (8.112)$$

Moreover, assuming that the links are homogeneous, the vectors $[\mathbf{d}_i]_i$ and $[\mathbf{r}_i]_{i+1}$, for $i = 1, 2$, are given by

$$[\mathbf{d}_i]_i \equiv \left[\frac{1}{2} a_i c_i \quad \frac{1}{2} a_i s_i \quad 0 \right]^T; \text{ and } [\mathbf{r}_i]_{i+1} = \left[\frac{a_i}{2} \quad 0 \quad 0 \right]^T \quad (8.113)$$

Let the two links be cylindrical rods of relatively small diameter. Then, the inertia matrix of link i about its center of mass C_i represented in a frame attached to it, i.e., frame $i+1$, is given by

$$[\mathbf{I}_i]_{i+1} = \frac{m_i a_i^2}{12} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \text{ for } i=1, 2 \quad (8.114)$$

Applying the Newton–Euler recursive algorithm to calculate the link velocities and accelerations, and then forces and moments, recursively, one gets the following:

(a) *Forward Computations:* First, the velocities and accelerations of link 1 are computed. Since link 1 is attached to the fixed base whose angular and linear velocities are zeros, its velocities are given by

$$[\boldsymbol{\omega}_1]_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}; \text{ and } [\dot{\boldsymbol{\omega}}_1]_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} \quad (8.115a)$$

$$[\dot{\mathbf{c}}_1]_1 = \dot{\theta}_1 \frac{a_1}{2} \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}; \text{ and } [\ddot{\mathbf{c}}_1]_1 = \ddot{\theta}_1 \frac{a_1}{2} \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} - \dot{\theta}_1^2 \frac{a_1}{2} \begin{bmatrix} c_1 \\ s_1 \\ 0 \end{bmatrix} \quad (8.115b)$$

It is assumed that the acceleration of gravity points in the negative Y_1 -direction of the first frame, i.e.,

$$[\mathbf{g}]_1 \equiv [0 \ -g \ 0]^T$$

Next, the velocities and accelerations of link 2 are computed. Substituting the velocities and acceleration of link 1 into the Eqs. (8.94b), (8.96b), (8.97b) and (8.98b),

$$[\boldsymbol{\omega}_2]_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix}; \text{ and } [\dot{\boldsymbol{\omega}}_2]_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 + \ddot{\theta}_2 \end{bmatrix} \quad (8.116a)$$

$$[\dot{\mathbf{c}}_2]_2 = \begin{bmatrix} -\frac{a_2}{2} s_2 \dot{\theta}_{12} \\ a_1 \dot{\theta}_1 + \frac{a_2}{2} c_2 \dot{\theta}_{12} \\ 0 \end{bmatrix}; \text{ and } [\ddot{\mathbf{c}}_2]_2 = \begin{bmatrix} -a_1 \dot{\theta}_1^2 - \frac{a_2}{2} (s_2 \ddot{\theta}_{12} + c_2 \dot{\theta}_{12}^2) \\ a_1 \ddot{\theta}_1 + \frac{a_2}{2} (c_2 \ddot{\theta}_{12} - s_2 \dot{\theta}_{12}^2) \\ 0 \end{bmatrix} \quad (8.116b)$$

where $\dot{\theta}_{12} \equiv \dot{\theta}_1 + \dot{\theta}_2$, $\ddot{\theta}_{12} \equiv \ddot{\theta}_1 + \ddot{\theta}_2$. The acceleration due to gravity expressed in frame 2 is $[\mathbf{g}]_2 = \mathbf{Q}^T_1 [\mathbf{g}]_1 = [-gs_1 \ -gc_1 \ 0]^T$.

(b) *Backward Computations:* For backward computations, first the forces exerted on link 2 and then link 1 are obtained. Assuming that there are no externally applied forces, i.e., $[\mathbf{f}_{32}]_3 = [\mathbf{n}_{32}]_3 = \mathbf{0}$. Since $\mathbf{f}_{23} = \mathbf{f}_{32}$ and $\mathbf{n}_{23} = \mathbf{n}_{32}$, it is obvious that $[\mathbf{f}_{23}]_3 = [\mathbf{n}_{23}]_3 = \mathbf{0}$. Hence, substituting the velocities and accelerations of link 2 obtained from the forward calculations into Eqs. (8.100–8.101), for $i = 2$, one finds

$$[\mathbf{f}_2]_2 = m_2 \begin{bmatrix} -a_1 \dot{\theta}_1^2 - \frac{a_2}{2} (s_2 \ddot{\theta}_{12} + c_2 \dot{\theta}_{12}^2) \\ a_1 \ddot{\theta}_1 + \frac{a_2}{2} (c_2 \ddot{\theta}_{12} - s_2 \dot{\theta}_{12}^2) \\ 0 \end{bmatrix}; \quad [\mathbf{n}_2]_2 = m_2 \frac{a_2^2}{12} \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_{12} \end{bmatrix} \quad (8.117)$$

$$[\mathbf{f}_{12}]_2 = m_2 \begin{bmatrix} -a_1 \dot{\theta}_1^2 - \frac{a_2}{2} (s_2 \ddot{\theta}_{12} + c_2 \dot{\theta}_{12}^2) + gs_1 \\ a_1 \ddot{\theta}_1 + \frac{a_2}{2} (c_2 \ddot{\theta}_{12} - s_2 \dot{\theta}_{12}^2) + gc_1 \\ 0 \end{bmatrix} \equiv \begin{bmatrix} f_{21}^x \\ f_{21}^y \\ 0 \end{bmatrix} \quad (8.118a)$$

$$[\mathbf{n}_{12}]_2 = \begin{bmatrix} 0 \\ 0 \\ -\frac{a_2}{2}s_2f_{12}^x + \frac{a_2}{2}c_2f_{12}^y + m_2 \frac{a_2^2}{12}\ddot{\theta}_{12} \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ n_{12}^z \end{bmatrix} \quad (8.118b)$$

Substituting the forgoing forces and moments obtained for Link 2 along with the velocities and accelerations of link 1 into Eqs. (8.100a-b) and (8.101c-d) for, $i = 1$, the following are obtained:

$$[\mathbf{f}_1]_1 = m_1 \frac{a_1}{2} \left(\ddot{\theta}_1 \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} - \dot{\theta}_1^2 \begin{bmatrix} c_1 \\ s_1 \\ 0 \end{bmatrix} \right); [\mathbf{n}_1]_1 = m_1 \frac{a_1^2}{12} \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} \quad (8.119)$$

$$[\mathbf{f}_{01}]_1 = \begin{bmatrix} c_1 f_{12}^x - s_1 f_{12}^y - m_1 \frac{a_1}{2} (s_1 \ddot{\theta}_1 + c_1 \dot{\theta}_1^2) \\ s_1 f_{12}^x + c_1 f_{12}^y + m_1 \frac{a_1}{2} (c_1 \ddot{\theta}_1 - s_1 \dot{\theta}_1^2) + m_1 g \\ 0 \end{bmatrix} \quad (8.120a)$$

$$[\mathbf{n}_{01}]_1 = \begin{bmatrix} 0 \\ 0 \\ \frac{m_1 a_1^2}{12} \ddot{\theta}_1 + n_{12}^z + a_1 f_{12}^y + m_1 g \frac{a_1}{2} c_1 \end{bmatrix} \quad (8.120b)$$

(c) *Joint-torque Computations:* Finally, applying Eq. (8.102) to compute the required joint torques, two dynamic equations of motion are obtained as

$$\begin{aligned} \tau_1 &= \frac{m_1 a_1^2}{3} \ddot{\theta}_1 + \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1^2 + \frac{m_2 a_2^2}{3} \ddot{\theta}_{12} + m_2 g \left(\frac{a_2}{2} c_{12} + a_1 c_1 \right) + \\ &\quad m_1 g \frac{a_1}{2} c_1 + m_2 \frac{a_1 a_2}{2} c_2 \dot{\theta}_1 + m_2 a_1 \left[a_1 \ddot{\theta}_1 + \frac{a_2}{2} (c_2 \ddot{\theta}_{12} - s_2 \dot{\theta}_{12}^2) \right] \end{aligned} \quad (8.121a)$$

$$\tau_2 = \frac{m_2 a_2^2}{3} \ddot{\theta}_{12} + \frac{m_2 a_1 a_2}{2} (c_2 \ddot{\theta}_1 + s_2 \dot{\theta}_1^2) + \frac{m_2 a_2}{2} g c_{12} \quad (8.121b)$$

As expected, the dynamical equation for the second joint is simpler than the first joint. The various dynamical effects, including the Coriolis and centrifugal velocity coupling, and gravitational effects are demonstrated clearly in this example. It can be seen from Eqs. (8.121a-b) that the dynamical model is fairly complex even for a simple two-DOF robot arm. Using the Newton–Euler recursive method, all the joint reaction forces are also found, namely, those in Eqs. (8.118) and (8.120).

Simulation

It is a step together with forward dynamics to obtain the joint rates and position through (numerical) integration of the joint accelerations.

Equations (8.121a-b) can be verified to be same as those obtained in Example 8.9 using Euler–Lagrange formulation.

8.5 DYNAMIC ALGORITHMS

In dynamics of robots, generally, two types of problems are commonly solved. One is called *inverse dynamics*, where given the robot geometric and inertial parameters, and the joint motions, i.e., its positions, velocities, and accelerations, corresponding joint torques and forces are calculated. The second one is called *forward* or *direct dynamics* in which given the geometric and inertial parameters, and the joint torques and forces, joint accelerations are obtained. While inverse dynamics is useful for robot control, forward dynamics is essential in simulation. Note from the equations of motion given by Eq. (8.44a) that inverse dynamics requires the evaluation of its left-hand side, whereas the forward dynamics requires the solution of the joint accelerations from the linear algebraic equations, which are nothing but the dynamic equations of Eq. (8.44a) with the known values for the joint angles and rates. It will be further explained in Section 8.5.2.

8.5.1 Inverse Dynamics

In the control of a robot, particularly, those based on robot dynamics, actuator torques and forces are calculated using inverse dynamics. For this, one can use either Eq. (8.44a) where the associated matrices and vectors evaluated in Section 8.2.4 or one can use the Recursive Newton–Euler (RNE) algorithm given in Section 8.4. For complex robotic systems, e.g., six-DOF industrial robot, one generally prefers RNE as illustrated in Example 8.14.

Example 8.15 Inverse Dynamics of One-link Planar Arm Using MATLAB

For the one-link robot arm shown in Fig. 8.7, consider, $a = 1$, $m = 1$, and the joint angle θ being varied as

$$\theta = \theta(0) + \frac{\theta(T) - \theta(0)}{T} \left[t - \frac{T}{2\pi} \sin\left(\frac{2\pi}{T}t\right) \right] \quad (8.122a)$$

where $\theta_i(0) = 0$; $\theta_i(T) = \pi$ and $T = 10$ sec. Using Eq. (8.122a) the joint rate and acceleration expressions are given by

$$\dot{\theta} = \frac{\theta(T) - \theta(0)}{T} \left[1 - \cos\left(\frac{2\pi}{T}t\right) \right], \text{ and } \ddot{\theta} = \frac{\theta(T) - \theta(0)}{T} \left[\frac{2\pi}{T} \sin\left(\frac{2\pi}{T}t\right) \right] \quad (8.122b)$$

Note the trajectory in Eq. (8.122b) is so chosen that the joint rates and accelerations at the beginning and end are zeros, i.e., $\dot{\theta}(0) = \dot{\theta}(T) = 0$ and $\ddot{\theta}(0) = \ddot{\theta}(T) = 0$. Using the MATLAB program shown in Fig. 8.12, the joint motions and the corresponding joint torques are evaluated based on Eqs. (8.110) and (8.122). The values are plotted in Figs. 8.13–8.14, where, ‘th’, ‘thd’, and ‘thdd’ represent the variables θ , $\dot{\theta}$, $\ddot{\theta}$, respectively.

```
% Input for trajectory and link parameters
T = 10; thT = pi; th0 = 0; m = 1; a = 1; g = 9.81;
con = 2*pi/T; delth = thT - th0;
iner = m*a*a/3; grav = m*g*a/2;
for i = 1:51,
    ti (i) = (i-1)*T/50;
    ang = con*ti(i);
% Joint trajectory
th (i) = th0 + (delth/T)*(ti (i) - sin(ang)/con);
thd (i) = delth*(1 - cos(ang))/T;
thdd (i) = delth*con*sin(ang)/T;
% Joint torque
tau (i)= iner*thdd (i) + grav*sin(th(i));
end
plot (ti,th,'-',ti,thd,:',ti,thdd,'-.')
figure
plot (ti, tau)
```

Fig. 8.12 MATLAB program ‘ch8idyn1.m’ for one-link arm

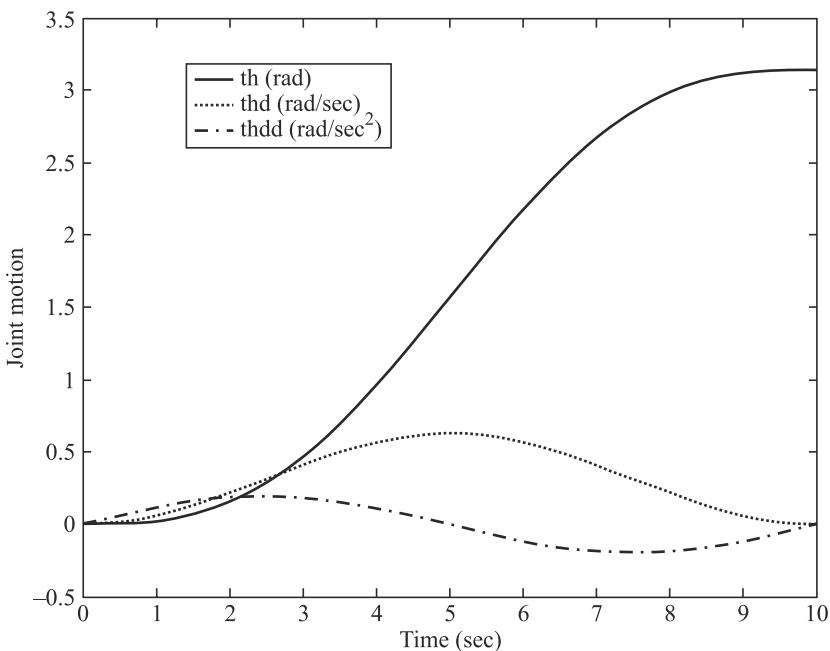
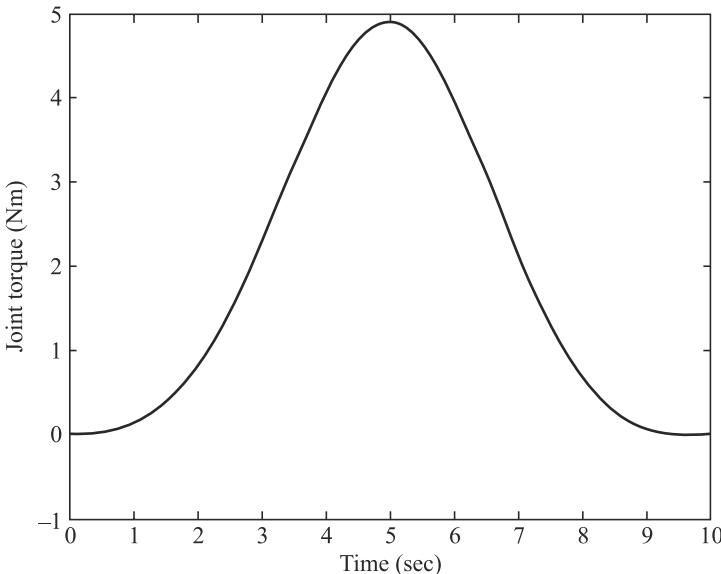


Fig. 8.13 Joint trajectory

**Fig. 8.14** Joint torque for the one-link arm

Example 8.16 Inverse Dynamics for the Two-link Robot Arm Using MATLAB

For the two-link robot arm or manipulator shown in Fig. 8.5, consider $a_1 = a_2 = 1$ m, and $m_1 = m_2 = 1$ kg, and the joint-angle variations for both the joints θ_1 and θ_2 are taken same as Eq. (8.122) except that their end points are different, i.e., $\theta_1(T) = \pi$ and $\theta_2(T) = \pi/2$. Using the MATLAB program shown in Fig. 8.15, the joint angles and torques are shown in Figs. 8.16 and 8.17, respectively.

```
clear all
% Input for trajectory and link parameters
T = 10; th1T = pi; th10 = 0; th2T = pi/2; th20 = 0;
m1 = 1; m2 = 1; a1 = 1; a2 = 1; g = 9.81;
con = 2*pi/T; delth1 = th1T - th10; delth2 = th2T - th20; iner21 =
m2*a1*a2;

for i = 1:51,
    ti (i) = (i-1)*T/50;      ang = con*ti(i);

    % Joint trajectory
    th1 (i) = th10 + (delth1/T)*(ti (i) - sin(ang)/con);
    th1d (i) = delth1*(1 - cos(ang))/T;    th1dd (i) =
    delth1*con*sin(ang)/T;

    th2 (i) = th20 + (delth2/T)*(ti (i) - sin(ang)/con);
    th2d (i) = delth2*(1 - cos(ang))/T;    th2dd (i) =
    delth2*con*sin(ang)/T;
```

(Contd.)

```

thdd = [th1dd(i);th2dd(i)];

%Inertia matrix
sth2 = sin(th2(i)); cth2 = cos(th2(i));
i22 = m2*a2*a2/3;i21 = i22 + iner21*cth2/2;i12 = i21;
i11 = i22 + m1*a1*a1/3 + m2*a1*a1 + iner21*cth2;
im = [i11, i12; i21, i22]

%h-vector
h1 = - (m2*a1*a2*th1d(i) + iner21/2*th2d(i))*th2d(i)*sth2;
h2 = iner21/2*sth2*th1d(i)*th1d(i); hv=[h1;h2]

%gamma-vector
cth1 = cos(th1(i)); cth12 = cos(th1(i) + th2(i));
gam1 = m1*g*a1/2*cth1 + m2*g*(a1*cth1 + a2/2*cth12);
gam2 = m1*g*a2/2*cth12; gv = [gam1;gam2]

% Joint torque
tau=im*thdd + hv + gv; tor1(i) = tau(1); tor2(i)=tau(2);
end
plot(ti,th1,'-',ti,th2,:')
figure
plot (ti, tor1,'-',ti,tor2,:')

```

Fig. 8.15 MATLAB program ‘ch8idyn2.m’ for two-link robot arm

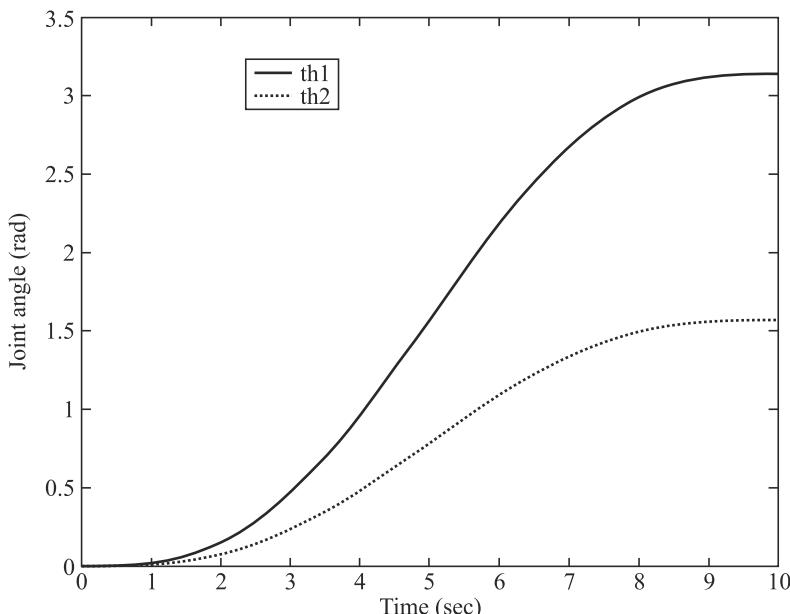


Fig. 8.16 Joint angles for the two-link robot arm

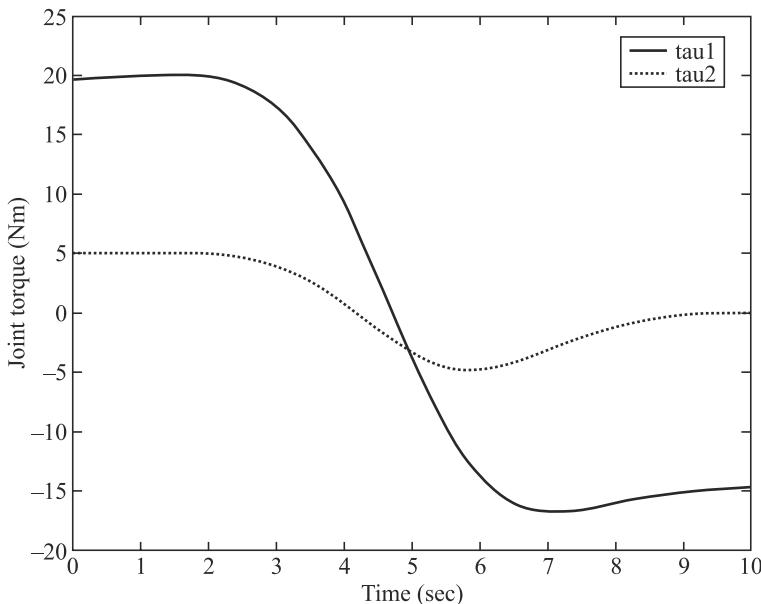
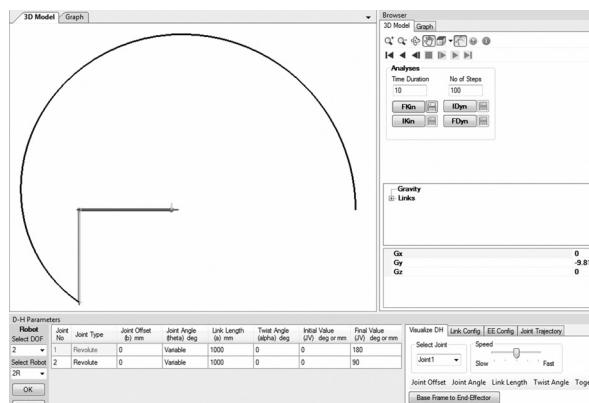


Fig. 8.17 Joint torques for the two-link robot arm

Example 8.17 Inverse Dynamics for the Two-link Robot Arm Using RoboAnalyzer¹

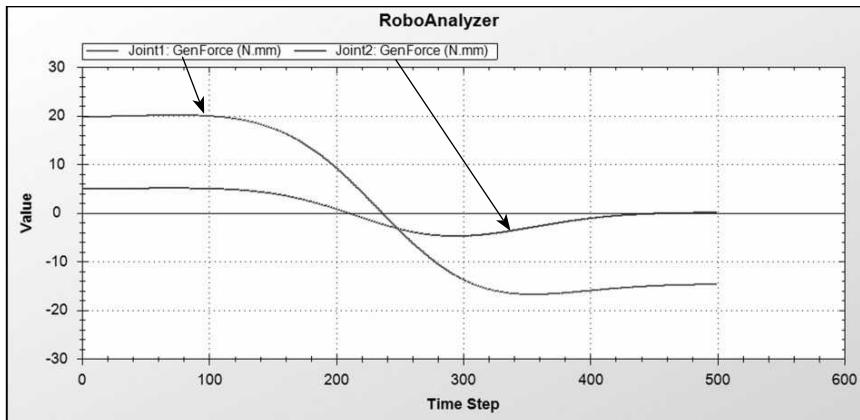
One may also use the RoboAnalyzer software developed at IIT Delhi for the inverse dynamics analysis of the two-link robot arm of Example 8.16. While the screenshot in Fig. 8.18(a) shows the robot with its DH parameters, Fig. 8.18(b) depicts the joint torques. The latter are same as Fig. 8.17 for the same input motions of Fig. 8.16.



(a) Two-link robot arm with traced path

(Contd.)

¹The dynamic algorithm of RoboAnalyzer (RA) software actually uses the concept of the Decoupled Natural Orthogonal Complement (DeNOC) matrices introduced in Chapter 6. The DeNOC can be used to derive the dynamic equations of motion which is explained in Chapter 9. The inverse dynamics results using RA are generated in this example to verify the results.



(b) Inverse dynamics results for the two-link robot arm

Fig. 8.18 Inverse dynamics results for the two-link robot arm using RoboAnalyzer

8.5.2 Forward Dynamics and Simulation

Simulation of a robot helps in understanding the behavior of a robot even though it may not exist in reality. It involves what is known as forward or direct dynamics, followed by the solution of the equations of motion. Forward dynamics is defined as “given the kinematic and inertial parameters and the joint torques and forces as functions of time, find the trajectory of the robot manipulator.” In other words, for known vector τ of Eq. (8.44a) find the joint angles, denoted with vector θ . The forward dynamics problem requires the solution of the joint accelerations $\ddot{\theta}$ from the equations of motion which is algebraic in nature for known values of joint angles and rates, i.e., θ and $\dot{\theta}$, respectively. This is given by

$$\ddot{\theta} = \mathbf{I}^{-1}(\tau - \mathbf{h} - \gamma) \quad (8.123)$$

Since inertia matrix \mathbf{I} is symmetric positive definite, it is always invertible. Moreover, Eq. (8.123) also represents a set of differential equations in θ which are highly non-linear, and cannot be solved or integrated analytically except in the simplest cases. Hence, obtaining θ for given τ and the initial conditions requires numerical integrations. MATLAB software has in-built routines like ODE45 and others to perform the numerical integration of the differential equations that are arranged in state-space form. A state-space form of Eq. (8.123) is obtained by defining the following $2n$ -dimensional state-vector $\mathbf{y}(t)$:

$$\mathbf{y}(t) \equiv [\mathbf{y}_1^T(t), \mathbf{y}_2^T(t)]^T, \text{ where } \mathbf{y}_1(t) = \theta; \mathbf{y}_2(t) = \dot{\theta} \quad (8.124a)$$

and t is time. The state-space form for the equations of motion, Eq. (8.123), is then expressed as

$$\dot{\mathbf{y}}(t) \equiv \begin{bmatrix} \dot{\mathbf{y}}_1(t) \\ \dot{\mathbf{y}}_2(t) \end{bmatrix} = \begin{bmatrix} \mathbf{y}_2(t) \\ \mathbf{I}^{-1}(\tau - \mathbf{h} - \gamma) \end{bmatrix} \quad (8.124b)$$

Example 8.18 Simulation of One-link Arm using MATLAB and MuPAD

For the one-link arm, joint acceleration $\ddot{\theta}$ is obtained from Eq. (8.55) or (8.110) as

$$\ddot{\theta} = \frac{3}{ma^2} \left(\tau - \frac{1}{2} mgas \right) \quad (8.125)$$

where $s = \sin \theta$. Hence, the state-space form is given by

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= \frac{3}{ma^2} \left(\tau - \frac{1}{2} mgas \right) \end{aligned} \quad (8.126)$$

where for brevity time t is not explicitly shown in Eq. (8.126). Moreover, the two-dimensional state-space vector is given by, $\mathbf{y} \equiv [y_1 \ y_2]^T = [\theta \ \dot{\theta}]^T$. Now, for the input of no joint torque, i.e., $\tau = 0$, acceleration due to gravity, $g = 9.81 \text{ m/s}^2$ and the initial values of the state-vector at time $t = 0$ as $\mathbf{y}(0) \equiv [\pi/2 \ 0]^T$, the numerical integration was performed using ODE45 of MATLAB. The MATLAB programs to generate the simulation results are shown in Fig. 8.19, whereas the results are shown in Fig. 8.20. It is clear from Fig. 8.20 that the arm is behaving like a solid pendulum, which is actually true when no joint torque is applied and the link moves due to gravity only.

```
%For one-link arm
function ydot =ch8fdyn1(t,y);
m = 1; a = 1; g = 9.81; tau=0;
iner = m*a*a/3; grav = m*g*a/2;
y d o t = [ y ( 2 ) ; ( t a u -
grav*sin(y(1)))/iner];
```

(a) Program for state-space form

```
%For one link arm
tspan=[0 10]; y0=[pi/2;
0];
[t,y]=ode45('ch8fdyn1',t
span,y0);
```

(b) Program to integrate numerically

Fig. 8.19 Simulation of one-link arm under gravity only

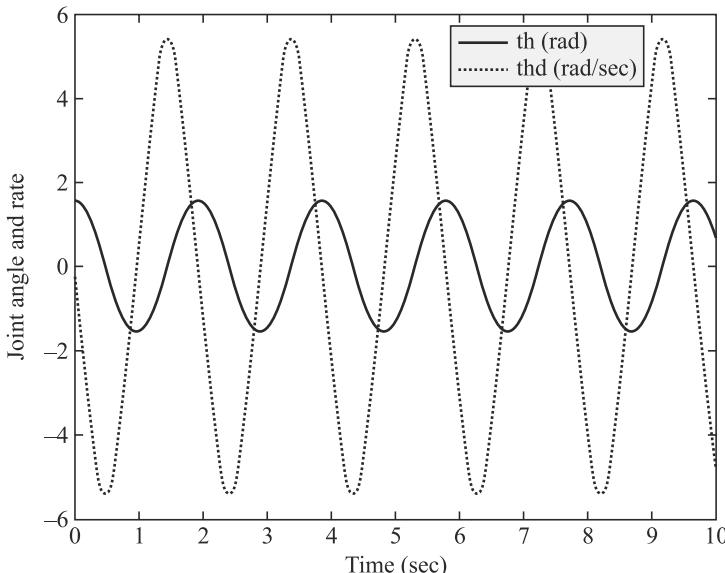


Fig. 8.20 Simulation results of one-link arm under gravity

The simulation results given by Fig. 8.20 can also be verified using symbolic computation code of MuPAD, as depicted in Fig. 8.21.

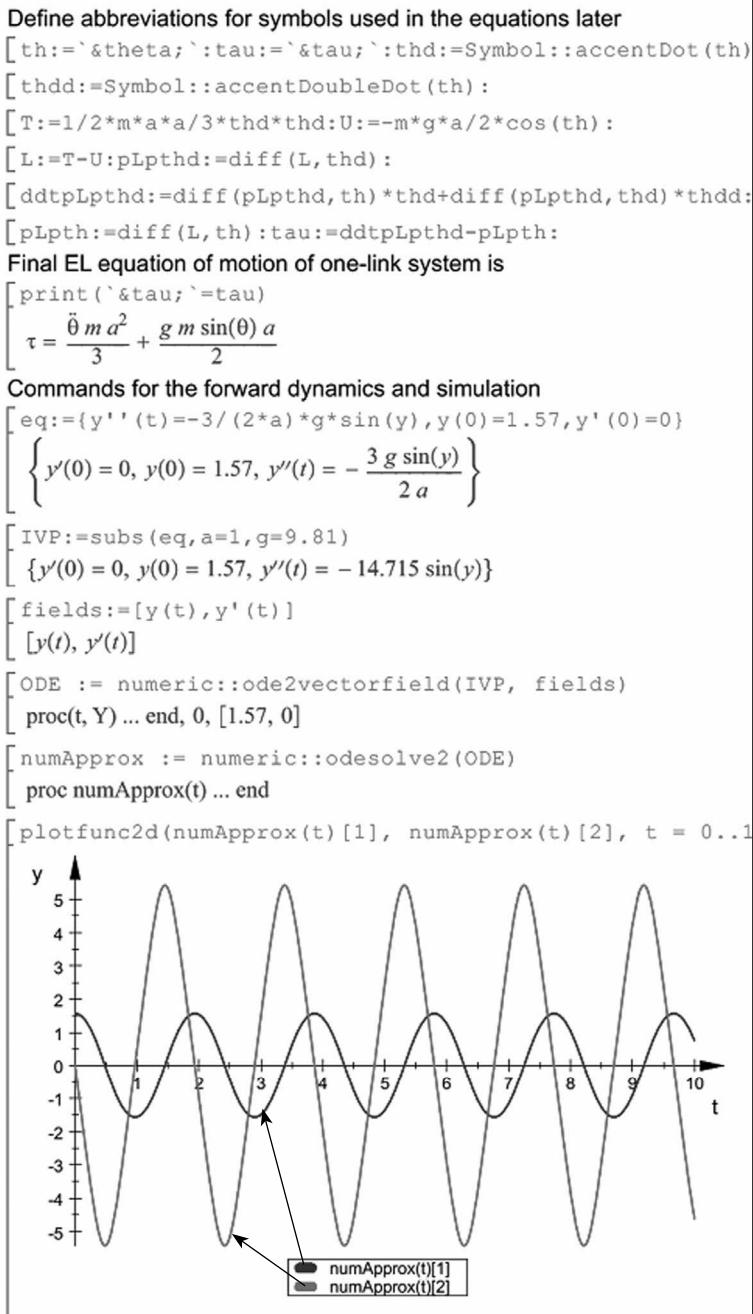


Fig. 8.21 Simulation results of one-link arm in MuPAD

Example 8.19 Simulation of Two-link Robot Arm using MATLAB

For the two-link robot arm, the two-dimensional joint acceleration vector $\ddot{\theta}$ can be algebraically solved in the form of Eq. (8.123) where the associated inertia matrix and the vectors are given in Eqs. (8.59)-(8.61). The state-space form is then obtained as

$$\begin{aligned}\dot{\mathbf{y}}_1 &= \mathbf{y}_2 \\ \dot{\mathbf{y}}_2 &= \mathbf{I}^{-1}(\boldsymbol{\tau} - \mathbf{h} - \boldsymbol{\gamma})\end{aligned}\quad (8.127)$$

where the 4-dimensional state-vector is given by, $\mathbf{y} \equiv [\mathbf{y}_1, \mathbf{y}_2]^T$ in which the 2-dimensional vectors, \mathbf{y}_1 and \mathbf{y}_2 are $\mathbf{y}_1 \equiv [\theta_1, \theta_2]^T$ and $\mathbf{y}_2 \equiv [\dot{\theta}_1, \dot{\theta}_2]^T$. Similar to the one-link arm, the inputs for the two-link robot arm are also taken as zeros, i.e., $\tau_1 = \tau_2 = 0$, and the initial values of the state-vector is $\mathbf{y}(0) = [0 \ 0 \ 0 \ 0]^T$, along with the value of g as $g = 9.81$. Numerical integrations were performed using ODE45 function of MATLAB. The programs are shown in Fig. 8.22, whereas the simulation results are given in Fig. 8.23.

```
%For two-link manipulator
function ydot =ch8fdyn2(t,y);
m1 = 1; m2 = 1; a1 = 1; a2 = 1; g = 9.81; iner21 = m2*a1*a2;
tau1 = 0; tau2 = 0;
th1=y(1); th2 =y(2); th1d=y(3); th2d=y(4);

%Inertia matrix
sth2 = sin(th2); cth2 = cos(th2);
i22 = m2*a2*a2/3;
i21 = i22 + iner21*cth2/2; i12 = i21;
i11 = i22 + m1*a1*a1/3 + m2*a1*a1 + iner21*cth2;
im = [i11, i12; i21, i22];

%h-vector
h1 = - (m2*a1*a2*th1d + iner21/2*th2d)*th2d*sth2;
h2 = iner21/2*sth2*th1d*th1d;
hv=[h1;h2];

%gamma-vector
cth1 = cos(th1); cth12 = cos(th1 + th2);
gam1 = m1*g*a1/2*cth1 + m2*g*(a1*cth1 + a2/2*cth12);
gam2 = m1*g*a2/2*cth12;
gv = [gam1;gam2];

% RHS
tau=[tau1;tau2];
phi=tau-hv-gv; thdd=im\phi;
ydot=[y(3);y(4);thdd(1);thdd(2)];
```

(a) Program for state-space form

```
%For two-link manipulator
tspan=[0 10]; y0=[0;0;0;0];
[t,y]=ode45('ch8fdyn2',tspan,y0);
```

(b) Program to integrate numerically

Fig. 8.22 MATLAB programs for the simulation of the two-link robot arm

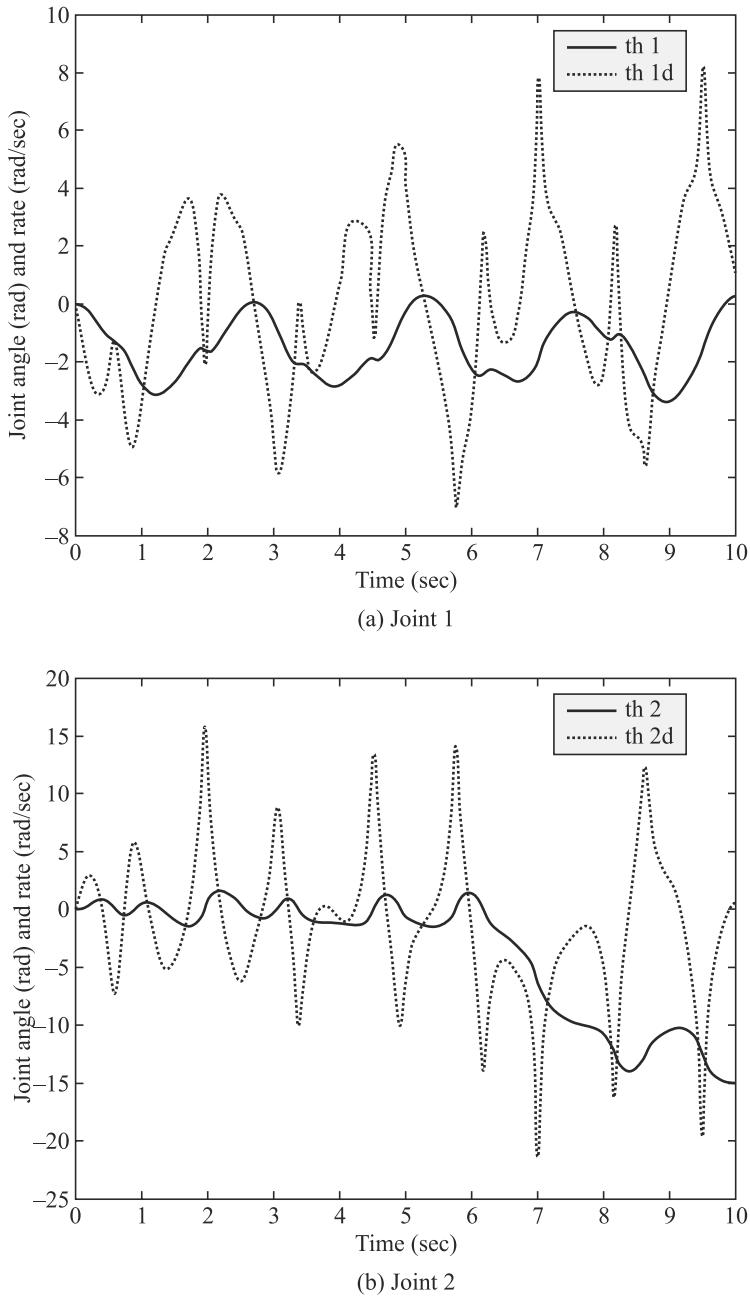
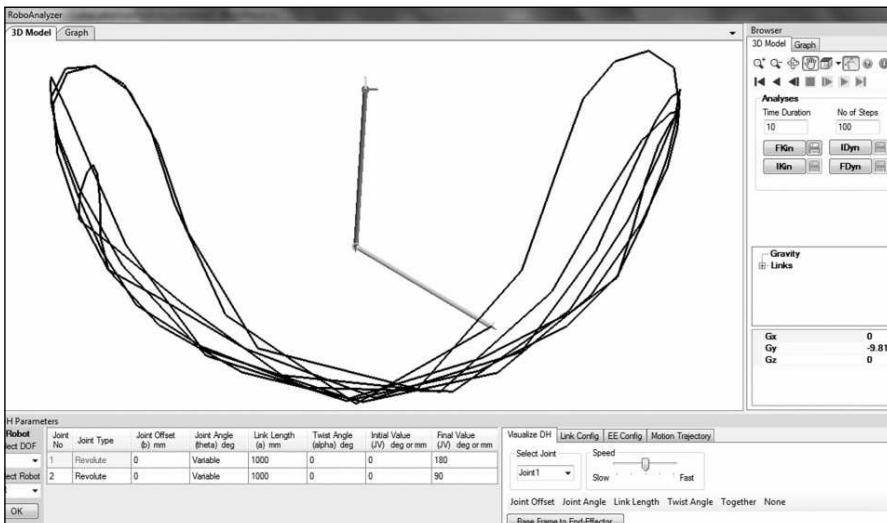


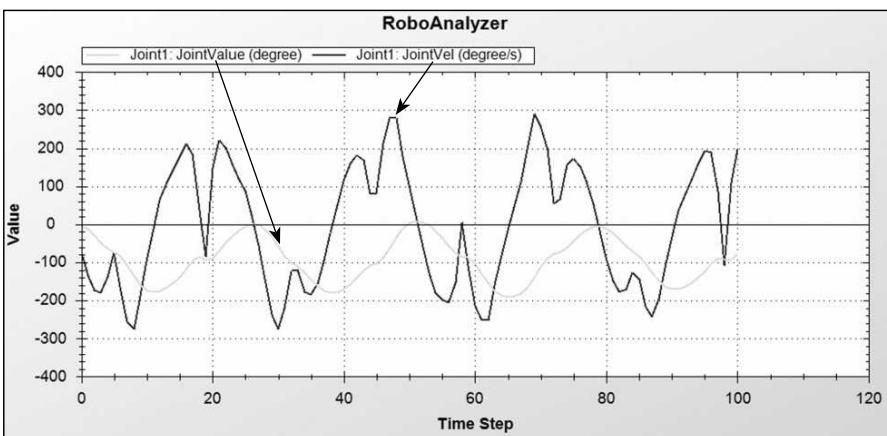
Fig. 8.23 Simulation results for the two-link robot arm under gravity

Example 8.20 Simulation of Two-link Robot Arm using RoboAnalyzer

Simulation results of the two-link robot arm given in Example 8.19 were verified using RoboAnalyzer (RA) software as well. While the animation screenshot from RA is shown in Fig. 8.24(a) which gives the idea about how the two link arm moves due to free-fall, joint angles and rates are shown in Figs. 8.24 (b–c). The plots show close match with those in Figs. 8.23 (b–c), respectively. The apparent variations in the plots are, however, due to the integrators used in two different software algorithms. Whereas MATLAB uses ODE45 function, which is based on what is known as Runge–Kutta, Dormand–Prince (4, 5) pair, RA uses Runge–Kutta 4th order method (Shampine, 1994) that is quite simple to use.

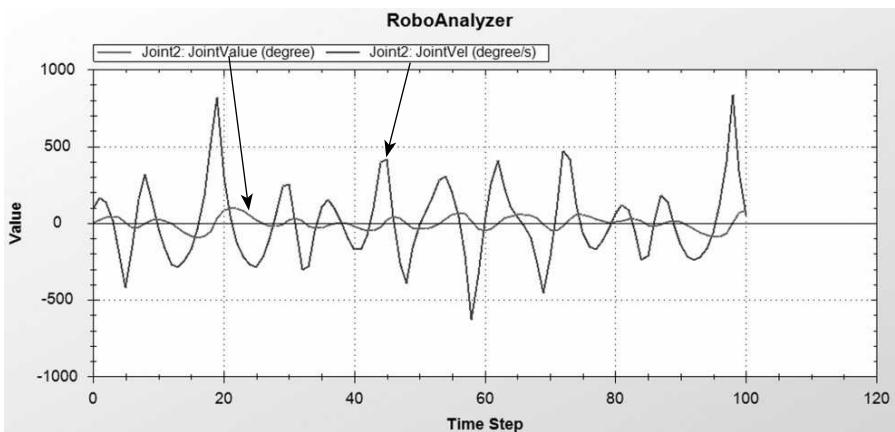


(a) Animation screenshot



(b) Variation of angle (degree) and rate of joint 1 in 10 seconds

(Contd.)



(c) Variation of angle (degree) and rate of joint 2 in 10 seconds

Fig. 8.24 RoboAnalyzer screenshots for the two-link robot arm

SUMMARY

In this chapter, rigid-body dynamics for a robot manipulator is presented using both Euler–Lagrange and Newton–Euler formulations. An example of the one-link one degree-of-freedom (DOF) arm, and two-link two-DOF robot arm are provided to illustrate both the above formulations. Inverse and forward dynamics algorithms necessary for robot control and simulation, respectively, are also included using MATLAB, MuPAD, and RoboAnalyzer software.

EXERCISES

- 8.1** Prove the matrix $[(\mathbf{p}^T \mathbf{p})\mathbf{1} - \mathbf{p}\mathbf{p}^T]$ associated to the definition of inertia matrix of a rigid body, as in Eq. (8.5) is symmetric.
- 8.2** How are parallel- and perpendicular-axis theorems useful for the determination of mass moment of inertia of complex shapes?
- 8.3** Using the recursive NE algorithm, find the joint torques of the three-link planar manipulator shown in Fig. 5.29.
- 8.4** Derive the Euler–Lagrange (EL) equations of motion for a Revolute–Prismatic (RP) jointed manipulators shown in Fig. 5.30.
- 8.5** Repeat Exercise 8.4 using the recursive Newton–Euler (NE) algorithm.
- 8.6** Repeat Exercise 8.4 for the Prismatic–Revolute (PR) manipulator shown in Fig. 5.31.
- 8.7** What are the apparent advantages and disadvantages of the Euler–Lagrange and Newton–Euler formulations?
- 8.8** For the one-link arm shown in Fig. 8.7, assume that the link is massless whereas the end-effector has mass m . Derive its equation of motion. How does it compare with the one given in Eq. (8.55)?
- 8.9** Define the inverse and forward dynamics problems of a robot manipulator.
- 8.10** What is a state-space form? What is its origin?

WEB-BASED EXERCISES

Based on Web search find the answers to the following questions:

- 8.11** Find at least three commercial software that are capable of performing dynamics of a robot manipulator.
- 8.12** What are other possible dynamic formulations (methodologies) for robot dynamics?
- 8.13** What are the commercial software for a specific architecture of a robot?

MATLAB BASED EXERCISES

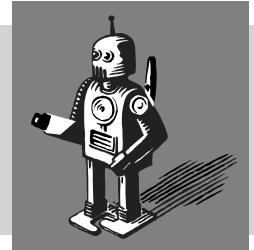
- 8.14** Find the joint torques for the manipulator in Exercise 8.3.
- 8.15** Find simulation results for the three-link manipulator while no torque is applied but the gravity is acting. The initial conditions for the generalized coordinates are
 $\theta_1(0) = \theta_2(0) = \theta_3(0) = 0 \text{ rad}$; $\dot{\theta}_1(0) = \dot{\theta}_2(0) = \dot{\theta}_3(0) = 0 \text{ rad/sec}$
 Take the geometric and inertial parameters as
 $a_1 = a_2 = 1 \text{ m}$; $a_3 = 0.5 \text{ m}$; $m_1 = m_2 = 1 \text{ kg}$; $m_3 = 0.5 \text{ kg}$
- 8.16** Write a program to generate the joint torque and force for the RP manipulator based on the equations of motion derived in Exercise 8.4 while the trajectories are defined using Eq. (8.122). Take $\theta(0) = 0$; $\theta(T) = \pi/2$; $b(0) = 0$; $b(T) = 0.1m$ — θ and b are respectively the joint variables for the revolute and prismatic joints. Consider $T = 10 \text{ sec}$.
- 8.17** Repeat Exercise 8.16 for the PR robot manipulators of Exercise 8.6.

ROBOANALYZER BASED EXERCISES

- 8.18** Validate the results of Exercise 8.14. Visualize the animation.
- 8.19** Generate inverse and forward dynamics results of two-link RP and PR arms whose dynamic equations were derived in Exercises 8.5 and 8.6, respectively. Take numerical data from Exercises 8.16.
- 8.20** Perform inverse and forward dynamics of the KUKA KR-5 robot available in RoboAnalyzer environment.

9

Recursive Robot Dynamics*



This chapter introduces one of the orthogonal-complement-based methodologies for the automatic generation of dynamic equations of motion and associated algorithms, namely, the inverse and forward dynamics. As mentioned in Chapter 8, inverse dynamics is essential for control of robot manipulators, whereas forward dynamics is required for computer simulation and real-time feedback control. To solve the inverse or forward dynamics problems of a complex robotic system, a set of dynamic equations of motion or the dynamic model of the robot under study is needed, which were derived in Chapter 8 using Euler–Lagrange (EL) and Newton–Euler (NE) formulations. The resultant equations of motion are expressed generally in the form of Ordinary Differential Equations (ODE). The same set can be obtained using alternate approaches, e.g., orthogonal complements of the velocity constraints (Huston and Passerello, 1974), Kane’s equations (Kane and Levinson, 1983), and others. Amongst these, the former approach has been adopted by many researchers for the automatic generation of the equations of motion for complex mechanical systems like the robots studied in this book. One such complement is the *Natural Orthogonal Complement* (NOC), which was originally proposed by Angeles and Lee (1988) for serial robots, but generalized later by Saha and Angeles (1991) to take into account the nonholonomic constraints of wheeled mobile robots. The NOC was eventually decoupled by the author of this book (Saha 1997, 1999, 2003) which was named as the Decoupled NOC (DeNOC). The decoupling of the NOC or the use of DeNOC has the following advantages:

- It allows one to obtain recursive order (n)— n being the number of links in the serial-chain robot—inverse and forward dynamics algorithms. The recursive forward dynamics algorithm was not possible with the original form of the NOC. Also, the recursive NE algorithm presented in Section 8.4 is meant only for the inverse dynamics.
- Each scalar element of the matrices and vectors associated with the equations of motion of the robot can be written analytically, which allows one to provide

Why Recursive

Recursive algorithms, where the computations are done based on some previous calculations, provide computationally efficient algorithms.

* This chapter requires in-depth understanding of Chapter 8. Hence, it is advised to offer to final year undergraduate students, postgraduate students at MTech, MS, and PhD levels.

many physical interpretation, e.g., articulated body inertia, etc., and helps a programmer debug the computer algorithms.

- Since the methodology is built upon basic mechanics and linear algebra theories, the concept can be easily understood even by undergraduate students.

The DeNOC-based modeling has been successfully applied to (i) serial manipulators with fixed-base, as used in industrial robots (Saha, 1997; 1999; 2003); (ii) serial robots with free-base, i.e., a configuration for free-floating space robots (Saha, 1996); (iii) closed-loop parallel Stewart platform-type robots (Saha and Schiehlen, 2001; Khan et al., 2005), and Hexapod machine tools (Koteswara Rao et al., 2006); (iv) multi-closed-loop general mechanical systems (Chaudhary and Saha, 2006); (v) serial flexible-link systems (Mohan and Saha, 2007a), and (vi) general tree-type systems (Shah and Saha, 2013). As emphasized in Mohan and Saha (2007b), and Shah et al. (2013), the DeNOC based formulation provides efficient and numerically stable algorithms.

9.1 DYNAMIC MODELING

In this section, dynamic equations of motion of an n -link n degree-of-freedom (DOF) serial robot, as shown in Fig. 9.1(a), are derived using the Decoupled Natural Orthogonal Compliment (DeNOC) matrices. First, the uncoupled Newton–Euler (NE) equations of motion, as introduced in Section 8.3, are written for all n links in a compact form. Next, the constraints between the links due to the joints, e.g., revolute, prismatic, etc., are expressed mathematically, which brings the DeNOC matrix into picture. The DeNOC relates the Cartesian velocities of all the links with the joint rates or velocities. Finally, the pre-multiplication of the DeNOC matrices with the uncoupled NE equations yields a set of independent equations of motion, which are nothing but the Euler–Lagrange (EL) equations of motion presented in Section 8.2. Hence, the EL equations are derived via DeNOC matrices without resorting to the complex partial derivatives given by Eq. (8.24).

Modeling

Modeling here means a way to be able to understand the behavior of a robot even without having a real one.

9.1.1 Uncoupled Newton–Euler Equations

For the n -link n -DOF open-loop serial-chain robot manipulator, Fig. 9.1(a), if m_i is the mass of the i^{th} link and \mathbf{I}_i denotes the 3×3 inertia tensor of the i^{th} link about its mass center C_i , as indicated in Fig. 9.1(b), the Newton–Euler (NE) equations of motion for the i^{th} link can be derived from its free-body diagram, and written as

$$\text{Euler's Equation: } \mathbf{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i \boldsymbol{\omega}_i = \mathbf{n}_i \quad (9.1a)$$

$$\text{Newton's Equation: } m_i \ddot{\mathbf{c}}_i = \mathbf{f}_i \quad (9.1b)$$

where $\boldsymbol{\omega}_i$ and $\dot{\boldsymbol{\omega}}_i$ are the 3-dimensional vectors of angular velocity and acceleration for the i^{th} link, respectively, whereas $\ddot{\mathbf{c}}_i$ is the 3-dimensional vector of acceleration of the mass center C_i . Moreover, \mathbf{n}_i and \mathbf{f}_i are the 3-dimensional vectors of the resultant moment about C_i and resultant force at C_i , respectively. Note here that no reference to the coordinate frame is made to express the vectors and matrices, as they can be

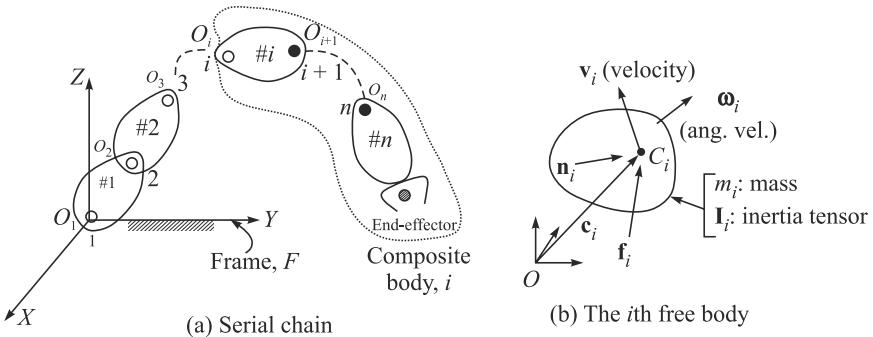


Fig. 9.1 A serial robot manipulator

represented in any frame of the analyst's choice. Typically, they are expressed in the frame attached to the i^{th} link, i.e., frame $i+1$. However, during the derivations of the equations of motion in the subsequent sections, they will be avoided. Combining Eqs. (9.1a-b), the six scalar uncoupled NE equations are written in a compact form as

$$\mathbf{M}_i \dot{\mathbf{t}}_i + \mathbf{W}_i \mathbf{M}_i \mathbf{t}_i = \mathbf{w}_i \quad (9.2a)$$

where the 6×6 mass matrix \mathbf{M}_i , and the the 6×6 angular velocity matrix \mathbf{W}_i , for the i^{th} link are given by

$$\mathbf{M}_i \equiv \begin{bmatrix} \mathbf{I}_i & \mathbf{O} \\ \mathbf{O} & m_i \end{bmatrix}; \text{ and } \mathbf{W}_i \equiv \begin{bmatrix} \boldsymbol{\omega}_i \times \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \quad (9.2b)$$

in which $\boldsymbol{\omega}_i \times \mathbf{1}$ is the 3×3 cross-product tensor associated with the angular velocity vector $\boldsymbol{\omega}_i$, which is defined as $(\boldsymbol{\omega}_i \times \mathbf{x}) \equiv \boldsymbol{\omega}_i \times \mathbf{x}$, for any three dimensional Cartesian vector \mathbf{x} . Moreover, $\mathbf{1}$ and \mathbf{O} are the 3×3 identity and zero matrices, respectively. Furthermore, the 6-dimensional vectors, twist \mathbf{t}_i and wrench \mathbf{w}_i , are as follows:

$$\mathbf{t}_i \equiv \begin{bmatrix} \boldsymbol{\omega}_i \\ \dot{\mathbf{c}}_i \end{bmatrix} \text{ and } \mathbf{w}_i \equiv \begin{bmatrix} \mathbf{n}_i \\ \mathbf{f}_i \end{bmatrix} \quad (9.2c)$$

where, in contrast to the definition of twist given in Eq. (6.93), the linear velocity of the mass center of the i^{th} link C_i is considered. In Eq. (9.2a), vector $\dot{\mathbf{t}}_i$ is the time derivative of the twist vector \mathbf{t}_i defined in Eq. (9.2c). Equation (9.2a) is now written for all n links, i.e., for $i = 1, \dots, n$, as

$$\mathbf{M}\dot{\mathbf{t}} + \mathbf{W}\mathbf{M}\mathbf{t} = \mathbf{w} \quad (9.3a)$$

where \mathbf{M} and \mathbf{W} are the $6n \times 6n$ generalized mass matrix, and the generalized matrix of the angular velocities, respectively. They are given by

$$\mathbf{M} \equiv \text{diag.}[\mathbf{M}_1, \dots, \mathbf{M}_n], \text{ and } \mathbf{W} \equiv \text{diag.}[\mathbf{W}_1, \dots, \mathbf{W}_n] \quad (9.3b)$$

Also, the $6n$ -dimensional vectors of generalized twist and wrench are defined as

$$\mathbf{t} \equiv [\mathbf{t}_1^T, \dots, \mathbf{t}_n^T]^T; \text{ and } \mathbf{w} \equiv [\mathbf{w}_1^T, \dots, \mathbf{w}_n^T]^T \quad (9.3c)$$

Equations (9.3a-c) represent the $6n$ uncoupled NE equations of motion of the n -links in the serial robot manipulator under study.

9.1.2 Kinematic Constraints

Links of the robot manipulator, Fig. 9.1(a), are coupled by kinematic pairs or joints which are either revolute or prismatic. From the rigid-body motion of the two bodies or links, namely, $\#i$ and $\#j$, as shown in Fig. 9.2, the angular and linear velocities of the i^{th} link, i.e., the twist of the i^{th} body defined in Eq. (9.2c), can be derived from the velocities of the j^{th} link or $\#j$, and the joint motion of the i^{th} joint. They are derived as follows:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_j + \mathbf{e}_i \dot{\theta}_i \quad (9.4a)$$

$$\dot{\mathbf{c}}_i = \dot{\mathbf{c}}_j + \boldsymbol{\omega}_j \times \mathbf{r}_j + \boldsymbol{\omega}_i \times \mathbf{d}_i \quad (9.4b)$$

The above six scalar velocity constraint equations can be written in compact form as

$$\mathbf{t}_i = \mathbf{B}_{ij} \mathbf{t}_j + \mathbf{p}_i \dot{\theta}_i \quad (9.4c)$$

where θ_i is the joint displacement, angular for a revolute joint and linear for a prismatic joint. Accordingly $\dot{\theta}_i$ is the joint rate. Moreover, the 6×6 matrix \mathbf{B}_{ij} and the 6-dimensional vector \mathbf{p}_i are functions of the positions of the mass centres of the two successive bodies, i.e., C_i and C_j of Fig. 9.2, and the axis of the joint joining them, namely, \mathbf{e}_i . They are defined as

$$\mathbf{B}_{ij} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{c}_{ij} \times \mathbf{1} & \mathbf{1} \end{bmatrix} \text{ and } \mathbf{p}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_i \times \mathbf{d}_i \end{bmatrix} \quad (9.4d)$$

$\mathbf{c}_{ij} \times \mathbf{1}$ being the cross-product tensor associated with vector \mathbf{c}_{ij} shown in Fig. 9.2, which is defined similar to $\boldsymbol{\omega}_i \times \mathbf{1}$ of Eq. (9.2b), i.e., $(\mathbf{c}_{ij} \times \mathbf{1})\mathbf{x} = \mathbf{c}_{ij} \times \mathbf{x}$, for any arbitrary 3-dimensional Cartesian vector \mathbf{x} . The vector \mathbf{c}_{ij} is given by $\mathbf{c}_{ij} = -\mathbf{d}_i - \mathbf{r}_j$. It is interesting to note here that the matrix \mathbf{B}_{ij} and the vector \mathbf{p}_i have the following interpretations:

- For two rigidly connected moving links, $\#i$ and $\#j$, \mathbf{B}_{ij} propagates the twist of $\#j$ to $\#i$. Hence, matrix \mathbf{B}_{ij} is termed here as the *twist propagation matrix*, which has the following properties:

$$\mathbf{B}_{ij} \mathbf{B}_{jk} = \mathbf{B}_{ik}; \mathbf{B}_{ii} = \mathbf{1}; \text{ and } \mathbf{B}_{ij}^{-1} = \mathbf{B}_{ji} \quad (9.5a)$$

- Vector \mathbf{p}_i on the other hand takes into account the motion of the i^{th} joint. Hence, \mathbf{p}_i is termed as the *joint-motion propagation vector*, which is dependent on the type of joint. For example, the expression of \mathbf{p}_i in Eq. (9.4d) is for a revolute joint shown in Fig. 9.2, whereas for a prismatic joint, vector \mathbf{p}_i is given by

$$\mathbf{p}_i \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_i \end{bmatrix}: \text{For a prismatic joint} \quad (9.5b)$$

where \mathbf{e}_i is the unit vector parallel to the axis of linear motion. Correspondingly, $\dot{\theta}_i$ of Eq. (9.4c) would mean the linear joint-rate. Other joints are not treated here because any other joint, e.g., a spherical or a cylindrical, can be treated as

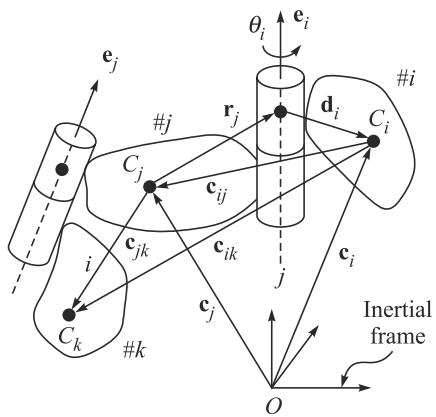


Fig. 9.2 Three coupled bodies

combination of three revolute, or revolute and prismatic pairs, respectively. For $i = 1, \dots, n$, Eq.(9.4c) is put in a compact form for all the n joints as

$$\mathbf{t} = \mathbf{N}\dot{\boldsymbol{\theta}}, \text{ where } \mathbf{N} \equiv \mathbf{N}_l \mathbf{N}_d \quad (9.6a)$$

where \mathbf{t} is the $6n$ -dimensional generalized twist defined in Eq. (9.3c). In Eq. (9.6a), it is expressed as a linear transformation of the n -dimensional joint-rate vector $\dot{\boldsymbol{\theta}}$. The $6n \times 6n$ matrix \mathbf{N}_l , the $6n \times n$ matrix \mathbf{N}_d , and the n -dimensional vector $\dot{\boldsymbol{\theta}}$, are defined as follows:

$$\mathbf{N}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{B}_{21} & \mathbf{1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \mathbf{B}_{n2} & \cdots & \mathbf{1} \end{bmatrix}; \mathbf{N}_d \equiv \begin{bmatrix} \mathbf{p}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{p}_n \end{bmatrix}; \text{ and } \dot{\boldsymbol{\theta}} \equiv \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} \quad (9.6b)$$

The $6n \times n$ matrix \mathbf{N} in Eq. (9.6a) is nothing but the Natural Orthogonal Complement (NOC) matrix of the velocity constraints (Angeles and Lee, 1988), and its decoupled form \mathbf{N}_l and \mathbf{N}_d , are referred as the Decoupled NOC (DeNOC) matrices (Saha, 1997; 1999). The expressions of the DeNOC matrices allow one to develop recursive inverse and forward dynamics algorithms required in control and simulation of robot manipulators, respectively.

Note however the difference in the expression of \mathbf{N}_l and \mathbf{N}_d in Eq. (9.6b) compared to those in Chapter 6 for the derivation of Jacobian, namely, Eq. (6.98b) and (6.101). This is mainly due to the choice of a point on the rigid link to specify its linear velocity. Whereas in Chapter 6, the point O_i at which the links #(i-1) and #i are coupled was chosen to define the linear velocity of #i, as explained after Eq. (6.93), the same is defined in this chapter with respect to C_i , i.e., the mass center of #i which is given after Eq. (9.4c). To point out these differences, two different notation for the twist-propagation matrices denoted with \mathbf{A}_{ij} and \mathbf{B}_{ij} defined in Eqs. (6.94b) and (9.4d), respectively, are introduced. Such differences will occur even for a third choice of a point to define the linear velocity of #i. In general, if \mathbf{t}_i^O and \mathbf{t}_i^C are used to define the twists with respect to point O_i (Chapter 6) and point C_i (this chapter), then one can draw the following correlations:

$$\mathbf{t}_i^C = (\mathbf{B}_{i,i-1} + \mathbf{P}_{i-1}^d)\mathbf{t}_{i-1}^O + (\mathbf{p}_i^O + \mathbf{p}_i^d)\dot{\theta}_i \quad (9.7a)$$

$$\mathbf{t}_i^O = (\mathbf{A}_{i,i-1} - \mathbf{P}_{i-1}^d)\mathbf{t}_{i-1}^C + (\mathbf{p}_i^C - \mathbf{p}_i^d)\dot{\theta}_i \quad (9.7b)$$

where \mathbf{p}_i^O and \mathbf{p}_i^C are the joint-motion propagation vectors, i.e., \mathbf{p}_i 's of Eqs. (6.94b) and (9.4d), respectively, whereas the matrix \mathbf{P}_{i-1}^d and vector \mathbf{p}_i^d are defined as

$$\mathbf{P}_{i-1}^d \equiv \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -\mathbf{d}_{i-1} \times \mathbf{1} & \mathbf{0} \end{bmatrix} \text{ and } \mathbf{p}_i^d \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_i \times \mathbf{d}_i \end{bmatrix} \quad (9.7c)$$

Even though both the definitions of twists were used in the derivations of dynamics equations of motion, e.g., Saha and Schiehlen (2001), and Chaudhary and Saha (2003) used \mathbf{t}_i^O , and Saha (1997, 1999) used \mathbf{t}_i^C , the latter has been adopted in this book mainly due to simplicity in the Euler's equations of rotational motion given by Eq. (8.92) where no explicit term associated with the linear acceleration appear.

9.1.3 Coupled Equations of Motion

The uncoupled NE equations of motion given by Eq. (9.3a) are rewritten as

$$\dot{\mathbf{M}} + \mathbf{WMt} = \mathbf{w}^E + \mathbf{w}^C \quad (9.8)$$

where \mathbf{w} of Eq. (9.3a) is substituted as $\mathbf{w} \equiv \mathbf{w}^E + \mathbf{w}^C - \mathbf{w}^E$ and \mathbf{w}^C being the external and constraint wrenches, respectively. The external wrench \mathbf{w}^E is contributed from the moments and forces due to the joint actuators, gravity, environmental effects, etc., whereas the constraint wrench \mathbf{w}^C is due to the presence of the joints that contains the reaction moments and forces at the joint interfaces. Since the constraint wrench \mathbf{w}^C does not do any useful work towards the motion of the robot links, the power consumed due to \mathbf{w}^C , i.e., $\Pi^C \equiv \mathbf{t}^T \mathbf{w}^C$, vanishes. The sole purpose of \mathbf{w}^C is to maintain the relative configuration of the links without getting separated. Using the expression for the generalized twist \mathbf{t} from Eq. (9.6a) the vanishing power due to \mathbf{w}^C , Π^C , is given by

$$\Pi^C \equiv \mathbf{t}^T \mathbf{w}^C \equiv \dot{\boldsymbol{\theta}}^T \mathbf{N}^T \mathbf{w}^C \equiv \dot{\boldsymbol{\theta}}^T \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^C = 0 \quad (9.9a)$$

For the n -link, n degree-of-freedom (DOF) serial robot, the n -dimensional joint-rate vector $\dot{\boldsymbol{\theta}}$ is independent. Hence, to satisfy Eq. (9.9a), the following condition must hold good:

$$\mathbf{N}^T \mathbf{w}^C \equiv \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^C = \mathbf{0} \quad (9.9b)$$

Now, upon multiplication of the transpose of the NOC, \mathbf{N}^T , to the uncoupled NE equations of motion, Eq. (9.8), the following set of independent dynamic equations of motion is obtained:

$$\ddot{\mathbf{I}\boldsymbol{\theta}} + \mathbf{h} = \boldsymbol{\tau}, \text{ where } \mathbf{h} \equiv \mathbf{C}\dot{\boldsymbol{\theta}} \quad (9.10a)$$

where the result of Eq. (9.9b), and the time derivative of the generalized twist \mathbf{t} from Eq. (9.6a), namely, $\dot{\mathbf{t}} = \mathbf{N}\ddot{\boldsymbol{\theta}} + \mathbf{N}\dot{\boldsymbol{\theta}}$, are used. Note that Eq. (9.10a), in comparison to the equations of motion derived in Chapter 8, namely, Eq. (8.44a), does not contain the term $\boldsymbol{\gamma}$ due to the gravity. In fact, $\boldsymbol{\tau}$ of Eq. (9.10a) contains the effect of $\boldsymbol{\gamma}$. In a way, $\boldsymbol{\tau}$ of Eq. (9.10a) is equal to $\boldsymbol{\tau}$ plus $\boldsymbol{\gamma}$ of Eq. (8.44a). Moreover,

$\mathbf{I} \equiv \mathbf{N}^T \mathbf{M} \mathbf{N} \equiv \mathbf{N}_d^T \tilde{\mathbf{M}} \mathbf{N}_d$: the $n \times n$ generalized inertia matrix (GIM), which is symmetric and positive definite;

$\mathbf{C} \equiv \mathbf{N}^T (\mathbf{M} \mathbf{N} + \mathbf{W} \mathbf{M} \mathbf{N}) \equiv \mathbf{N}_d^T (\tilde{\mathbf{M}}_l + \tilde{\mathbf{M}}_\omega + \tilde{\mathbf{M}}_e) \mathbf{N}_d$: the $n \times n$ matrix of convective inertia (MCI) terms;

$\mathbf{h} \equiv \mathbf{C}\dot{\boldsymbol{\theta}} = \mathbf{N}_d^T \tilde{\mathbf{w}}'$: the n -dimensional vector of convective inertia (VCI) terms;

$\boldsymbol{\tau} \equiv \mathbf{N}^T \mathbf{w}^E \equiv \mathbf{N}_d^T \tilde{\mathbf{w}}^E$: the n -dimensional vector of generalized forces due to driving torques/forces, and those resulting from the gravity, environment and dissipation.

Also, the $6n \times 6n$ matrices, $\tilde{\mathbf{M}}$, $\tilde{\mathbf{M}}_l$, $\tilde{\mathbf{M}}_\omega$, $\tilde{\mathbf{M}}_e$ and the $6n$ -dimensional vectors, $\tilde{\mathbf{w}}^E$ and $\tilde{\mathbf{w}}'$, are given by

$$\begin{aligned} \tilde{\mathbf{M}} &\equiv \mathbf{N}_l^T \mathbf{M} \mathbf{N}_l; \tilde{\mathbf{M}}_l \equiv \mathbf{N}_l^T \mathbf{M} \dot{\mathbf{N}}_l, \tilde{\mathbf{M}}_\omega \equiv \tilde{\mathbf{M}} \boldsymbol{\Omega}, \tilde{\mathbf{M}}_e \equiv \mathbf{N}_l^T \mathbf{W} \mathbf{M} \mathbf{N}_l \\ \tilde{\mathbf{w}}' &\equiv \mathbf{N}_l^T (\mathbf{M} \mathbf{t}' + \mathbf{W} \mathbf{M} \mathbf{t}); \mathbf{t}' \equiv (\dot{\mathbf{N}}_l + \mathbf{N}_l \boldsymbol{\Omega}) \mathbf{N}_d \dot{\boldsymbol{\theta}}; \text{ and } \tilde{\mathbf{w}}^E \equiv \mathbf{N}_l^T \mathbf{w}^E \end{aligned} \quad (9.10b)$$

where $\dot{\mathbf{N}}_d = \boldsymbol{\Omega} \mathbf{N}_d$ in which $\boldsymbol{\Omega} \equiv \text{diag.}[\boldsymbol{\Omega}_1, \dots, \boldsymbol{\Omega}_n]$ was used. Note that the 6×6 skew-symmetric matrix $\boldsymbol{\Omega}_i$ associated with the 3-dimensional angular velocity vector $\boldsymbol{\omega}_i$ is different from \mathbf{W}_i . The former is defined as $\boldsymbol{\Omega}_i \equiv \text{diag.}[\boldsymbol{\omega}_i \times \mathbf{1}, \boldsymbol{\omega}_i \times \mathbf{1}]$. Moreover, the matrices, \mathbf{N}_l , \mathbf{M} , \mathbf{W} , and the vector \mathbf{w}^E are defined in the previous sections, whereas the vector \mathbf{t}' in Eq. (9.10b) is nothing but the twist-rate vector while $\ddot{\boldsymbol{\theta}} = \mathbf{0}$. It is pointed

out here that Eq. (9.10a) was also derived in Chapter 8, i.e., Eq. (8.44a), using the Euler–Lagrange equations of motion. In the latter case, one required complex partial differentiations, whereas the former is derived here from the Newton–Euler equations of motion which are simpler to visualize in the three-dimensional Cartesian space, and simple linear-algebra concepts.

9.2 ANALYTICAL EXPRESSIONS

It can be shown that using the concept of the DeNOC matrices each element of the GIM, MCI, VCI and the generalized forces can be expressed analytically, which allow one to extract many physical interpretations and suggest ways to simplify the expressions and computational complexity. The analytical expressions are obtained next.

Computation Complexity

In computer algorithms, computational complexity can be estimated in terms of the number of arithmetic operations performed in terms of multiplications, additions, etc.

9.2.1 Generalized Inertia Matrix (GIM)

The expressions for the $n \times n$ GIM \mathbf{I} is given after Eq. (9.10a) as

$$\mathbf{I} \equiv \mathbf{N}^T \mathbf{M} \mathbf{N} \equiv \mathbf{N}_d^T \tilde{\mathbf{M}} \mathbf{N}_d \quad (9.11a)$$

where $\tilde{\mathbf{M}} \equiv \mathbf{N}_l^T \mathbf{M} \mathbf{N}_l$ is the $6n \times 6n$ symmetric *composite mass matrix*, which is obtained as

$$\tilde{\mathbf{M}} \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1 & & & & & \text{sym} \\ \tilde{\mathbf{M}}_2 \mathbf{B}_{21} & \tilde{\mathbf{M}}_2 & & & & \\ \tilde{\mathbf{M}}_3 \mathbf{B}_{31} & \tilde{\mathbf{M}}_3 \mathbf{B}_{32} & \tilde{\mathbf{M}}_3 & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \tilde{\mathbf{M}}_n \mathbf{B}_{n1} & \tilde{\mathbf{M}}_n \mathbf{B}_{n2} & \tilde{\mathbf{M}}_n \mathbf{B}_{n3} & \cdots & \tilde{\mathbf{M}}_n & \end{bmatrix} \quad (9.11b)$$

where *sym* denotes the symmetric elements of the matrix $\tilde{\mathbf{M}}$. The 6×6 matrix $\tilde{\mathbf{M}}_i$ can be evaluated from $i = n$ to 1 as

$$\tilde{\mathbf{M}}_i \equiv \mathbf{M}_i + \sum_{k=i+1}^n \mathbf{B}_{ki}^T \tilde{\mathbf{M}}_k \mathbf{B}_{ki} \quad (9.11c)$$

which requires order n^2 computations, as there is a summation over $k = i+1, \dots, n$. A close look into the equation, along with the first two properties of Eq. (9.5a), however, reveal that the summation expression can be evaluated recursively, for $i = n, \dots, 1$, as

$$\tilde{\mathbf{M}}_i \equiv \mathbf{M}_i + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{M}}_{i+1} \mathbf{B}_{i+1}, \text{ where } \tilde{\mathbf{M}}_n \equiv \mathbf{M}_n \quad (9.11d)$$

Equation (9.11d) has the following physical interpretations:

1. It is the mass matrix of a body composed of links, # n , ..., # i , that are rigidly connected. This is referred as *composite body* i , as indicated in Fig. 9.1(a), whose name can be justified from the 3×3 block matrices of $\tilde{\mathbf{M}}_i$. In Eq. (9.11d), if $i = n$,

$$\tilde{\mathbf{M}}_n = \mathbf{M}_n \equiv \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & m_n \mathbf{1} \end{bmatrix} \quad (9.12a)$$

and, for $i = n-1$,

$$\tilde{\mathbf{M}}_{n-1} \equiv \mathbf{M}_{n-1} + \mathbf{B}_{n,n-1}^T \tilde{\mathbf{M}}_n \mathbf{B}_{n,n-1} \quad (9.12b)$$

which can be rewritten as

$$\tilde{\mathbf{M}}_{n-1} \equiv \begin{bmatrix} \tilde{\mathbf{I}}_{n-1} & -\tilde{\boldsymbol{\delta}}_{n-1} \times \mathbf{1} \\ \tilde{\boldsymbol{\delta}}_{n-1} \times \mathbf{1} & \tilde{m}_{n-1} \mathbf{1} \end{bmatrix} \quad (9.12c)$$

where the scalar \tilde{m}_{n-1} , the 3-dimensional vector $\tilde{\boldsymbol{\delta}}_{n-1}$, and the 3×3 matrix $\tilde{\mathbf{I}}_{n-1}$ are given by

$$\tilde{m}_{n-1} \equiv m_{n-1} + \tilde{m}_n, \text{ where } \tilde{m}_n = m_n \quad (9.13a)$$

$$\tilde{\boldsymbol{\delta}}_{n-1} \equiv m_n \mathbf{c}_{n,n-1} + \tilde{\boldsymbol{\delta}}_n, \text{ where } \tilde{\boldsymbol{\delta}}_n = 0 \quad (9.13b)$$

$$\tilde{\mathbf{I}}_{n-1} \equiv \mathbf{I}_{n-1} + \tilde{\mathbf{I}}_n - \mathbf{c}_{n,n-1} \times (\tilde{\boldsymbol{\delta}}_{n-1} \times \mathbf{1}), \text{ where } \tilde{\mathbf{I}}_n = \mathbf{I}_n \quad (9.13c)$$

The matrix $\tilde{\mathbf{I}}_{n-1}$ is the inertia tensor of the body composed of rigidly connected links #(n-1) and #n with respect to the mass center of the $(i-1)^{\text{st}}$ link, i.e., C_{i-1} , in which the third term is nothing but the one associated with the transfer of the definition of \mathbf{I}_n from C_n to C_{n-1} , similar to the parallel-axis theorem given in Section 8.1.3. Continuing with $i = n-2, \dots, 1$, the scalar \tilde{m}_i , the 3-dimensional vector $\tilde{\boldsymbol{\delta}}_i$, and the 3×3 matrix $\tilde{\mathbf{I}}_i$ are calculated as follows:

$$\tilde{m}_i \equiv m_i + \tilde{m}_{i+1} \quad (9.14a)$$

$$\tilde{\boldsymbol{\delta}}_i \equiv \tilde{m}_{i+1} \mathbf{c}_{i+1,i} + \tilde{\boldsymbol{\delta}}_{i+1} \quad (9.14b)$$

$$\tilde{\mathbf{I}}_i \equiv \mathbf{I}_i + \tilde{\mathbf{I}}_{i+1} - \tilde{\boldsymbol{\delta}}_{i+1} \times (\mathbf{c}_{i+1,i} \times \mathbf{1}) - \mathbf{c}_{i+1,i} \times (\tilde{\boldsymbol{\delta}}_i \times \mathbf{1}) \quad (9.14c)$$

2. The inertia effect of the composite body $(i+1)$, i.e., $\tilde{\mathbf{I}}_{i+1}$, is taken into account with respect to C_i , and added with the inertia tensor of link i with respect to C_i , i.e., \mathbf{I}_i , to give the inertia of the composite body i with respect to C_i , i.e., $\tilde{\mathbf{I}}_i$. Other terms, i.e., (2,1) and (2,2)—blocks of Eq. (9.12c) are similarly added to define the mass matrix of the composite body i or the composite mass matrix $\tilde{\mathbf{M}}_i$. Now, using the expression, $\mathbf{I} \equiv \mathbf{N}_d^T \tilde{\mathbf{M}} \mathbf{N}_d$, the symmetric GIM \mathbf{I} is written as

$$\mathbf{I} \equiv \begin{bmatrix} \mathbf{p}_1^T \tilde{\mathbf{M}}_1 \mathbf{p}_1 & \cdots & \cdots & \cdots & \text{sym} \\ \mathbf{p}_2^T \tilde{\mathbf{M}}_2 \mathbf{B}_{21} \mathbf{p}_1 & \mathbf{p}_2^T \tilde{\mathbf{M}}_2 \mathbf{p}_2 & \cdots & \cdots & \vdots \\ \mathbf{p}_3^T \tilde{\mathbf{M}}_3 \mathbf{B}_{31} \mathbf{p}_1 & \mathbf{p}_3^T \tilde{\mathbf{M}}_3 \mathbf{B}_{32} \mathbf{p}_2 & \mathbf{p}_3^T \tilde{\mathbf{M}}_3 \mathbf{p}_3 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_n^T \tilde{\mathbf{M}}_n \mathbf{B}_{n1} \mathbf{p}_1 & \mathbf{p}_n^T \tilde{\mathbf{M}}_n \mathbf{B}_{n2} \mathbf{p}_2 & \mathbf{p}_n^T \tilde{\mathbf{M}}_n \mathbf{B}_{n3} \mathbf{p}_3 & \cdots & \mathbf{p}_n^T \tilde{\mathbf{M}}_n \mathbf{p}_n \end{bmatrix} \quad (9.15a)$$

where “sym” denotes the symmetric elements, and the (i,j) scalar element of the GIM, denoted by i_{ij} , is given analytically as

$$i_{ij} \equiv \mathbf{p}_i^T \tilde{\mathbf{M}}_i \mathbf{B}_{ij} \mathbf{p}_j, \text{ for } i = n, \dots, 1; j = i, \dots, 1 \quad (9.15b)$$

Note that the symmetric elements above the diagonal elements of the GIM, Eq. (9.15a), can be expressed as

$$i_{ji} \equiv \mathbf{p}_j^T \mathbf{B}_{ij}^T \tilde{\mathbf{M}}_l \mathbf{p}_i, \text{ for } i = n, \dots, 1; j = i, \dots, 1 \quad (9.15c)$$

In Eq. (9.15a), $i_{ij} = i_{ji}$, as obvious from the transpose of the right hand sides of Eqs. (9.15b) and (9.15c). Hence, one is required only to compute either (9.15b) or (9.15c) in a dynamics algorithm. In the examples of this chapter, the lower triangular elements of the GIM are computed using Eq. (9.15b), which require n^2 arithmetic operations— n being the number of links or joints in the serial-chain robot.

9.2.2 Matrix for Convective Inertia (MCI) Terms

Analytical expressions of the matrix of convective inertia (MCI) terms are derived from the MCI definition given after Eq. (9.10a), i.e.,

$$\mathbf{C} \equiv \mathbf{N}^T (\dot{\mathbf{M}}\mathbf{N} + \mathbf{W}\mathbf{M}\mathbf{N}) \equiv \mathbf{N}_d^T (\tilde{\mathbf{M}}_l + \tilde{\mathbf{M}}_\omega + \tilde{\mathbf{M}}_e) \mathbf{N}_d \quad (9.16a)$$

where the $6n \times 6n$ matrices $\tilde{\mathbf{M}}_l$, $\tilde{\mathbf{M}}_\omega$ and $\tilde{\mathbf{M}}_e$ are reproduced from Eq. (9.10b) as

$$\tilde{\mathbf{M}}_l \equiv \mathbf{N}_l^T \mathbf{M} \dot{\mathbf{N}}_l; \quad \tilde{\mathbf{M}}_\omega \equiv \tilde{\mathbf{M}} \boldsymbol{\Omega} \equiv \mathbf{N}_l^T \mathbf{M} \mathbf{N}_l \boldsymbol{\Omega}; \quad \text{and} \quad \tilde{\mathbf{M}}_e \equiv \mathbf{N}_l^T \mathbf{W} \mathbf{M} \mathbf{N}_l \quad (9.16b)$$

Matrix $\tilde{\mathbf{M}}_l$ is then obtained as

$$\tilde{\mathbf{M}}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n1}^T \\ \mathbf{0} & \mathbf{1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{B}_{n,n-1}^T \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{M}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{M}_n \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \dot{\mathbf{B}}_{21} & \mathbf{0} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \dot{\mathbf{B}}_{n1} & \cdots & \dot{\mathbf{B}}_{n,n-1} & \mathbf{0} \end{bmatrix} \quad (9.16c)$$

where the 6×6 matrix $\dot{\mathbf{B}}_{i+1,i}$ is given by

$$\dot{\mathbf{B}}_{i+1,i} \equiv \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -(\dot{\mathbf{r}}_i + \dot{\mathbf{d}}_{i+1}) \times \mathbf{1} & \mathbf{0} \end{bmatrix} \quad (9.16d)$$

in which $\dot{\mathbf{r}}_i = \boldsymbol{\omega}_i \times \mathbf{r}_i$, and $\dot{\mathbf{d}}_{i+1} = \boldsymbol{\omega}_{i+1} \times \mathbf{d}_{i+1}$. The expression of $\tilde{\mathbf{M}}_l$ is then rewritten as

$$\tilde{\mathbf{M}}_l \equiv \begin{bmatrix} \mathbf{B}_{21}^T \tilde{\mathbf{H}}_{21} & \mathbf{B}_{31}^T \tilde{\mathbf{H}}_{32} & \cdots & \mathbf{B}_{n1}^T \tilde{\mathbf{H}}_{n,n-1} & \mathbf{0} \\ \tilde{\mathbf{H}}_{21} & \mathbf{B}_{32}^T \tilde{\mathbf{H}}_{32} & \cdots & \vdots & \vdots \\ \tilde{\mathbf{H}}_{31} & \tilde{\mathbf{H}}_{32} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \mathbf{B}_{n,n-1}^T \tilde{\mathbf{H}}_{n,n-1} & \vdots \\ \tilde{\mathbf{H}}_{n1} & \tilde{\mathbf{H}}_{n2} & \cdots & \tilde{\mathbf{H}}_{n,n-1} & \mathbf{0} \end{bmatrix} \quad (9.16e)$$

In Eq. (9.16e), $\tilde{\mathbf{H}}_{ij} \equiv \tilde{\mathbf{M}}_l \dot{\mathbf{B}}_{ij} + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{H}}_{i+1,i}$, and for, $i = n$, $\tilde{\mathbf{H}}_{n+1,n} = \mathbf{0}$. Next, the $6n \times 6n$ matrix $\tilde{\mathbf{M}}_\omega$, as defined in Eq. (9.16b), is formed as

$$\tilde{\mathbf{M}}_\omega \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1 & \cdots & \cdots & \text{sym} \\ \tilde{\mathbf{M}}_2 \mathbf{B}_{21} & \tilde{\mathbf{M}}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{M}}_n \mathbf{B}_{n1} & \tilde{\mathbf{M}}_n \mathbf{B}_{n2} & \cdots & \tilde{\mathbf{M}}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \boldsymbol{\Omega}_n \end{bmatrix} \quad (9.16f)$$

which yields

$$\tilde{\mathbf{M}}_{\omega} \equiv \begin{bmatrix} \tilde{\mathbf{M}}_1 \boldsymbol{\Omega}_1 & \mathbf{B}_{21}^T \tilde{\mathbf{M}}_2 \boldsymbol{\Omega}_2 & \cdots & \mathbf{B}_{n1}^T \tilde{\mathbf{M}}_n \boldsymbol{\Omega}_n \\ \tilde{\mathbf{M}}_2 \mathbf{B}_{21} \boldsymbol{\Omega}_1 & \tilde{\mathbf{M}}_2 \boldsymbol{\Omega}_2 & \cdots & \mathbf{B}_{n2}^T \tilde{\mathbf{M}}_n \boldsymbol{\Omega}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{M}}_n \mathbf{B}_{n1} \boldsymbol{\Omega}_1 & \tilde{\mathbf{M}}_n \mathbf{B}_{n2} \boldsymbol{\Omega}_2 & \cdots & \tilde{\mathbf{M}}_n \boldsymbol{\Omega}_n \end{bmatrix} \quad (9.16g)$$

Finally, the matrix $\tilde{\mathbf{M}}_e$ is obtained as

$$\tilde{\mathbf{M}}_e \equiv \begin{bmatrix} 1 & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n1}^T \\ \mathbf{0} & 1 & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{B}_{n,n-1}^T \\ \mathbf{0} & \mathbf{0} & \cdots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \mathbf{M}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 \mathbf{M}_2 & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{W}_n \mathbf{M}_n \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{B}_{21} & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \mathbf{B}_{n2} & \cdots & 1 \end{bmatrix} \quad (9.16h)$$

which is written in compact form as

$$\tilde{\mathbf{M}}_e \equiv \begin{bmatrix} \tilde{\mathbf{M}}'_1 & \cdots & \cdots & sym \\ \tilde{\mathbf{M}}'_2 \mathbf{B}_{21} & \tilde{\mathbf{M}}'_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{M}}'_n \mathbf{B}_{n1} & \tilde{\mathbf{M}}'_n \mathbf{B}_{n2} & \cdots & \tilde{\mathbf{M}}'_n \end{bmatrix} \quad (9.16i)$$

where $\tilde{\mathbf{M}}'_i$ for $i = n, \dots, 1$, is calculated similar to matrix $\tilde{\mathbf{M}}_i$, Eq. (9.11d), as

$$\tilde{\mathbf{M}}'_i = \mathbf{M}'_i + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{M}}'_{i+1} \mathbf{B}_{i+1,i} \quad (9.16j)$$

in which $\mathbf{M}'_i = \mathbf{W}_i \mathbf{M}_i$, for $i = n, \dots, 1$. The scalar elements of the MCI \mathbf{C} , i.e., c_{ij} , for $i, j = n, \dots, 1$, are then obtained explicitly from Eqs. (9.16a-j) as

$$c_{i,j} \equiv \mathbf{p}_i^T [\mathbf{B}_{j+1,i}^T \tilde{\mathbf{H}}_{j+1,i} + \mathbf{B}_{ji}^T (\tilde{\mathbf{M}}_j \boldsymbol{\Omega}_j + \tilde{\mathbf{M}}'_j)] \mathbf{p}_j \quad \text{if } i \leq j \quad (9.16k)$$

$$c_{i,j} \equiv \mathbf{p}_i^T (\tilde{\mathbf{H}}_{ij} + \tilde{\mathbf{M}}_i \mathbf{B}_{ij} \boldsymbol{\Omega}_j + \tilde{\mathbf{M}}'_i \mathbf{B}_{ij}) \mathbf{p}_j \quad \text{otherwise.}$$

Comparing Eqs. (9.15b) and (9.16k), it is observed that the elements of the GIM and MCI are expressed in a uniform manner, i.e., $\mathbf{p}_i^T(\cdot)\mathbf{p}_j$, where (\cdot) denotes the matrix argument. In Eq. (9.16k), the explicit analytical expression for each element of the MCI is available, which is not advisable to be used for the calculation of vector $\mathbf{h} (\equiv \mathbf{C}\dot{\theta})$ given by Eq. (9.10a) but suitable for the physical interpretations and debugging. Whereas the explicit evaluation of the MCI \mathbf{C} requires order (n^2) — n being the degree of freedom of the robot—calculations, the VCI \mathbf{h} can be computed recursively needing order n operations. The order n algorithm for the VCI is given next. Now, as per as the physical interpretation of the MCI is concerned, note that for a planar robot with all revolute joints, the axes are all parallel and do not change their direction while the robot is in motion. Hence, their time derivative, $\dot{\mathbf{N}}_d = \boldsymbol{\Omega} \mathbf{N}_d = \mathbf{0}$. As a result, the term in Eq.(9.16a) associated with $\tilde{\mathbf{M}}_{\omega}$ vanishes to provide simpler expressions of the MCI \mathbf{C} .

9.2.3 Vector of Convective Inertia (VCI) Terms

The vector of convective inertia (VCI) terms \mathbf{h} is given after Eq. (9.10a) is reproduced below:

$$\mathbf{h} \equiv \mathbf{C}\dot{\theta} = \mathbf{N}_d^T \tilde{\mathbf{w}}', \text{ where } \tilde{\mathbf{w}}' = \mathbf{N}_l^T (\mathbf{M}\mathbf{t}' + \mathbf{W}\mathbf{M}\mathbf{t}) \text{ and } \mathbf{t}' = (\dot{\mathbf{N}}_l + \mathbf{N}_l \boldsymbol{\Omega})\mathbf{N}_d \dot{\theta} \quad (9.17)$$

Note that \mathbf{t}' is the generalized twist-rate vector while $\ddot{\theta} = \mathbf{0}$, i.e., it contains the centrifugal and Coriolis acceleration terms. Introducing the following notations

$$\mathbf{M}' = \mathbf{W}\mathbf{M} \text{ and } \mathbf{w}' = \mathbf{M}\mathbf{t}' + \mathbf{M}'\mathbf{t}$$

and substituting the expression for \mathbf{N}_l from Eq. (9.6b) into Eq. (9.17) yields

$$\tilde{\mathbf{w}}' \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n1}^T \\ \mathbf{0} & \mathbf{1} & & \vdots \\ \vdots & \ddots & \mathbf{B}_{n,n-1}^T & \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w}'_1 \\ \mathbf{w}'_2 \\ \vdots \\ \mathbf{w}'_n \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{w}}'_1 \\ \tilde{\mathbf{w}}'_2 \\ \vdots \\ \tilde{\mathbf{w}}'_n \end{bmatrix} \quad (9.18a)$$

where $\mathbf{w}'_i = \mathbf{M}_i \mathbf{t}'_i + \mathbf{M}'_i \mathbf{t}_i$, for $i = n, \dots, 1$. Note that the $6n$ -dimensional vector \mathbf{w}' can be interpreted as the generalized wrench due to the convective inertia terms. Moreover, the elements of the vector $\tilde{\mathbf{w}}'_i$, Eq. (9.18a), can be obtained recursively as

$$\tilde{\mathbf{w}}'_i = \mathbf{w}'_i + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{w}}'_{i+1}, \text{ where } \tilde{\mathbf{w}}'_n = \mathbf{w}'_n \quad (9.18b)$$

Furthermore, using the expression for \mathbf{N}_d , the VCI of Eq. (9.17) \mathbf{h} is given by

$$\mathbf{h} \equiv \begin{bmatrix} \mathbf{p}_1^T \tilde{\mathbf{w}}'_1 \\ \mathbf{p}_2^T \tilde{\mathbf{w}}'_2 \\ \vdots \\ \mathbf{p}_n^T \tilde{\mathbf{w}}'_n \end{bmatrix} \quad (9.19a)$$

in which each element of \mathbf{h}_i , denoted as h_i , is written as

$$h_i = \mathbf{p}_i^T \tilde{\mathbf{w}}'_i, \text{ for } i = n, \dots, 1 \quad (9.19b)$$

The expressions in Eqs. (9.19a-b) together provide an order n algorithm for the calculation of the VCI.

9.2.4 Generalized Force

The expressions for the elements of the generalized forces τ , given after Eq. (9.10a) are as follows:

$$\tau = \mathbf{N}_d^T \tilde{\mathbf{w}}^E, \text{ where } \tilde{\mathbf{w}}^E = \mathbf{N}_l^T \mathbf{w}^E \quad (9.20a)$$

Upon substitution of the expression for \mathbf{N}_l from Eq. (9.6b) and noting that, $\mathbf{w}^E \equiv [(\mathbf{w}_1^E)^T \dots (\mathbf{w}_n^E)^T]^T$, the $6n$ -dimensional vector $\tilde{\mathbf{w}}^E$ is written as

$$\tilde{\mathbf{w}}^E \equiv \begin{bmatrix} \mathbf{1} & \mathbf{B}_{21}^T & \cdots & \mathbf{B}_{n,1}^T \\ \mathbf{0} & \mathbf{1} & & \vdots \\ \vdots & \ddots & \mathbf{B}_{n,n-1}^T & \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w}^E_1 \\ \mathbf{w}^E_2 \\ \vdots \\ \mathbf{w}^E_n \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{w}}^E_1 \\ \tilde{\mathbf{w}}^E_2 \\ \vdots \\ \tilde{\mathbf{w}}^E_n \end{bmatrix} \quad (9.20b)$$

where $\tilde{\mathbf{w}}_i^E \equiv \mathbf{w}_i^E + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{w}}_{i+1}^E$ and $\tilde{\mathbf{w}}_n^E \equiv \mathbf{w}_n^E$. The generalized force τ is then found as

$$\boldsymbol{\tau} \equiv \begin{bmatrix} \mathbf{p}_1^T \tilde{\mathbf{w}}_1^E \\ \mathbf{p}_2^T \tilde{\mathbf{w}}_2^E \\ \vdots \\ \mathbf{p}_n^T \tilde{\mathbf{w}}_n^E \end{bmatrix} \quad (9.21a)$$

From Eq. (9.21a), each element of the n -dimensional vector $\boldsymbol{\tau}$, i.e., τ_i is then obtained from

$$\tau_i = \mathbf{p}_i^T \tilde{\mathbf{w}}_i^E \text{ for } i = n, \dots, 1 \quad (9.21b)$$

Equations (9.10a), (9.15b), (9.16k) or (9.19b), and (9.21b) together provide the explicit expressions for the dynamic equations of motion for the n -link, n -DOF serial robot.

Example 9.1 One-link Planar Arm

For the one-link arm shown in Fig. 9.3, the only equation of motion is derived here using the concept of the DeNOC matrices. Using Eqs. (9.6a-b), the DeNOC matrices for the one-link arm are given by

$$\mathbf{N} = \mathbf{N}_l \mathbf{N}_d = \mathbf{p}, \text{ where } \mathbf{N}_l = \mathbf{1}, \text{ and } \mathbf{N}_d = \mathbf{p} \quad (9.22)$$

The inertia term, which is a scalar for the one-link arm, is then obtained from Eq. (9.15b) as

$$I (\equiv i_{11}) = \mathbf{p}^T \tilde{\mathbf{M}} \mathbf{p} = \mathbf{e}^T \mathbf{I} \mathbf{e} + m(\mathbf{e} \times \mathbf{d})^T (\mathbf{e} \times \mathbf{d});$$

$$\text{where } \mathbf{p} \equiv \begin{bmatrix} \mathbf{e} \\ \mathbf{e} \times \mathbf{d} \end{bmatrix}; \tilde{\mathbf{M}} \equiv \mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & m\mathbf{1} \end{bmatrix} \quad (9.23a)$$

In Eq. (9.23a), no subscript is used to the vector and matrix notations, as there is only one link and one joint. Now, the 3-dimensional vectors \mathbf{e} and \mathbf{d} in the fixed frame, i.e., frame 1, are obtained as

$$[\mathbf{e}]_1 \equiv [0 \ 0 \ 1]^T; [\mathbf{d}]_1 \equiv \left[\frac{1}{2}ac \ \frac{1}{2}as \ 0 \right]^T \quad (9.23b)$$

where $s \equiv \sin \theta$ and $c \equiv \cos \theta$. Moreover, the 3×3 inertia matrix \mathbf{I} for the one-link arm about its mass center C represented in the link-fixed frame, i.e., frame 2, is written using Eq. (8.22b) as

$$[\mathbf{I}]_2 = \frac{ma^2}{12} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.23c)$$

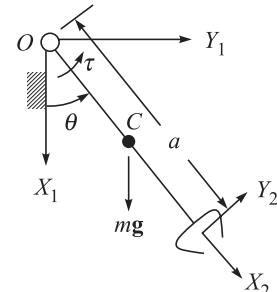


Fig. 9.3 One-link arm

Using Eq. (8.82), one can write

$$[\mathbf{I}]_1 = \mathbf{Q}[\mathbf{I}]_2 \mathbf{Q}^T = \frac{ma^2}{12} \begin{bmatrix} c^2 & sc & 0 \\ sc & s^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.23d)$$

where \mathbf{Q} is the 3×3 rotation matrix given by

$$\mathbf{Q} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.23e)$$

Substituting Eqs. (9.23d) in Eq. (9.23a), one gets

$$I = \frac{1}{3}ma^2 \quad (9.23f)$$

Next, the VCI is obtained using Eq. (9.16k), where for the one-link case, $\mathbf{B}_{11} = \mathbf{1}$, $\mathbf{B}_{21}^T \tilde{\mathbf{H}}_{21} = \mathbf{O}$, as it does not arise, and $\tilde{\mathbf{M}}' \equiv \mathbf{M}' = \mathbf{W}\mathbf{M}$. Hence,

$$C(\equiv c_{11}) = \mathbf{p}^T (\mathbf{M}\boldsymbol{\Omega} + \mathbf{W}\mathbf{M})\mathbf{p} = \dot{\theta}\mathbf{e}^T [(\mathbf{I}\mathbf{e} \times \mathbf{e}) + (\mathbf{e} \times \mathbf{I}\mathbf{e}) - m\mathbf{d} \times \mathbf{d}] = 0 \quad (9.24a)$$

where the expression of the 6×6 matrices $\boldsymbol{\Omega}$ and \mathbf{W} used above are given by

$$\boldsymbol{\Omega} = \begin{bmatrix} \dot{\theta}\mathbf{e} \times \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \dot{\theta}\mathbf{e} \times \mathbf{1} \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} \dot{\theta}\mathbf{e} \times \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (9.24b)$$

Moreover, the 6-dimensional vector \mathbf{p} and the 6×6 matrix \mathbf{M} are given in Eq. (9.23a). Furthermore, the properties of vector products were used in obtaining the result of Eq. (9.24a), i.e., $\mathbf{d} \times \mathbf{d} = \mathbf{0}$, and $\mathbf{e}^T(\mathbf{e} \times \mathbf{I}\mathbf{e}) = 0$ or $\mathbf{e}^T(\mathbf{e}\mathbf{I} \times \mathbf{e}) = 0$. The latter results are due to the fact that vector \mathbf{e} is orthogonal to the one that is obtained from the cross-product $\mathbf{e} \times \mathbf{I}\mathbf{e}$ or $\mathbf{I}\mathbf{e} \times \mathbf{e}$. Finally, the generalized force is obtained from Eq. (9.21b) as

$$\tau_1 = \mathbf{N}^T \mathbf{w}^E = [\mathbf{e}^T \quad (\mathbf{e} \times \mathbf{d})^T] \begin{bmatrix} \mathbf{n} \\ \mathbf{f} \end{bmatrix} = \tau - \frac{1}{2}mgas \quad (9.24c)$$

where the external moments due to the actuator and external force due to gravity represented in frame 1 are given as

$$[\mathbf{n}]_1 \equiv [0 \ 0 \ \tau]^T, [\mathbf{f}]_1 \equiv [mg \ 0 \ 0]^T \quad (9.24d)$$

It is pointed out here that for the simple systems like the one in this example, it may be easier to obtain the dynamic equation of motion given by Eqs. (9.23f) and (9.24c) from the direct application of the expressions in terms of the NOC matrix that appear after Eq. (9.10a).

Example 9.2 One-link Arm using MuPAD

Figure 9.4 shows how to develop the equation of motion of the one-link arm of Example 9.1 in the MuPAD environment.

Equation of motion of one-link system using DeNOC matrices

```

[ th:='&theta; `:thd:=Symbol::accentDot(`&theta; `):
[ thdd:=Symbol::accentDoubleDot(`&theta; `):tau:='&tau; `:
[ e:=matrix([0,0,1]):eT:=matrix([[0,0,1]]):
[ d:=matrix([a/2*cos(th),a/2*sin(th),0]):
[ exd1:=e[2]*d[3]-e[3]*d[2]:
[ exd2:=e[3]*d[1]-e[1]*d[3]:
[ exd3:=e[1]*d[2]-e[2]*d[1]:
[ exd:=matrix([exd1,exd2,exd3]):
[ exdT:=linalg::transpose(exd):
[ p:=matrix([e,exd]):
[ pT:=matrix([[eT,exdT]]):
[ izz:=iyy:=m*a^2/12:
[ Ip:=matrix([[0,0,0],[0,iyy,0],[0,0,izz]]):
[ cth:=cos(th):sth:=sin(th):
[ Q:=matrix([[cth,-sth,0],[sth,cth,0],[0,0,1]]):
[ Qt:=linalg::transpose(Q):
[ Ipp:=Q*Ip*Qt:
[ m1:=matrix([[m,0,0],[0,m,0],[0,0,m]]):
[ Om:=matrix([[0,0,0],[0,0,0],[0,0,0]]):
[ M:=matrix([[Ipp,Om],[Om,m1]]):
[ iner:=simplify(pT*M*p):
[ omv:=matrix([0,0,thd]):
[ omx1:=matrix([[0,-omv[3],0],[omv[3],0,0],[0,0,0]]):
[ W:=matrix([[omx1,Om],[Om,Om]]):
[ h:=pT*(M*W+W*M)*p:
[ n:=matrix([0,0,ta]):
[ f:=matrix([m*g,0,0]):
[ nT:=linalg::transpose(n):
[ w:=matrix([n,f]):
[ fT:=linalg::transpose(f):
[ wT:=matrix([[nT,fT]]):
[ tau:=pT*w
  [ ( ( τ -  $\frac{agm \sin(\theta)}{2}$  ) )

```

Final equation of motion

```

[ eq:=iner*thdd+h=tau
  [ ( (  $\frac{\bar{a}^2 m}{3}$  ) ) = ( ( τ -  $\frac{agm \sin(\theta)}{2}$  ) )

```

Fig. 9.4 MuPAD commands for the dynamics of one-link arm

Example 9.3 Two-link Planar Arm

A two-link planar arm with two revolute joints is shown in Fig. 9.5. The manipulator has two degrees of freedom, whose independent coordinates are the joint angles θ_1 and θ_2 . Correspondingly, the joint rates are $\dot{\theta}_1$ and $\dot{\theta}_2$. The 2-dimensional joint-rate vector $\dot{\theta}$ is then defined as

$$\dot{\theta} \equiv [\dot{\theta}_1, \dot{\theta}_2]^T \quad (9.25)$$

The link lengths are a_1 and a_2 , and the masses are m_1 and m_2 . The elements of the generalized inertia matrix (GIM) are then calculated using Eq. (9.15b) as follows: For $i, j = 2$, the calculation of i_{22} is shown below

$$i_{22} \equiv \mathbf{p}_2^T \tilde{\mathbf{M}}_2 \mathbf{B}_{22} \mathbf{p}_2 \quad (9.26a)$$

where

$$\mathbf{p}_2 \equiv \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_2 \times \mathbf{d}_2 \end{bmatrix}; \mathbf{B}_{22} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}; \text{ and } \tilde{\mathbf{M}}_2 = \mathbf{M}_2 \equiv \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & m_2 \mathbf{1} \end{bmatrix} \quad (9.26b)$$

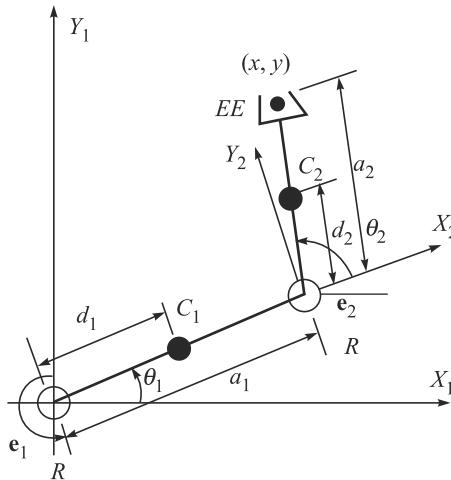


Fig. 9.5 Two-link planar arm

where $[\mathbf{e}_2]_2 \equiv [\mathbf{e}_2]_1 \equiv [0, 0, 1]^T$ is the unit vector along the axis of the second revolute joint, as indicated in Fig. 9.5. Vector \mathbf{d}_2 is the position of C_2 , whereas \mathbf{I}_2 is the inertia tensor about C_2 . Using Eq. (9.26b) and assuming the mass center lies at the center of the link length, i.e., $d_2 = a_2/2$, the (2,2)-element of the GIM i_{22} is then explicitly obtained, similar to $I(\equiv i_{11})$ of Example 9.1 as

$$\begin{aligned} i_{22} &= [\mathbf{e}_2]_1^T [\mathbf{I}_2]_1 [\mathbf{e}_2]_1 + m_2 [\mathbf{d}_2]_1^T [\mathbf{d}_2]_1 \\ &= \frac{1}{12} m_2 a_2^2 + \frac{1}{4} m_2 a_2^2 = \frac{1}{3} m_2 a_2^2 \end{aligned} \quad (9.27a)$$

Next, for $i = 2, j = 1$, the element i_{21} is given next as

$$i_{21} = \mathbf{p}_2^T \tilde{\mathbf{M}}_2 \mathbf{B}_{21} \mathbf{p}_1, \text{ where } \mathbf{p}_1 \equiv \begin{bmatrix} \mathbf{e}_1^T & (\mathbf{e}_1 \times \mathbf{d}_1)^T \end{bmatrix}^T \quad (9.27b)$$

and the matrix \mathbf{B}_{21} is given by

$$\mathbf{B}_{21} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -(\mathbf{r}_1 + \mathbf{d}_2) \times \mathbf{1} & \mathbf{1} \end{bmatrix} \quad (9.27c)$$

where $\mathbf{r}_1 = \mathbf{a}_1 - \mathbf{d}_1$. Hence, the (2,1)-element of the GIM, i.e., i_{21} is calculated, similar to i_{22} , as

$$\begin{aligned} i_{21} &= \mathbf{e}_2^T \mathbf{I}_2 \mathbf{e}_1 + m_2 (\mathbf{e}_2 \times \mathbf{d}_2)^T [\mathbf{e}_1 \times (\mathbf{a}_1 + \mathbf{d}_2)] = \mathbf{e}_2^T \mathbf{I}_2 \mathbf{e}_1 + m_2 \mathbf{d}_2^T \mathbf{a}_1 + m_2 \mathbf{d}_2^T \mathbf{d}_2 \\ &= \frac{1}{12} m_2 a_2^2 + \frac{1}{2} m_2 a_1 a_2 c_2 + \frac{1}{4} m_2 a_2^2 = \frac{1}{3} m_2 a_2^2 + \frac{1}{2} m_2 a_1 a_2 c_2 \end{aligned} \quad (9.27d)$$

In Eq. (9.27d), the vector product rule, namely, $(\mathbf{a} \times \mathbf{b})^T (\mathbf{c} \times \mathbf{d}) = (\mathbf{a}^T \mathbf{c})(\mathbf{b}^T \mathbf{d}) - (\mathbf{a}^T \mathbf{d})(\mathbf{b}^T \mathbf{c})$ was applied. Moreover, $\mathbf{e}_1 = \mathbf{e}_2$, $\mathbf{e}_2^T \mathbf{e}_2 = 1$ because \mathbf{e}_2 is a unit vector, and $\mathbf{e}_2^T \mathbf{d}_1 = 0$ because \mathbf{e}_2 is orthogonal to \mathbf{d}_1 . Furthermore, $\mathbf{d}_i = \mathbf{a}_i/2$, $\mathbf{r}_i + \mathbf{d}_i = \mathbf{a}_i$, for $i = 1, 2$, and $\mathbf{d}_2^T \mathbf{a}_1 = \frac{1}{2} a_1 a_2 c_2$. Next, matrix $\tilde{\mathbf{M}}_1$ is calculated as

$$\tilde{\mathbf{M}}_1 = \mathbf{M}_1 + \mathbf{B}_{21}^T \tilde{\mathbf{M}}_2 \mathbf{B}_{21} = \begin{bmatrix} \tilde{\mathbf{I}}_1 & -\tilde{\boldsymbol{\delta}}_1 \times \mathbf{1} \\ \tilde{\boldsymbol{\delta}}_1 \times \mathbf{1} & \tilde{m}_1 \mathbf{1} \end{bmatrix} \quad (9.28a)$$

where $\tilde{\mathbf{I}}_1$, $\tilde{\boldsymbol{\delta}}_1$, and \tilde{m}_1 are given below.

$$\tilde{\mathbf{I}}_1 = \mathbf{I}_1 + \mathbf{I}_2 - m_2 \mathbf{c}_{21} \times (\tilde{\boldsymbol{\delta}}_1 \times \mathbf{1}); \tilde{\boldsymbol{\delta}}_1 = m_2 \mathbf{c}_{21}; \text{ and } \tilde{m}_1 = m_1 + m_2 \quad (9.28b)$$

in which $\tilde{\mathbf{I}}_2 \equiv \mathbf{I}_2$, $\tilde{\boldsymbol{\delta}}_2 = \mathbf{0}$, and $\tilde{m}_2 = m_2$ are used. Now, for $i, j = 1$, the (1, 1)-element of the GIM i_{11} is now found as

$$i_{11} = \mathbf{e}_1^T \tilde{\mathbf{I}}_1 \mathbf{e}_1 + \tilde{m}_1 \mathbf{d}_1^T \mathbf{d}_1 = \frac{1}{3} (m_1 a_1^2 + m_2 a_2^2) + m_2 a_1^2 + m_2 a_1 a_2 c_2 \quad (9.29)$$

For $i, j = 2, 1$, the MCI elements are given from Eq. (9.16k) as

$$c_{22} = \mathbf{p}_2^T [\mathbf{B}_{32}^T \tilde{\mathbf{H}}_{32} + \mathbf{B}_{22}^T (\tilde{\mathbf{M}}_2 \boldsymbol{\Omega}_2 + \tilde{\mathbf{M}}'_2)] \mathbf{p}_2 = 0 \quad (9.30a)$$

$$c_{21} = \mathbf{p}_2^T [\tilde{\mathbf{H}}_{21} + \tilde{\mathbf{M}}_2 \mathbf{B}_{21} \boldsymbol{\Omega}_1 + \tilde{\mathbf{M}}'_2 \mathbf{B}_{21}] \mathbf{p}_1 = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1 \quad (9.30b)$$

$$c_{12} = \mathbf{p}_1^T [\mathbf{B}_{31}^T \tilde{\mathbf{H}}_{32} + \mathbf{B}_{21}^T (\tilde{\mathbf{M}}_2 \boldsymbol{\Omega}_2 + \tilde{\mathbf{M}}'_2)] \mathbf{p}_2 = -\frac{1}{2} m_2 a_1 a_2 s_2 (\dot{\theta}_1 + \dot{\theta}_2) \quad (9.30c)$$

$$c_{11} = \mathbf{p}_1^T [\mathbf{B}_{21}^T \tilde{\mathbf{H}}_{21} + \mathbf{B}_{11}^T (\tilde{\mathbf{M}}_1 \boldsymbol{\Omega}_1 + \tilde{\mathbf{M}}'_1)] \mathbf{p}_1 = -\frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_2 \quad (9.30d)$$

In Eqs. (9.30a-d), $\tilde{\mathbf{H}}_{32} \equiv \mathbf{0}$ and $\tilde{\mathbf{M}}'_2 \mathbf{p}_2 = \mathbf{0}$ were used. Moreover, for planar robots, it can be shown that the term $\mathbf{p}_i^T \mathbf{B}_{ji}^T \tilde{\mathbf{M}}_j \boldsymbol{\Omega}_j \mathbf{p}_j$, for $i, j = 1, 2$, vanishes. Finally, the elements of vector \mathbf{h} can be shown to have the following expressions:

$$h_2 = \mathbf{p}_2^T \tilde{\mathbf{w}}'_2 = \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1^2 \quad (9.31a)$$

$$h_1 = \mathbf{p}_1^T \tilde{\mathbf{w}}'_1 = -m_2 a_1 a_2 s_2 \left(\frac{1}{2} \dot{\theta}_2 + \dot{\theta}_1 \right) \dot{\theta}_2 \quad (9.31b)$$

which are nothing but the result of the multiplication of the elements of the MCI, Eqs. (9.30a-d), with those of the joint rate vector $\dot{\boldsymbol{\theta}}$ of Eq. (9.25).

9.3 RECURSIVE INVERSE DYNAMICS OF ROBOANALYZER[†]

Inverse dynamics is defined as “given a robot’s geometrical and inertial parameters, along with its joint motions, find the joint torques and forces.” In this section, it is emphasized that for the purpose of inverse dynamics analysis it is not required to explicitly evaluate the matrices and vectors appearing in Eq. (9.10a), as the required right-hand side can be evaluated using a recursive order (n) algorithm (Saha, 1999), where n is the number of links in the robot manipulator. A recursive algorithm in the C++ program was initially developed for the dynamic analysis of serial robots, which was given an acronym RIDIM (Recursive Inverse Dynamics for Industrial Manipulators) (Marothiya and Saha, 2003). Later, it was superseded with RoboAnalyzer (RA) which was written in C# and based on a similar algorithm called ReDySim (Recursive Dynamics Simulator) (Shah et al., 2013) using the DeNOC matrices. The RA in comparison to ReDySim has many additional features like 3-dimensional models of robots with animation, kinematic analyses, trajectory planning, etc. See Appendix C for more details.

The recursive algorithm (Saha, 1999) implemented in RoboAnalyzer is presented next. It has two recursions, namely, forward and backward. They are given below:

1. Forward Recursion (Kinematic Equations)

$$\mathbf{t}_1 = \mathbf{p}_1 \dot{\theta}_1; \quad \dot{\mathbf{t}}_1 = \mathbf{p}_1 \ddot{\theta}_1 + \boldsymbol{\Omega}_1 \mathbf{p}_1 \dot{\theta}_1 \quad (9.32a)$$

$$\begin{aligned} \mathbf{t}_2 &= \mathbf{B}_{21} \mathbf{t}_1 + \mathbf{p}_2 \dot{\theta}_2; & \dot{\mathbf{t}}_2 &= \mathbf{B}_{21} \dot{\mathbf{t}}_1 + \dot{\mathbf{B}}_{21} \mathbf{t}_1 + \mathbf{p}_2 \ddot{\theta}_2 + \boldsymbol{\Omega}_2 \mathbf{p}_2 \dot{\theta}_2 \\ &\vdots & &\vdots \end{aligned} \quad (9.32b)$$

$$\mathbf{t}_n = \mathbf{B}_{n,n-1} \mathbf{t}_{n-1} + \mathbf{p}_n \dot{\theta}_n; \quad \dot{\mathbf{t}}_n = \mathbf{B}_{n,n-1} \dot{\mathbf{t}}_{n-1} + \dot{\mathbf{B}}_{n,n-1} \mathbf{t}_{n-1} + \mathbf{p}_n \ddot{\theta}_n + \boldsymbol{\Omega}_n \mathbf{p}_n \dot{\theta}_n \quad (9.32c)$$

where the 6×6 matrix $\boldsymbol{\Omega}_i$ was defined as after Eq. (9.10b).

2. Backward Recursion (Dynamic Equations)

$$\mathbf{w}_n = \mathbf{M}_n \dot{\mathbf{t}}_n + \mathbf{W}_n \mathbf{M}_n \mathbf{t}_n; \quad \tilde{\mathbf{w}}_n = \mathbf{w}_n; \quad \tau_n = \mathbf{p}_n^T \tilde{\mathbf{w}}_n \quad (9.33a)$$

$$\begin{aligned} \mathbf{w}_{n-1} &= \mathbf{M}_{n-1} \dot{\mathbf{t}}_{n-1} + \mathbf{W}_{n-1} \mathbf{M}_{n-1} \mathbf{t}_{n-1}; \quad \tilde{\mathbf{w}}_{n-1} = \mathbf{w}_{n-1} + \mathbf{B}_{n,n-1}^T \tilde{\mathbf{w}}_n; \quad \tau_{n-1} = \mathbf{p}_{n-1}^T \tilde{\mathbf{w}}_{n-1} \\ &\vdots \end{aligned} \quad (9.33b)$$

$$\mathbf{w}_1 = \mathbf{M}_1 \dot{\mathbf{t}}_1 + \mathbf{W}_1 \mathbf{M}_1 \mathbf{t}_1; \quad \tilde{\mathbf{w}}_1 = \mathbf{w}_1 + \mathbf{B}_{21}^T \tilde{\mathbf{w}}_2; \quad \tau_1 = \mathbf{p}_1^T \tilde{\mathbf{w}}_1 \quad (9.33c)$$

where \mathbf{t}_i , $\dot{\mathbf{t}}_i$ and \mathbf{w}_i are the 6-dimensional vectors of twist, twist-rate, and the resultant wrench on the uncoupled i^{th} link. In the above algorithm, no moment and force, i.e., wrench, acting on the end-effector was considered. In case of their presence they should be appropriately included, i.e., Eq. (9.33a) is modified as

$$\mathbf{w}_n = \mathbf{M}_n \dot{\mathbf{t}}_n + \mathbf{W}_n \mathbf{M}_n \mathbf{t}_n - \mathbf{w}_n^N \quad (9.34)$$

where \mathbf{w}_n^N is the wrench acting on the end-effector by the environment. Rest of

Multiplication vs. Division

In computer calculations, a multiplication is treated similar to a division due to the internal algorithms used by a computer. Hence, $(a^*b)/c$ needs two multiplications (M).

[†] The dynamics algorithm of RoboAnalyzer reported here is basically same as that of Recursive Inverse Dynamics for Industrial Manipulators (RIDIM) appeared in the 1st edition of this book.

the step remains same. Moreover, if gravity is present it is taken into account by providing negative acceleration due to the gravity, denoted by \mathbf{g} , to the first body, i.e., link #1 (Luh, Walker and Paul, 1980). Accordingly, $\dot{\mathbf{t}}_1$ of Eq. (9.32a) is modified as

$$\dot{\mathbf{t}}_1 = \mathbf{p}_1 \ddot{\theta}_1 + \boldsymbol{\Omega}_1 \mathbf{p}_1 \dot{\theta}_1 + \boldsymbol{\rho}, \text{ where } \boldsymbol{\rho} \equiv [\mathbf{0}, -\mathbf{g}^T]^T \quad (9.35)$$

Now, for an all revolute-jointed serial robotic manipulator, the computational complexity of the above algorithm is shown in Table 9.1, which is compared with some other existing inverse dynamics algorithms. The details of how to calculate the computational complexity are appeared in Appendix A of Shah et al. (2013). The above algorithm is one of the best. Besides, it is very simple, as evident from the two-step algorithm where six-dimensional vectors are treated similar to those of three dimensional vectors of recursive Newton-Euler algorithm as presented in Chapter 8. A close look into the algorithms actually exposes that both are fundamentally same. The latter one is, however, more elegant and concise. That is, if one knows $\dot{\mathbf{t}}_1$ finding out $\dot{\mathbf{t}}_2$ is very similar to the evaluation of the linear velocity of link #2 from the known linear velocity of link #1.

Addition vs. Subtraction

In computer calculations, an addition is similar to a subtraction due to the internal computer algorithms. Hence, computation count to find $a + b - c$ is two additions (A).

Table 9.1 Computational complexities for inverse dynamics

Algorithm	Multiplications/ Divisions (M)	Additions/ Subtractions (A)	<i>n</i> = 6	
Hollerbach (1980)	$412n - 277$	$320n - 201$	2195M	1719A
Luh et al. (1980)	$150n - 48$	$131n + 48$	852M	834A
Walker and Orin (1982)	$137n - 22$	$101n - 11$	800M	595A
RIDIM (Saha, 1999)	$120n - 44$	$97n - 55$	676M	527A
Khalil et al. (1986)	$105n - 92$	$94n - 86$	538M	478A
Angeles et al. (1989)	$105n - 109$	$90n - 105$	521M	435A
ReDySim (Shah et al., 2013)	$94n - 81$	$82n - 75$	483M	417A
Balafoutis et al. (1991)	$93n - 69$	$81n - 65$	489M	421M

Example 9.4 Inverse Dynamics of Three-DOF Planar Arm

The three-link three-DOF manipulator under study, is shown in Fig. 9.6, whose DH and inertia parameters are shown in Table 9.2. It is assumed that the manipulator moves on the *X-Y* plane, where the gravity is working in the negative *Y* direction. Let \mathbf{i} and \mathbf{j} be the two unit vectors parallel to the axes *X* and *Y*, respectively, and $\mathbf{k} = \mathbf{i} \times \mathbf{j}$ is the one parallel to *Z*-axis. The three joint torques, namely, τ_1 , τ_2 , and τ_3 , were evaluated using RoboAnalyzer software. The joint angles used to generate the joint torques are as follows: For $i = 1, 2$ and 3 ,

$$\theta_i = \theta_i(0) + \frac{\theta_i(T) - \theta_i(0)}{T} \left[t - \frac{T}{2\pi} \sin\left(\frac{2\pi}{T}t\right) \right] \quad (9.36a)$$

$$\dot{\theta}_i = \frac{\theta_i(T) - \theta_i(0)}{T} \left[1 - \cos\left(\frac{2\pi}{T}t\right) \right] \quad (9.36b)$$

$$\ddot{\theta}_i = \frac{\theta_i(T) - \theta_i(0)}{T} \left[\frac{2\pi}{T} \sin\left(\frac{2\pi}{T}t\right) \right] \quad (9.36c)$$

where $T = 10$ seconds. The initial and final joint values were taken as $\theta_i(0) = 0$ and $\theta_i(T) = \pi$. Moreover, Eqs. (9.36a-c) guarantee that $\dot{\theta}_i(0) = \ddot{\theta}_i(0) = \dot{\theta}_i(T) = \ddot{\theta}_i(T) = 0$, for $i = 1, 2, 3$. The plots for the joint angle and their time derivatives are shown in Fig. 9.7, the joint torques obtained from RoboAnalyzer are then plotted in Fig. 9.8, where the terms, τ_1 , τ_2 , and τ_3 , are the joint torques. The plots were also verified with those obtained from the explicit expressions available in many textbooks on robotics, e.g., Angeles (2003).

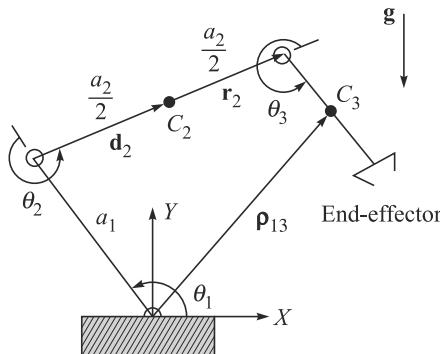


Fig. 9.6 Three-DOF planar robot arm

Table 9.2 The DH and inertia parameters of the three-DOF robot arm

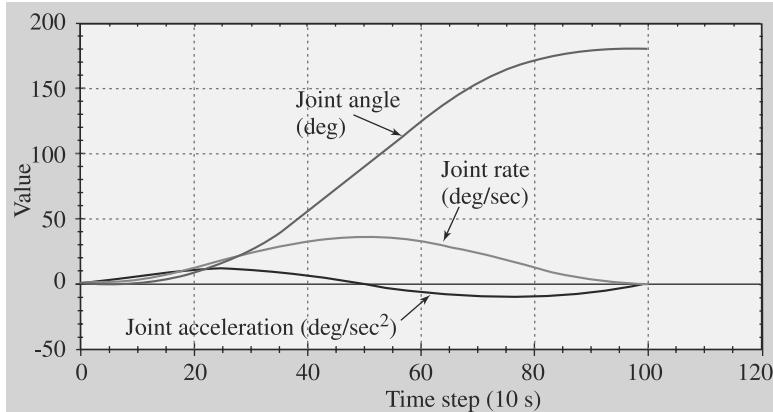
(a) DH parameters

Link	Joint	a_i (m)	b_i (m)	α_i (rad)	θ_i (rad)
1	r	0.3	0	0	JV [0]
2	r	0.25	0	0	JV [0]
3	r	0.22	0	0	JV [0]

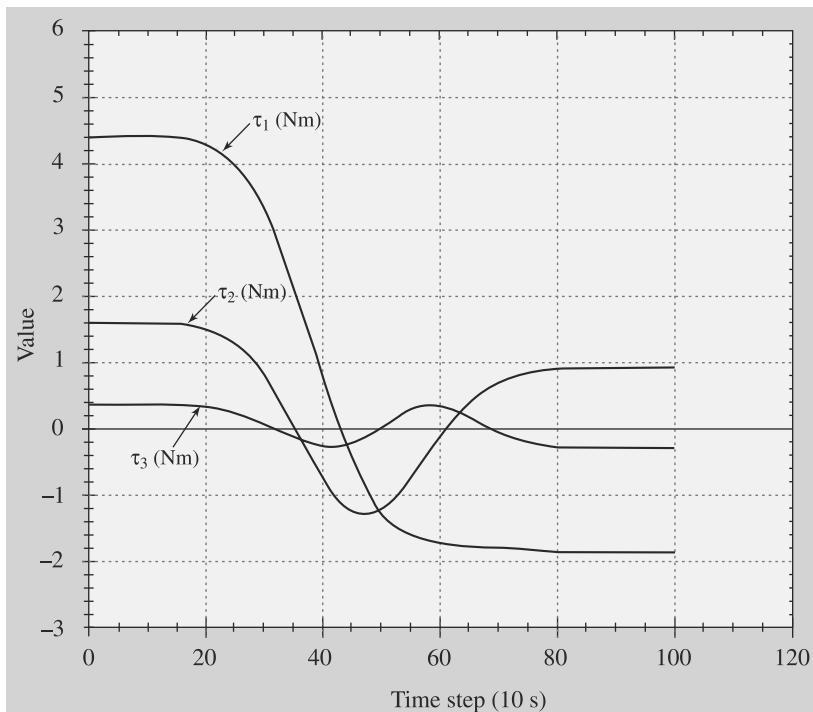
JV: Joint variable with initial values inside brackets [and]; r: Revolute joint

(b) Mass and inertia parameters

Link	m_i (kg)	$r_{i,x}$ (m)	$r_{i,y}$ (m)	$r_{i,z}$ (m)	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
	(m)				(kg·m ²)					
1	0.5	0.15	0	0	0	0	0	0.00375	0	0.00375
2	0.4	0.125	0	0	0	0	0	0.00208	0	0.00208
3	0.3	0.11	0	0	0	0	0	0.00121	0	0.00121

**Fig. 9.7** Joint trajectory

A screenshot of the 3-dimensional view of the 3-DOF robot arm in RoboAnalyzer with its kinematic parameters visible at the bottom of the screen are shown in Fig. 9.9.

**Fig. 9.8** Joint torques for the three-DOF arm

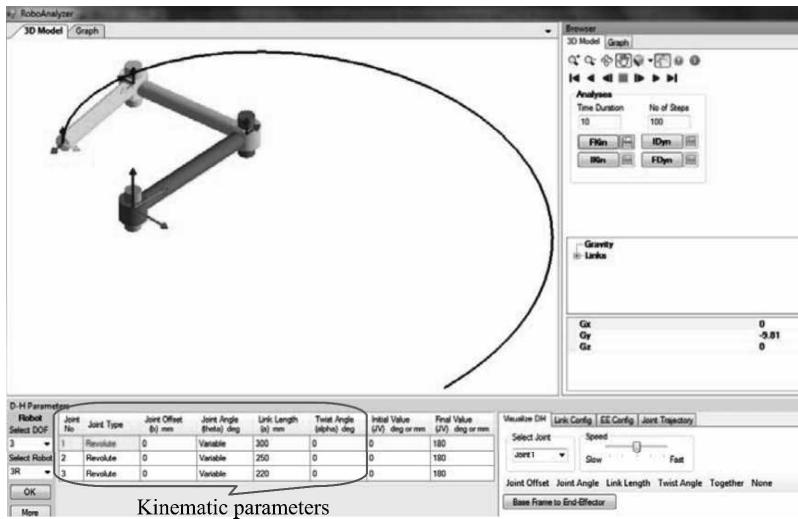


Fig. 9.9 RoboAnalyzer screenshot of three-link robot arm

Example 9.5 Inverse Dynamics of PUMA Architecture

For the six-DOF robot with PUMA architecture shown in Fig. 9.10, the inverse dynamics results were obtained for its DH and inertial parameters shown in Table 9.3.

Table 9.3 The DH and inertia parameters of PUMA robot

(a) DH parameters

Link	Joint	b_i (m)	θ_i (rad)	a_i (m)	α_i (rad)
1	r	0	JV [0]	0	$-\pi/2$
2	r	0.149	JV [0]	0.432	0
3	r	0	JV [0]	0.02	$-\pi/2$
4	r	0.432	JV [0]	0	$-\pi/2$
5	r	0	JV [0]	0	$-\pi/2$
6	r	0.05	JV [0]	0	0

JV: Joint variable with initial values inside [] and ; r: Revolute joint

(b) Mass and inertia parameters

Link	m_i (kg)	$r_{i,x}$ (m)	$r_{i,y}$ (m)	$r_{i,z}$ (m)	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
					(kg-m ²)	(kg-m ²)				
1	10.521	0	0	0.054	1.612	0	0	1.612	0	0.5091
2	15.761	0.292	0	0	0.4898	0	0	8.0783	0	8.2672
3	8.767	0.02	0	-0.197	3.3768	0	0	3.3768	0	0.3009
4	1.052	0	-0.057	0	0.181	0	0	0.1273	0	0.181
5	1.052	0	0	-0.007	0.0735	0	0	0.1273	0	0.0735
6	0.351	0	0	0.019	0.0071	0	0	0.0071	0	0.0141

Note that the DH frames in Fig. 9.10 are assigned little differently, namely, frames 5 and 6, compared to those in Fig. 6.6 for the same robot. Hence, there are some changes in the DH parameters in Table 9.3 compared to those in Table 6.2. This was done intentionally to illustrate the variation in the assignment of DH frames for the same architecture of a robot. The trajectory functions for each joint is taken same as for the three-link robot arm, as defined in Eq. (9.36a-c) for the same T , and initial and final joint values. Corresponding joint torque plots generated from RoboAnalyzer software are given in Figs. 9.11(a-f).

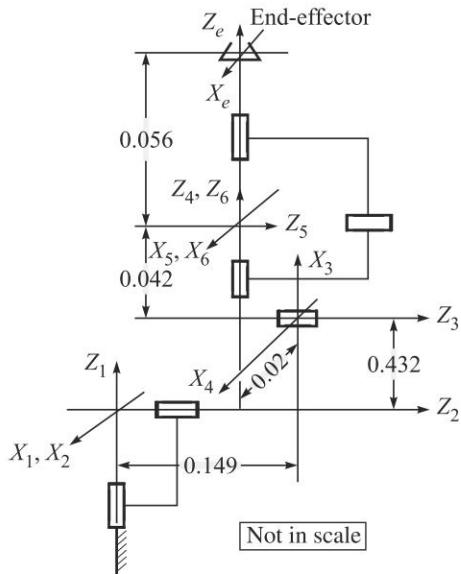
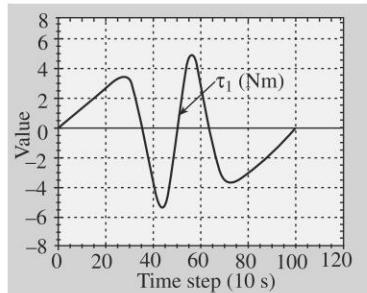
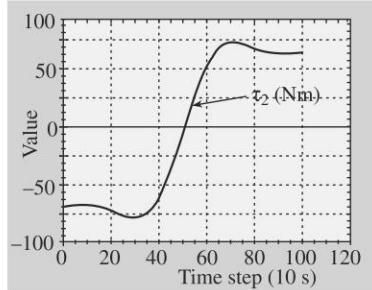


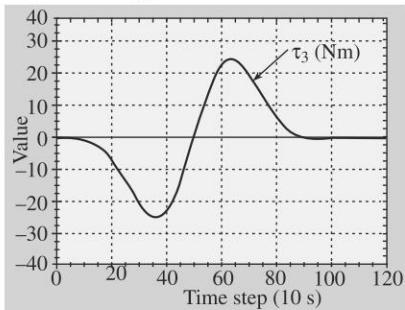
Fig. 9.10 A PUMA architecture



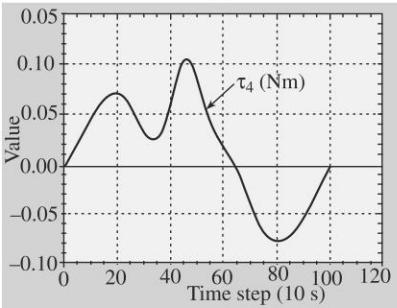
(a) Joint 1



(b) Joint 2



(b) Joint 3



(d) Joint 4

(Contd.)

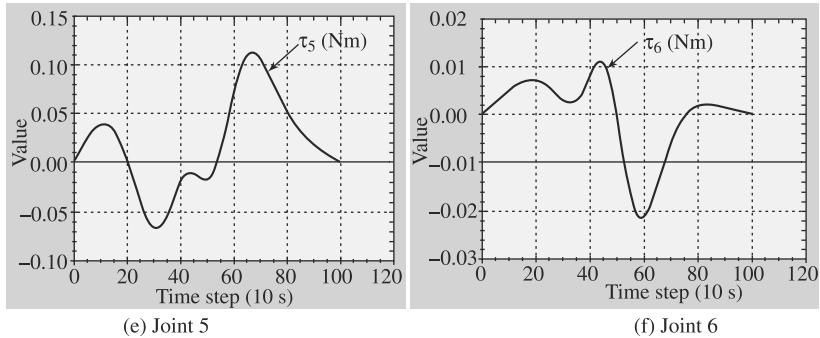


Fig. 9.11 Joint torques for PUMA architecture

Example 9.6 Inverse Dynamics of Stanford Arm

For the Stanford arm shown in Fig. 9.12, the DH and other parameters are shown in Table 9.4. Note that it differentiates from the PUMA robot in a way that it has a prismatic joint in joint location 3. Moreover, the DH frames are assigned differently than in Fig. 6.8(a) due to the reasons cited for PUMA robot of Example 9.5. The functions for the revolute joints were taken same as in Example 9.5, whereas the function of the prismatic joint variable, namely b_3 is taken as

$$b_i = b_i(0) + \frac{b_i(T) - b_i(0)}{T} \left[t - \frac{T}{2\pi} \sin\left(\frac{2\pi}{T}t\right) \right] \quad (9.37)$$

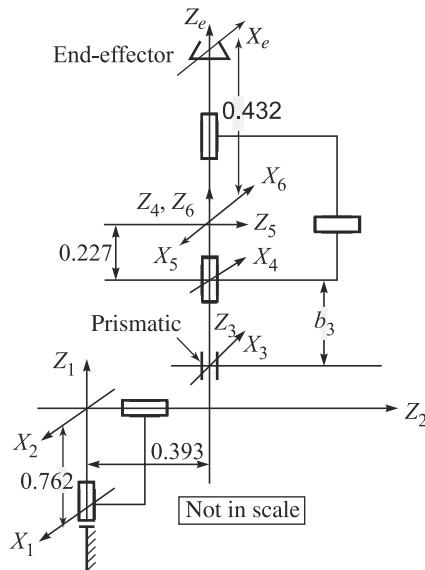


Fig. 9.12 The Stanford arm

Table 9.4 The DH and inertia parameters of the Stanford arm**(a) DH parameters**

Link	Joint	b_i (m)	θ_i (rad)	a_i (m)	α_i (rad)
1	r	0.762	JV [0]	0	$-\pi/2$
2	r	0.393	JV [- $\pi/2$]	0	$\pi/2$
3	p	JV [0.635]	0	0	0
4	r	0.227	JV [0]	0	$-\pi/2$
5	r	0	JV [π]	0	$-\pi/2$
6	r	0.432	JV [π]	0	0

JV: Joint variable with initial values within [and]; r: Revolute joint; p: Prismatic joint

(b) Mass and inertia parameters

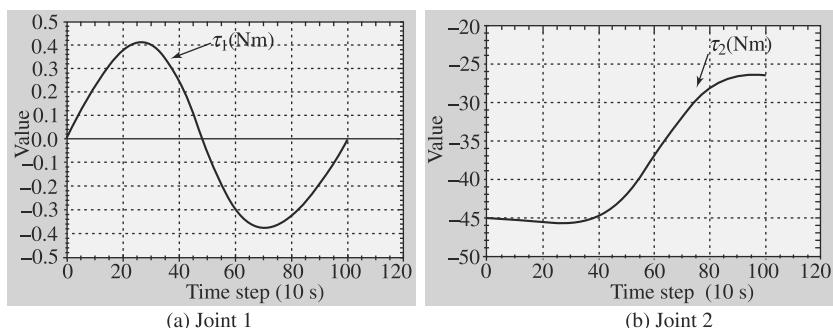
Link	m_i (kg)	$r_{i,x}$ (m)	$r_{i,y}$ (m)	$r_{i,z}$ (m)	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
(kg·m²)										
1	9	0	-0.1	0	0.01	0	0	0.02	0	0.01
2	6	0	0	0	0.05	0	0	0.06	0	0.01
3	4	0	0	0	0.4	0	0	0.4	0	0.01
4	1	0	-0.1	0	0.001	0	0	0.001	0	0.0005
5	0.6	0	0	-0.06	0.0005	0	0	0.0005	0	0.0002
6	0.5	0	0	-0.2	0.003	0	0	0.001	0	0.002

For $T = 10$ seconds, initial joint values were taken as follows:

$$\theta_1(0) = 0, \theta_2(0) = -\pi/2, b_3(0) = 0.635 \text{ m}, \theta_4(0) = 0, \theta_5(0) = \theta_6(0) = \pi$$

$$\theta_1(T) = \pi/3, \theta_2(T) = -5\pi/6, b_3(T) = 0.735 \text{ m}, \theta_4(T) = \pi/3, \theta_5(T) = \theta_6(T) = 5\pi/6$$

The above values were taken in a way so that the robot links do not interfere with each other. The joint torques and force were obtained using RoboAnalyzer are plotted in Figs. 9.13(a-f).



(Contd.)

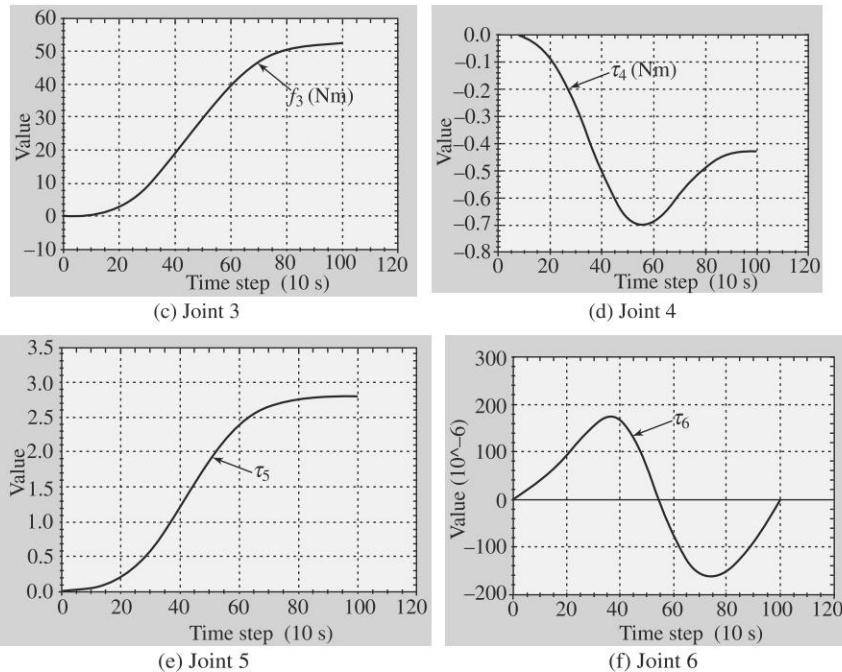


Fig. 9.13 Joint torques and force for Stanford arm

Example 9.7 Inverse Dynamics of KUKA KR-5 Robot

In this example, a practically used industrial robot, namely, KUKA KR-5 (5 stands for the 5 kg payload capacity of the robot), is taken up. The CAD model available from the company's website (WR: Kuka) has been processed in Autodesk Inventor software to extract the DH parameters of the robot using an in-house developed add-in for the Autodesk Inventor software (Rajeevlochana, 2012). They are shown in Table 9.5(a), whereas the mass and inertia properties were taken from the CAD model, as shown in Table 9.5(b). The function of the trajectory for each joint was taken same as in Eq. (9.36a-c) with $\theta_i(0) = 0$ and $\theta_i(T) = \pi/3$, and $T = 10$ seconds. Here, the joint trajectories were taken in a way that the robot links do not interfere with each other. The Cartesian trajectory followed by the robot is shown in Fig. 9.14, whereas the required torque plots are given in Fig. 9.15.

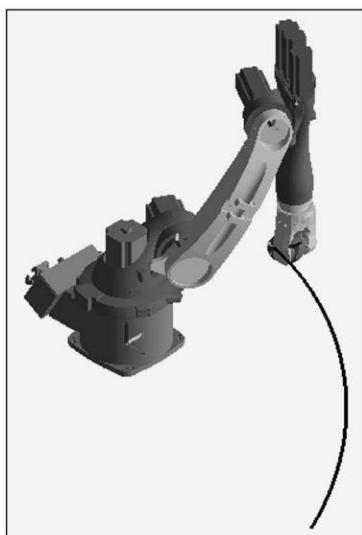


Fig. 9.14 KUKA KR-5 during its motion

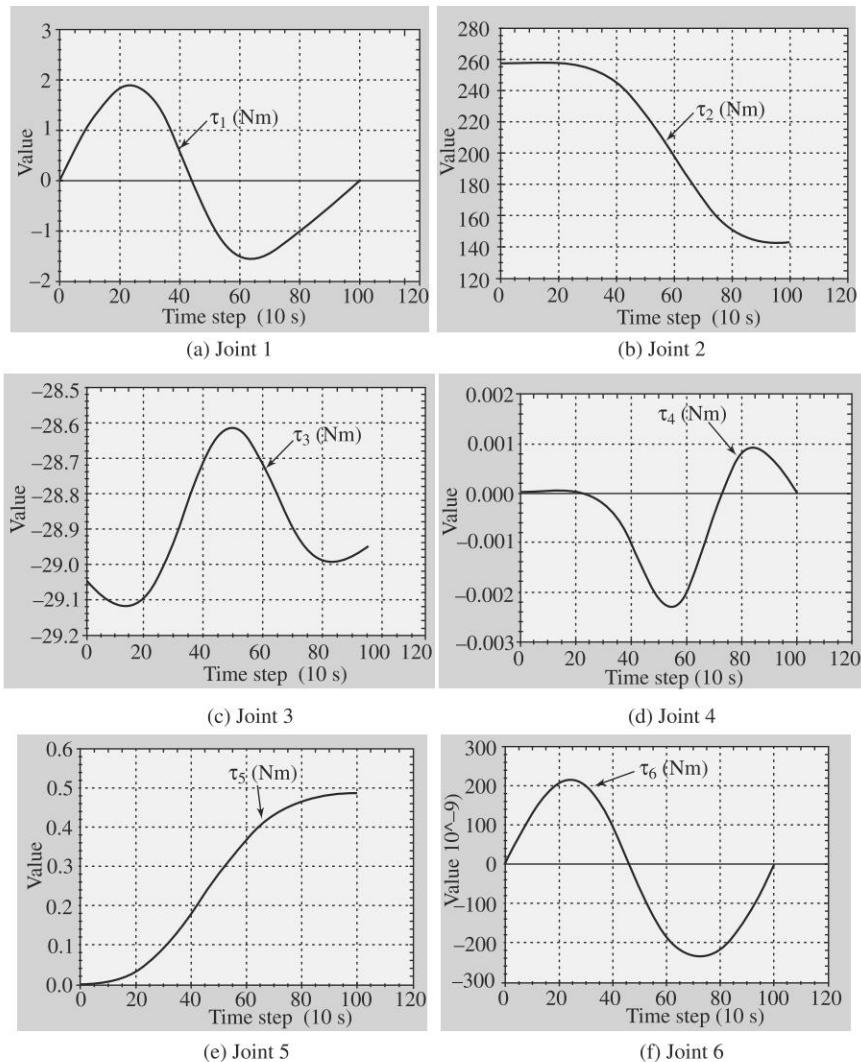


Fig. 9.15 Joint torques for KUKA KR-5

Table 9.5 The DH and inertia parameters of KUKA KR-5 robot

(a) DH parameters

Link	Joint	b_i (m)	θ_i (rad)	a_i (m)	α_i (rad)
1	r	0.4	JV [0]	0.180	$\pi/2$
2	r	0.135	JV [0]	0.600	π
3	r	0.135	JV [0]	0.120	$-\pi/2$
4	r	0.620	JV [0]	0	$\pi/2$
5	r	0	JV [0]	0	$-\pi/2$
6	r	0.115	JV [0]	0	0

JV: Joint variable with initial values within [] and ; r: Revolute joint

(Contd.)

(b) Mass and inertia parameters

m_i	$r_{i,x}$	$r_{i,y}$	$r_{i,z}$	$I_{i,xx}$	$I_{i,xy}$	$I_{i,xz}$	$I_{i,yy}$	$I_{i,yz}$	$I_{i,zz}$
(kg)	(m)			(kg·m ²)					
26.98	0.091	0.067	0.006	0.322	-0.018	-0.145	0.467	-0.014	0.478
15.92	0.333	0.002	0.039	0.541	0.000435	-0.005	0.552	0.017	0.044
25.852	0.032	-0.008	-0.034	0.775	-0.009	0.025	0.75	0.007	0.208
4.008	0	0.109	-0.008	0.01	0.002	0	0.02	0	0.024
1.615	0	-0.01	-0.033	0.002	0	0	0.004	0	0.004
0.016	0	0	-0.111	0	0	0	0	0	0

9.4 RECURSIVE FORWARD DYNAMICS AND SIMULATION

Forward dynamics is defined as “given the joint torques and forces, along with the robot’s physical parameters, find the joint accelerations, i.e., solve for the joint accelerations $\ddot{\theta}$ from the dynamic equations of motion Eq. (9.10a).” Simulation, on the other hand, involves forward dynamics followed by the solution of the differential equations in joint accelerations to obtain the joint velocities and positions, i.e., $\dot{\theta}$ and θ , respectively, for a given set of initial joint rates and positions of the manipulator under study, i.e., $\dot{\theta}(0)$ and $\theta(0)$.

Why “Forward?”

This step of simulation leads to the behavior of a robot as if a the robot exists and one moves it. Hence, the word *forward* is used.

9.4.1 Recursive Forward Dynamics Algorithm

Conventionally, joint accelerations are solved from Eq. (9.10a) using the Cholesky decomposition of the GIM \mathbf{I} , as done by Walker and Orin (1982), Angeles (2003), and others, or using the MATLAB command “ $\mathbf{I}\phi$ ” (WR-Matlab), where ϕ represents the vector of generalized forces due to external moments and forces, gravity and Coriolis terms, etc. The above approach requires order (n^3)— n being the number of links or joints in the robot—computations, and produces nonsmooth joint accelerations (Ascher et al., 1997). On the contrary, the dynamic formulation based on the Decoupled Natural Orthogonal Complement (DeNOC) matrices presented in this chapter allows one to solve $\ddot{\theta}$ from Eq. (9.10a) recursively with order (n) computational complexity (Saha, 1999; 2003). Such recursive algorithms are known to provide smooth profiles for $\ddot{\theta}$, as reported in Ascher et al. (1997), Mohan and Saha (2007), Shah et al. (2013), and others.

In this section, a recursive order (n) forward dynamics algorithm is presented, which requires the following inputs: For $i = 1, \dots, n$,

1. DH parameters, and the mass and inertia properties of all links, as they are shown in Tables 9.2–9.4.
2. Initial values for the variable DH parameters, i.e., θ_i , for a revolute joint, and b_i , for a prismatic joint, and their first time derivatives, i.e., $\dot{\theta}_i$ s and \dot{b}_i s.
3. Time history of the input joint forces/torques, i.e., τ_i .

4. Each component of the vector, $\phi \equiv \tau - \mathbf{h}$, obtained from Eq. (9.10a), i.e., ϕ_p which is to be calculated recursively using an inverse dynamics algorithm, e.g., the one given in Section 9.3 while $\ddot{\theta} = \mathbf{0}$.

The recursive forward dynamics algorithm presented here is based on the \mathbf{UDU}^T decomposition of the generalized inertia matrix, \mathbf{I} of Eq. (9.10a), i.e., $\mathbf{I} = \mathbf{UDU}^T$, where \mathbf{U} and \mathbf{D} are the upper triangular and diagonal matrices, respectively. Moreover, substituting $\mathbf{I} = \mathbf{UDU}^T$ and $\boldsymbol{\varphi} \equiv \boldsymbol{\tau} - \mathbf{h}$ in Eq. (9.10a), one obtains

$$\mathbf{UDU}^T \ddot{\theta} \equiv \boldsymbol{\phi} \quad (9.38)$$

A three step recursive scheme is then used to calculate the joint accelerations from the above equations, i.e.,

- 1. Solution for $\hat{\phi}$:** The solution $\hat{\phi} = \mathbf{U}^{-1}\boldsymbol{\phi}$ is evaluated in terms of the scalar terms as

$$\hat{\phi}_i = \phi_i - \mathbf{p}_i^T \boldsymbol{\eta}_{i,i+1}, \text{ for } i = n, \dots, 1 \quad (9.39a)$$

Note $\hat{\phi}_n \equiv \phi_n$, and the 6-dimensional vector $\boldsymbol{\eta}_{i,i+1}$ is obtained as

$$\boldsymbol{\eta}_{i,i+1} \equiv \mathbf{B}_{i+1,i}^T \boldsymbol{\eta}_{i+1} \text{ and } \boldsymbol{\eta}_{i+1} \equiv \boldsymbol{\psi}_{i+1} \hat{\phi}_{i+1} + \boldsymbol{\eta}_{i+1,i+2} \quad (9.39b)$$

in which $\boldsymbol{\eta}_{n,n+1} = \mathbf{0}$. The new variable $\boldsymbol{\psi}_{i+1}$ is the 6×6 matrix which is evaluated using

$$\boldsymbol{\psi}_i \equiv \frac{\hat{\boldsymbol{\Psi}}_i}{\hat{m}_i}, \text{ where } \hat{\boldsymbol{\Psi}}_i \equiv \hat{\mathbf{M}}_i \mathbf{p}_i \text{ and } \hat{m}_i \equiv \mathbf{p}_i^T \hat{\boldsymbol{\Psi}}_i \quad (9.39c)$$

In Eq. (9.39c), the 6×6 matrix $\hat{\mathbf{M}}_i$ called the *articulated body inertia* (Saha, 1997; 1999) that can be obtained recursively, similar to $\tilde{\mathbf{M}}_i$ in Eq. (9.11d), as

$$\hat{\mathbf{M}}_i \equiv \mathbf{M}_i + \mathbf{B}_{i+1,i}^T \bar{\mathbf{M}}_{i+1} \mathbf{B}_{i+1,i}, \text{ where } \bar{\mathbf{M}}_{i+1} \equiv \hat{\mathbf{M}}_{i+1} - \hat{\boldsymbol{\Psi}}_{i+1} \boldsymbol{\psi}_{i+1}^T \quad (9.39d)$$

for $i = n-1, \dots, 1$, and $\hat{\mathbf{M}}_n = \mathbf{M}_n$.

- 2. Solution for $\tilde{\phi}$:** The solution $\tilde{\phi} = \mathbf{D}^{-1}\hat{\phi}$ involves the inverse of the diagonal matrix, \mathbf{D} of Eq. (9.38), which is simple. The inverse \mathbf{D}^{-1} has only nonzero diagonal elements that are the reciprocal of the corresponding diagonal elements of \mathbf{D} . Vector $\tilde{\phi}$ is obtained as follows: For $i = 1, \dots, n$,

$$\tilde{\phi}_i = \hat{\phi}_i / \hat{m}_i \quad (9.40)$$

The scalar \hat{m}_i is defined in Eq.(9.39c).

- 3. Solution for $\ddot{\theta}$:** In this step, $\ddot{\theta} = \mathbf{U}^{-T}\tilde{\phi}$ is calculated for $i = 2, \dots, n$ as

$$\ddot{\theta}_i = \tilde{\phi}_i - \tilde{\boldsymbol{\psi}}_i^T \tilde{\boldsymbol{\mu}}_{i,i-1} \quad (9.41a)$$

where $\ddot{\theta}_1 \equiv \tilde{\phi}_1$, and the 6-dimensional vector $\tilde{\boldsymbol{\mu}}_{i,i-1}$ is obtained as

$$\tilde{\boldsymbol{\mu}}_{i,i-1} \equiv \mathbf{B}_{i,i-1} \tilde{\boldsymbol{\mu}}_{i-1} \text{ and } \tilde{\boldsymbol{\mu}}_{i-1} \equiv \mathbf{p}_{i-1} \ddot{\theta}_{i-1} + \tilde{\boldsymbol{\mu}}_{i-1,i-2} \quad (9.41b)$$

in which $\tilde{\boldsymbol{\mu}}_{10} \equiv \mathbf{0}$. The above forward dynamics algorithm (Saha, 1997; 2003) is the basis for ReDySim (Shah et al., 2013) which can also take care of the tree-type systems like biped and walking robots with one-, two-, and three-DOF joints, i.e., revolute, universal and spherical joints, respectively. Table 9.6 shows the comparison of various forward dynamics algorithms, along with the one for ReDySim which was

adopted in RoboAnalyzer software. Typically, recursive dynamics algorithms are known to be computationally efficient when $n \geq 10$ or 12. However, the ReDySim-based algorithm adopted in RoboAnalyzer is efficient even for $n \geq 7$.

It is pointed out here that the use of the DeNOC matrices for forward dynamics is more advantageous in comparison to inverse dynamics algorithm, as evident from Tables 9.1 and 9.6. Besides, as shown in Agrawal (2013), the ReDySim-based forward dynamics algorithm is extremely stable numerically for the reason explained in the paper.

9.4.2 Simulation

Simulation consists of forward dynamics to find the joint acceleration $\ddot{\theta}$ followed by its integration to obtain $\dot{\theta}$ and θ for a given set of initial conditions, i.e., $\dot{\theta}(0)$ and $\theta(0)$. As pointed out in Chapter 8 that except in extremely simple cases the integration needs to be done numerically either using well-established methods like Runge–Kutta–Fehlberg, Adams–Bashforth–Moulton and others (Shampine, 1994), or MATLAB commands like “ODE45”, etc. (WR-Matlab). In this section, simulation results were generated using RoboAnalyzer where a computer program in C# was written for the numerical integration using Runge–Kutta–Fehlberg formula.

Table 9.6 Computational complexities for forward dynamics

Algorithm	Multiplications/ Divisions (M)	Additions/ Subtractions (A)	$n = 6$	$n = 7$	$n = 10$
ReDySim (Shah et al., 2013)	$135n - 116$	$131n - 123$	694M 663A	829M 794A	1234M 1187A
Saha (2003)	$191n - 284$	$187n - 325$	862M 797A	1053M 984A	1626M 1545A
Featherstone (1983)	$199n - 198$	$174n - 173$	996M 871A	1195M 1045A	1792M 1527A
Stejskal and Valasek (1996)	$226n - 343$	$206n - 345$	1013M 891A	1239M 1097A	1917M 1715A
Brandl et al. (1988)	$250n - 222$	$220n - 198$	1278M 1122A	1528M 1342A	2278M 2002A
Walker and Orin (1982)	$n^3/6 + 23n^2/2 + 115n/3 - 47$	$n^3/6 + 7n^2 + 233n/3 - 46$	633M 480A	842M 898A	1653M 1209A

An important aspect of smooth joint acceleration profiles available from a recursive forward dynamics algorithm, as mentioned in Section 9.4.1, is elaborated here. When the joint acceleration profile for $\ddot{\theta}$ is not smooth convergence of its numerical integration to obtain the joint velocity and position, i.e., $\dot{\theta}$ and θ , is slow. Alternatively, with smooth $\ddot{\theta}$ profile obtained from a recursive forward dynamics algorithm convergence of the numerical integration results is much faster. Hence, the overall CPU time required for the forward dynamics and numerical integration together using the recursive algorithm may be smaller even for $n = 6$ compared to an order (n^3) algorithm which requires less forward dynamics computations when $n = 6$ (see Table 9.6) but may require more time to perform the numerical integration. This aspect was proven by Mohan and Saha (2007b).

Example 9.8 Simulation of the Three-DOF Planar Arm

The three-DOF arm shown in Fig. 9.6 is considered here to let fall freely under gravity from the horizontal initial configuration with no initial motion, i.e., $\theta_i(0) = \dot{\theta}_i(0) = 0$, for $i = 1, 2, 3$. Time step ΔT for the numerical integration was taken as $\Delta T = 0.01$ second. The results for the joint positions, namely, the variations of θ_1 , θ_2 , and θ_3 with time are shown in Fig. 9.16. Note from Fig. 9.6 that due to the gravity the first joint angle θ_1 will increase initially in the negative direction. This is evident from Fig. 9.16. Moreover, the system under gravity behaves as a three-link pendulum, which is clear from all the joint-angle variations of Fig. 9.16.

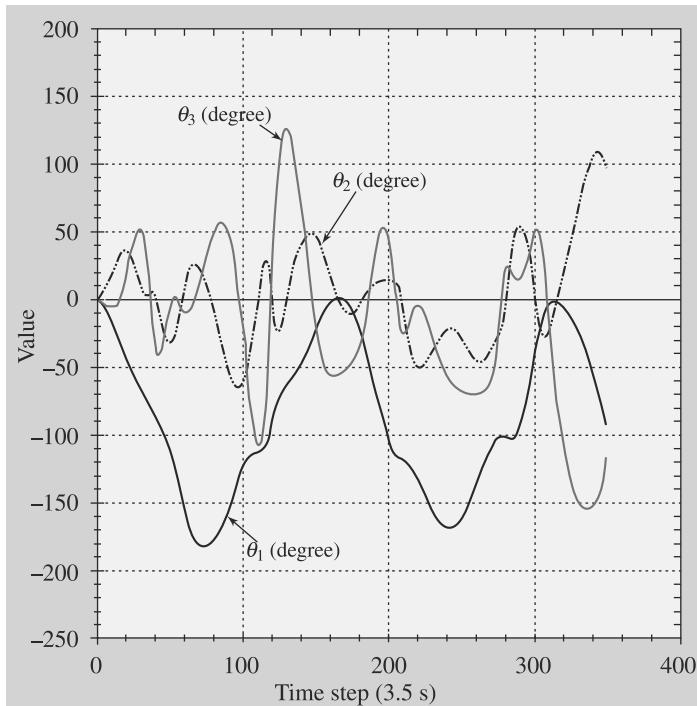


Fig. 9.16 Joint angles for the three-DOF arm

Example 9.9 Simulation of the PUMA Architecture

The robot with PUMA architecture is shown in Fig. 9.10. Its free-fall simulation, i.e., the robot was let fall freely under gravity, was carried out using RoboAnalyzer with the initial conditions of $\theta_i(0) = \dot{\theta}_i(0) = 0$, for $i = 1, \dots, 6$. The time step $\Delta T = 0.01$ second was taken for the numerical integration. Variations of the joint angles versus time are shown in Figs. 9.17(a–f). It is clear from Fig. 9.10 that due to the length $a_3 = 0.02$, joint 2 will rotate in the positive direction, which is evident from Fig. 9.17(b).

What is “Free”?

The joints are free without any applied torques.

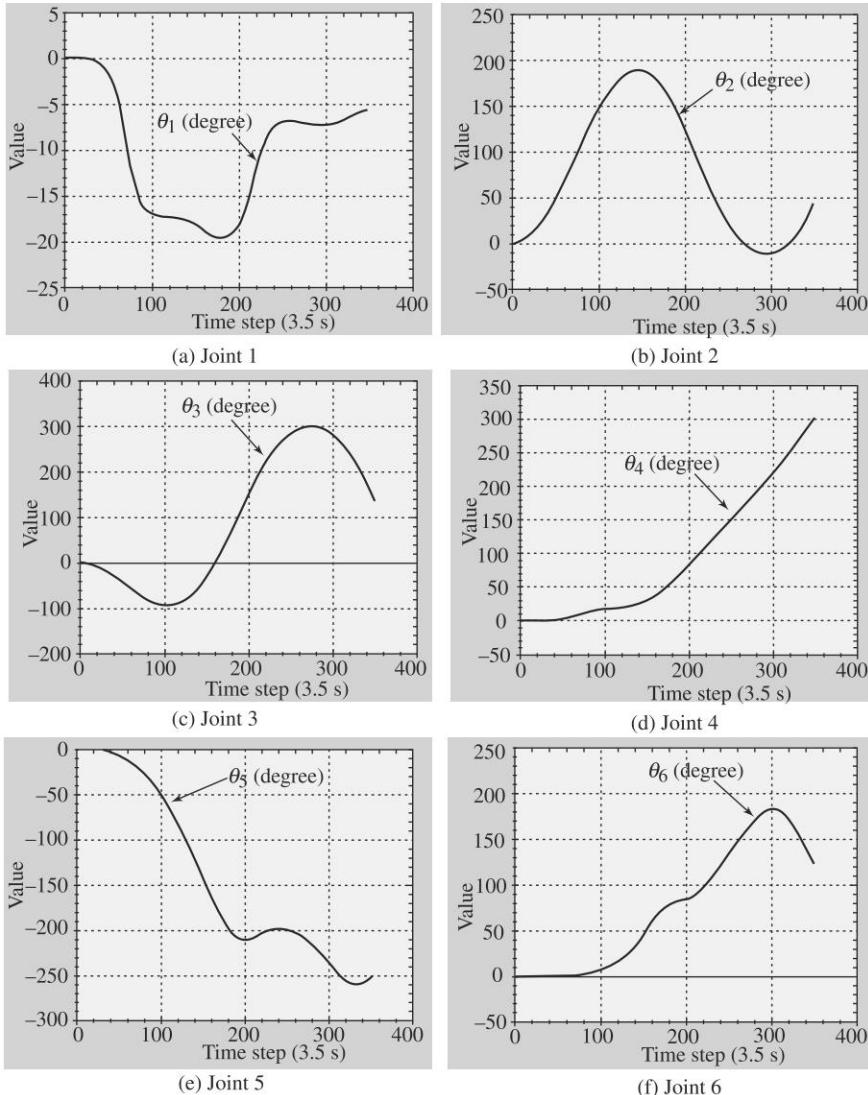


Fig. 9.17 Joint angles for the PUMA architecture

Example 9.10 Simulation of the Stanford Arm

Free-fall simulation of the Stanford arm shown in Fig. 9.12 was carried out with the following initial values of the joint positions and rates:

$$\begin{aligned} \theta_1(0) &= 0, \theta_2(0) = -\pi/2, b_3(0) = 0.635 \text{ m}, \theta_4(0) = 0, \theta_5(0) = \theta_6(0) = \pi \\ \dot{\theta}_i(0) &= 0, \text{ for } i = 1, 2, 4, 5, 6, \text{ and } \dot{b}_3(0) = 0 \end{aligned}$$

Time step ΔT for numerical integration was taken same as before, i.e., $\Delta T = 0.01$ second. The joint position results obtained from RoboAnalyzer are then shown in Figs. 9.18(a–f). Since the initial configuration of the Stanford arm given in

Table 9.4(a), the motion of Joint 3 should increase sharply after one second when Joint 2 turns more than 90° . This is evident from Figs. 9.18(b) and (c), respectively. One can visualize the same in the animation of RoboAnalyzer software.

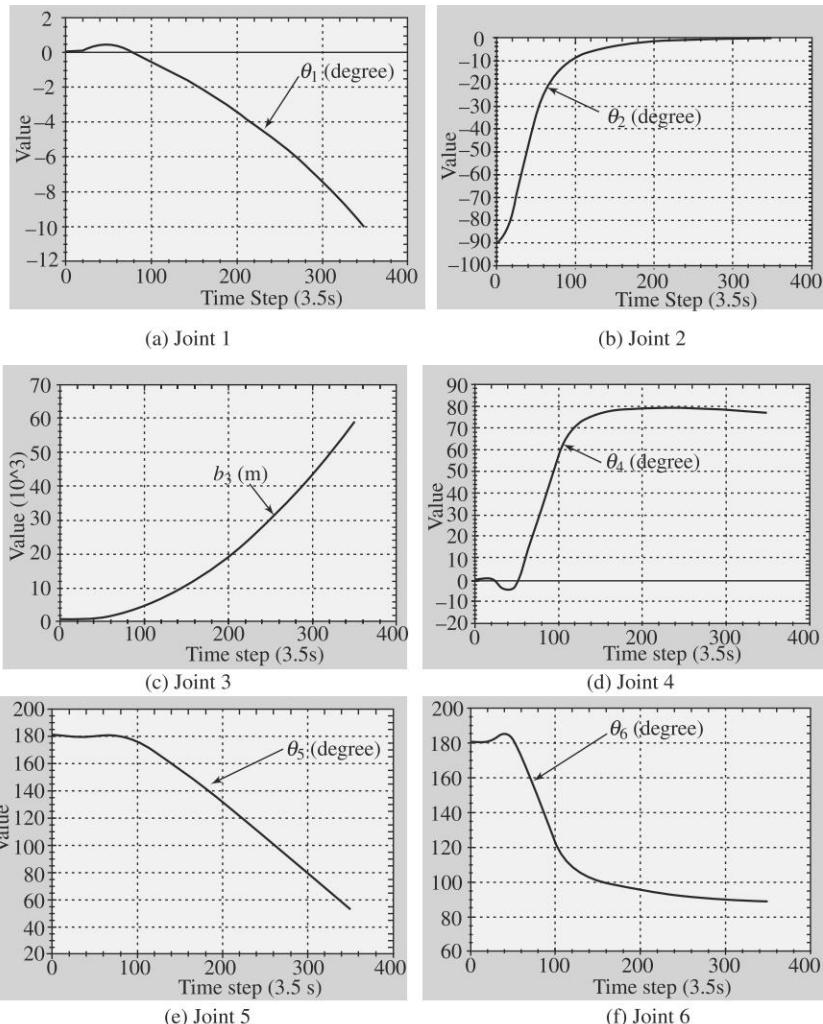


Fig. 9.18 Joint angles and displacement for the Stanford arm

Example 9.11 Simulation of the KUKA KR-5

The industrial robot considered for inverse dynamics analysis in Example 9.7 is considered here for free-fall simulation. The initial joint configuration is shown in Table 9.5. While variation of the joint angles are shown in Figs. 9.19(a-f), an intermediate configuration during the animation in RoboAnalyzer software is shown in Fig. 9.20. If one looks at the animation screenshot of Fig. 9.20 carefully, one can notice that the robot links interfere during the motion which is actually not

permissible. Hence, such simulation tool allows a motion planner to decide the range of motion be allowed during actual motion of the robot.

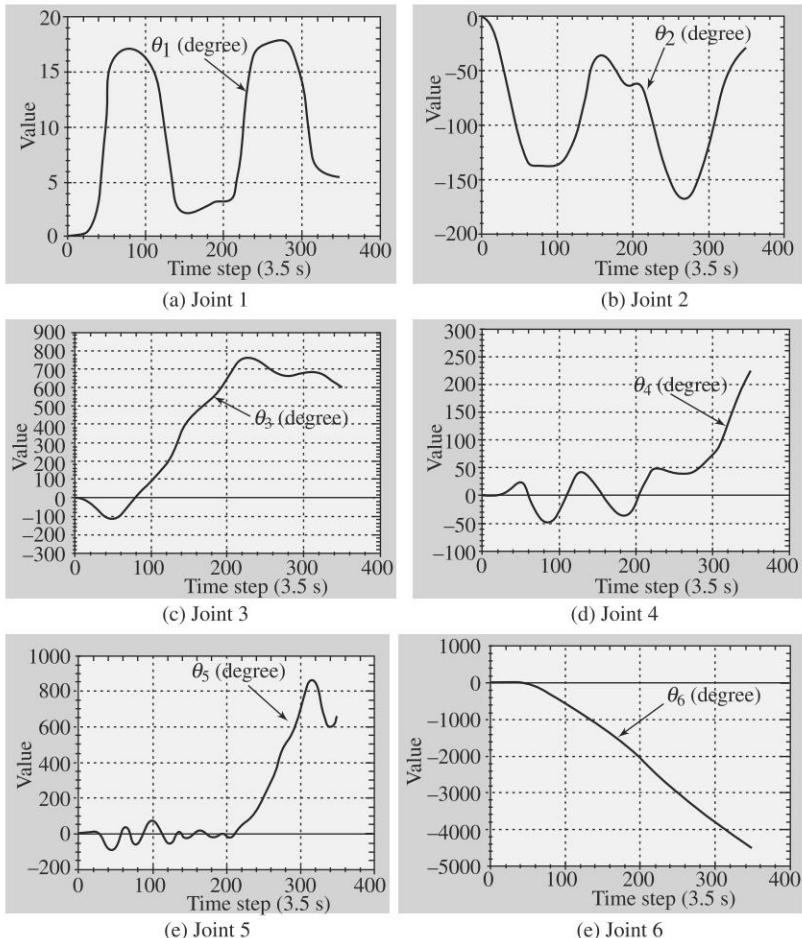


Fig. 9.19 Joint angles for KUKA KR-5 robot

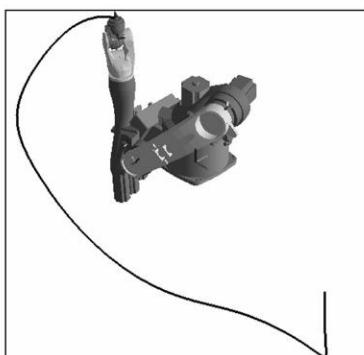


Fig. 9.20 RoboAnalyzer screenshot during free-fall simulation of KUKA KR-5

SUMMARY

In this chapter, dynamic modeling of serial robots using the Decoupled Natural Orthogonal Complements (DeNOC) is presented. Recursive inverse and forward dynamics algorithms are presented for robot control and simulation, respectively. Computational complexities of both the algorithms are reported with illustrative results for planar and spatial robotic systems using an in-house developed software called RoboAnalyzer.

EXERCISES

- 9.1** What is inverse dynamics?
- 9.2** Define forward dynamics and its difference in context with simulation.
- 9.3** Why the concept of the DeNOC matrices is preferable over other dynamic modeling approaches?
- 9.4** What is the meaning of *orthogonal* in DeNOC?
- 9.5** For forward dynamics using the DeNOC matrices, for how many links in a robot is the algorithm more efficient?
- 9.6** Find the expression of kinetic energy of one-link arm, Eq. (8.56a), using the DeNOC matrices.
- 9.7** Derive the equation of motion of one-link arm, Fig. 9.4, using the matrix and vector expressions of the NOC matrix \mathbf{N} appearing after Eq. (9.10a).
- 9.8** Write the expression of the total kinetic energy for an n -link robot manipulator using the definition of the generalized twist \mathbf{t} given in Eq. (9.3c).
- 9.9** Redo Exercise 9.7 using the definition of the generalized joint-rate $\dot{\boldsymbol{\theta}}$ of Eq. (9.3c).
- 9.10** What is \mathbf{UDU}^T and what does it perform?
- 9.11** Generate joint torque plots for the Example 9.4 using the following explicit expressions:

$$\ddot{\mathbf{I}\boldsymbol{\theta}} + \mathbf{C}\dot{\boldsymbol{\theta}} = \mathbf{\tau}^g + \boldsymbol{\tau}, \text{ where } \boldsymbol{\theta} \equiv [\theta_1 \quad \theta_2 \quad \theta_3]^T \quad (9.42a)$$

where the 3×3 GIM and MCI matrices \mathbf{I} and \mathbf{C} , respectively, and the 3-dimensional vector $\mathbf{\tau}^g$ due to gravity are given by

$$\begin{aligned} i_{11} &= \frac{1}{3}(m_1a_1^2 + m_2a_2^2 + m_3a_3^2) + m_2a_1^2 + m_3(a_1^2 + a_2^2) + (m_2 + 2m_3)a_1a_2c_2 \\ &\quad + m_3a_3(a_2c_3 + a_1c_{23}) \\ i_{12} = i_{21} &= \frac{1}{3}(m_2a_2^2 + m_3a_3^2) + m_3a_2^2 + \left(\frac{1}{2}m_2a_2 + m_3a_2\right)a_1c_2 \\ &\quad + \frac{1}{2}m_3a_3(2a_2c_3 + a_1c_{23}) \\ i_{13} = i_{31} &= \frac{1}{3}m_3a_3^2 + \frac{1}{2}m_3a_3(a_2c_3 + a_1c_{23}) \\ i_{22} &= \frac{1}{3}(m_2a_2^2 + m_3a_3^2) + m_3(a_2^2 + a_2a_3c_3) \\ i_{23} = i_{32} &= \frac{1}{3}m_3a_3^2 + \frac{1}{2}m_3a_2a_3c_3; \quad i_{33} = \frac{1}{3}m_3a_3^2 \end{aligned} \quad (9.42b)$$

$$\begin{aligned}
c_{11} &= -\frac{1}{2}[\{m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})\}\dot{\theta}_2 + m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_3] \\
c_{12} &= -\frac{1}{2}[\{m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})\}\dot{\theta}_{12} + m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_3] \\
c_{13} &= -\frac{1}{2}m_3(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_{123} \\
c_{21} &= \frac{1}{2}[\{m_2 a_1 a_2 s_2 + m_3(2a_1 a_2 s_2 + a_1 a_3 s_{23})\}\dot{\theta}_1 - m_3 a_2 a_3 s_3 \dot{\theta}_3] \\
c_{22} &= -\frac{1}{2}m_3 a_2 a_3 s_3 \dot{\theta}_3; \quad c_{23} = -\frac{1}{2}m_3 a_2 a_3 s_3 \dot{\theta}_{123} \\
c_{31} &= \frac{1}{2}m_3[(a_1 a_3 s_{23} + a_2 a_3 s_3)\dot{\theta}_1 + a_2 a_3 s_3 \dot{\theta}_2]; \quad c_{32} = \frac{1}{2}m_3 a_2 a_3 s_3 \dot{\theta}_{12} \\
c_{33} &= 0
\end{aligned} \tag{9.42c}$$

and

$$\begin{aligned}
\tau_1^g &= \frac{1}{2}g[-m_1 a_1 c_1 - 2m_2 \left(a_1 c_1 + \frac{1}{2}a_2 c_{12} \right) - 2m_3 \left(a_1 c_1 + a_2 c_{12} + \frac{1}{2}a_3 c_{123} \right)] \\
\tau_2^g &= \frac{1}{2}g \left[-m_2 a_2 c_{12} - 2m_3 \left(a_2 c_{12} + \frac{1}{2}a_3 c_{123} \right) \right] \\
\tau_3^g &= \frac{1}{2}g[-m_3 a_3 c_{123}]
\end{aligned} \tag{9.42d}$$

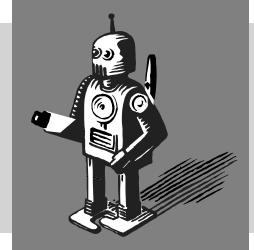
where $\theta_{12} \equiv \theta_1 + \theta_2$; $\theta_{123} \equiv \theta_{12} + \theta_3$; $\dot{\theta}_{12} \equiv \dot{\theta}_1 + \dot{\theta}_2$; $\dot{\theta}_{123} \equiv \dot{\theta}_{12} + \dot{\theta}_3$; and $s(\cdot) \equiv \sin(\cdot)$, $c(\cdot) \equiv \cos(\cdot)$. Find the joint actuator torques, τ , for plotting.

- 9.12** Verify the simulation results of Example 9.8 using the expressions of Eq. (9.42) and “ODE45” function of MATLAB.
- 9.13** What are the features of RoboAnalyzer (RA)?
- 9.14** Using the RA software find the joint torques for the two-DOF planar manipulator shown in Fig. 9.5.
- 9.15** Generate joint torques using RA for the three-DOF manipulator of Fig. 9.6 for the joint trajectory given by Eqs. (9.36a-c) with the following inputs:

$$\theta_i(0) = \dot{\theta}_i(0) = \ddot{\theta}_i(0) = 0; \theta_i(T) = \pi/2, \dot{\theta}_i(0) = \ddot{\theta}_i(0) = 0, \text{ for } i = 1, 2, 3 \tag{9.43}$$

10

Linear Control



Control of any system means to apply/exert moment(s) and force(s) on it so that the system moves/works according to the commanded instructions. In a robotic system also, a manipulator is required to carry out some specified tasks by moving its end-effector accurately and repeatedly.

In Chapter 6, equations were derived to obtain the time histories of all the joint motions for a desired end-effector motion. In this chapter, control laws will be developed, where the above joint histories can be input in order to make the robot track the commanded trajectory. In some cases, the Cartesian trajectory, i.e., position and orientation of the end-effector, is also controlled by calculating how much moment and force are required to achieve it and correspondingly converting them to joint torques, as any robot is basically moved or controlled by a set of joint motions only. In either case, it is assumed that the reader has the basic knowledge of control systems (Ogata, 1987; Kuo, 1991; Gopal, 1997).

Control of a robot requires the knowledge of the mathematical model of the robot, and some intelligence to act on it. Whereas the required mathematical model requires the physical laws governing the motion of the manipulator, intelligence needs sensory devices and the means to act and react against sensed variables. In robot control, individual joints are driven by actuators, be they electric, hydraulic or pneumatic, as covered in Chapter 3. The actuators apply force or torque to cause the links to move. Their commands are provided by the control system of the robot that will move the manipulator to achieve the specified task intended by the end-effector. These commands are based on the control set-points generated from the set of joint-torques versus time. Note that the joint histories, denoted in Fig. 10.1 as θ_d , $\dot{\theta}_d$, and $\ddot{\theta}_d$, i.e., the desired joint position, velocity and acceleration, respectively, are prepared by the trajectory generator or planner. It is basically an algorithm based on any of the schemes developed in Chapter 12. However, due to difficulties in accurate computations of the manipulator parameters, e.g., the link lengths, masses and inertias, etc., and nonconsideration of

Who controls a robot?

A personal computer or a dedicated electronics circuitry connected to the manipulator.

Robot, Manipulator, and Arm

Manipulator or arm is the mechanical part of a robot whose movement needs to be controlled. All three terms are used here interchangeably.

friction, backlash, etc., in the mathematical model, the end-effector of the robot will not follow the commanded trajectory exactly. Hence, the actual joint and/or end-effector positions and their derivatives can be fed back to the control system to find the deviations from the desired trajectory and accordingly to generate corrective force or torque for the joint actuators. The schematic diagram of a robot control is shown in Fig. 10.1, where θ and $\dot{\theta}$ represent the actual joint positions and velocities, respectively.

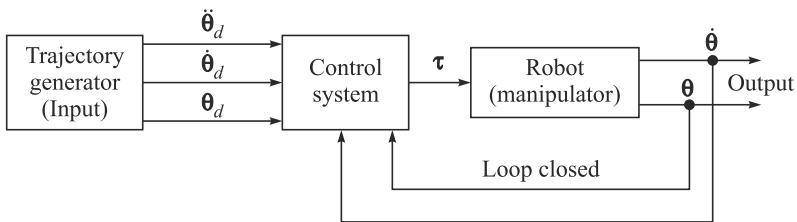


Fig. 10.1 Schematic diagram of a robot-control system

Control can be classified as *linear* and *nonlinear*. The use of linear control techniques in robotics is valid when a robot moves slowly and can be mathematically modeled by linear differential equations. This chapter deals with linear control of robots. For the case of an actual robot control, such linear methods are essentially viewed as approximate methods because dynamic equations of motion of a robot manipulator are actually nonlinear differential equations. However, the linear control theory forms the basis for more complex nonlinear control systems. For many practical applications, the use of linear control systems is often reasonable, as the robot manipulators used in major industries move relatively slowly. But if the joints move fast, the coupled dynamic forces are significant, and the nonlinearity cannot be ignored. In such situations, complex control algorithms are required, as explained in Chapter 11.

10.1 CONTROL TECHNIQUES

Several techniques can be employed to resolve a control problem of a robot. The technique followed and the way it is implemented may have a significant influence on the robot performance. For instance, the need for continuous motion differs from that allowing a point-to-point control where only reaching to the final position is of concern. On the other hand, the mechanical design of a robot has an influence on the kind of control scheme utilized. For instance, the control problem of a Cartesian robot is substantially different from that of an anthropomorphic articulated robot, as in the latter case, transformation of the end-effector motions need to be converted into the joint motions. The driving system of the joints has also an effect on the type of control strategy to be used. If a manipulator is actuated by electric motors with reduction gears of high ratios, the presence of gears tends to linearize the system dynamics. The effective mass and inertia terms of the links are greatly reduced, typically, by the square of the gear ratios as defined after Eq. (10.50b). Hence, the joint dynamics are decoupled, and each joint can be controlled independently. The presence of gears, however, introduces friction, gear train compliance, and backlash.

As a result, the control designer has to pay more attention to these affects rather than to the nonlinear inertia, Coriolis forces, etc., for the achievement of high performance. On the contrary, direct drives, as in the case of direct-drive robots, eliminate the drawbacks of gear train but introduce the nonlinear terms due to the couplings between the links and joints. As a result, different control strategies, e.g., computed-torque, etc., must be thought off to obtain high performances.

As indicated in Fig. 10.1, the manipulator control problem is a Multi-Input Multi-Output (MIMO) problem. To simplify the problem, each joint is considered to be independent and separately controlled. This is generally acceptable for high-gearred industrial robots as mentioned above.

Hence, each joint is a Single-Input Single-Output (SISO) system. The control designed for such a model is commonly termed *independent joint variable control*, and for an n -jointed manipulator, n independent SISO control systems need to be designed. Since each SISO system can be modeled as a linear second-order differential equation, its characteristics are studied in the next section, whereas different ways to control it are listed below:

- (i) **On-off or two-step control.**
- (ii) **Proportional (P) control:** It produces a control signal proportional to the error between the desired and actual motion, namely, the joint positions.
- (iii) **Derivative (D) control:** Here, a control signal is generated proportional to the rate of change of position error. Derivative control is known as an anticipatory control that measures the existing rate of change of error, anticipates incoming larger error, and applies correction before the larger error is arrived. Derivative control is never used alone.
- (iv) **Integral (I) control:** It produces a control signal proportional to the integral of errors accumulated over time. This type of controller can be considered as looking-back, summing all the errors and then respond.
- (v) **Combination modes:** In these modes, the above strategies such as proportional (P), derivative (D) and Integral (I) controls are combined, such as PD, or PI or PID.

Why SISO?

In highly geared robots, consideration of independent and separate control of each joint is equivalent to a SISO control problem.

10.2 DYNAMIC SYSTEMS

Behavior of a robot, or for that matter any system requiring control, is governed by its dynamical model, as presented in Chapters 8 and 9. For a given dynamic system, the behavior may not be as desired. Hence, a control action is required. In this section, a moving block shown in Fig. 10.2 is considered to show their natural behaviors and why a control action is required.

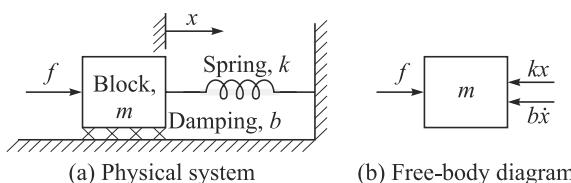


Fig. 10.2 A moving block

10.2.1 A Moving Block

Take the mechanical system of a moving block shown in Fig. 10.2. The block has mass m , which is attached to a spring of stiffness k , and subjected to damping friction of coefficient b . The block's zero position and the positive sense of x are indicated in Fig. 10.2(a). From the free-body diagram of the forces acting on the block, as shown in Fig. 10.2(b), the equation of motion of the block is obtained as

$$m\ddot{x} + b\dot{x} + kx = f \quad (10.1)$$

which is a second-order linear constant-coefficient differential equation. An alternative representation of Eq. (10.1) is given by

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2x = f' \quad (10.2a)$$

$$\text{where } f' = \frac{1}{m}f, \omega_n \equiv \sqrt{\frac{k}{m}} \text{ and } \zeta \equiv \frac{b}{2\sqrt{km}} \quad (10.2b)$$

The terms ω_n and ζ are called the *natural frequency* and *damping ratio* of the system at hand, respectively. The solution to the differential equation, Eq. (10.1) or Eq. (10.2), is a function of time, $x(t)$, which specifies the motion of the block. This solution will depend on the block's initial conditions, i.e., its initial position and velocity. From the study of differential equations (Wylie, 1975), if the homogeneous solution of x is assumed to be $x = e^{st}$ then the solution to Eq. (10.1) depends on the roots of its characteristic equation, namely,

$$ms^2 + bs + k = 0 \quad (10.3a)$$

which has the following roots:

$$s_1 = \frac{-b + \sqrt{b^2 - 4mk}}{2m} \text{ and } s_2 = \frac{-b - \sqrt{b^2 - 4mk}}{2m}. \quad (10.3b)$$

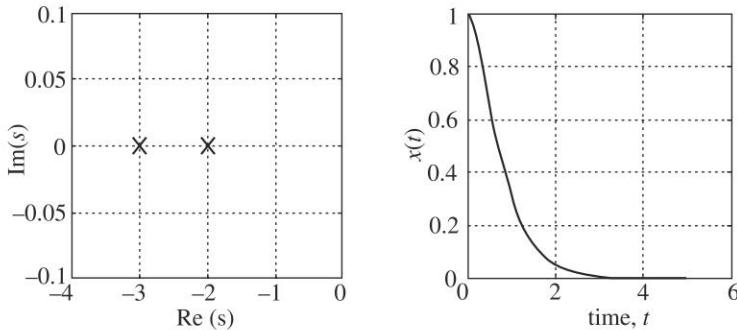
If s_1 and s_2 are real, the behavior of the system is sluggish and non-oscillatory. If s_1 and s_2 are complex, i.e., have an imaginary component, the behavior of the system is oscillatory. The limiting case between these two behaviors gives the third type of response, i.e., fastest non-oscillatory response. These cases are elaborated next.

Case I: Real and Unequal Roots This is the case when $b^2 > 4mk$, i.e., damping friction dominates, and sluggish behavior results. This response is called *overdamped*. The solution $x(t)$ has the following form:

$$x(t) = c_1e^{s_1t} + c_2e^{s_2t} \quad (10.4)$$

where s_1 and s_2 are given by Eq. (10.3b) which are also referred as the *poles* of the system at hand. The coefficients c_1 and c_2 are constants which can be determined for any given set of initial conditions, i.e., initial position and velocity of the block. Figure 10.3(a) shows an example of root or pole locations, and the corresponding time response to a nonzero initial condition is shown in Fig. 10.3(b). When the poles of a second-order system are real and unequal, the system exhibits sluggish or overdamped motion. In cases where one of the poles has a much greater magnitude than the other one, the pole of the larger magnitude can be neglected, as the term corresponding to it will decay to zero rapidly in comparison to the other more dominant pole. This same notion of dominance extends to higher order systems as well. For example, often a

third-order system can be studied as a second-order system by considering only two dominant poles.



(a) Pole locations

(b) Response

```
>> %Response of an overdamped system
>> poly = [1 5 6]; p = roots(poly);
>> subplot (1,2,1), plot(p,[0 0], 'X', 'MarkerSize', 15)
>> axis ([-4 0 -.1 .1]);
>> xlabel('Re (s)'), ylabel ('Im (s)'), grid on;
>> ti = 0:.1:5; c1 = -2; c2 = 3;
>> xt = c1*exp(p(1)*ti) + c2*exp(p(2)*ti);
>> subplot (1,2,2), plot(ti,xt)
>> xlabel('time, t'), ylabel ('x (t)'), grid on;
>>%MATLAB program ends
```

(c) MATLAB Program

Fig. 10.3 Root (poles) locations and system response of an overdamped system

Example 10.1 Motion of a Block with Real and Unequal Roots (Poles)

For the system in Fig. 10.2, if the system parameters are, $m = 1$, $b = 5$, and $k = 6$, the characteristic equation is given by

$$s^2 + 5s + 6 = 0 \quad (10.5)$$

which has roots (poles) at $s_1 = -2$ and $s_2 = -3$. Hence, the response has the following form

$$x(t) = c_1 e^{-2t} + c_2 e^{-3t} \quad (10.6)$$

If the block is initially at rest and released from the position of $x = 1$, i.e., $x(0) = 1$ and $\dot{x}(0) = 0$, the coefficients c_1 and c_2 can then be easily calculated from Eq. (10.6) that satisfy the initial conditions at $t = 0$, i.e.,

$$c_1 + c_2 = 1; \text{ and } -2c_1 - 3c_2 = 0 \quad (10.7)$$

Equation (10.7) yields $c_1 = 3$ and $c_2 = -2$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = 3e^{-2t} - 2e^{-3t} \quad (10.8)$$

The poles obtained from Eq. (10.5) and the response given by Eq. (10.8) are shown in Figs. 10.3(a-b), whereas a MATLAB program to generate them is shown in Fig. 10.3(c).

Case II: Complex Roots This is the case when $b^2 < 4mk$, i.e., stiffness dominates, and oscillatory behavior results. This response is called underdamped. For this case, the characteristic equation has complex roots of the form

$$s_1 = \lambda + \mu i; \text{ and } s_2 = \lambda - \mu i, \text{ where } \lambda = -\frac{b}{2m} \text{ and } \mu = \frac{\sqrt{4mk - b^2}}{2m} \quad (10.9)$$

Equation (10.9) has the solution of the form

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} \quad (10.10)$$

which is difficult to use directly since it involves imaginary numbers explicitly. It can be shown using Euler's formula that

$$e^{ix} = \cos x + i \sin x \quad (10.11)$$

Then the solution of Eq. (10.10) can be manipulated into the following form:

$$x(t) = e^{\lambda t} [\alpha_1 \cos(\mu t) + \alpha_2 \sin(\mu t)], \text{ where } \alpha_1 \equiv c_1 + c_2 \text{ and } \alpha_2 \equiv i(c_1 - c_2) \quad (10.12)$$

As before, the coefficients α_1 and α_2 are constants which can be determined for any given set of initial conditions, i.e., initial position and velocity of the block. Let the constants α_1 and α_2 be written as

$$\alpha_1 = r \cos \delta, \text{ and } \alpha_2 = r \sin \delta \quad (10.13)$$

Then, Eq. (10.12) can be rewritten in the following form:

$$x(t) = r e^{\lambda t} \cos(\mu t - \delta) \quad (10.14)$$

$$\text{where } r = \sqrt{\alpha_1^2 + \alpha_2^2}, \text{ and } \delta = \text{atan} 2(\alpha_2, \alpha_1) \quad (10.15)$$

It is now clear from Eq. (10.14) that the resulting motion is an oscillation whose amplitude is exponentially decreasing towards zero for negative λ . Figure 10.4 shows the pole locations and the corresponding time response to a nonzero initial condition. In case of no damping, i.e., $b = 0$, the behavior is completely oscillatory, as shown in Fig. 10.5, which is corresponding to the roots or poles $s_{1,2} = \pm \mu i$ of the characteristic equations given by Eq. (10.3a). Note from (10.9) that $\mu = \sqrt{k/m}$, which is nothing but the natural frequency of system under study.

Example 10.2 Motion of a Block with Complex Roots

If the parameters of the system shown in Fig. 10.2 are, $m = 1$, $b = 1$, and $k = 1$, the characteristic equation becomes

$$s^2 + s + 1 = 0 \quad (10.16)$$

whose roots are, $s_{1,2} = -\frac{1}{2} \pm \frac{\sqrt{3}}{2}i$. Hence, the response $x(t)$ from Eq. (10.12) has the following form:

$$x(t) = e^{-\frac{1}{2}t} \left(\alpha_1 \cos \frac{\sqrt{3}}{2}t + \alpha_2 \sin \frac{\sqrt{3}}{2}t \right) \quad (10.17)$$

For the block which is initially at rest and released from the position $x = 1$, the initial conditions are $x(0) = 1$, and $\dot{x}(0) = 0$. To determine the constants c_1 and c_2 , one has

$$\alpha_1 = 1, \text{ and } -\frac{1}{2}\alpha_1 + \frac{\sqrt{3}}{2}\alpha_2 = 0 \quad (10.18)$$

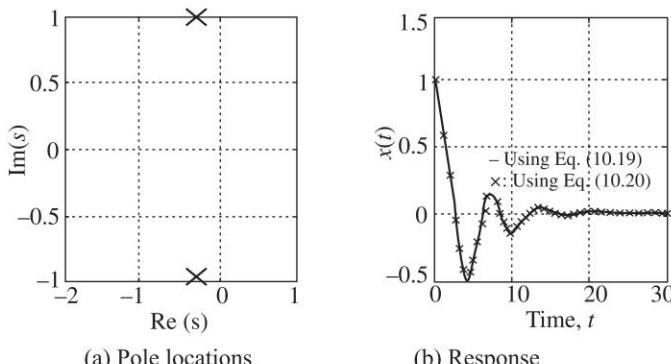
which give $\alpha_2 = 1/\sqrt{3}$. Hence, the motion of the system for $t \geq 0$ is given by

$$x(t) = e^{-\frac{1}{2}t} \left(\cos \frac{\sqrt{3}}{2}t + \frac{1}{\sqrt{3}} \sin \frac{\sqrt{3}}{2}t \right) \quad (10.19)$$

Alternately, the solution $x(t)$ in the form of Eq. (10.14) is

$$x(t) = \frac{2}{\sqrt{3}} e^{-\frac{1}{2}t} \cos \left(\frac{\sqrt{3}}{2}t - \frac{5\pi}{6} \right) \quad (10.20)$$

The pole placements from Eq. (10.16) and the responses from Eqs. (10.19) and (10.20) are shown in Figs. 10.4(a-b), which can also be generated using the MATLAB codes given in Fig. 10.4 (c). For $b = 0$ with the same initial conditions, the pole locations and the response are given in Fig. 10.5 in which the natural frequency from the plot of Fig. 10.5(b) can be visualized as approximately 1.5 cycles in 9.5 s, i.e., 0.15 cycles per second which in rad/s is $2\pi(0.15) \approx 1$ rad/s. The value is same as obtained from the expression of natural frequency, i.e., $\sqrt{k/m} = 1$ rad/s.



```
>>% Response of an underdamped system
>>poly = [1 .5 1]; p = roots(poly);pr = real(p);pim = imag(p);
>>subplot (1,2,1),plot(pr,pim,'x','MarkerSize', 15);
>>xlabel('Re (s)'),ylabel ('Im (s)'),grid on;
>>all = 1;a12 = 1/sqrt(3);ti = 0:.5:30;
>>% The symbol .* is to multiply two vectors term by term
>>xt1 = exp(pr(1)*ti).* (all*cos(pim(1)*ti) + a12*sin(pim(1)*ti));
>>xt2 = 2/sqrt(3)*exp(pr(1)*ti).*cos(pim(1)*ti-atan2(a12,all));
>>subplot (1,2,2), plot(ti,xt1,ti,xt2,'x');
>>xlabel('time, t'); ylabel ('x (t)'),grid on;
>>text(8,.3,'x: Using Eq.(10.20)');text(8,.5,'--: Using Eq.(10.19)');
>>% MATLAB program ends
```

(c) MATLAB Program

Fig. 10.4 Root (pole) locations and system response of an underdamped system

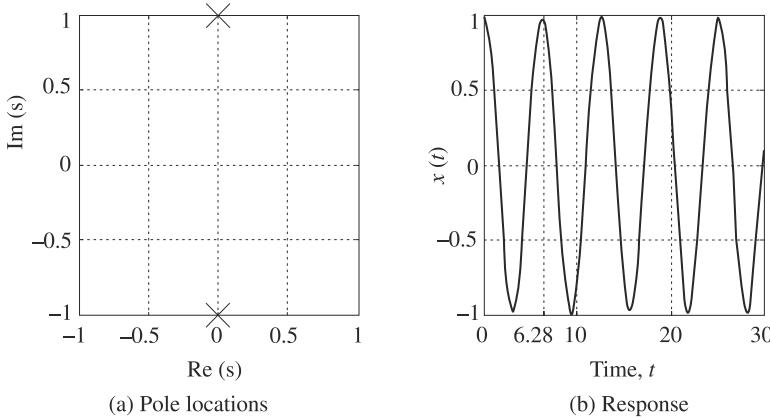


Fig. 10.5 Root (pole) locations and system response of an oscillatory system

Case III: Real and Equal Roots This is the special case when $b^2 = 4mk$, i.e., damping friction and stiffness are *balanced* yielding the fastest possible non-oscillatory response. This response is called *critically damped*. Using Eq. (10.3), it can be shown that it has real and equal repeated roots, i.e., $s_1 = s_2$. The solution has the following form:

$$x(t) = (c_1 + c_2 t)e^{st}, \text{ where } s_1 = s_2 = s = -\frac{b}{2m} \quad (10.21)$$

Equation (10.21) can be then rewritten as

$$x(t) = (c_1 + c_2 t) e^{-\frac{b}{2m}t} \quad (10.22)$$

Figure 10.6 shows the pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are real and equal, the system exhibits critically damped motion, i.e., the fastest possible non-oscillatory response. This third case of critical damping is generally a desirable situation since the system returns to its initial (nominal) position as rapidly as possible without any oscillatory behavior.

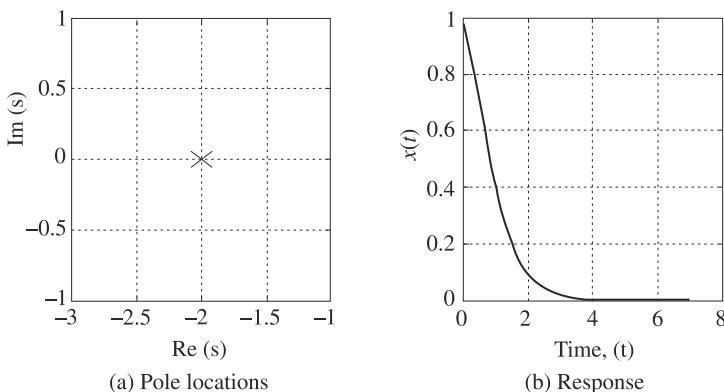


Fig 10.6 Root (pole) locations and system response of a critically damped system

Example 10.3 Motion of a Block with Equal and Real Roots

For the system in Fig. 10.2, the parameters are, $m = 1$, $b = 4$, and $k = 4$. The characteristic equation is

$$s^2 + 4s + 4 = 0 \quad (10.23)$$

whose roots are $s_1 = s_2 = s = -2$. Hence, the response has the following form:

$$x(t) = (c_1 + c_2 t)e^{-2t} \quad (10.24)$$

For the block initially at rest when released from the position $x = 1$, initial conditions are $x(0) = 1$, and $\dot{x}(0) = 0$. They will be used to determine c_1 and c_2 . To satisfy these conditions at $t = 0$ one has

$$c_1 = 1, \text{ and } -2c_1 + c_2 = 0 \quad (10.25)$$

Equation (10.25) is satisfied by $c_1 = 1$ and $c_2 = 2$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = (1 + 2t)e^{-2t} \quad (10.26)$$

Figures 10.6 (a) and (b) show the pole locations and plot of Eq. (10.26), respectively. The block returned to its initial steady-state position even though it was disturbed initially. In Examples 10.1–10.3, all the systems were stable.

For any passive physical system like the block of Fig. 10.2, this will be the case for which following properties hold:

$$m > 0; b > 0; \text{ and } k > 0 \quad (10.27)$$

In case a given system is not stable, it may not be always possible to change the parameters of the physical system. Hence, the concept of feedback control has to be introduced where one should be able to effectively change the values of one or more of the coefficients of the characteristic equation so that the resulting system is stable. This is discussed in the next section.

10.2.2 Moving Block with External Force

In Section 10.2.1, the natural behavior of the moving block was studied, i.e., if the block was disturbed from its stable position due to some non-zero initial position, the system responses in Figs. 10.3, 10.4 and 10.6 showed how the block comes back to its stable position due to the system parameters. In case the system forcing function f of Eq. (10.1) is not zero, the responses can be obtained by adding the particular integral (Wylie, 1975) corresponding to f to the homogeneous solutions given by Eqs. (10.4), (10.12), and (10.22). For a step forcing input, i.e., $f = 1$, a constant value, say, c_p can be assumed to be the particular solution of Eq. (10.1), which when substituted in it will result in the value of c_p in terms of the input f . Accordingly, the general solution can be given as

$$\text{For real and unequal roots: } x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} + c_p \quad (10.28a)$$

$$\text{For complex roots: } x(t) = e^{\lambda t} [\alpha_1 \cos(\mu t) + \alpha_2 \sin(\mu t)] + c_p \quad (10.28b)$$

$$\text{For real and equal roots: } x(t) = (c_1 + c_2 t) e^{-\frac{b}{2m} t} + c_p \quad (10.28c)$$

With appropriate initial conditions particular solutions can be obtained, as illustrated in Example 10.4.

Example 10.4 Motion of a Block with External Force

For the moving block of Examples 10.1 to 10.3, if a step forcing input $f = 1$ is exerted on the block, the responses of the system can be obtained for the zero initial conditions, i.e., $x(t) = 0$ and $\dot{x}(t) = 0$, as

For real and unequal roots ($m = 1, b = 5, k = 6$):

$$x(t) = \frac{1}{6}[1 - (3e^{-2t} - 2e^{-3t})] \quad (10.29a)$$

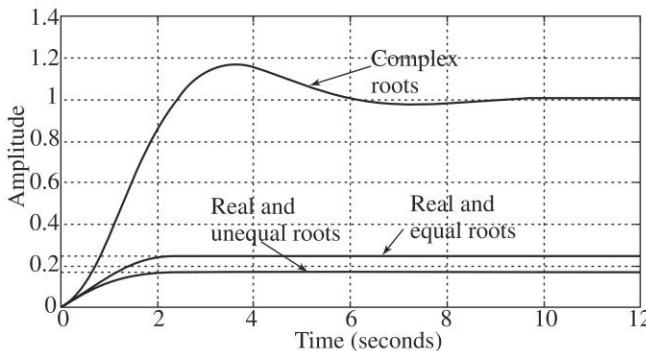
For complex roots ($m = 1, b = 1, k = 1$):

$$x(t) = 1 - e^{-\frac{1}{2}t} \left(\cos \frac{\sqrt{3}}{2}t + \frac{1}{\sqrt{3}} \sin \frac{\sqrt{3}}{2}t \right) \quad (10.29b)$$

For real and equal roots ($m = 1, b = 4, k = 4$):

$$x(t) = \frac{1}{4}[1 - (1 + 2t)e^{-2t}] \quad (10.29c)$$

The resulting behaviors are shown in Fig. 10.7(a), where at steady state the block reaches the position of unity force divided by the stiffness of the system at hand, as expected. Note that the MATLAB program to generate the plots of Fig. 10.7(a) is given in Fig. 10.7(b).



(a) Behavior with external step inputs

```
>>% Response of moving block with external force
>>num = [1]; den1 = [1 5 6]; den2 = [1 1 1]; den3 = [1 4 4];
>>sys1 = tf(num,den1); sys2 = tf(num,den2); sys3 = tf(num,den3);
>>step(sys1,sys2,sys3);grid on;
>>% MATLAB program ends
```

(b) MATLAB Program

Fig. 10.7 Behaviors of the moving block

10.3 TRANSFER FUNCTION AND STATE-SPACE REPRESENTATION

In this section, two popular representations of a dynamic system are introduced.

10.3.1 Transfer Function Representation

Note that the above behaviors can also be explained using the concept of Transfer Function (TF), which is widely used in the subject of control theory. For the damped

mass-spring system shown in Fig. 10.2, the open-loop TF $G(s)$ relating the output $x(s)$ with the input force $f(s)$ is given by

$$G(s) = \frac{x(s)}{f(s)} = \frac{1}{ms^2 + bs + k} \text{ or } G(s) = \frac{x(s)}{f(s)} = \frac{(1/m)\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (10.30a)$$

where $x(s)$ and $f(s)$ are the Laplace transforms of the time-varying parameters, namely, the displacement x and force f , respectively. They are indicated in Fig. 10.8. Moreover, ω_n and ζ are defined in Eq. (10.2b), which are called the *natural frequency* and *damping ratio* of the moving block, respectively. The two poles of the system are

$$s_{1,2} = \frac{-b \pm \sqrt{b^2 - 4mk}}{2m} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \quad (10.30b)$$

which determine the form of natural response of the system at hand, whereas the poles and zeros together affect the peaks and amplitudes of both the forced and natural responses. For a given forcing function $f(s)$, one can find the behavior of the block while initial conditions are both zeros.

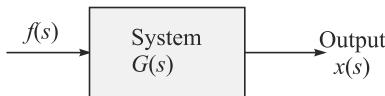


Fig. 10.8 Block diagram of the moving block

Laplace Transform and Role of TF

It changes differential equations to algebraic ones in order to simplify the solution strategies.

Hence, the zero initial conditions in Eqs. (10.29a-c) we intentional in order to be able to verify the results using the in-built ‘step’ command of MATLAB or using Simulink module of MATLAB which finds the responses of the block using its TF.

Example 10.5 Motion of the Block using MATLAB and Simulink

The responses of the moving block with unit external force, $f = 1$ with zero initial conditions, i.e., $x(t) = 0$ and $\dot{x}(t) = 0$, as shown in Fig. 10.7 can also be reproduced with the Simulink model (a tool in MATLAB) shown in Fig. 10.9. Note that the Simulink models are very useful for the study of a controller. The results are same as shown in Fig. 10.7.

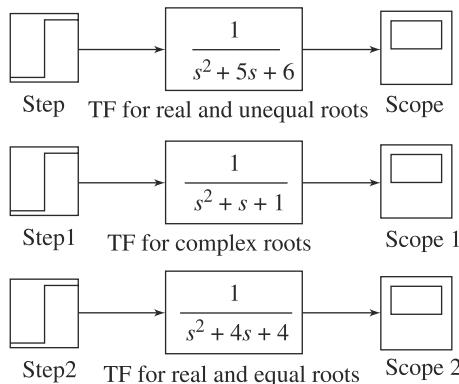


Fig 10.9 Simulink blocks for the moving block

10.3.2 State-Space Representation

In many instances, representation of a system dynamics in time domain or in the form of transfer function is not sufficient for its control. Hence, a more advanced way to represent the dynamics of a system, called *state-space representation*, is introduced here. In this, a higher-order differential equation, say, p^{th} order, is represented as p first-order differential equations. A dynamical system is typically represented using a set of n second-order differential equations. Hence, its state-space is represented using $2n$ first-order differential equations. Besides many control flexibilities, the representation is quite useful for numerical solution of differential equations using the first-order equation solvers which are well developed. In MATLAB, “ODE” series of solvers require inputs in the first-order or state-space form. If the system is linear, the set of $2n$ differential equations representing the dynamics of multiple-input multiple-output (MIMO) system can be represented as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \text{ and } \mathbf{y} = \mathbf{Cx} + \mathbf{Du} \quad (10.31)$$

where \mathbf{A} is the $2n \times 2n$ state matrix, \mathbf{B} is the $2n \times r$ input matrix, \mathbf{C} is the $m \times 2n$ output matrix, and \mathbf{D} is the $m \times r$ direct transmission matrix. For a nonlinear system, one can linearize it about an operating state and can obtain the resulting equation in the form of Eq. (10.31).

Note, for the single-input and single-output (SISO) system, one can derive the transfer function (TF) from the state-space representation, e.g., Eq. (10.31), where vectors \mathbf{y} and \mathbf{u} are scalars denoted by y and u , respectively. Correspondingly, matrices \mathbf{B} and \mathbf{C} are vectors of appropriate dimensions, and \mathbf{D} is a scalar. For that, the Laplace transform of Eq. (10.31) with zero initial conditions is expressed as

$$s\mathbf{x}(s) = \mathbf{Ax}(s) + \mathbf{Bu}(s) \quad (10.32a)$$

$$y(s) = \mathbf{c}^T \mathbf{x}(s) + du(s) \quad (10.32b)$$

Solving for $\mathbf{x}(s)$ from Eq. (10.32a) and substituting it in Eq. (10.32b) yields the relationship between the output $y(s)$ to the input $u(s)$ as

$$y(s) = [\mathbf{c}^T (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b} + d]u(s) \quad (10.33a)$$

Hence, the TF defined by $G(s)$ is extracted from Eq. (10.33a) as

$$G(s) = \frac{y(s)}{u(s)} = \mathbf{c}^T (s\mathbf{I} - \mathbf{A})^{-1} \mathbf{b} + d \quad (10.33b)$$

Note that the right side of Eq. (10.33b) involves $(s\mathbf{I} - \mathbf{A})^{-1}$. Hence, $G(s)$ can be written as

$$G(s) = \frac{q(s)}{|s\mathbf{I} - \mathbf{A}|} \quad (10.33c)$$

where $q(s)$ is a polynomial in s , and $|s\mathbf{I} - \mathbf{A}|$ represents the determinant of the matrix $(s\mathbf{I} - \mathbf{A})$ represented as $|s\mathbf{I} - \mathbf{A}|$. The denominator is equal to the characteristic polynomial of $G(s)$. In other words, the eigenvalues of \mathbf{A} are identical to the poles of

State-space

A state-space of a dynamical system is formed as a plot of its control variables and its derivatives plotted against the axes representing the above variables and derivatives.

$G(s)$. On the other hand, given a transfer function $G(s)$, one can also obtain the state-space representation by taking an inverse Laplace transformation of the expression for the output relating the TF and input, as illustrated in Example 10.6.

Note here that since more variables, e.g., position and velocity represented by the state-vector \mathbf{x} , are available to modify, many otherwise poorly performed dynamical systems represented by the classical approach of transfer function can be improved.

Example 10.6 State-space Representation of the Moving Block

Consider the moving block of Fig. 10.2 whose only dynamic equation of motion is given by Eq. (10.1). Let us define the following:

$$x_1 = x; \text{ and } x_2 = \dot{x} \quad (10.34a)$$

Then, Eq. (10.1a) can be expressed as

$$\dot{x}_1 = x_2; \text{ and } \dot{x}_2 = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}f \quad (10.34b)$$

where the output y can be given by

$$y = x_1 \quad (10.34c)$$

The above equations, Eqs. (10.34b-c), are now put in a matrix-vector form as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{bu}; \text{ and } y = \mathbf{c}^T \mathbf{x} \quad (10.35a)$$

where

$$\mathbf{x} \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \quad \mathbf{A} \equiv \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}; \quad \mathbf{b} \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad u \equiv f; \quad \mathbf{c} \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad d = 0 \quad (10.35b)$$

In Eqs. (10.35a-b), the 2-dimensional vector \mathbf{x} is the state vector, the 2×2 matrix \mathbf{A} is the state matrix, whereas the 2-dimensional vectors \mathbf{b} and \mathbf{c} are the input and output matrices, respectively. Moreover, the output y and direct transmission term d are scalars. Equations (10.35a-b) are the state-space representation of the moving block governed by Eq. (10.1).

Example 10.7 Transfer Function from State-space Representation

Consider the motion of the block in Fig. 10.2 again for which the state-space representation is given by Eqs. (10.35a-b). To find the TF, the matrix $s\mathbf{1} - \mathbf{A}$ is first evaluated as

$$s\mathbf{1} - \mathbf{A} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} = \begin{bmatrix} s & -1 \\ \frac{k}{m} & s + \frac{b}{m} \end{bmatrix} \quad (10.36a)$$

It is now easy to find the inverse of the 2×2 matrix in Eq. (10.36a) as

$$(s\mathbf{1} - \mathbf{A})^{-1} = \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}} \begin{bmatrix} s + \frac{b}{m} & 1 \\ -\frac{k}{m} & s \end{bmatrix} \quad (10.36b)$$

Finally, Eqs. (10.33b) and (10.35b) yield the following:

$$G(s) = [1 \ 0] \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}} \begin{bmatrix} s + \frac{b}{m} & 1 \\ -\frac{k}{m} & s \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} + 0 = \frac{1}{ms^2 + bs + k} \quad (10.36c)$$

Example 10.8 State-space Representations using MATLAB

A simple MATLAB program given in Fig. 10.10(a) can reproduce the plots of Fig. 10.7 corresponding to Example 10.1. One can change the values of m , b , and k to generate more plots.

```
% Response using state-space representation
>> t = 0:.05:10; m = 1; b = 5; k = 6;
>> A = [0 1; -k/m -b/m]; bv = [0; 1/m]; cv = [1 0];
>> sys = ss(A,bv,cv,0); step(sys,t);
```

(a) Response using state-space representation

```
>> m = 1; b = 5; k = 6; u = 1;
>> A = [0 1; -k/m -b/m]; bv = [0; 1/m]; cv = [1 0]; d = 0;
>> [num,den] = ss2tf(A,bv,cv,d,u)
```

(b) Conversion from state-space to transfer function

Fig.10.10 MATLAB programs with state-space representations

If one wants to convert the state-space representation to TF in MATLAB, the commands in Fig. 10.10(b) can be used to produce the results. The outputs are the coefficients of the numerator and denominator polynomials of the TF, namely,

num = 0 0 1.0000

den = 1 5 6

Example 10.9 State-space Representation from a Given Transfer Function

Consider the transfer function of the one-DOF robotic joint, which is derived in Section 10.4,

$$G(s) = \frac{\theta_m(s)}{v_a(s)} = \frac{K}{s(Ts + 1)} \quad (10.37a)$$

where K is gain and T is the time constant of the system. From Eq. (10.37a), one can write the following:

$$s^2 T \theta_m(s) + s \theta_m(s) = K v_a(s) \quad (10.37b)$$

Taking the inverse Laplace transform of Eq. (10.37b), the equation in time-domain is obtained as

$$T \ddot{\theta}_m + \dot{\theta}_m = K v_a \quad (10.37c)$$

Since Eq. (10.37c) is a second-order differential equation, two state variables are required, namely, x_1 and x_2 , which are defined as

$$x_1 = \theta_m, \text{ and } x_2 = \dot{\theta}_m \quad (10.38a)$$

This yields the state-space representation of the single-joint dynamics as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{bu}, \text{ and } y = \mathbf{c}^T \mathbf{x} \quad (10.38b)$$

where the 2-dimensional vectors \mathbf{x} , \mathbf{b} , \mathbf{c} , and the 2×2 matrix \mathbf{A} are as follows:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 0 \\ \frac{K}{T} \end{bmatrix}; \quad \mathbf{u} = v_a; \quad \mathbf{c} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (10.38c)$$

Example 10.10 State-space of a Single-DOF Planar Arm

Consider the single-DOF planar arm shown in Fig. 10.11. The torque τ and gravity g that is acting downward are also shown in the figure. Note that without any external torque τ , i.e., $\tau = 0$, the system is nothing but a compound or solid pendulum. The equation of motion for the single-DOF arm, as derived in Chapters 8 and 9, is given by

$$\frac{1}{3}ma^2\ddot{\theta} = \tau - \frac{1}{2}mga \sin \theta, \text{ or } \ddot{\theta} + \frac{3g}{2a} \sin \theta = \frac{3\tau}{ma^2} \quad (10.39)$$

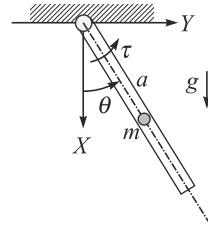


Fig. 10.11 One-DOF planar arm

Equation (10.39) is a nonlinear differential equation due to the appearance of ‘sine’ function. Since this is a second-order system, two state variables are needed, namely,

$$x_1 = \theta, \text{ and } x_2 = \dot{\theta} \quad (10.40)$$

Then, one obtains the following:

$$\dot{x}_1 = x_2; \text{ and } \dot{x}_2 = \frac{3\tau}{ma^2} - \frac{3g}{2a} \sin x_1 \quad (10.41)$$

Since there is no input torque τ , i.e., $u = 0$, output y is given by

$$y = \theta = x_1 \quad (10.42)$$

Note here that for nonlinear differential equations, it is difficult to express them in state-space form given by Eq. (10.31). For the system shown in Fig. 10.11, Eq.(10.31) can have the following form:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \quad \mathbf{Ax} = \begin{bmatrix} 0 & 1 \\ -\frac{3g}{2a} \sin x_1 & 0 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{3}{ma^2} \end{bmatrix}; \quad \mathbf{u} = \tau; \quad \mathbf{C} = [1 \ 0]; \quad \mathbf{D} = 0 \quad (10.43a)$$

In Eq. (10.43a), if the angle θ is small then the system can be linearized. For small angle θ , it is known that $\sin \theta \approx x_1$. Hence, the state-space representation of the linearized model in the form of Eq. (10.31) will have explicit expression for the state matrix \mathbf{A} , i.e.,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{3g}{2a} & 0 \end{bmatrix} \quad (10.43b)$$

10.4 A ROBOTIC JOINT

In this section, a mathematical model for a robotic joint will be developed. For this, first the model of the direct current (dc) motor shown in Fig. 3.9 which is commonly used in industrial robots will be developed, followed by the overall joint model. Since a typical electrically driven robotic joint has a motor equipped with a set of gears or belt-pulley or sprocket-chain arrangement to apply torque or force to the robot link, they will also be taken into account in the model.

10.4.1 Dynamics of a dc Motor

A dc motor shown in Fig. 3.9 works on the principle of current-carrying conductor in a magnetic field experiences a force. If the stator of the motor produces a radial magnetic flux and the current in the rotor (also called the armature) is i_a then there will be a torque on the rotor causing it to rotate. The magnitude of this torque is

$$\tau_m = k_m i_a \quad (10.44)$$

where τ_m and k_m are the motor torque and motor constant, respectively. The motor constant can be determined from a set of torque-speed curves shown in Fig. 3.10 for different values of the applied voltages. Equation (10.44) implies that the torque on the rotor is controlled by controlling the armature current. Note that when a motor rotates, it also acts like a generator, and a voltage develops across its armature. This voltage is proportional to the velocity of the conductor in the field, and is called the back *electromotive force* (emf). The back emf denoted as v_b opposes the current flow in the conductor which can be expressed as

$$v_b = k_e \dot{\theta}_m \quad (10.45)$$

where k_e is the back emf constant, and $\dot{\theta}_m$ denotes the angular velocity of the motor's rotor. Moreover, considering the circuit diagram for an armature controlled dc motor shown in Fig. 10.12, the differential equation for the circuit is obtained as

$$l_a \frac{di_a}{dt} + r_a i_a = v_a - k_e \dot{\theta}_m \quad (10.46)$$

where v_a , l_a , and v_b are the voltage, the inductance of the armature windings, and the generated back emf, respectively. In a robot control, it is desirable to control the torque generated by the motor rather than the velocity with electronic motor driver circuitry. These drive circuits sense the current through the armature and continuously adjust the voltage source v_a so that a desired current i_a flows through the armature. Such a circuit is called *current amplifier motor driver*. In these current drive systems, the rate at which the armature current change is limited by the motor inductance l_a and an upper limit on the voltage capability of the voltage source v_a .

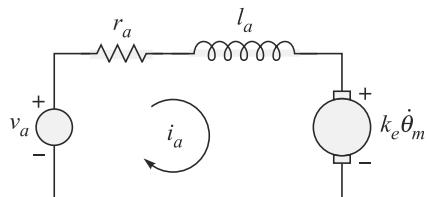


Fig. 10.12 The armature circuit of a dc motor

10.4.2 Actuator Dynamics

In this section, the differential equation and transfer function model treating each link of the robot manipulator as an independent single-input-single-output (SISO) system is derived. As mentioned in Chapter 1, actuator means an electric dc motor coupled with a gear train. In a way, a driving motor coupled with a transmission system means an actuator.

Why gears in dc motors?

Gears are needed for the speed reduction of a typical dc motor used in robots which are efficient only at high speed of 2000 rpm or more.

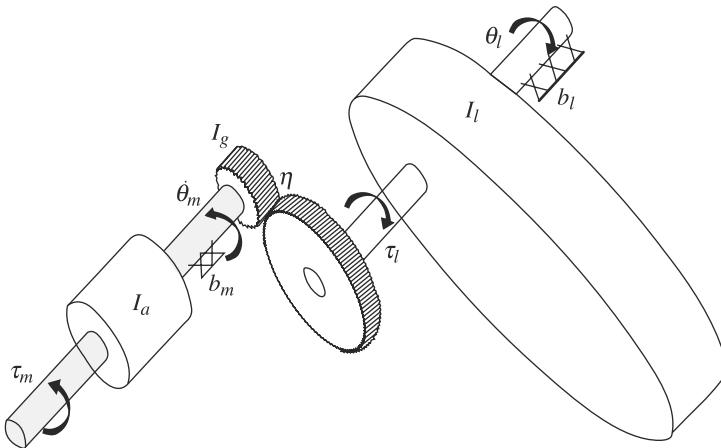


Fig. 10.13 A link connected to a dc motor through gear train

Figure 10.13 shows a link with inertia I_l connected to a dc motor through a pair of gears. The gear ratio is $1:\eta$, i.e., it is a reduction gear train. Typical gear reduction ratio η in robotic applications is in the range of 20 to 200 or more. Now, combine the inertias of the actuator (I_a) and gears (I_g) as $I_m = I_a + I_g$, where I_m is referred here as the rotor inertia. In terms of the motor rotation angle θ_m the equation of motion of the actuator (motor with gear train)-link system is then given by

$$I_m \ddot{\theta}_m + b_m \dot{\theta}_m = \tau_m - \frac{\tau_l}{\eta} = k_m i_a - \frac{\tau_l}{\eta} \quad (10.47)$$

Typical speed of a robot joint

It is around 60 rpm.

where the generated torque applied to the rotor is τ_m and the load torque is τ_l . The expression for τ_m given by Eq. (10.44) is a function of the current i_a flowing in the armature circuit and used in Eq. (10.47). Moreover, b_m is the coefficient of viscous friction that includes frictions in the brushes and gears. Further, the gear reduction ratio (η) causes an increase in the torque seen at the load and a reduction in the speed of the load, which yields

$$\tau_l = \eta \tau_m, \text{ and } \dot{\theta}_l = \frac{\dot{\theta}_m}{\eta} \quad (10.48)$$

where $\eta \gg 1$. Expressing the load torque τ_l in terms of the load parameters, namely, θ_l and b_l , i.e., $\tau_l = I_l \dot{\theta}_l + b_l \dot{\theta}_l$ — b_l being the viscous friction in the load bearings—, Eq. (10.47) is rewritten as

$$I_m \ddot{\theta}_m + b_m \dot{\theta}_m + \frac{1}{\eta} (I_l \ddot{\theta}_l + b_l \dot{\theta}_l) = \tau_m = k_m i_a \quad (10.49)$$

Using the relation of the motor and joint motion given by Eq. (10.48), one can write Eq. (10.49) in terms of the motor variables, θ_m and τ_m , as

$$\left(I_m + \frac{I_l}{\eta^2} \right) \ddot{\theta}_m + \left(b_m + \frac{b_l}{\eta^2} \right) \dot{\theta}_m = \tau_m \quad (10.50a)$$

whereas in terms of the load variables, i.e., θ_l and τ_l , the same can be expressed as

$$(I_l + \eta^2 I_m) \ddot{\theta}_l + (b_l + \eta^2 b_m) \dot{\theta}_l = \tau_l \quad (10.50b)$$

The term $I_l + \eta^2 I_m$ is called the *effective inertia* seen at the output link side of the gearing. Alternatively, the term $I_m + I_l/\eta^2$ is also termed in the literature as *effective inertia* seen by the motor shaft. Likewise, the terms $b_l + \eta^2 b_m$ and $b_m + b_m/\eta^2$ are referred as *effective damping* at the load and motor sides, respectively. Note that in a highly geared joint, i.e., $\eta \gg 1$, the inertia of the motor rotor can be a significant portion of the combined effective inertia. It is this effect that allows

Joint velocity and rate

Velocity is a vector quantity which has magnitude and direction, whereas rate is a scalar quantity. For a joint, its axis is the direction of its motion, which is known. Hence, joint velocity and joint rate are used in this book synonymously.

one to make the assumption that the effective inertia is a constant. It is known from dynamics that the link or load inertia I_l of a joint of the mechanism actually varies with configuration and load. However, in highly geared robots the variations represent a smaller percentage than they would be in a direct drive manipulator when $\eta = 1$. In order to find the transfer function (TF) representation of the joint with the dc motor, gears, and the link, Eqs. (10.46–10.47) are written in the Laplace domain as

$$(l_a s + r_a) i_a(s) = v_a(s) - k_e s \theta_m(s) \quad (10.51a)$$

$$(I_m s^2 + b_m s) \theta_m(s) = k_m i_a(s) - \frac{\tau_l(s)}{\eta} \quad (10.51b)$$

where the variables with (s) represent the Laplace transforms of the corresponding time domain functions. The block diagram of the joint with motor and gear train system is shown in Fig. 10.14. The TF from $v_a(s)$ to $\theta_m(s)$ is given when $\tau_l(s) = 0$ as

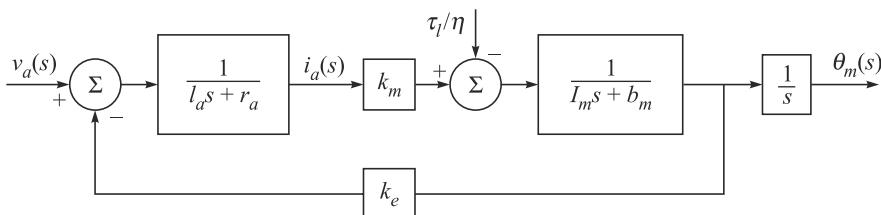


Fig. 10.14 Block diagram of a dc motor with gear train

$$\frac{\theta_m(s)}{v_a(s)} = \frac{k_m}{s[(l_a s + r_a)(I_m s + b_m) + k_e k_m]} \quad (10.52a)$$

whereas the TF from the load torque $\tau_l(s)$ to $\theta_m(s)$ is given when $v_a(s) = 0$, as

$$\frac{\theta_m(s)}{\tau_l(s)} = \frac{-(l_a s + r_a)/\eta}{s[(l_a s + r_a)(I_m s + b_m) + k_e k_m]} \quad (10.52b)$$

Notice that the magnitude of this latter transfer function, i.e., the effect of the load torque on the motor angle, is reduced by the gear reduction ratio η . For many electromechanical systems, e.g., the robots used in the industries, it is generally assumed that the electrical time constant given by l_a/r_a is much smaller than the mechanical time constant I_m/b_m . Hence, the order of the actuator dynamic model can be reduced by order one. This is done in a following manner. Divide the numerator and denominator of Eq. (10.52) by r_a and neglect the electrical time constant by setting it to zero. The above TFs are then rewritten as

$$\frac{\theta_m(s)}{v_a(s)} = \frac{k_m/r_a}{s(Is+b)}, \text{ and } \frac{\theta_m(s)}{\tau_l(s)} = \frac{-1/\eta}{s(Is+b)} \quad (10.53a)$$

where

$$I \equiv I_m, b \equiv b_m + \frac{k_e k_m}{r_a} \quad (10.53b)$$

In case one is interested with the angular speed of the motor shaft, i.e., $\dot{\theta}$, then the corresponding transfers from Eqs. (10.53a-b) can be rewritten as

$$\frac{\dot{\theta}_m(s)}{v_a(s)} = \frac{k_m/r_a}{Is+b} \text{ and } \frac{\dot{\theta}_m(s)}{\tau_l(s)} = \frac{-1/\eta}{Is+b} \quad (10.53c)$$

Where the term ‘ s ’ in the denominators of the right hand sides does not show up. In Eqs. (10.53a-c), the initial conditions are $\theta_m = 0$ and $\dot{\theta}_m = 0$, as required by the definition of the TF. Besides, the time-constant of the closed-loop system is I/b dictates the speed of the transient response before attaining a steady-state value.

Note the assumption that $\tau_l(s) = 0$ and $v_a(s) = 0$ while obtaining the open-loop TFs of Eqs. (10.52a-b), respectively. Hence, the total response of the reduced-order system in time domain is obtained by the superposition of the two TFs given by Eq. (10.53a). This is possible due to the linearity of the resultant model of the robotic joint at hand. Inverse Laplace transformation of the resulting expression is then performed, which by superposition yields the following time-domain expression:

$$I\ddot{\theta}_m + b\dot{\theta}_m = u_a - \tau_d \quad (10.54a)$$

where $u_a \equiv \frac{k_e}{r_a} v_a$, and $\tau_d \equiv \frac{\tau_l}{\eta}$ (10.54b)

In Eqs. (10.54a-b), b represents the effective damping, u_a is the control input, and τ_d represents the disturbance input. The corresponding block diagram is shown in Fig. 10.15.

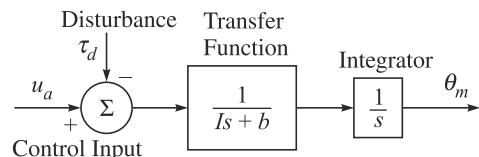


Fig. 10.15 A simplified dc motor

Example 10.11 Effective Inertia

If the link inertia varies from 2 to 6 kg-m², the rotor inertia is 0.01 kg-m², and the gear ratio is 1:30, the minimum and maximum effective inertias are calculated as

$$I_{\min} = I_l(\min) + \eta^2 I_m = 2 + (900)(0.01) = 11, \text{ where } \eta = 30$$

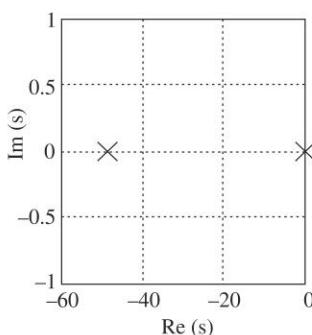
$$I_{\max} = I_l(\max) + \eta^2 I_m = 6 + (900)(0.01) = 15$$

Note that the percentage variation of the effective inertia is only about 27% compared to the original variation of 67%. This is due to the presence of the high-reduction gear ratio.

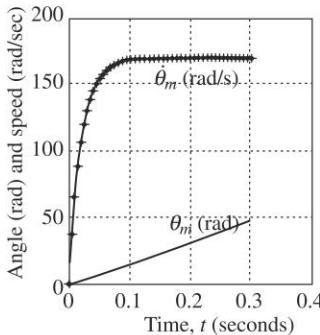
Example 10.12 Joint Response

Response of the characteristic equation $Is^2 + bs$ of a typical dc motor based on Eq. (10.53a) can be shown in Fig. 10.16 for which the numerical values are calculated from Maxon Motor's datasheet (Model: A-max 22-110017; 5 watt; 6 volts) (WR: Maxon) as

$$I \equiv I_m = 4.07 \times 10^{-7} \text{ kg-m}^2; b \equiv b_m + \frac{k_e k_m}{r_a} = 0.00002 \text{ Nm/rad/s}^2 \quad (10.55)$$



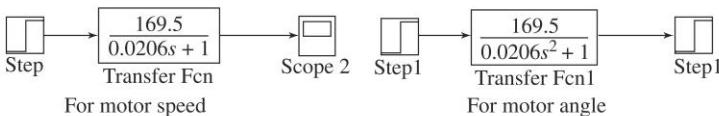
(a) Pole locations



(b) Response

```
>>% Response of a dc Motor
>>t = 0:.005:.3; ten7 = 10000000;
>>i_m = 4.07/ten7; b_m = 12/ten7; km = 5.9/1000;
ke = km; la = 0.106/1000; ra = 1.76;
>>i_eff = i_m; tau_mech = 20/1000; b_m = i_m/10*tau_mech;
b_eff = b_m + ke*km/ra
>>num = [km/ra]; den_w = [i_eff b_eff]; den = [i_eff b_eff 0];
>>sys_w = tf(num,den_w); sys = tf(num,den);
step(sys_w, sys_t); grid on;
>>% MATLAB program ends
```

(c) MATLAB program for a dc motor



(d) Simulink model of a dc motor

Fig. 10.16 Root (pole) locations and system response of a dc motor

In Eq. (10.55), the numerical value of k_m considered equal to k_e (Gopal, 1997), i.e., $k_m = 0.0059 \text{ Nm/A}$, and $k_e = 0.0059 \text{ V/rad/s}$. The value of r_a from the manufacturer's datasheet is 1.76 ohms, whereas b_m can be estimated from the mechanical time constant $\tau_m = 0.02 \text{ s}$ as $b_m = I_m/\tau_m = 814.0 \times 10^{-7} \text{ Nm-s}$. The initial values for θ_m and $\dot{\theta}_m$ are taken as zeros. Figure 10.16(a) then shows the pole locations corresponding to the characteristic equation $I\ddot{s}^2 + bs = 0$, whereas Fig. 10.16(b) shows the responses of θ_m and $\dot{\theta}_m$.

Figures 10.16 (c-d) respectively show the MATLAB program and Simulink model that can generate the response of the joint shown in Fig. 10.16(b). The responses can also be obtained as the solution of the time-domain equation given by Eq. (10.54a), where $r_a = 1$ (step input), $\tau_d = 0$ (no disturbance), and $\theta_m = 0$, $\dot{\theta}_m = 0$ at $t = 0$. The solution for θ_m consisting of homogeneous and particular integral is given by

$$\theta_m(t) = 169.5[t + 0.0206(e^{-48.6t} - 1)] \quad (10.56a)$$

The time-derivative of Eq. (10.56a) provides the relation for the angular speed $\dot{\theta}_m$ as

$$\dot{\theta}_m(t) = 169.5(1 - e^{-48.6t}) \quad (10.56b)$$

10.5 PERFORMANCE AND STABILITY OF FEEDBACK CONTROL

In the case of open-loop control of a robot, i.e., with no feedback, the actuator torques or forces can be directly computed from the dynamic model of the manipulator derived in Chapters 8 and 9. The dynamic model of a robot manipulator for a given trajectory, i.e., for known θ_d , $\dot{\theta}_d$, and $\ddot{\theta}_d$ the equations of motion are expressed as

$$\mathbf{I}\ddot{\theta}_d + \mathbf{h}_d + \boldsymbol{\gamma}_d = \boldsymbol{\tau}_d \quad (10.57)$$

where θ_d , $\dot{\theta}_d$, and $\ddot{\theta}_d$ denote the n -dimensional vectors— n being the number of joints in the manipulator—of desired joint positions, rates, and accelerations, respectively. These values are known from the trajectory generator, as indicated in Fig. 10.1. Moreover,

\mathbf{I} : the $n \times n$ Generalized Inertia Matrix (GIM), which is a function of the joint angles θ_d

\mathbf{h}_d : the n -dimensional vector of convective inertia (VCI) terms, which is a function of joint angles θ_d and rates $\dot{\theta}_d$

$\boldsymbol{\gamma}_d$: the n -dimensional vector of gravitational accelerations, which is a function of joint angles only, i.e., θ_d

$\boldsymbol{\tau}_d$: the n -dimensional vector of generalized forces which can be either joint torques or forces depending on the type of joints used, i.e., revolute or prismatic, respectively. The joint torques and forces are needed to drive the robot along the desired joint trajectory, i.e., θ_d , $\dot{\theta}_d$, and $\ddot{\theta}_d$.

Note that using Eq. (10.57), the robot will not follow the desired trajectory due to uncertainties in link lengths, masses and inertias, and the nonconsideration of friction, backlash, etc. Hence, feedback control is essential, where the actual joint and/or end-effector positions and sometimes their derivatives are fed back to the control system, as indicated by the terms *loop closed* in Fig. 10.1. To do this, robots

What is feedback?

Actual status of the robot, i.e., position and, quite often, the velocity of each joint, are fed back. Hence, the feedback is used in such control system.

are instrumented with sensors at each joint to measure the joint angles. Sometimes velocity sensors, e.g., tachometers, are also present in the joints. Principles of some of these sensors are explained in Chapter 4. For the feedback control, the robot accepts a vector of joint torques and forces, denoted with τ , from the control system. Equation (10.57) computes these joint torques and forces for the control system, whereas the manipulator's sensors read the vector of joint positions θ and joint velocities $\dot{\theta}$ for the controller. Note that the joint torques and forces τ_d are required to realize the desired trajectory θ_d . If the dynamic model was complete and accurate, and no *noise* or other disturbances were present, continuous use of Eq. (10.57) along the desired trajectory would realize it, and there will be no error between the desired and actual joint positions and velocities. Since this does not happen in reality due to the imperfections in the dynamic model and the inevitable presence of the disturbances, feedback is used to compute the error. This error is also called *servo error*, which is obtained by finding the difference between the desired and actual positions, and likewise between the desired and actual velocities, i.e.,

$$\mathbf{e} = \theta_d - \theta; \text{ and } \dot{\mathbf{e}} = \dot{\theta}_d - \dot{\theta} \quad (10.58)$$

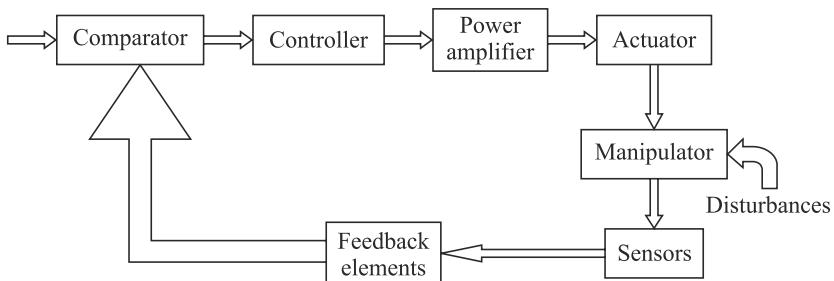


Fig. 10.17 A closed-loop controller

A control system then computes the torque as some function of the servo error to be sent to the actuators so that the servo error is minimum or zero. Such a control system that makes use of the feedback is called a closed-loop system. The *loop* closed by such a control system around the robot is apparent from Fig. 10.1, whereas Fig. 10.17 shows different interacting components of the closed-loop control system. In Fig. 10.17, the comparator is an electronics circuit that measures the error between the desired (also known as *reference*) output and the actual or measured output, and produces signals for the controller. A controller which can be a computer or a microcomputer or even a microprocessor implements a control strategy based on the above error signal. The signal from the controller is then converted to an analog signal using a Digital-to-Analog Converter (DAC) or any another electronics circuit, before it is amplified by the power amplifier to boost the small signal so that it can move the joint motor. A sensor is then used to detect the position and, possibly the velocity, of the joint, which are fed back to the comparator after the sensor's analog signal is converted to a digital signal using an Analog-to-Digital Converter (ADC).

In feedback control, control signal τ_d is not the only input which is acting on the system. Disturbances are also inputs, as shown in Fig. 10.17, which are not

controlled but influence the behavior of the output. Hence, the controller must be designed, in addition to the minimization of the servo error, to reduce the effects of the disturbances, namely, the robot should behave in a stable manner. If this is accomplished, the controller is said to reject the disturbances. The twin objectives of following the desired trajectory of *tracking* and *disturbance rejection* are central to any control methodology. It should be noted that an improperly designed control system can sometimes result in unstable performance in which servo errors are enlarged instead of reduced. Hence, the first task of a control engineer is to prove that his or her design yields a stable system, and the second is to prove that the closed-loop performance of the system is satisfactory, i.e., error is within the accepted limit. In practice, such proofs may range from mathematical proofs based on certain assumptions and models to more empirical results such as those obtained through numerical simulation or experimentation.

10.5.1 Performance of Feedback Control Systems

The ability to adjust the performance of a system is the distinct advantage of any feedback control. In order to analyze and design control systems, one must define and measure their performances. Then, based on the desired performance, the parameters of the system may be adjusted in order to provide the desired response. Since control systems are inherently dynamic systems, the performance is usually specified in terms of their time response for a specific input signal, and the resulting steady-state error. It is necessary to determine initially if the system is stable or not, i.e., whether the response to a specific input signal provides desired measures of performances. However, note that the actual input signal to a system is usually unknown. Hence, a standard test input signal is normally chosen. One of the standard test input signals is the step input. Step inputs are, in fact, common inputs to many physical processes as well. In control applications, a practice is to evaluate the behavior of a system on the basis of its step response. This approach is quite useful because there is a reasonable correlation between the response of a system to a standard test input and the system's ability to perform under normal operating conditions. Furthermore, using a standard input allows the designer to compare several competing designs.

In order to evaluate the performance of a control system, consider the system shown in Fig. 10.2, whose behavior will be evaluated against a unit step input. For such study, it is more suitable to use the form of its dynamic equation of motion given by Eq. (10.2a), where the corresponding transfer function (TF) is obtained in Eq. (10.30). For a unit step input, i.e., $f(s) = 1/s$, the output, $x(s)$, is given by

$$x(s) = G(s)f(s) = \frac{1}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \quad (10.59a)$$

which can be rearranged as

$$x(s) = G(s)f(s) = \frac{1}{s(s + s_1)(s + s_2)} \quad (10.59b)$$

where s_1 and s_2 are the roots of the characteristic equation or the poles of the system. Using the inverse Laplace transform formulas, the output x as a function of time t is obtained as

$$x = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin(\omega_n \sqrt{1-\zeta^2} t + \varphi), \text{ where } \varphi = \cos^{-1} \zeta \quad (10.60)$$

and the term $\omega_n \sqrt{1-\zeta^2}$ is called the damped natural frequency. Depending on the value of the damping ratio ζ , these poles can be real and unequal ($\zeta > 1$), or real and equal ($\zeta = 1$), or complex conjugates ($\zeta < 1$). Correspondingly, the system is referred as overdamped, critically damped and underdamped, respectively. For various values of the damping ratio ζ the responses of a typical second-order system with $m = 1$ and $k = 1$, i.e., $\omega_n = 1$, are shown in Fig. 10.18. As ζ decreases, poles of the system approach the imaginary axis and the response becomes increasingly oscillatory, whereas with increasing ζ , the response is sluggish.

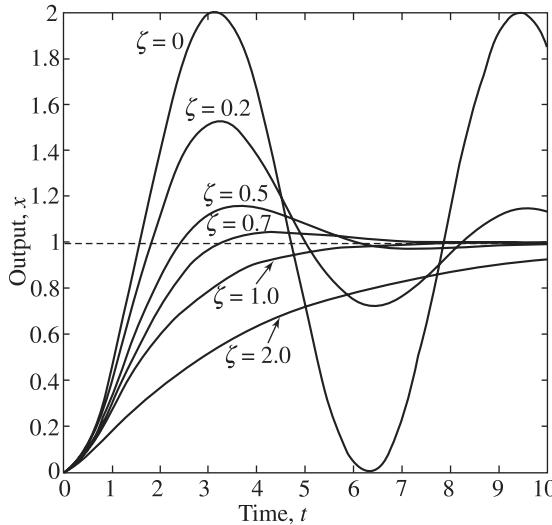


Fig. 10.18 Responses of a second-order system

Now, the performances measured with respect to the transient response of a step input to a system are shown in Fig. 10.19. The swiftness of the response is measured by the rise time T_r and the peak time T_p as indicated in the figure. For underdamped systems with $\zeta < 1$ and with overshoot, the rise time T_r is an useful index, whereas, for the overdamped systems with $\zeta > 1$ and no overshoot (Fig. 10.18), no peak time is defined. Hence, 10–90% rise time T_{rl} , as indicated in Fig. 10.19, is normally used. The similarity with which the actual response matches with the step input is measured by the percentage overshoot, and settling time T_s . The Percentage Overshoot (PO) is defined as

$$\text{PO} = \frac{M_{pt} - f_v}{f_v} \times 100\% \quad (10.61a)$$

where M_{pt} is the peak value of the time response and f_v is the final value of the response. Normally, f_v is the magnitude of the input, but many systems have a final value significantly different than the desired input magnitude. For the system with a

unit step represented by Eq. (10.59a), the value of f_v is one. The corresponding PO, as derived in a control text book, e.g., Mohan (1997), is given by

$$PO = 100e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \text{ \%} \quad (10.61b)$$

Next, the settling time T_s is defined as the time required for the system to settle within a certain percentage δ of the input amplitude. The band of $\pm\delta$ is shown in Fig. 10.19. For the second-order system at hand, the response remains within 2% after four time-constants $1/(\zeta\omega_n)$, i.e.,

$$T_s = \frac{4}{\zeta\omega_n} \quad (10.62)$$

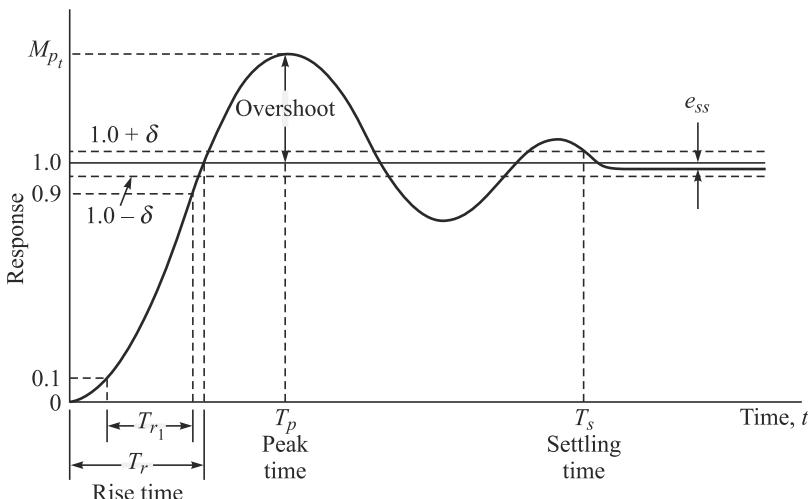


Fig. 10.19 Step response of a second-order system

Finally, the steady-state error of the system, denoted with e_{ss} , is measured on the step response of the system as shown in Fig. 10.19. On the other hand, the requirements of the transient response are the swiftness of response T_r and T_p , and the closeness of the response to the desired M_{pt} and T_s , which are contradictory. Hence, a compromise must be obtained. For a robotic system, oscillatory response cannot be allowed, particularly, in a pick and place operation, where an oscillatory end-effector may strike against the object before picking it to manipulate. Hence, highest possible speed and yet non-oscillatory response dictates that the controller design parameters shall be chosen to have the damping ratio equal to one or close to it but less than unity.

10.5.2 Lyapunov Stability

At the beginning of this chapter, stability of a linear control system was achieved in Examples 10.1 to 10.3 by introducing damping. The performance of such systems during transient behavior was shown in Section 10.2. The same analyses are valid for a nonlinear system that is decoupled and linearized by means of a model-based

controller, as presented in Chapter 11, because the overall system is again linear. However, when decoupling and linearization are not performed, or are incomplete or inaccurate, the overall closed-loop system remains nonlinear. For nonlinear systems, stability, and performance analysis can be performed using Lyapunov stability analysis or Lyapunov's second method. An interesting characteristic of this method of stability analysis is that one can conclude without solving the differential equation governing the system

or finding the roots (poles) of the characteristic polynomial of a linear system or using Routh's criterion for stability which may be a tedious job. However, the present method of Lyapunov stability does not provide any information about the transient response or performance of the system, for example, overdamped or underdamped or how long it would take to suppress a disturbance, etc. See Section 10.5.1 for definitions of several performance-related measures. It is important though to differentiate stability from performance because a stable system may have unsatisfactory control performance.

To explain Lyapunov's method which is also known in the literature, e.g., Slotine and Li (1991), Gopal (2004) and others, as Lyapunov's second or direct method, consider a nonlinear system described by a set of differential equations given in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \quad (10.63)$$

where \mathbf{x} is the n -dimensional state vector, as introduced in Section 10.3.2, and $\mathbf{f}(\mathbf{x})$ is the n -dimensional vector function. Note that any higher-order differential equations can always be written as a set of first-order equations in the form of Eq. (10.63). See Examples 10.6 and 10.10 where second-order systems were represented in state-space form. To prove a system is stable using the Lyapunov's method, one needs the following:

1. Propose a generalized energy function $v(\mathbf{x})$ called Lyapunov function that has continuous first partial derivatives, and $v(\mathbf{x}) > 0$ for all \mathbf{x} , except $v(\mathbf{0}) = 0$.
2. Evaluate first time-derivative of $v(\mathbf{x})$.
3. Check if $\dot{v}(\mathbf{x}) \leq 0$. Here, $\dot{v}(\mathbf{x})$ implies the change of $v(\mathbf{x})$ with time along all system trajectories.

The idea is that a positive definite energylke function of state is shown to always decrease or remain constant. Hence, the system is stable in the sense that the size of the state vector is bounded. When $\dot{v}(\mathbf{x})$ is strictly less than zero, the state asymptotically converges to the zero vector. However, for $\dot{v}(\mathbf{x}) \leq 0$, i.e., when $\dot{v}(\mathbf{x})$ may vanish at certain places of the trajectory, LaSalle and Lefschetz (1961) showed that the system still can be asymptotically stable under certain situations.

Note that the Lyapunov's method is equally applicable to linear systems. It provides a simple approach to stability analysis. For a linear system, a Lyapunov function can always be constructed, and both the necessary and sufficient conditions on stability can be established. For this, consider the state-space representation given in Eq. (10.31). When the system is not forced, i.e., $\mathbf{u} = \mathbf{0}$, the linear system can be represented as

Stability

Stability means that a system will not diverge from its intended position. For example, a ball kept inside a bowl is stable, whereas it is unstable while kept on the inverted bowl.

$$\dot{\mathbf{x}} = \mathbf{Ax} \quad (10.64)$$

The linear time invariant system, Eq. (10.64), is globally asymptotically stable at the origin if, and only if, for any given symmetric positive definite (SPD) matrix \mathbf{R} , there exists a SPD matrix \mathbf{P} that satisfies the matrix Lyapunov equation given by

$$\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A} = -\mathbf{R} \quad (10.65)$$

The corresponding Lyapunov function for the system of Eq. (10.64) given by $v(\mathbf{x})$ is then equal to $\mathbf{x}^T \mathbf{P} \mathbf{x}$, i.e., $v(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$. One important aspect of using Eq. (10.65) is the difficulty of proving if \mathbf{R} is SPD for a given choice of the SPD matrix \mathbf{P} . In fact, for the chosen SPD matrix \mathbf{P} , if \mathbf{R} turns out to be nonpositive definite then nothing can be stated about the stability of the system. Hence, an alternate approach is taken. Choose a matrix \mathbf{R} as SPD, and solve for \mathbf{P} using Eq. (10.65). If the solved matrix is SPD then the system is stable. Hence, the stability problem boils down to a solution of a linear system of equations.

Example 10.13 Stability of the Block

Consider the block of Fig. 10.2 whose governing equation is given by Eq. (10.1), i.e.,

$$m\ddot{x} + b\dot{x} + kx = f \quad (10.66)$$

The total energy of the system comprising of the kinetic energy (KE) due to the velocity of the block \dot{x} plus the potential energy (PE) due to the deflection of the spring x , i.e.,

$$v = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2 \quad (10.67)$$

where the first term in Eq. (10.67) represents the KE, and the second one is the PE. Note that the value of v is always nonnegative, as required in Step 1 of the Lyapunov's stability method explained above. The rate of change of v is then obtained by differentiating Eq. (10.67), as

$$\dot{v} = m\dot{x}\ddot{x} + kx\dot{x} \quad (10.68a)$$

Substitution of Eq. (10.66) for $m\ddot{x}$ into Eq. (10.68a) yields

$$\dot{v} = f - bx^2 \quad (10.68b)$$

which for no external excitation, i.e., $f = 0$, is always negative since $b > 0$. Thus, energy is always leaving the system unless $\dot{x} = 0$. This implies that an initially displaced block under no external excitation will lose energy until it rests. A steady-state analysis of the system (i.e., $\dot{x} = 0$, $\ddot{x} = 0$) at resting position (i.e., $x = 0$) reveals that $kx = 0$. Hence, the block initially at rest will continue to be at rest.

Example 10.14 Stability of a Single-DOF Planar Arm

To check the stability of the single-DOF planar arm, as shown in Fig. 10.11, at $\theta = 0$ when no external torque is acting, i.e., $\tau = 0$, consider the following Lyapunov function:

$$v = \frac{1}{2} \frac{ma^2}{3} x_2^2 + mg \frac{a}{2} (1 - \cos x_1) \quad (10.69)$$

where x_1 and x_2 are the state variables defined in Example 10.10. Moreover, Eq. (10.69) represents the total energy of the system which is always greater than zero, and zero when the state variables vanish, i.e., $x_1 = x_2 = 0$. Note that the second-term, i.e., the potential energy is calculated based on zero potential energy at the coordinate $x = a/2$. Time derivative of Eq. (10.69) is then obtained as

$$\dot{v} = \frac{ma^2}{3}x_2\dot{x}_2 + mg\frac{a}{2}\sin x_1\dot{x}_1 \quad (10.70)$$

Equation (10.70) is zero for the equilibrium condition of the link, i.e., $\theta = \dot{\theta} = 0$, or in terms of the state-space variables $x_1 = x_2 = 0$. Since \dot{v} is not less than zero asymptotic stability cannot be guaranteed.

Example 10.15 Proof of Lyapunov Equation, $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{R}$

For the given state-space representation of a linear system, Eq. (10.64), consider the Lyapunov function $v(\mathbf{x})$ as

$$v(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x} \quad (10.71)$$

where \mathbf{P} is a constant SPD matrix. Differentiating the positive definite function, Eq. (10.71), yields another matrix of the quadratic form, i.e.,

$$\dot{v}(\mathbf{x}) = \dot{\mathbf{x}}^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P} \dot{\mathbf{x}} \quad (10.72a)$$

Substituting the state-space form of Eq. (10.64) into Eq. (10.72a), one gets the following:

$$\dot{v}(\mathbf{x}) = \mathbf{x}^T \mathbf{A}^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P} \mathbf{A} \mathbf{x} = \mathbf{x}^T (\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A}) \mathbf{x} \quad (10.72b)$$

For stability, $\dot{v}(\mathbf{x})$ has to be negative, i.e., $\dot{v}(\mathbf{x}) < 0$. This implies that

$$\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A} = -\mathbf{R} \quad (10.73)$$

where matrix \mathbf{R} is the SPD.

Example 10.16 Stability of the Block using Lyapunov Equation

For the state-space representation of the moving block given by Eq. (10.35), the 2×2 state matrix \mathbf{A} is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix}, \text{ where } m = 1, b = 5 \text{ and } k = 6 \quad (10.74)$$

Consider \mathbf{R} as the 2×2 identity matrix. Then Eq. (10.73) is expressed as

$$\begin{bmatrix} 0 & -6 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad (10.75a)$$

Since matrix \mathbf{P} needs to be symmetric $p_{12} = p_{21}$. Substitution of this in Eq. (10.75a) yields the following set of linear equations:

$$12p_{21} = 1; p_{11} - 5p_{21} - 6p_{22} = 0; 2(p_{21} - p_{22}) = -1 \quad (10.75b)$$

Solving for the elements p_{ij} , the 2×2 matrix \mathbf{P} is then obtained as

$$\mathbf{P} = \frac{1}{60} \begin{bmatrix} 67 & 5 \\ 5 & 7 \end{bmatrix} \quad (10.75c)$$

Using Sylvester's criterion (Datta, 1999), one can check the positive definiteness of the matrix by simply checking if all its principal minors, i.e., p_{11} and $p_{11}p_{22} - p_{21}p_{12}$, are strictly positive or not. For the matrix \mathbf{P} given by Eq. (10.75c), they are true. Hence, the system under study is asymptotically stable, as seen from the negative real parts of the poles found in Example 10.4 and the time response shown in Fig. 10.7. For the solved matrix \mathbf{P} given by Eq. (10.75c), the Lyapunov function and its time-derivative are then equal to

$$v(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x} = \frac{67}{60} x_1^2 + \frac{1}{6} x_1 x_2 + \frac{7}{60} x_2^2, \text{ and } \dot{v}(\mathbf{x}) = -x_1^2 - x_2^2 = -(x^2 + \dot{x}^2) < 0 \quad (10.76)$$

where \dot{x}_1 and \dot{x}_2 are substituted from Eqs. (10.35) and (10.74), along with the definitions of state vector $x_1 = x$ and $x_2 = \dot{x}_1 (\equiv \dot{x})$. Equation (10.76) shows that the block is asymptotically stable.

10.6 PROPORTIONAL-DERIVATIVE-INTEGRAL (PID) CONTROL OF A MOVING BLOCK

In this section, an illustration of feedback control using the damped mass-spring system of Section 10.2 is provided. Suppose that the natural response of the block is not as desired. Perhaps it is underdamped and oscillatory and one likes it to be critically damped. It may be possible that the spring is missing altogether, i.e., $k = 0$. So, the system never returns to $x = 0$ once disturbed. Through the use of sensors, an actuator, and a control system, one can modify the system's behavior. As shown in Fig. 10.2(a), one can modify the input force f in a manner so that the desired behavior, say, critically damped for fastest transient response, is achieved. It is now assumed that there are sensors which are capable of detecting the block's position and velocity. A control law is then proposed which computes the force and should be applied by an actuator as a function of the sensed feedback, i.e.,

$$f = k_p(x_d - x) + k_v(\dot{x}_d - \dot{x}) \quad (10.77)$$

where x_d and \dot{x}_d are the desired position and velocity of the block. If the block has to be maintained at rest in the origin then $x_d = 0$ and $\dot{x}_d = 0$. The control layout is depicted in Fig. 10.20.

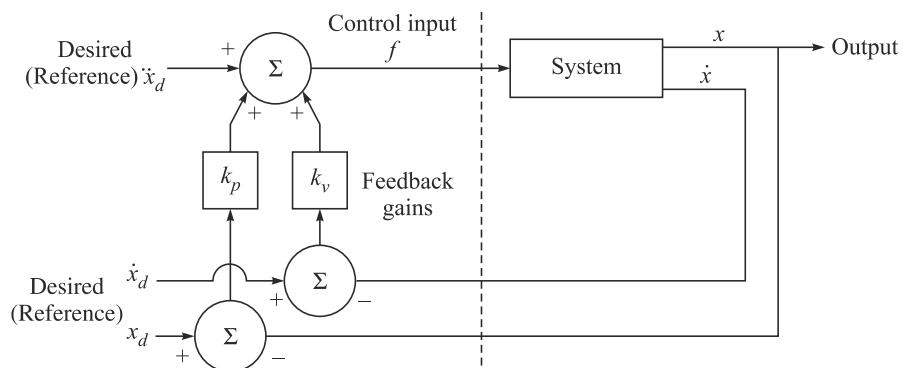


Fig. 10.20 A feedback control system

The controller reads sensor input and writes the actuator output commands where \ddot{x}_d represent the desired acceleration that is also taken as zero, i.e., $\ddot{x}_d = 0$. Note that the left of the dashed line in Fig. 10.20 is the control system (usually implemented in a computer) whereas the right of the dashed line is the physical system. Implicit in the figure are the interfaces between the control computer and the output actuator commands along with the input sensor information. The proposed control system is a position regulation system. It simply attempts to maintain the position of the block in one fixed place regardless of disturbance forces applied to the block. By equating the open-loop dynamics of the block, Eq. (10.1) with the control law of Eq. (10.77), the closed-loop dynamics is derived as

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} \quad (10.78a)$$

$$\text{or} \quad m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0 \quad (10.78b)$$

which yields

$$m\ddot{x} + b'\dot{x} + k'x = 0 \quad (10.78c)$$

where, $b' \equiv b + k_v$ and $k' \equiv k + k_p$. From Eqs. (10.78b-c), it is clear that by choosing the control gains k_v and k_p , the closed-loop system can be made behave like any of the second-order system explained in Section 10.2. Gains can be chosen to obtain critical damping, i.e., $b' = 2\sqrt{mk'}$. Note that k_v and k_p may be positive or negative depending on the parameters of the original system. However, if b' or k' became negative, the result would be an unstable system. This instability would be obvious if one writes down the solution of the closed-loop second-order differential equation, Eq. (10.78b), in the form of Eqs. (10.4), (10.12) or (10.21). It also makes intuitive sense that if b' or k' are negative, servo errors tend to get magnified rather than reduced.

Type-0, Type-1, and Type-2 Systems

Systems having a finite non-zero steady-state errors when the desired inputs is zero (a step)-, first (a ramp)-, and second (a parabola)-order polynomials are labeled as Type-0, Type-1 and Type-2 systems, respectively.

Note that the above control law can also be explained using the concept of transfer function (TF), as introduced in Fig. 10.8 and given by Eq. (10.30) for a given force input f . However, for feedback control, the forcing function f is chosen as per Eq. (10.77) so that the block follows or tracks a desired trajectory denoted by x_d which is also generally a function of time. Hence, the closed-loop system for the feedback control is shown in Fig. 10.21. Using the rules of the block diagram manipulations, the output $x(s)$ can be written as

$$x(s) = (x_d - x)G(k_p + k_v s) \quad (10.79a)$$

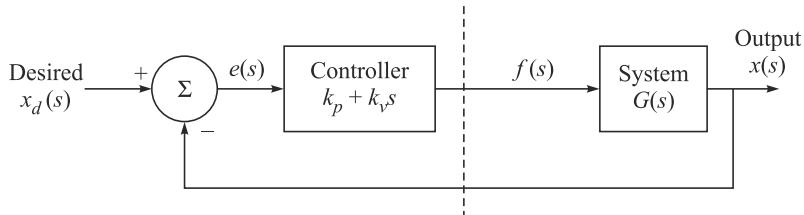


Fig. 10.21 Transfer function representation of feedback control

In Fig. 10.21, the input signal to the controller is the error $e(s)$ defined as the difference between the actual and desired outputs that generates the necessary force, $f(s)$, i.e., $f(s) = (k_p + k_v s)e(s)$. Rearranging Eq. (10.79a), one can then easily obtain the closed-loop TF relating the output $x(s)$ with the desired output $x_d(s)$ as

$$\frac{x(s)}{x_d(s)} = \frac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)} \quad (10.79b)$$

From Eq. (10.79b), it is clear that for a desired behavior of the closed-loop system, the poles of the characteristic equation should be chosen appropriately. From Eq. (10.79b), the characteristic equation is given by

$$1 + G(s)(k_p + k_v s) = 0 \quad (10.80)$$

Substituting the expression for the TF $G(s)$, given by Eq. (10.30), one can rewrite Eq. (10.80) to yield the characteristic equation in terms of the system's physical parameters and controller gains. The resultant closed-loop characteristic equation is critically damped when its poles are equal, i.e., $s_1 = s_2$. For such situation, $b' = 2\sqrt{mk'}$, which is also obtained from the time domain analysis presented in Section 10.6.

In many instances, the position and velocity gains may not be sufficient to satisfy the control requirements, particularly, to eliminate the steady-state errors. For such cases, it is suggested to introduce an integral controller in the control law of Eq. (10.79a) as

$$\begin{aligned} m\ddot{x} + b\dot{x} + kx &= -k_p x - k_v \dot{x} - k_i \int x dt \\ \text{or} \qquad x(s) &= (x_d - x)G\left(k_p + k_v s + \frac{1}{s}k_i\right) \end{aligned} \quad (10.81a)$$

Correspondingly, the TF of Eq. (10.79b) is modified as

$$\frac{x(s)}{x_d(s)} = \frac{G(s)\left(k_p + k_v s + \frac{1}{s}k_i\right)}{1 + G(s)\left(k_p + k_v s + \frac{1}{s}k_i\right)} \quad (10.81b)$$

Note that the introduction of integral gain to the system makes it a Type-1 (Gopal, 1997) system. Hence, the steady-state error to a step input is zero. As a result, any value of k_i will eliminate steady-state error due to the step input. However, for ramp-input there will be again some steady-state error. More discussions on these topics are available in any textbook on Control, e.g., Ogata (1987), Kuo (1991) and Gopal (1997).

Example 10.17 Control of a Block with Proportional Gain

For the parameters $m = 1$, $b = 1$, and $k = 1$ given in Example 10.4, the TF of the block is obtained from Eq. (10.30) as

$$G(s) = \frac{x(s)}{f(s)} = \frac{1}{s^2 + s + 1} \quad (10.82a)$$

The response of the block for unit step input, i.e., $f(t) = 1$, without any controller, i.e., when $k_p = 0$, is shown in Fig. 10.22. As such the behavior of this open-loop system may be acceptable. However, due to variations in system parameters, e.g., mass of the block which may change due to, say, consumption of the fuel of a car if it represents a block, and other disturbances, one may want to introduce a controller. If only position controller is used without any velocity gain, i.e., $k_v = 0$, the closed-loop TF from Eq. (10.79b) is given by

$$G(s) = \frac{x(s)}{x_d(s)} = \frac{k_p}{s^2 + s + (1 + k_p)} = \frac{k_p}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (10.82b)$$

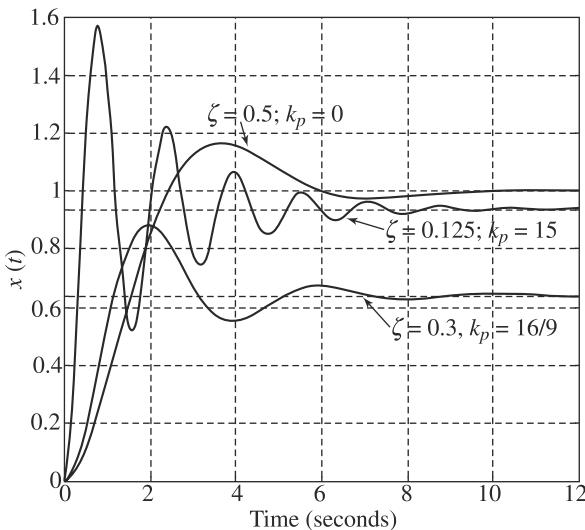


Fig. 10.22 Effect of position control gains

The responses of the block for unit step input, i.e., $x_d(t) = 1$, which is the case in set-point control for various control gains corresponding to various values of ζ and k_p such that $2\zeta\omega_n = 2\zeta\sqrt{1+k_p} = 1$ ($\omega_n^2 = 1 + k_p$) are shown in Fig. 10.22. Note that for the controlled system, as the values of k_p is increasing, the steady-state error is reducing. However, oscillations have increased, whereas for low k_p values steady-state error is high with reduced oscillations. Hence, a trade-off has to be made in order to achieve balanced response of a system at hand. One remedy to reduce oscillation for the reduced steady-state error is to introduce a velocity controller, i.e., consider non-zero k_v , which is shown in the next example.

Example 10.18 Control of the Block with Proportional and Derivative Gains

As mentioned in Example 10.17, velocity gain can be introduced to reduce the oscillation for $\zeta = 0.125$. To do this, one requires to increase the value of ζ . Hence, a value of k_v was calculated as 3.8 for $\zeta = 0.6$ from the condition of $2\zeta\sqrt{1+k_p} = 1+k_v$ (case of critical damping). Corresponding unit-step response is shown in Fig. 10.23. Note that for critical damping when the fastest non-oscillatory motion is possible, $\zeta = 1.0$. Corresponding velocity gain is $k_v = 7$ for which the response is also shown in Fig. 10.23. This condition is nothing but the same stated after Eq. (10.80), i.e., $b' = 2\sqrt{mk'}$. For parameters of the block, i.e., $m = 1$, $b = 1$ and $k = 1$, the gains k_p and k_v for a position regulation control law are to be found out for a critically damped system with the closed loop stiffness of 16.0.

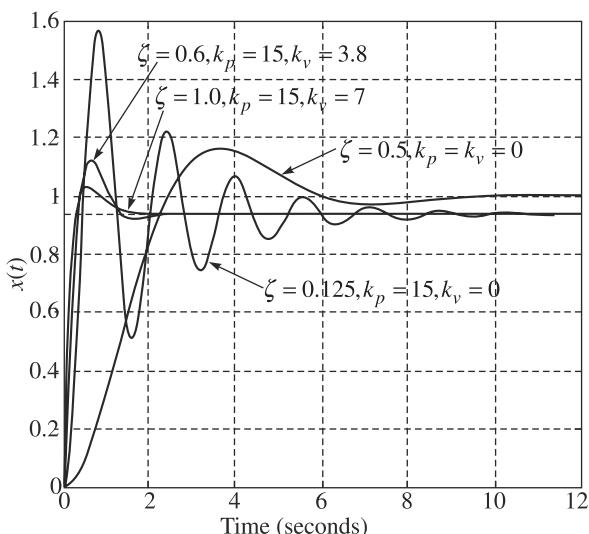


Fig. 10.23 Effect of velocity control gains

It is now clear that velocity gain has improved the oscillatory behavior of the response but not much the steady-state error. This is quite well known in the control literature. One needs to introduce an integral term or integral controller to minimize this error.

Example 10.19 Control of the Block with Proportional, Derivative and Integral Gains

As seen in Example 10.18, the introduction of derivative gain could not eliminate the steady-state errors. Hence, an integral controller, as proposed in Eq. (10.81), is introduced with $k_i = 2$. The corresponding response is shown in Fig. 10.24.

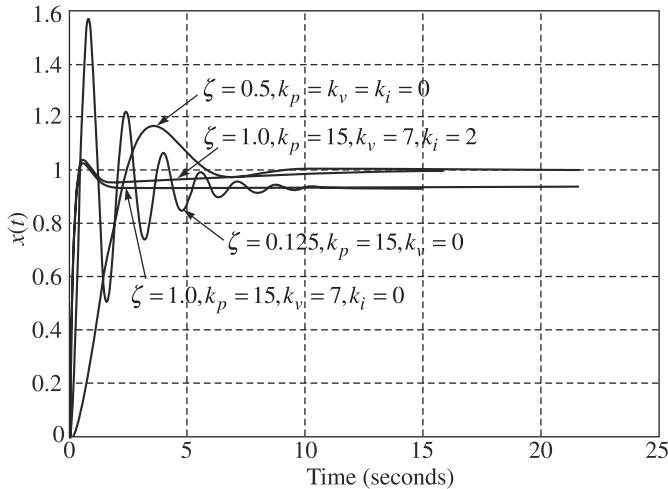


Fig. 10.24 Effect of integral gain

10.7 SELECTION OF PID CONTROLLER GAINS

One of the main problems occur in implementing the above controllers, mainly, the PID controller, is the choice of controller gains, which is also called tuning. One common design rule-of-thumb is to first set $k_i = 0$ and design the proportional and derivative gains k_p and k_v , respectively, to obtain the desired transient behavior, i.e., rise time, settling time, and so forth. Effect of joint flexibility can also be taken into account while choosing the gains. For example, let k_j be the effective stiffness of the joint. Then, the joint resonant frequency is given by, $\omega_j = \sqrt{k_j/I}$. It is common engineering practice to limit $\omega_n = \sqrt{k_p/I}$, where k_p is the proportional gain and I is the system's inertia to no more than half of ω_j to avoid excitation of the joint resonance. For the selection of k_p and k_v , one must remember that a proportional controller with gain k_p will reduce the rise time and the steady-state error, but will never eliminate it, whereas a derivative control with gain k_v will have the effect of increasing the stability of the system, i.e., reducing the overshoot, and improving the transient response. Finally, an integral control with gain k_i will eliminate the steady-state error, but it may make the transient response worse. These points are summarized in Table 10.1.

Resonant Frequency

In this frequency, a road-bridge will collapse if allowed to vibrate for a while.

It is pointed out here that the correlations given in Table 10.1 may not be exactly accurate, because k_p , k_v , and k_i are dependent on each other. In fact, changing one of these variables can change the effect of the other two. For this reason, the table should only be used as a reference when one determines the gain values. Once k_p and k_v are chosen, the gain k_i is obtained from Routh–Hurwitz criterion (Ogata, 1987), i.e.,

Table 10.1 Effect of controller gains

Closed-loop response	Rise time	Overshoot	Settling time	Steady-state error
k_p	Decrease	Increase	Small change	Decrease
k_v	Small change	Decrease	Decrease	Small change
k_i	Decrease	Increase	Increase	Eliminate

$$k_i < \frac{(b + k_v)}{I} k_p \quad (10.83)$$

where b and k_v are the damping coefficient and velocity gain of the controller respectively.

Finally, it can be kept in mind that one need not implement all three controllers (proportional, derivative, and integral) into a single system unless it is necessary. For example, if a PI controller gives a good enough response then it is not necessary to implement derivative controller to the system. A controller should be kept as simple as possible.

10.8 STATE-FEEDBACK CONTROL

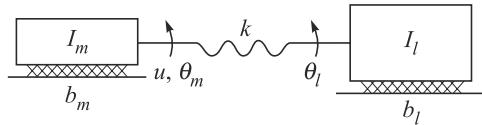
If each joint of the robot is considered independently controlled, then the state-space representation of each joint is given by Eqs. (10.31) and (10.38). A linear state-feedback control for it can be given by

$$u(t) = -\mathbf{k}^T \mathbf{x} + \theta_d \quad (10.84)$$

where each element of \mathbf{k} is constant gain to be determined and θ_d is a reference or desired input. Substituting Eq. (10.84) into Eq. (10.38b) one obtains

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{b}\mathbf{k}^T)\mathbf{x} + \mathbf{b}\theta_d \quad (10.85)$$

which shows that the linear feedback control has the effect of changing the poles of the system from those determined by \mathbf{A} to those determined by $(\mathbf{A} - \mathbf{b}\mathbf{k}^T)$. In Eq. (10.84), the control is determined as a linear combination of the system states which, in this case, are the motor position and velocity. Comparing this with the PD or PID control, as presented in Section 10.6, no advantage is gained. However, for a larger dimensional system, for example, a joint with its flexibility taken into account, as demonstrated in Fig. 10.25, the state-space feedback can be proven to be advantageous.

**Fig. 10.25** Single joint with flexibility

Note that joint flexibility can occur due to many reasons. For example, the torsional flexibility of the gears, shaft windup, bearing deformation, etc., can contribute substantially towards flexibility in the joints. If one writes its dynamic model, two second-order differential equations will result corresponding to its two-DOF, namely, θ_m and θ_l , corresponding to the motor and link or load rotations, respectively. In such systems, if the damping coefficients b_m and b_l are small, which is often the case in many industrial robots, it is difficult to control the system, as the two of four poles are located at the origin, whereas the other two lie on the imaginary axis.

Suppose if the traditional PD control of Eq. (10.77) or Fig. 10.20 is implemented to control the flexible joint then computation of its two gains depends on whether the position and velocity sensors are placed on the motor shaft or on the link shaft, i.e., whether the PD controller is a function of the motor variables or the load variables. In such control, it can be shown (Spong et al., 2006) that the system is unstable for large proportional gain.

Alternatively, if the state feedback control of the form given by Eq. (10.84) is applied, it is possible to achieve a much larger range of closed-loop poles, thereby, making the system stable. This is possible because there are four gain parameters in Eq. (10.84) corresponding to the 4-dimensional state vector \mathbf{x} , i.e., motor angle, motor velocity, load angle, and load velocity, which can be measured by putting appropriate sensors at motor and load shafts. The gain \mathbf{k} unlike the traditional PD control where the two gains are functions of either motor position and velocity, or load position and velocity, is function of all four. Hence, with the state-feedback there are more free parameters to obtain the stability. However, the issues like controllability and observability of the state are also important that determine the success of the state-feedback control law. A given state-space system is controllable if its state matrix \mathbf{A} and the state input vector \mathbf{b} satisfies the following condition:

$$\det(\mathbf{U}) \neq 0, \text{ where } \mathbf{U} \equiv [\mathbf{b} \quad \mathbf{Ab} \quad \dots \quad \mathbf{A}^{n-1}\mathbf{b}] \quad (10.86a)$$

The $2n \times 2n$ matrix \mathbf{U} is called the controllability matrix. In case the linear system is controllable then its states should be measurable or should be able to estimate. Whereas some states like position, velocity can be directly measured with relative ease using physical sensors, others cannot be measured so easily or very costly to do so. In such cases, one can theoretically estimate them using what is known as state observer provided the linear system at hand is observable. This can be checked from the given state matrix \mathbf{A} and the state output vector \mathbf{c} , i.e.,

$$\det(\mathbf{V}) \neq 0, \text{ where } \mathbf{V} \equiv [\mathbf{c} \quad \mathbf{A}^T\mathbf{c} \quad \dots \quad (\mathbf{A}^T)^{n-1}\mathbf{c}] \quad (10.86b)$$

The $2n \times 2n$ matrix \mathbf{V} is called the observability matrix. Assuming the system is both controllable and observable, Eqs. (10.35a-b), can be combined to provide the state differential equations for the closed-loop system with $\mathbf{u} = -\mathbf{k}^T\mathbf{x}$ as

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{b}\mathbf{k}^T)\mathbf{x} \quad (10.87)$$

The corresponding characteristic equation of the closed-loop system is given by

$$|s\mathbf{1} - (\mathbf{A} - \mathbf{b}\mathbf{k}^T)| = 0 \quad (10.88)$$

which yields an second-order polynomial for the moving block of Examples 10.1–10.3. However, in general it will lead to an n^{th} order polynomial. Then, the control-law design would be to pick up the gains, i.e., the elements of \mathbf{k} so that the roots of Eq. (10.88) are in desirable locations, as mentioned in Section 10.6. Note that the roots of the characteristic equation given by Eq. (10.88) are nothing but the eigenvalues of the matrix $(\mathbf{A} - \mathbf{b}\mathbf{k}^T)$. If all eigenvalues of $(\mathbf{A} - \mathbf{b}\mathbf{k}^T)$ are placed on the left-half plane, the closed-loop system is asymptotically stable. In order to determine the state-feedback gains, i.e., vector \mathbf{k} , one can first expand Eq. (10.88)

Cayley–Hamilton Theorem

It states that a matrix satisfies its own characteristic equation.

as a characteristic polynomial and choose the values for the gains k_i 's according to the desired natural frequency ω_n , damping ratio ζ , and pole locations s_i 's. Alternatively, one can use Ackermann's formula (Gopal, 2003), which is given by

$$\mathbf{k}^T = [0 \quad 1] \mathbf{U}^{-1} \Phi(\mathbf{A}) \quad (10.89a)$$

where \mathbf{U} is the 2×2 controllability matrix which is defined in Eq. (10.86a), and the matrix $\Phi(\mathbf{A})$ is given by

$$\Phi(\mathbf{A}) \equiv \mathbf{A}^n + \mathbf{a}_{n-1} \mathbf{A}^{n-1} + \cdots + \mathbf{a}_1 \mathbf{A} + \mathbf{a}_0 \mathbf{1} \quad (10.89b)$$

The coefficients of $\Phi(\mathbf{A})$, i.e., a_i 's, are those which are associated to the desired characteristic polynomial of the closed-loop system because as per Cayley–Hamilton theorem any matrix satisfies its characteristic equation. In Eq. (10.89b), $\mathbf{1}$ denotes the $2n \times 2n$ identity matrix.

A more advanced but very popular way to design the feedback gains is through an optimization procedure, which is covered in optimal control theory (Gopal, 2003). For example, if one chooses the performance criterion J as

$$\mathbf{J} = \int_0^{\infty} (\mathbf{x}^T \mathbf{R} \mathbf{x} + \bar{R} u^2) dt \quad (10.90)$$

where \mathbf{R} is a given symmetric matrix and $\bar{R} > 0$ then the control law that minimizes the integral of Eq. (10.90) is given by

$$u = -\mathbf{k}^T \mathbf{x}, \text{ where } \mathbf{k} \equiv R^{-1} \mathbf{b}^T \mathbf{P} \quad (10.91)$$

In Eq.(10.91), \mathbf{P} is the unique symmetric positive definite (SPD) matrix satisfying the so called matrix Riccati equation given by

$$\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A} - \mathbf{P} \mathbf{b} \bar{R}^{-1} \mathbf{b}^T \mathbf{P} + \mathbf{R} = \mathbf{0} \quad (10.92)$$

The control law given by Eq. (10.91) is referred to as a linear quadratic optimal control because the performance index is quadratic but the control system is linear. Note that in general \bar{R} is a matrix for multiple input system. Hence, the notation \bar{R}^{-1} is used even though for the case of Eq. (10.92) it is a scalar quantity.

Example 10.20 State-feedback Control of the Block

For the second-order system of moving block, if Eqs. (10.35a-b) are used to represent the state of the system at hand, a linear control law of the following form will allow one to achieve a desired performance:

$$u = -\mathbf{k}^T \mathbf{x}, \text{ where } \mathbf{k} \equiv [k_1 \quad k_2]^T \text{ and } \mathbf{x} \equiv [x_1 \quad x_2]^T \quad (10.93)$$

The scalars k_1 and k_2 in Eq. (10.93) are called state-feedback gains, where x_1 and x_2 are the state variables given by Eq. (10.34a). Note that for the single degree-of-freedom system under study with two state variables, there are only two gains which can be controlled in a way similar to, say, proportional and derivative gains mentioned in Section 10.6. Assume that it has to be controlled using state-feedback formulation of Section 10.8 in order to achieve the behavior so that $\zeta = 1.0$. Moreover, take desired settling time as 1 s. Using state-space representation of the block with feedback controller, as given by Eq. (10.87), first the transfer function is written using

Eq. (10.33c). It leads to the characteristic polynomial of order two with feedback gains appear as coefficients, i.e.,

$$|s\mathbf{1} - (\mathbf{A} - \mathbf{b}\mathbf{k}^T)| = s^2 + (1 + k_2)s + (1 + k_1) = 0 \quad (10.94)$$

Now, comparing Eq. (10.94) with the denominator of Eq. (10.30) with $\zeta = 1.0$ yields

$$(1 + k_2) = 2\omega_n, \text{ where } \omega_n \equiv \sqrt{(1 + k_1)} \quad (10.95)$$

Since the desired settling time is given by Eq. (10.62), one can have the following condition:

$$\frac{4}{\zeta\omega_n} = 1, \text{ where } \zeta = 1 \quad (10.96)$$

Equation (10.96) immediately gives $k_1 = 15$. Substituting the value of k_1 into Eq. (10.95), the value of k_2 is obtained as $k_2 = 7$.

It is interesting to note that the gain values, namely, k_1 and k_2 , are nothing but those obtained in Example 10.17 as proportional and derivative gains, k_p and k_v , respectively. This is mainly due to their association with the state variables which actually represent position and velocity of the block, respectively. The corresponding response is shown in Fig. 10.23. Moreover, the settling time, as defined in Section 10.5.1, can be verified from Fig. 10.23 as 1 s.

Example 10.21 State-feedback Gains using Ackermann Formula

If the gains of Example 10.19 have to be found out using Eq. (10.89a), the following steps need to be followed:

$$\mathbf{k} = \Phi^T(\mathbf{A})\mathbf{U}^{-T} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ where } \mathbf{U} \equiv [\mathbf{b} \quad \mathbf{Ab}] = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \quad (10.97)$$

From the above expression of \mathbf{U} , it is a simple matter to find the inverse as

$$\mathbf{U}^{-1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad (10.98)$$

Moreover, the characteristic matrix polynomial $\Phi(\mathbf{A})$ corresponding to the desired characteristic polynomial, given by $s^2 + 8s + 16$, can be calculated as

$$\Phi(\mathbf{A}) \equiv \mathbf{A}^2 + 8\mathbf{A} + 16(\mathbf{1}) = \begin{bmatrix} 15 & 7 \\ 0 & 15 \end{bmatrix} \quad (10.99)$$

where the coefficients of $\Phi(\mathbf{A})$ are same as those of desired characteristic polynomial. Evaluation of Eq. (10.97) then yields

$$\mathbf{k} = \Phi^T(\mathbf{A})^T \mathbf{U}^{-T} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 & 0 \\ 7 & 15 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 \\ 7 \end{bmatrix} \quad (10.100)$$

which are same as those obtained in Example 10.20.

Example 10.22 Optimal Controller

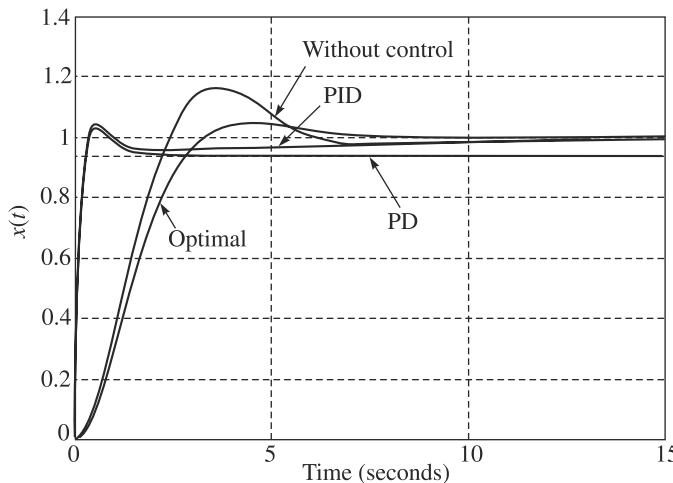
Consider the control of Example 10.21 again. This time a controller is designed based on the law provided in Eq. (10.92). For the following conditions,

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \bar{R} = 1 \quad (10.101)$$

The values of the elements of matrix \mathbf{P} and correspondingly gain matrix \mathbf{k} can then be evaluated as

$$\mathbf{P} = \begin{bmatrix} 0.4142 & 0 \\ 0 & 0.4142 \end{bmatrix} \text{ and } \mathbf{k} = \begin{bmatrix} 0 \\ 0.4142 \end{bmatrix} \quad (10.102)$$

The behavior of the system is shown in Fig. 10.26 along with responses of the original and other control systems presented earlier, namely, PD and PID. The MATLAB program shown in Fig. 10.26(b) was used to generate Fig. 10.26(a).



(a) Response

```
% Response for optimal controller
>> num = [1]; den = [1 1 1]; sys = tf(num,den);
>> numpd = [7 15]; denpd = [1 1 + 7 1 + 15]; syspd = tf(numpd,denpd);
>> numpid = [7 15 2]; denpid = [1 1 + 7 1 + 15 2]; syspid =
tf(numpid,denpid);
>> R = 1; q11 = 1; q22 = 1; p21 = -R + sqrt(R*R*q11);
>> p22 = -R + sqrt(R*R + R*(2*p21 + q22)); p11 = p21 + p22 +
p21*p22/R;
>> P = [p11 p21;p21 p22]; eig(P);
>> t = 0:.05:15; m = 1; b = 1; k = 1; u = 1;
>> A = [0 1; -1 -1]; bv = [0; 1]; cv = [1 0]; d = 0;
>> kv = 1/R*bv.'*P; Af = A-bv*kv;
>> sysop = ss(Af,bv,cv,0)
>> step(sys,syspd,syspid,sysop,t);
>>%MATLAB program ends
```

(b) MATLAB code

Fig. 10.26 Comparison of optimal controller

10.9 JOINT CONTROLLERS

In this section, what is known in control literature as set-point tracking is presented for the feedback control of a robotic joint with the help of proportional (P), proportional and derivative (PD), and proportional, integral and derivative (PID) gains. Such control is sufficient for robots without fast motion, especially in robots with large gear reduction between the actuators and links. The control laws are derived based on the joint model presented in Section 10.4.

10.9.1 Proportional (P) Controller of a Joint

For the control of the robotic joint shown in Fig. 10.13 whose behavior is governed by Eq. (10.54), assume the voltage input is proportional to the error in joint angles of the motor shaft, i.e.,

$$v_a = k_p e \quad (10.103)$$

where, $e \equiv \theta_d - \theta_m$ in which θ_d and θ_m are the desired or set and actual joint angles of the motor shaft, respectively. The scalar k_p is called the proportional control gain.

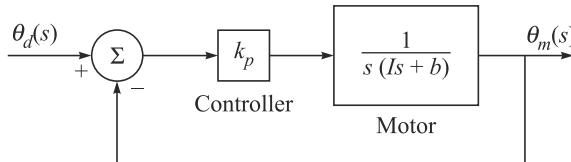


Fig. 10.27 Proportional controller of a robotic joint

Upon substitution of Eq. (10.103) into Eq. (10.54a) its motion in the s -domain is given by

$$s^2 I(s) \theta_m(s) + sb(s) \theta_m(s) = \frac{k_e}{r_a} k_p e(s) - \tau_d(s), \text{ where } e(s) \equiv \theta_d(s) - \theta_m(s) \quad (10.104a)$$

Rearrangement of Eq. (10.104a) yields the following closed-loop equation for the proportional feedback control law:

$$\theta_m(s) = \frac{(k_e/r_a)k_p}{s(Is+b)+k_p} \theta_d(s) - \frac{1}{s(Is+b)+k_p} \tau_d \quad (10.104b)$$

Assuming no disturbance, i.e., $\tau_d=0$, the closed-loop transfer function (TF) relating the actual and desired motor-shaft angles represented by θ_m and θ_d , respectively, can be given by

$$\frac{\theta_m(s)}{\theta_d(s)} = \frac{(k_e/r_a)k_p}{s(Is+b)+k_p} \quad (10.105a)$$

Figure 10.27 shows the closed-loop block diagram for the transfer function given by Eq. (10.105). Using an alternative representation of the TF in Eq. (10.30), i.e., in terms of the natural frequency ω_n and damping ratio ζ of the system, Eq. (10.105a) can also be represented as

$$\frac{\theta_m(s)}{\theta_d(s)} = \frac{(k_e/r_a)\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \text{ where } \omega_n \equiv \sqrt{\frac{k_p}{I}}, \text{ and } \zeta \equiv \frac{b}{2\sqrt{k_p I}} \quad (10.105b)$$

Based on Eq. (10.105b), the system behavior of the system at hand can be controlled by changing the values of ω_n and ζ . Since the system's physical parameters like I , b , k_e , r_a and others cannot be changed for a given system, by varying the values of the controller gain k_p one can make the system behave the way one likes, e.g., critically damped with $\zeta = 1$ which makes the system reach the steady-state at the fastest non-oscillatory manner.

10.9.2 Proportional and Derivative (PD) Controller

In this controller, the gain is chosen as $k_p + k_v s$ instead of k_p as shown in Fig. 10.28, where k_v is called the derivative or velocity gain. For this controller, the input voltage v_a is taken as proportional to the errors in both the angular positions and velocities, namely, $e(s)$ and $\dot{e}(s)$. If $\dot{e}(s)$ is defined by $\dot{e}(s) = \dot{\theta}_d(s) - \dot{\theta}_m(s)$, then the input control law can be expressed as

$$v_a(s) = k_p e(s) + k_v \dot{e}(s) \quad \text{or} \quad v_a(s) = k_p [\theta_d(s) - \theta_m(s)] + k_v s [\theta_d(s) - \theta_m(s)] \quad (10.106)$$

where the time-derivative of any joint angle $\theta(t)$, i.e., $\dot{\theta}(t)$, in s -domain is substituted by its Laplace transform as $\dot{\theta}(t) = s\theta(s)$. Replacing $k_p e(s)$ in Eq. (10.104a) by Eq. (10.106), the new closed-loop equation is given by

$$s^2 I(s) \theta_m(s) + sb(s) \theta_m(s) = \frac{k_e}{r_a} [k_p \{\theta_d(s) - \theta_m(s)\} + k_v s \{\theta_d(s) - \theta_m(s)\}] - \tau_d(s) \quad (10.107)$$

Again if no disturbance is assumed, i.e., $\tau_d = 0$, the modified TF is given by

$$\frac{\theta_m(s)}{\theta_d(s)} = \frac{k_v s + (k_e/r_a) k_p}{Is^2 + (b + k_v)s + (k_e/r_a) k_p} = \frac{k_v s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (10.108a)$$

where the definition of the natural frequency ω_n is defined same as in Eq. (10.105b), whereas the damping ratio is defined as

$$\zeta \equiv \frac{b + k_v}{2\sqrt{k_p I}} \quad (10.108b)$$

For no desired velocity, i.e., if one desires to attain a given joint angle only, Eq. (10.108a) can be further simplified as

$$\frac{\theta_m(s)}{\theta_d(s)} = \frac{(k_e/r_a) k_p}{Is^2 + (b + k_v)s + (k_e/r_a) k_p} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (10.108c)$$

From Eqs. (10.105b) and (10.108b), one can write the expressions for the gains k_p and k_v in terms of the natural frequency ω_n and the damping ratio ζ as

$$k_p = \omega_n^2 I, \text{ and } k_v = 2\zeta I \omega_n - b = 2\zeta \sqrt{k_p I} - b \quad (10.109)$$

Here, both the natural frequency and damping ratio ω_n and ζ , respectively, are now determined by k_p and k_v . It can be shown that if k_p is increased, the settling time reduces (the system becomes faster), whereas if k_v is increased, the overshoot decreases but the system becomes slower. Hence, by proper choice of values k_p and k_v , one can choose ω_n and ζ to satisfy the requirements of the settling time and overshoot. For critical damping, one can get k_v from Eq. (10.109) as $k_v = 2\sqrt{k_p I} - b$. The controller described above is called the proportional and derivative or PD

controller. It is quite versatile and is extensively used for robot control. However, it is not capable of removing the steady-state error due to backlash or unmodeled friction in the mechanical system, etc., which can be modeled as the disturbance input, i.e., $\tau_d \neq 0$. This can be verified from Fig. 10.28, where the angle θ_m is first obtained as

$$\theta_m(s) = \frac{k_v s + (k_e/r_a) k_p}{P(s)} \theta_d(s) - \frac{\tau_d(s)}{P(s)} \quad (10.110)$$

where the characteristic polynomial $P(s)$ is given by $P(s) \equiv Is^2 + (b + k_v)s + (k_e/r_a)k_p$. The error e can then be obtained as

$$e(s) = \theta_d(s) - \theta_m(s) = \frac{Is^2 + bs}{P(s)} \theta_d(s) + \frac{\tau_d(s)}{P(s)} \quad (10.111a)$$

For a step input and constant disturbance, i.e., when

$$\theta_d(s) = \frac{\theta_d}{s}; \text{ and } \tau_d(s) = \frac{\tau_d}{s} \quad (10.111b)$$

the steady-state error e_{ss} is obtained using the final value theorem (Ogata, 1987). This is given by

$$e_{ss} = \lim_{s \rightarrow 0} se(s) = \frac{\tau_d}{k_p} \quad (10.112)$$

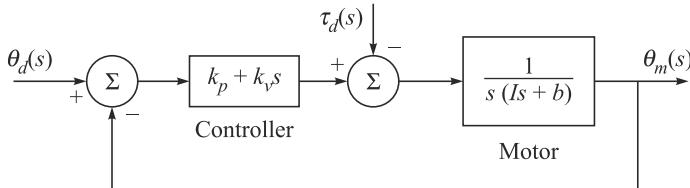


Fig. 10.28 Proportional and derivative controller with disturbance

Since τ_d is proportional to the gear reduction $1/\eta$, the effect of gear reduction is to reduce the nonlinearities in the dynamics. Larger the reduction, smaller is the error. However, at high speed or robots without gear reduction the nonlinearities have a much larger effect on the performance. Considering these nonlinear coupling terms as a disturbance will generally cause large tracking or steady-state error. This error represented by e_{ss} in Eq. (10.112) can be made smaller by making the proportional gain k_p larger. However, the value of k_p cannot be increased to any value as it may excite the structural oscillation. Hence, an alternate controller with an additional integral term is proposed.

10.9.3 Proportional, Integral, and Derivative (PID) Controller

Using an integral term with the PD control law, one obtains the following:

$$f(s) = k_p + k_v s + \frac{k_i}{s} \quad (10.113)$$

where k_i is the integral gain, and Eq. (10.113) is the well-known PID control law, as shown in Fig. 10.29. Such PID controller can make the steady-state error zero. However, the integral term needs to be chosen carefully, as a large k_i can make the

system unstable. Note that with the integral term in the closed-loop system the TF is given by

$$\frac{\theta(s)}{\theta_d(s)} = \frac{k_v s^2 + k_p s + k_i}{P(s)}, \text{ where } P(s) = Is^3 + (b + k_v)s^2 + k_p s + k_i \quad (10.114)$$

which is a third-order system because the highest order of the characteristic polynomial is three. Applying a stability criterion, say, Routh–Hurwitz (Ogata, 1987) to the above polynomial, it follows that the closed-loop system is stable if all the gains k_p , k_v , and k_i are positive, and in addition

$$k_i < \frac{(b + k_v)k_p}{I} \quad (10.115)$$

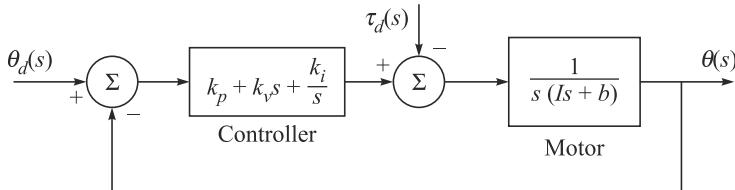


Fig. 10.29 Proportional, integral, and derivative controller

The steady-state error can then be calculated similar to Eqs. (10.110)–(10.112) as

$$\theta(s) = \frac{k_i + k_p s + k_v s^2}{P(s)} \theta_d(s) - \frac{s \tau_d(s)}{P(s)} \quad (10.116a)$$

$$e(s) = \theta_d(s) - \theta(s) = \frac{Is^3 + bs^2}{P(s)} \theta_d(s) + \frac{s \tau_d(s)}{P(s)} \quad (10.116b)$$

$$e_{ss} = \lim_{s \rightarrow 0} s e(s) = \lim_{s \rightarrow 0} \frac{s \tau_d}{P(s)} = 0 \quad (10.116c)$$

where $\theta_d(s)$ and $\tau_d(s)$ are defined in Eq. (10.111b). Note that the PID controller causes the robot to reach the desired target even in the presence of a constant disturbance.

Example 10.23 Response of a Joint Controllers using MATLAB

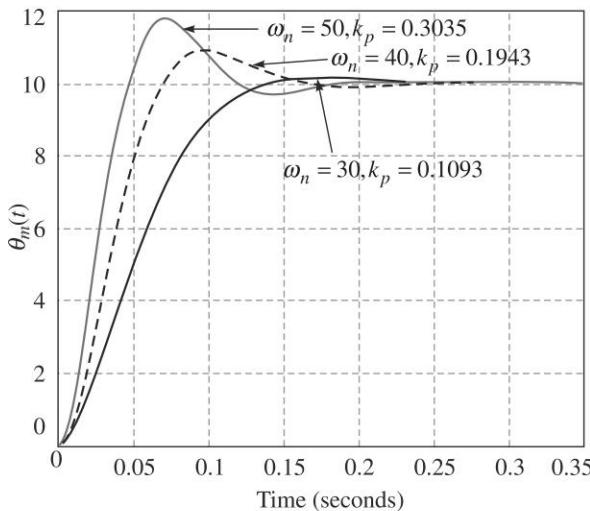
Consider the Maxon motor of Example 10.12 whose physical parameters needed for its control are given in Eq. (10.55). Based on its response given by Fig. 10.16, it is clear that its motor shaft cannot attain a specified joint angle. This is a typical requirement in a robotic application. Hence, a control law needs to be introduced. For that, its transfer function is expressed first as

$$\frac{\theta_m(s)}{\theta_d(s)} = \frac{(k_e/r_a)k_p}{Is^2 + bs + k_p} = \frac{33.52 \times 10^7 k_p}{4.07s^2 + 200s + 10^7 k_p} \quad (10.117)$$

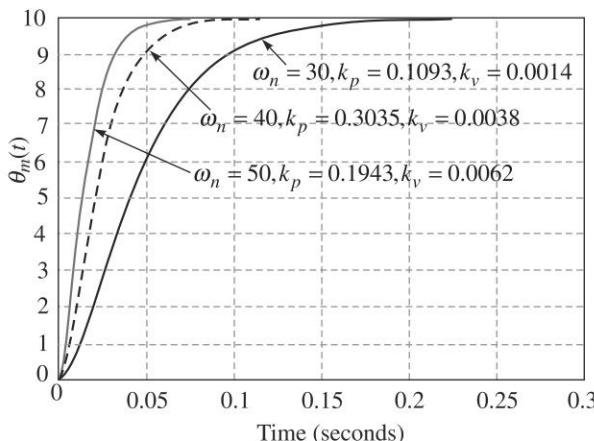
where the values of I , b , k_e and r_a are given in Eq. (10.55) and after it. The value of the gain k_p can then be evaluated for the desired characteristic of damping ratio,

$\zeta = 1$. The MATLAB program shown in Fig. 10.31 can generate the plots of Fig. 10.30.

Note the unity steady-state value for the amplitude that indicates the desired joint angle θ_d . Now, in order to reduce the overshoot of Fig. 10.30(a), a derivative gain is introduced with proportional gain k_p . For the step input of the desired joint angle $\theta_d = 10$, the response is shown in Fig. 10.30(b), which shows the asymptotically reaching to the desired steady-state value. Moreover, the other characteristics of Table 10.1, e.g., decrease of settling time, etc., are quite obvious from Fig. 10.30(b). The PID controller is obtained from the characteristic polynomial of Eq. (10.114). For the value of k_i shown in Fig. 10.30(c) with the step input of $\theta_d = 10$, the response is given, which shows that the steady-state error vanishes after sometime. This is also well-known in the literature of control theories, e.g., Gopal (1997) and others.

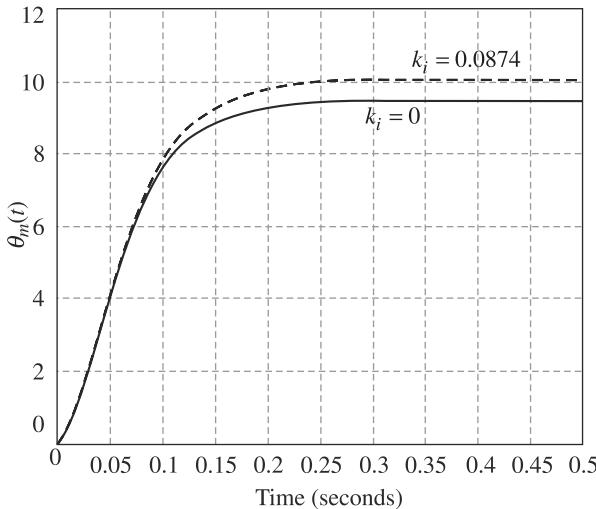


(a) Effect of proportional controller



(b) Effect of proportional and derivative controller with disturbance ($\zeta = 1.0$)

(Contd.)

(c) Effect of PID controller with disturbance ($\omega_n = 30$, $\zeta = 1.0$, $k_p = 0.3035$, $k_v = 0.0038$)**Fig. 10.30** Controllers for a robotic joint

```
>> % MATLAB program to generate response of a robotic joint
>> t = 0.:0.02:5;
>> ten7 = 10000000; i_m = 4.07/ten7; b_m = 12/ten7; km = 5.9/1000; ke = km;
>> la = 0.106/1000; ra = 1.76; i_eff = i_m; tau_mech = 20/1000 ;
>> b_eff = b_m + ke*km/ra; K = km/ra;
>> num = [km/ra]; den_om = [i_eff b_eff]; den = [i_eff b_eff 0];
>> om_n1 = 30; om_n2 = 40; om_n3 = 50;
>> om_s1 = om_n1*om_n1; om_s2 = om_n2*om_n2; om_s3 = om_n3*om_n3;
>> th_d = 10; kp1 = i_eff*om_s1/K; kp2 = i_eff*om_s2/K; kp3 = i_eff*om_s3/K
>> num_p1 = [K/i_eff*kp1]*th_d; den_p1 = [1 b_eff/i_eff K*kp1/i_eff];
>> num_p2 = [K/i_eff*kp2]*th_d; den_p2 = [1 b_eff/i_eff K*kp2/i_eff];
>> num_p3 = [K/i_eff*kp3]*th_d; den_p3 = [1 b_eff/i_eff K*kp3/i_eff];
>> zeta = 1; kv1 = (i_eff*2*zeta*om_n1-b_eff)/K;
>> kv2 = (i_eff*2*zeta*om_n2-b_eff)/K; kv3 = (i_eff*2*zeta*om_n3-b_eff)/K
>> num_pd1 = [K*kp1/i_eff]*th_d; den_pd1 = [1 (b_eff + K*kv1)/i_eff K*kp1/i_eff];
>> num_pd2 = [K*kp2/i_eff]*th_d; den_pd2 = [1 (b_eff + K*kv2)/i_eff K*kp2/i_eff];
>> num_pd3 = [K*kp3/i_eff]*th_d; den_pd3 = [1 (b_eff + K*kv3)/i_eff K*kp3/i_eff];
>> kil = (b_eff + K*kv1)*kp1/(i_eff*75)
>> num_pid1 = [kp1 kil]*K/i_eff*th_d;
>> den_pid1 = [1 (b_eff + K*kv1)/i_eff K*kp1/i_eff K*kil/i_eff];
>> di = 500; num_di = [1]*di; num_dii = [1 0]*di ;
>> sys_di1 = tf(num_di,den_pd1);sys_di2 = tf(num_di,den_pd2);
>> sys_di3 = tf(num_di,den_pd3);
>> sys_dii1 = tf(num_dii,den_pid1);
>> sys_p1 = tf(num_p1,den_p1);sys_p2 = tf(num_p2,den_p2);sys_p3 =
tf(num_p3,den_p3);
```

(Contd.)

```

>> sys_pd1 = tf(num_pd1,den_pd1); sys_pd2 = tf(num_pd2,den_pd2);
>> sys_pd3 = tf(num_pd3,den_pd3);
>> sys_pid1 = tf(num_pid1,den_pid1);
>> [ypd1,t] = step(sys_pd1,t); [ypd2,t] = step(sys_pd2,t);
>> [ypd3,t] = step(sys_pd3,t);
>> [di1,t] = step (sys_di1,t); [di2,t] = step (sys_di2,t);
>> [di3,t] = step (sys_di3,t);
>> [ypid1,t] = step(sys_pid1,t); [diil1,t] = step(sys_diil1,t);
>> step(sys_p1,sys_p2,sys_p3); grid on;
>> figure(2); step(sys_pd1,sys_pd2,sys_pd3); grid on;
>> figure (3); plot (t,ypd1-di1,t,ypid1-diil1);
>> title('Effect of PID control');grid on;
>>% MATLAB program ends

```

Fig. 10.31 MATLAB Program**Example 10.24 Stability of Single-DOF Planar Arm with PD Controller**

Assume that the single-DOF arm of Example 10.14 is now under PD control, i.e., where the actuator output is given by $\tau = -k_p x_1 - k_v x_2$. The terms k_p and k_v are the proportional and derivative gains, respectively, which are always positive. Let us now define the Lyapunov function as

$$v = \frac{1}{2} \frac{ma^2}{3} x_2^2 + mg \frac{a}{2} (1 - \cos x_1) + \frac{1}{2} k_p x_1^2 \quad (10.118)$$

Differentiating Eq. (10.118) with respect to time, one gets

$$\dot{v} = \frac{ma^2}{3} x_2 \dot{x}_2 + mg \frac{a}{2} \sin x_1 \dot{x}_2 + k_p x_1 \dot{x}_1 \quad (10.119a)$$

Substituting for \dot{x}_2 from Eq. (10.41) in Eq. (10.119a), the expression for \dot{v} is obtained as

$$\dot{v} = -k_v x_2^2 \quad (10.119b)$$

which guarantees asymptotic stability since $k_v > 0$.

SUMMARY

In this chapter, concepts of control in general and linear motion control for robots in particular are introduced. The topics are introduced first with a simple example of a moving block. Several approaches like transfer function, state-feedback, etc. to represent a dynamic system are presented. Model of a robotic joint is presented, along with its different control strategies like P, PD, and PID.

EXERCISES

- 10.1** For the motion of the block shown in Fig. 10.2, if $m = 2$, $b = 6$, $k = 4$, and is initially at rest find its motion when released from the position $x = 1$.
- 10.2** What are the poles of a control system?
- 10.3** For a second-order system, define overshoot and time constant.
- 10.4** What is PID control? Why is the integral term important?

- 10.5** Given a joint drive system that consists of a dc servomotor with total inertia of 0.02 kg-m^2 , bearing friction of 0.5 N/s , and a gearbox with gear ratio of 32. The link inertia is 5 kg-m^2 and the link bearing friction is 2 N/s . Determine the effective inertia and effective damping for the joint. Assume that all shafts are rigid and massless.

- 10.6** Transfer function (TF) of a system is given as

$$G(s) = \frac{0.2}{0.1s^2 + 0.6s + 1} \quad (10.120)$$

Find its natural frequency and damping ratio.

- 10.7** Write the state-space representation of the problem in Exercise 10.1.

- 10.8** Find the TF from the state-space representation of Exercise 10.6.

- 10.9** What is optimal control? Write the matrix Riccati equation.

- 10.10** How can one find state feedback gains?

MATLAB BASED EXERCISES

- 10.11** For the TF defined in Exercise 10.6, obtain the time response of the system for a unit step input.

- 10.12** Obtain the time response $y(t)$ for a unit step input if the Laplace transfer function of a linear second-order system $G(s)$ is given by

$$G(s) = \frac{16}{s^2 + 14s + 49} \quad (10.121)$$

What is type of damping?

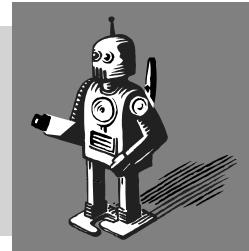
- 10.13** Convert Eq. (10.121) into state-space representation.

- 10.14** Use Simulink to model Eq. (10.121). Generate response for a step input.

- 10.15** Find the steady-state errors for all the controllers of Example 10.23.

11

Nonlinear and Force Controls



In the previous chapter, the control or feedback control was explained with respect to the systems whose dynamics can be represented by a set of Linear Time-Invariant (LTI) differential equations. In short, they are referred as LTI systems. Some of the above systems are not truly linear, rather nonlinear. For example, the one-link arm of Example 10.10 has the nonlinear term “ $\sin \theta$ ” which can be linearized with the assumption that the system variables vary only sufficiently small about an operating point. For example, if one is interested to know the behavior of the link in Example 10.10 for the small variations of θ (say, within 5° – 10°) about $\theta = 0$, it can be quite realistically assumed $\sin \theta \approx \theta$. The most fundamental property of a linear system is the *principle of superposition*, which guarantees that if a system is tested to behave satisfactorily against a test signal, e.g., step or sinusoidal, it will also behave satisfactorily against all other signals which may arise in real situations. The amplitude of input signal has no influence on the outputs as it is only scaled with no change in the basic response characteristics. For example, the steady-state response of a stable linear system for a sinusoidal input is the sinusoidal of same frequency with phase angle shifted.

When the linear assumptions are not valid, nonlinear differential equations must be used to describe the system’s behavior, as it is the case even for a simple one-link robot arm of Example 10.10. Two- or more-degrees-of-freedom robot manipulators are, of course, governed by a set of nonlinear differential equations given in Chapters 8 and 9. Nonlinearities in a given system can be broadly classified into two categories, namely, *inherent* and *intentional*. Inherent nonlinearities are those which correspond to the imperfections with respect to the linear model of a physical system at hand. It would be much simpler if these nonlinearities did not exist. For a robotic system, these are due to the nature of the system architecture, i.e., the links coupled with the joints. Also, the gear backlash, stick-slip motion, etc., could be other reasons for inherent nonlinearities. On the other hand, intentional nonlinearities are those which are deliberately inserted into the system to reduce the costs while achieving the satisfactory performance. An example of such nonlinearity would be on-off controller of a robot. Some commonly encountered nonlinear phenomena in a control system could be due signal saturation, dead zone, preload, and others.

As nonlinearities exist in many physical systems, there were attempts to develop analytical tools for control of such systems, and predict their stabilities. One such analytical tool is Describing Function Analysis, which is similar to frequency-

response method for linear systems. Refer to textbooks on nonlinear control, e.g., Slotine and Li (1991), Gopal (1997), and others. It is an approximate method but adequately accurate in many cases. In fact, the field of nonlinear control theory is very large. It is almost impossible to cover all in the present textbook on robotics. Hence, a few relevant one, e.g., computed-torque, feedforward, etc., are taken up here. First, the concepts are introduced using the simple LTI system of the moving block before proceeding to develop the controls for the nonlinear robots.

11.1 CONTROL OF A MOVING BLOCK

In preparation for designing control laws for more complex multi-degrees-of-freedom robots, consider a slightly different controller structure for the simple problem of a moving block shown in Fig. 10.20 or Fig. 10.21. In this method, partition of the controller is done into a model-based portion and a servo portion. The result is that the system's parameters (i.e., m , b and k) appear only in the model-based portion, and the servo portion is independent of these parameters. This approach is known as *control law partitioning*. In some cases, e.g., in rigid manipulators, this is same as *inverse dynamics control*.

What is Nonlinear?

The coefficients of the differential equations representing the robot dynamics are functions of the control variables and their derivatives. In case of linear systems, these coefficients are constants.

11.1.1 Position Control of the Block

Consider the open-loop equation of motion for the moving block system shown in Fig. 10.2. This is given by Eq. (11.1) where the model-based portion of the control law will make use of the knowledge of m , b , and k . This portion of the control law is set up such that it *reduces the system so that it appears to have a unit mass*. The second part of the control law makes use of feedback to modify the behavior of the apparent unit mass system. Since the model-based portion of the control law has the effect of making the system appear as a unit mass, the design of the servo portion is very simple. The gains are chosen to control a system composed of a single unit mass only with no friction and no stiffness. The model-based portion of the control appears in the control law as

$$f = (m)u + (c\dot{x} + kx) \quad (11.1)$$

where the terms within “()” are taken same as in the original model so that the *new input u* to the system makes the *system appears to be a unit mass*. With this structure of control law the closed-loop system equation using Eqs. (10.1) and (11.1) is as follows:

$$m\ddot{x} + b\dot{x} + kx = mu + b\dot{x} + kx \quad (11.2a)$$

Clearly, the following new linear system of unit mass is obtained:

$$\ddot{x} = u \quad (11.2b)$$

which is the equation of motion for a unit mass. Since the control input u of Eq. (11.1) has the unit of acceleration, as also evident from Eq. (11.2b), the control law computation of Eq. (11.1) for the given values of m , c and k is actually the

inverse dynamics calculation defined in Chapters 8 and 9. Hence, the control law of Eq. (11.1) is called *inverse dynamics* or *computed torque* control. Note that Eq. (11.2b) is a simple linear second-order system as treated in Chapter 10. One of the proposed control law for regulation control is then given by

$$u = -k_p x - k_v \dot{x} \quad (11.3)$$

Combination of this control law with Eq. (11.2b) yields

$$\ddot{x} + k_v \ddot{x} + k_p x = 0 \quad (11.4)$$

which is independent of the system parameters, m , b or k . Using Eq. (11.4), the setting of the control gains is simple. It was proposed after Eq. (10.78c). Based on critical damping, i.e., $\zeta = 1$,

$$k_v = 2\sqrt{k_p} \quad (11.5)$$

Figure 11.1 shows the block diagram for the partitioned controller used to control the system of moving block shown in Fig. 10.2.

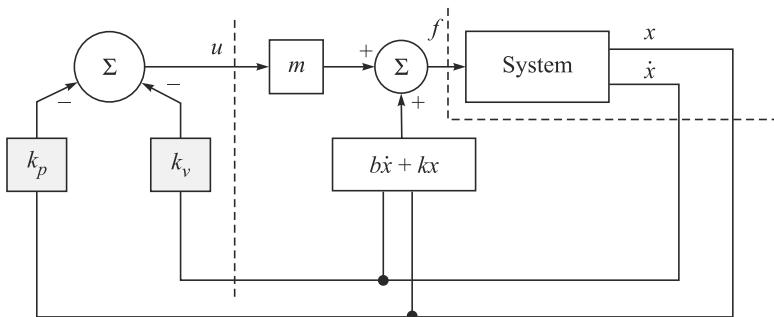


Fig 11.1 A closed-loop control system with partitioned control

Example 11.1 Partitioned Control of the Moving Block

Assume the parameters of the system in Fig. 10.2 as $m = 1$, $b = 1$ and $k = 1$. Gains k_p and k_v are to be found for a position-regulation control law that results in the system being critically damped with a closed-loop stiffness of 16.0, i.e., $k_p = 16$. Let us set gain $k_v = 2\sqrt{k_p}$ for critical damping. Hence, $k_v = 8.0$.

11.1.2 Trajectory Control of the Block

Referring to Fig. 10.2, rather than just maintaining this block at a desired location, let us enhance the controller so that the block can be made to follow a trajectory. The trajectory is given by a function of time, say, $x_d(t)$, which specifies the desired position of the block. Assume that the trajectory is smooth (i.e., the first two time derivatives exist) and that the trajectory generator provides x_d , \dot{x}_d and \ddot{x}_d at all times t . The servo error between the desired and actual trajectory is defined by $e = x_d - x$. Accordingly, a servo control law which will then cause the trajectory to be followed is given by

$$u = \ddot{x}_d + k_v \dot{e} + k_p e \quad (11.6)$$

Combining Eqs. (11.2b) and (11.6) yields

$$\ddot{x} = \ddot{x}_d + k_v \dot{e} + k_p e \quad (11.7a)$$

or $\ddot{e} + k_v \dot{e} + k_p e = 0 \quad (11.7b)$

Since this is a second-order differential equation in error for which one can choose the coefficients to obtain any desired response (often critical damping is the choice). Such an equation is in error space since it describes the evolution of errors relative to the desired trajectory. Figure 11.2 shows a block diagram of a trajectory-following controller.

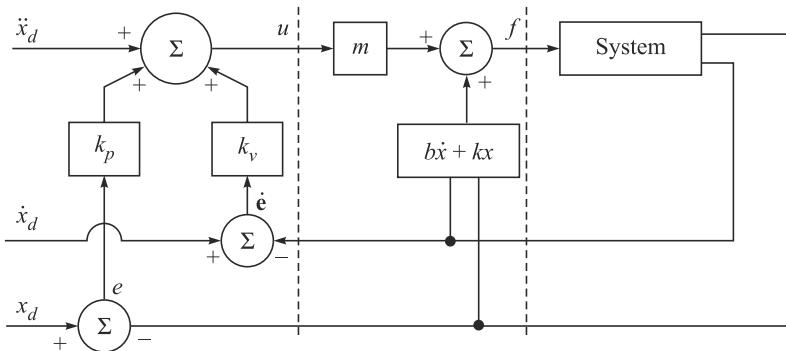


Fig. 11.2 A trajectory-following controller

If the model is perfect (i.e., complete knowledge of m , b , and k is available), no noise, and no initial error, the block will follow the desired trajectory exactly. If there is an initial error, it will be suppressed according to Eq. (11.7b), and thereafter the system will follow the trajectory exactly.

11.2 MULTIVARIABLE ROBOT CONTROL

In this section, the trajectory control of the block will be extended to a multi-degree-of-freedom (DOF) robot. It must be noted that the multi-DOF robot manipulator, unlike a single-DOF robotic joint explained in Section 10.4, is a nonlinear system. However, as most industrial robots have high gearing ratio, their joint dynamics are assumed to be linear and independent. Hence, the linear control scheme developed for the individual joint is applied independently to all the joints of a robot. Such a control scheme may work reasonably well in many industrial applications but does not perform well compared to a nonlinear control scheme. In particular, one cannot guarantee critical damping or the same settling time for all the joints uniformly at every point in the workspace. It is also not possible to analyze and predict the exact performance of a PD or PID scheme for the nonlinear robot. In such control, the assumption is that the system is linear and the inertias of the links are constant. However, it was noticed in Chapter 8 that the inertia matrix is dependent on the robot's configuration. In addition, there are nonlinear Coriolis and centrifugal terms. A linear control scheme when used on a nonlinear robot does not give uniform performance at every point in the workspace of a robot. Hence, an advanced control scheme based on the control law partitioning or other principles (Canudas de Wit et al., 1996) is

required for better performance as compared to the linear control scheme. The major techniques used in nonlinear control are as follows:

1. Trial and Error Similar to linear control system analyses, one attempts to find controller gains which are tested for the robot's stability through analysis and simulations. Experience plays an important role in successful choice of the controller design.

2. Linear Control In this control, a nonlinear dynamic model is linearized about an operating point before the linear control theories are applied for the control of original nonlinear system, as will be explained in Section 11.4.

3. Gain Scheduling In gain scheduling, a number of operating points are chosen over a range. Then, linearization is carried out about each of these points. The result is a linear time-invariant approximation of the original nonlinear systems. Several linear controllers are designed for all the linearized systems. Between the operating points, the parameters of the controllers are then interpolated or scheduled, which acts as a global controller. Gain scheduling is conceptually simpler and practically successful in many applications. One of the drawbacks is that stability is not guaranteed. Moreover, it is computationally expensive as many linear controllers have to be designed.

4. Feedback Linearization The nonlinearity inherent in many robotic applications, particularly, at high-speed operations, is so dominant that the linearization approach explained above, for example, in gain scheduling, cannot meet the system performances like tracking, disturbance rejections, etc. As shown in Fig. 11.3, feedback linearization transforms an original nonlinear system into a linear system of simpler form. Unlike the first-order approximation in the case of linearization about an operating point, where the higher-order terms are ignored, this approach utilizes the feedback to render the given system a linear input-output dynamics.

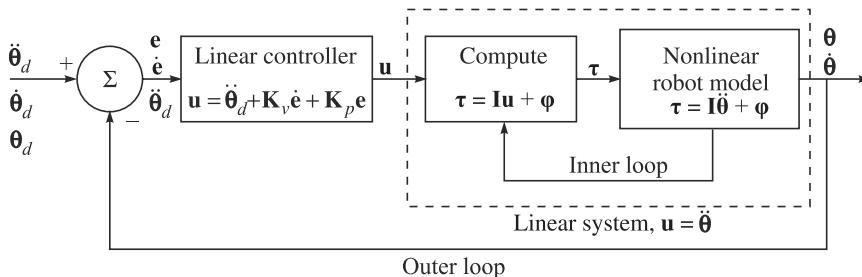


Fig. 11.3 Feedback linearization

On the basis of resultant linear system, the well-developed linear control techniques can be then applied easily. It is a two-stage control method. In the first stage, an inner loop control is constructed which exactly linearizes the nonlinear system after a suitable state-space change of coordinates. In the second stage, an outer-loop control law is designed in terms of the new coordinates so that the traditional control specifications, e.g., tracking, disturbance rejections, etc., are satisfied. In the case of rigid robots, which are the topics of this book, feedback linearization is equivalent to *computed torque* or *inverse dynamics* control. Hence, it will be treated separately in

Section 11.6 due to its wide uses in robotics literature. Feedback linearization, which is same as control-law partitioning of Section 11.1, actually generalizes the concept of inverse dynamics of rigid robots. However, its power will be only exposed when one includes transmission dynamics, elasticity in the gears, shafts, etc. The details are avoided here but available in Spong and Vidyasagar (2004).

5. Robust and Adaptive Controls In a pure model-based nonlinear control, as mentioned in feedback linearization above or computed-torque control of Section 11.6, the control law is designed using a nominal model of the robot. In the presence of uncertainties, e.g., in mass, link length, gear backlash, etc., it is not clear how the robot will perform. The uncertainties can be *structured* or *unstructured*. A structured uncertainty is due to incorrect parameters in link masses, lengths, friction characteristics, etc., of a correct structured dynamic model of a robot, whereas unstructured uncertainties are those which are very difficult to model, e.g., hysteresis, dead zones, etc.

Two major and complimentary approaches to deal with model uncertainties are *robust control* and *adaptive control*. In robust control design, both aspects, i.e., nominal model and some characterization of the model uncertainties, are considered in which the model has a fixed structure. In adaptive control, even though the control architecture is similar, the model is actually updated during operation based on some measure of performance. The basic idea here is to estimate the uncertain parameters on-line and continuously reconfigure the controller in order to adjust for the varying model parameters. Both the approaches are very effective in many practical applications.

11.3 STABILITY OF MULTI-DOF ROBOT

Before different nonlinear controllers are studied, their stability issue is presented in this section using the concepts of Lyapunov's function. It is demonstrated here how a PD controller gives rise to a stable behavior, namely, the error diminishes asymptotically. To this end, consider the dynamic equations of motion for an n -DOF robot derived in Chapters 8 and 9, e.g., Eq. (8.44a), as

$$\ddot{\mathbf{I}}\dot{\boldsymbol{\theta}} + \boldsymbol{\varphi} = \boldsymbol{\tau}, \text{ where } \boldsymbol{\varphi} \equiv \mathbf{h} + \boldsymbol{\gamma} \quad (11.8)$$

In Eq. (11.8), \mathbf{I} is the $n \times n$ Generalized Inertia Matrix (GIM), \mathbf{h} is the n -dimensional Vector of Convective Inertia (VCI) terms, and $\boldsymbol{\gamma}$ is the n -dimensional vector of gravity terms, whereas $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$ are the vectors of joint positions, velocities, and accelerations respectively. For brevity, symbol $\boldsymbol{\varphi}$ is introduced to represent the sum of \mathbf{h} and $\boldsymbol{\gamma}$. A PD control is then proposed as

$$\boldsymbol{\tau} = \mathbf{K}_p \mathbf{e} + \mathbf{K}_v \dot{\mathbf{e}} + \boldsymbol{\gamma} \quad (11.9)$$

where $\mathbf{e} = \boldsymbol{\theta}_d - \boldsymbol{\theta}$ and $\dot{\mathbf{e}} \equiv -\dot{\boldsymbol{\theta}}$ because $\dot{\boldsymbol{\theta}}_d$ vanishes due to the constant desired trajectory. Moreover, \mathbf{K}_p and \mathbf{K}_v are the diagonal gain matrices which are positive definite. With the above control law, the closed-loop system is obtained from Eq. (11.8) as

$$\ddot{\mathbf{I}}\dot{\boldsymbol{\theta}} + \mathbf{h} + \mathbf{K}_p \boldsymbol{\theta} + \mathbf{K}_v \dot{\boldsymbol{\theta}} = \mathbf{K}_p \boldsymbol{\theta}_d \quad (11.10)$$

Consider next the following Lyapunov function:

$$v = \frac{1}{2} \dot{\boldsymbol{\theta}}^T \mathbf{I} \dot{\boldsymbol{\theta}} + \frac{1}{2} \mathbf{e}^T \mathbf{K}_p \mathbf{e} \quad (11.11)$$

The function v in Eq. (11.11) is always positive or zero because the robot's GIM \mathbf{I} and the position gain matrix \mathbf{K}_p are positive definite matrices. Differentiating Eq. (11.11) with respect to time yields

$$\dot{v} = \dot{\theta}^T \mathbf{I} \ddot{\theta} + \frac{1}{2} \dot{\theta}^T \mathbf{I} \dot{\theta} - \mathbf{e}^T \mathbf{K}_p \dot{\theta} \quad (11.12a)$$

Using Eq. (11.10), Eq. (11.12a) is rewritten as

$$\dot{v} = -\dot{\theta}^T \mathbf{K}_v \dot{\theta} + \frac{1}{2} \dot{\theta}^T (\dot{\mathbf{I}} - 2\mathbf{C}) \dot{\theta} \quad (11.12b)$$

where $\dot{\mathbf{I}}$ is time-derivative of \mathbf{I} , and \mathbf{C} is the Matrix of Convective Inertia (MCI) terms that was defined in Eq. (9.10a) as $\mathbf{h} = \mathbf{C}\dot{\theta}$. As shown in Example 8.5, the matrix $\dot{\mathbf{I}} - 2\mathbf{C}$ is skew-symmetric. Hence, $\dot{\theta}^T (\dot{\mathbf{I}} - 2\mathbf{C}) \dot{\theta}$ vanishes. As a result, Eq. (11.12b) leads to

$$\dot{v} = -\dot{\theta}^T \mathbf{K}_v \dot{\theta} \quad (11.12c)$$

which is nonpositive as long as \mathbf{K}_p is positive definite. This, however, does not automatically guarantee the asymptotic stability, but Lasalle and Lefschetz (1961) proved its global asymptotic stability.

11.4 LINEARIZED CONTROL

The easiest way to control a nonlinear robot is to design a linear controller based on the linearization of the dynamic model about an operating point. Such a linear controller guarantees the local stability of the robot, where proportional, derivative and integral or any combination of them can be used as done for the linear systems. Consider a nonlinear control system as given below:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\tau}) \quad (11.13a)$$

where \mathbf{x} is the state vector containing the joint angles and their rates, whereas $\boldsymbol{\tau}$ is the controlling torque. If \mathbf{x}_o is the state obtained due to $\boldsymbol{\tau}_o$ then Eq. (11.13a) can be approximated by its Taylor series expansion as

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_o} \mathbf{x} + \frac{\partial \mathbf{f}}{\partial \boldsymbol{\tau}_o} \boldsymbol{\tau} \quad (11.13b)$$

where $\partial \mathbf{f} / \partial \mathbf{x}_o$ and $\partial \mathbf{f} / \partial \boldsymbol{\tau}_o$ are the partial derivative matrices which are evaluated at the operating point $(\mathbf{x}_o, \boldsymbol{\tau}_o)$. These matrices when compared to the linear state-space representation of Eq. (10.31) are equivalent to state and input matrices, respectively, i.e., $\mathbf{A} \equiv \partial \mathbf{f} / \partial \mathbf{x}_o$ and $\mathbf{B} \equiv \partial \mathbf{f} / \partial \boldsymbol{\tau}_o$.

Example 11.2 Linear Control of One-link Arm

As derived in Chapters 8 and 9, the dynamic model or the equation of motion of the one-link robot arm is given by

$$\tau = \frac{1}{3} ma^2 \ddot{\theta} + \frac{1}{2} mg a \sin \theta \quad (11.14)$$

where m , a , and θ are the mass, link length, and joint angle, respectively, whereas g is the acceleration due to gravity and τ is the controlling torque. Equation (11.14) can be written in the form of Eq. (11.13a) by setting $x_1 = \theta$ and $x_2 = \dot{\theta}$ as

$$\dot{x}_1 = x_2 \text{ and } \dot{x}_2 = \frac{3}{2ma^2} (2\tau - mg a \sin x_1) \quad (11.15)$$

The linearized form of Eq. (11.15) is then expressed in the state-space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{3g}{2a} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{3}{ma^2} \end{bmatrix} \tau \quad (11.16)$$

where $\sin \theta \approx \theta$ was used for small variation of the joint angle θ about any operating point, say, θ_o . Using the state-space representation of Eq. (11.16), one can easily verify that the eigenvalues or the poles of the characteristic equation are zeros as there is no damping. However, one can design a controller to achieve a desired performance based on the methodologies presented in Section 10.6.

11.5 PROPORTIONAL-DERIVATIVE (PD) POSITION CONTROL

Consider a robot which is a Multi-Input-Multi-Output (MIMO) system. Dynamic model of an n -DOF robot can be expressed as given in Eq. (11.8). One of the simplest control laws to achieve a set of desired joint angles θ_d even for a nonlinear MIMO system could be the one used in Eq. (10.106), i.e.,

$$\tau = \mathbf{K}_p \mathbf{e} + \mathbf{K}_v \dot{\mathbf{e}} \quad (11.17)$$

where $\mathbf{e} = \theta_d - \theta$ and \mathbf{K}_p and \mathbf{K}_v are the $n \times n$ constant symmetric positive definite matrices. To analyze the closed-loop behavior of the controlled system, consider the Lyapunov function as the mechanical energy that would be associated with the presence of physical springs and dampers, i.e.,

$$v = \frac{1}{2} [\mathbf{e}^T \mathbf{K}_p \mathbf{e} + \dot{\theta}^T \mathbf{I} \dot{\theta}] \quad (11.18a)$$

The time derivative of Eq. (11.18a) yields

$$\dot{v} = -\dot{\theta}^T \mathbf{K}_p \mathbf{e} + \frac{1}{2} \frac{d}{dt} (\dot{\theta}^T \mathbf{I} \dot{\theta}) \quad (11.18b)$$

where $\dot{\theta}_d = \mathbf{0}$ was used. Moreover, the second term of Eq. (11.18b) is nothing but the time rate of change of kinetic energy which is equal to the *power* given to the robot, i.e., $\dot{\theta}^T \tau$, in which τ is the torque applied to the robot joints. Equation (11.18b) is rewritten as

$$\dot{v} = -\dot{\theta}^T (\mathbf{K}_p \mathbf{e} - \tau) \quad (11.18c)$$

Now substitution of the control law given by Eq. (11.17) into Eq. (11.18c) yields

$$\dot{v} = \dot{\theta}^T \mathbf{K}_v \mathbf{e} = -\dot{\theta}^T \mathbf{K}_v \dot{\theta} \leq 0 \quad (11.18d)$$

Hence, the closed-loop system is asymptotically stable. Note that Eq. (11.18d) was also obtained in Eq. (11.12c) using a slightly different approach.

Example 11.3 Set-point Control of a 2-link Robot Arm Moving Horizontally

Consider the 2-link robot arm shown in Fig. 6.2, which is moving on a horizontal plane with no gravity acting on the links, i.e., $g = 0$. Physical parameters of the robot are taken as follows: Link lengths $a_1 = 1$ m, $a_2 = 0.5$ m, and masses $m_1 = 1$ kg,

$m_2 = 2$ kg. The mass center locations are assumed to be in the middle of the link lengths. The desired joint angles are taken as $\theta_d = [\pi/3 \quad \pi/2]^T$, whereas vanishing joint velocities are assumed, i.e., $\dot{\theta}_d = [0 \quad 0]^T$. For some choices of the 2×2 matrices of the proportional and derivative gains, respectively, i.e., $\mathbf{K}_p = \text{diag.}[15 \quad 15]^T$ and $\mathbf{K}_v = \text{diag.}[4 \quad 4]^T$, the step response of the joint angles, along with the errors, are shown in Fig. 11.4.

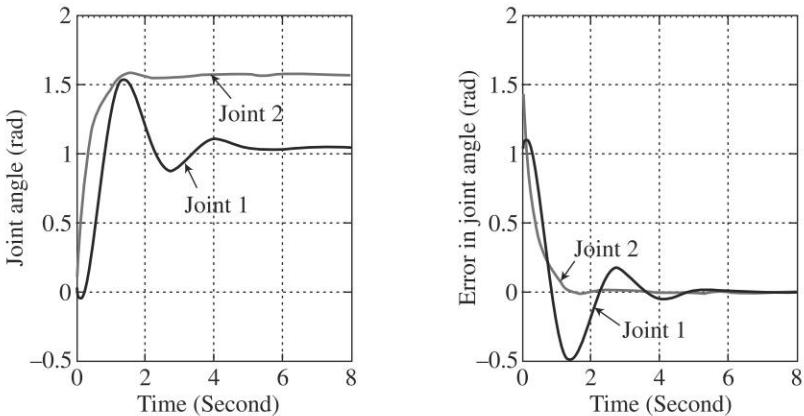


Fig. 11.4 Step response of the 2-link robot arm moving on a horizontal plane

Example 11.4 Set-point Control of a 2-link Robot Arm Moving Vertically

Consider the same 2-link robot arm considered in Example 11.3 but moving vertically, i.e., gravity is acting on the links. It is clear from the step responses and the corresponding errors shown in Fig. 11.5 that the robot is not in a position to attain the desired angles for the same gains. Hence, one needs to investigate further to choose appropriate gains. This is done in the next section.

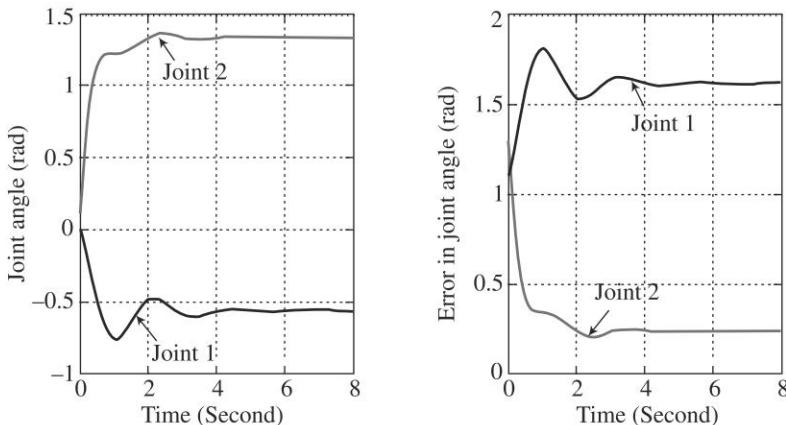


Fig. 11.5 Step response of the 2-link robot arm moving vertically

11.6 COMPUTED-TORQUE (INVERSE DYNAMICS) CONTROL

Here, the nonlinear controller cancels the nonlinearities in the dynamic model of the robot under control using the control-law partitioning approach, as introduced in Section 11.1 for the control of a moving block. As a result, a set of decoupled linear equations of motion are obtained. Such a model-based controller, as mentioned in Section 11.1, is called *computed torque* or *inverse dynamics* control. For the dynamic model of an n -DOF robot given by Eq. (11.8), assume that the output of the controller can be written as

$$\tau = \mathbf{I}\mathbf{u} + \boldsymbol{\phi} \quad (11.19)$$

Substitution of Eq. (11.20) into Eq. (11.8) yields

$$\mathbf{u} = \ddot{\boldsymbol{\theta}} \quad (11.20)$$

The above equation can be viewed as a unit inertia system with a new input \mathbf{u} . In a sense, the nonlinearities in the original nonlinear dynamic model of the system are cancelled. Now, one can apply, for example, a PD controller to the unit inertia system as

$$\mathbf{u} = \ddot{\boldsymbol{\theta}}_d + \mathbf{K}_v \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} \quad (11.21a)$$

where $\mathbf{e} \equiv \boldsymbol{\theta}_d - \boldsymbol{\theta}$ and \mathbf{K}_p and \mathbf{K}_v are the $n \times n$ diagonal matrices. Substituting Eq. (11.21a) into Eq. (11.20), one gets

$$\ddot{\mathbf{e}} + \mathbf{K}_v \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} = \mathbf{0} \quad (11.21b)$$

The above vector equation is the decoupled n scalar error equations that correspond to the n -joints of the robot. Equation (11.21b) is linear and describes the natural behavior of the error dynamics. It can be expressed in a state-space form as $\dot{\mathbf{x}} = \mathbf{Ax}$ where the state matrix \mathbf{A} and the state vector \mathbf{x} are given by

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{K}_p & -\mathbf{K}_v \end{bmatrix} \text{ and } \mathbf{x} \equiv \begin{bmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{bmatrix} \quad (11.21c)$$

From the stability of linear system explained in Section 10.6, the errors will reduce asymptotically to zero if the eigenvalues of the matrix \mathbf{A} or the poles of the corresponding system have negative real part. This is equivalent to choose the gain matrices \mathbf{K}_p and \mathbf{K}_v which play the role of natural frequency and damping, respectively, as $\mathbf{K}_p = \text{diag.}[\omega_n^2]$ and $\mathbf{K}_v = \text{diag.}[2\zeta\omega_n]$, for $\omega_n > 0$ and $0 < \zeta \leq 1$. The resultant system will be underdamped or critically damped at every point in the workspace of the robot. Since \mathbf{K}_p and \mathbf{K}_v are typically taken as diagonal matrices, the response speed of each joint can be controlled independently by changing the diagonal elements of those matrices. Equation (11.21b) is equivalent to a PD feedback control loop for each θ_i so that each component of the error e_i converges to zero with appropriate damping coefficient. As a result, the effect of modeling error and disturbance, if any will be reduced.

Such control is effectively a two-stage or two-loops control system, where one stage or the inner-loop takes care of making the nonlinear system appearing linear system of unit inertia, and the outer-loop or the second stages takes care of the servo compensation. The above proposed control scheme uses feedback linearization, in contrast to the linearization of a nonlinear system where an operating point is chosen

about which the linearization is done. A block diagram of this control is shown in Fig. 11.6. It can be seen that the controller has two parts, namely, (1) an error-driven portion that computes \mathbf{u} based on the PD or PID control law, and (2) a model-based portion where \mathbf{I} , \mathbf{h} and $\boldsymbol{\gamma}$ are computed.

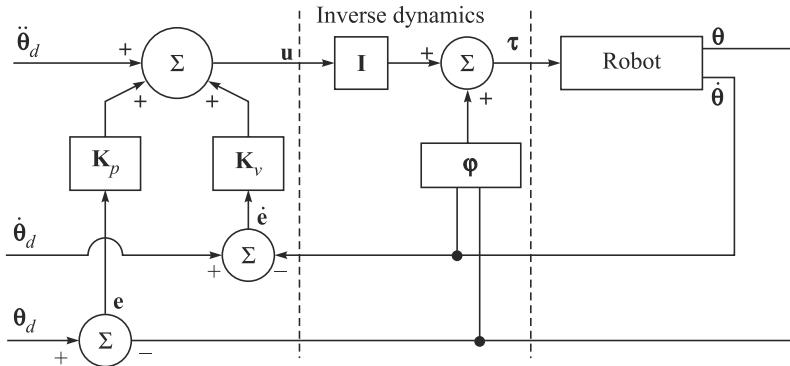


Fig. 11.6 A trajectory following controller of a robot

Example 11.5 Computed-torque Control Law of a Two-link Robot Arm

Assume the desired joint angles and the associated control torque inputs for the two joints of a two-link robot, denoted by θ_d and \mathbf{u} , respectively, are given by

$$\theta_d = [\theta_{d1} \quad \theta_{d2}]^T, \text{ and } \mathbf{u} = [u_1 \quad u_2]^T \quad (11.22)$$

Using Eq. (11.21a) the control laws are given by

$$u_i = \ddot{\theta}_{di} + k_{vi}\dot{\theta}_i + k_{pi}e_i, \text{ where } \dot{\theta}_i \equiv \dot{\theta}_{di} - \dot{\theta}_i \text{ and } e_i \equiv \theta_{di} - \theta_i \quad (11.23a)$$

for $i = 1, 2$. The scalars k_{vi} and k_{pi} are respectively the derivative and proportional gains for the joints 1 and 2. Rewriting Eq. (11.23a) in terms of the natural frequency ω_n and the damping ratio ζ yields

$$u_i = \ddot{\theta}_{di} + 2\zeta\omega_n(\dot{\theta}_{di} - \dot{\theta}_i) + \omega_n^2(\theta_{di} - \theta_i) \quad (11.23b)$$

In writing Eq. (11.23b), it is assumed that control gains for both the joints are same. Now, the control torques τ_i can be given from Eq. (11.20a) by

$$\tau_i = i_{i1}u_1 + i_{i2}u_2 + \varphi_i, \text{ for } i = 1, 2 \quad (11.24)$$

where $\varphi_i \equiv h_i + \gamma_i$, in which the terms h_i and γ_i for the two-link robot are obtained in Chapter 8. The control laws for the outer and inner loops given respectively by Eqs. (11.23b) and (11.24) will be used in the next example.

Example 11.6 Simulation of Computed-torque Control of the 2-link Robot Arm

The two-link robot arm of Examples 11.4–11.5 is taken here again to show if the computed-torque control approach will make the arm reach the desired joint angles. The plots in Fig. 11.7 show that the joints reach their desired angles.

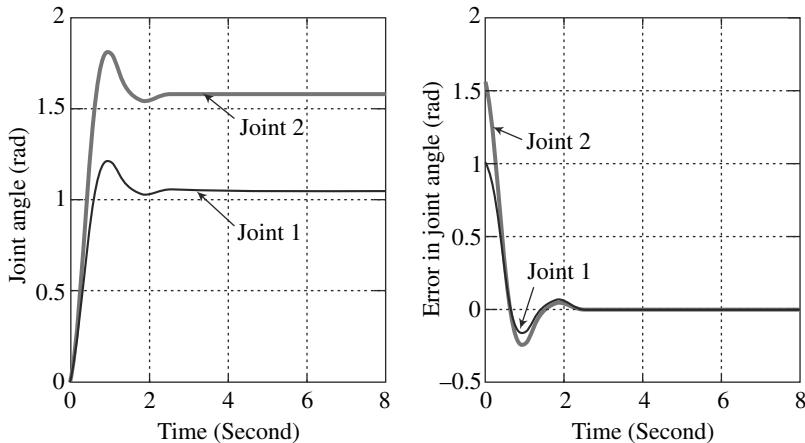


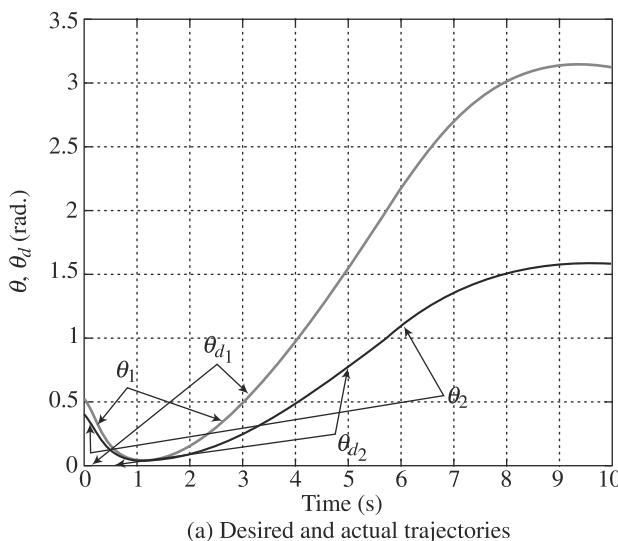
Fig. 11.7 Step response of the 2-link robot arm with computed-torque control

Example 11.7 Computed-torque Controller of the Two-link Robot Arm

Consider the 2-link robot arm whose physical parameters are as follows: Link lengths $a_1 = 1$ m, $a_2 = 1$ m, and masses $m_1 = 1$ kg, $m_2 = 1$ kg. The mass center locations are assumed to be in the middle of the link lengths. Figure 11.8 shows the desired and controlled trajectories for the following proportional and derivative feedback gains respectively.

$$\mathbf{K}_p = \text{diag.}[25, 25], \text{ and } \mathbf{K}_v = \text{diag.}[2\sqrt{25}, 2\sqrt{25}] \quad (11.25)$$

Note that the derivative gains of \mathbf{K}_v were taken to have non-oscillatory response of the joint motions, i.e., for critical damping with $\zeta = 1$. Even if the joints have started initially from different points than desired, finally they reach the desired trajectory, as shown in Fig. 11.8(a), with very minimal errors indicated by Fig. 11.8(b).



(a) Desired and actual trajectories

(Contd.)

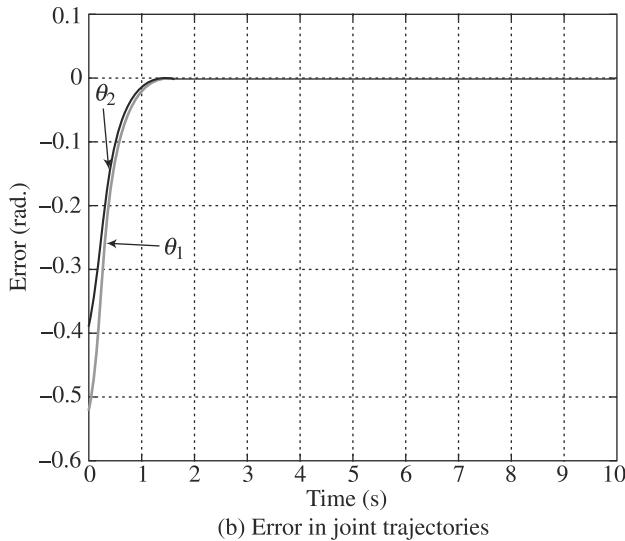


Fig. 11.8 Trajectory control of a two-link robot arm

11.7 FEEDFORWARD CONTROL

The basic feedback control structure is shown in Fig. 10.1, where the disturbances are counteracted by some control laws based on the measurements of the output, namely, the joint positions and velocities of a robotic system. Hence, one of the requirements of such control actions is that the disturbances must show up in the output before the controller can act upon it. Alternatively, if one can measure the disturbances, the performance of the controller can be improved by using the measurements to augment the control signals sent to the robot. This is the principle of the feedforward control for disturbance rejection, as depicted in Fig. 11.9. Note that both feedback and feedforward schemes may be used in the same controller.

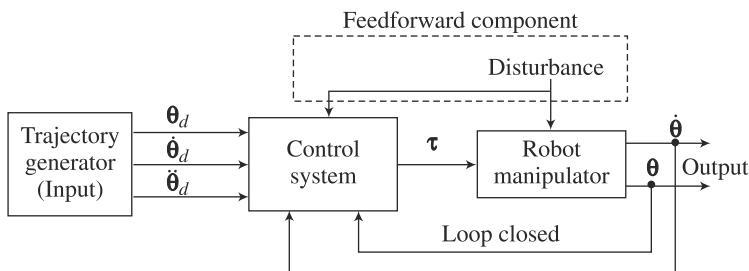


Fig. 11.9 A feedforward control scheme

In the feedforward control of a robotic system, the nonlinear dynamic model, namely, the equations of motion of the robot manipulator governed by Eq. (11.8) can be treated as disturbance. This is in contrast to the computed-torque or inverse-

dynamics control of Fig. 11.3 where the dynamic model is kept inside the servo-loop. In the computed-torque control, the model-based term, i.e., the dynamic model, of the control law is in the servo loop. As a result, the dynamic model of the robot must be computed at the same rate as the desired input θ_d and its time derivatives. If one selects a sample rate of 200 Hz then the dynamic model has to be computed at this rate, which is possible only if the controlling computer can do so. In order to reduce the computing load of the controlling computer and achieve faster motion of the robot under control, the feedforward control approach can be adopted where the model-based control is kept outside the servo loop, as shown in Fig. 11.10. In such a situation, one can have faster inner servo loop, while model-based torques are added at a slower rate.

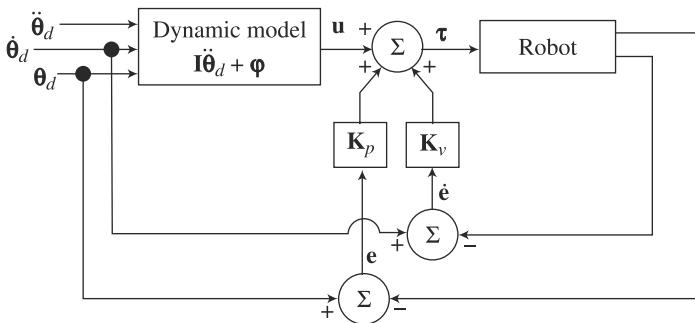


Fig. 11.10 Feedforward controller of a robot

One drawback of the feedforward control is, however, that it cannot completely decouple the control dynamics. This can be shown as follows: From Fig. 11.10, the feedforward control law can be given by

$$\tau = u + K_p e + K_v \dot{e}, \text{ where } u \equiv I(\theta_d) \ddot{\theta}_d + \phi(\theta_d, \dot{\theta}_d) \quad (11.26)$$

In Eq. (11.26), θ_d , $\dot{\theta}_d$, $\ddot{\theta}_d$ are the desired joint motion and their first and second time-derivatives respectively. Substitution of Eq. (11.26) into the dynamic model of Eq. (11.8) yields

$$I(\theta) \ddot{\theta} + \phi(\theta, \dot{\theta}) = I(\theta_d) \ddot{\theta}_d + \phi(\theta_d, \dot{\theta}_d) + K_p e + K_v \dot{e} \quad (11.27a)$$

Assuming $I(\theta_d) \equiv I(\theta)$ and $\phi(\theta_d, \dot{\theta}_d) \equiv \phi(\theta, \dot{\theta})$, Eq. (11.27a) is rewritten as

$$I\ddot{e} + K_v \dot{e} + K_p e = 0 \quad (11.27b)$$

From Eq. (11.27b), it is clear that the first two terms on the right-hand side of Eq. (11.27a) are nothing but the desired computed torque τ_d . Hence, some researchers like Spong and Vidyasagar (2004) refers the control law of Eq. (11.26) as the *method of computed torque*, even though the control law given in Section 11.6, namely, by Eq. (11.21a), is popularly known as computed-torque or inverse-dynamics method. Compared to the latter control, the former, i.e., feedforward control given by Eq. (11.26) do not decouple the control equations completely. In fact, this is also evident from Eq. (11.27b) as the error equations are also coupled through the robot's GIM. As the configuration of the robot changes, the effective closed-loop gain should also change. Since the dynamic model is computed as a function of the desired path

only, which is known in advance, its values could be calculated off-line before the motion begins. That way there is no computational burden on the robot controller, thereby saving the hardware resources substantially or enhancing the servo rate of the robot with a given computing resources. At runtime, the pre-computed torques can be read from the memory. Similarly, if time-varying gains are also calculated they too could be computed beforehand and stored.

Note that feedforward controller compensates the nonlinear terms, namely, the inertia couplings, Coriolis, centrifugal, and gravitational terms arising out of the motion of the robot in a feedforward manner. In the case of no differences in these values, the controller perfectly tracks the trajectory with zero errors. In case of nonzero differences, which is more practical, the tracking errors are also small. This is proven in Example 11.8.

Example 11.8 Disturbance Rejection in a Feedforward Control

Consider the block diagram of Fig. 11.11 showing a system with feedforward control. When disturbance $w = 0$, the block diagram analysis, e.g., in Ogata (1987) or Gopal (1997), evaluates the transfer function (TF) relating the output y and input u as

$$\frac{y}{u} = \frac{C(s)G(s)}{1 + C(s)G(s)H(s)} \quad (11.28a)$$

where $C(s)$, $G(s)$ and $H(s)$ are the TFs of the controller, robot, and feedback respectively. Moreover, if the input $u = 0$, the TF relating the output y with the disturbance w is obtained as

$$\frac{y}{w} = \frac{[1 - F(s)C(s)]G(s)}{1 + C(s)G(s)H(s)} \quad (11.28b)$$

Applying the principle of supervision for a linear system, the output y can now be represented in terms of the input u and disturbance w as

$$y = \frac{C(s)G(s)}{1 + C(s)G(s)H(s)}u + \frac{[1 - F(s)C(s)]G(s)}{1 + C(s)G(s)H(s)}w \quad (11.29)$$

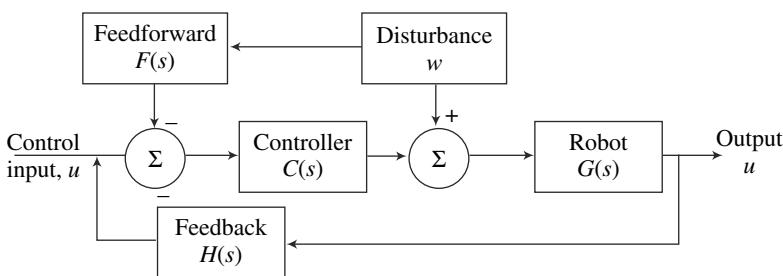


Fig. 11.11 Disturbance in a feedforward control

Note in Eq. (11.29) that the first term is same as the one without any feedforward loop. With the presence of the loop, if $F(s)C(s) = 1$ or $F(s) = 1/C(s)$, the second term vanishes from the right-hand side of Eq. (11.29). Hence, the disturbance is completely rejected. Even if complete rejection is not possible, the effect of rejection is reduced due to the negative sign with $F(s)C(s)$.

11.8 ROBUST CONTROL

In a pure model-based nonlinear control, as mentioned in feedback linearization or computed-torque control, the control law is designed using a nominal model of the robot. In the presence of uncertainties, e.g., in mass, link length, gear backlash, etc., it is not clear how the robot will perform. Issues related to the uncertainties were explained in Item 5 of Section 11.2. Due to uncertainties in the robot parameters, the control law used in the inner-loop of the inverse-dynamics or computed-torque control is actually of the form

$$\tau = \hat{\mathbf{I}}\dot{\mathbf{u}} + \hat{\phi} \quad (11.30a)$$

In Eq. (11.30a), $\hat{\mathbf{I}}$ and $\hat{\phi}$ are the nominal values of the corresponding terms. The uncertainties can be modeled as

$$\Delta\mathbf{I} = \hat{\mathbf{I}} - \mathbf{I} \text{ and } \Delta\phi = \hat{\phi} - \phi \quad (11.30b)$$

Substitution of computed-torque control law in Eq. (11.30a) into the dynamic model of the robot given by Eq. (11.8) yields

$$\ddot{\theta} + \phi = \hat{\mathbf{I}}\dot{\mathbf{u}} + \hat{\phi} \Rightarrow \ddot{\theta} = \mathbf{I}^{-1}(\hat{\mathbf{I}}\dot{\mathbf{u}} + \Delta\phi) \quad (11.31a)$$

Using next the outer-loop control law for the inverse-dynamics control, Eq. (11.21a), the closed-loop dynamics of Eq. (11.31a) leads to

$$\ddot{\mathbf{e}} + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e} = \eta, \text{ where } \eta \equiv (\mathbf{I}^{-1}\hat{\mathbf{I}} - \mathbf{I})\ddot{\theta}_d + \mathbf{I}^{-1}(\Delta\phi) \quad (11.31b)$$

The term η is called *uncertainty*. Unlike error dynamics in Eq. (11.21b), Eq. (11.31b) is a coupled nonlinear system as η is a nonlinear function of \mathbf{e} . As a result, one cannot confirm if the system of errors given by Eq. (11.31b) is stable or not irrespective of whatever may be the values of the gains in matrices \mathbf{K}_v and \mathbf{K}_p . It is, therefore, essential to design a suitable control law for \mathbf{u} so that the system error vanishes. This can be done based on some assumptions on the range of variation of the uncertainty η even if it is not known (Spong and Vidyasagar, 2004; Siciliano et al., 2011).

11.9 ADAPTIVE CONTROL

For an adaptive controller, it is important to note the fact that the dynamic model in the form of Eq. (11.8) can be expressed as a linear combination of the robot's physical parameters, which was pointed out in Chapter 8 and illustrated in Example 8.12. For the inaccurate values of the parameters, the dynamic model of Eq. (11.8) is expressed as

$$\hat{\mathbf{I}}\ddot{\theta} + \hat{\phi} = \mathbf{Y}\hat{\mathbf{p}}' \quad (11.32)$$

where $\hat{\mathbf{I}}$ and $\hat{\phi}$ are defined after Eq. (11.30a), and the matrix \mathbf{Y} is a function of joint angles, rates, and accelerations, whereas the vector $\hat{\mathbf{p}}'$ contains the constant inaccurate parameters. Such representation which is not unique is the central to the development of adaptive control laws. Using the same inner-loop control law as used in robust control, Eq. (11.32) is rewritten as

$$\hat{\mathbf{I}}\ddot{\theta} + \phi = \hat{\mathbf{I}}(\ddot{\theta}_d + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e}) + \hat{\phi} \quad (11.33a)$$

Subtracting $\hat{\mathbf{I}}\ddot{\theta}$ from both sides of Eq. (11.33a), one obtains

$$\hat{\mathbf{I}}(\ddot{\mathbf{e}} + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e}) = \tilde{\mathbf{I}}\ddot{\theta} + \tilde{\phi} \quad (11.33b)$$

where $\tilde{\mathbf{I}} \equiv \mathbf{I} - \hat{\mathbf{I}}$ and $\tilde{\boldsymbol{\varphi}} \equiv \boldsymbol{\varphi} - \hat{\boldsymbol{\varphi}}$. Subtracting Eq. (11.32) from the linear form of the dynamic equations given by Eq. (8.67), i.e., $\mathbf{I}\ddot{\boldsymbol{\theta}} + \boldsymbol{\varphi} = \mathbf{Y}\dot{\mathbf{p}}'$, one can also write the following:

$$\tilde{\mathbf{I}}\ddot{\boldsymbol{\theta}} + \tilde{\boldsymbol{\varphi}} = \mathbf{Y}\tilde{\mathbf{p}}, \text{ where } \tilde{\mathbf{p}} \equiv \dot{\mathbf{p}}' - \hat{\mathbf{p}} \quad (11.33c)$$

Assuming $\tilde{\mathbf{I}}$ is invertible, the error dynamics can then be expressed as

$$\ddot{\mathbf{e}} + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e} = \boldsymbol{\Gamma}\tilde{\mathbf{p}}, \text{ where } \boldsymbol{\Gamma} \equiv \hat{\mathbf{I}}^{-1}\mathbf{Y} \quad (11.33d)$$

Equation (11.33d) is now a linear system which can be expressed in state-space form, i.e., in the form of $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$, where the state matrix \mathbf{A} and state vector \mathbf{x} have the same structures as in Eq. (11.21c), whereas the matrix \mathbf{B} and vector \mathbf{u} are given by $\mathbf{B} = [\mathbf{O} \ \mathbf{1}]^T - \mathbf{O}$ and $\mathbf{1}$ being the zero and identity matrices of compatible sizes, respectively,—and $\mathbf{u} = \boldsymbol{\Gamma}\tilde{\mathbf{p}}$. Based on the concept of Lyapunov's stability criterion, one can then obtain parametric adaptive law in the following form (Spong and Vidyasagar, 2004):

$$\dot{\tilde{\mathbf{p}}} = -\mathbf{K}_{\tilde{p}}^{-1}\boldsymbol{\Gamma}^T\mathbf{B}^T\mathbf{P}\mathbf{x}$$

where the matrices \mathbf{P} and $\mathbf{K}_{\tilde{p}}$ are symmetric positive definite matrices corresponding to the Lyapunov function, $v = \mathbf{x}^T\mathbf{P}\mathbf{x} + \tilde{\mathbf{p}}^T\mathbf{K}_{\tilde{p}}\tilde{\mathbf{p}}$.

11.10 CARTESIAN CONTROL

In the control schemes presented above, the desired trajectory was assumed to be time histories of joint positions, velocities, and accelerations, and the corresponding effort was joint torques. In a robot, however, it is the robot end-effector which actually performs the work or interacts with its environment. Hence, it is more natural to specify the motion of the robot in terms of the position and orientation of the end-effector, its velocities and accelerations, and/or the corresponding forces and moments acting on it. Note here that the end-effector motion of a robot is achieved by controlling a set of joint motions, as also pointed out in Chapter 5, because that is the way the robots are designed. Hence, the joint motion specifications are used for controlling a robot. In case one wishes to specify the Cartesian motions, along with the forces and moments acting on the end-effector, the controller should have the ability to do the necessary transformation from the Cartesian space of the end-effector to the joint space so that appropriate signals are communicated to the joint actuators. For Cartesian control, the following three approaches can be typically adopted.

11.10.1 Resolved Motion Control

Resolved motion means that the joint motions are combined and resolved into separately controllable end-effector motion along the three Cartesian axes. Such control enables a user to specify the direction, speed, and acceleration, if required, along any arbitrary path of the end-effector. It simplifies the specification of the sequence of motions for the completion of a task because a human operator is usually more adapted to the Cartesian coordinates than the robot's joint coordinates. Based on what is controlled, resolved motion control is accordingly classified as *resolved-rate* or *resolve-acceleration* or *resolved-force* control. Given the position, orientation, and their first two derivatives of the end-effector of a robot, one approach is to use the inverse kinematics for position, velocity, and acceleration to obtain the joint position, velocity, and acceleration, respectively, and use one of the joint control approaches

presented in the previous sections. Inverse kinematic expressions can be represented as

$$\boldsymbol{\theta} = \mathbf{f}^{-1}(\mathbf{Q}, \mathbf{p}), \dot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\mathbf{t}_e, \text{ and } \ddot{\boldsymbol{\theta}} = \mathbf{J}^{-1}(\dot{\mathbf{t}}_e - \dot{\mathbf{J}}\mathbf{t}_e) \quad (11.34)$$

where $\mathbf{f}^{-1}(\mathbf{Q}, \mathbf{p})$ — \mathbf{Q} and \mathbf{p} being the 3×3 orientation matrix and the 3-dimensional position vector of the robot's end-effector—denotes the solutions for the joint angles $\boldsymbol{\theta}$ using nonlinear kinematic relationships, whereas \mathbf{J} and \mathbf{t}_e are the Jacobian matrix and the end-effector's twist of the robot under study. This is depicted in Fig. 11.12. The trajectory conversions at all levels, i.e., position, velocity, and acceleration, are quite complex and time consuming. Hence, only the inverse kinematics for position is performed to obtain the desired joint angles $\boldsymbol{\theta}_d$ which are then numerically differentiated to obtain the joint velocities and accelerations. Such numerical differentiations tend to amplify the noise leading to other difficulties in control.

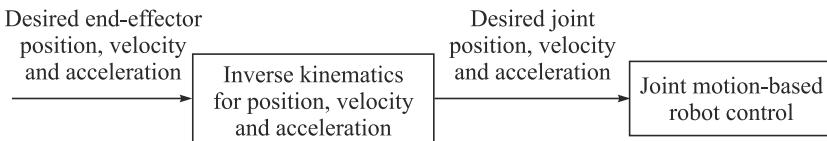


Fig. 11.12 Cartesian control using inverse kinematics and joint motion controller

An alternate way to avoid the above difficulties of numerical noise, etc., is to use a forward kinematics algorithm, as presented in Chapter 6, to convert the actual joint trajectories available from the joint sensors to the end-effector trajectories and apply a control law to obtain forces and moments to be applied on the end-effector.

1. Resolved-Motion Position or Inverse-Jacobian Control In this scheme, the error in end-effector configuration, i.e., orientation and positions, denoted with $\Delta\mathbf{x}$, are transformed into a small change in joint angles, i.e., $\Delta\boldsymbol{\theta}$, to accommodate the changes, i.e.,

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{-1}\Delta\mathbf{x}, \Delta\mathbf{x} \equiv \mathbf{x}_d - \mathbf{x} \quad (11.35)$$

where \mathbf{J} is the well-known velocity Jacobian matrix derived in Chapter 6. Moreover, \mathbf{x} represents the end-effector's configuration consisting of an appropriate orientation representation, say, using Euler angles defined in Chapter 5, and the position. The subscript d denotes the desired Cartesian configuration. The controlling torque can then be obtained either using any combination of proportional (P), derivative (D), and integral (I) controllers or using the dynamics model of the robot given in Chapters 8 or 9. The scheme is shown in Fig. 11.13.

2. Resolved-Motion Force or Transpose-Jacobian Control Since computing the Jacobian of a robot is quite expensive computationally, and which does not exist at singular configuration points, another approach based on the transpose of the Jacobian, i.e., $\boldsymbol{\tau} = \mathbf{J}^T \mathbf{w}_e$, is proposed, where $\mathbf{w}_e \equiv [\mathbf{n}_e^T \quad \mathbf{f}_e^T]^T$ is the wrench vector consisting of applied moment and force on the end-effector denoted by \mathbf{n}_e and \mathbf{f}_e , respectively. In this scheme, the inverse-Jacobian block of Fig. 11.13(a) is replaced with the transpose of the Jacobian shown in Fig. 11.13(b).

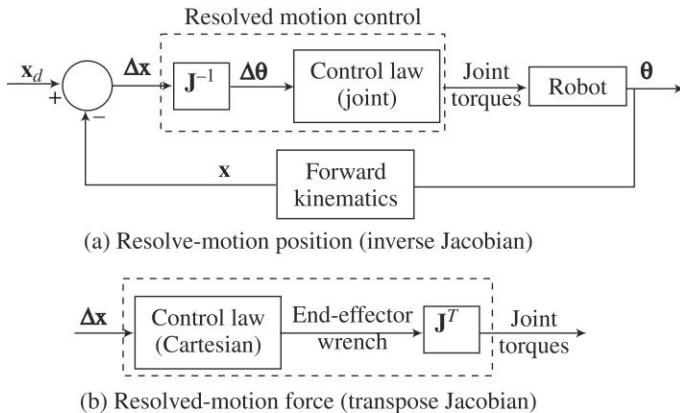


Fig. 11.13 Resolved-motion (Cartesian) control schemes

11.10.2 Feedback Linearization in Cartesian Space

It has been observed in Section 11.6 that the nonlinear terms are cancelled to convert the system to a linear one of unit inertia. They were in terms of joint variables. Here, the same will be presented in terms of the Cartesian variables which in many instances are more desirable from robot's task point of view. As derived in Chapter 6 and shown in Eq. (11.34), the joint variables are available in terms of Cartesian space variables. Recalling the inverse dynamics or computed-torque control law given by Eq. (11.19), one can write a similar control law using Eq. (11.34) as

$$\tau = \mathbf{I}\mathbf{J}^{-1}(\mathbf{u}_e - \dot{\mathbf{J}}\mathbf{t}_e) + \boldsymbol{\varphi} \quad (11.36a)$$

Substitution of Eq. (11.36a) into the dynamic equations of motion of the robot, Eq. (11.8) leaves the closed-loop system as

$$\dot{\mathbf{t}}_e = \mathbf{u}_e \quad (11.36b)$$

which is linear and decoupled with respect to the Cartesian variables. Hence, as in the case of Eq. (11.21a), a servo controller to the linear system of Eq. (11.36b) can be introduced as

$$\mathbf{u}_e = \dot{\mathbf{t}}_e + \mathbf{K}_v \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} \quad (11.36c)$$

where \mathbf{e} and $\dot{\mathbf{e}}$ are the errors in terms of the Cartesian configurations, i.e., orientation \mathbf{Q} and position \mathbf{p} , and its time-rate, respectively, whereas \mathbf{K}_v and \mathbf{K}_p are their corresponding gain matrices which are typically taken as diagonals.

Example 11.9 Feedback Linearization of a Two-link Robot Arm in Cartesian Space

The Jacobian matrix and its derivative relating the end-effector twist and twist-rates, i.e., Cartesian velocities and accelerations, for the two-link robot arm are reproduced from Examples 6.24 and 6.30, respectively, as

$$\mathbf{J} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \text{ and } \dot{\mathbf{J}} \equiv \begin{bmatrix} -a_1 c_1 \dot{\theta}_1 - a_2 c_{12} \dot{\theta}_{12} & -a_2 c_{12} \dot{\theta}_{12} \\ -a_1 s_1 \dot{\theta}_1 - a_2 s_{12} \dot{\theta}_{12} & -a_2 s_{12} \dot{\theta}_{12} \end{bmatrix} \quad (11.37)$$

where $s_1 \equiv \sin \theta_1$, $c_1 \equiv \cos \theta_1$, $s_{12} \equiv \sin(\theta_1 + \theta_2)$, $c_{12} \equiv \cos(\theta_1 + \theta_2)$, and $\dot{\theta}_{12} \equiv \dot{\theta}_1 + \dot{\theta}_2$. Moreover, the GIM \mathbf{I} and the vector $\boldsymbol{\varphi}$ required for the control are defined as

$$\mathbf{I} = \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix}, \quad \mathbf{t}_e \equiv \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} \text{ and } \boldsymbol{\varphi} \equiv \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} \quad (11.38a)$$

From Chapter 8,

$$\begin{aligned} i_{11} &\equiv \frac{m_1}{4} a_1^2 + I_{1,ZZ} + m_2 \left(a_1^2 + \frac{a_2^2}{4} + a_1 a_2 c_2 \right) + I_{2,ZZ} \\ i_{21} (\equiv i_{12}) &= m_2 \left(\frac{a_2^2}{4} + \frac{a_1 a_2 c_2}{2} \right) + I_{2,ZZ} \\ i_{22} &= m_2 \frac{a_2^2}{4} + I_{2,ZZ} \end{aligned} \quad (11.38b)$$

and

$$\begin{aligned} \varphi_1 &\equiv -m_2 a_1 a_2 s_2 \dot{\theta}_2 \left(\dot{\theta}_1 + \frac{1}{2} \dot{\theta}_2 \right) + m_1 g \frac{a_1}{2} c_1 + m_2 g \left(a_1 c_1 + \frac{a_2}{2} c_{12} \right) \\ \varphi_2 &\equiv \frac{1}{2} m_2 a_1 a_2 s_2 \dot{\theta}_1^2 + m_2 g \frac{a_2}{2} c_{12} \end{aligned} \quad (11.38c)$$

In Eqs. (11.38a-c), a_i , m_i , $I_{i,ZZ}$ for $i = 1, 2$, are the length, mass, and inertia about the mass-center perpendicular to the plane of motion of the i^{th} link, respectively. Considering the two control input components of the \mathbf{u}_e as $\mathbf{u}_e \equiv [u_{e1} \ u_{e2}]^T$, the control laws can be given from Eq. (11.36c) as

$$\begin{aligned} \tau_1 &= \frac{1}{a_1 a_2 s_2} [i_{11} a_2 c_{12} (u_{e1} - a_1 c_1 \dot{\theta}_1 \dot{x}_e - a_2 c_{12} \dot{\theta}_{12} \dot{y}_e) \\ &\quad - i_{12} a_2 s_{12} (u_{e1} - a_1 c_1 \dot{\theta}_1 \dot{x}_e - a_2 c_{12} \dot{\theta}_{12} \dot{y}_e)] + \varphi_1 \end{aligned} \quad (11.39a)$$

$$\begin{aligned} \tau_2 &= \frac{1}{a_1 a_2 s_2} [i_{21} (-a_1 c_1 - a_2 c_{12}) (u_{e2} - a_1 s_1 \dot{\theta}_1 \dot{x}_e - a_2 s_{12} \dot{\theta}_{12} \dot{y}_e) \\ &\quad + i_{12} (-a_1 s_1 - a_2 s_{12}) (u_{e2} - a_1 s_1 \dot{\theta}_1 \dot{x}_e - a_2 s_{12} \dot{\theta}_{12} \dot{y}_e)] + \varphi_2 \end{aligned} \quad (11.39b)$$

For the outer servo loop, the control will then be given by

$$u_{e1} = \ddot{x}_e + k_{v1} (\dot{x}_e - \dot{x}) + k_{p1} (x_e - x), \text{ and } u_{e2} = \ddot{y}_e + k_{v2} (\dot{y}_e - \dot{y}) + k_{p2} (y_e - y) \quad (11.39c)$$

The above two control laws, namely, Eqs. (11.39a-b) and Eq. (11.39c), will drive the error to zero. However, at $\theta_2 = 0$, the robot is in singular configuration which implies the torques τ_i , for $i = 1, 2$ become infinitely large. This was not the case in the case of feedback linearization with respect to joint variables. Hence, some measure like singularity avoidance techniques have to be introduced to overcome such problems. This control scheme is same as computed-torque control or the resolved-acceleration control for the nonlinear robot under study.

11.11 FORCE CONTROL

In the control schemes presented so far, the controlling joint torques were obtained using only joint or Cartesian trajectories without any reference to what force or moment the robot exert on its environment. This is important as the robot's end-effector has to exert some moment or force or both, which were defined in Chapter 9 as wrench, on an object handled by it. In many tasks, e.g., cleaning a glass, the robot should not exert more than certain force in order not to break it. In

other instances, where robots and human beings are working together, a robot must stop when it is touched, i.e., a threshold value of force is crossed to make the environment safe. In applications where the end-effector has to apply a controlled force, e.g., on a glass which is cleaned by a robot, the position control schemes presented in the previous sections will not be appropriate unless the tool at the end-effector is sufficiently compliant, e.g., like a sponge. If the end-effector is rigid like a scrapping tool then any uncertainty in the position of the glass surface or the positional error in the end-effector will either cause the glass to break or the scrapper will not touch the glass at all. In such situations, the force, not the position, should be specified in order to obtain satisfactory performance by the robot. In fact, the robot motion can be subdivided into *gross* and *fine* motions. While the former is usually fast and generally should be used for position control, the latter is slow when the end-effector's position relative to the environment is accurately adjusted based on force, which could be either *implicit* or *explicit*, based on an external force/torque sensor fitted at the wrist before the end-effector. Since the force control is generally slow, nonlinear dynamics of the robot is of little importance.

An approach to maintain an interaction force between a robot and its environment is by introducing some kind of compliance into the robot. If this compliance is large, a small position error will cause a small change in interaction force. A special passive compliance device, e.g., Remote Center Compliance (WR: RCC), can be placed in the robot's wrist to control the applied forces onto the environment. Such devices help in automated assembly, e.g., peg-in-hole or grinding tasks. Passive compliance devices can be undesirable as it may lead to oscillations if the manipulator moves very fast. Alternatively, it is possible to introduce compliance by reducing position feedback gain, which is referred as active compliance. In this case, a small manipulator displacement will cause a corresponding small variation in the control torques. Unlike mechanical compliance, as in the case of RCC, such artificial compliance is software-adjustable depending on the specific task. Controlling end-effector wrenches through the joint-actuator torques is known as *implicit force control*. Here, position error is related to the contact force through a mechanical stiffness or impedance with adjustable parameters, i.e., mass, damping ratio, and stiffness of the system. A robot under impedance control is equivalent to an equivalent mass-spring-damper system with contact force as input. One can, however, use an external force/torque sensor to measure its output to feedback in the drive control system. Such control can be referred to as *explicit force control*.

Wrench Control

Wrench means both moment and force, as defined in Chapter 9. Force control, which is more popular in the literature, is equivalent to wrench control.

In order to control a force (which includes moment and force both) at the end-effector, one may do it indirectly or directly. While the former can be through what is referred in the literature as impedance control, i.e., maintaining relation between the external force and the robot's motion, the latter actually tracks the forces based on explicit measurement of it, as in admittance or hybrid control. If a detailed model of the environment is available, a preferred strategy is hybrid control, which aims to control position along the unconstrained directions and force along the constrained task directions. A detailed treatment on the force control of a robotic system may be found in Gorinevsky et al. (1997).

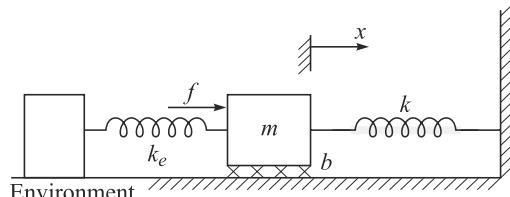


Fig. 11.14 Block interacting with environment

11.11.1 Impedance Control of a Moving Block

In frequency domain, impedance $z(s)$ of a moving block shown in Fig. 11.14 is defined as the ratio between force $f(s)$ and velocity $v(s)$, i.e.,

$$z(s) = \frac{f(s)}{v(s)} \quad (11.40a)$$

Equation (11.40a) can be expressed in terms of the displacement error as

$$z(s) = \frac{f(s)}{se(s)} \quad (11.40b)$$

where $e(s)$ is the Laplace transform of the displacement error given by $e(s) \equiv x_d(s) - x(s) - \dot{x}_d(s)$ and $x(s)$ being the desired and actual positions of the system at hand. This is illustrated with respect to Fig. 11.15.

For the moving block, the equation of motion is given by

$$m\ddot{x} + b\dot{x} + kx = f - f_e, \text{ where } f_e = k_e(x - x_e) \quad (11.41)$$

In Eq. (11.41), m , b , and k are the actual body mass, damping ratio, and stiffness of the mass-spring-damper system shown in Fig. 11.14, where f is the controlling force. Note that when $f_e \neq 0$, the simple PD control scheme, i.e., $f = -k_v\dot{x} - k_p x$, will not ensure the desired trajectory tracking by the system. It is assumed that the desired characteristics is achieved when the mass, damper, and spring are modified to m_d , b_d and k_d , respectively. Hence, the desired impedance dynamics model can be expressed as

$$m_d\ddot{x} + b_d(\dot{x} - \dot{x}_d) + k_d(x - x_d) = -f_e \quad (11.42)$$

where x_d is the desired motion that typically *slightly penetrates* inside the compliant environment, as shown in Fig. 11.15, thus, keeping a contact. Equation (11.42) can be proven by introducing an inverse dynamics or computer-torque like controller to Eq. (11.41) given by

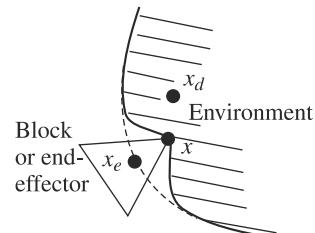


Fig. 11.15 Concept of stiffness

$$f = (m - m_d)u + b\dot{x} + kx, \text{ where } u = \ddot{x} + \frac{1}{m - m_d}(b_d\dot{e} + k_d e) \quad (11.43a)$$

In Eq. (11.43a), the error or difference in positions and velocities are $e \equiv x_d - x$ and $\dot{e} \equiv \dot{x}_d - \dot{x}$. It is now clear that if Eq. (11.43a) is substituted into Eq. (11.41), Eq. (11.42) results. As shown in Section 11.6, the control law of Eq. (11.21a) makes the errors converging to zeros. This is usually the case under no external force, i.e., $f_e = 0$. If $f_e \neq 0$, at steady-state, i.e., at $\dot{x} = 0$ and $\ddot{x} = 0$, the equilibrium of the closed-loop system using Eqs. (11.41–11.42) leads to

$$x = \frac{1}{k_d + k_e}(k_d x_d + k_e x_e) \quad (11.43b)$$

where $k_d + k_e$ is the stiffness of the closed-loop system and its inverse, i.e., $1/(k_d + k_e)$ is called the compliance. Hence, the stiffness or compliance control can be achieved by changing the control law according to Eq. (11.43a). Based on Eq. (11.43b), the following can be inferred:

- If $k_d \gg k_e$, then $x = x_d \Rightarrow$ Rigid controller
 - If $k_e \gg k_d$, then $x = x_e \Rightarrow$ Rigid environment
- (11.44)

11.11.2 Impedance Control of a Robot

The impedance control methodology for a moving block explained above can be extended to a multi-degree-of-freedom robot whose dynamics in Cartesian or task space is written first. For that, dynamic equations of motion of a robot under some external wrench, i.e., moment and force, applied to the end-effector is written from Eq. (11.8) as

$$\mathbf{I}\ddot{\boldsymbol{\theta}} + \boldsymbol{\varphi} = \boldsymbol{\tau} + \mathbf{J}^T \mathbf{w}_e^E \quad (11.45a)$$

and $\mathbf{w}_e^E \equiv \begin{bmatrix} (\mathbf{n}_e^E)^T & (\mathbf{f}_e^E)^T \end{bmatrix}^T$ is the external or environmental wrench applied on the end-effector, in which \mathbf{n}_e^E and \mathbf{f}_e^E are the three-dimensional external moment and force respectively. Matrix \mathbf{J} is the Jacobian matrix for the robot relating the twist of the end-effector, denoted with \mathbf{t}_e , with its joint rates denoted with $\dot{\boldsymbol{\theta}}$, i.e., $\mathbf{t}_e = \mathbf{J}\dot{\boldsymbol{\theta}}$. Using this and its time derivatives, Eq. (11.45a) can be rewritten as

$$\mathbf{I}\mathbf{J}^{-1}(\dot{\mathbf{t}}_e - \mathbf{J}\dot{\boldsymbol{\theta}}) + \boldsymbol{\varphi} = \boldsymbol{\tau} + \mathbf{J}^T \mathbf{w}_e^E \quad (11.45b)$$

Assuming that the robot is nonredundant such that matrix \mathbf{J} is square and its inverse exists unless the robot is in a singular position, matrix \mathbf{J}^{-T} is pre-multiplied with both sides of Eq. (11.45b) resulting the following set of dynamic equations written in Cartesian space of the end-effector motion:

$$\mathbf{I}_e \dot{\mathbf{t}}_e + \boldsymbol{\varphi}_e = \mathbf{w}_e + \mathbf{w}_e^E, \text{ where } \mathbf{I}_e \equiv \mathbf{J}^{-T} \mathbf{I} \mathbf{J}^{-1}; \boldsymbol{\varphi}_e \equiv \mathbf{J}^{-T}(\boldsymbol{\varphi} - \mathbf{I} \mathbf{J}^{-1} \mathbf{J} \mathbf{J}^{-1} \dot{\mathbf{t}}_e); \mathbf{w}_e \equiv \mathbf{J}^{-1} \boldsymbol{\tau} \quad (11.45c)$$

In Eq. (11.45c), \mathbf{w}_e^E is the external wrench. If the inverse dynamics or compute-torque-like controller is introduced, as done for the moving block in Eq. (11.43a), in order to achieve a desired impedance in the robot, the wrench its end-effector has to be produced by the joint actuators can be given as

$$\mathbf{w}_e = (\mathbf{I}_e - \mathbf{I}_d)\mathbf{u}_e + \boldsymbol{\varphi}_e, \text{ where } \mathbf{u}_e = \dot{\mathbf{t}}_e + (\mathbf{I}_e - \mathbf{I}_d)^{-1}(\mathbf{K}_{ve}\dot{\mathbf{e}} + \mathbf{K}_{pe}\mathbf{e}) \quad (11.46)$$

Moreover, $\mathbf{e} \equiv \mathbf{x}_d - \mathbf{x}_e$, $\dot{\mathbf{e}} \equiv \mathbf{t}_d - \mathbf{t}_e$. Then the resulting impedance dynamics similar to Eq. (11.42) which can be achieved is given below:

$$\mathbf{I}_d \ddot{\mathbf{t}}_e + \mathbf{K}_{ve}(\mathbf{t}_e - \mathbf{t}_d) + \mathbf{K}_{pe}(\mathbf{x}_e - \mathbf{x}_d) = \mathbf{w}_e^E \quad (11.47)$$

where \mathbf{I}_d is the desired inertia, and \mathbf{t}_d and $\dot{\mathbf{t}}_d$ are the desired twist and twist-rate of the end-effector, respectively, whereas \mathbf{x} and \mathbf{x}_d represent the actual and desired configurations of the end-effector, respectively, containing the information about orientation in the form of say, Euler angles or Euler parameter, etc., and the position. Moreover, the gain matrices \mathbf{K}_{ve} and \mathbf{K}_{pe} actually represent the desired damping and stiffness at the end-effector, respectively, as denoted by b_d and k_d in Eq. (11.43a) for the moving block. Now, in steady condition, i.e., when $\mathbf{t}_e = \mathbf{t}_d = \mathbf{0}$, and $\dot{\mathbf{t}}_e = \dot{\mathbf{t}}_d = \mathbf{0}$, Eq. (11.47) yields

$$\mathbf{K}_{pe} \Delta \mathbf{x}_e = \mathbf{w}_e^E, \text{ where } \Delta \mathbf{x}_e \equiv \mathbf{x}_e - \mathbf{x}_d \quad (11.48)$$

which is nothing but the stiffness control law for the robot at hand. In order to implement the above law, the required controlling torque τ can then be obtained as $\tau = \mathbf{J}^T \mathbf{w}_e$, where \mathbf{w}_e is obtained Eq. (11.46).

11.11.3 Force Control of a Moving Block

In one of the force control approaches for a robot, the force experienced by the end-effector due to its interaction with the environment must be controlled. This is illustrated here with the help of a simple block. Similar derivations are also available in Craig (2009). Referring to Fig. 11.14, the equation describing the physical system is given by Eq. (11.41). If the environmental force f_e is to be controlled, the equation of motion shown on the left equation of Eq. (11.41) is rewritten by double differentiating $f_e = k_e(x - x_e)$ and writing $\ddot{x} = \ddot{f}_e/k_e$ as

$$\frac{m}{k_e} \ddot{f}_e + f_{\text{dist}} + f_e = f, \text{ where } f_{\text{dist}} \equiv b\dot{x} + kx \quad (11.49)$$

In Eq. (11.49), f_{dist} denotes the disturbance force due to unknown friction and stiffness effects. Now, a control in the form of Eq. (11.43a) can be taken as

$$f = \frac{m}{k_e} u_f + f_e + f_{\text{dist}}, \text{ where } u_f = \ddot{f}_d + k_{vf} \dot{e}_f + k_{pf} e_f \quad (11.50)$$

Substitution of Eq. (11.50) into Eq. (11.49) then yields the following closed-loop system:

$$\ddot{e}_f + k_{vf} \dot{e}_f + k_{df} e_f = 0, \text{ where } e_f \equiv f_d - f_e \quad (11.51)$$

For positive values of k_{vf} and k_{df} , it is well-known that the error will vanish, i.e., the desired force value f_d will be achieved. It is pointed out here that the knowledge of disturbance force f_{dist} in Eq. (11.50) is not known. Hence, it cannot be included in the control law. It is advisable to use the desired force f_d instead of $f_e + f_{\text{dist}}$ which provides a better estimate of the steady-state error for a stiff environment. Whenever desired, an appropriate integral term in the control law can reduce the steady-state error, as done with the PID position controllers earlier. Note also from the practical considerations that the force trajectories are generally constants, i.e., one is more

interested to maintain a constant contact force rather than maintaining a force which is a function of time. Hence, $\dot{f}_d = 0$ and $\ddot{f}_d = 0$. Besides, the sensed force f_e is quite noisy whose numerical differentiation to obtain \dot{f}_e is not advised. However, $f_e = k_e(x - x_e)$ can be used to obtain $\dot{f}_e = k_e\dot{x}$, which is more realistic for the rest position of x_e . With the above considerations the control law of Eq. (11.50) is simplified as

$$f = m \left(\frac{k_{pf}}{k_e} e_f - k_{vf} \dot{x} \right) + f_d \quad (11.52)$$

Figure 11.16 depicts the control of Eq. (11.52).

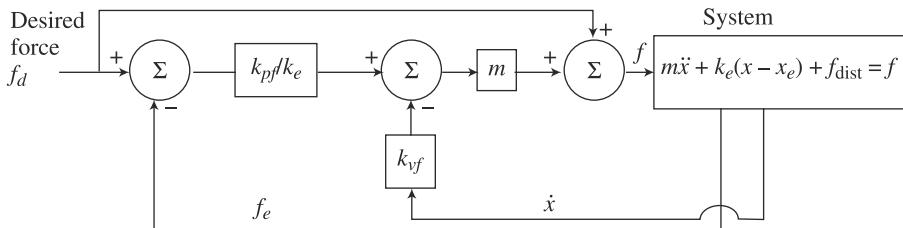


Fig. 11.16 Force control of a moving block

11.11.4 Natural and Artificial Constraints

In hybrid control, which will follow in Section 11.12, both position and force are of interest. For that note an interaction task of a robot's end-effector with its environment, where it can be broken into two tasks, namely, as end-effector force and its velocity. A robot moving freely is unconstrained in orientation and position, i.e., configuration, and can exert no moment and force, i.e., wrench, on the environment due to no source of reaction wrench from the environment. When the robot touches, say, a rigid surface, a degree-of-freedom in position is lost since the robot cannot move through the surface. At the same time, the robot can exert force normal to the surface. In order to specify the force control tasks, a constraint frame is typically attached to the end-effector of the robot. The constraint frame may vary with the time. However, for the task at hand, it can be decomposed along the directions of the constraint frame. For each axis of the constraint frame, there are two associated degrees-of-freedom (DOF). One is to define linear velocity or a force along that axis, and the second one is to specify the angular velocity or moment about the same axis.

Note that when the end-effector of a robot is in contact with a surface, certain *natural constraints* arise. Any other force and position variables that can be controlled during motion are referred as *artificial constraints*. For a rigid environment without friction, these constraints with respect to the constraint frame that may be attached to the robot's end-effector are defined below.

1. Natural Constraints When a robot's end-effector is in contact with a surface it cannot go through it, i.e., it cannot translate along certain direction of the axis or rotate about it. A natural position constraint exists. Another possibility is that when the contacting surface is frictionless, the robot cannot apply any force tangent to the surface. Also, when the end-effector moves in a free space, no force or moment can

be applied along or about the constraint frame axis. Hence, natural force constraints exist. They are called *natural* because these are strictly governed by the task geometry.

Note that when any two surfaces are in contact, the above natural constraints can be specified as the position constraint along the normal to the surface while the force constraint can be denied along the tangent to the surface. These two types of natural constraints, i.e., force and position, partition the degrees of freedom of possible end-effector motion into two orthogonal sets that must be controlled according to different criteria.

2. Artificial Constraints As mentioned above, a robot cannot control variables on those directions where natural constraints occur. For example, if the robot moves in a free space, it cannot control any force or moment but certainly can control a motion trajectory. Hence, reference values for such motion trajectory are referred to as *artificial constraints*.

It is pointed out here that the numbers of natural and artificial constraints together are equal to the total number of degrees-of-freedom (DOF) of the task space which is six for Cartesian operation by a robot's end-effector. Examples 11.10-11.12 illustrate the natural and artificial constraints.

Example 11.10 Natural and Artificial Constraints in a Sliding Box

Figure 11.17 shows a box sliding in a guideway. It is allowed to move in the Y_c direction. Hence, the natural constraint corresponds to the force along Y_c direction vanishes, i.e., $f_y = 0$. No force in this direction can be applied. Similarly, no moment can be applied about X_c . Hence, the moment about X_c also vanishes, i.e., $n_x = 0$. Additionally, no motion can happen along X_c and Z_c directions. As a result, there will be natural constraints corresponding to these motions as $v_x = v_z = 0$, and $\omega_y = \omega_z = 0$. The variables v and ω represent a linear velocity and an angular velocity, respectively. These natural constraints are shown in Table 11.1.

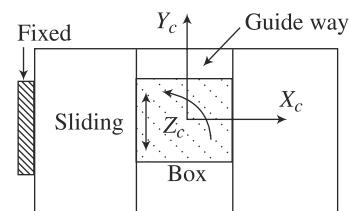


Fig. 11.17 A box sliding in a guide way

Table 11.1 Natural and artificial constraints of a sliding box

Direction	Natural constraints	Artificial constraints
1	$\omega_y = 0$	$n_y = n_{dy} = 0$
2	$\omega_z = 0$	$n_z = n_{dz} = 0$
3	$v_x = 0$	$f_x = f_{dx} = 0$
4	$v_z = 0$	$f_z = f_{dz} = 0$
5	$n_x = 0$	$\omega_x = \omega_{dx} = 0$
6	$f_y = 0$	$v_y = v_{yd}$
DOF = 6	4 motion + 2 force	2 motion + 4 force

$\omega_{dk}, v_{dk}, n_{dk}, f_{dk}$: Desired angular velocity, linear velocity, moment, and force about or along k -axis ($k = X_c, Y_c, Z_c$)

Referring to Table 11.1, it is clear that the box is allowed to move along Y_c directions with velocity v_{yd} . If the definition of twist of the box or the end-effector, which is holding the box to move along the guideway, is revoked as $\mathbf{t}_e \equiv [\mathbf{\omega}_e^T \quad \mathbf{v}_e^T]^T$ then \mathbf{t}_e can be expressed in terms of the artificial constraints that are to be specified by the users through a robot's controller, namely, ω_{dx} and v_{dy} , as

$$\mathbf{t}_e = \mathbf{S}_q \dot{\mathbf{q}}_d, \text{ where } \mathbf{S}_q \equiv \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } \dot{\mathbf{q}}_d \equiv \begin{bmatrix} \omega_{dx} \\ v_{dy} \end{bmatrix} \quad (11.53a)$$

where \mathbf{S}_q is the 6×2 matrix corresponding to the twist of end-effector, whereas $\dot{\mathbf{q}}_d$ is the two-dimensional vector containing the artificial constraints associated to desired linear and angular velocities. Similarly, the artificial constraints on forces and moments can relate to the wrench exerted on the end-effector as

$$\mathbf{w}_e^C = \mathbf{S}_\lambda \lambda_d, \text{ where } \mathbf{S}_\lambda \equiv \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \lambda_d \equiv \begin{bmatrix} n_{dy} \\ n_{dz} \\ f_{dx} \\ f_{dz} \end{bmatrix} \quad (11.53b)$$

where \mathbf{S}_λ is the 6×4 switch matrix corresponding the wrench of end-effector, whereas λ_d is the four-dimensional vector containing artificial constraints related to desired moments and forces. Note here that a concept of diagonal *selector switch* matrices, denoted with \mathbf{S}_t and \mathbf{S}_w , is introduced to relate the twist and wrench of the end-effector, i.e., \mathbf{t}_e and \mathbf{w}_e , in terms of the desired twist and wrench denoted with \mathbf{t}_{ed} and \mathbf{w}_{ed} , respectively, as

$$\mathbf{t}_e = \mathbf{S}_t \mathbf{t}_{ed} \text{ and } \mathbf{w}_e^C = \mathbf{S}_w \mathbf{w}_{ed} \quad (11.54a)$$

where the six-dimensional vectors of desired end-effector twist and wrench, \mathbf{t}_{ed} and \mathbf{w}_{ed} , and the 6×6 selector switch matrices \mathbf{S}_t and \mathbf{S}_w are defined below:

$$\mathbf{t}_d \equiv \left[\mathbf{\omega}_d^T \quad \mathbf{v}_d^T \right]^T \equiv \left[\omega_{dx} \quad \omega_{dy} \quad \omega_{dz} \quad v_{dx} \quad v_{dy} \quad v_{dz} \right]^T \quad (11.55a)$$

$$\mathbf{w}_d \equiv \left[\mathbf{n}_d^T \quad \mathbf{f}_d^T \right]^T \equiv \left[n_{dx} \quad n_{dy} \quad n_{dz} \quad f_{dx} \quad f_{dy} \quad f_{dz} \right]^T \quad (11.55b)$$

$$\mathbf{S}_t \equiv \text{diag.}[1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0], \text{ and } \mathbf{S}_w \equiv \text{diag.}[0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1] \quad (11.55c)$$

Comparing Eqs. (11.53a-b) with Eqs. (11.55a-c), vectors $\dot{\mathbf{q}}_d$ and λ_d contain the 1st and 5th elements of \mathbf{t}_d , and 2nd, 3rd, 4th, and 6th elements of \mathbf{w}_d , respectively. Accordingly, matrix \mathbf{S}_q is nothing but the one consisting of the 1st and 5th columns of \mathbf{S}_t , whereas matrix \mathbf{S}_w comprises of the 2nd, 3rd, 4th, and 6th columns of \mathbf{S}_w .

Example 11.11 Natural and Artificial Constraint during Turning of a Crank

Similar to Example 11.10, the natural and artificial constraints for turning a crank, as shown in Fig. 11.18, are listed in Table 11.2. For the equations shown in Eqs. (11.53a-b), the corresponding vectors and matrices for the crank turning task are given below:

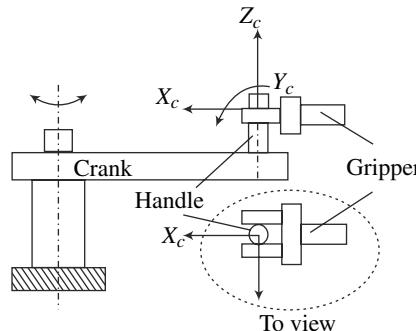


Fig. 11.18 The end-effector of a robot turning a crank

$$\mathbf{S}_q \equiv \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \dot{\mathbf{q}}_d \equiv \begin{bmatrix} \omega_{dz} \\ v_{dy} \end{bmatrix} \text{ and } \mathbf{S}_\lambda \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \lambda_d \equiv \begin{bmatrix} n_{dx} \\ n_{dy} \\ f_{dx} \\ f_{dz} \end{bmatrix} \quad (11.56a)$$

Moreover,

$$\mathbf{S}_t \equiv \text{diag.}[0 \ 0 \ 1 \ 0 \ 1 \ 0], \text{ and } \mathbf{S}_w \equiv \text{diag.}[1 \ 1 \ 0 \ 1 \ 0 \ 1] \quad (11.56b)$$

Table 11.2 Natural and artificial constraints during turning a crank

Direction	Natural constraints	Artificial constraints
1	$\omega_x = 0$	$n_x = n_{dx} = 0$
2	$\omega_y = 0$	$n_y = n_{dy} = 0$
3	$v_x = 0$	$f_x = f_{dx} = 0$
4	$v_z = 0$	$f_z = f_{dz} = 0$
5	$n_z = 0$	$\omega_z = \omega_{dz} = 0$
6	$f_y = 0$	$v_y = v_{dy}$
DOF = 6	4 motion + 2 force	2 motion + 4 force

$\omega_{dk}, v_{dk}, n_{dk}, f_{dk}$: Desired angular velocity, linear velocity, moment, and force about or along k -axis ($k = X_c, Y_c, Z_c$)

Example 11.12 Natural and Artificial Constraints during Peg-in-Hole Operations

As shown in Figs. 11.19(a-d), the complete task can be divided into four subtasks, namely,

Subtask 1: Free-space motion, Fig. 11.19(a)

Subtask 2: Peg touching the surface, Fig. 11.19(b)

Subtask 3: Inserting peg in hole, Fig. 11.19(c)

Subtask 4: Assembly complete, Fig. 11.19(d)

The natural and artificial constraints corresponding to the above subtasks are given in Table 11.3.

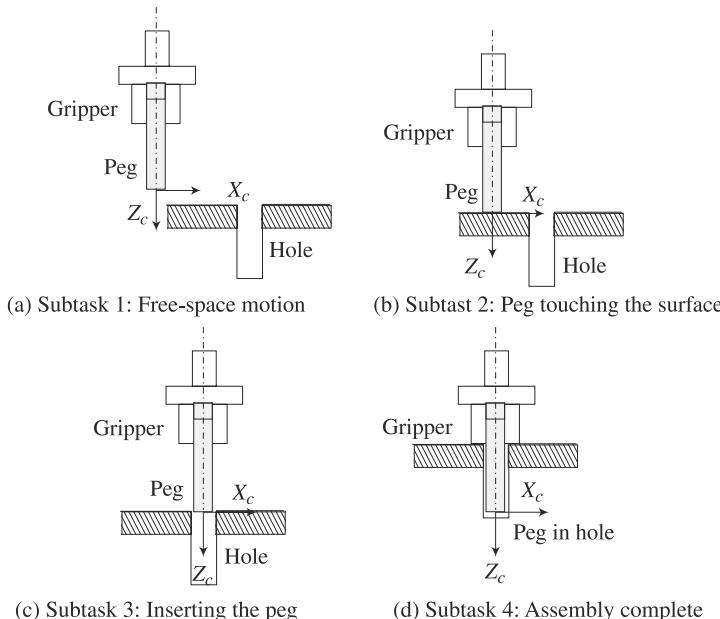


Fig. 11.19 Subtasks in peg-in-hole assembly

Table 11.3 Natural and artificial constraints for peg-in-hole assembly

Subtask 1 constrains		Subtask 2 constrains		Subtask 3 constrains		Subtask 4 constrains	
Natural	Artificial	Natural	Artificial	Natural	Artificial	Natural	Artificial
$n_x = 0$	$\omega_x = \omega_{dx} = 0$	$\omega_x = 0$	$n_x = n_{dx} = 0$	$\omega_x = 0$	$n_x = n_{dx} = 0$	$\omega_x = 0$	$n_x = n_{dx} = 0$
$n_y = 0$	$\omega_y = \omega_{dy} = 0$	$\omega_y = 0$	$n_y = n_{dy} = 0$	$\omega_y = 0$	$n_y = n_{dy} = 0$	$\omega_y = 0$	$n_y = n_{dy} = 0$
$n_z = 0$	$\omega_z = \omega_{dz} = 0$	$v_z = 0$	$f_z = f_{dz}$	$v_x = 0$	$f_x = f_{dx} = 0$	$\omega_z = 0$	$n_z = n_{dz} = 0$
$f_x = 0$	$v_x = v_{dx} = 0$	$n_z = 0$	$\omega_z = \omega_{dz} = 0$	$v_y = 0$	$f_y = f_{dy} = 0$	$v_x = 0$	$f_x = f_{dx} = 0$
$f_y = 0$	$v_y = v_{dy} = 0$	$f_x = 0$	$f_x = f_{dx} = 0$	$n_z = 0$	$\omega_z = \omega_{dz} = 0$	$v_y = 0$	$f_y = f_{dy} = 0$
$f_z = 0$	$v_z = v_{zd}$	$f_y = 0$	$f_y = f_{dy} = 0$	$f_z = 0$	$v_z = v_{zd}$	$v_z = 0$	$f_z = f_{dz} = 0$
6F	6M	3M+3F	3F+3M	4M+2F	2M+4F	6M	6F

F: force, M: motion; ω_{dk} , v_{dk} , n_{dk} , f_{dk} : Desired angular velocity, velocity, moment, and force about or along k-axis ($k = X_c, Y_c, Z_c$)

Based on Table 11.3, one can easily form the associated vectors and matrices equivalent to Eq. (11.56a), which are left as exercises. From Eq. (11.55c), it is clear that whichever diagonal element of the matrix \mathbf{S}_t has one, i.e., position control is in effect, the corresponding diagonal element of the matrix \mathbf{S}_w is zero, i.e., no force control is possible. It is also clear from Eq. (11.55c), that $\mathbf{S}_w = \mathbf{1} - \mathbf{S}_t$, where $\mathbf{1}$ is the 6×6 identity matrix. The complimentary of the selector switch matrices actually confirms that both motion and force cannot be controlled along one direction. Either of the one is selected. Since the constraint moments and forces cannot perform any work, power due to them is zero. Hence, the following is true:

$$\mathbf{t}_e^T \mathbf{w}_e^C = 0 \Rightarrow \ddot{\mathbf{q}}_d^T \mathbf{S}_q^T \boldsymbol{\lambda}_d = 0 \quad \text{or} \quad \mathbf{t}_d^T \mathbf{S}_t^T \mathbf{S}_w \mathbf{w}_d = 0 \quad (11.57a)$$

Equation (11.57a) implies that

$$\mathbf{S}_t^T \mathbf{S}_w = \mathbf{0} \quad \text{or} \quad \mathbf{S}_t^T \mathbf{S}_w = \mathbf{S}_t^T (\mathbf{1} - \mathbf{S}_t) = \mathbf{S}_t^T - \mathbf{S}_t^T \mathbf{S}_t = \mathbf{0} \quad (11.57b)$$

where $\mathbf{0}$ denotes the matrix of zeros whose dimension is compatible to the expression where it appears. For example, the first $\mathbf{0}$ in Eq. (11.57b) is the 2×4 matrix of zeros, whereas the second one is the 6×6 matrix of zeros. Also, $\mathbf{S}_t^T \mathbf{S}_t = \mathbf{S}_t = \mathbf{S}_t^T$ was used due to the structure of the diagonal matrix \mathbf{S}_t , which has only ones or zeros in its diagonal elements.

11.12 HYBRID CONTROL

Based on pure force control, which was illustrated with a simple system of a moving block in Section 11.11.3, and the concept of task space decomposition into motion and force, as explained in Section 11.11.4, one can now present the hybrid control algorithm. Earlier, in Sections 11.11.1–11.11.2, the concept of impedance control was presented, where not the actual force is of interest rather the relationship between the motions versus force on the end-effector was controlled through the joint motion control. However, in many applications, it is more realistic to track the force directly in a certain direction while making the robot move in other directions. One typical example is cleaning a glass window. On the particular direction of the glass, the robot must not exert a force than specified while moving the cleaning brush on the plane of the glass. Such a control scheme is referred to as hybrid position/force control. In such control algorithms, the robot's task is broken down into several subtasks. A typical illustration of a position/force hybrid controller is shown in Fig. 11.20, where \mathbf{S}_t and \mathbf{S}_w are diagonal matrices which act as switch mode to decide whether the position or force controller is in action, as given in Examples 11.10–11.12. Note that such switching using selection matrix \mathbf{S}_t or \mathbf{S}_w are useful for planar contact surfaces only. For general curved contact surfaces, one needs explicit constraint equations. Here, the discussion will be restricted to planar contact only, whereas the same based on curved surfaces is given in Yoshikawa (1990).

For hybrid control, the objective is to impose desired trajectory to the components of the velocities, denoted with $\dot{\mathbf{q}}_d$, and the moments and forces, denoted with $\boldsymbol{\lambda}_d$. The control law is designed in such a way that the closed-loop system behaves like a unit mass, namely,

$$\dot{\mathbf{q}}_h = \mathbf{u}_h, \text{ where } \ddot{\mathbf{q}}_h \equiv \begin{bmatrix} \ddot{\mathbf{q}}_d^T & \boldsymbol{\lambda}_d^T \end{bmatrix}^T \text{ and } \mathbf{u}_h \equiv \begin{bmatrix} \mathbf{u}_q^T & \mathbf{u}_{\lambda}^T \end{bmatrix}^T \quad (11.58)$$

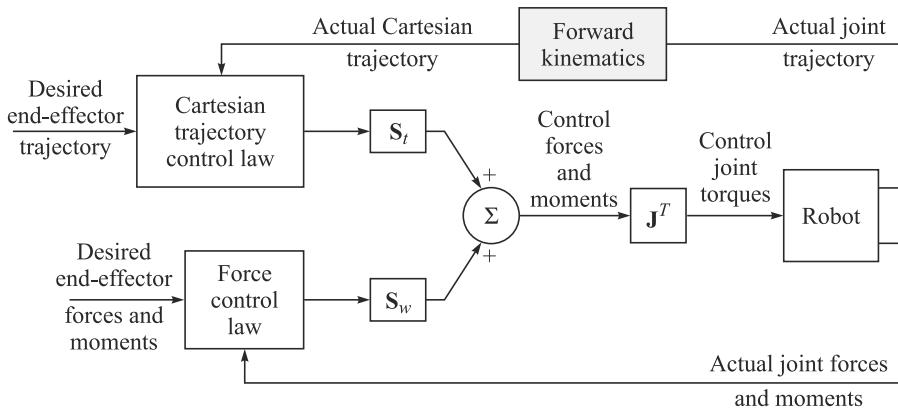


Fig. 11.20 Hybrid position/force controller

Subscript h is used to denote vectors associated with the hybrid control. Based on Eq. (11.58), the dynamic equations of motion of a multi-DOF robot can be re-written as

$$\mathbf{I}_h \ddot{\mathbf{q}}_h + \boldsymbol{\varphi}_h = \boldsymbol{\tau}, \text{ where } \mathbf{I}_h \equiv [\mathbf{I}_q \quad \mathbf{J}_\lambda] \quad (11.59a)$$

Matrices \mathbf{I}_q , \mathbf{J}_λ , and the vector $\boldsymbol{\varphi}_h$ with suitable dimensions are given by

$$\mathbf{I}_q \equiv \mathbf{I}\mathbf{J}^{-1}\mathbf{S}_q, \mathbf{J}_\lambda \equiv -\mathbf{J}^T\mathbf{S}_\lambda, \text{ and } \boldsymbol{\varphi}_h \equiv \boldsymbol{\varphi} + \mathbf{I}\mathbf{J}^{-1}(\dot{\mathbf{S}}_q - \mathbf{J}\mathbf{J}^{-1}\mathbf{S}_q)\dot{\mathbf{q}}_d \quad (11.59b)$$

Like the way it was done for the inverse dynamics control of Section 11.6, the control input for the hybrid control is considered as

$$\boldsymbol{\tau} = \mathbf{I}_h \mathbf{u}_h + \boldsymbol{\varphi}_h \quad (11.60a)$$

where the control laws for the decoupled components of \mathbf{u}_h , namely, \mathbf{u}_q and \mathbf{u}_λ , can be taken as

$$\mathbf{u}_q \equiv \ddot{\mathbf{q}}_d + \mathbf{K}_v \dot{\mathbf{e}}_q + \mathbf{K}_p \mathbf{e}_q \quad \text{and} \quad \mathbf{u}_\lambda \equiv \boldsymbol{\lambda}_d + \mathbf{K}_i \boldsymbol{\varepsilon}_\lambda \quad (11.60b)$$

In Eq. (11.60b), $\mathbf{e}_q \equiv \mathbf{q}_d - \mathbf{q}$ and $\boldsymbol{\varepsilon}_\lambda \equiv \int (\boldsymbol{\lambda}_d - \boldsymbol{\lambda}) d\tau$. The control laws that govern the error equations $\ddot{\mathbf{e}}_q + \mathbf{K}_v \dot{\mathbf{e}}_q + \mathbf{K}_p \mathbf{e}_q = \mathbf{0}$ and $\dot{\boldsymbol{\varepsilon}}_\lambda + \mathbf{K}_i \boldsymbol{\varepsilon}_\lambda = \mathbf{0}$, respectively, make sure that $\mathbf{e}_q \rightarrow \mathbf{0}$ and $\boldsymbol{\varepsilon}_\lambda \rightarrow \mathbf{0}$ for suitable values of diagonal matrices, \mathbf{K}_v , \mathbf{K}_p , and \mathbf{K}_i .

SUMMARY

In this chapter, nonlinear and force control algorithms for robots are presented. Stability for nonlinear control is addressed using Lyapunov's function. Different force control schemes are also explained with examples.

EXERCISES

11.1 Consider the nonlinear system

$$\ddot{x} + 7\dot{x}^2 + x\dot{x} + x^3 = u$$

where u is the control input. Design a suitable control system based on the control-law partitioning such that the error response is critically damped. Draw the block diagram.

- 11.2** Why are Cartesian control and force control necessary in a robotic application?
11.3 Explain Lyapunov's stability criterion for nonlinear systems.
11.4 Find the selector switch matrices for the interaction in Fig. 11.21.

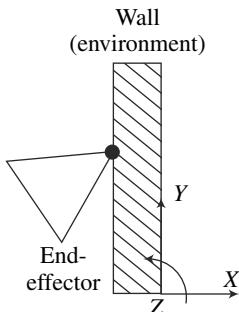


Fig. 11.21 Robot end-effector against a wall

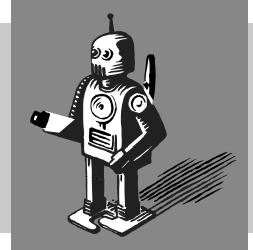
- 11.5** What is the principle of computed-torque method in robotics?
11.6 When does one require adaptive control?
11.7 Mention the necessity of a hybrid controller.

MATLAB BASED EXERCISES

- 11.8** Verify the results of Figs. 11.4–11.5.
11.9 Draw a Simulink block to produce the result of Fig. 11.8 using computed-torque control of a two-link arm.
11.10 Verify the results of Example 11.9 using the symbolic variables.

12

Motion Planning



The goal of motion planning of a robot is to generate a function according to which a robot's joints will move. This function generation depends on the robot tasks, for example, to pick up an object from one point to place it in another point, or to weld two metal pieces along a curve. In the former case, initial and final points are assigned, i.e., *point-to-point* motion, whereas in the latter situation a finite sequence of points needs to be specified, i.e., *continuous-path* motion. A robot user typically specifies a number of parameters to describe a point-to-point or continuous-path task. An algorithm then computes the inputs for the motion controller of the robot.

Motion planning, which is also referred in the literatures as trajectory planning, can be done either in the joint space, i.e., in terms of joint positions, velocities, and accelerations, or Cartesian space (also called operational space) i.e., in terms of the end-effector positions, orientations, and their time derivatives. Usually, the latter is preferred by an operator since it allows a natural description of the task the robot has to perform. However, the control action on the robot is carried in the joint space. Thus, a suitable inverse kinematics algorithm is to be used to reconstruct the time sequence of joint variables corresponding to the above sequence in the Cartesian space.

Motion planning in the Cartesian space naturally allows accounting for the presence of any constraints along the end-effector path. These are due to regions of workspace which are forbidden to the robot, e.g., due to the presence of obstacles. Such constraints are better described in the Cartesian space because their corresponding points in the joint space are difficult to compute. But in the neighborhood of singular configurations, trajectory planning in the Cartesian space is not possible. This is due to the fact that no joint variable can be computed for given end-effector variables due to non-existence of the inverse of the associated motion transfer matrix, called the Jacobian matrix. In such cases, it is advisable to specify the end-effector movement in terms of the joint variables, i.e., joint space planning. Hence, a time sequence of joint variables has to be generated. For completeness, both types of motion planning are presented in the following sections.

Why Motion Planning?

Just like a person to decide a route from one place to another, a robot's path needs to be decided through motion planning.

12.1 JOINT SPACE PLANNING

In this section, methods to determine a desired trajectory of joint variables over time are considered. In the joint-space scheme, it is assumed that the initial and final joint positions are given or calculated from corresponding initial and final configurations of the end-effector or are known from the inverse kinematics solutions for position, as provided in Chapter 6. Then, the basic problem is how to select a trajectory between a given initial and final joint positions with the time interval allowed to move between them. A simple straightforward approach could be to join the initial and final joint angles, denoted with θ_0 at time $t = t_0$ and θ_f at time $t = t_f$, respectively, using a straight-line function, which can be expressed as

$$\theta(t) = a_0 + a_1 t, \text{ where } a_0 \equiv \theta(t_0) \text{ and } a_1 \equiv \frac{\theta(t_f) - \theta(t_0)}{t_f - t_0} \quad (12.1)$$

where $\theta(t)$ is the joint angle at any time t , and a_1 is the slope of the straight line shown in Fig. 12.1(a), which is constant. This implies that the joint rate or velocity has a constant value of a_1 shown in Fig. 12.1(b), whereas the acceleration is zero as indicated in Fig. 12.1(c). If more than one points have to be connected, a series of such straight lines have to be joined.

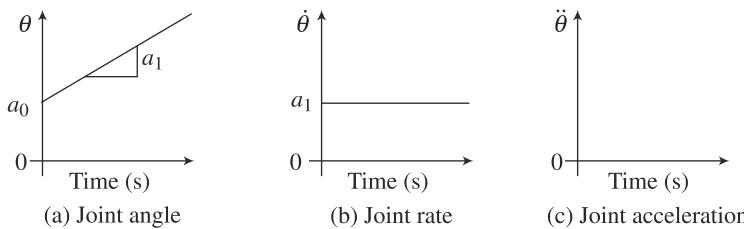


Fig. 12.1 A straight-line trajectory

Such first-order polynomial trajectory is generally not desired due to its discontinuities in joint rates. Note that for a real robotic joint to follow a straight-line trajectory of Eq. (12.1), the actuating motor, say, an electric one will require to achieve almost an instantaneous velocity. This implies that the electric motor must provide an almost infinite acceleration requiring very high current. As a result, the motor may burn. Hence, such a trajectory should not be used.

12.1.1 Cubic Polynomials

An alternative choice where both the velocity and acceleration-level continuity need to be maintained is to use a third-order or cubic polynomial given by

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (12.2a)$$

where the four coefficients a_0 , a_1 , a_2 , and a_3 are calculated from the velocity and acceleration conditions at time t_0 and t_f , i.e.,

$$\theta(t_0) = \theta_0; \theta(t_f) = \theta_f \text{ and } \dot{\theta}(t_0) = \dot{\theta}_0; \dot{\theta}(t_f) = \dot{\theta}_f \quad (12.2b)$$

To calculate the coefficients, first time-derivatives of Eq. (12.2a) are obtained as

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (12.2c)$$

Now, if one substitutes the conditions of Eq. (12.2b) into the velocity expression of Eq. (12.2c), it yields the following set of linear algebraic equations:

$$\mathbf{Ax} = \mathbf{b} \quad (12.3a)$$

where the 4×4 matrix \mathbf{A} , the 4-dimensional unknown vector \mathbf{x} containing the polynomial coefficients, and the 4-dimensional known vector \mathbf{b} of end-conditions are defined below:

$$\mathbf{A} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix}, \mathbf{x} \equiv \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}, \text{ and } \mathbf{b} \equiv \begin{bmatrix} \theta_0 \\ \theta_f \\ \dot{\theta}_0 \\ \dot{\theta}_f \end{bmatrix} \quad (12.3b)$$

Solution to the set of linear algebraic equations given by Eqs. (12.3a-b) can be obtained as $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, which results in the following:

$$a_0 = \theta_0; a_1 = \dot{\theta}_0 \quad (12.3c)$$

$$a_2 = \frac{1}{t_f^2} [3(\theta_f - \theta_0) - (2\dot{\theta}_0 + \ddot{\theta}_f)t_f] \quad (12.3d)$$

$$a_3 = \frac{1}{t_f^3} [2(\theta_0 - \theta_f) + (\dot{\theta}_0 + \dot{\theta}_f)t_f] \quad (12.3e)$$

Note that if the relation shown in Fig. 12.2 is satisfied by θ_0 , θ_f , $\dot{\theta}_0$ and $\dot{\theta}_f$ that is, if

$$\theta_f - \theta_0 = \frac{t_f}{2} (\dot{\theta}_0 + \dot{\theta}_f) \quad (12.3f)$$

then $a_3 = 0$ from Eq. (12.3e), which implies that only a second-order polynomial is required for $\theta(t)$. The corresponding coefficient a_2 is then given by

$$a_2 = \frac{1}{2t_f} (\dot{\theta}_f - \dot{\theta}_0) \quad (12.3g)$$

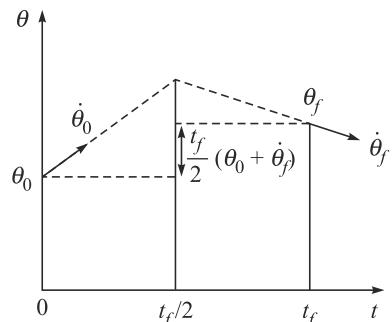


Fig. 12.2 Boundary conditions

12.1.2 Cubic Polynomials with Intermediate Points

During a robot motion, it is often desired that one or more intermediate points need to be specified. There are also possibilities to specify only velocities, or velocities and accelerations both at these intermediate points. If one chooses to use cubic polynomials, there could be $n + 1$ segments for n specified intermediate points. It is possible to find the coefficients for each segment as a_{0i}, \dots, a_{3i} , for $i = 1, \dots, n$ using the methodology explained in Eq. (12.3). As seen earlier, only the velocity continuity can be maintained but not the accelerations. This is illustrated in Example 12.1 below. In case only the position of the intermediate points are given and there is no restriction on velocity and acceleration, one can obtain a smooth continuous curve. This is also illustrated in Example 12.1.

Example 12.1 Cubic Polynomials with Specified Intermediate Velocities

Assume a robot joint has to move from 0° to 90° in 5 seconds. The initial and final joint rates are respectively $25^\circ/\text{s}$ and $-25^\circ/\text{s}$. In the intermediate point of 45° at time $t = 3 \text{ s}$, the joint rate is assumed to be $-8^\circ/\text{s}$. Two cubic trajectories obtained using Eq. (12.3) are as follows:

$$\theta_1(t) = 25t + t^2 - \frac{13}{9}t^3 \quad (12.4a)$$

$$\theta_1(t) = \frac{4335}{4} - 860t + \frac{919}{4}t^2 - \frac{39}{2}t^3 \quad (12.4b)$$

where first and second time-derivatives symbolizing joint velocity and acceleration respectively are given by

$$\dot{\theta}_1(t) = 25 + 2t - \frac{13}{3}t^2, \ddot{\theta}_1(t) = 2 - \frac{26}{3}t \quad (12.4c)$$

$$\dot{\theta}_2(t) = -860 + \frac{919}{2}t - \frac{117}{2}t^2, \ddot{\theta}_2(t) = \frac{919}{2} - 117t \quad (12.4d)$$

The plots for Eqs. (12.4a-d) are shown in Fig. 12.3(a). The discontinuities in joint acceleration denoted as “Accn.” in the figure is clearly visible.

In case one wants to have a continuous acceleration profile then some existing constraints have to be relaxed, say, the velocity at the intermediate point is not specified. In such a situation one can put the constraint of continuous velocity and acceleration at the intermediate point. With this, the first two coefficients of the first cubic polynomial joining the initial and the intermediate points can be obtained easily as $a_{01} = \theta_1(0) = 0$, $a_{11} = \dot{\theta}_1(0) = 25$, where the first subscript corresponds to the order of time, and the second one corresponds to the first trajectory. The other coefficients are obtained by solving six linear algebraic equations obtained from six constraints, which are the position of the intermediate point expressed using the first and second cubic polynomials, position and velocity of the final point using second polynomial, and the equal velocity and acceleration obtained from two polynomials. If they are expressed in the form of $\mathbf{Ax} = \mathbf{b}$, their elements can be given by

$$\mathbf{A} = \begin{bmatrix} 3 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 9 & 27 \\ 0 & 0 & 1 & 5 & 25 & 125 \\ 0 & 0 & 0 & 1 & 10 & 75 \\ 6 & 27 & 0 & -1 & -6 & -27 \\ 1 & 9 & 0 & 0 & -1 & -9 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} -10 \\ 45 \\ 90 \\ -25 \\ -25 \\ 0 \end{bmatrix} \quad (12.5a)$$

Solution of Eq. (12.5a) can be obtained easily using MATLAB as

- For the first cubic polynomial:

$$a_{21} = -\frac{49}{4} \text{ and } a_{31} = \frac{107}{36} \quad (12.5b)$$

- For the second cubic polynomial:

$$a_{02} = \frac{5415}{16}, a_{12} = -\frac{5015}{16}, a_{22} = \frac{1609}{16}, a_{32} = -\frac{153}{16} \quad (12.5c)$$

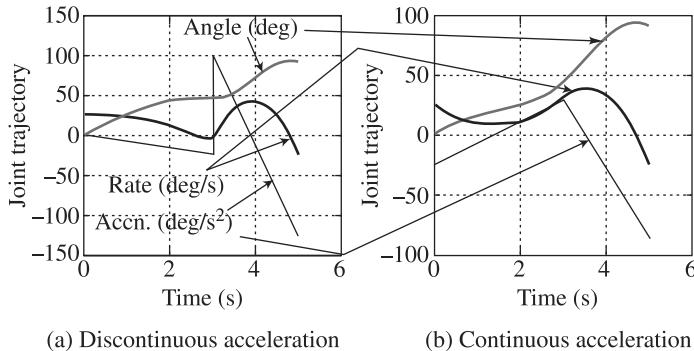


Fig. 12.3 Two cubic polynomials with one intermediate point

Figure 12.3 (b) shows the continuous acceleration at the intermediate point. In the above example, only one intermediate point was considered. If more intermediate points are considered, the analysis can be similarly extended. On the other hand, if velocity and acceleration are to be satisfied at the intermediate point, the order of the polynomial has to be increased to higher values. Additionally, if the accelerations at the initial and final points are also needed to be specified then the polynomial order has to be further increased, as explained next.

12.1.3 Quintic Polynomial

A typical cubic polynomial plot is shown in Fig. 12.3. In the third-order trajectory, as done in Eq. (12.2a), one can only specify a total of four conditions at the two ends of the trajectory. There is no control on acceleration. Hence, at the beginning and the end, acceleration values almost instantly increase or decrease. This may cause damage to the motors controlling the robot joints. Hence, one needs to use fifth-order or quintic polynomial to satisfy all six conditions, i.e., positions, velocity, and accelerations at the two ends in order to have a smooth trajectory. A quintic polynomial is defined by

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (12.6a)$$

whose first two time-derivatives are

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \quad (12.6b)$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \quad (12.6c)$$

The end-conditions are taken as

$$\theta(t_i) = \theta_0; \theta(t_f) = \theta_f; \dot{\theta}(t_i) = \dot{\theta}_0; \dot{\theta}(t_f) = \dot{\theta}_f, \text{ and } \ddot{\theta}(t_i) = \ddot{\theta}_0; \ddot{\theta}(t_f) = \ddot{\theta}_f \quad (12.7)$$

Although many smooth functions can satisfy the constraints given by Eq. (12.7), polynomials of time t are chosen here due to their ease of computation and simplicity of expression. The lowest order that can satisfy all the six conditions of Eq. (12.7) is

five. Substituting the conditions of Eq. (12.7) into Eqs. (12.6a-c), six equations with unknowns a_0, \dots, a_5 are obtained as

$$\theta(0) = \theta_0 = a_0 \quad (12.8a)$$

$$\theta(t_f) = \theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \quad (12.8b)$$

$$\dot{\theta}(0) = a_1 \quad (12.8c)$$

$$\dot{\theta}(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \quad (12.8d)$$

$$\ddot{\theta}(0) = 2a_2 \quad (12.8e)$$

$$\ddot{\theta}(t_f) = 3a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \quad (12.8f)$$

Equations (12.8a-f) can be put in a more compact matrix-vector form as $\mathbf{Ax} = \mathbf{b}$, where the 6×6 matrix \mathbf{A} , the 6-dimensional unknown vector \mathbf{x} containing the polynomial coefficients, and the known 6-dimensional vector \mathbf{b} of end-conditions are expressed below:

$$\mathbf{A} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}, \mathbf{x} \equiv \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}, \text{ and } \mathbf{b} \equiv \begin{bmatrix} \theta_0 \\ \theta_f \\ \dot{\theta}_0 \\ \dot{\theta}_f \\ \ddot{\theta}_0 \\ \ddot{\theta}_f \end{bmatrix} \quad (12.8g)$$

Solution to the set of linear algebraic equations, namely, Eq. (12.8g) can be obtained as $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, which yields the unknown coefficients, a_0, \dots, a_5 , as

$$a_0 = \theta_0; a_1 = \dot{\theta}_0; a_2 = \frac{1}{2}\ddot{\theta}_0 \quad (12.9a)$$

$$a_3 = \frac{1}{2t_f^3} \left[20(\theta_f - \theta_0) - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2 \right] \quad (12.9b)$$

$$a_4 = \frac{1}{2t_f^4} \left[30(\theta_0 - \theta_f) + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f + (3\ddot{\theta}_0 - 2\ddot{\theta}_f)t_f^2 \right] \quad (12.9c)$$

$$a_5 = \frac{1}{2t_f^5} \left[12(\theta_f - \theta_0) - 6(\dot{\theta}_f + \dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2 \right] \quad (12.9d)$$

Note that if $\ddot{\theta}_0 = \ddot{\theta}_f = 0$, and the relation shown in Fig. 12.2 is satisfied by $\theta_0, \theta_f, \dot{\theta}_0$, and $\dot{\theta}_f$, that is, if Eq. (12.3f) is valid then $a_5 = 0$ from Eq. (12.9d). This implies that only a fourth-order polynomial is required for $\theta(t)$. Accordingly, the coefficients a_3 and a_4 are given by

$$a_3 = \frac{1}{t_f^2}(\dot{\theta}_f - \dot{\theta}_0), \text{ and } a_4 = -\frac{1}{2t_f^3}(\dot{\theta}_f - \dot{\theta}_0) \quad (12.9e)$$

Using the combinations of fourth-order polynomials and straight lines, trajectories for various cases can be determined fairly easily. Two such cases will be explained in Sections 12.1.4–12.1.5.

Note that a quintic polynomial can also be used to plan a motion which passes through six specified points. In fact, any n specified intermediate points require $(n-1)$ -degree polynomials. In this case, however, no control exists over the velocity and acceleration. For the quintic polynomial passing through six points, the problem can be formulated in the form of $\mathbf{Ax} = \mathbf{b}$ where the 6×6 matrix \mathbf{A} and the 6-dimensional vector \mathbf{b} are given by

$$\mathbf{A} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & t_1^5 \\ 1 & t_2 & t_2^2 & t_2^3 & t_2^4 & t_2^5 \\ 1 & t_3 & t_3^2 & t_3^3 & t_3^4 & t_3^5 \\ 1 & t_4 & t_4^2 & t_4^3 & t_4^4 & t_4^5 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \end{bmatrix}, \text{ and } \mathbf{b} \equiv \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_f \end{bmatrix} \quad (12.10)$$

where $t_0 = 0$ was assumed. The above formulation is not desired for large size of n , as there will be oscillation in the trajectory. In addition, the problem is ill-conditioned and computationally expensive. An illustration is given in Example 12.2.

Example 12.2 Motion Planning of a Joint

The MATLAB program shown in Fig. 12.4 calculates the analytical expression of the coefficients given by Eq. (12.9a-d).

```
>> % MATLAB program for quintic polynomial coefficients
>> clear
>> syms tf tf2 tf3 tf4 tf5 th_0 th_f th_d_0 th_d_f th_dd_0 th_dd_f
>> tf2 = tf*tf; tf3 = tf2*tf; tf4 = tf3*tf; tf5 = tf4*tf;

>> A=[1 0 0 0 0 0; 1 tf tf2 tf3 tf4 tf5; ...
0 1 0 0 0 0; 0 1 2*tf 3*tf2 4*tf3 5*tf4;...
0 0 2 0 0 0; 0 0 2 6*tf 12*tf2 20*tf3]
>> b=[th_0;th_f;th_d_0;th_d_f;th_dd_0;th_dd_f]
>> x=A\b
>> pretty(simplify(x))
>> %MATLAB program ends
```

Fig. 12.4 MATLAB program to find the coefficients of quintic polynomial

Example 12.3 Trajectory of a Joint as Quintic Polynomial

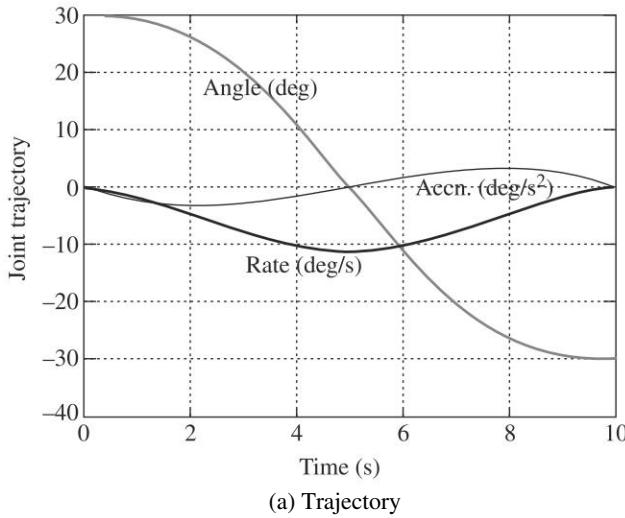
For a given set of initial and final conditions of joint angles, rates or velocities and accelerations of a robot joint, the known vector \mathbf{b} reads as follows:

$$\mathbf{b} = \begin{bmatrix} \frac{\pi}{6} & \frac{-\pi}{6} & 0 & 0 & 0 & 0 \\ \text{rad} & \text{rad} & \frac{\text{r/s}}{\text{s}} & \frac{\text{r/s}}{\text{s}} & \frac{\text{r/s}}{\text{s}^2} & \frac{\text{r/s}}{\text{s}^3} \end{bmatrix}^T \quad (12.11)$$

Corresponding quintic polynomial and their time derivatives are shown in Fig. 12.5(a), where $t_f = 10$ seconds, and time interval of $\Delta t = 0.01$ second. The MATLAB program used to generate Fig. 12.5(a) is given in Fig. 12.5(b). Note in Fig. 12.5(a) that the initial and final velocities and accelerations are zeros. This will ensure smooth starting and stopping of the joint motion.

Example 12.4 Comparison of a Cubic vs Quintic Polynomial

Using the first four conditions of Eq. (12.11), a cubic polynomial trajectory is also determined and compared in Fig. 12.6. Note that even though the quintic polynomial has no discontinuity in acceleration at the ends of the trajectory, it shoots in the middle of all three plots, i.e., angles, velocities, and accelerations. As the order of polynomial increases, such shooting up is more and more prominent.



(a) Trajectory

```
%% MATLAB program for quintic polynomial
clear
tf = 10;
th_0 = pi/6; th_f = -pi/6; th_d_0 = 0; th_d_f = 0; th_dd_0 = 0;
th_dd_f = 0;
tf2 = tf*tf; tf3 = tf2*tf; tf4 = tf3*tf; tf5 = tf4*tf;

A=[1 0 0 0 0 0; 1 tf tf2 tf3 tf4 tf5; ...
    0 1 0 0 0 0; 0 1 2*tf 3*tf2 4*tf3 5*tf4;...
    0 0 2 0 0 0; 0 0 2 6*tf 12*tf2 20*tf3];
b=[th_0;th_f;th_d_0;th_d_f;th_dd_0;th_dd_f]
x=A\b
delt = tf/10;
for i=1:11
    t (i) = (i-1)*delt
    th(i)=x(1)+x(2)*t(i)+x(3)*t(i)^2+x(4)*t(i)^3+...
    x(5)*t(i)^4+x(6)*t(i)^5;
    th_d(i)=x(2)+2*x(3)*t(i)+3*x(4)*t(i)^2+4*x(5)*t(i)^3+...
    5*x(6)*t(i)^4;
    th_dd(i)=2*x(3)+6*x(4)*t(i)+12*x(5)*t(i)^2+20*x(6)*t(i)^3;
end
plot(t,th,t,th_d,t,th_dd); grid on
xlabel ('time (sec)')
ylabel ('angle (rad), rate (r/s), and acceleration (r/s^2)')
%% MATLAB program ends
```

(b) MATLAB program

Fig. 12.5 A quintic polynomial trajectory for a robot joint

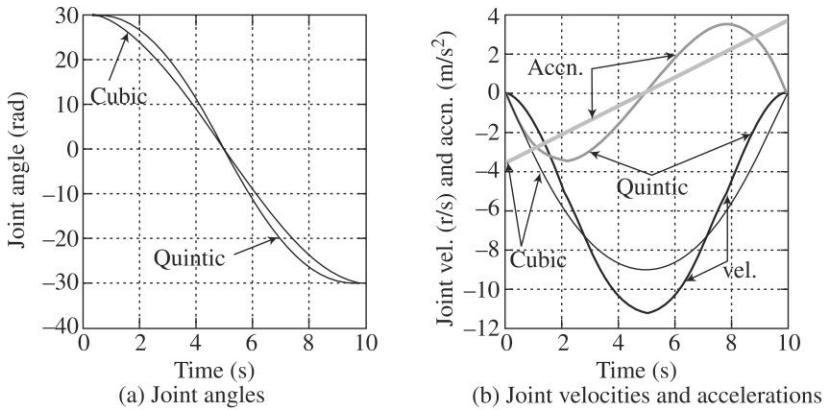


Fig. 12.6 Comparison of cubic and quintic polynomials

12.1.4 Trajectory with Given Initial and Final Points θ_0 and θ_f

As in Fig. 12.7, the trajectory is developed that starts from rest at θ_0 , passes through the phases of acceleration, constant velocity, and deceleration, and finally comes to a complete stop at θ_f . For this purpose, first the value of parameter Δ that denotes one-half of the acceleration and deceleration period is chosen. Second, the auxiliary points θ_{02} and θ_{f1} are determined using the following procedure: two points θ_{01} and θ_{f2} at time $t = \Delta$ and $t = t_f - \Delta$ are taken as $\theta_{01} = \theta_0$ and $\theta_{f2} = \theta_f$. Next, θ_{01} and θ_{f2} are connected by a straight line and θ_{02} and θ_{f1} are determined as the points on the straight line at time $t = 2\Delta$ and $t = t_f - 2\Delta$. The trajectory segments between θ_0 and θ_{02} , and θ_{f1} and θ_f are then determined using fourth-order polynomials such that their velocities coincide with the composition of straight lines connecting θ_0 - θ_{01} - θ_{f2} - θ_f at boundary points θ_0 , θ_{01} , θ_{f2} and θ_f , and such that their accelerations are zero at these boundary points. The trajectory segment between θ_{02} and θ_{f1} is specified by the straight line. Thus, the whole trajectory is given by the bold solid line in Fig. 12.7.

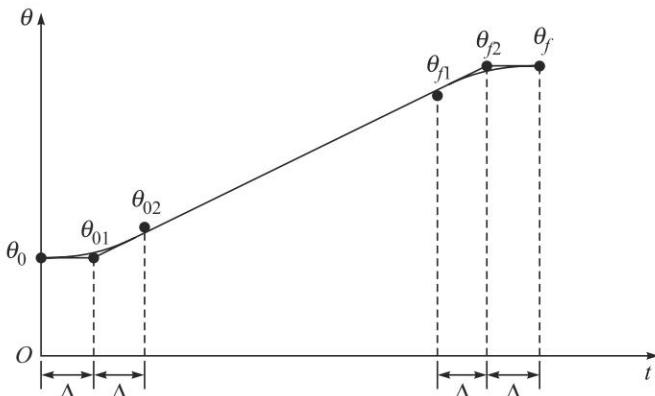


Fig. 12.7 Trajectory with acceleration, constant velocity, and deceleration

Example 12.5 Trajectory Planning for a Two-link Planar Arm

Applying the above method to the two-link planar arm shown in Fig. 12.8, a trajectory that starts from rest at the initial position of Fig. 12.8(a) and comes to a complete stop at the final position shown in Fig. 12.8(b) in 3.0 seconds is generated. Suppose we choose $\Delta = 0.3$ s. Then the first joint angle θ_1 is obtained as follows: First, choose $\theta_{01} = 90^\circ$ at time, $t = 0.3$ s, and $\theta_{f1} = 45^\circ$ at $t = 2.7$ s. Straight line equation is then obtained as

$$\theta_1 = -\frac{150}{8}t + \frac{675}{8} \quad (12.12a)$$

Hence, θ_{02} at $t = 0.6$ s, and θ_{f1} at $t = 2.4$ s are found as

$$\theta_{02} = \frac{675^\circ}{8}; \text{ and } \theta_{f1} = \frac{405^\circ}{8} \quad (12.12b)$$

Now, for the fourth-order polynomial between θ_0 and θ_{02} , the coefficients are obtained as

$$a_0 = 90^\circ; a_1 = a_2 = 0^\circ; a_3 = -\frac{1875^\circ}{36}; \text{ and } a_4 = \frac{9375^\circ}{216} \quad (12.12c)$$

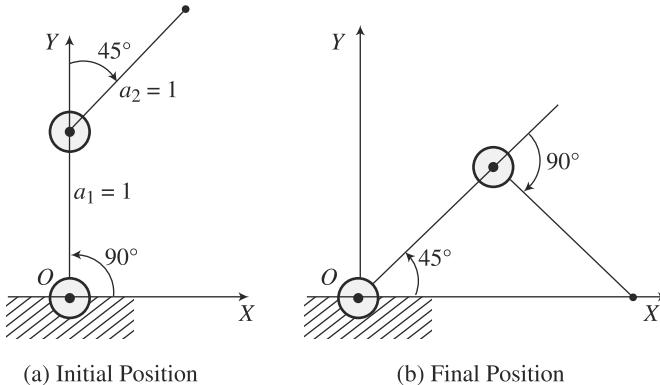


Fig. 12.8 Two-link planar arm

Note that, $\dot{\theta}_0 = \ddot{\theta}_0 = \ddot{\theta}_f = 0$, and $\theta_f - \theta_0 = t_f/2(\dot{\theta}_f + \dot{\theta}_0)$ conditions were used to find the coefficient values. For the interval, $0 < t \leq 0.5$, the joint trajectory is as follows:

$$\theta_1 = 90 - \frac{1875}{36}t^3 + \frac{9375}{216}t^4 \quad (12.12d)$$

For the straight line between θ_{02} and θ_{f1} , i.e., $0.6 < t \leq 2.4$, it can be easily obtained as

$$\theta_1 = \frac{675}{8} - \frac{150}{8}(t - 0.6) \quad (12.12e)$$

Finally, the fourth-order polynomial between θ_{f2} and θ_f i.e., $1.5 < t \leq 2.0$, can be written as

$$\theta_1(t) = a_0 + a_1(t - 2.4) + a_3(t - 2.4)^3 + a_4(t - 2.4)^4 \quad (12.12f)$$

whose coefficients are calculated as

$$a_0 = \frac{405^\circ}{8}; a_1 = -\frac{150^\circ}{8}; a_3 = \frac{625^\circ}{12}; \text{ and } a_4 = -\frac{3125^\circ}{72} \quad (12.12g)$$

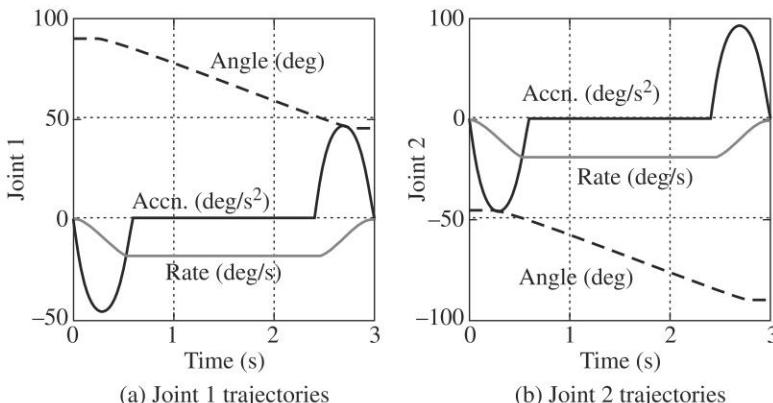
Hence, the final trajectory for θ_1 is written as

$$\theta_1(t) = \begin{cases} 90 - \frac{1875}{36}t^3 + \frac{9375}{216}t^4, & 0 \leq t \leq 0.6 \\ \frac{675}{8} - \frac{150}{8}(t-0.6), & 0.6 < t \leq 2.4 \\ \frac{405}{8} - \frac{150}{8}(t-2.4) + \frac{625}{12}(t-2.4)^3 - \frac{3125}{72}(t-2.4)^4, & 2.4 < t \leq 3.0 \end{cases} \quad (12.12h)$$

Similarly, the second joint angle θ_2 is obtained as

$$\theta_2(t) = \begin{cases} -45 - \frac{625}{12}t^3 + \frac{3125}{72}t^4, & 0 \leq t \leq 0.6 \\ -\frac{405}{8} - \frac{150}{8}(t-0.6), & 0.6 < t \leq 2.4 \\ -\frac{675}{8} - \frac{150}{8}(t-2.4) + \frac{625}{12}(t-2.4)^3 - \frac{3125}{72}(t-2.4)^4, & 2.4 < t \leq 3.0 \end{cases} \quad (12.13)$$

Figures 12.9(a-b) show the resulting end-point trajectories for the two joints of the two-link arm, whereas Fig. 12.9(c) shows the corresponding MATLAB program which generated the plots of Figs. 12.9(a-b). For better visualization, Fig. 12.10 shows the RoboAnalyzer screenshots during the animation of the two-link arm.



```
>>% MATLAB program for joint trajectories of 2-link arm
>> clear
>> tf = 3; N = 100; delt = tf/N
>> for i=1:N+1
>>t (i) = (i-1)*delt
>>if (t (i) <= 0.6)
>>    th_1 (i) = 90-1875/36*t(i)^3+9375/216*t(i)^4
>>    th_d_1 (i) = -1875/12*t(i)^2 + 9375/54*t(i)^3
>>    th_dd_1 (i) = -1875/6*t(i) + 9375/18*t(i)^2
>>    th_2 (i) = -45-625/12*t(i)^3+3125/72*t(i)^4
>>    th_d_2 (i) = -625/4*t(i)^2 + 3125/18*t(i)^3
```

(Contd.)

```

>> th_dd_2 (i) = -625/2*t(i) + 3125/6*t(i)^2
>> elseif (t(i) <= 2.4)
>> th_1 (i) = 675/8 - 75/4*(t(i)-0.6)
>> th_d_1 (i) = -75/4
>> th_dd_1 (i) = 0
>> th_2 (i) = - 405/8 - 75/4*(t(i)-0.6)
>> th_d_2 (i) = -75/4
>> th_dd_2 (i) = 0
>> else
>> th_1 (i) = 405/8-150/8*(t(i)-2.4)+625/12*(t(i)-2.4)^3 ...
>> -3125/72*(t(i)-2.4)^4
>> th_d_1 (i) = -150/8+625/4*(t(i)-2.4)^2-3125/18*(t(i)-2.4)^3
>> th_dd_1 (i) = 625/2*(t(i)-2.4)-3125/6*(t(i)-2.4)^2
>>
>> th_2 (i) = -675/8-150/8*(t(i)-2.4)+625/12*(t(i)-2.4)^3 ...
>> -3125/72*(t(i)-2.4)^4
>> th_d_2 (i) = -150/8+625/4*(t(i)-2.4)^2-3125/18*(t(i)-2.4)^3
>> th_dd_2 (i) = 625/2*(t(i)-2.4)-3125/6*(t(i)-2.4)^2
>> end
>> end
>> subplot (1,2,1), plot(t,th_1,t,th_d_1,t,th_dd_1), grid on
>> subplot (1,2,2), plot(t,th_2,t,th_d_2,t,th_dd_2), grid on
>> % MATLAB program ends

```

(c) MATLAB program

Fig. 12.9 Plot of joint trajectories for two-link arm

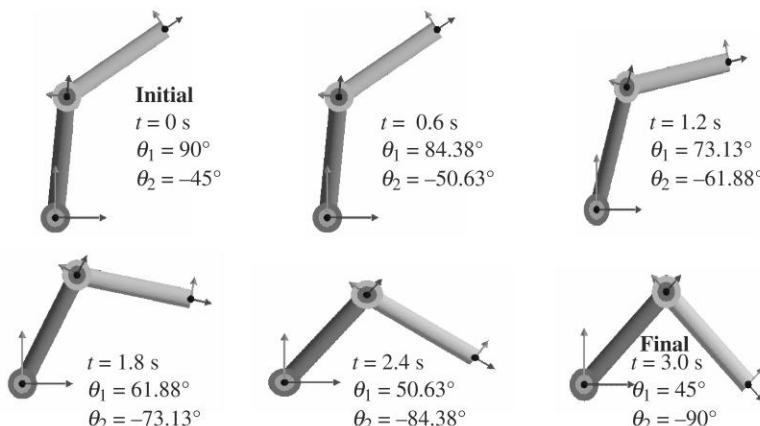


Fig. 12.10 RoboAnalyzer screenshots of two-link arm

12.1.5 Trajectory with Given Initial, Several Intermediate, and Final Points

Here, a trajectory is obtained that starts from rest at initial position θ_0 , passes through intermediate positions θ_1 and θ_2 , and stops at the final position θ_f . First, the case where the trajectory exactly passes through θ_1 and θ_2 is considered, where the auxiliary points $\theta_{02}, \theta_{11}, \theta_{12}, \dots, \theta_{f1}$ are specified, as in Fig. 12.11. The values at points θ_{01} and θ_0, θ_2 and θ_f are same. Moreover, θ_{02} and θ_{11} are on the straight line between θ_{01} and θ_1 . Similarly, points $\theta_{12}, \dots, \theta_{f1}$ are determined. One then can obtain the trajectory

between (θ_0, θ_{02}) , $(\theta_{11}, \theta_{12})$, ..., (θ_{f1}, θ_f) using fourth-order polynomials, and between $(\theta_{02}, \theta_{11})$, $(\theta_{12}, \theta_{21})$, and $(\theta_{22}, \theta_{f1})$ using straight lines. The final trajectory is shown with bold lines in Fig. 12.11. In case the trajectory has to pass exactly through θ_1 and θ_2 , the number of auxiliary points needs to be increased, as illustrated in Fig. 12.12(a). These auxiliary points are determined as follows: Referring to Fig. 12.12(b), θ_{i2} ($i = 1, 2$) is the midpoint between A , which is nothing but the intersection of the line of

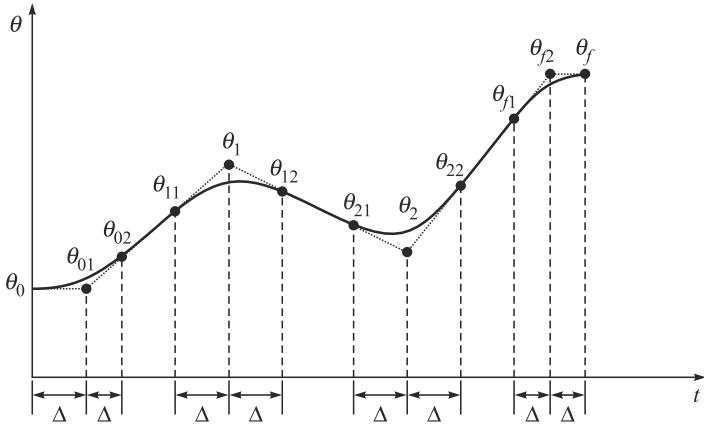
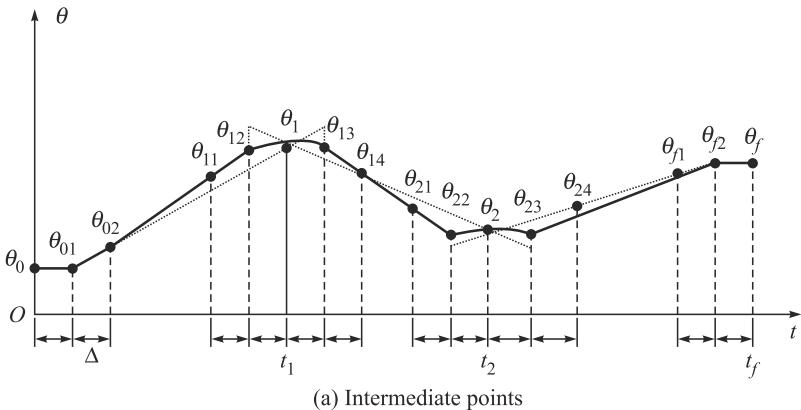
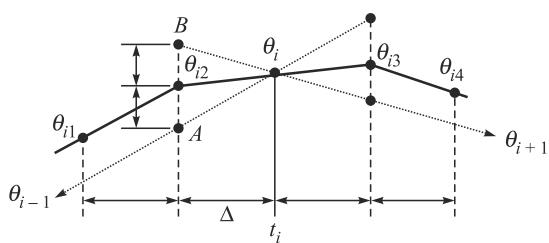


Fig. 12.11 Trajectory passing near the intermediate points



(a) Intermediate points



(b) Determination of points θ_{i2} and θ_{i3}

Fig. 12.12 Trajectory passing through the intermediate points

$t = t_i - \Delta$ with the straight line between θ_{i-1} (θ_{01} when $i = 1$) and θ_i , and B , i.e., the intersection of the line of $t = t_i - \Delta$ with straight line between θ_i and θ_{i+1} (θ_{fj} when $i = 1$). Point θ_{i3} is similarly obtained. Then segments between (θ_0, θ_{02}) , (θ_{11}, θ_1) , (θ_1, θ_{14}) , ..., (θ_{f1}, θ_f) are obtained using fourth-order polynomials, and those between $(\theta_{02}, \theta_{11})$, $(\theta_{14}, \theta_{21})$ and (θ_{24}, θ_f) using straight lines, as done above.

12.1.6 Linear Segments with Parabolic Blend (LSPB)

An alternate blend of trajectories which is more common in industries is a combination of parabolic arcs. It has typically three parts, namely, (1) constant acceleration, i.e., ramp velocity or parabolic position, (2) zero acceleration, i.e., constant velocity or linear position, and (3) constant deceleration, i.e., ramp velocity or parabolic position. Generally, acceleration and deceleration times are taken same. Hence, a symmetric trajectory results with respect to the center position, i.e., $(t_f - t_0)/2$, as indicated in Fig. 12.13. Such a trajectory is also referred as a *trapezoidal velocity profile* due to the trapezoidal shape of the velocity plot.

Considering the first part of the trajectory as parabolic in position level, the joint angle $\theta_1(t)$ between $t_0(=0) \geq t \leq t_c$ can be expressed as a second-order polynomial, namely,

$$\theta_1(t) = a_{01} + a_{11}t + a_{21}t^2 \quad (12.14a)$$

whose time derivatives are given as

$$\dot{\theta}_1(t) = a_{11} + 2a_{21}t \text{ and } \ddot{\theta}_1(t) = 2a_{21} \quad (12.14b)$$

The following three constraints are then imposed to compute the three unknown coefficients of Eq. (12.14a):

$$\theta_1(0) = \theta_0, \dot{\theta}_1(0) = 0, \dot{\theta}_1(t_c) = \dot{\theta}_c \quad (\text{Constant}) \quad (12.14c)$$

Using Eq. (12.14a-c), the coefficients are computed as $a_{01} = \theta_{01}$, $a_{11} = 0$, and $a_{21} = \ddot{\theta}_c/2$. The resulting equation for the trajectory during constant acceleration, i.e., for $0 \geq t \leq t_c$, is given by

$$\theta_1(t) = \theta_0 + \frac{1}{2}\ddot{\theta}_c t^2, \text{ where } \ddot{\theta}_c = \dot{\theta}_c/t_c \quad (\text{Constant}) \quad (12.15)$$

Next, the constant velocity segment, i.e., the straight-line trajectory for the joint denoted as $\theta_2(t)$ is given by

$$\theta_2(t) = a_{02} + a_{12}t \quad (12.16a)$$

Noting that for continuous trajectory $\theta_2(t_c) = \theta_1(t_c)$, and $\dot{\theta}_2(t_c) = \dot{\theta}_c$. The coefficients of the straight-line equation, Eq. (12.16a), are calculated as $a_{02} = \theta_{01} - \dot{\theta}_c t_c/2$ and $a_{12} = \dot{\theta}_c$. Hence, the equation of the straight-line trajectory for $t_c \geq t \leq t_f - t_c$ is obtained as

$$\theta_2(t) = \theta_0 + \dot{\theta}_c(t - t_c/2) \quad (12.16b)$$

Continuing in the same manner as the acceleration part, one can assume another parabolic function for the joint denoted with $\theta_3(t)$ as

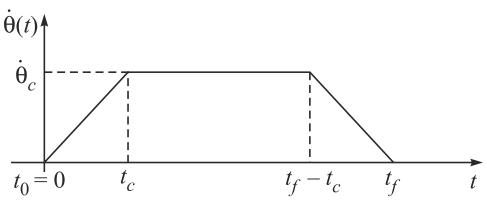


Fig. 12.13 Trapezoidal velocity profile

$$\theta_3(t) = a_{03} + a_{13}t + a_{23}t^2 \quad (12.17a)$$

whose time-derivatives are given as

$$\dot{\theta}_3(t) = a_{13} + 2a_{23}t \text{ and } \ddot{\theta}_3(t) = 2a_{23} \quad (12.17b)$$

where the constraints imposed are as follows:

$$\theta_3(t_f) = \theta_f, \dot{\theta}_3(t_f) = 0, \dot{\theta}_3(t_f - t_c) = \dot{\theta}_c \quad (12.17c)$$

Using Eq. (12.17a-c), the coefficients are computed as $a_{03} = \theta_f - \dot{\theta}_c t_f^2/2$, $a_{13} = \dot{\theta}_c t_f$ and $a_{23} = -\dot{\theta}_c/2$. The resulting equation for the trajectory during constant acceleration, i.e., for $t_f - t_c \geq t \leq t_f$, is given by

$$\theta_3(t) = \theta_f - \dot{\theta}_c(t_f - t)^2/2 \quad (12.18)$$

The three sections of the trajectory can now be summarized as

$$\begin{aligned} \theta(t) &= \theta_0 + \dot{\theta}_c t^2/2 && \text{for } 0 \leq t \leq t_c \\ &= \theta_0 + \dot{\theta}_c(t - t_c)/2 && \text{for } t_c \leq t \leq t_f - t_c \\ &= \theta_f - \dot{\theta}_c(t_f - t)^2/2 && \text{for } t_f - t_c \leq t \leq t_f \end{aligned} \quad (12.19)$$

Note that in order to have continuity in the position trajectory, the condition $\theta_2(t_c) = \theta_3(t_f - t_c)$ needs to be satisfied, which yields

$$\dot{\theta}_c = \frac{\theta_f - \theta_0}{t_f - t_c} \quad (12.20)$$

Figure 12.14 shows the plots for LSPB trajectory where Eq. (12.20) is satisfied for the numerical values of Example 12.6.

Example 12.6 Trajectory with LSPB or Trapezoidal Velocity Profile

To generate the plots for the trajectory of linear segments with parabolic blend or the trapezoidal velocity profile, the following numerical values are considered:

$$t_f = 5\text{s}, t_c = 2\text{s}, \theta_1(0) = 30^\circ, \theta_3(t_f) = -30^\circ, \dot{\theta}_c = 20^\circ/\text{s} \quad (12.21)$$

The plots are given in Fig. 12.14.

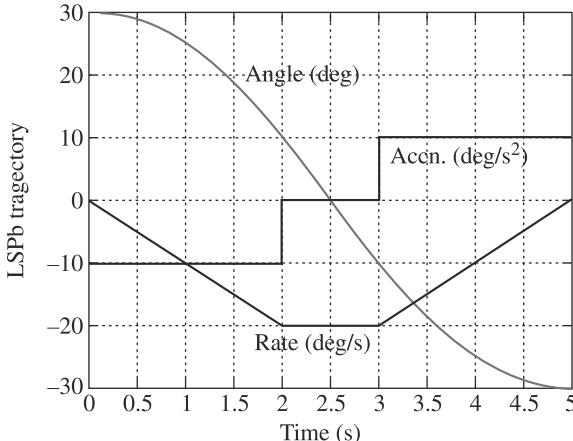


Fig. 12.14 Trajectory with LSPB or trapezoidal velocity profile

12.1.7 Cycloidal Trajectory

Without resorting to a higher-order polynomial, e.g., a quintic polynomial, which can have higher shoot-ups in the middle of a trajectory, as seen in Fig. 12.6, one can satisfy zero velocity and acceleration at the beginning and end of a trajectory, i.e., at the initial and final points, respectively, using a cycloidal trajectory. This can be given by

$$\theta(t) = \theta_0 + \frac{\theta_f - \theta_0}{T} \left[\Delta t - \frac{T}{2\pi} \sin \frac{2\pi\Delta t}{T} \right] \quad (12.22a)$$

where $T \equiv t_f - t_0$ and $\Delta t \equiv t - t_0$. Its time derivatives can be given by

$$\dot{\theta}(t) = \frac{\theta_f - \theta_0}{T} \left[1 - \cos \frac{2\pi\Delta t}{T} \right], \text{ and } \ddot{\theta}(t) = \frac{\theta_f - \theta_0}{T} \left[\frac{2\pi}{T} \sin \frac{2\pi\Delta t}{T} \right] \quad (12.22b)$$

A typical plot of Eq. (12.22a) and its time derivatives given by Eq. (12.22b) are shown in Fig. 12.15. Note that due to the smooth behavior of such a trajectory, it was used in Chapters 6, 8 and 9, besides implementing in RoboAnalyzer software as a default trajectory.

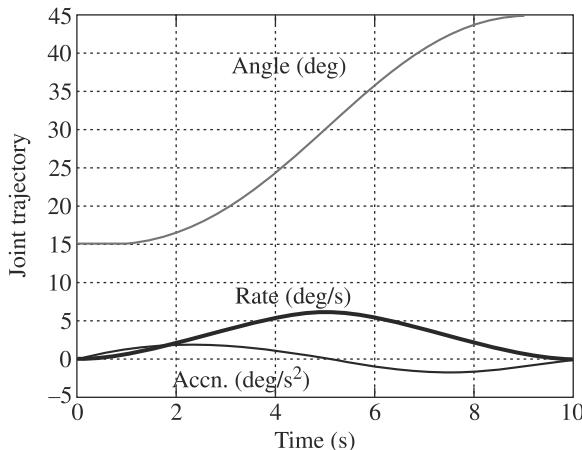


Fig. 12.15 Cycloidal trajectory of a joint motion

Example 12.7 Cycloidal Trajectory

For the generation of a cycloidal trajectory using Eqs. (12.22a-b), the following numerical inputs for the total time of $T = 10$ s are taken as

Initial and final joint angles: $\theta_0 = 15^\circ$, $\theta_f = 45^\circ$

Initial and final joint velocities: $\dot{\theta}_0 = \dot{\theta}_f = 0$ deg/s; $\ddot{\theta}_0 = \ddot{\theta}_f = 0$ deg/s²

Initial and final joint accelerations: $\ddot{\theta}_0 = \ddot{\theta}_f = 0$ deg/s²

Cycloid

A cycloid is defined as the curve traced by a point on a circle as it rolls on a straight line without slipping. It is a geometric entity used in gears and cams.

12.1.8 Spline Trajectory

A trajectory passing through N points can be obtained by connecting an $(N - 1)$ -degree polynomial. If N is large then the presence of oscillations as observed in Fig. 12.6 make the trajectory unacceptable, particularly, in robotic applications. For

Spline

The term *spline* comes from drafting, where splines are flexible strips guided by points on a paper. They are used to draw curves.

N intermediate points, one can choose $N - 1$ polynomials of lower order instead of considering one polynomial of order $N - 1$, as illustrated in Fig. 12.16. The polynomials can be chosen so that not only continuity in position is maintained but also in velocity and acceleration. One such could be the cubic polynomial treated in Section 12.1.2. Let any two consecutive intermediate points, say, (t_k, θ_k) and (t_{k+1}, θ_{k+1}) , are connected by the k th cubic polynomial, denoted with ρ_k , which can be given by

$$\rho_k = a_{0k} + a_{1k}(t - t_k) + a_{2k}(t - t_k)^2 + a_{3k}(t - t_k)^3 \quad (12.23)$$

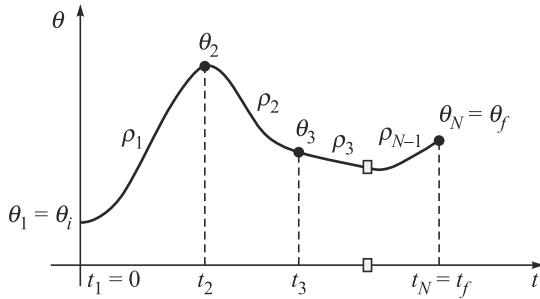


Fig. 12.16 Spline trajectory connecting intermediate points

for $t_k \leq t \leq t_{k+1}$. Thus, for $k = 1, \dots, N - 1$, splines, $4(N - 1)$ coefficients are to be evaluated. These coefficients will be evaluated in terms of the function values and its second derivatives, namely, θ_k 's and $\ddot{\theta}_k$'s, as done in Angeles (2003). Alternative method of evaluating the coefficients in terms of the velocities, i.e., $\dot{\theta}_k$'s, is also possible, as presented in Biagotti and Melchiorri (2008). The steps to find out the expressions of the coefficients are as follows:

1. First and Second Derivatives of Eq. (12.23) In this step, the expressions of the first and second derivatives of the k th cubic spline are obtained as

$$\dot{\rho}_k = a_{1k} + 2a_{2k}(t - t_k) + 3a_{3k}(t - t_k)^2 \quad (12.24a)$$

$$\ddot{\rho}_k = 2a_{2k} + 6a_{3k}(t - t_k) \quad (12.24b)$$

2. Expressions for the Coefficients a_{0k} , a_{1k} and a_{2k} Substituting $t = t_k$, in Eqs. (12.23) and (12.24a-b), one can find the coefficients of the polynomial as

$$a_{0k} = \theta_k, a_{1k} = \dot{\theta}_k, \text{ and } a_{2k} = \ddot{\theta}_k/2 \quad (12.25)$$

where $\theta_k \equiv \rho_k(t_k)$, $\dot{\theta}_k \equiv \dot{\rho}_k(t_k)$ and $\ddot{\theta}_k \equiv \ddot{\rho}_k(t_k)$.

3. Equality Conditions of Two Consecutive Splines at Supporting Point $k + 1$ Note that the two polynomials ρ_k and ρ_{k+1} at the supporting point $k + 1$, namely, at t_{k+1} , also referred as *knot* in some literatures, have same values, i.e.,

$$\rho_k(t_{k+1}) = \theta_{k+1}, \dot{\rho}_k(t_{k+1}) = \dot{\theta}_{k+1}, \text{ and } \ddot{\rho}_k(t_{k+1}) = \ddot{\theta}_{k+1} \quad (12.26)$$

in which, as per definitions after Eq. (12.25), θ_{k+1} , $\dot{\theta}_{k+1}$ and $\ddot{\theta}_{k+1}$ are equal to $\rho_{k+1}(t_{k+1})$, $\dot{\rho}_{k+1}(t_{k+1})$, and $\ddot{\rho}_{k+1}(t_{k+1})$, respectively.

4. Expression for the Coefficient a_{3k} In order to find the expression for the coefficients of the spline ρ_k in Eq. (12.23), the expression for $\ddot{\rho}_k(t_{k+1})$ from Eq. (12.24b) is equated with $\ddot{\theta}_{k+1}$, i.e.,

$$2a_{2k} + 6a_{3k}(t_{k+1} - t_k) = \ddot{\theta}_{k+1} \text{ or } 6a_{3k}\Delta t_k = \ddot{\theta}_{k+1} - \ddot{\theta}_k \quad (12.27)$$

where $\Delta t_k \equiv t_{k+1} - t_k$ and the expression for the coefficient a_{2k} is substituted from Eq. (12.25).

5. Rewrite the Expression for a_{1k} Since the objective here is to express the coefficients in terms of the position and its double derivatives, Eq. (12.23) is rewritten at t_{k+1} and equated with $\rho_{k+1}(t_{k+1}) \equiv \theta_{k+1}$, i.e.,

$$a_{0k} + a_{1k}\Delta t_k + a_{2k}\Delta t_k^2 + a_{3k}\Delta t_k^3 = \theta_{k+1} \quad (12.28a)$$

Substituting all the coefficients obtained above except a_{1k} , it is obtained as

$$a_{1k} = \frac{\Delta\theta_k}{\Delta t_k} - \frac{\Delta t_k}{6}(\Delta\ddot{\theta}_{k+1} + 2\Delta\ddot{\theta}_k) \quad (12.28b)$$

where $\Delta\theta_k \equiv \theta_{k+1} - \theta_k$ and $\Delta\ddot{\theta}_k \equiv \ddot{\theta}_{k+1} - \ddot{\theta}_k$, whereas Δt_k is defined after Eq. (12.27).

6. Continuity in First-Derivatives Continuity in first-derivatives between two polynomials ρ_{k-1} and ρ_k at t_k are formulated as

$$a_{0,k-1} + a_{1,k-1}(t_k - t_{k-1}) + a_{2,k-1}(t_k - t_{k-1})^2 + a_{3,k-1}(t_k - t_{k-1})^3 = \theta_k \quad (12.29a)$$

Substituting the coefficients from Eqs. (12.25) and (12.28b) into Eq. (12.29a), a linear equation relating $\ddot{\rho}_{k+1}$, $\ddot{\rho}_k$ and $\ddot{\rho}_{k-1}$ is obtained as

$$\Delta t_k \ddot{\theta}_{k+1} + 2(\Delta t_{k-1} + \Delta t_k)\ddot{\theta}_k + \Delta t_{k-1}\ddot{\theta}_{k-1} = 6\left(\frac{\Delta\theta_k}{\Delta t_k} - \frac{\Delta\theta_{k-1}}{\Delta t_{k-1}}\right) \quad (12.29b)$$

7. Formulation of Linear Equations in Unknowns $\ddot{\rho}_k$'s Equation (12.29b) is now combined for $k = 1, \dots, N - 1$, and written in a compact form as

$$\mathbf{A}\ddot{\theta} = \mathbf{B}\theta \quad (12.30a)$$

where $(N - 2) \times N$ matrices, \mathbf{A} and \mathbf{B} , and the N -dimensional vectors θ and $\ddot{\theta}$ are given by

$$\mathbf{A} \equiv \begin{bmatrix} \Delta t_1 & 2\Delta t_{12} & \Delta t_2 & 0 & \cdots & 0 & 0 \\ 0 & \Delta t_2 & 2\Delta t_{23} & \Delta t_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \Delta t_{N-3} & 2\Delta t_{N-3,N-2} & \Delta t_{N-2} & 0 \\ 0 & 0 & 0 & \cdots & \Delta t_{N-2} & 2\Delta t_{N-2,N-1} & \Delta t_{N-1} \end{bmatrix} \quad (12.30b)$$

$$\mathbf{B} \equiv \begin{bmatrix} \Delta t_1^{-1} & -\Delta t_{12}' & \Delta t_2^{-1} & 0 & \cdots & 0 & 0 \\ 0 & \Delta t_2^{-1} & -\Delta t_{23}' & \Delta t_3^{-1} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \Delta t_{N-3}^{-1} & -\Delta t_{N-3,N-2}' & \Delta t_{N-2}^{-1} & 0 \\ 0 & 0 & 0 & \cdots & \Delta t_{N-2}^{-1} & -\Delta t_{N-2,N-1}' & \Delta t_{N-1}^{-1} \end{bmatrix} \quad (12.30c)$$

$$\boldsymbol{\theta} \equiv [\theta_1 \ \dots \ \theta_N]^T \text{ and } \ddot{\boldsymbol{\theta}} \equiv [\ddot{\theta}_1 \ \dots \ \ddot{\theta}_N]^T \quad (12.30d)$$

where $\Delta t_{ij} \equiv \Delta t_i + \Delta t_j$ and $\Delta t'_{ij} \equiv \Delta t_i^{-1} + \Delta t_j^{-1}$.

8. Evaluation Strategy of the Unknowns $\ddot{\theta}_k$'s Note that the number of equations are less by two compared to the number of unknowns. Hence, two of the $\ddot{\theta}_k$'s can be assigned arbitrarily. One possibility is to assign the first and last accelerations, i.e., $\ddot{\theta}_1$ and $\ddot{\theta}_N$, as zeros. Such splines are called *natural splines*. In this case, no control is possible over the values of the velocities at the two ends. This is obvious from the plots of Fig. 12.17.

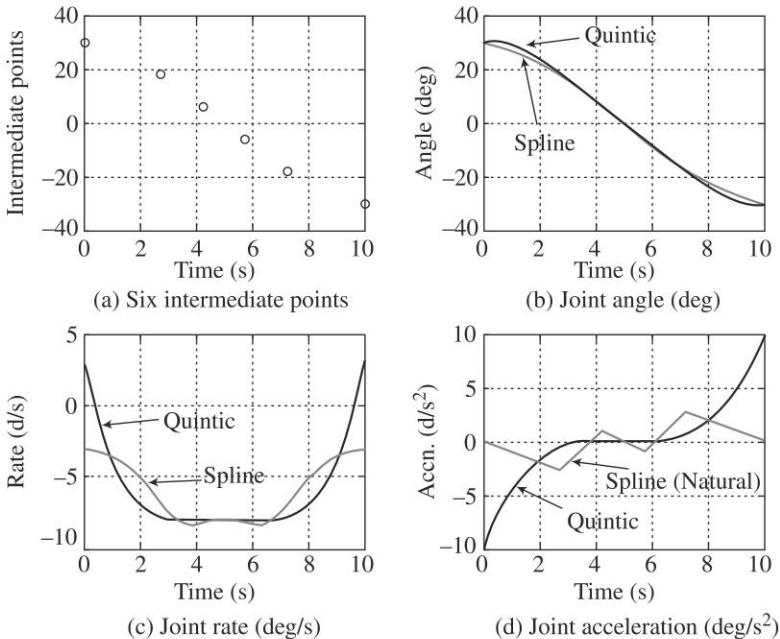


Fig. 12.17 Trajectory passing through six specified intermediate points

In order to have full control over the velocities and accelerations at the two ends, one can introduce the concept of virtual points (Siciliano et al., 2011) in the first and last polynomials and introduce the continuity in all three positions, velocities, and accelerations. This way the number of equations and number of polynomial coefficients are same to make Eq. (12.30a) a set of determined algebraic equations providing unique solutions.

Example 12.8 Spline and Quintic Trajectories through Six Points

Based on the formulation in Section 12.1.8, a spline trajectory is generated with six intermediate points, i.e., $N = 6$. Other inputs are as follows:

Time intervals:

$$t_0 (\equiv t_1) = 0; t_2 = 2.75; t_3 = 4.25; t_4 = 5.75; t_5 = 7.25; t_6 (\equiv t_N) = 10 \text{ sec} \quad (12.31\text{a})$$

Joint angles:

$$\theta_0 (\equiv \theta_1) = 30^\circ; \theta_k = [30 + 10(k - 1)] \text{ deg, for } k = 2 \dots 5; \theta_f (\equiv \theta_N) = -30^\circ \quad (12.31\text{b})$$

For the end conditions of zero accelerations, i.e., $\ddot{\theta}_0 (\equiv \ddot{\theta}_1) = \ddot{\theta}_f (\equiv \ddot{\theta}_N) = 0$, the reduced 4×4 matrix **A** and 4×6 matrix **B** are given by

$$\mathbf{A} \equiv \begin{bmatrix} \frac{17}{2} & \frac{3}{2} & 0 & 0 \\ \frac{3}{2} & 6 & \frac{3}{2} & 0 \\ 0 & \frac{3}{2} & 6 & \frac{3}{2} \\ 0 & 0 & \frac{3}{2} & \frac{17}{2} \end{bmatrix}; \quad \mathbf{B} \equiv \begin{bmatrix} \frac{24}{11} & -\frac{68}{11} & 4 & 0 & 0 & 0 \\ 0 & 4 & -8 & 4 & 0 & 0 \\ 0 & 0 & 4 & -8 & 4 & 0 \\ 0 & 0 & 0 & 4 & -\frac{68}{11} & \frac{24}{11} \end{bmatrix} \quad (12.32\text{a})$$

Corresponding joint-accelerations at the intermediate points are solved using a MATLAB program as

$$\ddot{\theta} \equiv [-30/11 \quad 10/11 \quad -10/11 \quad 30/11]^T \quad (12.32\text{b})$$

Using the joint accelerations at the supporting points, the coefficients for the five cubic polynomials are obtained as follows:

$$\begin{aligned} a_{01} &= 30, a_{11} = -137/44, a_{21} = 0, a_{31} = -20/121, \text{ for } 0 \leq t \leq 2.75; \\ a_{02} &= 18, a_{11} = -151/22, a_{21} = -15/11, a_{31} = 40/99, \text{ for } 2.75 < t \leq 4.25; \\ a_{03} &= 6, a_{11} = -181/22, a_{21} = 5/11, a_{31} = -20/99, \text{ for } 4.25 < t \leq 5.75; \\ a_{04} &= -6, a_{11} = -181/22, a_{21} = -5/11, a_{31} = 40/99, \text{ for } 5.75 < t \leq 7.25; \\ a_{05} &= -18, a_{11} = -151/22, a_{21} = 15/11, a_{31} = -20/121, \text{ for } 7.25 < t \leq 10 \end{aligned} \quad (12.33)$$

It is now simple to generate the spline curve shown in Fig. 12.17, which is compared with a fifth-order (quintic) polynomial, as explained in Section 12.1.3. Note the smaller variations in angles, rates, and accelerations, which are considered as advantages for spline trajectory with lower-order polynomials compared to a higher-order polynomials.

12.2 CARTESIAN SPACE PLANNING

When a trajectory between two configurations of the robot is generated by a joint variable scheme, as described in the preceding section, it is sometimes difficult to predict the motion of the end-effector in Cartesian space. There are also cases in which some specific end-

Joint Space vs. Cartesian Space

In joint space, joint angles and displacements are used to define a robot's configuration, whereas the Cartesian coordinates are used for the same purpose in Cartesian space.

effector trajectory is required. For example, in arc welding, the electrode should follow a seam precisely. In such other cases, one may wish to generate a trajectory in terms of the position and orientation of the end-effector. There are several ways to describe the position and orientation, i.e., the configuration, of the end-effector. Since the end-effector is a rigid body in the 3-dimensional Cartesian space it has six degree of freedom, six variables are required to prescribe the end-effector configuration. Once the way is chosen, given path points can be interpolated with a sequence of, say, a quintic, a cubic, or linear polynomials with parabolic blends (LSBP), as done for the joint trajectories in the above sections. A trajectory of an end-effector obtained this way is very simple and easy to visualize. Alternatively, end-effector motions satisfying some geometric features need to be expressed analytically. This can be done by introducing the concept of path primitives, as done in Section 12.3.

Example 12.9 Trajectory of Two-link Robot Arm in Cartesian Space

Let us obtain a trajectory for the problem in Example 12.5 not in terms of joint variables but in terms of end-effector position variables, say, $r \equiv [x, y]^T$. Since for the link length $a_1 = a_2 = 1$, the initial position r_0 is $\left[\frac{\sqrt{3}}{2}, \frac{3}{2}\right]^T$

and the final position r_f is $[1, 0]^T$, the trajectory $r(t) = [x(t), y(t)]^T$ is obtained similar to the joint-variable scheme, i.e.,

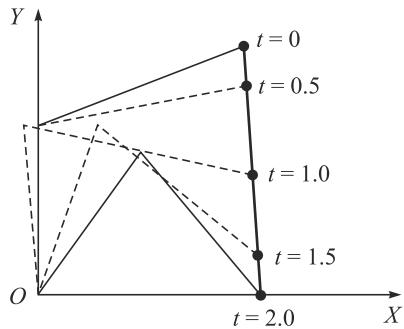


Fig. 12.18 End-point trajectory

$$x(t) = \begin{cases} \frac{\sqrt{3}}{2} + 4(2-\sqrt{3})(t^3 - t^4)/3 & 0 \leq t \leq 0.5 \\ (5\sqrt{3}+2)/12 + (2-\sqrt{3})(t-0.5)/3 & 0.5 < t \leq 1.5 \\ (10+\sqrt{3})/12 + (2-\sqrt{3})(t-1.5)/3 & 1.5 < t \leq 2 \end{cases} \quad (12.34a)$$

$$y(t) = \begin{cases} \frac{3}{2} - 4t^3 + 4t^4 & 0 \leq t \leq 0.5 \\ \frac{5}{4} - (t-0.5) & 0.5 < t \leq 1.5 \\ \frac{1}{4} - (t-1.5) + 4(t-1.5)^3 - 4(t-1.5)^4 & 1.5 < t \leq 2.0 \end{cases} \quad (12.34b)$$

Figure 12.18 shows the resulting endpoint trajectory.

As for the end-effector orientation concerned, assume that one uses Euler-angles or roll-pitch-yaw angles. Then the end-effector moves in a straight line with respect to these variables. However, the corresponding motion in the reference frame is difficult to understand intuitively, because it is given as a combination of these three rotations about three different time-varying axes.

Example 12.10 Trajectory for End-effector Orientation

Suppose that one wants to find a trajectory that transfers the end-effector orientation from rest at the initial orientation shown in Fig. 12.19(a) to the final rest orientation of Fig. 12.19(b) in time $t_f = 1$. Solve the problem by the single-axis rotation method. It is easily seen from Fig. 12.19 that the rotation matrix \mathbf{Q} representing the final orientation with respect to the initial one is

$$\mathbf{Q} \equiv \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (12.35)$$

Comparing Eqs. (5.37) and (12.35), the following are obtained:

- (i) Adding the diagonal elements, one gets

$$(e_x^2 + e_y^2 + e_z^2)C'\alpha + 3C\alpha = 0$$

Since \mathbf{e} is the unit vector $e_x^2 + e_y^2 + e_z^2 = 1$,

and $1 + 2C\alpha = 0$, i.e., $C\alpha = -1/2$. Hence, $\alpha = 120^\circ$.

- (ii) Subtracting the off-diagonal elements of matrix \mathbf{Q} in the following manner will provide the components of the unit vector \mathbf{e} (Angles, 2003), i.e.,

$$\frac{1}{2} \begin{bmatrix} q_{32} - q_{23} \\ q_{13} - q_{31} \\ q_{21} - q_{12} \end{bmatrix} \equiv \begin{bmatrix} \Delta q_x \\ \Delta q_y \\ \Delta q_z \end{bmatrix} \quad (12.36)$$

The above expression immediately results in $\mathbf{e} \equiv \Delta \mathbf{q} / \| \Delta \mathbf{q} \|$, where $\Delta \mathbf{q} \equiv [\Delta q_x \ \Delta q_y \ \Delta q_z]^T$ and $\| \Delta \mathbf{q} \| = \sqrt{3}$. Hence, the unit vector representing the axis of rotation is given by

$$\mathbf{e} \equiv [1/\sqrt{3}, \ 1/\sqrt{3}, \ 1/\sqrt{3}]^T \quad (12.37)$$

If one uses a fifth-order polynomial for interpolation in the form of Eq. (12.6a), the trajectory of this orientation is given by $[\mathbf{e}, \alpha]$, for $0 \leq t \leq 1$, where

$$\alpha = 1200t^3 - 1800t^4 + 720t^5 \quad (12.38a)$$

An example of the rotation matrix representing an intermediate orientation at $t = 0.5$ of this trajectory is as follows:

$$\mathbf{Q} \equiv \begin{bmatrix} 2/3 & -1/3 & 2/3 \\ 2/3 & 2/3 & 1/3 \\ -1/3 & 2/3 & 2/3 \end{bmatrix} \quad (12.38b)$$

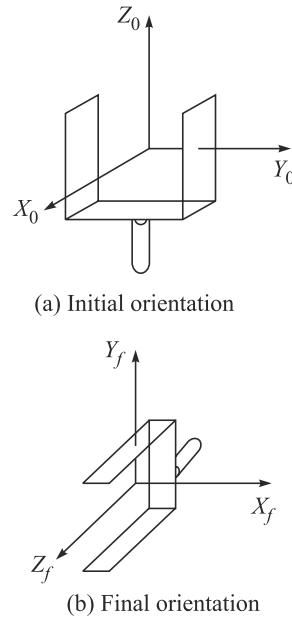


Fig. 12.19 Initial and final orientations of the end-effector

This orientation is shown in Fig. 12.20(a).

Next, the problem is solved by the double-axis rotation method as explained in Chapter 5. Euler angles for the initial and final orientations are $[\phi_0, \theta_0, \psi_0]^T = [0^\circ, 0^\circ, 0^\circ]^T$, and $[\phi_f, \theta_f, \psi_f]^T = [0^\circ, 90^\circ, 90^\circ]^T$. Hence, the trajectory is given by $\mathbf{Q}(\mathbf{e}, \theta)\mathbf{Q}(\mathbf{k}', \psi + \phi)$, for $0 \leq t \leq 1$,

$$\theta = \psi + \phi = 900t^3 - 1350t^4 + 540t^5 \quad (12.39a)$$

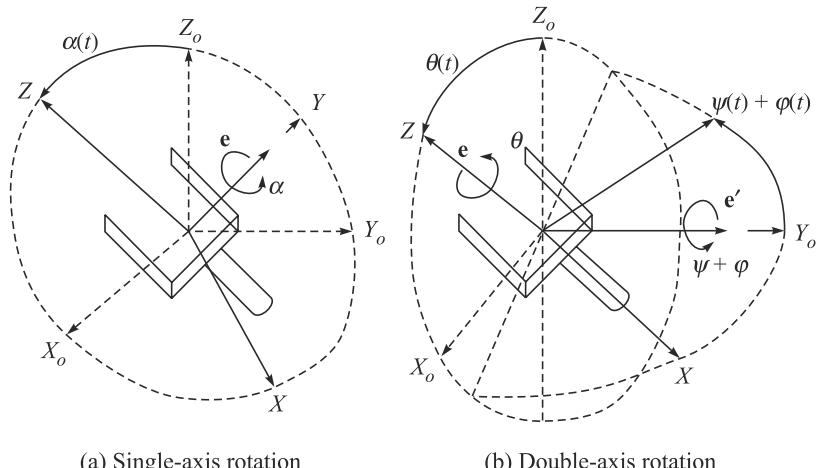


Fig. 12.20 Intermediate orientation methods

The rotation matrix \mathbf{Q} for an intermediate orientation at time $t = 0.5$ s for this case is calculated from $\mathbf{e}' \equiv [0, 1, 0]^T$ and $\theta = \psi + \phi = 45^\circ$ at $t = 0.5$ s, i.e.,

$$\mathbf{Q} = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/2 & 1/2 & 1/\sqrt{2} \end{bmatrix} \quad (12.39b)$$

This orientation is shown in Fig. 12.20(b). Comparing this with Fig. 12.20(a), one sees that the approach vector is always perpendicular to the Y_o axis, resulting in its shorter path. On the other hand, if the end-effector motion has to follow a prescribed trajectory of motion, this must be geometric analytically. It is then necessary to refer to motion primitives defining the geometric features of the path and time primitives defining the time law on the path itself, as presented next.

12.3 PATH PRIMITIVES

Let \mathbf{x} be the vector of operational space variables expressing *orientation* and *position* of the robot manipulator's end-effector. Generating a trajectory in the operational space means to determine a function $\mathbf{x}(t)$ taking the end-effector frame from the initial to the final location in a time t_f along a given path with a specific motion-time law. In order to define *path primitives*, parametric description of paths in space is

introduced. Assume \mathbf{p} be the 3-dimensional vector and $\mathbf{f}(s)$ is a continuous vector function, which is defined within the interval of s_i and s_f , as

$$\mathbf{p} = \mathbf{f}(s) \quad (12.40)$$

The sequence of values for \mathbf{p} with varying s is called *path* in space. Equation (12.40) defines the *parametric representation* of the path Γ and the scalar s is called *parameter*. While s increases, the point \mathbf{p} moves on the path. A path is closed if $\mathbf{p}(s_f) = \mathbf{p}(s_i)$. Otherwise, it is open. Let \mathbf{p}_i be a point on the open path Γ on which a direction is also fixed. The *path coordinate* s is then interpreted as the point \mathbf{p} , which is nothing but the length of the arc of Γ with extreme points \mathbf{p} and \mathbf{p}_i , if \mathbf{p} follows \mathbf{p}_i . Otherwise, the opposite is true if \mathbf{p} precedes \mathbf{p}_i . The point \mathbf{p}_i is the origin of the path coordinate, i.e., $s = 0$. From the above description, it is clear that for each value of s , there is a corresponding path point. Then the coordinate s can be used as a parameter to define the path Γ under consideration, i.e.,

$$\mathbf{p} = \mathbf{f}(s) \quad (12.41a)$$

The values of s are actually the sequence of path coordinates. Referring to Fig. 12.21 and Eq. (12.41a), let \mathbf{p} be a point corresponding to a given value of s . Except for special cases, the vector \mathbf{p} defines three unit vectors characterizing the path, namely, tangent, normal, and bi-normal vectors, while their directions depend on the path direction. The *tangent unit vector* denoted by \mathbf{e}_t is oriented along the direction dictated by s . The second unit vector is the *normal unit vector*, which is denoted by \mathbf{e}_n . This vector is oriented along the line intersecting \mathbf{p} at a right angle with \mathbf{e}_t and lies on what is called *osculating plane*, as shown in Fig. 12.21. The direction of \mathbf{e}_n is so that the path (Γ), in the neighborhood of \mathbf{p} with respect to the plane containing \mathbf{e}_t and normal to \mathbf{e}_n , lies on the same side of \mathbf{e}_n . The third unit vector is the *binormal unit vector* denoted by \mathbf{e}_b , which is defined as the cross-product of other two unit vectors, i.e., $\mathbf{e}_b = \mathbf{e}_t \times \mathbf{e}_n$. The triad $(\mathbf{e}_t, \mathbf{e}_n, \mathbf{e}_b)$ forms a right-handed coordinate frame shown in Fig. 12.21. Notice that such a definition of triad is useful for welding tasks, etc. It can be shown that the above three unit vectors are related to the path representation Γ which is a function of the path coordinate s . They are given by (Siciliano et al., 2011)

$$\mathbf{e}_t = \frac{d\mathbf{p}}{ds}; \mathbf{n} = \frac{1}{\left\| \frac{d^2\mathbf{p}}{ds^2} \right\|} \frac{d^2\mathbf{p}}{ds^2} \text{ and } \mathbf{e}_b = \mathbf{e}_t \times \mathbf{e}_n \quad (12.41b)$$

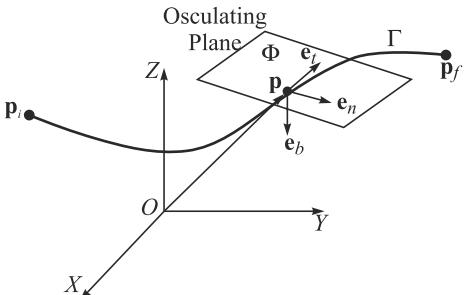


Fig. 12.21 Parametric representation of a path

Some examples below will demonstrate the concept better.

Example 12.11 Line in Space

Consider the line in space joining points \mathbf{p}_i and \mathbf{p}_f . The parametric representation of this path is

$$\mathbf{p}(s) = \mathbf{p}_i + \frac{s}{\delta_p} \boldsymbol{\delta}_p, \text{ where } \boldsymbol{\delta}_p \equiv \mathbf{p}_f - \mathbf{p}_i \quad (12.42a)$$

and $\delta_p \equiv \|\boldsymbol{\delta}_p\|$ is the magnitude of $\boldsymbol{\delta}_p$. Moreover, $\mathbf{p}(0) = \mathbf{p}_i$ and $\mathbf{p}(\delta_p) = \mathbf{p}_f$. Equation (12.41a) is the parametric representation of the path from \mathbf{p}_i to \mathbf{p}_f . Differentiating (12.42a) with respect to s , one gets the derivative of the path as

$$\frac{d\mathbf{p}}{ds} = \frac{\boldsymbol{\delta}_p}{\delta_p}; \text{ and } \frac{d^2\mathbf{p}}{ds^2} = \mathbf{0} \quad (12.42b)$$

The slope of the path is then given by $d\mathbf{p}/ds$, which defines the tangent vector \mathbf{e}_t . However, since the double-derivative is zero, the normal vector \mathbf{e}_n vanishes. Hence, no unique frame $(\mathbf{e}_t, \mathbf{e}_n, \mathbf{e}_b)$ exists.

Example 12.12 Circle in Space

Take a circle in space. Its significant parameters referring to Fig. 12.22 are as follows:

- the unit vector representing the axis of the circle, denoted with \mathbf{e} ,
- the position vector of a point along the axis, which is denoted with \mathbf{d} , and
- the position vector of a point on the circle, denoted with vector \mathbf{p} .

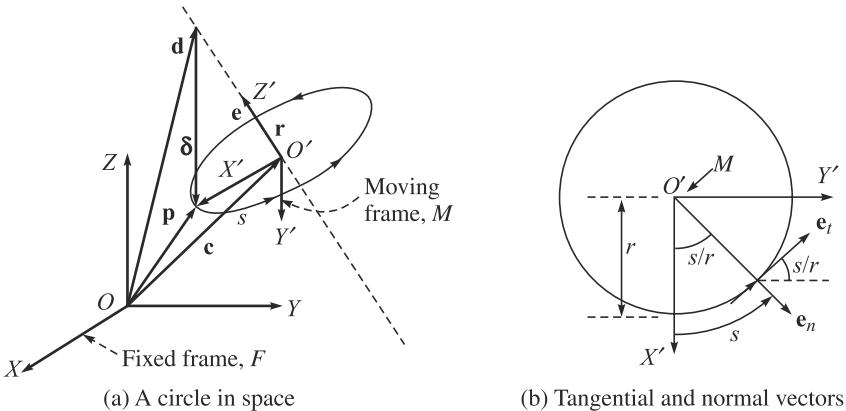


Fig. 12.22 Parametric representation of a circle in space

With the above parameters defined, one can express the position vector of the center of the circle, namely, the vector \mathbf{c} , as

$$\mathbf{c} = \mathbf{d} + (\boldsymbol{\delta}^T \mathbf{e}) \mathbf{e} \quad (12.43)$$

where $\boldsymbol{\delta} \equiv \mathbf{p} - \mathbf{d}$, and the condition $|\boldsymbol{\delta}^T \mathbf{e}| < \|\boldsymbol{\delta}\|$ must be satisfied.

Based on Eq. (12.43), it is possible to express a parametric representation of the circle. For a suitable choice of the reference frame, say, $O'-X'Y'Z'$ of Fig. 12.22, where O' coincides with the center of the circle, the axis X' is oriented along the direction of the vector $\mathbf{p} - \mathbf{c}$, axis Z' is oriented along \mathbf{e} , and the axis Y' is chosen so as to complete a right-handed frame. In this reference frame, denoted as M , the parametric representation of the circle is given by

$$[\mathbf{p}(s)]_M \equiv [r \cos(\bar{s}) \quad r \sin(\bar{s}) \quad 0]^T, \text{ where } \bar{s} \equiv s/r \quad (12.44)$$

and r is the radius of the circle given by $r = \|\mathbf{p} - \mathbf{c}\|$. The path representation of any point on the circle in the fixed frame F is then obtained as

$$[\mathbf{p}(s)]_F = [\mathbf{c}]_F + \mathbf{Q}[\mathbf{p}(s)]_M \quad (12.45a)$$

where $[\mathbf{c}]_F$ is expressed in the frame F , i.e., $O\text{-XYZ}$, and \mathbf{Q} is the rotation matrix relating the frames F and M , which can be written from the knowledge of Chapter 5 as

$$\mathbf{Q} \equiv [\mathbf{x}' \mathbf{y}' \mathbf{z}'] \quad (12.45b)$$

in which \mathbf{x}' , \mathbf{y}' , \mathbf{z}' indicate the unit vectors of frame M expressed in the frame F . Differentiating Eq. (12.45a) with respect to s , one gets the following:

$$\frac{d\mathbf{p}}{ds} = -\sin(\bar{s})\mathbf{x}' + \cos(\bar{s})\mathbf{y}'; \text{ and } \frac{d^2\mathbf{p}}{ds^2} = -\frac{1}{r}[\cos(\bar{s})\mathbf{x}' + r\sin(\bar{s})\mathbf{y}'] \quad (12.46)$$

12.4 CARTESIAN TRAJECTORIES

Based on the concept of path primitives, it is now possible to specify a position or orientation trajectory based on their parametric representations. They are presented next.

12.4.1 Position

Let $\mathbf{p} = \mathbf{f}(s)$ be the 3-dimensional vector representing a Cartesian path in parametric form, i.e., as a function of the path coordinate s . The origin of the end-effector frame is assumed to move from \mathbf{p}_i to \mathbf{p}_f in a time t_f . The values of s are then $s = 0$ at $t = 0$ and $s = s_f$ (path length) at $t = t_f$. In order to find an analytical function in $s(t)$, any of the trajectory-generation techniques explained in Section 12.1 for joint motions, for example, cubic, quintic polynomials, etc., can be used. Note that the velocity of point \mathbf{p} is given as

$$\dot{\mathbf{p}} = \dot{s} \frac{d\mathbf{p}}{ds} = \dot{s} \mathbf{e}_t \quad (12.47)$$

where \mathbf{e}_t is the tangent vector to the path at the point \mathbf{p} in Eq. (12.41b). Then, \dot{s} is the magnitude of the vector $\dot{\mathbf{p}}$ and \mathbf{e}_t is the direction.

Example 12.13 Velocity and Acceleration of the Line Path

Consider the path of Example 12.11 connecting points between \mathbf{p}_i and \mathbf{p}_f . The parametric representation of this path is given by Eq. (12.41a). Velocity and acceleration of \mathbf{p} can then be easily computed using the rule of differentiation as

$$\dot{\mathbf{p}} = \frac{\dot{s}}{\delta_p} \boldsymbol{\delta}_p = \dot{s} \mathbf{e}_t; \text{ and } \ddot{\mathbf{p}} = \frac{\ddot{s}}{\delta_p} \boldsymbol{\delta}_p = \ddot{s} \mathbf{e}_t \quad (12.48)$$

where vector $\boldsymbol{\delta}_p$ is defined after Eq. (12.42a).

Example 12.14 Velocity and Acceleration of the Circular Path

Here, consider the circle of Example 12.12. From the parametric representation derived above and in view of Eq. (12.46), velocity and acceleration of the point \mathbf{p} on the circle are

$$\dot{\mathbf{p}} = \dot{s}[-\sin(\bar{s})\mathbf{x}' + \cos(\bar{s})\mathbf{y}'] \quad (12.49a)$$

and $\ddot{\mathbf{p}} = \ddot{s}[-\sin(\bar{s})\mathbf{x}' + \cos(\bar{s})\mathbf{y}'] - \dot{s}^2/r[\cos(\bar{s})\mathbf{x}' + \sin(\bar{s})\mathbf{y}'] \quad (12.49b)$

Notice that according to Fig. 12.22(b), the above velocity vector $\dot{\mathbf{p}}$ is aligned with \mathbf{e}_t , whereas the acceleration vector is a combination of two contributions, namely, the first one is aligned with \mathbf{e}_t and represents the tangential acceleration, while the second one is aligned with \mathbf{e}_n and represents the centripetal acceleration.

12.4.2 Orientation

For the orientation of an end-effector, typically, it is specified using a time-varying rotation matrix of the frame attached to the end-effector. It is well known that the three columns of the rotation matrix are nothing but those unit vectors associated to the end-effector frame, namely, \mathbf{e}_t , \mathbf{e}_n , and \mathbf{e}_b . To generate a rotation trajectory, however, it is not convenient to refer the rotation matrix in terms of the initial and final representations of the unit vectors \mathbf{e}_t , \mathbf{e}_n , and \mathbf{e}_b as orthonormality condition may not be guaranteed during the intermediate time steps. In view of the above difficulty, one may prefer to use the angle triplet $\boldsymbol{\phi} \equiv [\phi, \theta, \psi]^T$, e.g., the Euler ZYZ angles as a function of time law. Usually, $\boldsymbol{\phi}$ moves from its initial value $\boldsymbol{\phi}_i$ to its final value $\boldsymbol{\phi}_f$. A cubic polynomial or a linear segment with parabolic blends (LSBP) can be used. Accordingly, velocity and acceleration profiles can be expressed for each scalar value of the angles in $\boldsymbol{\phi}$ as

$$\dot{\boldsymbol{\phi}} = \boldsymbol{\phi}_i + \frac{\dot{s}}{\delta_i} \boldsymbol{\delta}_\phi; \ddot{\boldsymbol{\phi}} = \frac{\ddot{s}}{\delta_i} \boldsymbol{\delta}_\phi \text{ and } \ddot{\boldsymbol{\phi}} = \frac{\ddot{s}}{\delta_i} \boldsymbol{\delta}_\phi \quad (12.50)$$

where $\boldsymbol{\delta}_\phi \equiv \boldsymbol{\phi}_f - \boldsymbol{\phi}_i$ and δ_i is the difference in angle $i = \phi, \theta$, or ψ . The three unit vectors \mathbf{e}_t , \mathbf{e}_n , and \mathbf{e}_b associated to the end-effector frame are then computed with reference to the rotation expression using Euler angles, as given in Eq. (5.31e).

An alternative way to generate a trajectory for orientation can be from the expression using the single-axis of rotation presented in Chapter 5. Given two coordinate frames in the Cartesian space with the same origin and different orientation, it is always possible to determine a unit vector \mathbf{e} so that the second frame is obtained from the first frame by a fixed rotation α about the axis of rotation parallel to the unit vector \mathbf{e} . Let \mathbf{Q}_i and \mathbf{Q}_f denote respectively the rotation matrices of the initial frame $O_i-X_iY_iZ_i$ and the final frame $O_f-X_fY_fZ_f$, both with respect to the base frame. The rotation matrix between the two frames is then computed by recalling that $\mathbf{Q}_f = \mathbf{Q}_i \mathbf{Q}_{if}$. Hence, the matrix \mathbf{Q}_{if} is obtained as

$$\mathbf{Q}_{if} = \mathbf{Q}_i^T \mathbf{Q}_f = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \quad (12.51)$$

If the matrix \mathbf{Q} represents rotation from \mathbf{Q}_i to \mathbf{Q}_f then $\mathbf{Q}(0) = \mathbf{Q}_i$ and $\mathbf{Q}(t_f) = \mathbf{Q}_f$. Hence, the matrix \mathbf{Q}_{if} is the rotation about a fixed axis in space which is parallel to unit vector \mathbf{e} by an angle of rotation α . They can be computed using Example 12.10 as

$$\alpha = \cos^{-1} \left(\frac{q_{11} + q_{22} + q_{33} - 1}{2} \right), \text{ and } \mathbf{e} \equiv \frac{1}{2 \sin \alpha} \begin{bmatrix} q_{32} - q_{23} \\ q_{13} - q_{31} \\ q_{21} - q_{12} \end{bmatrix} \quad (12.52)$$

for $\sin \alpha \neq 0$. The matrix \mathbf{Q}_{if} is the function of the angle α . It is sufficient to assign a time law to angle α , similar to a single joint motion generation explained in Sections 12.1 with $\alpha(0) = \alpha_i$ and $\alpha(t_f) = \alpha_f$. The values of $\alpha(t)$ are then used to compute the components of the unit vector \mathbf{e} , namely, e_x, e_y, e_z . Any intermediate rotation matrix can be given by $\mathbf{Q}(t) = \mathbf{Q}_i \mathbf{Q}_{if}(\alpha)$, which describes the end-effector orientation with respect to the fixed frame as a function of time. Once a path and a trajectory have been specified in the Cartesian space in terms of \mathbf{p} and ϕ or \mathbf{Q} , inverse kinematics can be performed to evaluate the corresponding joint trajectories in terms of the joint angles $\boldsymbol{\theta}(t)$.

12.5 POINT-TO-POINT VS. CONTINUOUS PATH PLANNING

Irrespective of the type of joint-space or Cartesian-space motion planning, one comes across two types of trajectories, namely, *point-to-point* and *continuous*. The former is typically applicable for pick-and-place operations, whereas the latter is more applicable for applications like welding, etc.

In the *Point-To-Point* (PTP) motion of a robot, it has to move from an initial to a final joint configuration in a given time t_f . Here, the actual end-effector trajectory is not important. The motion-planning algorithm should generate a trajectory which may be capable of optimizing some performance criteria when each joint is moved from one position to another. The types of trajectories presented in Sections 12.1 can be used to generate a PTP motion.

In several applications like welding of two pipes, on the other hand, the path needs to be described in terms of a number of points which are typically greater than two. A set of intermediate positions are set for lifting off and setting down a work-piece so that reduced velocities are obtained with respect to direct transfer of the object. For more complex applications, it is desirable to specify a series of points so as to guarantee better monitoring of the executed trajectories. The points need to be specified more densely in those sections of the trajectory where obstacles have to be avoided or a high path curvature is expected. The problem is to generate a trajectory with N points either in joint or Cartesian space.

Application of PTP Motion

Where pick-and-place operations are required, e.g., picking up a workpiece from a conveyor belt and place on a machine table for processing.

Application of CP Motion

Where continuous motion of the end-effector along a path is required, e.g., welding two pipes.

SUMMARY

In this chapter, several motion planning techniques are presented. Joint and Cartesian space trajectories to be followed by the joint actuators and the end-effector, respectively, are discussed. Path primitives are also explained when the robot has to maintain certain relationships with the trajectory it has to follow. Strategies for both point-to-point and continuous motion are also mentioned.

EXERCISES

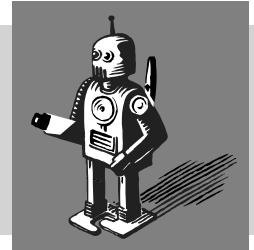
- 12.1** It is desired to have the first joint of a six-axis robot to move from the initial, $\theta_0 = 15^\circ$, to a final position, $\theta_f = 75^\circ$, in 3 seconds using a cubic polynomial.
- Determine the trajectory.
 - Calculate joint angle at 2 seconds.
 - Comment on its end point velocities and accelerations.
 - Compare the present result with those obtained in Example 12.1.
- 12.2** Repeat the problem in Exercise 12.1 but this time initial acceleration and final deceleration are specified as $5^\circ/\text{s}^2$.
- 12.3** A two-degree-of-freedom planar robot is to follow a straight line between the start (3,10) cm and the end (8,15) cm points of the motion segment. Find the joint variable for the robot if the path is divided into 5 segments. Assume each link is 10 cm long.
- 12.4** Find the coefficients of two cubic polynomials which are connected in a two-segment spline with continuous acceleration at the intermediate via points.
- 12.5** A single cubic trajectory is given by
- $$\theta(t) = 10 + 90t^2 - 60t^3 \quad (12.53)$$
- which is used between $t = 0$ and 1 s. What are the starting and final positions, velocities, and accelerations?
- 12.6** Find the coefficients of a cubic polynomial
- $$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (12.54)$$
- when $\theta(0)$, $\dot{\theta}(0)$, $\ddot{\theta}(0)$ and $\theta(t_f)$ are specified.
- 12.7** A rotary joint moves from -15° to $+45^\circ$ in 4 seconds. Determine a smooth polynomial trajectory if the initial and final velocities and accelerations are zero. What is the order of the polynomial?
- 12.8** For a 3-degree-of-freedom manipulator, design a linear trajectory with parabolic blends. The initial and final orientations of the end-effector, denoted by \mathbf{Q}_i and \mathbf{Q}_f , respectively, are specified using the following matrices:
- $$\mathbf{Q}_i \equiv \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \text{ and } \mathbf{Q}_f \equiv \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (12.55)$$
- 12.9** The velocity profile of a trajectory is trapezoidal with constant acceleration segment for 0.5 sec duration and constant velocity of $10^\circ/\text{s}$. Determine the parameters of the smooth trajectory for interpolating the time sequence of position with this type of trajectory.
- 12.10** What is the ill effect of higher degree polynomial to connect N points?
- 12.11** Mention the benefits of spline functions.

MATLAB BASED EXERCISES

- 12.12** Plot positions, velocities and accelerations for the trajectory given in Exercise 12.5. Use 100 steps between $t = 0$ and 1 s.
- 12.13** For Exercise 12.6, find the numerical values of the coefficients by formulating the problem as a linear algebraic equations in the form of $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x} \equiv [a_0 \ a_1 \ a_2 \ a_3]^T$. Take $\theta(0) = 0$, $\dot{\theta}(0) = 0$, $\ddot{\theta}(0) = 0$, $\theta(t_f) = 30^\circ$ and $t_f = 5$ sec. Plot $\theta(t)$, $\dot{\theta}(t)$ and $\ddot{\theta}(t)$.
- 12.14** Determine the coefficients of a quintic polynomial by formulating the problem as a linear algebraic equations in the form of $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x} \equiv [a_0 \ a_1 \ a_2 \ a_3 \ a_4 \ a_5]^T$, and $\theta(0) = 0$, $\theta(t_f) = 30^\circ$, $\dot{\theta}(0) = \ddot{\theta}(0) = \dot{\theta}(t_f) = \ddot{\theta}(t_f) = 0$, $t_f = 5$ sec. Plot $\theta(t)$, $\dot{\theta}(t)$ and $\ddot{\theta}(t)$.
- 12.15** Plot the position, velocity, and acceleration plots of Exercise 12.7.
- 12.16** Write a program to calculate tangent, normal, and bi-normal unit vectors of a point on an ellipse.

13

Control Hardware and Robot Programming



In this chapter, the interface between the human user and the robot will be considered. It is normally the control hardware of the robot to which a user communicates using a robot-programming language. A number of algorithms associated with the kinematics, dynamics, and control of robotic systems has been presented in earlier chapters. Many of these calculations need to be performed not only efficiently but have to be effectively communicated to the servo controllers of the robot joints. Hence, their architecture has to be understood. Besides, a robot operator or a user communicates to a robot using a robot-programming language. They are explained towards the end of this chapter.

13.1 CONTROL CONSIDERATIONS

The *control system* of a robot is to supervise the activities of a robot. It should have the following characteristics:

- **Manipulation ability:** Ability to move objects within the workspace
- **Sensory ability:** Able to obtain information on the state of the system and working environment
- **Intelligence:** Ability to modify the system behavior based on the sensor information
- **Data-processing ability:** Capability of storing, elaborating, and providing data on system activity

The above objectives can be achieved through an effective implementation of a *functional architecture*, which is nothing but the supervision of several activities arranged in a hierarchical structure. The term *hierarchical* is equivalent to the command structure of the military, where the generals pass orders to the officers who pass orders to the sergeants, and in turn orders are passed to the troops. In such hierarchy, control is supposed to flow from top to bottom, and reporting is intended to flow up from the bottom. The lower levels are meant for physical motion execution, whereas the higher levels are responsible for logical action planning. The levels are interconnected with the flow of measurement and/or results of action data from bottom to top, whereas the directions are transmitted from top to bottom.

13.1.1 Function Modules

In order to achieve the above control architecture, three functional modules at each level are assigned. They are as follows:

- The first *sensory module* is devoted to sensory data management. These modules acquire, elaborate, correlate, and integrate sensory data in time and space so that the system data and environment characteristics are recognized.
- The second *modeling module* is to provide knowledge of the relevant world. These modules contain models based on prior knowledge of the system and environment, which are updated with the information coming from the sensory modules.
- The third *decision module* is to decide the action policy, i.e., to decompose high-level tasks into low-level actions. Such task decomposition is in terms of both the time of sequential actions and the space of concurrent actions. Each decision module is responsible to the functions concerning management of elementary action assignments, task planning, and execution. A decision module determines the level of hierarchy and the associated functions of the modeling and sensory modules operating at that level.

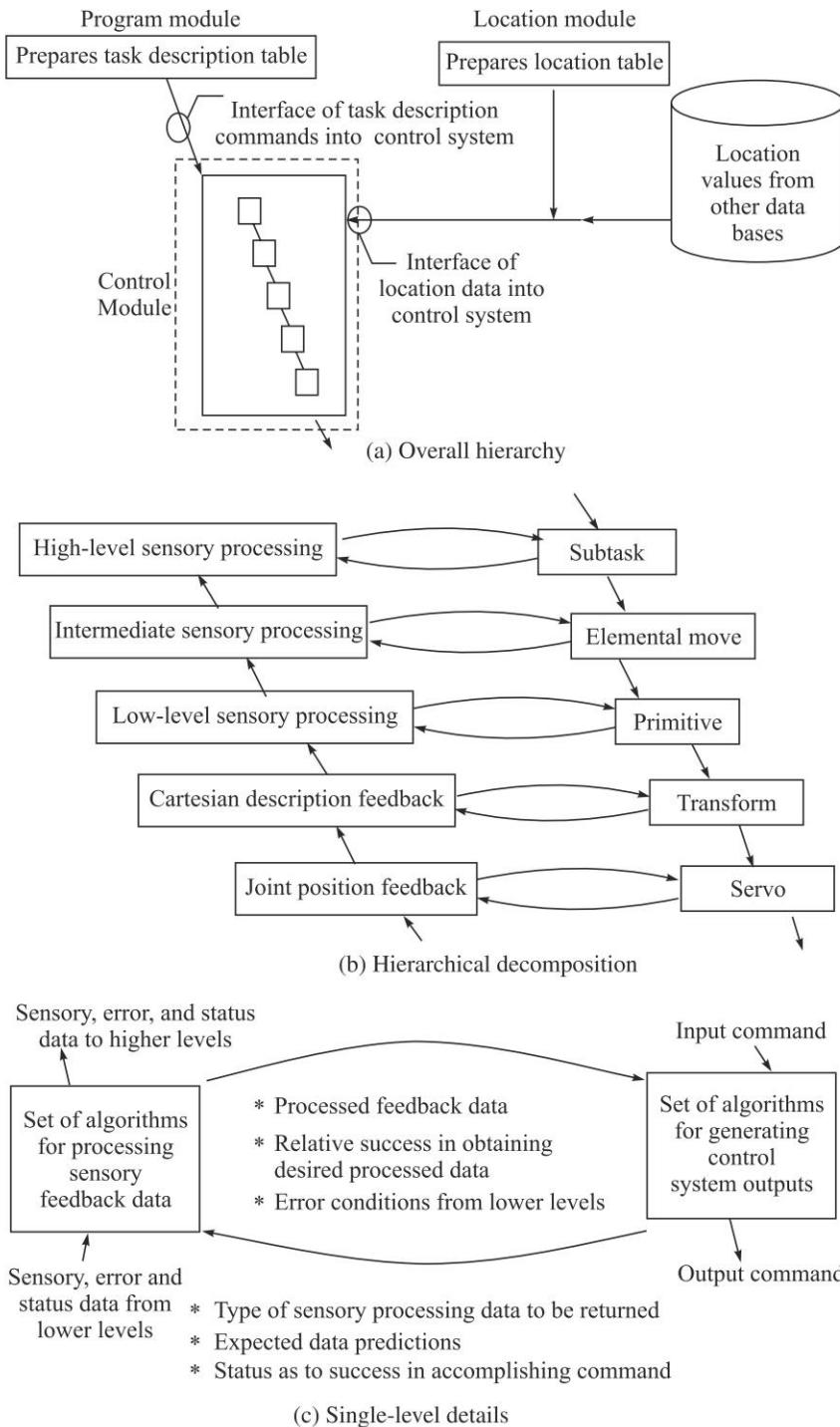
Note that an operator interface at each level of hierarchy is needed to allow an operator to perform supervisory and intervention functions of the robotic system. Although the basic philosophy of hierarchical control is acceptable to the researchers, there is a difference in the way they are implemented. For example, one could consider the control of a multivariate system in a rigorous mathematical form and will find the optimal or sub-optimal solution for the controlling of the entire system. If such techniques are properly structured, it can be implemented in a hierarchically structured array of processors.

13.1.2 Modules for Advanced Robots

An advanced robot system can be partitioned into three distinct modules, as shown in Fig. 13.1(a). These modules are as follows:

- The *program module* which accepts inputs from the user interactively or in the form of a robot program. The inputs are translated into a set of lower-level commands, perhaps including conditional branches, much as a compiler might convert from FORTRAN to machine language.
- The *location module* then serves to relate symbolic names of locations, for example, *clamp*, to Cartesian descriptions of points in the workspace.
- The *control system module* is finally responsible to control the motion of the robot so that it can achieve the tasks described by it at the locations described by the location module. This module has a real-time requirement which needs to be further decomposed. Figure 13.1(b) shows the decomposition of the real-time control functions. The exact number of levels in this decomposition, as well as the exact responsibilities of each level are those that are the concern of the system designer.

Note that at higher levels, the control functions are independent of the arm geometry. That is, the determination of the path is performed in Cartesian coordinates relative to constraints, for it may be necessary to specify locations in a moving frame such as on a conveyor (which means that the location module must be capable of real-time updates). Since the task description does not require the knowledge of the robot geometry at this stage, the same algorithms and software may be used for many different robots. At the lower levels of the control hierarchy, however, the details

**Fig 13.1** Advanced robot systems

of the algorithms become closely related to the arm geometry. Once the kinematics transformations have been performed, the control algorithms may vary from one robot to another. This is observed at the two lower levels of Fig. 13.1(b). The lowest level compares set points (provided by the next level) with joint positions (provided by sensors) and moves the joint to reduce the error. The previous level computes the kinematic transformations and provides those set points to the bottom level. One may observe from Fig. 13.1(b) that there are really two hierarchies involved. A top-down hierarchy of control, and a bottom-up hierarchy of status. Figure 13.1(c) shows this clearly by giving the details of a single level in the structure. Since each level in the hierarchy is well defined functionally, the robot control task can be divided into processing modules in which each module has well-defined inputs and outputs and, therefore, can be associated with an independent processing element.

13.2 HARDWARE ARCHITECTURE

Based on the control requirements of a robot, a model for *hardware architecture* of an industrial robot is shown in Fig. 13.2. It has the following components:

- 1. Bus** A bus connects several boards of the control system by allowing flow of communication data. The bus bandwidth shall be wide enough so as to satisfy the requirements by real-time constraints.

What is Hardware?

Hardware are the physical things like integrated circuits in the form of electronic chips, electric circuits on printed circuit boards, etc., which are felt hard once touched.

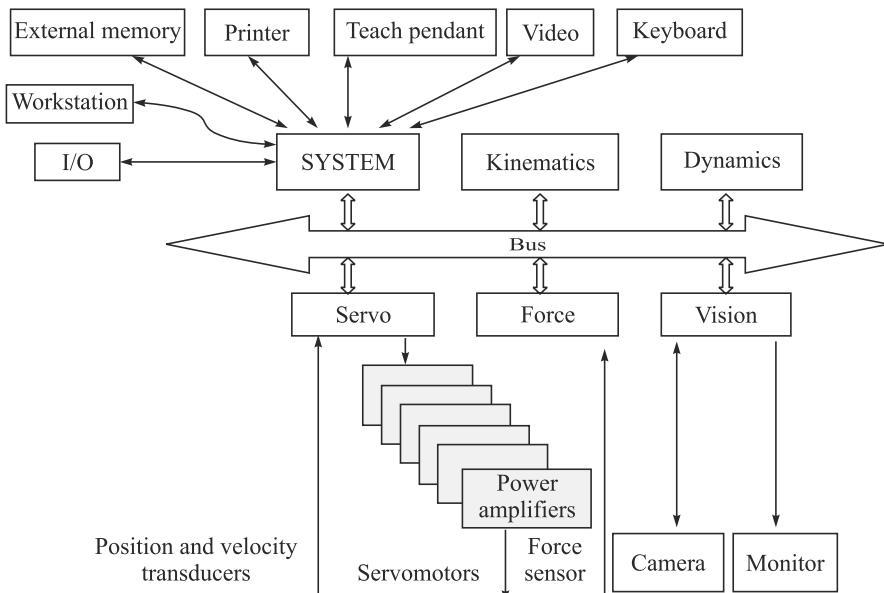


Fig. 13.2 A typical hardware architecture of an industrial robot

2. System Board It is typically a CPU consisting of

- a microprocessor with mathematical coprocessor,
- a bootstrap EPROM memory,
- a local RAM memory,
- a RAM memory shared with the other boards through the bus,
- a number of serial and parallel ports interfacing the bus and the external world,
- counters, registers and timers, and
- an interrupt system.

As indicated in Fig. 13.2, the system board is required to interact with the following modules:

- Operator interface through teach pendant, keyboard, video, and printer,
- Interface with an external memory (hard disk) used to store data and application programs,
- Interface with workstations and other control systems by means of local communication network, e.g., Ethernet,
- I/O interface with peripheral devices in the working area, e.g., feeders, conveyors, and ON/OFF sensors,
- System bootstrap,
- Programming language interpreter,
- Bus arbiter.

The other boards facing the bus may, besides the basic components of the system board, should have a supplementary or alternative processor (DSP, Transputer) for implementation of computationally demanding or dedicated functions. With reference to the hardware architecture of Fig. 13.2, these boards are explained below:

3. Kinematics Board This board must perform the

- computation of motion primitives,
- computation of direct kinematics, inverse kinematics, and Jacobian,
- test for trajectory feasibility, and
- handling of kinematic redundancy.

4. Dynamics Board The *dynamics* board is to compute inverse dynamics.

5. Servo Board This board has the functions of

- micro-interpolation of references,
- computation of control algorithm,
- digital-to-analog conversion and interface with power amplifiers,
- handling of position and velocity transducer data, and
- motion interruption in case of malfunction.

6. Force Board The *force* board performs the following operations:

- conditioning of data provided by the force sensor, and
- representation of forces in a given coordinate frame.

7. Vision Board This board is in charge of

- processing data provided by the camera,
- extracting geometric features of the scene, and
- localizing objects in given coordinate frames.

Note that the force and vision boards are responsible for the sensing capabilities of the robot. Hence, they require local processing capabilities to retrieve significant information from the given data. Additionally, the frequency at which data are exchanged between different modules through a common bus needs to be the same for each board. Those boards connected to the sensors indeed have the necessity to exchange data with robot at the highest possible frequency (from 100 to 1000 Hz) to ensure high dynamic performance to motion control as well as to reveal end-effector contact in a very short time. On the other hand, the kinematics and dynamics boards implement modeling functions and, as such, they do not require data update at a rate as high as required by the servo board. In fact, manipulator configuration does not appreciably vary in a very short time, at least with respect to typical operational velocities and/or accelerations of the industrial robots. Common sampling frequencies are in the range of 10–100 Hz. Also, the vision board does not require a high update rate, both because the scene is generally quasi-static, and because processing of interpretive functions are typically complex. Typical frequencies are in the range of 1–10 Hz. In sum, the board access to the communication bus of hardware control architecture may be performed according to a multi-rate logic which allows to solve bus data overflow problems.

13.3 HARDWARE FOR JOINT CONTROLLERS

In this section, several implementation strategies for joint controllers are explained. Figure 13.3 shows joint controllers for the PUMA robot, where joints are distributed. A separate processor is attached to the input and output transducers of each joint. This processor maintains current angular position and velocity of a joint and servos that joint towards a set point provided by the central processor. The central processor reads angular position from the joint processors, computes kinematic transformations, and provides set points to the joint processors.

By performing all the kinematics in the same processor, the architecture provides simplicity of design and programming, while offloading concurrent control of joints to parallel processors.

Figure 13.4 shows an alternative architecture with distribution of computations by function. Like the PUMA, this system associates one processor with each joint. However, the calculation of kinematics is also distributed. Each processor maintains the position and velocity of its joint, as well as the various trigonometric functions of those parameters. These quantities are stored in a multi-ported memory and, hence, are immediately available to the other processors. Calculation of the other kinematic terms is distributed uniformly among the processors. This scheme distributes the computational burden more uniformly, at the expense of a complex piece of hardware (the multi-ported memory 9511) and more complex software.

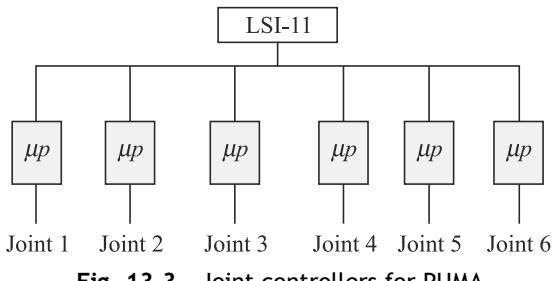


Fig. 13.3 Joint controllers for PUMA

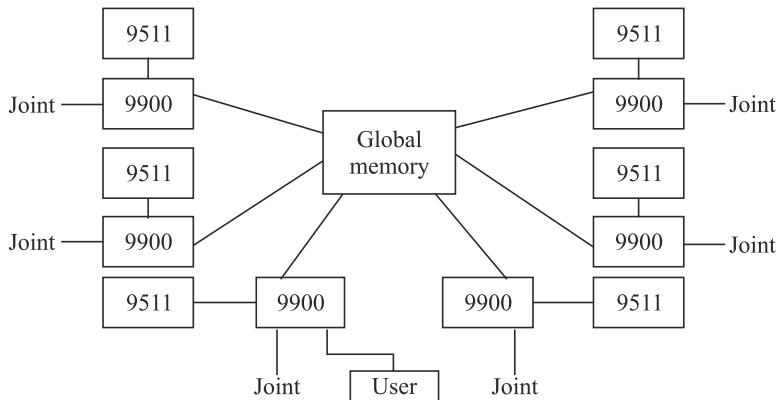


Fig. 13.4 Robot control distributed by function and by joint

An immediate extension of the above architecture for further improvement is shown in Fig. 13.5, where servo calculations are disassociated from the kinematics. Figure 13.5, in fact, is the PUMA architecture with the single central processor replaced by an array of processors and common memory. Rapid developments of devices with advanced technologies when suitably combined with general-purpose computing can provide tremendous increases in performance. The state of the art is changing so rapidly that coordinate transform chips, and a chip which computes the Jacobian and even the inverse Jacobian in real time is a real possibility. Such chips have the following characteristics:

- Fast, repetitive calculations (typically those which do not require conditional branches)
- Easy interface to a general-purpose processor's bus system
- Direct control by such a processor

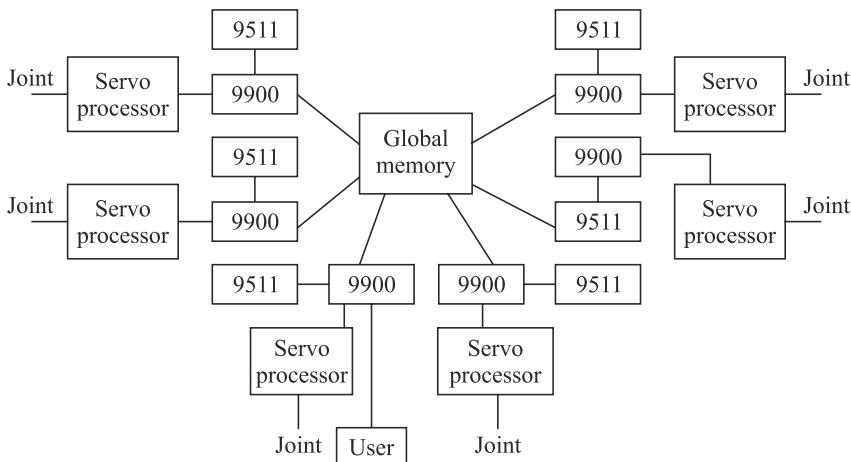


Fig. 13.5 A proposed architecture to decouple the servo calculations

13.4 COMPUTATIONAL SPEED

There exist a number of algorithms for controlling various aspects of robot motion for which it will be shown how complex calculations in these algorithms can be performed in a short time. Ways to organize the hardware and software to meet the objectives will also be presented. In this section, techniques for increasing the speed of computations in a conventional computer will be addressed. These techniques take advantage of the unique properties of many of the calculations that are required for robot control.

13.4.1 Using Integer Arithmetic

Here, the numbers will be represented as integers and avoid the use of floating-point (real) representations. This can be accomplished by a technique known as *scaling*, in which a mixed number, e.g., 22.75, is represented by an integer 2275 multiplied or scaled by a constant, 1/100. It is discovered that if the scaling is constant, considerable speed advantage can be gained over a floating point. Even with the advent of high-speed floating processors, integer arithmetic is typically 5 to 10 times as fast as floating-point one. Furthermore, while using purely integer arithmetic, fractional values can still be handled. Before using such techniques, one must make sure that the loss of accuracy inherent in the scaling will not seriously impact the performance of the robot.

Requirement of Computation Speed

Computation speed should be less than that of a robot's speed.

13.4.2 Computing Trigonometric Functions

In commercial computing systems, trigonometric functions are evaluated by several techniques including rational functions or Chebychev polynomials. The arithmetic involved is usually done in floating point, resulting in high accuracy and low speed. In robotic applications, however, the most appropriate mechanism for computing trigonometric functions is by table look-up. For example, the values of $\sin\theta$ are simply stored in a table for $\theta = 0 \dots 89$ degrees. This table of 90 words provides instantly the value of the sine of an angle to a 1° precision. In fact, the 90 words require less memory than that required by a program to compute the function explicitly. If higher accuracy than the nearest degree is required, two options exist. The first is linear interpolation between the table entries. Since sine and cosine functions are quite linear between 1° intervals, the interpolation gives excellent results. Another interpolation technique makes the use of trigonometric identity for the sine of two summed angles, i.e.,

$$\sin(\alpha + \beta) = \sin \alpha \cos \beta + \cos \alpha \sin \beta \quad (13.1)$$

The angle is segmented into two parts, an integer α , and a fraction β . Next, apply Eq. (13.1). The exact evaluation of Eq. (13.1) then requires four table look-ups for the values of $\sin \alpha$, $\cos \beta$, $\cos \alpha$, and $\sin \beta$, two multiplications and one addition. An approximate evaluation can be obtained even faster. For this, the angle is scaled so that the fractional part is very small. Let β be the fractional part. Then

$$\cos \beta \approx 1, \text{ and } \sin \beta \approx \beta \quad (13.2)$$

This is illustrated numerically in the following example.

Example 13.1 Value of sin (0.8034)

Assume that the values of the sine function are tabulated in 0.01 rad increments, i.e., the values of $\sin(0.80)$ and $\sin(0.81)$, etc., are known. Moreover, one can write

$$0.8034 \text{ rad} = 0.8 \text{ rad} + 0.0034 \text{ rad} \quad (13.3a)$$

Then, using Eq. (13.3a), Eq. (13.1) can be written as

$$\sin(0.8034) = \sin(0.80) \cos(0.0034) + \cos(0.80) \sin(0.0034) \quad (13.3b)$$

For the very small angle of 0.0034 rad, $\cos(0.0034) \approx 1$ and $\sin(0.0034) \approx 0.0034$. Hence,

$$\begin{aligned} \sin(0.8034) &= \sin(0.80) + 0.0034 \cos(0.80) \\ &= 0.7174 + 0.0034 \times 0.6967 = 0.7198 \end{aligned} \quad (13.3c)$$

in which $\sin(0.80)$ and $\cos(0.80)$ are available from the tabulated results given in Table 13.1.

Note that direct evaluation of $\sin(0.8034)$ using a MATLAB function gives the value as 0.7197, whereas the above calculation evaluated as 0.7198. The difference is only 0.0001. This interpolation technique requires two table look-ups (one for *sine* function and another for *cosine*), one multiplication, and one addition only. Note that in actual implementation for a robot system, the calculation of trigonometric functions is one of the fastest.

Table 13.1 Tabular values of sine and cosine functions

Sine function	Value	Cosine function	Value
$\sin(0.7)$	0.6442	$\cos(0.7)$	0.7648
$\sin(0.8)$	0.7174	$\cos(0.8)$	0.6967
$\sin(0.9)$	0.7833	$\cos(0.9)$	0.6216

13.4.3 Matrix Operations

In robot control, explicit matrix inversion using the definitions of adjoint and determinant is generally not required, rather a set of linear algebraic equations are solved which is done numerically. In case an explicit inversion must be computed, it is done numerically by solving a series of linear algebraic equations whose right-hand sides are the columns of an identity matrix, whereas the corresponding unknowns are the columns of the desired inverse. For example, if an $n \times n$ matrix \mathbf{X} is the inverse of the $n \times n$ matrix \mathbf{J} , i.e., $\mathbf{X} = \mathbf{J}^{-1}$ then one needs to solve only n linear algebraic equations n times based on the fact that $\mathbf{J}\mathbf{X} = \mathbf{1}$, where $\mathbf{1}$ is the $n \times n$ identity matrix. The algebraic equations for $j = 1, \dots, n$ are

$$\mathbf{J}\mathbf{x}_j = \mathbf{e}_j, \text{ where } \mathbf{X} \equiv [\mathbf{x}_1, \dots, \mathbf{x}_n] \text{ and } \mathbf{1} \equiv [\mathbf{e}_1, \dots, \mathbf{e}_n] \quad (13.4)$$

The vectors, $\mathbf{x}_j \equiv [x_{1j}, \dots, x_{nj}]^T$, and $\mathbf{e}_i \equiv [e_{ij}]^T$, for $i = 1, \dots, n$, contain n scalar elements. Furthermore, $e_{ij} = \delta_{ij}$, where δ_{ij} is the *Kronecker delta* which takes the value of one when $i = j$; otherwise zero if $i \neq j$. Note that using the approach shown in Eq. (13.4), one would require only order n^3 computations whereas using the definitions of adjoint and determinant, the latter itself requires order n^3 computations plus more calculations for the adjoints. Hence, the explicit computation to find inverse of a matrix using adjoint and determinant is always avoided. Moreover, in numerical-analysis literature, high accuracy and stability or robustness are important, whereas in robotics, speed is of the essence, whereas accuracy, although important, is not as

difficult to maintain, as the control law can take care of such errors by correcting them through control gains. Thus, in many instances, simple and fast methods are substituted over complexity and high precision.

Example 13.2 Inversion of a 3×3 matrix

Inverse of a 3×3 matrix, \mathbf{J} , as given below, is obtained using the methodology explained for Eq. (13.4). Matrix \mathbf{J} is as follows:

$$\mathbf{J} \equiv \begin{bmatrix} 2 & 5 & 5 \\ 1 & 3 & 4 \\ 5 & 6 & 8 \end{bmatrix} \quad (13.5)$$

Considering the vectors $\mathbf{e}_1 \equiv [1 \ 0 \ 0]^T$, $\mathbf{e}_2 \equiv [0 \ 1 \ 0]^T$, and $\mathbf{e}_3 \equiv [0 \ 0 \ 1]^T$, the solutions \mathbf{x}_j , for $j = 1, 2, 3$, can be easily obtained using, say, MATLAB command of “ $\mathbf{x}_1 = \mathbf{J} \backslash \mathbf{e}_1$ ”, as

$$\mathbf{x}_1 = \frac{1}{5} \begin{bmatrix} 0 \\ 4 \\ -3 \end{bmatrix}, \mathbf{x}_2 = \frac{1}{15} \begin{bmatrix} -10 \\ -9 \\ 13 \end{bmatrix}, \text{ and } \mathbf{x}_3 = \frac{1}{15} \begin{bmatrix} 5 \\ -3 \\ 1 \end{bmatrix} \quad (13.6a)$$

Based on Eq. (13.6a), the inverse of the matrix \mathbf{J} , i.e., \mathbf{J}^{-1} , is given by

$$\mathbf{J}^{-1} \equiv \frac{1}{15} \begin{bmatrix} 0 & -10 & 5 \\ 12 & -9 & -13 \\ -9 & 13 & 1 \end{bmatrix} \quad (13.6b)$$

It will now be a simple matter to verify that $\mathbf{J}\mathbf{J}^{-1}$ is an identity matrix of size 3×3 .

13.4.4 Hardware Considerations

Distributed computing, i.e., the use of several computers interacting in real time to provide the required computational speed and power, finds an immediate application in robotics. So far it is assumed that any computation is performed by a single general-purpose computer. Here, for two simple functions, namely, reading encoders for position information, and computing joints servo error signals are explained which can be performed using distributed computing.

1. Reading Encoders The rate at which the data comes from an encoder depends upon the rotational speed of the joint, the resolution of the encoder, and the algorithm used for analyzing the signals. If quadrature decoding (a new count for each of the four states of an incremental encoder) with 2500 counts/rev encoder, and maximum joint speed of one rev/s are assumed, a new piece of data is found in every 100 μs . Using interrupts, the absolute minimum set of instructions in any software associated with the computer interface to the encoders would be

```

REGISTER SAVE
BIT TEST
JUMP
INCREMENT (OR DECREMENT)
REGISTER RESTORE
RETURN

```

This process would typically require about 5 to 10 μs on any computer. If the processor speed is taken as 5 μs and six joints to be controlled at the maximum speed then just to keep track of the positions, it would require $5 \times 6 = 30 \mu\text{s}$, which is 30% of the computer resource (i.e., loops).

2. Computation of Servo Error Signals With a PD (Proportional Derivative) controller, as given in Chapter 10, one must compute

$$\tau = \mathbf{K}_p(\theta_d - \theta) - \mathbf{K}_v \dot{\theta} \quad (13.7)$$

where τ is the vector of desired torques, and θ_d and θ are the desired and actual joint angles, respectively, whereas $\dot{\theta}$ is the actual joint rate. Moreover, the terms, \mathbf{K}_p and \mathbf{K}_v are the matrices of proportional and derivative gains, respectively. The program instructions to implement this type of calculation is

```

LOAD θd
LOAD θ
SUBTRACT
MULTIPLY (θd - θ) BY Kp
LOAD ̇θ
MULTIPLY ̇θ BY Kv
SUBTRACT
OUTPUT
RETURN

```

Optimistically, this program would take 15 μs . For stability, the joint servo needs to be calculated roughly every 2 ms. This number varies significantly with the structural resonances of the manipulator. Thus, the joint servo calculation for the six joints takes about

$$\frac{6 \times 15 \times 10^{-6}}{2 \times 10^{-3}} \times 100 = 4.5\% \quad (13.8)$$

of the computer resources. These estimates vary tremendously with implementation details. The point, however, is that a significant amount of the computing involved in robotics is localized and easily distributable among processing elements.

13.5 ROBOT LANGUAGES

A robot language is required for programming purpose. A robot program consists of commands that tell about the robot's state or the function it will perform. It also provides data about the speed of the robot, its dwell or delay times of the robots, along with the status of input or output device, its execution, etc. In order to control the robot, the program requires establishing relationships between the joint motions and end-effector's configurations, and the devices it has to interact with. To make sure the robot functions within its workspace, the above configurations must lie within the robot's workspace. Hence, one needs to teach those configurations to the robot manually and physically using teach pendant through what is known as online programming, whereas offline programming is written in textual languages like AL or VAL. These languages allow one to write programs to define the logic and sequence for more complex actions using say sensors' inputs, etc. Most of the robots' languages actually use a combination of teach pendant programming and textual

programming. These programmes are written using languages which are proprietary to the robot manufacturers. Hence, they are known as Domain-Specific Languages (DSL).

Note that knowing all programming syntaxes and commands of one particular language are not very useful in a book like this, as the reader may not have the access to that particular robot in his or her laboratory/organization for which the language is learnt. As a result, the features of different robot languages are summarized here with the demonstration of few commands and programs using KUKA Robot Language (KRL) which were used to run the KUKA robots available in the PAR Laboratory of IIT Delhi, as shown in Fig. 13.6.

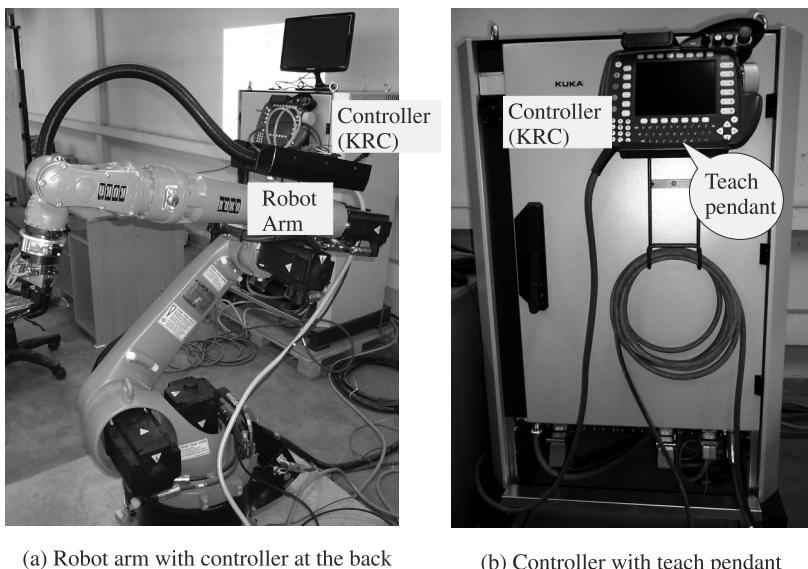


Fig. 13.6 KUKA KR-5 Arc Robot in the PAR laboratory of IIT Delhi

13.5.1 Different Robot Languages

Some of the robot languages which have been used with different robotic systems are presented next. A good comparison of several robot programming languages appeared in Gruver et al. (1984).

1. WAVE It was developed in 1973 at the Stanford Artificial Intelligence Laboratory. The wave system was interfaced with a vision system to successfully demonstrate the hand-eye coordination of a robot.

2. AL In 1974, a second robot language was developed at Stanford which was based on Algol computer programming language to control multiple robot arms in parallel and cooperative tasks. It was designed to facilitate operations for assembly using force sensors.

3. AML A Manufacturing Language (AML) is a robot programming language developed in 1982 by IBM Thomas J. Watson Research Center at Yorktown Heights to be used in IBM's robotic products.

It is a high-level language based on subroutines that can manage assembly robots, end-effectors, active force feedback, and Cartesian arm. Before AML, IBM had, however, developed several other robot languages, namely, Emily, ML, and Autopass, since 1976.

4. RPL RPL or Robot Programming Language, was developed by SRI International which was based on computer-programming languages. It was designed to control various machines of a work cell, i.e., robots, machine tools, and vision systems. It was designed to be used by those who are not skilled programmers such as factory production engineers or line foremen.

5. Help It is a commercial robot-programming language from General Electric Company for the control of its multiple Cartesian arms in assembly tasks. *Help* is an interpreter based on the computer-programming language *Pascal*.

6. Jars It is a language from NASA's Jet Propulsion Laboratory and was based on the *Pascal* computer-programming language for the control of robots used in assembly of solar-cell arrays.

7. MCL MCL or Manufacturing Control Language, was developed by McDonnell-Douglas under the sponsorship of US Air Force. It is an extension of the Automatically Programmed Tooling (APT) part programming language used for numerically controlled machine tools. MCL was used for robot's motion specification, vision-system operation, image modeling, and real-time conditional logic.

8. Rail Rail, designed by Automatix, Inc., was introduced in 1981. It was a language for control of both vision and manipulation. It is basically an interpreter based on *Pascal*. Rail has incorporated many constructs to support inspection and arc-welding systems of Automatix's products.

9. VAL Victor's Assembly Language or VAL, was introduced in 1979 by Unimation Inc. for their PUMA robot series. Its upgraded version was released in 1984. The language was meant to define robot tasks easily. VAL follows the structure of *Basic* computer-programming language with many new commands. VAL has its own operating system, namely, the VAL monitor, which comprises a user interface, editor, and file manager.

10. KRL KUKA Robot Language or KRL, is a robot-programming language similar to *Pascal*, and used for all KUKA industrial robots (Fig. 13.6). It allows typical programming statements such as variable assignments, conditionals, and loops. KRL also provides robot-specific statements, e.g., to specify motions and interact with tools.

13.5.2 Generations of Robot Languages

Robot-programming languages can be classified based on their structures and capabilities, namely, first-generation language, second-generation language, and future generations.

1. First-Generation Languages First-generation languages combine teach pendant procedures with the command statements. VAL is an example of a first-generation language. These languages are used mainly to control motion, and hence are sometimes referred as *motion-level* languages (Groover et al., 2012). Typical features of these languages are as follows:

- Ability to define robot motions, i.e., teach pendant to define locations and statements to define motion sequences
- Define line, interpolation, branching (i.e., subroutines), and elementary sensor command, e.g., on and off signals
- Inability to use complex sensor data, and limited communication ability with other computers

Note that to write programs with low and medium complexity, a person with computer-programming experience finds the first-generation languages easy to use compared to a shop person who finds the teach pendant method more convenient.

2. Second-Generation Languages Second-generation languages overcome the limitations of first-generation languages, which have added capabilities so that robots appear more intelligent. These languages are also referred as *structured programming languages* because of their structured control sequences, similar to computer languages. Hence, a computer programmer's skill is required, which might be disadvantageous in certain cases. Commercially available second generation languages are AML, RAIL, MCL, VAL II, and others. Typical features of such languages are

- Motion control abilities, which are same as first generation languages
- Capability of integrating advanced sensors' data other than simply on and off signals of a simple binary device
- Ability to process information received about environment and modify system behavior
- Ability to interact with computers to keep record of the data, generate reports, and control activities in the workcell
- Extensibility of the language, i.e., to handle the future requirements of sensing and other aspects of future robots by developing commands, subroutines, and macro statements

Using a second-generation language, a robot is capable of controlling the gripper forces on an object based on its analog sensor's data, i.e., a force sensor. This was not possible with the first-generation language where only on and off commands could be given to a gripper. Also, in case of faults or errors in the robot, the first-generation robot will probably stop, whereas a robot with second-generation language programming can recover using an intelligent algorithm programmed in it.

3. Future-Generation Languages Future-generation robot languages allow world modeling, i.e., they should be model-based or task-based languages, rather than specifying the details of every action the robot has to perform. Here, the programmer will be able to include instructions in the application program at a higher level than in an explicit robot-program language. The robot should have the knowledge of the environment and should be able to develop a step-by-step procedure to perform a task based on a stated objective of what needs to be achieved by a robot. For example, the robot must understand a command like *Tighten a nut*. For this command, the robot must find the nut and the spanner, pick them up, and place them in an orderly fashion. Finally, it should tighten the nut with the spanner. These languages should allow complete offline programming, as explained in Section 13.6.2.

13.5.3 Structure of a Robot Language

A robot language must be designed to operate with a robot. As illustrated in Fig. 13.7, a robot-programming language must be able to not only program a robot but also to control it, interface with other peripherals in the workcell, e.g., sensors and other equipment, and communicate with other computers in the factory.

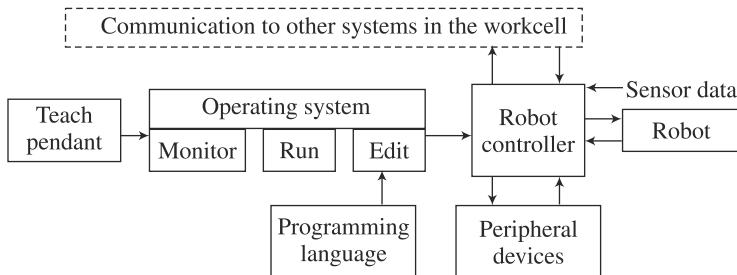


Fig. 13.7 Functions of programming

A key component in a robot language is an operating system. It is basically a software, similar to a computing system in which the operating system supports its internal operations, has the three modes of operations, namely, monitor, run, and edit. In the *monitor mode*, supervisory control of a robot is accomplished, i.e., using a teach pendant one can define robot configuration, its speed, store programs, etc. The *run mode* is used to execute a robot program and make the robot move as per program instructions. The *edit mode* allows a user to change the program by editing it or writing a new one. For example, if a point has to be added in an already defined set of points, one can add in edit mode. For the programming of a robot, however, one needs to create several objects, which are explained below:

1. Program Each program module has its specific purpose. Hence, it contains corresponding data and routines. The program is divided in such a manner that it facilitates the program handling. Each subdivision represents a robot action. It is generally written by a user.

2. Data These are the values and definitions set in the program. Depending on their types, they can be refined during the execution of the program.

3. Entry Routine It is the starting of a program. Generally, it is referred to as *main* without which the program will not execute.

4. Subroutine In a robot program, it is often necessary to branch off to execute a set of instructions which may be common to several parts of the main program. For example, to report where the robot's configuration is, it may require some data to execute the instructions.

5. Instruction An instruction is a request for an event to take place, i.e., “Move a robot to a certain configuration” or “Select a digital output”, etc. The syntax and function of the instructions are generally listed in the technical manual of a robot language.

13.5.4 Requirements of a Robot Language

Each robot is programmed using its domain-specific language. However, it should be capable of defining action of the robot in the three-dimensional Cartesian space. Hence, there is a need to describe actions using what is called *types*. For example, a geometric type, namely, *frame*, can be defined to describe the configuration of a workpiece with respect to another *frame* describing a table on which the workpiece is placed, etc. One can build a world model using the type of frames.

Note that there are other requirements for a robot language. These are to provide motions to a robot from one goal to another, reading and interpreting a sensor data, etc. Different programming languages of industrial robots have relatively similar structure and similar set of commands even though nomenclature may be different. Because of these characteristics, the acquired programming skills can be used for programming in other languages as well. In general, the programming instructions for a robot can be divided into the following categories:

1. Instructions for Motion Control and Position This contains information about locations, coordinates, trajectory interpolations, tool motion speed and time of intervals, etc.

2. Instructions for Gripper or Tool Control It contains information about starting and stopping a program, branching, repeating of cycles, program interruptions, etc.

3. Instruction for Gripper Control These instructions are about gripper closing and opening conditions. Sometimes these may be used to control tool or gripper fingers.

4. Input and Output Instructions These contain information signal for input and output, and to cooperate with other machines and devices of the robotic system at hand.

5. Communication Instructions These are for data communication and transfer of robot's inner information to the computer.

6. Other Instructions These instructions are used for parameter settings, program selection, fault diagnosis, etc.

13.5.5 Problems of Robot Languages

Robot-programming languages face additional difficulties other than those which are common with any computer-programming language. This is due to the fact that, in contrast to a computer, a robot has to interact with the physical world. Some of the aspects due to which problems may occur are highlighted as follows:

1. Internal vs. External Models of a Robot There is a transformation involved between the robot's understanding of the world (joint space) and the user's way of describing it (Cartesian space). Any small discrepancies between these two internal and external models due to say, inertia of the robot while running at higher speeds may result in poor grasping of an object or collision. There are also other reasons for these variations. For example, long usage of a robot, may wear out its components, thereby, causing a change in its link dimensions and their relative

orientation specified by a set of constant Denavit and Hartenberg (DH) parameters introduced in Chapter 5. One needs to perform kinematic identifications to extract these changes in order to correct the relationship between the internal and external world descriptions of the robot.

In dealing with the problems caused by a robot not able to reach a configuration specified in the program of the robot, one may take the help of external sensors, say, a vision system, to find out if the object is truly placed in configuration for which the robot was programmed. Sometimes, assembly operations require greater precision than the robot's accuracy. A typical example of such application is the classical peg-in-hole operation. Moreover, the robot's accuracy varies over its workspace. Hence, after the program is written, the robot should be moved very slowly to see whether the intended functions are performed correctly or not. Then, the speed should be increased gradually for testing because the higher speed may not only cause higher inertia forces but also higher servo errors. Both may lead to missing a location or failing to hold an object by the gripper of the robot.

2. Program Hierarchy While writing a large computer program, a standard approach is to develop it in smaller pieces and test them separately before putting them together. In general, such an approach functions satisfactorily for computer programs. However, in robot programs, it may fail as a robot program is sensitive to the robot's initial position and other factors. These problems generally arise mainly due to the robot's dependencies on its configuration and speed of motion. For example, the accuracy of a robot changes from one configuration of the workspace to the other. Hence, a program written for a certain accuracy will not be valid for the same task performed at different configurations of the robot's workspace. Additional care must be taken to correct the situation.

3. Error Detection and Recovery One of the important aspects in robot programming is that it must detect if there is an error in the program or the motion is as per the instructions. Otherwise, there may be damage to the robot or its environment with which it is interacting. In order to detect, a robot must perform some explicit tests, which of course will complicate the robot programming and may slow down it a bit but will keep the robot safe during operation. If an error is detected the robot should be able to recover automatically or manually. For this, the program developer should be trying to predict which section of the program may fail, which is extremely difficult though, and trying to take precautions by testing the particular section again and again under various operating conditions, and adding some additional checks to recover or to stop with an error message.

13.6 ROBOT PROGRAMMING

The greatest advantage in industrial robot-based automations is their flexibility, ability to rearrange against new production and their large movement range. However, a major obstacle in using these robots as general-purpose assembly machines is the lack of suitable and efficient communication between the user and the robotic system so that the user can direct the manipulator to accomplish a given task. Programming of

Software

Software is a set of programs that makes the hardware function properly.

industrial robots for a specific application is still very difficult, time-consuming, and expensive. For example, manually programming an arc-welding robot for the manufacture of a large vehicle hull takes more than 8 months, while the cycle time of the welding process itself is only 16 hours (Pan et al., 2012). In this case, programming time is exorbitantly more (almost 360 times) than the execution time. Hence, smaller companies are not able to benefit from the application of robots in their factories. For practical uses of robots, there are two basic modes of programming to communicate with them. They are (1) online programming (including walk-through and lead-through), and (2) offline programming. They are explained in the following sections.

13.6.1 Online Programming

Online programming takes place at the site of production itself by skilled robot operators and involves the workcell. It has the following advantages and disadvantages compared to offline programming:

Advantages

- Simplicity in programming so that even an operator with virtually no qualification can do it
- Easily accessible
- In concordance with the actual position of equipment and pieces

Disadvantages

- Robot cannot be used in production while programming is carried out
- Slow movement of the robot while programming
- Jogging a robot using teach pendant is not intuitive as many coordinate systems are usually defined in a robotic system
- The operator must always track which coordinate frame the robot is set while jogging
- Program logic and calculations are hard to program
- Difficult to incorporate sensor data
- Cost equivalent to production value
- Online programming is not yet readily compatible with CAD/CAM technologies, data communications, networking, and other computer-based technologies.

Online programming, which is also known in the literature as *teaching-by-showing* (Koren, 1987), is by far more commonly used in practice. In online programming, an actual part is placed in the workspace exactly as it would be during production, and programming is done based on teaching a robot the points or trajectories, e.g., a circle, it has to follow. Teaching can be carried out using a teach pendant or propelling the robot manually by the operator who uses his/her muscles to shift the arm from one position to another. Other devices, e.g., joysticks or a scaled-down replica of the manipulator, can also be used to move the robot. A typical teach pendant for KR C2 controller of KUKA robots is shown in Fig. 13.8. Layout, weight, buttons, and command sequences of a teach pendant may vary from manufacturer to manufacturer. Once teaching is done, it can be replayed

Online

Online means the robots are in line with other machines when the programs are written on their controllers.

automatically. Regardless of the way of propelling the arm during teaching, there are two ways of recording the trajectory of the arm motion, namely, point to point, or PTP, and continuous path, or CP.



Fig. 13.8 Teach pendant for KUKA KR C2 controller
(Courtesy: PAR Lab., IIT Delhi)

In the PTP method, the arm is transferred to each characteristic point of the trajectory, stopped there, and by pressing a special button on the control panel, this position is memorized by the control system. During playback, the robot arm goes through these points using some form of interpolation between them. In the CP (Continuous Path) method, on the other hand, the positions are being recorded automatically at constant intervals of time as the arm is transferred. As the points are very near to each other, no special interpolation routines are necessary.

Teach Pendant

It is a handheld robot control terminal that provides a convenient means to move the robot, teach locations, and run robot programs.

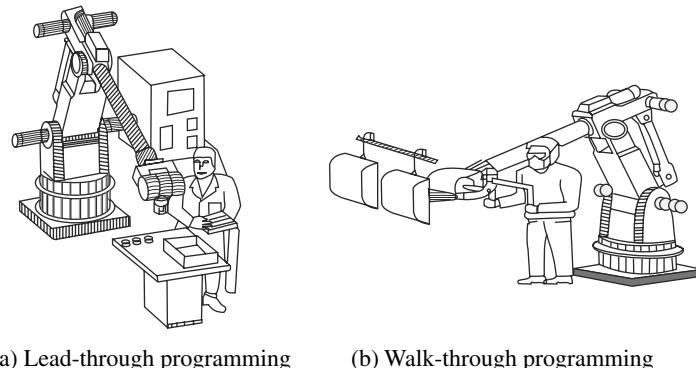
In either type of motion, i.e., PTP or CP, the motions can be played back at different speeds, e.g., 20%, 50% or 80% of the maximum achievable speed allowed by the motors. Even though online systems generate procedural programming code, users can create waypoints without editing this code and, as a result, are typically viewed as less intimidating and more intuitive than offline systems. Although users generally view online programming as less intimidating, they still spend an inordinate amount of time simply moving the robot between waypoints instead of transferring any real knowledge. Moreover, due to the difficult robot-positioning process, users tend to discard their previous work and create waypoints from scratch every time. This is very wasteful since robot programs tend to contain repeated subtasks, and product designs contain many similarities to previous designs.

1. Lead-through Programming One of the online programming methods is *lead-through* programming, which is also referred in the literature, e.g., in Groover et al. (2012), as *powered lead-through*. This method, as illustrated in Fig. 13.9(a), involves teaching the robot by leading it through the motions the user wishes the robot to perform, which is replayed during the actual operation of the robot. Hence,

it is also called *teach and playback* or *guiding* method, which can be accomplished by the following steps:

- Lead the robot in slow motion using manual control through the entire assembly task and recording the joint angles of the robot at appropriate locations in order to replay the motion. For KUKA robots, one can use the jog keys or 6D mouse of the teach pendant shown in Fig. 13.8 to move the robot's end-effector using either *World*, *Tool*, *Base* (defined in the environment where the workpiece is placed) coordinate system or axis-specific jogging option.
- Record the configuration, i.e., position and orientation, reached in the previous step using *Touch Up* softkey of the teach pendant.
- A robot program is then written or automatically generated by the controller, i.e., KR C2 for KUKA robots.
- Editing the above program to make sure that the robot will not collide with obstacles while completing the task.
- Play back the taught configurations in slow motion.
- If the taught configurations are correct then the robot is run at an appropriate speed in repetitive mode.

If the task is changed then the above steps need to be repeated. The main advantage of lead-through programming is that it requires only a relatively small memory space to record angular positions of the joints and it is simple to learn.



[Courtesy: https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html]

Fig. 13.9 Online programming

- 2. Walk-through Programming** *Walk-through* programming is basically a manual lead-through programming where an experienced operator physically grasps the robot arm and manually moves it through a set of desired configuration, as required in, say, arc welding, spray painting, etc. This is illustrated in Fig. 13.9(b). If the robot is large and awkward to physically move, assistive devices, as explained in Item 3 of this section, or a scaled-down replica are used. The teach button near the wrist of the robot or the assistive devices has to be kept pressed during the movements that will be part of the programmed cycles. The path cycle is divided into many closely spaced points (100's or 1000's based on the quality of the motion demanded).

Note that the interaction with the teach pendant or whatever device is used to program a robot during lead-through or walk-through programming is generally difficult. A software professional may be good in programming but he or she has limited knowledge or skill in welding, painting, etc., whereas a welder has the skill of good welding but little knowledge about robot programming unless some training has been imparted on that person. As a result, both groups of people should work in tandem for effective development of the robot programs to automate the process of, say, welding, and bring real benefits, i.e., product quality, equipment durability, reduced manufacturing time, and enhanced profit.

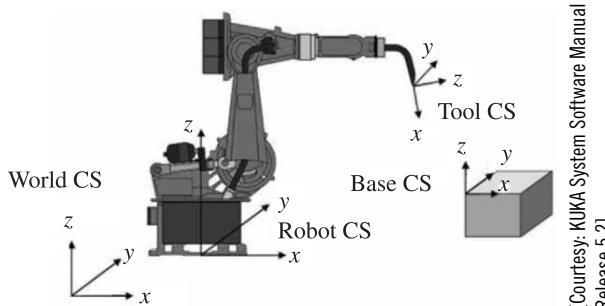
3. Assistive Devices for Walk-through Programming In order to overcome the difficulties mentioned above to perform walk-through programming by an operator, several assistive devices have been proposed. Some of these are summarized in Pan et al. (2012). They are described here.

- Support devices to measure the position and direction vector of the dummy tool on the tip of the posture-measuring unit, which can be used to generate robot programs. Such a device was developed for a deburring and finishing robot.
- Force and moment sensor that can be placed on the robot's wrist which can be pushed, pulled, or twisted by the operator to move the robot in the direction he or she desires to follow a desired trajectory.
- Vision-based programming where the path is captured by camera(s) to move the robot.

Note that most of the assistive devices for online programming are used in research laboratories. Such devices are not commercially available as they were applied in specific set-ups and yet to apply to general applications.

Example 13.3 KUKA Coordinate Systems

Every robotic system has several coordinate systems to conveniently allow a user to move the robot. As an example, KUKA robots use the following coordinate systems: Referring to Fig. 13.10.



[Courtesy: KUKA System Software Manual
Release 5.2]

Fig. 13.10 Different coordinate systems of KUKA KR-5

1. Axis-specific Motion In this system, each of the six axes is rotated along positive or negative direction using the jog keys of Fig. 13.8. After selecting the axis-specific coordinate system by toggling one of the jog keys of the teach pendant, one can rotate a joint by pressing the corresponding jog key which is active after the driver-switch located at the back of the teach pendant is on by pressing it.

2. World Coordinate System This is a fixed coordinate system whose origin is located at the base of the robot. This is a rectangular Cartesian system.

3. Tool Coordinate System It is a rectangular Cartesian coordinate system whose origin is located at the tool. Depending on the type of tool used, one can define its origin and define its axes, as explained in Example 13.4.

4. Base Coordinate System It is also a rectangular Cartesian coordinate system whose origin is defined on the workpiece that has to be processed. This can be referred to as workpiece coordinate system as well.

Example 13.4 Tool Calibration

The process to teach the robot its location of the *Tool Center Point* (TCP) with respect to the world coordinate system and assigning a coordinate system at TCP is defined as *tool calibration*. This is helpful in the controlled movement of the robot with respect to the tool frame. The Homogenous Transformation Matrix (HTM) which consists of position and the orientation of the tool, as explained in Chapter 5, will define the tool calibration. For this, two steps are required:

1. Calculation of TCP, i.e., Position of Tool Center Relative to the Robot Coordinate System: One of the methods to define the position of the tool with respect to flange is *X-Y-Z-4-Point method*. In this method, the TCP of the tool is moved to a reference point from four different directions as shown in Fig. 13.11. The TCP of the tool is then calculated from a different flange position and orientation.

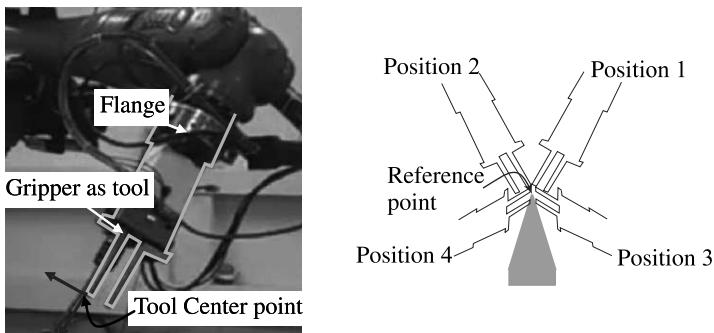


Fig. 13.11 KUKA robot and X-Y-Z-4 point method for TCP calibration

2. Definition of Rotation of the Tool Coordinate System: The definition of the axis of the tool frame can be done using the *ABC-2 point method*. In this method, the TCP is moved along the *X-axis* of the tool as shown in Fig. 13.12(a). The tool is now moved so that the reference point is located with a positive *Y* value, as shown in Fig. 13.12(b).

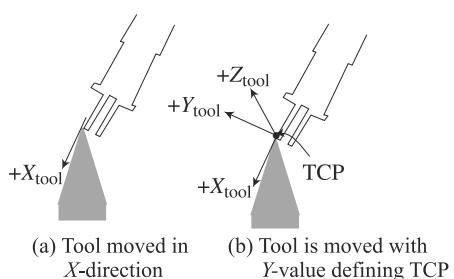


Fig. 13.12 ABC-2 point method

Example 13.5 Base Calibration

During base calibration, the user assigns a Cartesian coordinate system (i.e., BASE coordinate system) to a work surface or the workpiece. The BASE coordinate system has its origin at a user-defined point. Advantages of base calibration are as follows:

- The TCP can be jogged along the edges of the work surface or workpiece.
- Points can be taught relative to the base. If it is necessary to offset the base, the points moved with it do not need to be re-taught.

The base is defined by a *3-point method* where the calibrated tool is placed at a point to mark the origin of the base frame. Then the point on X -axis is given to define X_{base} , and similarly the orthogonal direction, i.e., Y_{base} is defined. The Z_{base} is obtained through the right hand rule cross product as shown in Fig. 13.13.

Cardioid

A cardioid (from a Greek word meaning *heart*) is a curve traced by a point on the perimeter of a circle rolling on a fixed circle of the same radius.

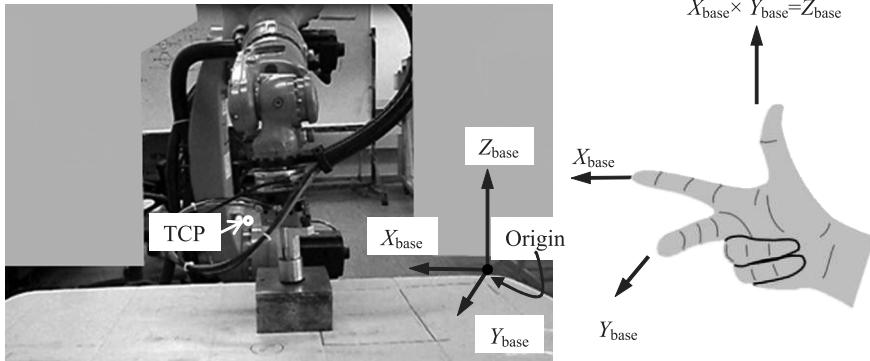


Fig. 13.13 Base calibration of a robot

Example 13.6 Programming a Task

In this example, a robot programming using KRL (Kuka Robot Language) is illustrated. Here, a geometrical curve known as *cardioid* is drawn by KUKA KR-5 robot available in the PAR Lab. of IIT Delhi. The parametric equations for the cardioid are given by

$$x = 16 \sin^3 \theta \quad (13.9)$$

$$y = 13 \cos \theta - 5 \cos (2\theta) - 2 \cos (3\theta) - \cos (4\theta) \quad (13.10)$$

The parametric equations given by Eqs. (13.9–13.10) are scaled 10 times to draw on a larger piece of paper as shown in Fig. 13.14(a). Figure 13.14(b) shows the KRL code with the commented lines starting with the symbol of a semicolon.

Figure 13.14(a) shows the KUKA robot drawing the curve using a spring-loaded gripper to hold the marker at the end-effector.



(a) Carioid drawn by KUKA KR-5 robot (b) KRL program to draw a cardioid curve

```

DEF cardoid( )
DECL POS SETPOS
DECL POS PREPOS
;SETPOS and PREPOS are the predefined
structure in KRL
;DECL POS CIRC
DECL INT THET
INI
;Make a Cardaiod with the equation
;Declare a home poisiton
PTP P5 Vel=100 % PDAT6 Tool[8]:saha
Base[8]:saha-base
;Declare a starting position for curve
where the marker
;touches the Base Plane
PTP P3 Vel=50 % PDAT3 Tool[8]:saha
Base[8]:saha-base

FOR THET = 0 TO 360
SETPOS.X = 160*
SIN(THET)*SIN(THET)*SIN(THET) ;
SETPOS.Y = 130* COS(THET)-50*COS(2*THET)-
20*COS(3*THET)-10*COS(4*THET) ;
XP3=SETPOS
PTP P3 CONT Vel=100 % PDAT4 Tool[8]:saha
Base[8]:saha-base
ENDFOR

END

```

Fig. 13.14 Cardoid curve drawn by KUKR KR-5 robot using KRL

13.6.2 Offline Programming

Offline programming, or OLP, takes place on a computer with the models of the workcell containing robots, workpieces, and surroundings. The robot programs can be created by reusing the CAD data generated during its design so that the programming will be quick and effective. The advantages and disadvantages of offline programming are as follows:

Offline

Offline means the programs need not be written on the robot controllers when they are in line with other machines.

Advantages

- No engagement of the robot which can continue functioning during the programming
- Reduced cost as the robot is in operation during programming
- Programs can be changed quickly
- Debugging can be done using some kind of simulators which are generally provided by the robot manufacturers
- Reuse of existing CAD data
- Optimization of the worospace layout and planning for the robot's tasks by rerunning the simulations

Disadvantages

- Need of expert programmers
- Additional investment in an offline programming system

1. Offline Programming Software Robots are highly complex systems. Hence, their precise modeling and close to real-life simulations require sophisticated computational tools, both from the numerical computations aspect as well as computer-hardware requirements. Hence, besides the commercial robot makers, there are many other developments of such software, e.g., *Delmia* from Dassault Systems, *RobCAD* from Technomatrix, *Cosimir* from Festo, and others. Some of the commercially available OLP software are as follows (Pan et al., 2012):

- (i) *RoboStudio* from ABB; (ii) *MotoSim* by Motoman; (iii) *KUKA-Sim* and *CAMrob* from KUKA Robotics; (iv) *Roboguide* by Fanuc; (v) *3D STUDIO* by Staubli; (vi) *MELFA WORKS* from Mitsubishi; and (vii) *Pc-ROSET* by Kawasaki.

These software were made to suitably model the robots made by their company in order to conveniently download the programs to the actual robots.

2. Steps in Offline Programming Typical steps in offline programming (OLP) are explained next (Pan et al., 2012).

CAD Model Generation Offline programming starts with the three-dimensional CAD model of the robots and the workpieces. In this step, one may reuse the CAD models that were used to design and fabricate them. In case such robot models are not available or to capture the workpieces, one may use a three-dimensional scanner or vision-based Microsoft Kinect. The point clouds are converted to surface models after smoothing/filtering the raw CAD data.

Tag Creation It is basically a step to recognize the position of an object. It can be extracted from the feature of a CAD model such as corners and edges, etc.

Motion Planning Since inverse kinematics of an industrial robot, as explained in Chapter 6, has multiple solutions, the robot's configuration must be selected by considering the issues of reachability, collision avoidance, etc. Besides, a suitable trajectory should be selected for a task based on the interpolation functions available with the robot controllers.

Process Planning In case more than one robot is in operation handling a large workpiece or several workpieces, the complete process needs to be planned for the minimization of cycle time.

Post-processing This step includes the addition of input-output control signals for equipment in the workcell with which the robot interacts.

Simulation Simulation allows the users to verify the motion of the real robot in the shop floor, check its collision with the objects in the environment, etc.

Calibration Ideally, the program generated using OLP should be downloaded to a real robot and put into action. However, in reality, there are deviations between the actual geometry of the elements in the workcell such as workpiece and the nominal geometry used in the robot makes the calibration almost compulsory.

3. Features of OLP Robot-programming environments, besides having more features in common with computer-programming environments, present a number of issues related to the effects on the physical world. In other words, even if a very accurate description of physical reality is available in the programming environment, a number of situations will unavoidably occur which have not been or cannot be

predicted. As a consequence, a robot-programming environment should have the following features:

- real-time operating system,
- world modeling,
- motion control,
- input/output,
- sensory data reading,
- interaction with physical system,
- error detection capability,
- recovery of correct operational functions, and
- specific language structure.

One can identify several considerations which must be handled by any robot programming. To illustrate these, note that the motion-control commands that are used to direct a robot to move to some defined configuration, e.g., “move P1”, which might be used to direct the robot to a point in space called P1. Input/output commands are employed to control the receipt of signals from sensors and other devices in the workcell and to initiate control signals to other pieces of equipment in the cell.

Offline programming is based on external means of expressing the task that the robot system has to accomplish. The task is expressed in a *robot language*. This can be either a specially defined language for robots or a universal computer-programming language. The advantage of using robot languages is associated with making the robot more productive, the ease of utilization of sensor data, and creation of program documentation. To make a robot more productive, the phase in which it is required for programming has to be as short as possible. In other words, robot programming has to be made independent of the robot. Two such approaches which are classified as *robot-oriented programming*, and *object-oriented* or *task-level programming*, are explained in the next two sections.

13.6.3 Robot-oriented Programming

Robot-oriented programming is a type of offline programming, as mentioned above. It is a *structured programming* language which can incorporate high-level statements and have the characteristic of an interpreted language in order to obtain an interactive environment allowing the programmer to check the execution of each source program statement before proceeding to the next one. In robot-oriented programming, an assembly task is explicitly described as a sequence of robot motions. The robot is guided and controlled by the program throughout the entire task with each statement of the program roughly corresponding to one action of the robot. Common features of such languages are

- text editor,
- complex data representation structures,
- use of predefined state variables,
- matrix algebra operations,
- symbolic representations for coordinate frames,
- specifying the coordinated motion of more frames rigidly attached to objects by means of single frame,
- subroutines with data and parameter exchange,

- logic conditioning and queuing by means of flags,
- capability of parallel computing, and
- functions of programmable logic controller (PLC).

The most common approach taken in designing a robot-oriented language is to extend an existing high-level computer programming language to meet the requirements of robot programming. There are several characteristics common to all robot-oriented languages. For example, consider the task of inserting a bolt into a hole, as shown in Fig. 13.15. This requires moving the robot to the feeder, picking up the bolt, moving it to the bracket and inserting the bolt into one of the holes. The steps required to develop a program are as follows:

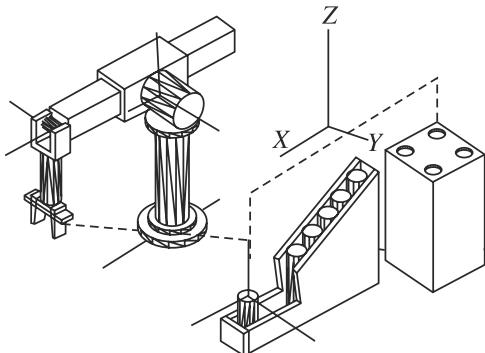


Fig. 13.15 A simple insertion task

- (a) The workspace is set up and the parts are fixed by the use of fixtures and feeders.
- (b) The configuration (orientation and position) of the parts (feeder, bracket, etc.) and their features (bracket-bore, bolt-grasp, etc.) are defined using the data structures provided by the language.
- (c) The assembly task is partitioned into a sequence of actions such as moving the robot, grasping objects, and performing an insertion.
- (d) Sensory commands are added to detect abnormal situations (such as inability to locate the bolt while grasping) and monitor the progress of the assembly task.
- (e) The program is debugged and refined by repeating steps (b) to (d).

The important characteristics are position specification, Step (b), motion specification, Step (c), and sensing, Step (d), which can be programmed in any of the programming-languages, say, AL, and AML, and others.

Example 13.7 AL and AML Definitions for Base Frames

```
AL:    base ← FRAME (nilrot, VECTOR (20, 0, 15)*inches);
       beam ← FRAME (ROT(A, 90*deg), VECTOR (20, 15, 0)*inches);
       feeder ← FRAME (nilrot, VECTOR (25, 20, 0)*inches);
```

Notes: nilrot is a predefined frame which has the value ROT(Z, 0*deg). The “←” is the assignment operator in AL. A semicolon terminates a statement. The “*” is a type-dependent multiplication operator. Here, it is used to append units to the elements of the vector.

```
AML:   base = << 20, 0, 15 >, EULERROT (< 0, 0, 0 >) >;
       beam = << 20, 15, 0 >, EULERROT (< 0, 0, 90 >) >;
       feeder = << 25, 20, 0 >, EULERROT (< 0, 0, 0 >) >;
```

Note: EULERROT is a subroutine which forms the rotation matrix given the angles.

Example 13.8 AL and AML Motion Statements

AL: {Move are from rest to frame A and then to *bold-grasp*}

MOVE *barm* TO *A*;

MOVE *barm* TO *bold-grasp*;

{Another way of specifying the above statement}

MOVE *barm* TO *bold-grasp* VIA *A*;

{Move along the current Z axis by 1 inch, i.e., move relative}

MOVE *barm* TO $\otimes - 1^*Z^*inches$;

Notes: Refer Fig. 13.15 for the point *A*, *Z-axis*, etc. Moreover, *barm* is the name of the robot arm. The sign \otimes indicates the current location of the arm which is equivalent to *base * T6 * E*. Statements inside brackets { . . . } are comments.

AML:

-- Move joint 1 and 4 to 10 inches and 20 degrees, respectively (absolute move)

MOVE (<1, 4>, <10, 20>);

-- Move joints 1, 3 and 6 by 1 inch 2 inches, and 5 degrees, respectively (relative move)

DMOVE (<1, 3, 6>, <1, 2, 5>);

Notes: Statements preceded by “--” are comments.

13.6.4 Task-level Programming

Task-level programming is also a type of OLD that describes the assembly task as a sequence of positional goals of the objects rather than the motion of the robot needed to achieve these goals, and hence no explicit robot motion is specified. A task-level programming system allows the user to describe the task in a high-level language (task specification). A task planner will then consult a database (world models) and transform the task specification into a robot-level program (robot program synthesis) that will accomplish the task. Based on this description, one can conceptually divide task planning into three phases, namely, world modeling, task specification, and program synthesis. It should be noted that these three phases are not completely independent. They are computationally related. Figure 13.16 shows one possible architecture for the task planner. The task specification is decomposed into a sequence of subtasks by the task decomposer and information such as initial state, final state, grasping position, operand, specifications, and attachment relations are extracted. The subtasks are then passed through the subtask planner which generates the required robot program. The concept of task planning is quite similar to the idea of automatic program generation in artificial intelligence. The user supplies the input-output requirements of a desired program, and the program generator then generates a program that will produce the desired input-output behavior. Some of the difficulties and corresponding solutions in task-level programming are given below.

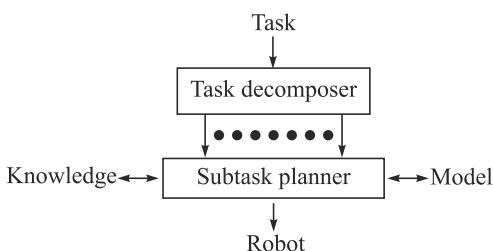


Fig. 13.16 Task planner

1. Geometric and Physical Models For the task planner to generate a robot program that performs a given task, it must have information about the objects and the robot itself. These include the geometric and physical properties of the objects which can be represented by models. A geometric model provides the spatial information (dimension, volume, shape) of the objects in the workspace. Numerous techniques exist for modeling three-dimensional objects required. The most common approach is Constructive Solid Geometry (CSG), where objects are defined as constructions or combinations using regularized set operations (such as union, intersection), of primitive objects (such as cube, cylinder). The primitives can be represented in various ways:

- A set of edges and points
- A set of surfaces
- Generalized cylinders
- Cell decomposition

Physical properties such as inertia, mass, and coefficient of friction may limit the type of motion that the robot can perform. Instead of storing each of the properties explicitly, they can be derived from the object model. However, no model can be 100 percent accurate and identical parts may have slight differences in their physical properties. To deal with this, tolerances must be introduced into the model.

2. Representing World States The task planner must be able to stimulate the assembly steps in order to generate the robot program. Each assembly step can be succinctly represented by the current state of the world. One way of representing these states is to use the configurations of all the objects in the workspace. The AL language provides an attachment relation called AFFIX that allows frames to be attached to other frames. This is equivalent to physically attaching a part to another part and if one of the parts moves, the other parts attached will also move. AL automatically updates the locations of the frames by multiplying the appropriate transformations. For example,

```
AFFIX beam_bore TO beam RIGIDLY;
beam_bore = FRAME(nilrot, VECTOR (1, 0, 0)*inches);
```

13.6.5 Programming using Augmented Reality

Augmented Reality (AR) is an emerging technology that has evolved from virtual reality, or VR. In AR, the computer-generated three-dimensional objects are blended into a real-world scene to enhance the user's interaction with the real world. The robot programming using AR appears to be the concept of the future OLP. As shown in Fig. 13.17, a virtual model of a robot can be moved in a real environment. The virtual robot

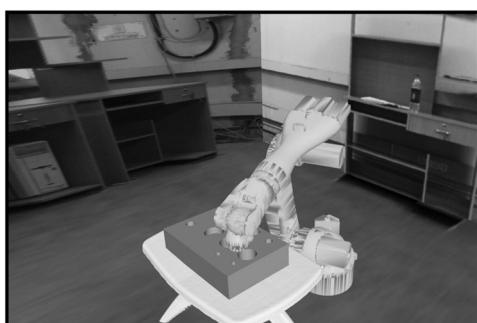


Fig. 13.17 Virtual robot for peg in a hole-insertion task [Courtesy: PAR Lab., IIT Delhi]

model can be moved around to generate a sequence of robot movements that can be calibrated and programmed for an actual peg in a hole-insertion task.

SUMMARY

In this chapter, the software and hardware requirements for robot control are explained. Different robot-programming languages are explained, along with online and offline robot-programming methodologies. Advantages and disadvantages of both programming are mentioned. Examples with a KUKA robot are also provided.

EXERCISES

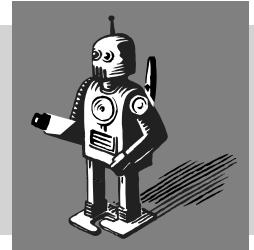
- 13.1** What are the hardware and software requirements for a robot controller?
- 13.2** How can computing speed associated with trigonometric function be enhanced?
- 13.3** Explain controller hardware structure of PUMA robot.
- 13.4** Describe online programming.
- 13.5** What are the advantages and disadvantages of online programming?
- 13.6** What are the types of online programming?
- 13.7** What is the difference between lead-through and walk-through programming?
- 13.8** What are the typical features of a teach pendant?
- 13.9** What are the types of offline programming?
- 13.10** What are the features offline programming environments should have?
- 13.11** When one should prefer online over offline programming?
- 13.12** Name a few robot languages and mention their features.
- 13.13** What are the generations of robot languages?
- 13.14** Mention the feature of generation of robot languages.
- 13.15** Why is task-level programming difficult?

WEB-BASED EXERCISES

- 13.16** What are the types of robot-programming languages used by commercial robot manufacturers?
- 13.17** Name some robot simulation software for offline programming (other than mentioned in the chapter).
- 13.18** Find control architecture of KUKA KR C2 controller.
- 13.19** Which research facilities attempted programming using augmented reality?
- 13.20** Explain at least two assistive devices for walk-through programming.

14

Student Projects



In this chapter, several robotic projects carried out by the students of IIT Delhi are reported. All these projects required some sort of synthesis and design, followed by fabrication and control, or writing software codes. The robots and software presented here are of two types, namely, those which were built as a part of robotic contests, or those developed as a part of the B.Tech and M.Tech final-year projects by the undergraduate and postgraduate students. The first set was strongly pursued by the author for his concept of *Robotics Competition Based Education in Engineering* (RoCK-BEE). RoCK-BEE was introduced by the author for fun and effective learning in robotics. From the experiences of the author, the following aspects should be emphasized for successful implementation of the concept of RoCK-BEE:

1. Proper planning keeping in mind the 5P's (Proper Planning Prevents Poor Performance).
2. Maintaining a project diary by each student to record day-to-day activity, sketches, information, etc.
3. Strictly follow a well-planned Gantt chart. In case the deadlines are not met, reasons are to be found out and measures need to be taken without redefining the Gantt chart.
4. It is extremely important that students learn how to work in a team. In bigger projects like ROBOCONs, as explained in Section 14.1, the coordinators of different teams, say, Mechanical, Electrical, Fabrication, etc., should distribute the work amongst other members of his or her group. Otherwise, they may end up doing most of the jobs themselves, while others have no job. The latter group may get frustrated, and even may leave the project too.

Above all, dedication, sincerity, honesty, and positive thinking are a must amongst members for successful participation in bigger competitions, where the robots actually move and perform their intended task, rather than just winning a game.

Besides the hardware or software made by the students, there are commercial products like RoboCNC, KUKA YouBot, LEGO and Fischertechnik kits, etc., which can also be used for student projects to understand particular aspects of robotics. These products encourage students to do the programming to control the robots that are provided by the company or made by the students themselves using the components of the kits. These relatively economical products are unlike industrial robots which are expensive and do not allow their users to access the controllers. A few such products are discussed in Section 14.4.

14.1 ROBOTIC CONTESTS

ROBOCON is an abbreviation for ROBOTic CONtest. It is a student robotic competition which is organized by the Asian Broadcasting Union (ABU) every year since 2002. In order to select a representative team from India, an Indian competition is organized in March every year by Doordarshan (Indian Television Broadcasting Company). So far, it has been held in Japan (2002, 2009), Thailand (2003, 2011), South Korea (2004), China (2005), Malyasia (2006), Vietnam (2007, 2013), India (2008, 2014), Egypt (2010), and Hongkong (2012).

Under the guidance of the author, teams from IIT Delhi have been participating in the Indian ROBOCON since 2003. In the following two subsections, experiences from ROBOCON 2014 when India would host the international ABU-ROBOCON 2014, and ROBOCON 2007 when IIT Delhi was the Indian champion to represent the country in Vietnam are explained.

14.1.1 ROBOCON 2014

Under the theme of *A Salute to Parenthood* in this year's ROBOCON, a *parent robot* (manual robot) has to be controlled by a human operator to carry a *child robot* (automatic robot). The child robot has to autonomously play the games of Seesaw, Swing, Pole-walk, and Jungle Gym once put by the parent robot on those areas. The game field is shown in Fig. 14.1. Under a set of rules, the game continues and the teams score points.

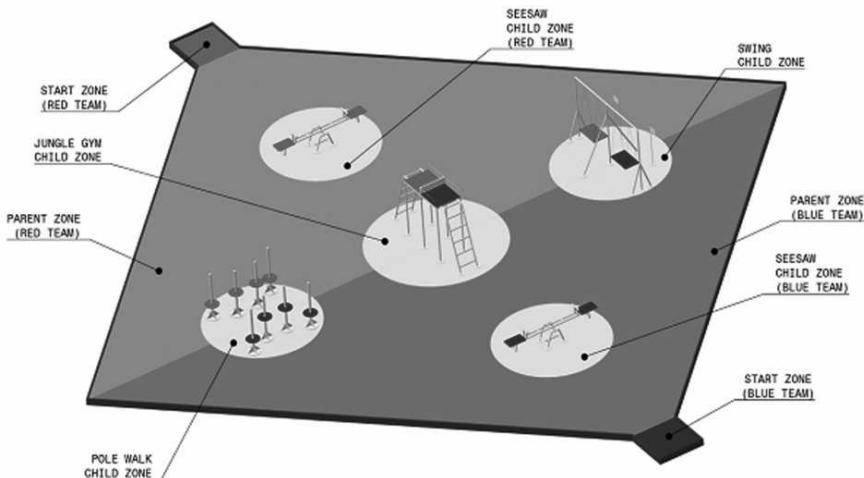


Fig. 14.1 Game field of ROBOCON 2014

Typical steps involved in designing the robots for such competitions are like any new product design by a company. Hence, the participating activities in such competitions quite well simulate the real-world industrial product-development cycle, thereby, making the students industry-ready. The following design steps were followed:

1. Understand the rules of the games provided by the organizer of the competitions in August 2013.

2. Divide the team of about 20 students into four smaller teams: Two teams were to generate at least 2 ideas for the manual robot each (total of four ideas). Similarly, four ideas were to be generated for the automatic robot.
3. Discuss the ideas with all team members and some senior members, which included faculty members, technical staff, and senior students.
4. Upon tentative finalization of the two ideas (one for manual and one for automatic), they were checked for availability of components and fabrication methods.
5. Regrouped the team into two subgroups: One for manual and another for automatic.
6. While the mechanical student members prepared the CAD drawings shown in Fig. 14.2 for manufacturing, the electrical students planned for the electrical circuitry and programming.

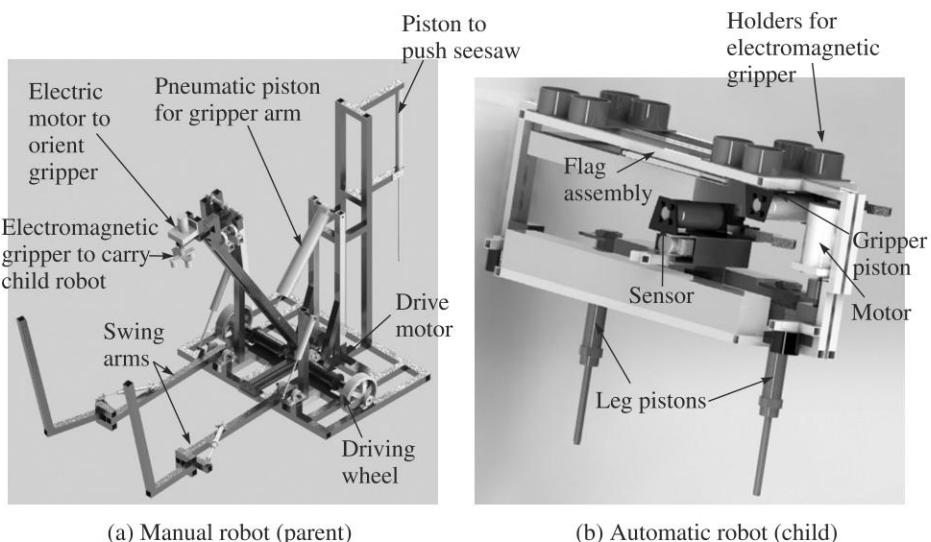


Fig. 14.2 Robots for ROBOCON 2014

7. As per the Gantt chart shown in Fig. 14.3, the manufacturing of the components were carried out during the winter break of the students in December 2013.
8. Assembly and testing were conducted in January 2014.
9. During regular meetings with the senior members, good practices from both electrical and mechanical domains were highlighted to the students. For example, if long electrical wires are to be connected between the motors and the batteries, they must be provided with filters. Otherwise, there may be some alternating current components to the motors due to the inductance of the wires. From mechanical domain, wheel shafts are to be double-bearings supported instead of a cantilever arrangement.
10. Finally, the robots were planned to be transported to the venue of the competition (Pune, India) from New Delhi, India, by the end of February, 2014.

	Task	Start	End	Dur	smartDraw 2013			
					Sep	Oct	Nov	Dec
1	Manual Robot	17/9/13	26/12/13	69				
2	Design Visualization	17/9/13	21/9/13	4				
3	Advance ordering of components	22/9/13	23/9/13					
3.1	Design Modeling	23/9/13	15/10/13	16				
3.2	All components and assembly	23/9/13	28/9/13	5				
3.3	minors	29/9/13	7/10/13	6				
3.4	CAD + Simulation and analysis of cad	8/10/13	15/10/13	5				
4	List of items required	16/10/13	17/10/13	2				
5	Buying all the items	17/10/13	18/10/13	2				
6	Winter Break	19/10/13	27/10/13	5				
7	Fabrication	28/10/13	7/12/13	28				
7.1	Drive	28/10/13	29/10/13	2				
7.2	Individual mechanisms	30/10/13	12/11/13	9				
7.3	Major	16/11/13	29/11/13	9				
7.4	Assembly	1/12/13	4/12/13	3				
7.5	Wiring	5/12/13	7/12/13	2				
7.6	piping	5/12/13	7/12/13	2				
8	Mechanical iterations	8/12/13	12/12/13	4				
9	Final practise	13/12/13	26/12/13	9				

Fig. 14.3 Gantt chart for the manual robot of ROBOCON 2014

14.1.2 ROBOCON 2007

In this competition also, two types of robots were made as per the rules of the game.

One was the *automatic* type whose job was to transfer blocks from the outer edges of the 10-sided polygon in the game field shown in Fig. 14.4 to the vertices of the triangles at the center. It was a preprogrammed robot. Hence, the term *automatic* was used. The other type was *manual* which was controlled using a switchboard by a human operator. The task of this robot was to transfer blocks from the corners of the square field to the edges of the 10-sided polygon. Each successful transfer accrued points, and the game proceeded. Both the automatic and manual robots had a gripper each to hold the blocks of about 300 mm diameter. It was important to synthesize an appropriate mechanism which was holding the blocks with soft materials so that the objects did not get crushed. At the same time, the blocks should not slip. Another aspect was lifting and placing. The automatic and

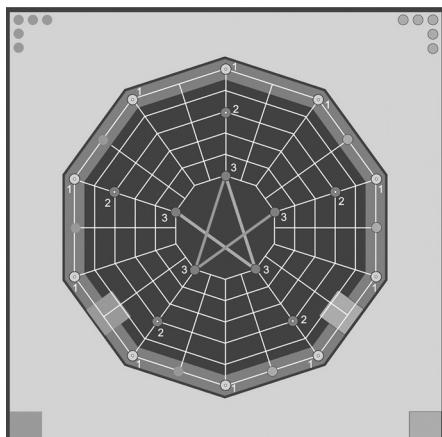


Fig. 14.4 Game field of ROBOCON 2007

manual robots are shown in Figs. 14.5(a) and (b), respectively. The automatic robot used a pulley arrangement to lift the gripper assembly, while a four-bar parallelogram mechanism was used in the manual robot. The latter could extend almost about half a meter in front of the robot to be able to place the blocks inside the boundary line of the 10-sided polygon. Once the design was done, the next challenge was to fabricate, assemble, program, and make them run successfully. Steps similar to Section 14.1.1 were followed as per the Gantt chart shown in Fig. 14.6.

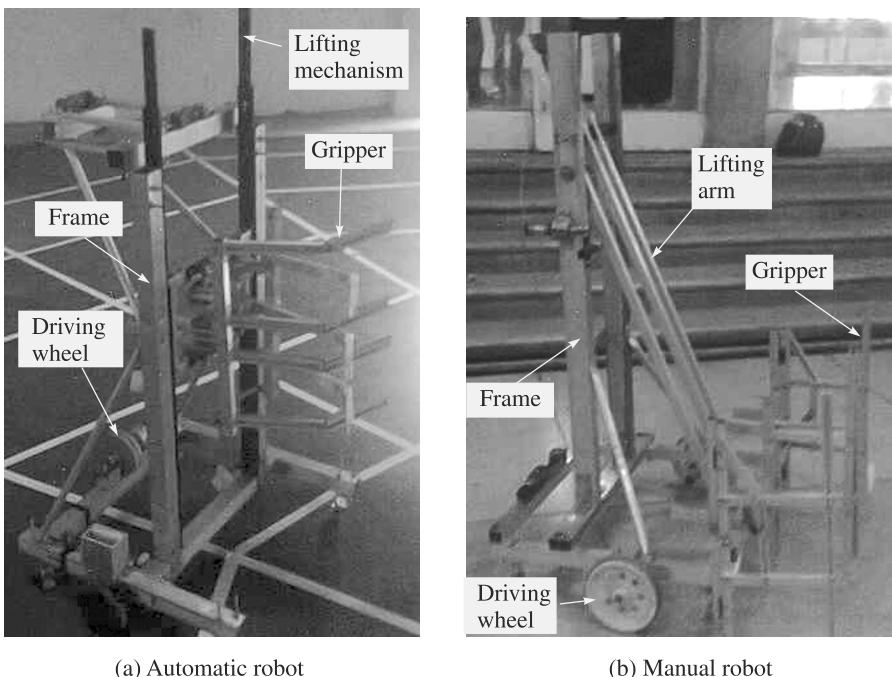


Fig. 14.5 Robots for ROBOCON 2007

14.2 HARDWARE DEVELOPMENTS

In this section, several other hardware developments of robots are explained.

14.2.1 RoboMuse

RoboMuse, as shown in Fig. 14.7, is a line-following mobile robot capable of handling few kilograms of payload. It was originally manufactured as a part of ROBOCON 2008 by IIT Delhi's robotics team. After participation in the event, a live 24×7 demo of the robot was planned in the Student Activity Centre of the campus. The salient features of the RoboMuse were as follows:

1. It had three Maxon motors: Two were for driving wheels and one to lift the charging rod up and down.
2. Supply voltage was 12 V dc from a lead-acid battery (current capacity: 4.3 Ah) which used to charge automatically when the robot was in the docking configuration.

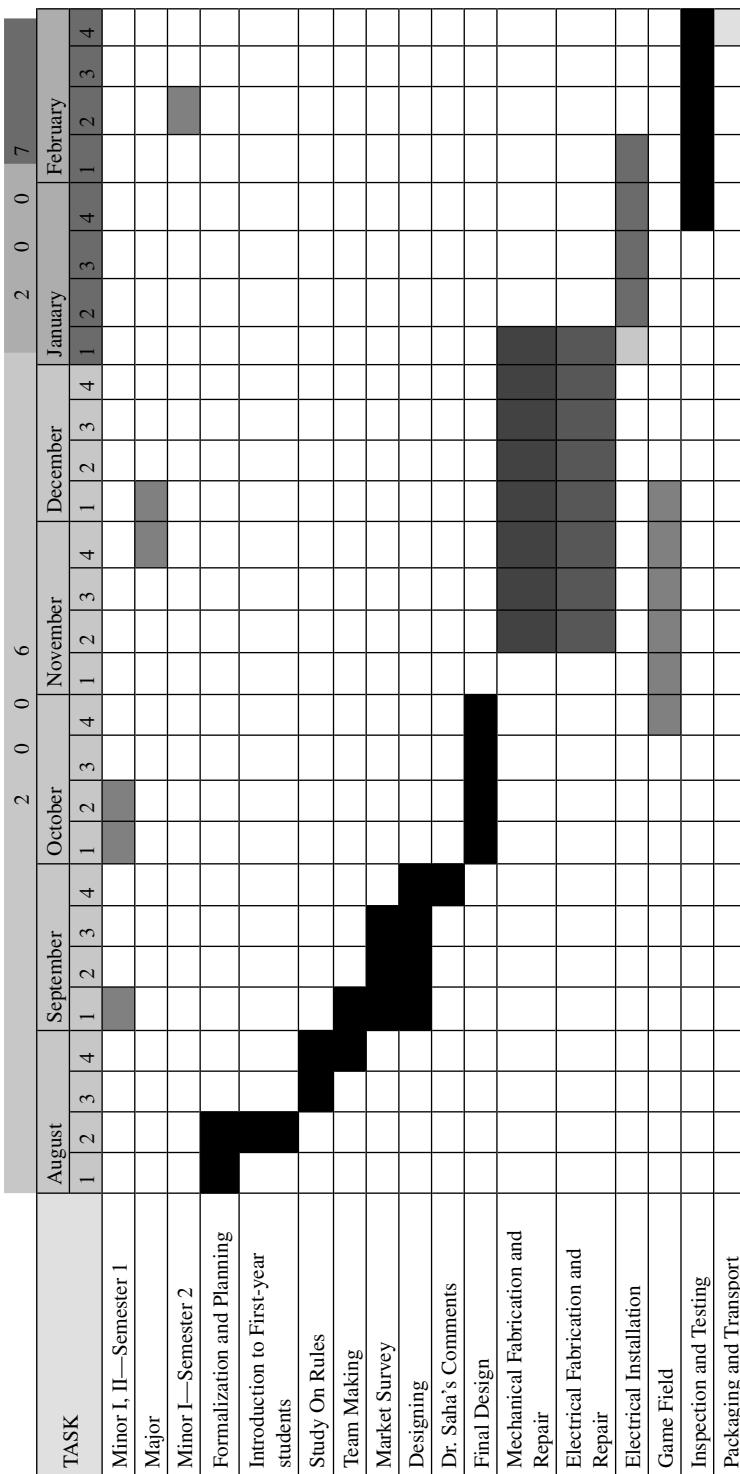


Fig. 14.6 Gantt chart for ROBOCON 2007

3. A series of six IFM fiber-optic sensors were used to detect the line on the floor.
4. To initiate the robot motion, wireless module (433 MHz radio frequency) was used.
5. An interactive console consisting of a 16 cm × 4 cm LCD and a 3 × 4 keypad were used for entering the password to run the system (password was the answer to a simple puzzle).
6. All actions were controlled using the PIC18F4550 control board.

The robot was supposed to move along a straight line of about 1.5 meters, take 180° turn and come back to the home position to dock at the charging station to charge the batteries. There were two versions. One was made in 2008 and another one was made in 2009. Both the projects worked for about 3–4 months each before they were discontinued, mainly due to the lack of maintenance by the students who graduated. The project, however, helped the students appreciate the importance of reliability, maintainability, and robustness of a robot which has to work continuously in an industrial environment.

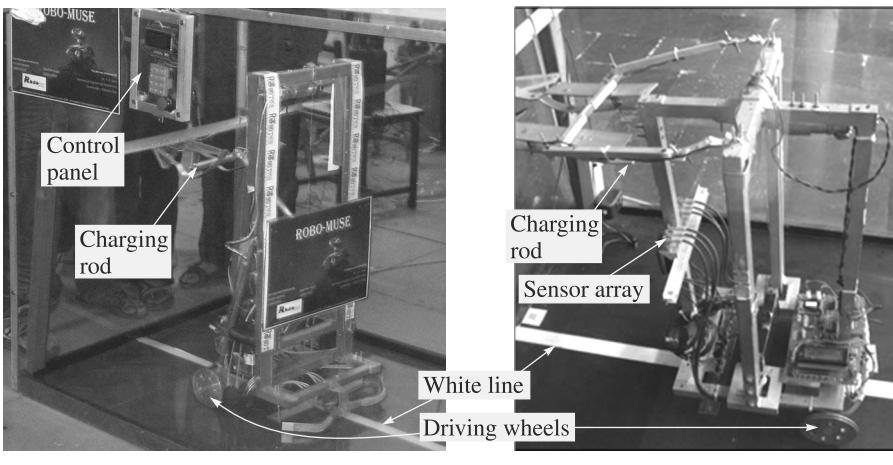


Fig. 14.7 RoboMuse mobile robots

14.2.2 Six-DOF Parallel Robot

In this B.Tech final-year project (Bhagat and Choudhury, 2008), a six-degrees-of-freedom parallel robot was analyzed and fabricated. It was basically a closed-loop Stewart platform, as mentioned in Chapter 1 of this book. Such system is suitable for applications in motion simulations, e.g., as flight simulators, precision positioning devices, etc. It had six prismatic joints in six legs which could expand and contract to actuate the top platform with respect to the bottom fixed platform, as shown in Fig. 14.8. At the bottom of the leg, it had a universal joint and a spherical joint on the top. Due to parallel architecture in such robots, they possess high stiffness and high payload-to-weight ratio. As a result, these robots expected to provide better accuracy compared to the serial-chain robots used in the industries.

In this project, architecture was taken from another research project where a parallel manipulator was intended to be used as a truck simulator. This system was

the scaled-down architecture. The system was first modelled in Autodesk Inventor to check the functioning of different components of the assembly, as shown in Fig. 14.8(a). The final assembly is shown in Fig. 14.8(b), which was actuated with six Firgelli L-12 actuators.

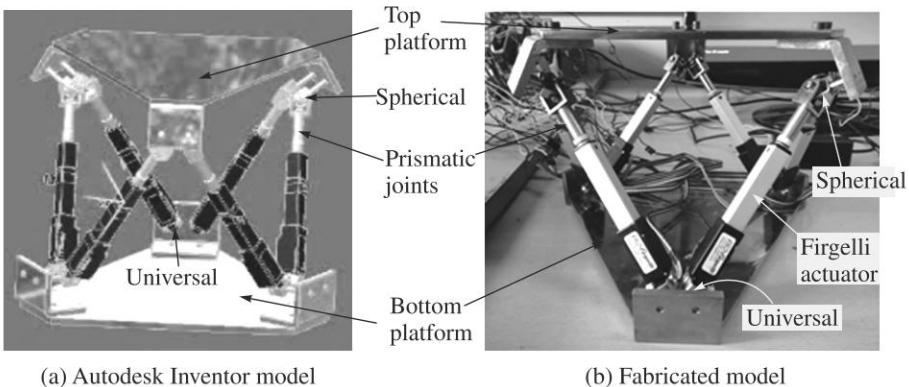


Fig. 14.8 Six-DOF parallel robot

Control of the leg movements was carried out through the LabView software in a PC. As shown in Fig. 14.9(a), two National Instrument (NI) Data Acquisition or DAQ cards were used to send signals to the motors from the LabView software. Actuators had two inputs, namely, position control signal (0-5 V) from DAQ Cards, and supply voltage of 6 V dc from a power supply. Actuators returned analog voltage feedback proportional to their extensions. For the actuator displacements, inverse kinematics calculations were used. LabView commands and C-codes were used for the programming. Two types of programs were made, namely, the real-time control and pre-defined trajectories. See Fig. 14.9(b) for the LabView control panel. Based on final testing, the specifications obtained are shown in Table 14.1, which were compared with a commercially available PI-M840 micro-positioner. The B.Tech project was awarded the *BOSS Award for Best Hardcore Experimental Project in Mechanical Engineering 2010* at IIT Delhi. Later, the results were published in an international conference (Bhagat et al., 2011).

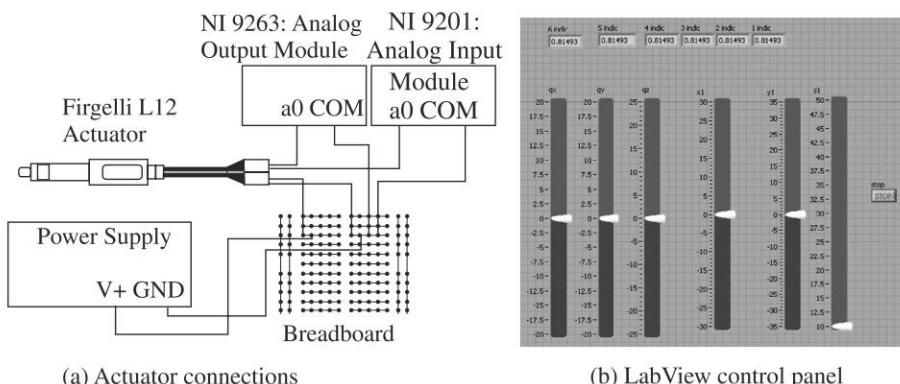


Fig. 14.9 Control of 6-DOF parallel robot

Table 14.1 Specifications of the 6-DOF parallel robot (Stewart platform)

Parameters	Bhagat and Choudhury (2008)	Commercial: PI-M 840 (WR: PI)	Remarks
Load carrying capacity	3.3 kg*	10 kg	*Not tested to failure
Actuator stroke length	50 mm	25 mm	Better in Bhagat and Choudhury (2008)
Motion range along Z	0-58.5 mm	\pm 25 mm	Specifications of Bhagat and Choudhury (2008) are comparable with the commercial one.
Motion range along Y	-60.5 to +42.5 mm	\pm 50 mm	
Motion range along X	\pm 46 mm	\pm 50 mm	
Rotation about Z (yaw)	\pm 28°	\pm 30°	
Rotation about Y (roll)	\pm 18.3°	\pm 15°	
Rotation about X (pitch)	-18° to +15°	\pm 15°	
Accuracy of motion	\pm 1 mm	\pm 1 μ m	
Maximum velocity along Z	12.5 mm/s	50 mm/s	
Maximum velocity along Y	20.3 mm/s	50 mm/s	
Maximum velocity along X	25.6 mm/s	50 mm/s	
Maximum velocity about Z (yaw)	14.5°/s	34°/s	
Maximum velocity about Y (roll)	8.1°/s	34°/s	
Maximum velocity about X (pitch)	8.9°/s	34°/s	

14.2.3 Hanging Planar Robotic Arm (HaPRA)

As a part of mainly two B.Tech projects, the three-degrees-of-freedom Hanging Planar Robotic Arm (HaPRA) was developed. In the first project (Venugopal and Shabeer, 1996), an industrial task of lifting cylindrical objects, e.g., piston heads, from one conveyor and placing it to another conveyor was considered. It is depicted in Fig. 14.10. The configuration of the robot arm, along with its dimensions, etc. were evaluated in this first project, which was presented in a conference (Venugopal and Saha, 1997).

In the second B.Tech project (Sharma and Kumar, 1997), fabrication of the same was carried out. During the second project, emphasis was given on low-cost aspect. Since the payload was only about 500 grams, easily available chain-drives, namely, bicycle chains, were chosen. The complete control was achieved using stepper motors and a PC. The fabricated arm with its control equipment is shown in Fig. 14.11. The detail design and development work was presented in another conference (Baghla et al., 1999).

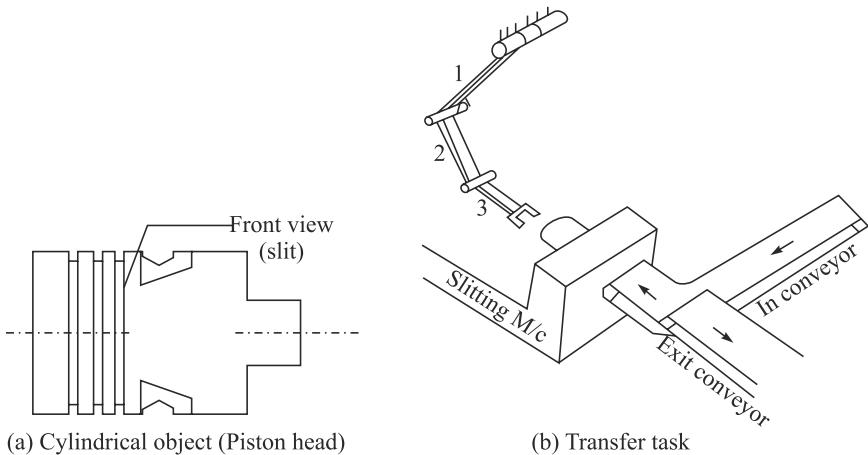


Fig. 14.10 An industrial task by a robot arm

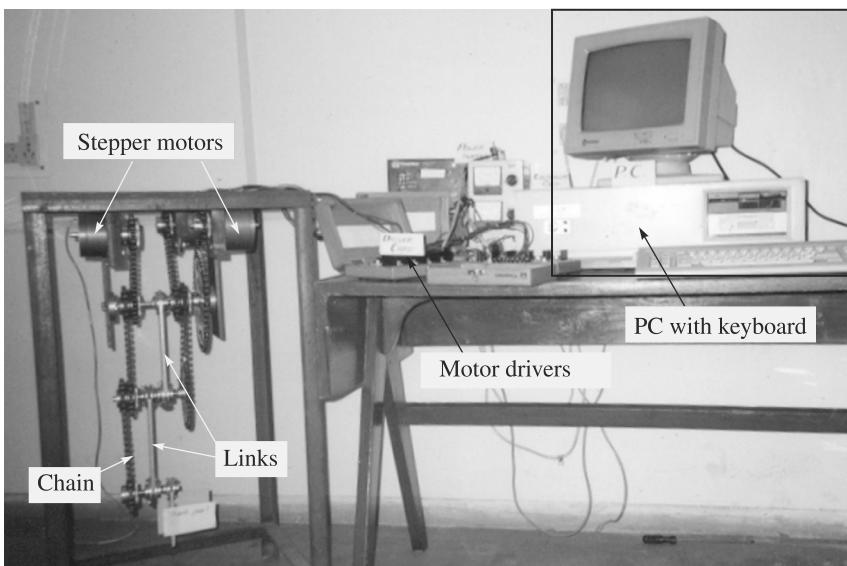
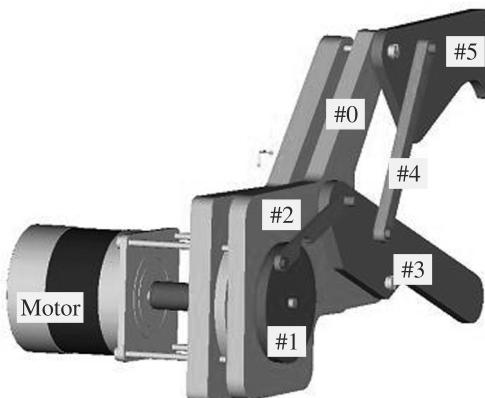


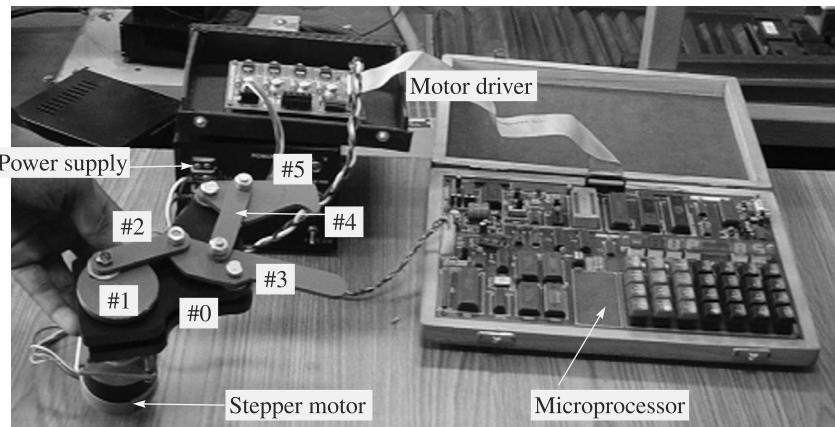
Fig. 14.11 Photograph of the HaPRA with its controller

14.2.4 Gripper for Cylindrical Objects

This gripper was designed in another B.Tech final-year project keeping in mind that the same can be used by HaPRA explained in Section 14.2.3 (Agarwal and Singh, 2004). For a range of cylindrical shapes, the gripper was synthesized as a six-bar linkage, as shown in Fig. 14.12(a). Its complete kinematic analysis was performed using the motion module of Pro-E software. Finally, the gripper was fabricated and interfaced with a stepper motor to demonstrate its motion capabilities, Fig. 14.12(b). The project was awarded in IIT Delhi as the *Padma Shri Man Mohan Suri Best Hardware B.Tech Project Award for the Year 2004*.



(a) Pro-E drawing



(b) Control-interface

Fig. 14.12 Gripper for cylindrical objects

14.2.5 Direct-Drive Robot

The direct-drive robot shown in Fig. 14.13 was developed in two M.Tech final-year projects. In the first one, the kinematic and dynamic analyses were performed and the arm containing two degrees of freedom was fabricated (Barma, 2001). Later, a third axis was added in another M.Tech project (Ramvath, 2004), which should be useful for real application of the robot in pick-n-place operations. The detail design methodology was presented in a national conference (Ramvath et al., 2005).

14.3 SOFTWARE PROJECTS

In this section, two software developments made by the students of IIT Delhi are reported which would help one to design a gripper to hold regular-shaped objects like cylinders, cubes, etc., and to decide the dimensions of the wheel base of a mobile robot. These were made freely available through an online link <http://www.roboconhelp.com> which helps students build various types of robots.

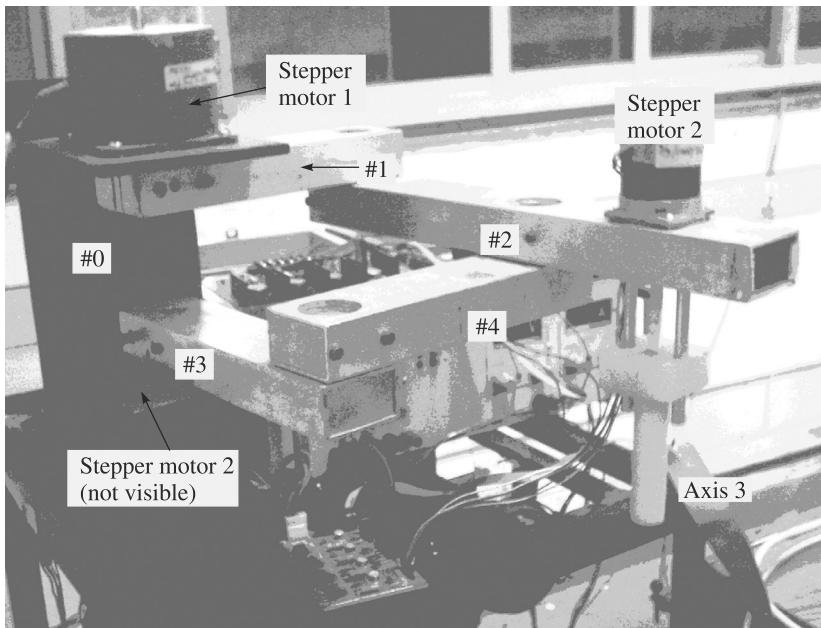


Fig. 14.13 Direct-drive robot arm

14.3.1 Gripper Design using SiRoD 1.0

Simple Robot Designer (SiRoD) is a software developed as a part of two B.Tech final-year projects (Mittal et al., 2009, and Sethi et al., 2011) for the design of gripper to hold regular-shaped objects like cylinders, cubes, and spheres. In this version of SiRoD 1.0, the software can provide a gripper design on the basis of inputs given by the user. It takes inputs from the user in a Microsoft Excel file and then processes the inputs to give the dimensions as per the templates of the links of the gripper mechanism. While Figs. 14.14(a-b) show the screenshot of the SiRoD 1.0, Fig. 14.14(c) shows a typical gripper design for the inputs provided.

Since regular shapes of cylinder, cube and sphere are used in plenty in our daily lives, the need was felt to synthesize a gripper which should be able to hold such shapes. That way, ROBOCON students could also be helped as their objects often fall in these categories. Based on several mechanisms available, the project chose the one, namely, the one shown in Fig. 14.14(c), which could handle a range of sizes and weights shown in Table 14.2. The lengths, cross sections and material of the links were chosen based on the friction surfaces, bending strengths, etc., by considering the least factor of safety of 2.5. Almost 900 iterations were performed. Cross sections were then found and compared with a database comprising standard sizes available in the market. Final dimensions are those which can be easily procured and assembled to get the desired gripper. This process involved extensive changes in the Visual Basic code in-built in Microsoft Excel. Both the projects received awards at IIT Delhi, namely, *2009 BOSS Award for Best Hardcore Experimental Project in Mech. Eng.* and *2011 Suresh Chandra Memorial Trust Award*, respectively.

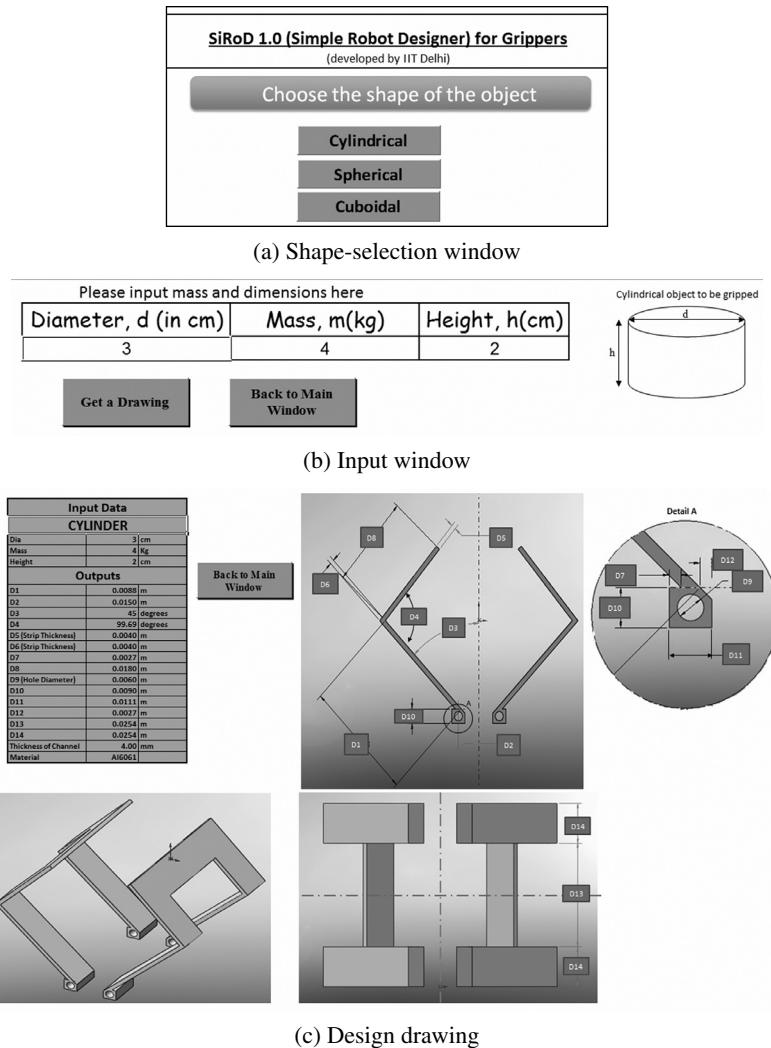


Fig. 14.14 A gripper design using SiRoD 1.0

Table 14.2 Ranges of sizes and weights for the gripper in SiRoD 1.0

Cylinder	Cube	Sphere
$0 \text{ kg} < \text{Mass} < 10 \text{ kg}$	$0 \text{ kg} < \text{Mass} < 10 \text{ kg}$	$0 < \text{Mass} < 10 \text{ kg}$
$2 \text{ cm} < \text{Diameter} < 20 \text{ cm}$	$4 \text{ cm} < \text{Diameter} < 27 \text{ cm}$	$2 \text{ cm} < \text{Side} < 20 \text{ cm}$
$0 \text{ cm} < \text{Height} < 500 \text{ cm}$	$0 \text{ cm} < \text{Height} < 500 \text{ cm}$	

14.3.2 Wheel Base for Mobile Robots using SiRoD 1.3

This is a development over three projects. It started with the one which is called as *Summer Undergraduate Research Assistantship* (Rathore and Jain, 2009) in IIT Delhi available to the second-year students. Later, it continued over two other

final-year B.Tech projects (Sethi et al., 2011, and Kumar and Kishan, 2012). They have numbered it SiRoD 1.1, 1.2, and 1.3, respectively. Three typical wheel-base configurations shown in Fig. 14.15(a) were finally implemented in SiRoD 1.3. Two of the wheels are used as driving two wheels in a differential-drive mode, and others act as castors. The following analyses were carried out for the final implementation of the wheel-base design of a mobile robot:

1. Preliminary Torque Analysis The preliminary torque analysis was performed using the concepts of balancing forces and moments on different components of the robot. Simplified equations for the torque required for the robot to achieve the required acceleration were then obtained in terms of the normal reaction on the wheel, the radius, mass and moment of inertia of the wheel.

2. Maximum Torque to Prevent Slipping The theory of maximum traction, as given in Gillespie (1992), was used to estimate the maximum torque that the motors can provide without slipping occurring at the wheels. This can then be used to estimate the maximum acceleration that the robot can achieve in reality for the chosen size of wheel.

3. Toppling Analysis The basic check for toppling was done by calculating the equivalent normal reaction on each wheel according to the loading like the mass of the robot, the position of the centre of mass, and the acceleration to which it is subjected. If the normal reaction on any wheel is less than or equal to zero then there is a possibility of toppling.

4. Dynamic Modelling of a Mobile Robot In order to simulate the above design, kinematic and dynamic modeling of the wheel configurations were performed in MATLAB environment, and using RecurDyn multibody dynamics simulation software (WR: Function Bay).

5. Design and Analysis The process of deciding the wheel-base configuration of a mobile robot was not simple, as it involved consideration of a number of factors like load distribution, points of contact, required size of base, etc. It was not possible to analyse each and every configuration. Thus, in order to generalize the process of base selection before incorporating it in the software, those used in the industries and by the students of IIT Delhi's ROBOCON teams for the last eight years were studied. Few basic wheel configurations were identified for analyses, as indicated in Fig. 14.15(a).

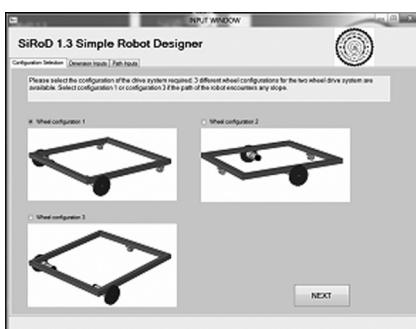
According to the analysis, the optimal designs of the wheel base for different load conditions, different base lengths and different length-to-breadth ratios were obtained in terms of the material and dimension of square channels to be used. The designs were obtained through iterative finite element analysis for 150 sets of design values. Similarly, the design of wheels, as shown in Fig. 14.16, was carried out using SolidWorks Simulation Xpress for finding the optimal thickness of the wheel and the torque acting on the wheel. The number of holes was also varied from 4 to 6 as per the radius of the wheel. Based on the above considerations, an in-house (SiRoD 1.3) software was developed in Microsoft Visual Basic 2008 environment. The software model will provide the user with the design of the base module on the basis

of inputs. The design of the wheel-base frame and wheel itself are based on the look-up tables generated for both. The look-up tables list the design conditions for which the analysis had been performed and the resulting optimal base dimensions. The calculated values of these conditions according to the user inputs are then mapped into the look-up table and the corresponding result of the dimension is then provided as output to the user. The inputs required are

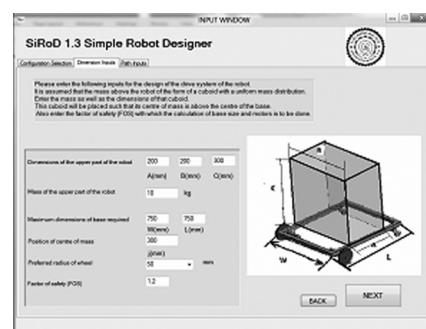
- Approximate position of center of mass of the mobile base
- Minimum/Maximum base dimensions
- Maximum velocity and acceleration desired
- Trajectory configurations (maximum slope on the path, minimum turning radius, etc.)

On the basis of the above inputs, the SiRoD 1.3 software will provide the wheel-base design consisting of the following outputs:

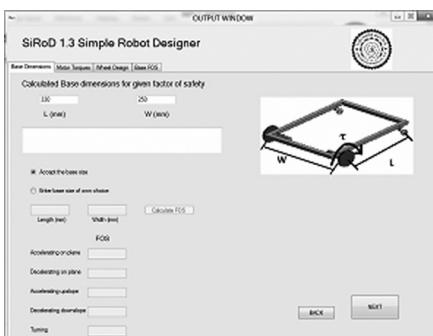
- Toppling and slipping condition
- Optimum base dimensions
- Design of wheel
- Torque and power required by the driving motors
- Database for selection of motors according to design provided



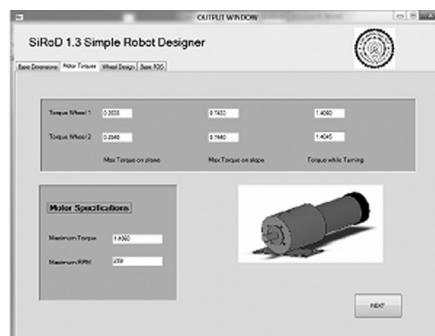
(a) Choice of wheel-base configurations



(b) Specify overall size and CM location



(c) Calculated size of the base



(d) Specification of motors

Fig 14.15 Wheel-base design using SiRoD 1.3

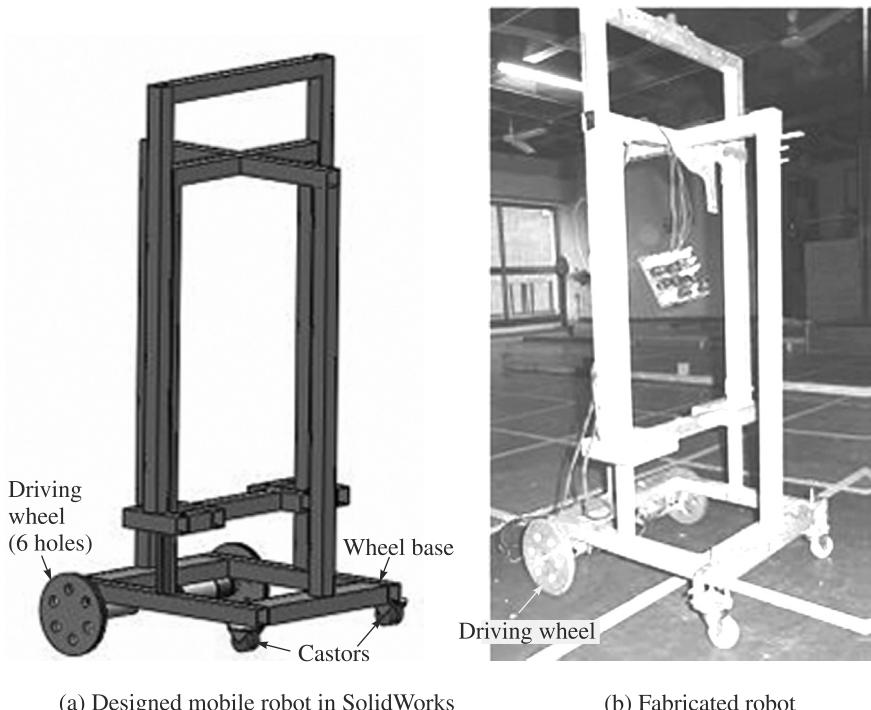


Fig. 14.16 Designed and fabricated mobile robot

In order to check how reliable or how close are the performances of the designed mobile robot base, a base with a chassis on it was designed as shown in Fig. 14.16(a). The same was fabricated which is shown in Fig. 14.16(b). The fabricated robot was instrumented for simple motion along a straight line with and without slopes. The experiments performed on it were as follows:

1. Initially keep payload of 10 kg on the robot at any height and run the robot horizontally along a straight line.
2. If it topples then reduce its payload height. Set it to a height such that the robot just topples. Note down the height of the payload and calculate the position of the center of mass.
3. Now, to measure acceleration corresponding to the toppling at that height, lower the height a little such that the robot does not topple anymore. The acceleration was measured indirectly by measuring the time duration for a short run of the robot. The distance taken was small such that the robot does not attain its maximum velocity. Then, acceleration was found from $s = ut + (1/2)at^2$, where s is the distance traversed, u is the initial velocity which was taken as zero, and t is the elapsed time.
4. Repeat the procedure for the cases of different slopes.
5. Feed the above calculated values in the software. Compare the outputs of designed and fabricated robots.

The results are tabulated in Table 14.3, which can be interpreted as fairly close keeping in mind that the effect of bearing frictions, joint clearances, etc., were ignored.

Table 14.3 Comparison of toppling of the mobile robot

SN	Parameters	Distance (m)	Time (s)	Acceleration (m/s ²)		Remarks
				Experimental	SiRoD 1.3	
1	Accelerating on horizontal plane	0.5	0.72	1.93	2.28	SiRoD overestimates
3	Accelerating on a slope of 10°	0.5	0.75	1.77	1.94	SiRoD overestimates
2	Accelerating on a slope of 17°	0.5	0.98	1.05	0.97	SiRoD underestimates

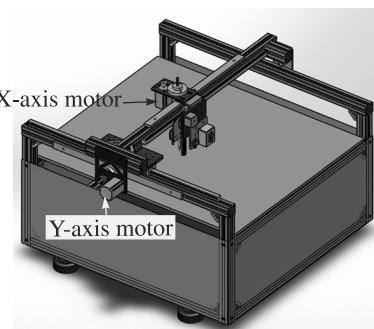
14.4 COMMERCIAL PRODUCTS

In this section, some commercial robotic products will be introduced which can also be used to learn different aspects of robotics, e.g., control programming, etc. Some of them are listed below.

14.4.1 RoboCNC

RoboCNC, as shown in Fig. 14.17, is a four-axis robot-training platform developed by SVP Laser Technologies Pvt. Ltd. The RoboCNC is basically a Cartesian robot with a camera. Such architecture can be used in industries for applications like Computer Numeric Controlled (CNC) profile cutting machines, 3D printers, milling, Laser-, Plasma-, Gas-, Waterjet-, and electro-discharge-machining, Coordinate Measuring Machines (CMMs), overhead crane robot,

auto-inventory retrieval systems, pick and place, palletizing, etc. Basic tools on the RoboCNC include a pen plotter for writing, hot knife for foam cutting, camera, red laser, and an electromagnetic gripper. The robot can be programmed at an operator level in G-codes, and also at a developer level (using .NET for front end programming and using Ladder-logic PLC programming at an embedded level).



[Courtesy: <http://www.svlaser.com>]

14.4.2 KUKA YouBot

KUKA youBot is an omni-directional mobile platform with an arm on it, as shown in Fig. 14.18. It is from the manufacturer of the KUKA KR-5 robot explained in earlier chapters. However, the latter being an industrial robot does not allow access to its controller for various control implementations. On the contrary, the YouBot has the following key features:

- Open interfaces
- On-board PC, mini ITX PC-Board with embedded CPU, 2 GB RAM, 32 GB SSD Flash, USB

- Basic control software
- Real-time EtherCAT communication

Besides, it has

- Omni-directional mobile platform
- Five degrees-of-freedom manipulator
- Two-fingers gripper
- The above arm and mobile platform can be used independently as well

Due to its open interfaces the robot can be configured in several applications for the learning of various aspects of robotics, e.g., vision, sensor integration, etc.

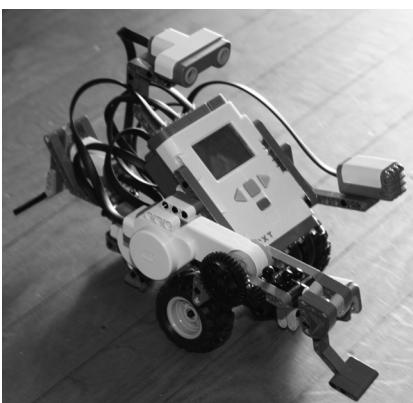
14.4.3 LEGO and Fischertechnik Kits

LEGO kits are very popular for building, basically, toys and also many robotic devices. It comes with its own programmable brick (e.g., Mindstorm NXT) to control the robots made using its plastic components. The programs can be made in a PC using the company provided Graphics User Interface (GUI) which can be transferred to the brick. Figure 14.19(a) shows a LEGO robot with the NXT brick controller and other accessories. Fischertechnik is another toy company which sells components to build models as well. Figure 14.19(b) shows its robot to test several sensors. Its ROBO TX controller can be programmed and used to move the robots. Both the kits are quite suitable for the preliminary-level visualization of a product by joining small pieces of links, gears, cables, etc., before they can be attached with several sensors and motors to make a full-fledged system as used in real life.



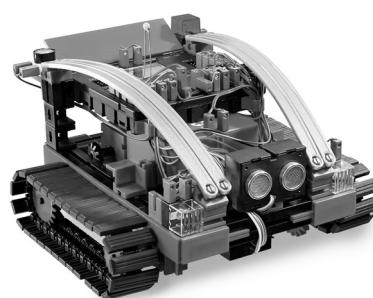
Fig. 14.18 KUKA YouBot

[Courtesy: <http://www.youbot-store.com/>]



(a) LEGO

[Courtesy: <http://www.lego.com/en-us/mindstorms/>]



(b) Fischertechnik

[Courtesy: <http://www.fischertechnik.de/en/Home.aspx>]

Fig. 14.19 Robots using kits

SUMMARY

In this chapter, several robotics hardware and software projects carried out by the students of IIT Delhi are reported. Without going into the details of the design calculations, only the outlines of the philosophies are provided to give an idea to the readers as to how a robot can be designed or programmed to run. Besides, some commercial products are also explained which can be used as student projects as well to understand various aspects of robotics.

EXERCISES

- 14.1** What is ROBOCON?
- 14.2** What are the important aspects for successful implementation of a big robotic project?
- 14.3** What are the degrees of freedom of a parallel robot reported in this chapter?
- 14.4** Name few applications of a parallel robot.
- 14.5** Why are direct-drive robots preferred over serial-type industrial robots?
- 14.6** What is the name of the software which can be used to design a gripper for cylindrical, cubical, and spherical objects?
- 14.7** Find out a robot-gripper dimensions to hold a cylinder of 25 mm diameter and 7 kg weight using SiROD 1.0 software.
- 14.8** What are the inputs required to design a mobile wheel base using SiROD 1.3 software?
- 14.9** What is a programmable bricket?

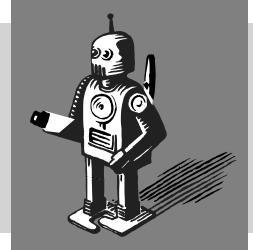
WEB-BASED EXERCISES

Based on Web search, find the answers for the following questions:

- 14.10** Where did the ROBOCON competition start?
- 14.11** Find at least two commercial companies who sell grippers.
- 14.12** Name two companies who sell stepper motors.
- 14.13** Who sells simulators?
- 14.14** Name a couple of commercially available direct-drive robots.
- 14.15** Name a few toy companies who make robotic kits.

A

Mathematical Fundamentals



In this book, it is assumed that the reader has familiarity with the mathematical definitions from the areas of Linear Algebra, Trigonometry, and other related areas. Some of them are introduced in this Appendix to make the readers quickly understand the derivations and notations used in this book.

A.1 FUNCTION ‘ATAN2’

The usual inverse tangent function denoted by $\text{atan}(z)$, where $z = y/x$, returns an angle in the range of $(-\pi/2, \pi/2)$. In order to express the full range of angles, it is useful to define the so-called two-argument arctangent function denoted by $\text{atan2}(y, x)$, which returns an angle in the entire range, i.e., $(-\pi, \pi)$. This function is defined for all $(x, y) \neq 0$, and equals the unique angle θ such that

$$\cos \theta = \frac{x}{\sqrt{x^2 + y^2}}, \text{ and } \sin \theta = \frac{y}{\sqrt{x^2 + y^2}} \quad (\text{A.1})$$

The function uses the signs of x and y to select the appropriate quadrant for the angle θ , as explained in Table A.1.

In Table A.1, $z = y/x$, and $\text{Sgn}(\cdot)$ denotes the usual sign function, i.e., its value is -1 , 0 , or 1 depending on the positive, zero, and negative values of y , respectively.

However, if both x and y are zeros, ‘atan2’ is undefined. Now, using the table, $\text{atan2}(-1, 1) = -\pi/4$ and $\text{atan2}(1, -1) = 3\pi/4$, whereas the $\text{atan}(-1)$ returns $-\pi/4$ in both the cases.

x	$\text{atan2}(y, x)$
+ve	$\text{atan}(z)$
0	$\text{Sgn}(y) \pi/2$
-ve	$\text{atan}(z) + \text{Sgn}(y) \pi$

A.2 VECTORS

Unless otherwise stated differently, a vector will be understood as an array of numbers written in a column. It is indicated in this book with lower-case bold letters. An n -dimensional vector \mathbf{a} is then defined as

$$\mathbf{a} \equiv \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \quad (\text{A.2})$$

where a_1, \dots, a_n are the elements of the vector \mathbf{a} . The vector \mathbf{a} can also be represented as

$$\mathbf{a} = [a_1, \dots, a_n]^T \quad (\text{A.3})$$

in which superscript T denotes transpose. The magnitude or length or norm of the vector \mathbf{a} , denoted with italicics letter a , is given by

$$a = \sqrt{\mathbf{a}^T \mathbf{a}} = \sqrt{a_1^2 + \dots + a_n^2} \quad (\text{A.4})$$

A.2.1 Unit Vector

A unit vector is defined as a vector with unit magnitude. Hence, the unit vector along vector \mathbf{a} , which is denoted with $\bar{\mathbf{a}}$ can be defined as

$$\bar{\mathbf{a}} = \frac{\mathbf{a}}{a} \quad (\text{A.5})$$

where a is given by Eq. (A.4). Hence, $\bar{a} = 1$. Now, if \mathbf{i}, \mathbf{j} , and \mathbf{k} denote the unit vectors along the axes X , Y , and Z , respectively, of a coordinate frame shown in Fig. A.1, any 3-dimensional Cartesian vector shown in Fig. A.1, say, $\mathbf{a} \equiv [a_1, a_2, a_3]^T$, can be expressed as

$$\mathbf{a} = a_1 \mathbf{i} + a_2 \mathbf{j} + a_3 \mathbf{k} \quad (\text{A.6})$$

in which the unit vectors, \mathbf{i}, \mathbf{j} , and \mathbf{k} , have the following representations:

$$\mathbf{i} \equiv \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{j} \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } \mathbf{k} \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{A.7})$$

Note in Eq. (A.6) that the magnitudes of the unit vectors using Eq. (A.4) are one.

A.2.2 Scalar and Dot Product

The scalar product of two n -dimensional vectors \mathbf{a} and \mathbf{b} , denoted as $\mathbf{a}^T \mathbf{b}$, is a scalar number defined by

$$\mathbf{a}^T \mathbf{b} = a_1 b_1 + \dots + a_n b_n \quad (\text{A.8})$$

The scalar product is commutative, i.e.,

$$\mathbf{a}^T \mathbf{b} = \mathbf{b}^T \mathbf{a} \quad (\text{A.9})$$

Now, for 2- and 3-dimensional Cartesian vectors representing physical quantities like the position of a point or force at a point, etc., the scalar product is also referred as dot product, which is given by

$$\mathbf{a}^T \mathbf{b} = ab \cos \theta \quad (\text{A.10})$$

where θ is the angle between the vectors \mathbf{a} and \mathbf{b} , as shown in Fig. A.2. Physical interpretation of the dot product is that the vector \mathbf{b} is projected onto the vector \mathbf{a} , as indicated by OA in Fig. A.2, and is multiplied by the magnitude of \mathbf{a} . Alternatively, it

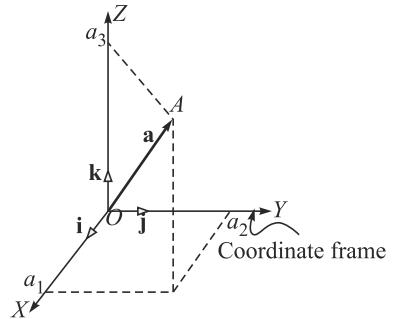


Fig. A.1 Coordinate frame and unit vectors

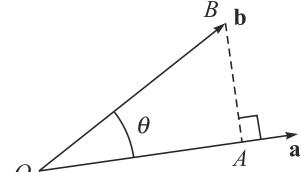


Fig. A.2 Scalar (dot) product of two Cartesian vectors

can be interpreted as the projection of \mathbf{a} onto the vector \mathbf{b} whose result is multiplied by the magnitude of the latter. Now, if these two vectors are orthogonal to each other, i.e., if $\theta = 90^\circ$, the dot product vanishes, namely, $\mathbf{a}^T \mathbf{b} = 0$.

A.2.3 Vector or Cross Product

A vector or cross product between two Cartesian vectors, say, \mathbf{a} and \mathbf{b} , denoted by \mathbf{c} , is defined as

$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2 b_3 - a_3 b_2) \mathbf{i} + (a_3 b_1 - a_1 b_3) \mathbf{j} + (a_1 b_2 - a_2 b_1) \mathbf{k} \quad (\text{A.11a})$$

where \times denotes the symbol for the cross product, whereas $| \bullet |$ represents the determinant of the arguments \bullet . The result of Eq. (A.11a) can also be expressed as

$$\mathbf{c} \equiv \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} \quad (\text{A.11b})$$

In Eq. (A.11a), the result of the cross product is also a vector \mathbf{c} , as indicated in Eq. (A.11b), which is orthogonal to the two vectors \mathbf{a} and \mathbf{b} . The magnitude of vector \mathbf{c} is denoted as c that can be given by

$$c = ab \sin \theta \quad (\text{A.12})$$

where θ is the angle between the vectors \mathbf{a} and \mathbf{b} , as shown in Fig. A.3. In order to obtain the direction of the resultant vector \mathbf{c} , the right-hand rule is applied, i.e., if the palm of a right hand is placed along the vector \mathbf{a} , and then turned towards the vector \mathbf{b} , the thumb points out in the direction of the vector \mathbf{c} . It is also shown in Fig. 13.13.

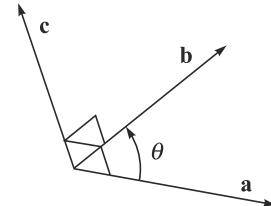


Fig. A.3 Vector (cross)-product of two Cartesian vectors

The cross product has the following properties:

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}; \mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a}^T \mathbf{c}) \mathbf{b} - (\mathbf{a}^T \mathbf{b}) \mathbf{c}; \text{ and } \mathbf{a}^T (\mathbf{b} \times \mathbf{c}) = |\mathbf{a} \mathbf{b} \mathbf{c}| \quad (\text{A.13})$$

A.2.4 Cross-Product Matrix

A cross-product matrix is always associated with a three-dimensional Cartesian vector, say, \mathbf{a} . When the matrix is pre-multiplied with another three-dimensional Cartesian vector, say, \mathbf{b} , it results in the cross product between the two vectors, as shown in Section A.2.3. If $\mathbf{a} \times \mathbf{1}$ denotes the cross-product matrix associated with the vector \mathbf{a} then

$$(\mathbf{a} \times \mathbf{1}) \mathbf{b} = \mathbf{a} \times \mathbf{b} \quad (\text{A.14a})$$

The 3×3 matrix $(\mathbf{a} \times \mathbf{1})$ is a skew-symmetric matrix and singular. Its representation in terms of the components of the vector $\mathbf{a} \equiv [a_1 \ a_2 \ a_3]^T$ is given by

$$\mathbf{a} \times \mathbf{1} \equiv \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \quad (\text{A.14b})$$

It is now a simple matter to verify that Eq. (A.14b) results from Eq. (A.11b).

A.2.5 Differentiation of Vectors

If the vector $\mathbf{a} \equiv [a_1 \ \dots \ a_n]^T$ is the n -dimensional vector function of time, then its time derivative, denoted by $\dot{\mathbf{a}}$, is defined as

$$\dot{\mathbf{a}} \equiv [\dot{a}_1 \ \dots \ \dot{a}_n]^T \quad (\text{A.15})$$

The scalar and vector products satisfy the following product rules for differentiation similar to the product rule for differentiation of ordinary functions:

$$\frac{d}{dt}(\mathbf{a}^T \mathbf{b}) = \dot{\mathbf{a}}^T \mathbf{b} + \mathbf{a}^T \dot{\mathbf{b}}; \text{ and } \frac{d}{dt}(\mathbf{a} \times \mathbf{b}) = \dot{\mathbf{a}} \times \mathbf{b} + \mathbf{a} \times \dot{\mathbf{b}} \quad (\text{A.16})$$

Similar statements hold for the integration of vectors also.

A.2.6 Linear Independence

A set of vectors $\mathbf{a}_1, \dots, \mathbf{a}_n$ is said to be linearly independent if and only if

$$\sum_{i=1}^n \alpha_i \mathbf{a}_i = \mathbf{0} \text{ implies } \alpha_i = 0 \text{ for all } i \quad (\text{A.17})$$

A.3 MATRICES

A matrix is defined as an ordered array of numbers. An $m \times n$ matrix, say, \mathbf{A} , with m rows and n columns has mn elements denoted as a_{ij} , for $i = 1, \dots, m$; and for $j = 1, \dots, n$. If each column of the matrix \mathbf{A} is represented by the m -dimensional vector $\mathbf{a}_j \equiv [a_{1,j} \ \dots \ a_{m,j}]^T$, for $j = 1, \dots, n$ then the matrix \mathbf{A} has the following representation:

$$\mathbf{A} \equiv [\mathbf{a}_1 \ \dots \ \mathbf{a}_n] \quad (\text{A.18a})$$

One can similarly define each row as the n -dimensional vector $\mathbf{a}_i \equiv [a_{i,1} \ \dots \ a_{i,n}]^T$, for $i = 1, \dots, m$, and can represent the matrix \mathbf{A} as

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix} \quad (\text{A.18b})$$

where $\mathbf{a}_i^T \equiv [a_{i,1} \ \dots \ a_{i,n}]$. Note that both Eqs. (A.18a-b) represent the same matrix. Depending on the requirements, one may choose one over the other.

A.3.1 Determinant and Inverse

The determinant and inverse of a matrix are defined for square matrices only. For an $n \times n$ matrix \mathbf{A} , its determinant is a scalar denoted by $\det(\mathbf{A})$ or $|\mathbf{A}|$, whereas the inverse is denoted with \mathbf{A}^{-1} . Their definitions are readily available in any text book on Linear Algebra, e.g., in Datta (1999), Strang (1980), and others. Some properties are however cited here for quick reference.

- If all the elements of a row or column of the matrix are zero then $\det(\mathbf{A}) = 0$.
- Determinants of \mathbf{A} and its transpose \mathbf{A}^T are equal, i.e., $\det(\mathbf{A}) = \det(\mathbf{A}^T)$.
- For two matrices, \mathbf{A} and \mathbf{B} , $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$.
- If \mathbf{A} is an identity matrix, i.e., $\mathbf{A} = \mathbf{I}$ then its determinant is unity, i.e., $\det(\mathbf{A}) = 1$.
- If two rows or columns are dependent then $\det(\mathbf{A}) = 0$. Such matrices are called singular.

- If all the rows or columns are independent then $\det(\mathbf{A}) \neq 0$. Such matrices are called non-singular.
- $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{1}$, where $\mathbf{1}$ is an identity matrix.
- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$, and $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$.
- $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$.

A.3.2 Eigenvalues and Eigenvectors

The eigenvalues of the matrix \mathbf{A} are the solutions in s of the following equation:

$$\det(s\mathbf{1} - \mathbf{A}) = 0 \quad (\text{A.19})$$

where $\mathbf{1}$ is the identity matrix of same dimension as matrix \mathbf{A} . The function $\det(s\mathbf{1} - \mathbf{A})$ is a polynomial in s called *characteristic polynomial* of \mathbf{A} . If s_e is an eigenvalue of \mathbf{A} , an eigenvector of \mathbf{A} corresponding to s_e is a nonzero vector \mathbf{x} satisfying the system of linear equations given by

$$(s_e\mathbf{1} - \mathbf{A})\mathbf{x}_e = 0 \quad (\text{A.20})$$

A.3.3 Jacobian

The Jacobian is a term used in mathematics which represents a multi-dimensional form of the derivatives. For example, consider m functions in which each one is a function of n independent variables, i.e.,

$$\mathbf{y} = \mathbf{f}(\boldsymbol{\theta}) \quad (\text{A.21a})$$

$$\text{where } \mathbf{y} \equiv [y_1 \ \cdots \ y_m]^T; \mathbf{f} \equiv [f_1 \ \cdots \ f_m]^T; \text{ and } \boldsymbol{\theta} \equiv [\theta_1 \ \cdots \ \theta_n]^T \quad (\text{A.21b})$$

In Eq. (A.21b), \mathbf{y} and \mathbf{f} are the m -dimensional vectors, and $\boldsymbol{\theta}$ is the n -dimensional vector. Now, the time derivative of \mathbf{y} with respect to $\boldsymbol{\theta}$ can be given by

$$\dot{\mathbf{y}} = \mathbf{J}\dot{\boldsymbol{\theta}}, \text{ where } \mathbf{J} \equiv \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \quad (\text{A.22a})$$

The matrix \mathbf{J} is the $m \times n$ Jacobian matrix. Its (i, j) th entry is defined as $\partial f_i / \partial \theta_j$, i.e.,

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \equiv \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \cdots & \frac{\partial f_1}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1} & \cdots & \frac{\partial f_m}{\partial \theta_n} \end{bmatrix} \quad (\text{A.22b})$$

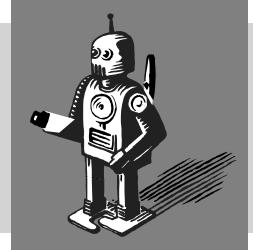
With respect to a six-degrees-of-freedom robot shown in Fig. 6.6, the terms $\dot{\mathbf{y}}$ and $\dot{\boldsymbol{\theta}}$ correspond to the 6-dimensional vectors of end-effector twist \mathbf{t}_e , and the joint rates $\dot{\boldsymbol{\theta}}$, respectively, whereas the 6×6 Jacobian matrix can be obtained using Eq. (6.86) or (6.105b).

SUMMARY

In this appendix, some basics on linear algebra are presented which would help the readers to understand the concepts used in this book, mainly, in chapters 5–12.

B

MATLAB with Allied Toolbox and Software



In this appendix, steps to use the commercial software MATLAB (WR: Matlab) and allied software like MuPAD, Simulink are explained. Two independent developments based on MATLAB software, e.g., Robotics Tool Box for MATLAB (Corke, 2011), and Recursive Dynamic Simulator or ReDySim (Shah et al., 2013) are also explained. A good account of how to use MATLAB is given in Pratap (2010).

B.1 USE OF MATLAB

MATLAB is a commercial software by MathWorks Inc., USA. It has large number of mathematical operators and commands that can perform a wide range of analysis, e.g., matrix operations, algebraic and differential equation solutions, optimizations, control experiments, etc.

B.1.1 Initialize MATLAB

It is expected that MATLAB is installed in a computer where the user will be performing computations. If an icon is available on the Desktop, the user has to double-click on it using the left button of the mouse connected to the computer. Alternatively, the user can left-click, in sequence, on the button/menus that pop up: Start -> Programs -> Matlab-> MATLAB.

B.1.2 How to use MATLAB

When MATLAB software starts, the MATLAB screen appears with the “>>” prompt. This window is called MATLAB command window. Some basic operations are shown here. For detailed description, one may refer to books available on MATLAB, e.g., Pratap (2010), and use the “Demos” and “Help” menus of the software.

B.1.3 Vectors in MATLAB

For any column vector or what is simply referred in this book as a vector, say, $\mathbf{a} \equiv [a_1 \ a_2 \ a_3]^T$, the following commands are used:

```
>> syms a1 a2 a3  
>> av = [a1; a2; a3]  
av =  
[a1]  
[a2]  
[a3]
```

where “`syms`” is a MATLAB command to define the variables as *symbolic*, instead of numerical. Moreover, “`av`” is a user defined variable name for the vector **a**. In contrast, a row vector, say, $\mathbf{a} \equiv [a_1 \quad a_2 \quad a_3]$, can be generated as

```
>> syms a1 a2 a3
>> av = [a1 a2 a3]
av =
[a1, a2, a3]
```

If the elements of the row vector are numbers, say, $\mathbf{a} \equiv [2 \quad 1 \quad 5]$ then

```
>> av = [2 1 5]
av =
2      1      5
```

Addition of two vectors, say, $\mathbf{a} \equiv [2 \quad 1 \quad 5]^T$ and $\mathbf{b} \equiv [1 \quad 4 \quad 2]^T$, are obtained as

```
>> av = [2; 1; 3];
>> bv = [1; 4; 2];
>> cv = av+bv
cv =
3
5
5
```

The scalar product of the above two vectors **a** and **b** is obtained as

```
>> av' * bv
ans =
12
```

where the prime (`'`) operator in MATLAB means *transpose*. Moreover, no variable is assigned on the left of the operation. Hence, it is given in MATLAB as “`ans`” meaning *answer*.

B.1.4 Matrices in MATLAB

Let two matrices, **A** and **B**, are given by

$$\mathbf{A} \equiv \begin{bmatrix} 2 & 3 & 1 \\ 3 & 5 & 2 \\ 5 & 1 & 6 \end{bmatrix}; \text{ and } \mathbf{B} \equiv \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 8 & 2 & 9 \end{bmatrix} \quad (\text{B.1})$$

Their representations in MATLAB are as follows:

```
>> am = [2 3 1; 3 5 2; 5 1 6]
am =
2      3      1
3      5      2
5      1      6
>> bm = [1 2 3; 2 3 5; 8 2 9]
bm =
1      2      3
2      3      5
8      2      9
```

Addition of the above two matrices **A + B** is obtained as

```
>> am+bm
ans =
3      5      4
5      8      7
13     3      15
```

whereas the multiplication of **A** and **B**, denoted by **AB**, is given by

```
>> am*bm
ans =
16    15    30
29    25    52
55    25    74
```

Besides algebraic operations, one can find the properties of the matrices as well, e.g., determinant, eigenvalues, etc. For example, the determinant of **A** is calculated as

```
>> det(am)
ans =
10
```

whereas eigenvalues of **A** are obtained using the following command:

```
>> eig(am)
ans =
9.2085
0.3121
3.4794
```

Moreover, the inverse of **A** can be evaluated as

```
>> inv(am)
ans =
2.8000   -1.7000    0.1000
-0.8000    0.7000   -0.1000
-2.2000    1.3000    0.1000
```

Finally, the solution of a set of algebraic equations written in the form of $\mathbf{Ax} = \mathbf{b}$, i.e.,

$$\begin{aligned} 2x + 3y + z &= 1 \\ 3x + 5y + 2z &= 4 \\ 5x + y + 6z &= 2 \end{aligned} \quad (\text{B.2})$$

is given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Note that the matrix **A** is given in Eq. (B.1), whereas the vector **b** has appeared in Section B.1.3. Vector **x** of Eq. (B.2) is defined as $\mathbf{x} \equiv [1 \ 4 \ 2]^T$.

The solution **x** in MATLAB is obtained as

```
>> xv = am\bv
xv =
-3.8000
1.8000
3.2000
```

One can now check the multiplication of the matrix **A** with the vector **b** as

```
>> am*xv
ans =
1.0000
4.0000
2.0000
```

which is same as the vector **b**.

It is pointed out here that whatever has been done above in the MATLAB command window can also be typed in a .m file, say, “soln.m” which can then be run from the MATLAB window or clicking “Debug->Run” in the file editor. Here, the software may prompt for MATLAB path setting. The user should specify the one where “soln.m” is saved. In either case, the results will appear in the MATLAB command window.

B.1.5 Ordinary Differential Equations

A set of differential equations can be solved using the in-built commands of MATLAB, e.g., “ode45”, etc. These commands require that the ODEs are written in the first-order state-space form shown in Eq. (8.126). It can be solved as follows:

```
>>%% The following commands can be used in the command window also.
>>tspan = [0 10]; y0 = [pi/2; 0]; %Initial conditions of the state
variables.
>> [t,y] = ode45('ch8fdyn1',tspan,y0) % Calling "ode45" function
```

In the third line above, ‘ch8fdyn1’ is the name of a .m file containing the first-order state-space description of the equation of motion. Its contents are as follows:

```
>>% Contents ch8fdyn1.m
>>%For one-link arm
>>function ydot = ch8fdyn1(t,y); %Function defines the state-space
form.
>>m = 1; a = 2; g = 9.81; tau = 0; % System parameters
>>iner = m*a*a/3; grav = m*g*a/2;
>>ydot = [y(2);(tau-grav*sin(y(1)))/iner]; %State-space equations
```

B.1.6 Plots

MATLAB has plot commands that provide the variation of, say, y data with time t of Section B.1.5. For example,

```
>> plot [t,y]
```

will give the plot shown in Fig. 8.20.

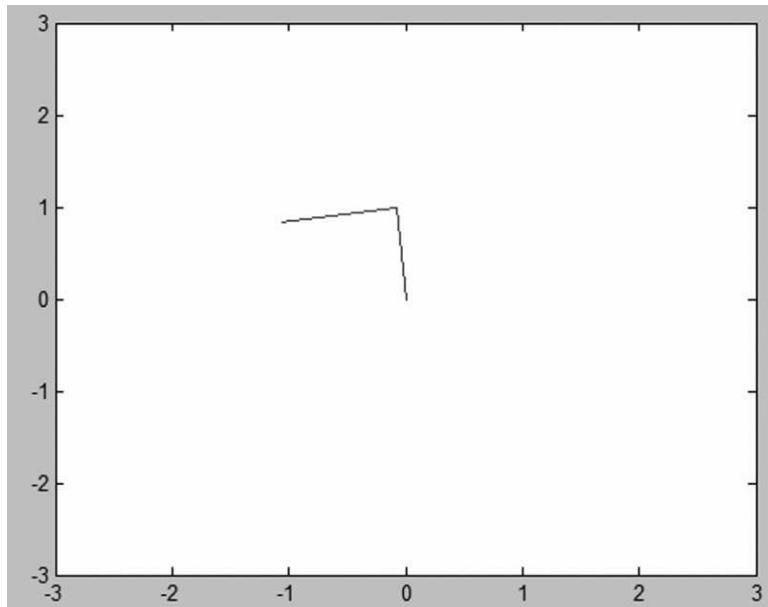
B.1.7 Animation

MATLAB provides a feature of animation also. A simple set of commands as shown in Fig. B.1(a) can generate an animation whose screenshot is shown in Fig. B.1(b).

```
>>% Animation of a 2-link robot arm
>>a1 = 1; a2 = 1; % Link lengths
>>t = 0:0.01:4/3; % Time
>>xmin = -(a1+a2)*1.5; xmax = (a1+a2)*1.5;
>>ymin = -(a1+a2)*1.5; ymax = (a1+a2)*1.5;
>>for i = 1:length(t)
>>dth1 = 2*pi*45/60; dth2 = 2*pi*45/60; % Velocity
>>th1 = dth1*t(i); th2 = dth2*t(i); % Displacement
>>xy1 = [a1*cos(th1); a1*sin(th1)];
>>xy2 = xy1+[a2*cos(th1+th2); a2*sin(th1+th2)];
>>xy = [[0; 0] xy1 xy2];
>>plot(xy(1,:),xy(2,:))
>>axis([xmin xmax ymin ymax]);
>>drawnow % Draws at every instant of the lines
>>end
>>% MATLAB program ends
```

(a) MATLAB program for animation

(Contd.)



(b) Animation screenshot

Fig. B.1 Animation program of a two-link robot manipulator

B.2 MuPAD

MuPAD is a package for computer algebra. In the recent versions of MATLAB, MuPAD can be accessed with a simple command “mupad”. A new window pops up which is the MuPAD notebook where formulas can be typed, as shown in Fig. 5.17. With the press of “Enter” button, results are displayed. Referring to Fig. B.2, some commands and syntax of MuPAD are explained below:

- Variable assignment is done with “:=” and not with “=” as in MATLAB. Thus, one must type “y := a*x^2+b*x+c” or “a := 10” in the MuPAD notebook. While the input command texts appear in red, the output that immediately follows the input is displayed in blue.
- Output suppression is done with “:” at the end of the command and by NOT “;” as in MATLAB. Thus, to suppress the output, one must type “y := a*x^2+b*x+c:.”
- Recall of the previous answer is done with “%” and not “ans” as done in MATLAB. Thus, typing “subs (%,x = a)” will substitute a for x in the previous answer.
- One can find the solution of the function “y” using the command “soln := solve (y,x).”
- An expression, e.g., “f := (a+b)^2”, can be expanded using the command “g := expand(f).”
- Using some numerical values for the function of “y” above, e.g., $a = 2$, $b = 3$, and $c = 4$, the function is plotted for the range of x values, say, $x = 1, \dots, 10$. It is shown at the bottom of Fig. B.2.

```

y:=a*x^2+b*x+c
ax2 + b x + c
subs(%,x=a)
a3 + b a + c
soln:=solve(y,x)
{ { -b+sqrt(b2-4 a c) / (2 a), -b-sqrt(b2-4 a c) / (2 a) } if a ≠ 0
{ { -c/b } if a = 0 ∧ b ≠ 0
C if a = 0 ∧ b = 0 ∧ c = 0
∅ if a = 0 ∧ b = 0 ∧ c ≠ 0
f:=(a+b)^2
(a+b)2
g:=expand(f)
a2 + 2 a b + b2
ynum:=subs(y,a=2,b=3,c=4)
2 x2 + 3 x + 4
fplot:=plot(ynum,x=1..10)

```

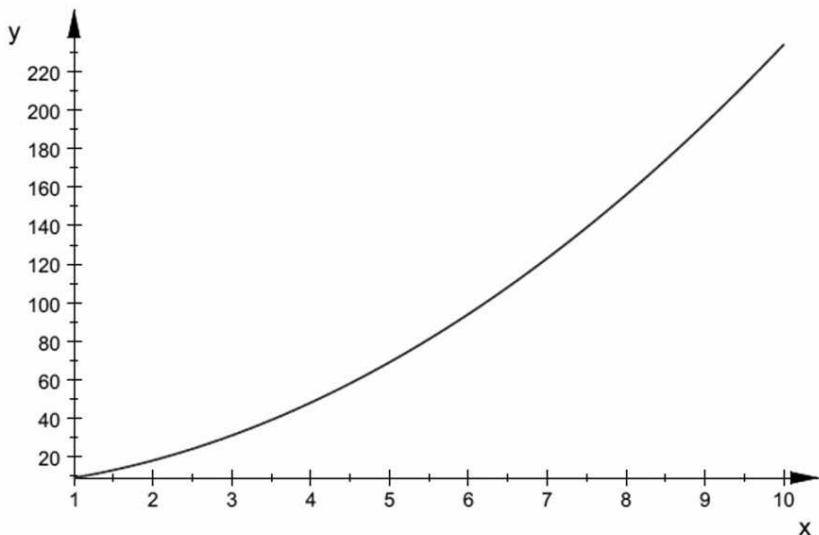


Fig. B.2 MuPAD screenshot

B.3 SIMULINK

It is a toolbox used to model a system in block-diagram form. Once the MATLAB window is open upon typing “simulink,” a new window pops up with the name of a Simulink module and elementary blocks, as shown on the left of Fig. B.3. Once a new model is selected from the drop-down “file” menu (top-left), another window pops up where one can draw several boxes by choosing elementary blocks. This is indicated with “Two systems modeled.” Clicking the run button on the top of this window, analysis is performed whose results can be seen by double-clicking the “scope1” and “scope2” appearing on the right of “Two systems modeled.” The plots are then visible on the right. Any system parameter which needs a change can be done by double-clicking the blocks and changing the values appearing against some variables.

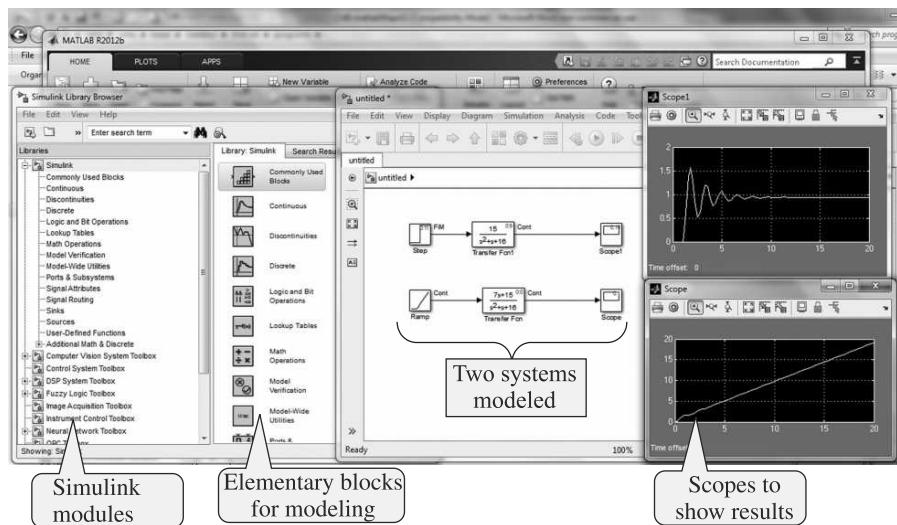


Fig. B.3 Simulink windows

B.4 ROBOTICS TOOLBOX

The Robotics Toolbox is a set of MATLAB functions for the analyses of robot kinematics, dynamics, vision, and other similar aspects (WR: Robot Toolbox). Once downloaded from the link <http://www.petercorke.com/robot>, one can run “start_rvc” by double-clicking on it. The window in MATLAB will be open and the toolbox is ready to use. Upon typing the following command:

```
>> rtbdemo
```

the window on the left of Fig. B.4 pops up, and different demos with their commands show up on the right. For example, a command with its answer appears as follows:

```
>> rotx(pi/2)
ans =
    1.0000      0      0
            0   0.0000 -1.0000
            0   1.0000   0.0000
```

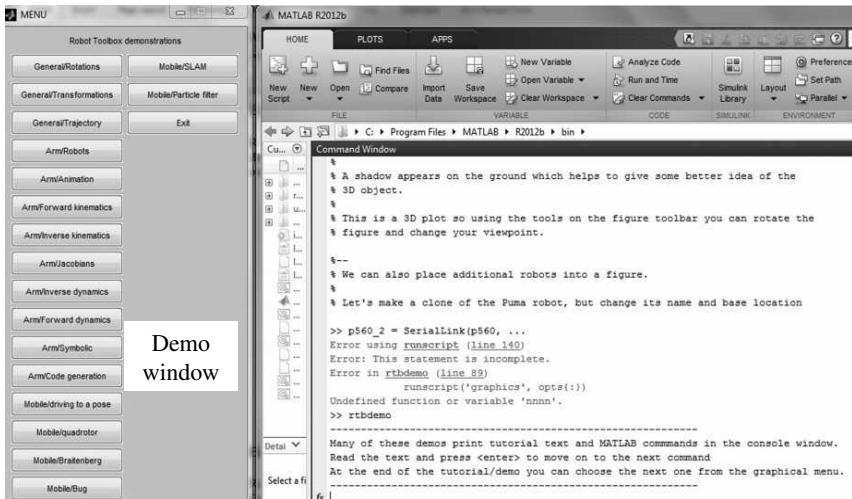


Fig. B.4 MATLAB window running Robotics Toolbox (rtbdemo)

For the forward kinematics of a PUMA robot, time and joint trajectory are generated as

```

>> mdl_puma560
>> t = [0:0.05:2]'; % generate a time vector
>> q = jtraj(qz, qr, t); % generate joint coordinate trajectory

```

Upon generation of the results, the following command

```
>> p560.plot(q);
```

will show an animation as shown in Fig. B.5. Similarly, there are many more commands which can be explored and can be used to validate many of the formulas and results given in this book by putting numerical values in them.

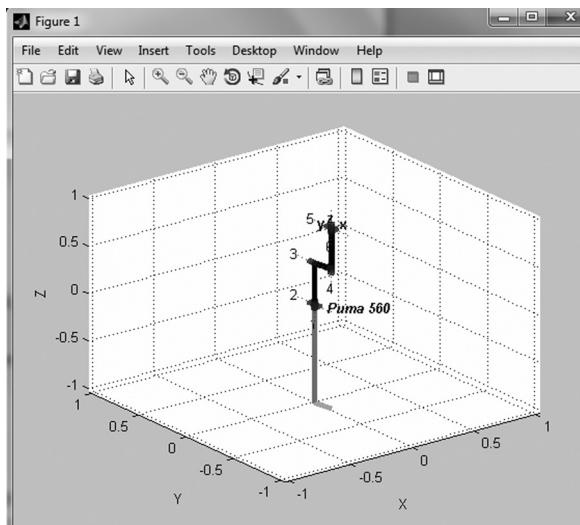


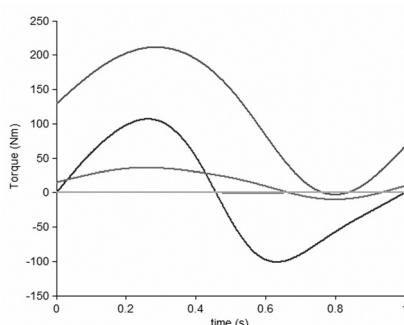
Fig. B.5 Animation screenshot generated by Robotics Toolbox

B.5 RECURSIVE DYNAMICS SIMULATOR (ReDySim)

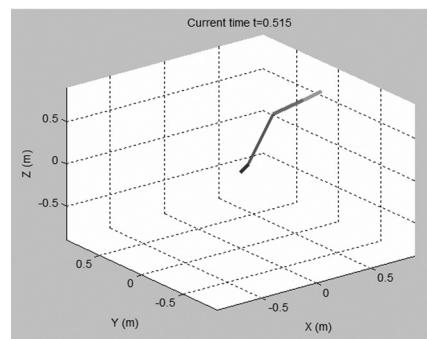
Recursive Dynamics Simulator (ReDySim) is a MATLAB-based recursive solver (Shah et al., 2013) for the dynamic analysis of robotic and multibody systems. ReDySim has the capability to incorporate any control algorithm and trajectory planner with utmost ease. This ability provides flexibility to the user/researcher in incorporating any customized algorithms. ReDySim showed considerable improvement over the commercial software and existing algorithms in terms of both computational time and numerical accuracy. ReDySim has three modules as shown in Table B.1, where the screenshot from simulation and animation results are shown in Fig. B.6.

Table B.1 Modules in ReDySim

1	Basic module: Fixed-base systems (Open- and closed-loop systems)	Three-link robot, gripper, KUKA robot, four-bar mechanism, biped, long chain, 3-RRR parallel robot and robotic leg
2	Specialized module: Floating-base Systems (a) Module for space robots (b) Module for legged robots	Three- and 7-link space robots, dual-arm space robot, biped, quadruped and hexapod legged robots
3	Symbolic module (a) Module for fixed-base systems (b) Module for floating-base systems	Fixed-base prismatic-revolute (PR)-robot, 2-link robot, KUKA robot, and floating-base 2-link space robot



(a) Joint torques (3 of them are close to zero)



(b) Animation screenshot

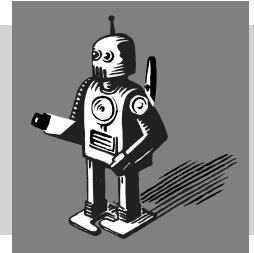
Fig. B.6 Plots and animation screenshot of KUKA KR-5 robot in ReDySim

SUMMARY

In this appendix, the use of MATLAB and its associated toolbox and software are explained. Very brief explanations of each of the items are outlined to get started with the software. This will help the readers use any or all of them to verify some of the examples and solve exercises given in this book.

C

Use of RoboAnalyzer¹



In this appendix, the steps to use the RoboAnalyzer software developed by the author and his students at IIT Delhi are explained. It is an improved version of the RIDIM (Recursive Inverse Dynamics for Industrial Manipulators) programs appeared in the first edition of this book in 2008. RoboAnalyzer (RA) has the visualization feature through 3-dimensional models of robots, including many standard robots like KUKA, ABB, Fanuc, and others. It can be used to learn DH parameters, kinematics, and dynamics of serial robots, and allows 3-dimensional (3D) animation and graph plots as outputs. In essence, learning the physics of robotics with joy is emphasized through RA, rather than only mathematics.

RoboAnalyzer can be installed on a computer with windows operating system by downloading from the website mentioned in the footnote.

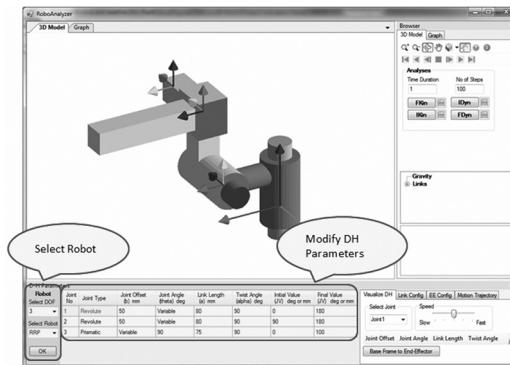


Fig. C.1 Robot-model selection and redefine the DH parameters

The following sections explain the main features of RoboAnalyzer.

C.1 VISUALIZE DENAVIT-HARTENBERG PARAMETERS

After selecting a robot and redefining its Denavit-Hartenberg (DH) parameters, as shown in Fig. C.1, users can visualize each DH parameter by selecting a joint and

¹ RoboAnalyzer (RA) software and the users' manual can be downloaded free from <http://www.roboanalyzer.com>. Also available from <http://www.mhhe.com/saha/ir2>. Several students since 2009 (Mr. C.G. Rajeevlochana and others) are sincerely acknowledged for their contributions to develop the RA software.

then selecting a DH parameter. Once it is done, the corresponding DH parameter is highlighted in the DH parameter input table and the corresponding coordinate frame moves in the 3D robot model.

Users can click on **Together** button and a coordinate frame moves covering all the four DH parameters corresponding to the selected joint. Users can also click on **Base Frame to End-Effector** button to see a coordinate frame moving from base frame to the end-effector frame covering all the DH parameters of the robot model.

C.2 FORWARD KINEMATICS

After selecting a robot and redefining DH parameters as explained in Section C.1, forward kinematics (FKin) is performed which updates the 3D model. Play button can be clicked to see the animation whereas clicking on **Graph** tab will allow the users to see various plots.

C.3 INVERSE KINEMATICS

To select a robot and view the solutions of its inverse kinematics, the following are the steps to be followed:

1. Click on **IKin** button. It shows a separate window (Fig. C.2).
2. Select a robot.
3. Enter input parameters.
4. Click on **IKin** button.
5. View the possible solutions.
6. Click on **Show** button. It shows the selected solution in the 3D model window. To see this, go back to the main window by minimizing IKin window.
7. Select any of the obtained solutions as initial and final solutions.
8. Click on **OK**. This step replaces the initial and final joint values in the DH parameters table (main window) by values selected in Step 7.
9. Click on **FKin** button to view animation, i.e., how the robot moves from one solution to another solution selected in Step 7.

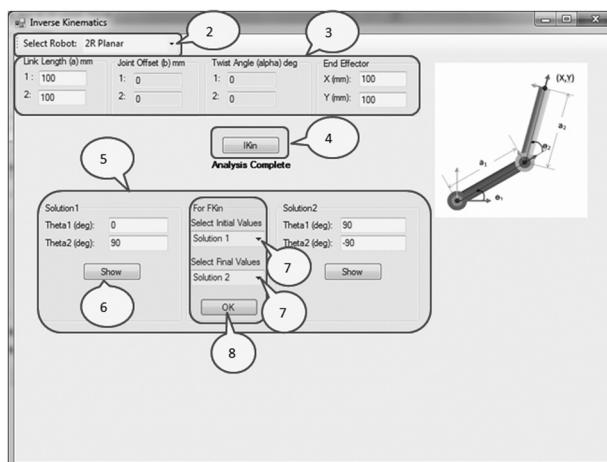


Fig. C.2 Inverse kinematics of a 2R planar robot manipulator

C.4 INVERSE DYNAMICS

Inverse Dynamics (IDyn) is a dynamics solver. Select a robot and redefine DH parameters as explained in Section C.1, if required, to solve for IDyn of the robot between the initial and final values of the joint variables. The following steps are followed to obtain the results:

1. Set the **initial** and **final** values of joint variables.
2. Set **Time Duration** and **Number of Steps**.
3. Set **Gravity** (all values should be in SI units, i.e., m/s^2).
4. Select a robot-link to enter its Center of Gravity (**CG**) location. It corresponds to a vector from the CG of the robot-link to the origin of the co-ordinate frame attached to that link measured in the coordinate frame attached to that link.
5. Select **Mass Properties** of a robot-link. Set **Mass** of each robot-link (values should be in SI units, i.e., kg) and set **Inertia** tensor of each robot-link with respect to the coordinate frame attached at the CG of the robot-link and the coordinate frame is parallel to the one attached to the robot-link (values should be in SI units, i.e., kgm^2). These values are to be entered manually and not calculated automatically from the shape of the robot-links.
6. Click on **FKin** button (required to populate the input joint trajectory).
7. Click on **Play** button to see the animation (only for visualization purpose, not necessary for IDyn).
8. Click on **IDyn** button to perform inverse dynamics.
9. Click on **Graph** tab to view the graph, as shown in Fig. C.3.

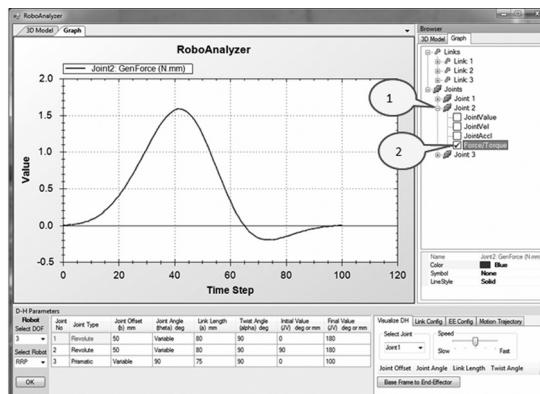


Fig. C.3 Graph plot of joint torque/force

C.5 FORWARD DYNAMICS AND SIMULATION

For Forward Dynamics (FDyn) and simulation, select a robot and redefine its DH parameters, if required. To solve for the FDyn of the robot for a given initial values of joint variables, the following steps are to be followed:

1. Set the **initial** value of joint variables.
- 2 to 5. Same as the steps in Section C.4.
6. Click on **FDyn** button to perform Forward Dynamics. The robot is simulated for free-fall due to the action of gravity. In future, joint torques/forces can be set as input.

7. Click on **Play** button to see the animation.
8. Click on **Graph** tab to view the graph.

C.6 JOINT TRAJECTORY

Select a robot, as explained in Section C.1. For given initial values of joint variables, motion planning of the selected robot can be performed by selecting a particular joint trajectory from the list available.

C.7 VIRTUAL ROBOT MODULE

The Virtual Robot Module inside RoboAnalyzer lets the user select an industrial robot model (CAD Model) and change the joint angles using a slider or using buttons. It can be used as a learning tool to teach joint-level jogging of robots. Several CAD models of industrial robots are stored. The following steps are to be followed:

1. Click on “More Robots” button appearing below “OK” of “Select Robots.”
2. A new window/form is shown. By default CAD model of a robot is displayed, as shown in Fig. C.4.

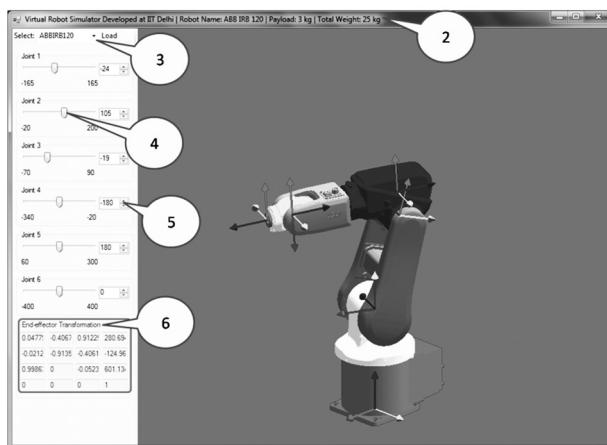


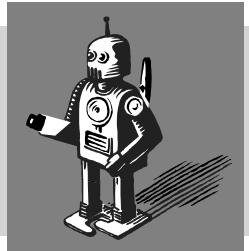
Fig. C.4 A robot in Virtual Robot Module

3. Select a robot from the drop-down and click on “Load”.
4. Use the slider on the left to change each joint angle. Note that all the joint angles have minimum and maximum values as per their specifications (joint limit).
5. Buttons can also be used to change the value of the joint angle.
6. The end-effector transformation is updated with every change in the joint angle(s).

SUMMARY

In this appendix, the use of RoboAnalyzer software is explained so that the reader can download and install it for the visualization of a robot in motion. Plots from different analyses can also be made. Several industrial models are also made available to the readers to help them understand their motion characteristics as well.

References



- Agarwal, A., Shah, S.V., Bandyopadhyay, S., and Saha, S.K., 2013, "Dynamics of serial kinematic chains with large number of degrees-of-freedom," Int. J. of Multibody System Dynamics, DOI 10.1007/s11044-013-9386-3, pp. 1–26.
- Agarwal, A., and Singh, S., 2004, *Design and Manufacturing of Robotic Gripper*, B. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Angels, J., 2003, *Fundamentals of Robotic Mechanical Systems*, Springer-Verlag, New York.
- Angeles, J., and Lee, S., 1988, "The formulation of dynamical equations of holonomic mechanical systems using a natural orthogonal complement," ASME J. Appl. Mech., V. 55, Mar., pp. 243–244.
- Angeles, J., Ma, O., and Rojas, A., 1989, "An algorithm for the inverse dynamics of n-axis general manipulator using Kane's formulation of dynamical equations," Computers and Mathematics with Applications, V. 17, N. 12, pp. 1545–1561.
- Asada, H., and Slotine, J.-J. E., 1986, *Robot Analysis and Control*, John Wiley and Sons, New York.
- Ascher, U.M., Pai, D.K., and Cloutier, B.P., 1997, "Forward dynamics, elimination methods, and formulation stiffness in robot simulation," Int. J. Rob. Res., V. 16, N. 6, pp. 749–758.
- Baghla, D., Anurag, A., Saha, S.K., Sharma, Prasenjit S., and Mehta, G.R., 1999, "Development of a hanging planar robotic arm (HaPRA)," Proc. of the 11th ISME Conf., IIT Delhi, Feb. 3–5, pp. 93–98.
- Bahuguna, J., 2012, *Kinematics of Industrial Robots*, M. Tech Report, Dept. of Mech. Eng., IIT Delhi, May.
- Balafoutis, C. A., and Patel, R. V., 1991, *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*, Kluwer Academic Publishers, Boston.
- Barma, D., 2001, *Design and Fabrication of a Planar Robot*, M. Tech Report, Dept. of Mech. Eng., IIT Delhi, Dec.
- Bhagat, R., Choudhury, Siddharth B., 2010, *Design and Development of a 6-DOF Parallel Manipulator*, B. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Bhagat, R., Choudhury, Siddharth B., and Saha, S.K., 2011, "Design and development of a 6-DOF parallel manipulator," Int. Conf. on Multi Body Dynamics, Vijayawada, India, Feb. 24–26, pp. 15–24.
- Bhangale, P.P., 2005, *Dynamics Based Modeling, Computational Complexity, and Architecture Selection for Robot Manipulators*, Ph. D thesis, submitted to the Dept. of Mech. Eng., IIT Delhi.
- Biagiotti, L., and Melchiorri, C., 2008, *Trajectory Planning for Automatic Machines and Robots*, Springer, New York.

- Bhandari, V.B., 2010, *Design of Machine Elements*, 3rd Ed., Tata McGraw-Hill, New Delhi.
- Brandl, H., Johanni, R., and Otter, M. 1988. "A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion of the Mass Matrix," *Theory of Robots*, Oxford: Pergamon Press, pp. 95–100.
- Canudas de Wit, C., Siciliano, B., Bastin, G. (Eds.), 1996, *Theory of Robot Control*, Springer Verlag, New York.
- Capek, Karel, 2001, *R.U.R.*, Dover Publications, New York.
- Chaudhary, H., and Saha, S.K., 2009, *Dynamics and Balancing of Multibody Systems*, Springer, Germany.
- Chaudhary, H., and Saha, S.K., 2006, "Optimal design of an Indian carpet weaving loom structure," *J. of Scientific and Industrial Research*, V. 66, pp. 410–415.
- Corke, P.I., 2011, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, Springer.
- Craig, J.J., 2009, *Introduction to Robotics: Mechanics and Control*, 3rd Ed., Pearson, New Delhi.
- Datta, K.B., 1999, *Matrix and Linear Algebra*, Prentice-Hall of India, New Delhi.
- de Silva, C.W., 2004, *Mechatronics—An Integrated Approach*, CRC Press, New York.
- Deb, S.R., and Deb, S., 2010, *Robotics Technology and Flexible Automation*, 2nd Ed., Tata McGraw-Hill, New Delhi.
- Denavit, J., and Hartenberg, R.S., 1955, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. ASME J. Appl. Mech.*, V. 23, pp. 215–221.
- Doebelin, E.O., 1990, *Measurement Systems: Application and Design*, 4th Ed., McGraw-Hill, Singapore.
- Engelberger, J.F., 1980, *Robotics in Practice*, AMACOM (American Management Association), New York.
- Featherstone, R., 1983, "The Calculation of Robotic Dynamics Using Articulated Body Inertias," *Int. J. of Robotics Research*, V. 2, pp. 13–30.
- Fuller, J.L., 1999, *Robotics: Introduction, Programming, and Project*, Prentice Hall, New Jersey.
- Ghosh, A., and Mallik, A.K., 1998, *Theory of Mechanisms and Machines*, 3rd Ed., Affiliated East West Press (Pvt.) Ltd., New Delhi.
- Ghosal, A., 2006, *Robotics: Fundamental Concepts and Analysis*, Oxford University Press, New Delhi.
- Gillespie, T.D., 1992, *Fundamentals of Vehicle Dynamics*, Society of Automotive Engineers Inc., Warrendale, USA.
- Gopal, M., 1997, *Control Systems—Principles and Design*, Tata McGraw-Hill, New Delhi.
- Gopal, M., 2003, *Digital Control and State Variable Methods*, Tata McGraw-Hill, New Delhi, 2nd Edition.
- Gorinevsky, D.M., Formalsky, A.M., and Schneider, A.Y., 1997, *Force Control of Robotics Systems*, CRC Press, New York.
- Gruver, W.A., Soroka, B.I., Craig, J.J., and Turner, T.L., 1984, "Industrial robot programming languages: A comparative evaluation," *IEEE Trans. Systems, Man, and Cybernetics*, V. SMC-14, N.4, July/August, pp. 565–570.
- Groover, M.P., Weiss, M., Nagel, R.N., Odrey, N.G., and Dutta, A., 2012, *Industrial Robotics: Technology, Programming, and Applications*, Tata McGraw-Hill, Special Indian Edition, New Delhi.

- Hollerbach, J.M., 1980, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," IEEE Trans. on Systems, Man, and Cybernetics, V. SMC-10, pp. 730–736.
- Haralick, R.M., and Shapiro, L.G., 1992, *Computer and Robot Vision*, Vol. I, Addison-Wesley, New York.
- Haralick, R.M., and Shapiro, L.G., 1993, *Computer and Robot Vision*, Vol. II, Addison-Wesley, New York.
- Huston, H., and Passerello, C.E., 1974, "On constraint equation—A new approach," ASME J. Appl. Mech., V. 41, pp. 1130–1131.
- Kane, T.R., and Levinson, D.A., 1983, "The use of Kane's dynamical equations for robotics," Int. J. Rob. Res., V. 2, N. 3, pp. 3–21.
- Khalil, W., and Kleinfinger, J.F., 1986, "A new geometric notation for open and closed-loop robots," IEEE Int. Conf. on Robotics and Automation, V. 3, pp. 1174–1179.
- Kumar, Nihit R., and Kishan, S. Kamlesh, 2012, *Development and Validation of an Interface for Actuation System Design for Wheeled Mobile Robots*, B. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Kuo, B.C., 1991, *Automatic Control Systems*, 6th Ed., Englewood Cliffs, Prentice-Hall, New Jersey
- Koivo, A.J., 1989, *Fundamental for Control of Robotic Manipulators*, John Wiley & Sons, New York.
- Koren, Y., 1987, *Robotics for Engineers*, McGraw Hill, New Delhi.
- Koteswara Rao, A.B., Saha, S.K., and Rao, P.V.M., 2006, "Dynamics modeling of hexaslides using the decoupled natural orthogonal complement matrices," Int. J. of Multibody System Dynamics, V. 15, pp. 159–180.
- LaSalle, J., and Lefschetz, S., 1961, *Stability by Liapunov's Direct Method with Applications*, Academic Press, New York.
- Luh, J.Y.S., Walker, M.W., and Paul, R.P.C., 1980, "On-line computational scheme for mechanical manipulators," ASME J. Dynamic Systems, Measurement, and Control, V. 102, pp. 69–76.
- Marothiya, P., and Saha, S.K., 2003, "Robot inverse kinematics and dynamics algorithms for windows," Recent Trends in Manufacturing (Proc. of the Conference on Advances and Recent Trends in Manufacturing, Kalyani Govt. Eng. College, WB, India), Elite Publishing House, New Delhi, pp. 229–237.
- Meirovitch, L., 1970, *Methods of Analytical Dynamics*, McGraw-Hill, New York.
- Mittal, G., Sinha, J., and Singh, K.V., 2009, *Modular Design of Robotic Systems*, B. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Mohan, A., and Saha, S.K., 2007a, "A computationally efficient and numerically stable simulation algorithm for serial robots with flexible links," CD-Proc. of the ECCOMAS Thematic Conf. (Multibody Dynamics 2007), Milano, Italy, June 25–38, pp. 1–20.
- Mohan, A., and Saha, S.K., 2007b, "A recursive, numerically stable, and efficient simulation algorithm for serial robots," Int. J. of Multibody System Dynamics, V. 17, N. 4, May, pp. 291–319.
- Nakra, B.C., and Chaudhry, K.K., 2009, *Instrumentation, Measurement and Analysis*, 3rd Ed., Tata McGraw-Hill, New Delhi.
- Niku, S.B., 2001, *Introduction to Robotics: Analysis, Systems, Applications*, Pearson Education Asia, New Delhi.

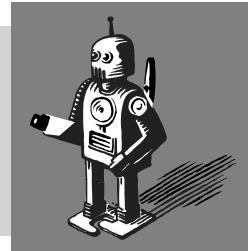
- Norton, R.L., 2001, *Machine Design—An Integrated Approach*, Prentice Hall, New Delhi.
- Ogata, K., 1987, *Modern Control Engineering*, Prentice-Hall of India, New Delhi.
- Pan, Z, Polden, J., Larkin, N., Duin, S.V., and Norrish, J., 2012, “Recent progress on programming methods for industrial robots,” *Robotics and Computer-Integrated Manufacturing*, V. 28, pp. 87–94.
- Pratap, R., 2010, *Getting Started with MATLAB*, Oxford University Press, New York.
- Raghavan, M., and Roth, B., 1993, “Inverse kinematics of the 6R manipulator and related linkages,” *ASME J. of Mech. Des.*, V. 115, pp. 502–508.
- Rajeevlochana, C.G., Saha, S.K., and Shivesh Kumar, 2012, “Automatic extraction of DH parameters of serial manipulators using Line Geometry,” CD-Proc. of the 2nd Joint International Conference on Multibody System Dynamics, May 29-June 1, Stuttgart, Germany
- Ramvath, P., 2004, *Design and Analysis of a Parallel Drive Robot*, M. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Ramavath, P., Barma, D., Saha, S.K., Seth, B., and Jaitly, D., 2005, “Development of a direct-drive based three-DOF robot,” Nat. Conf. on Industrial Problems on Machines and Mechanisms, IIT Kharagpur, Feb. 24-25, pp. 67–74.
- Rathore, A., and Jain, R., 2009, *Design and Fabrication of Four-wheel Bases for Mobile Robots*, SURA Project Report, IIT Delhi, July.
- Rivin, Eugene I., 1988, *Mechanical Design of Robots*, McGraw-Hill, New York.
- Saha, S.K., 1996, “Inverse dynamics algorithm for space robots,” *Trans. of the ASME, J. of Dynamic Systems, Measurement and Control*, V. 118, Sept., pp. 625–629.
- Saha, S.K., 1997, “A decomposition of the manipulator inertia matrix,” *IEEE Trans. on Robotics and Automation*, V. 13, N. 2, Apr., pp. 301–304.
- Saha, S.K., 1999, “Dynamic modelling of serial multi-body systems using the decoupled natural orthogonal complement matrices,” *ASME J. of Applied Mechanics*, V. 66, Dec. pp. 986–996.
- Saha, S.K., 2003, “Dynamic simulation of serial manipulators using the UDU^T decomposition of the inertia matrix,” *Int. J. of Multibody System Dynamics*, V. 9, N. 1, Feb., pp. 63–85.
- Saha, S.K., and Schiehlen, W.O., 2001, “Recursive kinematics and dynamics for closed loop multibody systems,” *Int. J. of Mechanics of Structures and Machines*, V. 29, N. 2, Aug., pp. 143–175.
- Saha, S.K., and Angeles, J., 1991, “Dynamics of nonholonomic mechanical systems using a natural orthogonal complement,” *ASME J. Appl. Mech.*, V. 58, Mar., pp. 238–243.
- Sethi, D.S., Kumar, K., and Kushwaha, R., 2011, *Development and Verification of Interface for Modular Design of Mobile Robots*, B. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Shah, S., Saha, S.K., and Dutt, J.K., 2012, “Denavit-Hartenberg (DH) Parametrization of Euler Angles,” *ASME J. of Computational and Nonlinear Dynamics*, V. 7, Apr., pp. 021006-1 to 10.
- Shah, S., Saha, S.K., and Dutt, J.K., 2013, *Dynamics of Tree-type Robotic Systems*, Springer, London.
- Shampine, L., 1994, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, New York.

- Sharma, P. and Kumar, A., 1997, *Design of an Industrial Robot*, B. Tech Project Report, Dept. of Mech. Eng., IIT Delhi, May.
- Shell, R.L., and Hall, E.L., 2000, *Handbook of Industrial Automation*, Marcel Dekker, New York.
- Shigley, J.E., Mischke, C.R., Budynas, R.G., and Nisbett, N.G., 2008, *Mechanical Engineering Design*, 8th Ed., Tata McGraw-Hill, New Delhi.
- Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G., 2011, *Robotics: Modeling, Planning and Control*, Springer (India), New Delhi.
- Slotine, J.-J.E., and Li, W., 1991, *Applied Nonlinear Control*, Prentice-Hall, New Jersey.
- Spong, M.W., and Vidyasagar, M., 2004, *Robot Dynamics and Control*, John Wiley and Sons, Canada.
- Spong, M.W., Hutchison, S., and Vidyasagar, M., 2006, *Robot Modeling and Control*, Wiley India, New Delhi.
- Stejskal, V. and Valasek, M., 1996, *Kinematics and Dynamics of Machinery*, Marcel Dekker, New York.
- Strang, G., 1980, *Linear Algebra and Its Applications*, Harcourt Brace Jovanovich, Publishers, Orlando, Florida.
- Todd, T.J., 1985, *Walking Machines*, Kogan Page, London.
- Tsai, L., 1999, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*, John Wiley & Sons, New York.
- Venugopal, S., and Shabeer, S., 1996, *Simulation and Animation for Robot Design*, Dept. of Mech. Eng., IIT Madras, May.
- Venugopal, S., and Saha, S.K., 1997, "Interactive design for industrial robots," Proc. of the AIMTDR Conf., Jan. 9–11, Warangal, pp. 201–205.
- Walker, M.W., and Orin, D.E., 1982, "Efficient dynamic computer simulation of robotic mechanisms," ASME J. Dyn. Sys., Meas., and Cont., V. 104, Sept., pp. 205–211.
- Wylie, C.R., 1975, *Advanced Engineering Mathematics*, McGraw-Hill, New Delhi.
- Yoshikawa, T., 1990, *Foundation of Robotics*, The MIT Press, Massachusetts.

Web References [WR]

- Function Bay: <http://eng.functionbay.co.kr/>
- Hydraulic: For cylinder design, www.driveworks.co.uk [Accessed on May 08, 2013]
- Hydraulic-video: http://www.youtube.com/watch?v=KW3u-_6uEKg [Accessed on May 08, 2013]
- RCC, http://www.ati-ia.com/products/compliance/Compensator_main.aspx [Accessed: Sept. 03, 2013]
- Matlab, "MATLAB: The Language of Technical Computing," <http://www.mathworks.in/products/matlab/> [Accessed on April 03, 2014]
- Maxon Motors and Gears: <http://www.maxonmotor.in/maxon/view/content/index> [Accessed on Aug. 04, 2013]
- PI: Physik Instrumente, http://www.pi-usa.us/products/PDF_Data/M840_Hexapod_Platform_Parallel_Positioner.pdf [Accessed on Dec. 14, 2013]
- Robot Toolbox, *Robotics Toolbox for MATLAB*, Release 9, <http://www.petercorke.com/robot>
- Stepper (includes video): www.pcbheaven.com/wikipages/How_Stepper_Motors_Work/ [Accessed on May 07, 2013]

Index



A

ABB 42
ac Motors 58
ac Servomotor 61
Acceleration analysis 206
Acceleration sensors 84
Accuracy 112, 225
Actuation systems 32
Actuator 24, 41, 62, 225
Actuator dynamics 334
Adaptive control 370, 380
ADC 27, 109, 339
AGV 4
Algebraic solution 175
Amplifier 29, 106
Analytical expressions 289
Angular acceleration 263
Angular momenta 257
Angular momentum 258
Angular velocity 245
Animation 483
Anthropomorphic 168
Applications 8, 34
Architecture 116
Articulated 31
Articulated arm 181, 183, 184
Articulated body inertia 310
Artificial constraints 390
Assembling 39
 atan2 181, 182, 184m 475
Automatic robot 457

B

Backlash 62
Backward computations 265
Bandwidth 112
Belt 25
Bridge 110

Brushless 53
Bus 429

C

CAD 492
Camera 92
Capacitive 87, 88
Cartesian 29
Cartesian control 381
Cartesian space 416
Cayley–Hamilton 353
Center of mass 236
Chain 25
Characteristic polynomial 479
Christoffel symbol 250, 255
Classification 21, 29
CNC 2
Complex roots 323
Composite body 289
Computation complexity 289
Computational complexities 300, 311
Computational speed 433
Computed torque 367
Condition number 229
Configuration 122, 224
Continuous path 424
Control 33, 318
Control law partitioning 366
Control subsystem 23, 27
Controllable 353
Coordinate system 29, 447
Coriolis force 249
Critically damped 325, 341
Cross product 477
Cross-product matrix 477
Cubic 398
Cycloidal 412
Cylindrical 30, 118

D

DAC 27, 109, 339
 Damping ratio 328
 dc Motor 50, 333
 Deburring 38
 Decision module 427
 Defence 11
 Degree of freedom 120
 Demodulators 108
 Denavit and Hartenberg (DH) Parameters
 146
 Denavit-Hartenberg 489
 DeNOC 202, 283
 Determinant 478
 Dexterity 229
 DH frames 148
 Direct-drive 466
 Direction cosine 124
 Disturbance rejection 340, 379
 DOF 120
 Domestic 11
 Dot product 476
 Drivers 54
 Duty cycle 56
 Dynamic algorithms 270
 Dynamic modeling 284
 Dynamic systems 320
 Dynamics 235
 Dynamics board 430

E

Economics 13
 Edge detection 99
 Effective damping 335
 Effective inertia 335
 Eigenvalues 479
 Eigenvector 479
 Electric 42
 Encoder 77
 End-effector 24
 Entertainment 12
 Equations of motion 247, 259, 284
 Equilibrium 209
 Euler angles 124, 131, 159, 195
 Euler parameters 124, 138
 Euler–Lagrange 243, 270, 283, 284

F

FDyn 491
 Feature extraction 100
 Feedback control 338
 Feedback linearization 369, 383

Feedforward control 377

FFT 110
 Filters 107
 Fischertechnik 473
 Fixed-axes 124, 128
 FKin 490
 Force 365
 Force board 430
 Force control 385, 388
 Force ellipsoids 230
 Force sensors 85
 Forces and moments balance 210
 Forward computations 263
 Forward dynamics 275, 491
 Forward kinematics 162, 490
 Four-bar 121
 Free-fall simulation 312

G

Gain scheduling 369
 Gantry robot 32
 Gauge factor 85
 Gear 25, 26
 Generalized coordinates 243, 244
 Generalized force 293
 Generalized inertia matrix 247, 289
 Geometric solution 177
 GIM 289, 290, 370
 Gripper 41, 69, 465
 Grubler Kutzbach 121

H

Hall-effect sensor 83
 HaPRA 464
 Hardware 426
 Hardware architecture 429
 Harmonic drive 47
 Helical 117
 Higher pair joints 119
 High-level vision 103
 Histogram 98
 Homogeneous transformation 141
 HTMs 163
 Hybrid 48
 Hybrid control 394
 Hydraulic 42, 63
 Hysteresis 113

I

IDyn 491
 IKin 490
 Image acquisition 97

- Impedance control 386, 387
 Induction motor 59
 Inductive 87
 Industrial robot 2
 Inertia 55
 Inertia properties 236
 Inertia tensor 237, 245
 Interfacing 114
 Intermediate-level vision 103
 Inverse 478
 Inverse dynamics 261, 270, 374, 491
 Inverse dynamics control 366
 Inverse kinematics 162, 177, 179, 180, 490
 Inverse position analyses 175, 205
 Inverse-Jacobian control 382
 ISO 2
 Isotropic 231
 ISRO 10
- J**
 Jacobian 193, 205, 206, 207, 219, 227, 479
 Joint angle 148, 156
 Joint controllers 357, 431
 Joint offset 148, 156
 Joint space 398
 Joint torques 261
 Joint trajectory 492
 Joint-motion propagation 203
 Joint-motion propagation vector 286
 Joints 116
- K**
 Kinematic chain 119
 Kinematic constraints 286
 kinematic pairs 116
 Kinematics 162
 Kinematics board 430
 Kinetic energy 244
 Kinetostatic 223
 Kronecker delta 434
 KUKA 19, 307
- L**
 Lagrangian 243
 Laplace transforms 328
 Laws of robotics 1
 Lead-through 444
 LEGO 473
 Lighting 95
 Limit switch 87
 Linear 318
 Linear independence 478
- Linear momentum 257
 Linear velocity 263
 Linearity 112
 Linearized control 371
 Link length 148, 156
 Links 116
 Lower 116
 Lower pair joint 116
 Low-level vision 103
 LSPB 410
 LTI 365
 LU decomposition 206
 LVDT 81
 Lyapunov equation 345
 Lyapunov function 370
 Lyapunov Stability 342
- M**
 Machining 37,
 Manipulability 114, 229
 Manipulator 23, 119
 Manipulator design 222
 Manual robot 457
 Masking 98
 Material handling 34
 MATLAB 252, 276, 331, 480
 MATLAB program 403
 Matrices 481
 Matrix 478
 Matrix of convective inertia 288, 291
 Matrix operations 434
 Mechanism 25, 119
 Medical robots 8
 MIMO 320
 Mining 9
 Mobile robots 468
 Mobility 224
 Modeling module 427
 Modified DH 156
 Modulators 108
 Moment 262
 Moment of inertia 236, 237
 Motion 33, 397
 Motion subsystem 22
 Motor 41
 Moving block 321
 MuPAD 252, 276, 295, 480, 484
- N**
 NASA 2
 Natural constraints 389
 Natural frequency 328

Newton–Euler 283, 284
 Newton–Euler formulation 257
 NOC 204, 283
 Noncommutative 139
 Nonlinear 365

O

Observable 353
 ODE45 276
 ODEs 483
 Offline programming 449
 Online programming 443
 Ordinary differential equations 483
 Orientation 120, 124, 423
 Orientation problem 189
 Overdamped 321, 341

P

Painting 37
 Parallel robot 6, 462
 Parallel-axis theorem 239
 Path primitives 419
 Payload 223
 PD 358, 372
 Percentage overshoot 341
 Permanent-magnet 47
 Perpendicular-axis theorem 239
 PID 346, 359
 Piezoelectric 86
 Planar 118
 Planning 397
 Plots 483
 Pneumatic 42, 66
 Point-to-point 424
 Polar 30
 Pole 324, 325, 328
 Poly-phase 59
 Population 13
 Pose 120, 122
 Position 123, 422
 Position analysis 163
 Position problem 186
 Position sensors 77
 Potential energies 243
 Potential energy 247
 Potentiometer 80
 Power 288
 Precision 113
 Principal moments of inertia 239
 Prismatic 117
 Prismatic joint 146, 163, 198
 Programming 33

Projects 456
 Proximity 87
 PUMA 23, 171
 PUMA robot 170

Q

Quintic 401

R

Range 112
 Recognition subsystem 23
 Recursive calculations 212
 Recursive forward dynamics 309
 Recursive inverse dynamics 299
 Recursive Newton–Euler algorithm 261
 Recursive robot dynamics 283
 ReDySim 480, 488
 Regressor 257
 Reliability 114
 Repeatability 113, 225
 Resolution 113
 Resolved motion 381
 Resolver 82
 Resonant frequency 351
 Revolute 31, 117
 Revolute joint 119, 198
 Riccati equation 354
 RIDIM 299, 489
 RoboAnalyzer 155, 165, 280, 299, 302, 408, 489
 RoboCNC 472
 ROBOCONs 456
 RoboMuse 460
 Robot languages 436
 Robot programming 426, 442
 Robot usage 7
 Robotic joint 333
 Robotics toolbox 486
 Robots 2
 Robust 370
 Robust control 380
 Roots 321
 ROV 10
 RUR 1

S

Safety 16
 Scalar product 476
 SCARA 31, 167
 Security 11
 Selection 68
 Selection of grippers 74

- Semiconductor 89
 Sensitivity 112
 Sensor classification 77
 Sensor selection 111
 Sensors 27, 76, 225
 Sensory module 427
 Servo board 430
 Servomotors 54
 Signal conditioning 106
 Simulation 269, 275, 309, 311, 491
 Simulink 328, 486
 Single-phase 59
 Singularity 182, 227
 SiRoD 467
 SISO 320
 SLI 227
 Slip 60
 Software 442, 480
 Software Projects 466
 Space 9
 SPD 344
 Speed 225
 Spherical 30, 118
 Spherical arm 167
 Spline 413
 Stability 338, 343, 370
 Stanford Arm 173, 305
 State-feedback 352
 State-space 275, 329
 Statics and manipulator design 209
 Stepper 43
 Strain-gauge 85
 Strength 231
 Structural 223
 Subsystems 22
 Synchronous 59
 Synchronous motor 61
 Synchros 82
 System board 430
- T**
 Tachometer 83
 Teach pendant 444
 Three-phase 60
 Threshold 113
 Thresholding 98
 Tool calibration 447
 Torque 47, 213, 265
 Touch 77
- Tracking 340
 Transducer 76
 Transfer function 327
 Transformations 116
 Transmission 24
 Transpose-Jacobian control 382
 Trapezoidal 410
 Trapezoidal velocity 411
 Twist angle 148, 156
 Twist propagation 203
 Twist propagation matrix 286
- U**
 UDUT 310
 Underdamped 323, 341
 Underwater 10
 Unimation 2
 Unit vector 476
 Universal joint 119
- V**
 Variable-reluctance 46
 VCI 293, 370
 Vector of convective inertia 288
 Vectors 475
 Velocity analysis 193
 Velocity sensors 83
 Virtual displacement 219
 Virtual robot module 492
 Virtual work 219, 243
 Vision 76, 90
 Vision board 430
 Vision systems 90
- W**
 Walking robot 5, 6
 Walk-through 445
 Welding 35
 WMR 4
 Work envelope 19
 Workspace 19, 226
 Wrist 168, 184
 Wrist-partitioned 186
- Y**
 YouBot 472
- Z**
 Zeros 328