Numerical Simulation of Mechanisms and Manipulator Robots with Flexible Links and Flexible Joints
Numerical Simulation of Mechanisms and Manipulator Robots with Flexible Links and Flexible Joints

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op dinsdag 10 januari 1995 te 13.30 uur

door

Diguang GONG

Master of Science in Engineering
geboren te Taojiang County, Hunan Province, China
Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. J.F. Besseling

Samenstelling promotiecommissie:
Rector Magnificus (Technische Universiteit Delft)
Prof. dr. ir. J.F. Besseling (Promotor, Technische Universiteit Delft)
Prof. dr. ir. P. Meijers (Technische Universiteit Delft)
Prof. dr. ir. K. van der Werff (Technische Universiteit Delft)
Prof. ir. H.A. Crone (Technische Universiteit Delft)
Prof. dr. ir. H.B. Pacejka (Technische Universiteit Delft)
Dr. ir. J.B. Jonker (Technische Universiteit Twente)
Dr. ir. J.P. Meijaard (Technische Universiteit Delft)

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Gong, Diguang

Numerical Simulation of Mechanisms and Manipulator Robots with Flexible Links and Flexible Joints /

ISBN 90-370-0117-3

Subject headings: simulation / CAD / flexible robot / finite element / mechanical systems / dynamics
Copyright © 1995 by Diguang Gong
All rights reserved. No part of this thesis may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, and information storage and retrieval system, or otherwise, without written permission from the copyright owner.

PRINTED IN THE NETHERLANDS
To my wife Xiaode and my son Jiachen

To my parents
# Contents

Summary ................................................................. 1

1. Introduction ......................................................... 3

2. Software Engineering ............................................... 7
   2.1. Introduction .................................................. 7
   2.2. Integrated program network and data base .................. 8
   2.3. X Window system and interface of X SPACAR ............... 12
      2.3.1. The X Window ......................................... 12
      2.3.2. Design principles of the interface .................... 18
      2.3.3. The main interface ..................................... 20
   2.4. Object-oriented analysis, design and programming ...... 21
      2.4.1. Introduction ............................................ 21
      2.4.2. Abstraction ............................................. 22
      2.4.3. Encapsulation .......................................... 23
      2.4.4. Modularity .............................................. 23
      2.4.5. Hierarchy ............................................... 24
      2.4.6. Object-oriented finite element programming for the SPACAR system 25

3. Dynamics of Electrical Machines .................................. 31
   3.1. Introduction .................................................. 31
   3.2. Dynamics of generalized electrical machines .............. 31
   3.3. Equations of the induction motor .......................... 36
      3.3.1. General equations ..................................... 36
      3.3.2. Steady state equations ............................... 39
      3.3.3. Equations for small oscillations .................... 43
3.4. Dynamic equations of alternators .................................................. 47
  3.4.1. General equations ................................................................. 47
  3.4.2. Steady state equations ......................................................... 49
  3.4.3. Equations for small oscillations .......................................... 50
3.5. Dynamics of d.c. motors ............................................................. 51
  3.5.1. Separately excited d.c. motors ............................................ 51
  3.5.2. Series-excited d.c. machines .............................................. 52
3.6. Controlled electric-mechanical equations ...................................... 53

4. Rigid and Flexible Links and Flexible Joints .................................... 55
  4.1. Introduction ............................................................................. 55
  4.2. Finite element representation for mechanisms and robots ............. 56
  4.3. Flexible links .......................................................................... 60
  4.4. Flexible joints .......................................................................... 66
    4.4.1. The spatial hinge element .................................................. 66
    4.4.2. The flexible joint and its finite element representation .......... 67
  4.5. A rigid beam element ................................................................ 69
    4.5.1. The rigid beam element ..................................................... 69
    4.5.2. Examples ......................................................................... 71

5. Dynamic Analysis of Mechanical Systems ....................................... 73
  5.1. Introduction ............................................................................. 73
  5.2. Numerical method for equation reduction .................................... 74
    5.2.1. Equation reduction method ................................................ 75
    5.2.2. Computational implementation .......................................... 79
  5.3. Linearization of the equations of motion ..................................... 79
  5.4. Stationary solution of mechanical systems .................................. 82
  5.5. Periodic solutions of mechanical systems ................................... 84
  5.6. Numerical examples of dynamic systems .................................... 85
    5.6.1. The four bar mechanism ..................................................... 85
    5.6.2. A governor mechanism ....................................................... 89
    5.6.3. Spatial slide crank mechanism ........................................... 93
    5.6.4. Spatial slide crank mechanism with flexible joint ................. 97

6. Simulation of Manipulator Robots .................................................. 101
  6.1. Introduction ............................................................................. 101
6.2. Linearization for control systems ........................................ 104
   6.2.1. Variational model around a nominal trajectory ............... 104
   6.2.2. Feedback linearization ........................................... 106
6.3. Inverse kinematic solutions ............................................ 113
6.4. Primary and secondary controller design for manipulators ........ 115
   6.4.1. Primary controller design ..................................... 115
   6.4.2. Secondary controller design .................................. 117
6.5. Simulation examples .................................................... 119
   6.5.1. Two link rigid-flexible manipulator .......................... 119
   6.5.2. Flexible joint manipulator .................................... 121
   6.5.3. Spatial manipulator with a flexible joint and flexible links .... 126

Bibliography ............................................................................ 133

Samenvatting ........................................................................... 141
Acknowledgements ................................................................... 143
Curriculum Vitae ....................................................................... 144
Summary

This thesis deals with numerical simulation of mechanical systems like mechanisms and manipulator robots. The mechanical system may be considered in a two or three dimensional space, may have electrical machines, and may contain flexible links and flexible joints. Methods and algorithms to perform static analysis, kinematic analysis, dynamic analysis, linearization of nonlinear systems, periodic solution of periodically forced systems and control simulation of robots have been applied. A finite element method is used to get the mathematical model for a mechanical system. This has the advantage of systematically representing various kinds of components and connections normally used in mechanical systems. Friendly interfaces for pre- and post-processing have been designed by using X Window on Sun workstations.

An integrated program network and data base has been applied in the software system SPACAR. The FORTRAN language is used in most of the calculations for its calculating efficiency and the C language and X Window are used for their rich and complex environment to the programmers and users of application software. The basic concept, the design structure and the principles by which the software system have been designed are introduced. As the object-oriented approach is the latest trend in programming practice, the object-oriented programming for the SPACAR system has also been presented.

The general electrical machine theory has been employed to perform the dynamic analysis of electrical machines. Using this theory, we can analyse all kinds of electrical machines which can be transformed to general electrical machines. Combining this theory with the finite element representation of mechanical systems results in a powerful computer simulation tool for mechanical-electrical systems. Simplified models have been derived for steady state analysis of electrical machines. Two linearization models have been derived, one around the steady state equations and the other for the general dynamic equations for stability analysis. As the d.c. motors are often used in control systems, the dynamics and inverse dynamics of different d.c. motors have been introduced.
The model of a mechanical system, with or without flexible links and flexible joints, is derived by using the finite element method. Mechanical systems are composed of components. These different components can be represented by different finite elements. The main principles to define a finite element has been introduced so that one can define an element according to his own requirements. Typical elements, BEAM and HINGE elements, have been described in detail. Flexibility of links and joints in the mechanical systems can be easily taken into account or ignored.

The two models of flexible joints established in this thesis can be used to represent many different kinds of flexible joints. Model I can be considered as direct representation of the mechanical system. Model II has some advantages when it is used in the simulation of robots with flexible joints.

The introduction of the rigid beam elements has two advantages. On one hand, it can improve calculating efficiency. An appreciable reduction of calculating time has been found when these rigid beam elements are used for rigid body systems in both kinematic and dynamic analyses of mechanical systems. On the other hand, it is closer to the physical model of a rigid body because only one set of orientation coordinates (one rotation node) is used in the model.

The coordinate reduction method which has been used in the finite element approach has been generalized to reduce differential-algebraic equations to differential equations. The implementation of the method into the SPACAR system makes it possible to use constraint conditions by the user himself. An automatic linearization of nonlinear system has been introduced and the linearized results can be used for control design, for stability analysis, for periodic solution and for calculation of the main natural frequencies. The shooting method is used to find the periodic solution of a periodically forced mechanical system. Examples have shown that the use of the periodic solution as the initial conditions for dynamic simulation for periodic forced systems is very important.

For the simulation of robots with flexible links and flexible joints, two linearization methods have been studied: the variational linearization method and the feedback linearization method. Numerical examples for simulation of robots with flexible links and flexible joints have been presented to show the elegance and powerfulness of the introduced methods and the designed software system.
Chapter 1

Introduction

To design and manufacture marketable products of high quality at lower cost is one of the major goals of the engineering profession. Today's industries are utilizing computers in every phase of the design, management, manufacture, and storage of their products. The process of design and manufacture, beginning with an idea and ending with a final product, is a closed-loop process. Almost every link in the loop can benefit from the power of digital computers. As real product experiments are normally expensive, simulation, especially dynamic simulation of the products is an important way to show the properties of the products.

As computers are widely used in engineering, numerical simulation plays an essential role in kinematic analysis, dynamic analysis, and design of mechanical systems like mechanisms and manipulator robots. The advantages of using computer simulation are obvious. It is much faster and easier compared with analytical methods. For some complex systems, for example, a complex nonlinear system, for which analytical analysis may be too difficult to perform, if not impossible, numerical simulation may be the only choice. It is much cheaper and faster than performing an experiment. An experiment may take some weeks or months, and it costs much money, while a numerical simulation may take just a few hours or days and certainly will cost less money. Though it may not be possible to replace experimental and analytical methods for mechanisms and robots design completely by numerical simulation, it can provide otherwise unattainable insight and thus play an essential role in industrial design.

A great deal of research has been done on computer simulation of complex multibody systems in the last two decades. This research work has been summarized in the books written by Nikravesh (1988), Roberson and Schwertassek (1988), Haug (1989), Shabana (1989), Huston (1990) and Garcia de Jalon and Bayo (1993). Several general-purpose computer programs for analysis and simulation of multibody systems have been developed [Schiehlen, 1990].

Due to the emphasis placed by industry on realistic modelling of mechanical systems,
systems like light-weight spatial structures and manipulators or high-speed machinery, research in the dynamics of flexible mechanism and robots has gained tremendous attention from many researchers. A number of models has been introduced to establish the equations of motion for mechanisms and robots. In the earlier stage, the models were based on the assumption that small deformations of the bodies do not significantly affect the nominal rigid body motion, so that the rigid motion can be uncoupled from the elastic motion [Winfrey, 1972], [Salder and Sandor, 1973], [Bahgat and Willmert, 1976], [Sunada and Dubowsky, 1981], [Erdman and Sandor, 1984], [Ho and Herber, 1985], [Naganathan and Soni, 1987]. Then the elastic motion is determined on the basis of the nominal rigid motion, and it is superposed on the nominal rigid motion to obtain the total motion of the system.

Some researchers have used the multibody formalism to model flexible systems. In this approach, a set of rigid body variables and a set of deformation variables are defined. The rigid variables describe the overall motion of the moving frame attached to each body. A set of nonlinear constraint equations is used to describe the coupling of the bodies at the joints [Shabana, 1985], [Yoo and Haug, 1986], [Garca de Jalon, et al., 1993] or Kane’s equations are used to eliminate the non-working internal constraint forces without differentiation of scalar energy functions [Huston, 1981], [Xie, 1994]. The elastic deformation of each flexible body is described with respect to its own rigid frame. This elastic deformation may be described by a finite element method [Shabana, 1985], [Yoo and Haug, 1986], [Xie, 1994], [Sunada and Dubowsky, 1981], [Geradin et al., 1984] or it may be described by a vibration mode approach, [Cetinkunt et al., 1986], [Johanni, 1988].

Flexibilities do not only exist in links, but they also exist in joints. Good and others (1985) presented analytical models and experimental data to show that interactions between electromechanical drives and compliant linkages to the drive points of the arm links are of fundamental importance to robot control system design. They showed that flexibility in the joints and bending and torsion of the links are the two principal sources of mechanical flexibility in a robot. Spong (1987) and Jankowski (1993) gave a simple model to represent the dynamics of elastic joint manipulators with this assumption and the model is useful for cases where the elasticity in the joints is of greater significance than gyroscopic interactions between the motors and links. Lieh (1994) studied the dynamic modelling of a slide-crank mechanism with coupler and joint flexibility and showed that the joint motion and coupler deformation are highly coupled.

Controlling flexible robots is characterized by the attempt to use those control concepts and algorithms, that are being used successfully in the control of rigid robots. Control design strategies depend on the problem to be solved. If a manipulator is only expected to hold in a fixed position, a PI (Proportional Integration) controller designed on the basis of linearized models will yield satisfactory results [Troch and Kopacek, 1988]. If no high-frequency
vibrations are present and only deformations of the links must be controlled, open-loop control based on the Lagrange equations with an PD (Proportional Derivative) controller may be applied successfully [Kalker and Olsder, 1987], [Yigit, 1994]. Jonker (1990) compared the actuator joint feedback with the end-point feedback control introduced by Schmitz (1985). The design of feedback controls for flexible arms is normally based on approximate models for the arms and simplification is achieved by linearization being performed around a reference trajectory.

Based on the finite element method described by Besseling (1963, 1975), a finite element method for the kinematic and dynamic analysis of planar and spatial mechanisms and manipulators with flexible links has been initiated and developed by van der Werff (1977, 1983), Besseling (1977, 1979, 1982), [Besseling and et al., 1985], Jonker (1984, 1986a-b, 1988, 1989, 1990, 1991), [Jonker and Keus, 1988], Meijaard (1991a, 1991b) and some others [Schwab, 1983], and a software package, SPACAR, has been constructed for the dynamic analysis and simulation of spatial mechanisms and robots. In this finite element approach for mechanisms and manipulators, both links and joints can be modelled as finite elements. Instead of using constraint equations, permanent contact between bodies is achieved by letting them share common nodal points. This finite element description may be looked upon as an algebraic analogue to the continuous field description of deformations. As the description starts out from the expressions for the deformation modes of the elements as non-linear functions of the coordinates of the element-nodes, it can be easily used to analyse and simulate mechanisms and manipulators with flexible links.

This research work is based on the previous work mentioned above. First an organic structure of the whole software system is constructed using an integrated program network. And a friendly interface is designed on workstations using the X Window software environment which provides a rich and complex environment to the programmer and user of application software. Chapter 2 describes the basic structure of the software system and the X Window system. Objected-oriented programming has some special characteristics for finite element analysis and multibody system dynamics. It has the advantage of good maintainability and modularity. The finite element approach for mechanisms and manipulators we used has the advantage of consistency of representation of mechanisms and manipulators and the ability of solving complex spatial dynamic problems. A combination of these two may lead to an attractive product. So the concept of object-oriented programming is also introduced in this chapter.

Electrical machines are widely used in mechanisms and manipulators. For dynamic analysis and simulation, one may also need to include the effect of the dynamics of the electrical machines. In chapter 3, the dynamics of electrical machines is studied and its
implementation into the SPACAR system is performed. The generalized electrical machine theory gives us the opportunity to construct general procedures to include all kinds of electrical machines which can be transformed to the general machine.

In chapter 4, first the basic concept of the finite element representation of rigid and flexible links is given. Then the model of the flexible joint is introduced and its finite element representation and implementation into the SPACAR system is presented. Since calculating efficiency is one of the main concerns in dynamic simulation, rigid planar and spatial beam elements are introduced and examples have been given to show their calculating efficiency.

In chapter 5, the coordinate reduction method used in the finite element approach has been generalized to reduce differential-algebraic equations (DAEs) to differential equations (DEs). This generalization, on the one hand, is an approach to reduce DAEs, and on the other hand, makes it possible for the user of the SPACAR system to use constraint conditions specified by himself. Then the linearization of the reduced differential equations is discussed. The methods for the determination of stationary and periodical solutions for mechanical system are applied to calculate the initial conditions for a dynamic system. Finally, some numerical examples for the dynamic analysis of mechanical systems are presented.

Chapter 6 deals with the simulation of manipulators with flexible links and flexible joints. Both the variational linearization method and the feedback linearization method are discussed and implemented into the SPACAR systems. Trajectory control of manipulators with flexible links or/and flexible joints is studied and simulation examples are presented.
Chapter 2

Software Engineering

2.1 Introduction

In a numerical simulation design, one should first give the mathematical model of a physical system. Then according the mathematical model, the simulation software system is designed and the simulation can be performed. Both the correctness of the mathematical model and the software have great effects on the performance of the simulation. Without a correct mathematical model, the simulation could not represent the reality of the physical system. On the other hand, without a correct software, the simulation would also fail to reflect the physical system even if a correct mathematical model has been constructed.

Software is the collection of computer programs, procedures, rules, and associated documentation and data. It can be noticed that software is not just the program, but comprises all of the associated data and documentation as well. So the methodologies for software development should also focus on the development of data and documentation, and not simply on the development of computer programs. Software is not merely a collection of computer programs. A program is generally complete in itself, and is generally used only by the author of the program. A programming system product, on the other hand, is used largely by people other than the developers of the system. A program to solve a problem, and a programming system product to solve the same problem are two entirely different things. Much more effort and resources are required for a programming system product. As a rule of thumb, a programming systems product costs approximately ten times as much as a corresponding program [Jalote, 1991].

Software engineering is the systematic approach to the development, operation, maintenance, and retirement of software [Jalote, 1991]. It is considered as a triad of structured programming, structured documentation, and structured management [Ejiogu, 1983]. The term "systematic approach" for the development of software means that software engineering is to provide methodologies for developing software that are as close to the scientific method as
possible. The goal of software engineering is, in essence, to take software development closer
to science and to produce high quality software at low cost.

In this chapter, the structured programming we used in the design of SPACAR will be
briefly introduced. Also a basic introduction of the X Window and its toolkit will be given. As
objected-oriented programming has some special characteristics for finite element analysis
[Forde et al., 1990], [Dubois-Pelerin and Zimmermann, 1992], [Dubois-Pelerin et al., 1992],
[Zimmermann et al., 1992] and multibody system dynamics [Koh and Park, 1994], we
introduce the principle of object-oriented programming at the last part of this chapter.

2.2 Integrated program network and data base

In the SPACAR software system, a mechanical system is represented by finite elements. The
analysis of kinematics, dynamics, inverse dynamics, linearization, periodic solutions of
mechanical systems and control simulation of a robot can be performed. With so many
problems concerned, it is not a good idea to treat them within a single discipline. And on the
other hand, this software system may need to communicate with other commercial software.
For example, when we want to design a manipulator, we may simulate the system using
SPACAR and use MATLAB to determine the suitable control parameters. So there is data
communication between these two software systems. We meet a similar case when we use the
calculated data from SPACAR for a text editor through some graphic software like GRTOOL
to transfer the pictures. In the latter case, the data communication is a one way direction. With
several programs integrated, we use the concept of an integrated program network.

An integrated program network (IPN) is a set of independently-executable program
modules coupled through some interfacing mechanism that enforces some degree of
operational compatibility [Felippa, 1979]. There are many advantages of using IPN. As the
range of application of computer analysis expands, it can easily contain more technical
disciplines amenable to compatible modelling procedures. And the IPN approach also provides
a way of utilizing off-the-shelf programs that would be of little use as isolated entities and
permit to use the useful portions of obsolescent programs. In computer-aided design, if
relatively small program modules can be individually run in interactive-demand or
conversational mode, applications can be achieved more easily. The IPN approach is also the
solution of money shortage problem when we deal with a large software system. We can
organize the system into comparatively small executable programs. And also, the development
and maintenance of software by different programming groups is simplified.

There are three ways of organizing an IPN as shown in Fig.2.1, direct link (DL) approach,
super-executive (SE) approach and data base (DB) approach. In the DL approach, programs
communicate directly through made-to-order interfaces. There is one interface for every
possible connection path. The SE approach links the network components to an executive program that functions as a network operating system. In this case, the IPN is tightly coupled in the form that input data format, inter-program data flow and control procedures must conform to the specifications dictated by the developer of the executive program. The program components cannot run without the executive when they are connected to the IPN. The DB approach lets the network components communicate through a database management system (DBMS). Data streams flowing among IPN components go in and out of database through the DBMS. In this case, network components can keep their own control input data format and internal data handling conventions and the IPN is loosely coupled.

When arranging an IPN, one can choose one of the three approaches or mix more than one approach together. Here we introduce the combination of the DB approach and the SE approach (DB-SE) as shown in Fig. 2.2. Here the DBMS of the system also functions as the super executive program of P3, P4 and P5. The programs P3, P4 and P5 are tightly coupled and their control procedures, input data format, and inter-program data flow are dictated by the executive program. Program P1 and P2 are just loosely connected to the IPN through the DBMS.

With the DB approach and a mixed approach, one normally needs a database and a database management system (DBMS).

A database is a collection of stored operation data used by the applications system of some particular enterprise. Here the operation data include all the information used by more than one program in an integrated program network. For example, in the dynamic simulation of a robot, the material mass, stiffness and damping parameters, external forces, geometric information of the robot, information of the electrical or hydraulic drivers, prescribed trajectories and their velocity profiles and so on are all treated as operation data because they are used by, e.g. inverse dynamic analysis and simulation analysis. Operational data do not include transitory information such as, for instance, intermediate results generated during an analysis run, as the first or second order geometric transfer functions. It should be noticed that operation data and non-operational data is not an absolute concept. For example, the time history of the coordinates of a node in an inverse dynamic analysis is non-operational data when it is not to be used later on. But when we want to compare this data with that from the simulation results or when we want to use it in feedback control, it is operational data, because we want to keep it to be used in a word processor or in the simulation analysis later on.
a. Direct link (DL) approach

b. Super-executive (SE) approach

c. Database (DB) approach

Fig. 2.1 Arrangement of an integrated program network (IPN)

Fig. 2.2 A mixed approach of DB and EX (EBEX)
We use the database structure shown in Fig.2.3. In this database structure, a mechanism or a robot is defined as a collection of rigid and flexible finite elements and nodes. A mechanism or robot is represented by the finite elements and the nodes are connected with elements. But many parameters of a mechanism or robot, as lumped masses, geometrical configuration, external forces, are directly given by the nodal point. From the point of view of information, a mechanism or robot is represented by elements and nodes.

![Diagram](image)

**Fig.2.3 The database structure**

In the database structure, a single line illustrates one to one relationship and a fork illustrates a one to several or many to many relationships. For example, the relationship between mechanism and elements is a one to several, and the relationship between elements and nodes is a many to many relationship.

With a database, one also needs a database management system (DBMS). A DBMS is the software complex that handles all access requests and transactions against the database. In the
DB-SE approach, the DBMS is combined with the super executive program. The database structure shown in Fig. 2.3 is implemented with the super-executive. So the communication between the database, the applications and the user interface are taken care of and controlled by the super executive. For example, the creation and modification of elements and nodes, and information input are performed in the super executive. When one needs to transfer the calculated results to a word processor after an analysis, one should also use the super executive and have the data transferred to a suitable form for the word processor. On the other hand, if one needs to design the control parameters for a robot, first the robot is represented by elements and nodes and is input through the super executive. Then the state parameters are calculated and transferred to a form, suitable for use in MATLAB, and one can design the control parameters in MATLAB. After the design of the control parameters, one can input the parameters and run the simulation program through the super executive.

2.3 X Window system and interface of X SPACAR

To make a software system usable, a friendly interface is as important as the calculation scheme or database management. X Windows is widely used in work stations as a graphic interface. We also use X Windows in the pre- and post-processors for the SPACAR system. So in this section, the basic concept of the X Windows system is introduced and then the structures of the pre- and post-processors of the SPACAR system are introduced.

2.3.1 The X Window

The X Window System is a software environment for workstations. It has a rich and complex environment to the programmers and users of application software. The foundation of X Window is the base window system. The overall X environment consists of layers upon the base window system as shown in Fig.2.4. The base window system interfaces with the outside world using the X network protocol. The X network protocol specifies what makes up each packet of information that gets transferred between the server and Xlib in both directions. The network protocol interface is designed to work either within a single processor or between processors. Even when the server and Xlib are running on the same machine, the protocol is used for communications, through some internal channel instead of the external network. Normally, X application programs do not use the network protocol directly, but rather work through a programming interface, e.g. the Xlib. And more often, application programs use the high-level X toolkit to mask some of the complexity of the network protocol.
# 2.3 X Window system and interface of X SPACAR

<table>
<thead>
<tr>
<th>Windows, Session Managers</th>
<th>High-level X Toolkit</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low-level Programming Interface (Xlib)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X Network Protocol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Base Window System</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.4 The structure X Window

The user’s interface to the workstation is the combination of the capabilities provided by the base window system, the X toolkit, windows and session managers, and application software. The base window system provides mechanisms to support a wide variety of human interface policies and leave the implementation and enforcement of those policies to other software.

The purpose of the X Window System is to provide a network-transparent and vendor-independent operating environment for workstation software. The operating environment supports multiple overlapping windows on a variety of colour and monochrome workstations. Network transparency is that X applications running on one processor can show their output using a display connected to either the same cpu or some other cpus. A variety of computing styles ranging from stand-alone workstations, running application locally, to time-shared mainframe using many different workstations as if they were terminals, can co-exist in the X environment. Network transparency makes applications run on whatever cpu is most convenient. For example, applications requiring extensive computations can run on a network-connected super-computer while an application requiring a large shared data base can run on the cpu containing that data base and connect to users’ workstations over the network using X. An X workstation can connect many different host processors to the user at the same time. It can connect to many different applications on any cpu at the same time and each of these applications can use as many windows as it needs to display output or receive input.

The vendor-independent means that X applications are portable. They deal with X, so they do not need to know the detail of any particular workstations’ display hardware. As long as an X application is able to establish a connection to a workstation, it can use all the capabilities of the base window system of that workstation. An X application running on a cpu from one
vendor can use any workstation model either from that vendor or from another because the workstation hardware is hidden by the protocol.

For an application with graphical user interface, one needs to use X toolkit instead of using the Xlib directly. At the beginning of the design of the SPACAR X version, we had available XView, the OPEN LOOK Toolkit for X11. So we have used it in the design of the pre- and post-processors. Now with the Motif, X Toolkit Intrinsics, the Motif version is under development. So here we introduce briefly the basic concept of both XView and Motif.

XView (X Window-System-based Visual/Integrated Environment for Workstations) is a user-interface toolkit to support interactive, graphics-based applications running under the X Window System. It provides a set of pre-built, user-interface objects such as canvases, scrollbars, menus and control panels. XView simplifies application development under the X Window System by providing the programmer with a set of pre-defined user interface components. To the programmer, XView is an object-oriented toolkit. Its objects can be considered as building blocks from which the user interface of the application is assembled. Each object is from a particular package. Each package provides a list of properties from which we can choose to configure the object. By selecting objects from the available packages, we can build the user interface for an application.

XView defines classes of objects in a tree hierarchy. Fig.2.5 show all the classes available, the class hierarchy and the relationships between the classes [Heller, 1990]. Each class has its identifying features that make it unique from other classes. All objects of a particular class inherit the properties of the parent class. The Generic Object contains certain basic properties that all objects share. When an object is created, the XView function returns a handle for the object. Later, when we wish to manipulate the object or inquire about its state, we pass its handle to the appropriate function. For every object, there is a large set of attributes attached to it. XView provides a common set of functions that allows the programmer to manipulate any object by referencing the object handle and set the values of the attributes of the object. The functions are listed in Table 2.1.
Fig.2.5 XView class and its hierarchy

With these six functions in Table 2.1, we can create all the objects we need, destroy them when necessary. We can also set values for any attributes of any objects and inquire their values.

Motif is a set of pre-build widgets using X Toolkit Intrinsics. The Motif widget is a set of functions and procedures that provide quick and easy access to the lower levels of the X Window System. It is based on the Xt Intrinsics. As shown on Fig.2.6, the Motif widget system is layered on top of the Xt Intrinsics, which in turn are layered on top of the X Window System.
Table 2.1. Generic functions

<table>
<thead>
<tr>
<th>function</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>xv_init()</td>
<td>Establishes the connection to the server, initializes the Notifier and the Defaults/Resource-manager database, loads the Server Resource-Manager database, and reads the ~/.Xdefaults database and any passed attributes</td>
</tr>
<tr>
<td>xv_create()</td>
<td>Creates an object.</td>
</tr>
<tr>
<td>xv_destroy()</td>
<td>Destroys an object</td>
</tr>
<tr>
<td>xv_find()</td>
<td>Finds an object, if doesn’t exist, creates it.</td>
</tr>
<tr>
<td>xv_get()</td>
<td>Gets the value of an attribute.</td>
</tr>
<tr>
<td>xv_set()</td>
<td>Sets the value of an attribute.</td>
</tr>
</tbody>
</table>

![Diagram](attachment:image.png)

**Fig.2.6 User interface development model**

The Motif widget system [OSF/Motif, 1991a, 1991b] consists of a number of different widgets, each of which can be used independently or in combination to aid in creating complex applications. Every widget is dynamically allocated and contains state information. Every widget belongs to one class and each class has a structure that is statically allocated and initialized and contains operations for that class. Fig2.7 shows the Motif widget set, its class hierarchy and its relationship with the classes defined by Xt [Nye and O’Reilly, 1990a, 1990b]. The Core widget class, defined by the Intrinsic, is the root of the hierarchy and contains resources that are inherited by all other classes. It defines characteristics common to all widgets. Two classes, the Composite class and the Primitive class, are layered beneath the Core class. Each lower class can inherit some or all of the resources belonging to a higher class. The details of the Motif widgets can be found in references [Nye and O’Reilly, 1990a, 1990b].
Motif contains a variety of widgets and gadgets, each designed to accomplish a specific set of tasks, either individually or in combination with others. Widgets are grouped into several classes depending on the function of the widget. A widget class consists of the procedures and data associated with all widgets belonging to that class. These procedures and data can be inherited by subclasses. Physically, a widget class is a pointer to a structure. The contents of this structure are constant for all widgets of the widget class. An instance of widget is allocated and initialized by the functions XmCreateWidgetname, XtCreateWidget, or XtCreateManagedWidget.

Gadgets can be thought of as windowless widgets. They do not have windows, translations,
actions, or pop-up children. Gadgets provide essentially the same functionality as the equivalent primitive widgets. The gadgets and their class names of Motif are shown in Table 2.2.

<table>
<thead>
<tr>
<th>Gadget name</th>
<th>Gadget class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>objectClass</td>
</tr>
<tr>
<td>RectObj</td>
<td>rectObjClass</td>
</tr>
<tr>
<td>XmGadget</td>
<td>xmGadgetClass</td>
</tr>
<tr>
<td>XmArrowButtonGadget</td>
<td>xmArrowButtonGadgetClass</td>
</tr>
<tr>
<td>XmSeparatorGadget</td>
<td>xmSeparatorGadgetClass</td>
</tr>
<tr>
<td>XmLabelGadget</td>
<td>xmLabelGadgetClass</td>
</tr>
<tr>
<td>XmCascadeButtonGadget</td>
<td>xmCascadeButtonGadgetClass</td>
</tr>
<tr>
<td>XMpushButtonGadget</td>
<td>XMpushButtonGadgetClass</td>
</tr>
<tr>
<td>XmToggleButtonGadget</td>
<td>xmToggleButtonGadgetClass</td>
</tr>
</tbody>
</table>

### 2.3.2 Design principles of the interface

The interface of the X SPACAR is designed using the X Window System and its easy programming Toolkits. They are designed as a menu driven graphic interface. To design a friendly interface, the following basic principles are proposed [Foley and van Dam, 1984]

- **Consistency**: This is the similarity of patterns which may be perceived in tasks, in presentation of information and other facets of an interface design. Consistency reduces the human learning load and increases recognition by presenting a familiar pattern. The more consistent patterns are, the easier an interface will be and the less the user has to learn. The conceptual model, semantics, command language syntax, and display formats of the system are uniform and lack exceptions and special conditions. Consistency includes: The same codings are always employed; System status messages are shown at a logically fixed place on the display surface; Menu items are displayed in the same relative position within a menu; Keyboard characters always have the same function and can be used whenever text is being input; Global commands can be invoked at any time; Keywords in a character-string-oriented language can be abbreviated with a constant-length string.

- **Compatibility**: The reality of an interface design should be compatible with the user’s expectation. This principle follows on from consistency to state that new designs should be
compatible with, and therefore based upon, the user's previous experience. If this is followed, once again recognition is enhanced, learning is reduced and the interface will be easier to use. If there is a choice to make things easier either for the user or for the designer, make the choice in favor of the user. Minimize both mental and physical work.

- Adequate feedback: Feedback is information received by the user from the computer, indicating that a previous operator action has had an effect. Feedback is as essential an ingredient in conversation with a computer as it is in human conversation. There are three possible levels of feedback corresponding to the levels of the language: lexical, syntactic, and semantic in interactive systems. Whether feedback should be present and if so, what form it should take must be decided. Lexical is the lowest level of feedback. Each lexical action in the input language can be provided with a lexical response in the output language: e.g., echoing characters typed on a keyboard and moving a screen cursor as the user changes the position of a locator. The syntactic feedback is the response by the system when a unit of input language (command, position, picked object, etc.) is received. The semantic feedback tells the user that the requested operation has been completed or gives some indication that the computer is working on the problem. This is usually done with a new or modified display which explicitly shows the results.

- Guidance not control: Give the operator control. The interface should function at the user's pace according to the user's command and should not attempt to control the user. This implies two things: the user should be able to forecast what to do next from a system's current state and user should be able to backtrack at will when mistakes are made. The users' experience and skill may be different and the interface should be flexible to deal with the differing preferences and skills of different operators.

This principle can be described in the flowing common-sense terms: Do not present information too quickly. Let users control the rate at which they receive information by letting them start and stop the information as necessary; do not present too much information at once. Present what is needed for accomplishing the task at hand, and let users access other information as they need it; Filter information. Select and display the relevant information; Keep error messages and the like brief and factual and show them so that they do not sound scolding or humorous.

- Economy: Interface designs should be economic in the sense that they achieve an operation in the minimum number of steps necessary to support the user and lessen the work of users whenever possible.

- Structure: Interface design should be structured to reduce complexity. Structuring should be compatible with the user's organization of knowledge and not overburden memory. This leads to a sub-component of simplicity and relevance. Information should be organized so
that only relevant information is presented to the user in a simple manner. To help to structure the information, it is helpful to divide the view surface into different areas where different specific types of information are presented. Prompts, error messages, system status, and graphical representation of the information of interest to the user each has its own area.

- Minimize memorization: Users need to memorize information when they learn to use a system. To make the memorization as simple as possible and indeed to avoid it whenever possible is an important design principle. In interface design, one may need to make decisions that determine whether users will be required to recognize or to recall information. It is easier to recognize familiar information than to recall it from memory.

2.3.3 The main interface

The interface of the X SPACAR is designed according to the principles mentioned above. The main window of the interface is shown in Fig.2.7. All the commands are displayed as a menu form. The detailed description of the pre- and post-processors can be found in a separate report [Gong, 1992].

![SPACAR Interface](image)

Fig.2.7 The main window of the X SPACAR interface
2.4 Object-oriented analysis, design and programming

2.4.1 Introduction

A feature of a software system like SPACAR is its evolution: in practice the areas covered by the software, the algorithms it implements, the continuing contributions from generations of master and doctoral students, its graphical facilities and so on, are never fully foreseen at the time the program is initially designed. So the brain and man power invested in a large software system are spent mostly during its evolution and the share of the initial design and coding phase is usually small. Therefore maintainability is one of the key qualities. Maintainability includes understandability, extensibility and easy debugging. Understandability is the feature that a stranger can easily find in the program any given feature, and figure out what any feature encountered in the program means. This is mainly obtained through readability, homogeneity and self-description. Readability requires a simple language syntax and instructions. Homogeneity comes from the same conceptual complexity of the operations. And self-description is that the program automatically contains enough explanations on itself without requesting from the programmer any extra documentation such as a user’s manual. Extensibility is the practical possibility for the programmer to add new features to the software and reuse the existing portions of code.

Modularity has often been used to improve code maintainability. But the traditional procedural programming uses external control on the data and that limits the modularity. External control on the data is that a subroutine does not really manage the data it operates upon. That is an obvious transgression to modularity. So it is not so easy to get maintainability by the traditional procedural programming.

The object-oriented approach is the latest trend in programming practice. This trend is part of a general move towards a technology that reflects the concerns of software developers, and it has many advantages. In the object-oriented approach, given a set of objects, applications can be created within an expansible framework. Development time and size of the resulting code can be reduced considerably. Expansible application frameworks allow developers to create fully functional application software without having to worry about the implementation of many tedious details. They may be hybrid development environments, which graft object-oriented concepts onto existing procedural languages, so that existing products can be maintained. Object-oriented analysis and design is the method that leads us to an object-oriented decomposition. That is what we have been doing in the finite element analysis of mechanisms and manipulators. So it is most fitting to use the objected-oriented method in the finite element analysis of mechanisms and manipulator robots.

Object-oriented software engineering includes object-oriented analysis, object-oriented
design and object-oriented programming. Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain. It emphasizes the building of real-world models, using an object-oriented view of the world. Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design. A design method emphasizes the proper and effective structuring of a complex system. The object-oriented design leads to an object-oriented decomposition and uses different notations to express different models of the logical and physical design of a system in addition to the static and dynamic aspects of the system. It uses class and object abstractions to logically structured systems instead of using algorithmic abstractions in the traditional structured design. Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and the whose classes are all members of a hierarchy of classes united via inheritance relationships. In the object-oriented programming, objects, not algorithms, are used. Each object is an instance of some class and classes are related to one another via inheritance relationships.

In this section, we introduce the basic concept of object-oriented modelling and its use in finite element analysis of mechanical systems. The conceptual framework of object-oriented programming is the object model. This model has four major elements, abstraction, encapsulation, modularity and hierarchy. As declared by Booch (1994), without any one of these elements, the model is not object-oriented.

2.4.2 Abstraction

An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer [Booch, 1994]. An abstraction serves to separate an object’s essential behaviour from its implementation by focusing on the outside view of an object.

The spectrum of abstraction includes entity abstraction, action abstraction, virtual machine abstraction and coincidental abstraction. Entity abstraction is an object that represents a useful model of a problem-domain or solution-domain entity. Action abstraction is an object that provides a generalized set of operations, all of which perform the same kind of function. Virtual machine abstraction is an object that groups together operations that are all used by some superior level of control, or operations that all use some junior-level set of operations. Coincidental abstraction is an object that packages a set of operations that have no relation to each other. The entity abstraction directly parallels the vocabulary of a given problem domain and is the most useful abstraction.
2.4 Object-oriented analysis, design and programming

The central problem in object-oriented design is to decide upon the right set of abstractions for a given domain.

2.4.3 Encapsulation

Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behaviour; encapsulation serves to separate the contractual interface of an abstraction and its implementation [Booch, 1994]. Abstraction and encapsulation are complementary concepts: abstraction focuses upon the observable behaviour of an object, whereas encapsulation focuses upon the implementation that gives rise to this behaviour. Encapsulation hides the details of the implementation of an object and it is most often achieved through information hiding. Information hiding is the process of hiding all the secrets of an object that do not contribute to its essential characteristics; typically, the structure of an object and the implementation of its methods are hidden.

In C++, members of an object may be placed in the public, private, or protected parts. Members declared in the private are fully encapsulated; members declared in the public parts are visible to all clients; and members declared in the protected parts are visible only to the class itself and its subclasses.

2.4.4 Modularity

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules. Partitioning a program into individual components can reduce its complexity and creates a number of well-defined, documented boundaries within the program. These boundaries, or interfaces, are invaluable in the comprehension of the program.

Modularization consists of dividing a program into modules which can be compiled separately, but have connection with other modules. Classes and objects form the logical structure of a system and these abstractions in modules are placed in modules to produce the system's physical architecture. The use of modules is essential to help to manage complexity for larger applications. Modularity and encapsulation go hand in hand. Encapsulation dramatically improves modularity by enhancing code understandability in the following two ways. First, encapsulation structures data into objects characterized by attributes. This natural way of thinking simplifies comprehension for the programmer. Thus a method's interface is significantly shorter than that of a comparable classical subroutine. Second, encapsulation protects the attributes of the objects. The attributes of an object can be modified only by the object itself.

Decomposition of a system into small modules may be quit difficult because the solution
may not be known when the design stage starts. Also modularization in object-oriented design is different from that in traditional structured design. In traditional structured design, modularization is primarily concerned with the meaningful grouping of sub-programs by using the criteria of coupling and cohesion. In object-oriented design, the task is to decide where to package physically the classes and objects from the design’s logical structure, which are distinctly different from sub-programs.

The reduction of software cost by allowing modules to be designed and revised independently is the overall goal of the decomposition into modules. Some basic guidelines for decomposition are as follows.

- Each module’s structure should be simple enough that it can be fully understood.
- The implementation of a module can be changed without knowledge of the implementation of other modules and without affecting the behavior of other modules.
- A module’s interface should be as narrow as possible under the condition that the needs of all using modules are satisfied.
- It is easy to make a change in design when it is needed.
- Classes and objects are chosen to be packaged into modules in a way that makes their reuse convenient.

2.4.5 Hierarchy

Hierarchy is a ranking or ordering of abstractions. A set of abstractions often forms a hierarchy. A problem can be easily understood by identifying these hierarchies in the design. There are two most important hierarchies in a complex system: inheritance (class structure) and aggregation (object structure).

Inheritance is the most important hierarchy and it is an essential element of object-oriented systems. Inheritance defines a relationship among classes, wherein one class shares the structure or behaviour defined in one or more classes. It represents a hierarchy of abstractions in which a subclass inherits from one or more super-classes. The subclass augments or redefines the existing structure and behaviour of its super-classes. It denotes generalization/specialization relationships.

Aggregation relationships are also a hierarchy. It is called “part of” hierarchies. For example, in the definition of a class, some variables may be defined as the type of some other classes. So the new class contains some objects of other classes. This is a kind of aggregation. It represents a kind of ownership, the objects belong to the newly defined class.

The combination of inheritance and aggregation is powerful: aggregation permits the
physical grouping of logically related structures, and inheritance allows these common groups to be easily reused among different abstractions.

2.4.6 Object-oriented finite element programming for the SPACAR system

As C++ has the advantages of object-oriented programming with the computational efficiency of the C language and is likely to be the most commonly used object-oriented language in the years to come due to its compatibility with the heavily-used C language, and to the fact it is in the public domain [Dubois-Pelerin and Zimmermann, 1993], we use it for the constructing of the object-oriented finite element programming of the SPACAR system.

Objects are basic elements in object-oriented programming. An object is provided with a set of variables, referred to as its attributes. An object can also perform operations which are called its methods. A system usually involves several objects of the same type. Object-oriented programming organizes the objects of the same type in groups called classes. An object is said to be an instance of its class. In C++, all classes are designed by the programmer. Unlike traditional procedure programming, that one programs by adding procedures to the library, in object-oriented programming, programs are made by enlarging a library which contains classes. So to design classes is one of the main concerns of object-oriented programming.

In the SPACAR software system, mechanical systems are represented by finite elements. All the information, like inertial properties, stiffness, load forces and so on, are related to an element or to a node. And all the operations are performed on arrays or matrices. So the most often used classes are elements, nodes, arrays and matrices. In this section, the basic structure of the classes Element and Node will be presented.

Fig. 2.8 shows the above mentioned basic classes and their hierarchy. In the figure, the sub-classes of class Element include the classes Plane-beam-element, Plane-truss-element, Plane-hinge-element, Spatial-beam-element, Spatial-truss-element, Spatial-hinge-element and so on. Class Element is the super-class of these sub-classes, which inherit the properties of the super-class, class Element.

In can be noticed from Fig. 2.8 that for different elements, different classes are defined. For different nodes, just one class, class Node, is defined so we treat different type of nodes under the same class. This depends on the taste of the programmer and there is not much difference between these two structures concerning the performance of the program.

The attributes of class Element is given in Table 2.3 and the methods of class Element are listed in Table 2.3. In Table 2.3, all the methods except the last ten deal with the attributes of the element itself. The last ten methods return the current value array or matrix of the element. These values are stored in the system arrays and matrices, that may be defined as objects. To
get these values in a method defined in class Element, we may define a method in class Element as “friend” of the corresponding system array or matrices. For example, the method give-deformation defined in class Element (Fig. 2.9) may be defined as a friend of class system-deformation so that give-deformation can access the attributes of class system-deformation (Fig. 2.10).

For the structure of the sub-classes of class Element, as we have given all the attributes of an element class in the super-class Element, we just need to define some methods for these sub-classes to calculate their properties. For example, the methods, except the ones defined in class Element, for sub-class Spatial-beam-element are shown in Table 2.5.

The structure of class Node is somewhat like that of class Element. The attributes of class Node are shown in Table 2.6, while the methods are given in Table 2.7.

Element
  Plane-beam-element
  Plane-truss-element
  Plane-hinge-element
  Spatial-beam-element
  Spatial-truss-element
  Spatial-hinge-element


Node
Integer-array
Float-array
Integer-matrix
Float-matrix

Fig. 2.8 The basic classes and their hierarchy
Table 2.3. Attributes of class Element

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>number-of-nodes</td>
<td>the number of nodes the element has</td>
</tr>
<tr>
<td>number-of-deformation</td>
<td>the number of deformation functions</td>
</tr>
<tr>
<td>number-of-coordinates</td>
<td>the number of nodal coordinates connected</td>
</tr>
<tr>
<td>location-of-deformation</td>
<td>system defined deformation number</td>
</tr>
<tr>
<td>location-of-coordinate</td>
<td>system defined coordinate number</td>
</tr>
<tr>
<td>nodes-array</td>
<td>array containing the node numbers connected to the element</td>
</tr>
<tr>
<td>deformation</td>
<td>array containing the initial deformation</td>
</tr>
<tr>
<td>speed</td>
<td>array containing the initial speed of deformation</td>
</tr>
<tr>
<td>acceleration</td>
<td>array containing the initial acceleration of deformation</td>
</tr>
<tr>
<td>mass</td>
<td>mass per-unit length of the element</td>
</tr>
<tr>
<td>length</td>
<td>length of the element (beam and truss element)</td>
</tr>
<tr>
<td>stiffness-matrix</td>
<td>stiffness matrix of the element</td>
</tr>
<tr>
<td>damping-matrix</td>
<td>damping matrix of the element</td>
</tr>
<tr>
<td>stress</td>
<td>array containing the initial stresses</td>
</tr>
</tbody>
</table>

Table 2.4. Methods of class Element

<table>
<thead>
<tr>
<th>Methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>set-number-of-nodes</td>
<td>set the number of nodes of the element</td>
</tr>
<tr>
<td>give-number-of-nodes</td>
<td>give the number of nodes of the element</td>
</tr>
<tr>
<td>set-number-of-deformation</td>
<td>set the number of deformation of the element</td>
</tr>
<tr>
<td>give-number-of-deformation</td>
<td>give the number of deformation of the element</td>
</tr>
<tr>
<td>set-location-of-deformation</td>
<td>set the system defined deformation number array</td>
</tr>
<tr>
<td>give-location-of-deformation</td>
<td>give the system defined deformation number array</td>
</tr>
<tr>
<td>set-location-of-coordinates</td>
<td>set the system defined coordinate number array</td>
</tr>
<tr>
<td>give-location-of-coordinates</td>
<td>give the system defined location of the coordinate array</td>
</tr>
<tr>
<td>set-nodes</td>
<td>set the node array of the element</td>
</tr>
<tr>
<td>give-nodes</td>
<td>give the node number array</td>
</tr>
</tbody>
</table>
Table 2.4. Methods of class Element (continued)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>set-initial-deformation</td>
<td>set the initial values of deformation, its rate and acceleration</td>
</tr>
<tr>
<td>give-initial-deformation</td>
<td>give the initial values of deformation, its rate and acceleration</td>
</tr>
<tr>
<td>set-mass</td>
<td>set the value of the mass per-unit length</td>
</tr>
<tr>
<td>give-mass</td>
<td>give the mass per-unit length</td>
</tr>
<tr>
<td>set-stiffness</td>
<td>set the stiffness matrix</td>
</tr>
<tr>
<td>give-stiffness</td>
<td>give the stiffness matrix</td>
</tr>
<tr>
<td>set-damping</td>
<td>set the damping matrix</td>
</tr>
<tr>
<td>give-damping</td>
<td>give the damping matrix</td>
</tr>
<tr>
<td>set-initial-stress</td>
<td>set the initial stress array</td>
</tr>
<tr>
<td>give-initial-stress</td>
<td>give the initial stress array</td>
</tr>
<tr>
<td>give-deformation</td>
<td>give current values of the deformation array</td>
</tr>
<tr>
<td>give-deformation-rate</td>
<td>give current values of the deformation rate array</td>
</tr>
<tr>
<td>give-def-acceleration</td>
<td>give current values of the deformation acceleration array</td>
</tr>
<tr>
<td>give-coordinates</td>
<td>give the current values of the coordinates array</td>
</tr>
<tr>
<td>give-coordinate-rate</td>
<td>give the current coordinate rate array</td>
</tr>
<tr>
<td>give-coordinate-acceleration</td>
<td>give the current coordinate acceleration array</td>
</tr>
<tr>
<td>give-first-order-matrix</td>
<td>give the first order geometric transfer function matrix</td>
</tr>
<tr>
<td>give-second-order-matrix</td>
<td>give the second order geometric transfer function matrix</td>
</tr>
<tr>
<td>give-inertial-stress</td>
<td>give the inertial stress array</td>
</tr>
<tr>
<td>give-stress</td>
<td>give the stress array</td>
</tr>
</tbody>
</table>

Table 2.5. Methods of class Spatial-beam-element

<table>
<thead>
<tr>
<th>Methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate-d-matrix</td>
<td>calculate the first and/or second order partial derivatives of the deformation functions with respect to its coordinates</td>
</tr>
<tr>
<td>calculate-dynamic-contribution</td>
<td>calculate the contributions of the element to the mass matrix, inertial forces and inertial stress</td>
</tr>
</tbody>
</table>
class element {
    int number-of-deformation;
    int-array location-of-deformation;
    
    public:
    float-array give-deformation(int&, int-array&);
    
};

Fig. 2.9 Method give-deformation in class Element

class system-deformation-array {
    float deformation[NDEF];
    
    friend float-array element::give-deformation(int&, int-array&);

};

Fig. 2.10 Method give-deformation in class Element defined as friend of class system-deformation-array

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>node-type</td>
<td>the type of the node, e.g., translational or rotational</td>
</tr>
<tr>
<td>number-of-coordinates</td>
<td>the number of coordinates of the node</td>
</tr>
<tr>
<td>location-of-coordinates</td>
<td>system defined nodal coordinates of the node</td>
</tr>
<tr>
<td>initial-coordinates</td>
<td>array containing the initial values of the coordinates</td>
</tr>
<tr>
<td>initial-speed</td>
<td>array containing the initial values of speed</td>
</tr>
<tr>
<td>initial-acceleration</td>
<td>array containing the initial values of acceleration, mainly for an input degree of freedom</td>
</tr>
<tr>
<td>mass</td>
<td>inertial properties of the node</td>
</tr>
<tr>
<td>force</td>
<td>force/moment properties of the node</td>
</tr>
</tbody>
</table>
Table 2.7. Methods of class Node

<table>
<thead>
<tr>
<th>Methods</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>set-node-type</td>
<td>set the node type</td>
</tr>
<tr>
<td>give-node-type</td>
<td>give the node type</td>
</tr>
<tr>
<td>set-number-of-coordinates</td>
<td>set the number of coordinates of the node</td>
</tr>
<tr>
<td>give-number-of-coordinates</td>
<td>give the number of coordinates of the node</td>
</tr>
<tr>
<td>set-location-of-coordinates</td>
<td>set the system defined coordinate number array</td>
</tr>
<tr>
<td>give-location-of-coordinates</td>
<td>give the system defined coordinate number array</td>
</tr>
<tr>
<td>set-initial-values</td>
<td>set the initial values of the nodal coordinates</td>
</tr>
<tr>
<td>give-initial-values</td>
<td>give the initial values of the nodal coordinates</td>
</tr>
<tr>
<td>set-mass</td>
<td>set the array of inertial properties of the node</td>
</tr>
<tr>
<td>give-mass</td>
<td>give the inertial property array of the node</td>
</tr>
<tr>
<td>set-force</td>
<td>set the force array of the node</td>
</tr>
<tr>
<td>give-force</td>
<td>give the force array of the node</td>
</tr>
<tr>
<td>give-coordinates</td>
<td>give the current coordinate array of the node</td>
</tr>
<tr>
<td>give-speed</td>
<td>give the velocity array</td>
</tr>
<tr>
<td>give-acceleration</td>
<td>give the acceleration array</td>
</tr>
<tr>
<td>give-first-order-matrix</td>
<td>give the first order geometric transfer matrix</td>
</tr>
<tr>
<td>give-second-order-matrix</td>
<td>give the second order geometric transfer matrix</td>
</tr>
<tr>
<td>calculate-dynamic-contribution</td>
<td>calculate the contributions of the node to the mass matrix and inertial forces</td>
</tr>
</tbody>
</table>
Chapter 3

Dynamics of Electrical Machines

3.1 Introduction

Electrical machines are widely used as driving sources in machinery systems. When a dynamic analysis of a machinery system is performed, the dynamic properties of the electrical drivers may need to be taken into account, especially in the case of control and active damping of a machine system where electrical control signals are given. The software system SPACAR is used to simulate machines and manipulator robots with flexible links by using the finite element method. So the installation of electrical drivers into the system is necessary.

By using the finite element method in the SPACAR system, we can analyse and simulate many machines and manipulator robots. But concerning the electrical drivers, with so many electrical motors and alternators in use today, it would be a hard task to develop general software to analyse all kinds of machinery systems with different electrical drivers. Fortunately, the generalized machine theory for industrial electrical machines established by Park [Sengupta and Lynn, 1980], [Adkins, 1975], [Jones, 1967] and the generalities of the FE method used in the SPACAR system give us the opportunity to consider the dynamics of many kinds of electrical machines in general software.

3.2 Dynamics of generalized electrical machines

Although there are many essential differences in design and construction of electrical machines, which make the performance of a particular machine so different from another, analytically all electrical machines, synchronous, induction or d.c. machines, are basically similar and can approximately be treated as a primitive machine shown in Fig. 3.1. The rotor takes the place of the armature. The original rotor windings are replaced by fixed windings,
two by two on perpendicular axes. In Fig. 3.1, d is the direct axis and q is the quadrature axis of the generalized electrical machine. \( d_1 \) and \( d_2 \) are the two coils of the stator in the direct axis, and \( q_1 \) and \( q_2 \) are the two coils of the stator in the quadrature axis of the generalized machine. The rotor and stator coils may be increased or decreased to represent different machines.

![Diagram of generalized electrical machine](image)

**Fig. 3.1 Generalized electrical machine**

The flux-linkage equations for the generalized machine are [Sengupta and Lynn, 1980]

\[
\psi = Li
\]  
(3.1)

where

\[
\psi^T = \begin{bmatrix}
\psi_d & \psi_q & \psi_{d1} & \psi_{d2} & \psi_{q1} & \psi_{q2}
\end{bmatrix}
\]  
(3.2)

\[
i^T = \begin{bmatrix}
i_d & i_q & i_{d1} & i_{d2} & i_{q1} & i_{q2}
\end{bmatrix}
\]  
(3.3)

and
\[ L = \begin{bmatrix}
    L_d & 0 & M_{d1} & M_{d2} & 0 & 0 \\
    0 & L_q & 0 & 0 & M_{q1} & M_{q2} \\
    M_{d1} & 0 & L_{d1} & M_{d12} & 0 & 0 \\
    M_{d2} & 0 & M_{d12} & L_{d2} & 0 & 0 \\
    0 & M_{q1} & 0 & 0 & L_{q1} & M_{q12} \\
    0 & M_{q2} & 0 & 0 & M_{q12} & L_{q2}
\end{bmatrix} \] 

(3.4)

The subscript d indicates the direct axis, q the quadrature axis, L represents the self inductance and M the mutual inductance. E.g., \( M_{d1} \) is the mutual inductance between the rotor winding d and the stator winding d1, and \( M_{d12} \) the mutual inductance between stator windings d1 and d2.

It can be noticed that the flux-linkage in d-axis and q-axis are independent, that is

\[
\begin{bmatrix}
    \psi_d \\
    \psi_{d1} \\
    \psi_{d2}
\end{bmatrix} =
\begin{bmatrix}
    L_d & M_{d1} & M_{d2} \\
    M_{d1} & L_{d1} & M_{d12} \\
    M_{d2} & M_{d12} & L_{d2}
\end{bmatrix}
\begin{bmatrix}
    i_d \\
    i_{d1} \\
    i_{d2}
\end{bmatrix}
\]

(3.5)

and

\[
\begin{bmatrix}
    \psi_q \\
    \psi_{q1} \\
    \psi_{q2}
\end{bmatrix} =
\begin{bmatrix}
    L_q & M_{q1} & M_{q2} \\
    M_{q1} & L_{q1} & M_{q12} \\
    M_{q2} & M_{q12} & L_{q2}
\end{bmatrix}
\begin{bmatrix}
    i_q \\
    i_{q1} \\
    i_{q2}
\end{bmatrix}
\]

(3.6)

The voltage equations are,

\[
\begin{align*}
    u_d &= R_d i_d + \psi_d + \dot{\psi}_d \\
    u_q &= R_q i_q + \psi_q - \psi_d \\
    u_{d1} &= R_{d1} i_{d1} + \psi_{d1} \\
    u_{q1} &= R_{q1} i_{q1} + \psi_{q1} \\
    u_{d2} &= R_{d2} i_{d2} + \psi_{d2} \\
    u_{q2} &= R_{q2} i_{q2} + \psi_{q2}
\end{align*}
\]

(3.7)

The torque supplied by the electrical machine is
\[ T = \psi_d i_q - \psi_q i_d \quad (3.8) \]

To replace an electrical machine containing a rotor with two axes of symmetry and continuously distributed windings, placed in a stator with continuously distributed three phase windings, we use currents \( i_d \) and \( i_q \) to replace the three phase currents \( i_1, i_2 \) and \( i_3 \) with the condition that they give the same magnetomotive force (M.M.F.) in the air gap at any time. This condition is satisfied by letting the components of the M.M.F. due to the three phase currents \( i_1, i_2 \) and \( i_3 \) be equal to the M.M.F. of the direct axis current \( i_d \) and the quadrature axis current \( i_q \) respectively. So we have

\[ i_g = S i_r \quad (3.9) \]

where \( i_g \) is the current vector of the generalized machine and \( i_r \) the three phase current of the real machine,

\[ i_g^T = \begin{bmatrix} i_d & i_q & i_0 \end{bmatrix} \]

\[ i_r^T = \begin{bmatrix} i_1 & i_2 & i_3 \end{bmatrix} \]

\( i_0 \) is the zero sequence current when the three phases of the armature windings are connected in delta or in case of star, the neutral point is connected to earth. \( S \) is the transfer matrix

\[ S = \frac{1}{3} \begin{bmatrix}
2 \cos \theta & 2 \cos(\theta - \frac{2\pi}{3}) & 2 \cos(\theta + \frac{2\pi}{3}) \\
2 \sin \theta & 2 \sin(\theta - \frac{2\pi}{3}) & 2 \sin(\theta + \frac{2\pi}{3}) \\
1 & 1 & 1
\end{bmatrix} \quad (3.10) \]

The above relations can easily be reversed, with the real stator phase currents being derived as functions of the direct and quadrature currents as

\[ i_r = S^{-1} i_g \quad (3.11) \]

with
3.2 Dynamics of generalized electrical machines

\[
S^{-1} = \begin{bmatrix} \cos \theta & \sin \theta & 1 \\ \cos(\theta - \frac{2}{3}\pi) & \sin(\theta - \frac{2}{3}\pi) & 1 \\ \cos(\theta + \frac{2}{3}\pi) & \sin(\theta + \frac{2}{3}\pi) & 1 \end{bmatrix} \tag{3.12}
\]

Likewise the three phase voltage and M.M.F.'s of the armature windings are expressed in terms of the direct axis, quadrature axis and zero sequence components by the equations

\[
u_g = Su_r \tag{3.13}
\]

and

\[
\psi_g = S\psi_r \tag{3.14}
\]

where

\[
u_g^T = \begin{bmatrix} u_d & u_q & u_0 \end{bmatrix}
\]

\[
u_r^T = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}
\]

\[
\psi_r^T = \begin{bmatrix} \psi_1 & \psi_2 & \psi_3 \end{bmatrix}
\]

\[
\psi_g^T = \begin{bmatrix} \psi_d & \psi_q & \psi_0 \end{bmatrix}
\]

The inverse relations are

\[
u_r = S^{-1}u_g \tag{3.15}
\]

and

\[
\psi_r = S^{-1}\psi_g \tag{3.16}
\]
3.3 Equations of the induction motor

3.3.1 General equations

To consider a three phase winding induction motor, the windings d1 and q1, d2 and q2 in the
generalized machine in Fig.3.1 are equivalent because the air gap in an induction motor is
uniform and they are also derived from a symmetrical three phase rotor winding or its
equivalent in a cage rotor. The coils d and q are identical, having been obtained from
transformation of three similar stator coils which are symmetrically distributed. Hence we
have the following relations

\[ R_d = R_q = R \]
\[ R_{d1} = R_{q1} = R_1 \]
\[ R_{d2} = R_{q2} = R_2 \]
\[ L_d = L_q = L \]
\[ L_{d1} = L_{q1} = L_1 \]
\[ L_{d2} = L_{q2} = L_2 \]
\[ M_{d1} = M_{q1} = M_1 \]
\[ M_{d2} = M_{q2} = M_2 \]

(3.17)

The rotor voltages are zero since the rotor coils in an induction motor are short-circuited. So
from Eq.(3.1)-(3.7) we have the following voltage equations and the equations for the M.M.F.'s
of the induction motor

\[ u_d = R_i_d + \dot{\psi}_d + \dot{\psi}_q \]
\[ u_q = R_i_q + \dot{\psi}_q - \dot{\psi}_d \]
\[ u_0 = R_0 i_0 + \psi_0 \]
\[ 0 = R_1 i_{d1} + \psi_{d1} \]
\[ 0 = R_1 i_{q1} + \psi_{q1} \]
\[ 0 = R_2 i_{d2} + \psi_{d2} \]
\[ 0 = R_2 i_{q2} + \psi_{q2} \]
\[ \psi_d = L i_d + M_1 i_{d1} + M_2 i_{d2} \]
\[ \psi_q = L i_q + M_1 i_{q1} + M_2 i_{q2} \]
\[ \psi_{d1} = L_1 i_{d1} + M_1 i_d + M_{12} i_{d2} \]
\[ \psi_{q1} = L_1 i_q + M_1 i_{q1} + M_{12} i_{q2} \]
\[ \psi_{d2} = L_2 i_{d2} + M_2 i_d + M_{12} i_{d1} \]
\[ \psi_{q2} = L_2 i_{q2} + M_2 i_q + M_{12} i_{q1} \]
\[ \psi_0 = L_0 i_0 \]

Eliminating the rotor currents and the rotor M.M.F.'s from Eq(3.18), we get

\[ u_d = R i_d + \psi_d + \psi_q \dot{\theta} \]
\[ u_q = R i_q + \psi_q - \psi_d \dot{\theta} \]
\[ u_0 = R i_0 + \psi_0 \]  \hspace{1cm} (3.19)

\[ \psi_d + k_1 \psi_d + k_2 \psi_d = L i_d + l_1 i_d + l_2 i_d \]
\[ \psi_q + k_1 \psi_q + k_2 \psi_q = L i_q + l_1 i_q + l_2 i_q \]

where

\[ k_1 = \frac{L_1 R_2 + L_2 R_1}{R_1 R_2} \]
\[ k_2 = \frac{L_1 L_2 - M_{12}^2}{R_1 R_2} \]  \hspace{1cm} (3.20)
\[
\begin{align*}
I_1 &= \frac{L(L_1R_2 + L_2R_1) - M_1^2R_2 - M_2^2R_1}{R_1R_2} \\
I_2 &= \frac{L(L_1L_2 - M_{12}^2) + (-L_1M_{12}^2) - M_1(L_2M_1 - 2M_{12}M_2)}{R_1R_2}
\end{align*}
\]

The torque equation is

\[
T = \psi_d i_q - \psi_q i_d \tag{3.21}
\]

With the load \(T_i\) on the motor shaft, we have

\[
J_r \ddot{\Theta} + T_i = \psi_d i_q - \psi_q i_d \tag{3.22}
\]

When the voltage is given, the whole system can be determined numerically by integration. For the numerical integration, the terms with second order time derivatives should be uncoupled. With the time derivatives of the first two equations of Eq(3.19), the second order time derivatives \(\dot{\psi}_d\) and \(\dot{\psi}_q\) can be calculated as

\[
\begin{align*}
\dot{\psi}_d &= \dot{u}_d - Ri_d - \psi_q \dot{\Theta} - \psi_q \ddot{\Theta} \\
\dot{\psi}_q &= \dot{u}_q - Ri_q + \psi_d \dot{\Theta} + \psi_d \ddot{\Theta}
\end{align*}
\tag{3.23}
\]

Here \(\dot{\psi}_d\) and \(\dot{\psi}_q\) are expressed in terms of \(\dot{\Theta}\) which can be calculated from the decomposition of the equations of motion of the mechanical system without the consideration of the electrical machines. The reason for this is that the electrical variables are not coupled with other mechanical variables except the rotating angle, its speed and acceleration. With the substitution of Eq(3.23), Eq(3.19) becomes

\[
\begin{align*}
u_d &= Ri_d + \psi_d + \psi_q \dot{\Theta} \\
u_q &= Ri_q + \psi_q - \psi_d \dot{\Theta} \\
u_0 &= Ri_0 + \psi_0 
\end{align*}
\tag{3.24}
\]

\[
\begin{align*}
Li_d + l_1i_d + l_2\dddot{i}_d &= \psi_d + k_i \psi_d + k_2 \left( \dot{u}_d - Ri_d - \psi_q \dot{\Theta} - \psi_q \ddot{\Theta} \right) \\
Li_q + l_1i_q + l_2\dddot{i}_q &= \psi_q + k_i \psi_q + k_2 \left( \dot{u}_q - Ri_q + \psi_d \dot{\Theta} + \psi_d \ddot{\Theta} \right)
\end{align*}
\]

It can be noticed that the zero sequence current \(i_0\) and voltage \(u_0\) are not coupled with other
equations. So they will not be considered further.

3.3.2 Steady state equations

If the motor is connected to an infinite bus with three phase voltage in star style, then

\[ u_1 = U \cos \omega t \]

\[ u_2 = U \cos \left( \omega t - \frac{2}{3} \pi \right) \]  
\[ u_3 = U \cos \left( \omega t + \frac{2}{3} \pi \right) \]  
\[ (3.25) \]

The synchronous solution of Eq(3.19) and (3.21) is

\[ u_{d0} = U \cos \theta_0 = Ri_{d0} + \omega \psi_{q0} \]

\[ u_{q0} = U \sin \theta_0 = Ri_{q0} - \omega \psi_{d0} \]  
\[ (3.26) \]

\[ \psi_{d0} = Li_{d0} \]

\[ \psi_{q0} = Li_{q0} \]

\[ T_0 = \psi_{d0} i_{q0} - \psi_{q0} i_{d0} \]

where

\[ \theta = \theta_0 + \omega t \]

The currents can be calculated from Eq(3.26) and they are

\[ i_{d0} = \frac{R \cos \theta_0 - \omega L \sin \theta_0}{\sqrt{R^2 + \omega^2 L^2}} U \]  
\[ (3.27) \]

\[ i_{q0} = \frac{R \sin \theta_0 + \omega L \cos \theta_0}{\sqrt{R^2 + \omega^2 L^2}} U \]

From Eq(3.27) and (3.11), we have

\[ i_1 = \frac{U}{\sqrt{R^2 + \omega^2 L^2}} \cos (\omega t - \varphi) \]
\[ i_2 = \frac{U}{\sqrt{R^2 + \omega^2 L^2}} \cos \left( \omega t - \varphi - \frac{2}{3} \pi \right) \]  \hspace{1cm} (3.28)

\[ i_3 = \frac{U}{\sqrt{R^2 + \omega^2 L^2}} \cos \left( \omega t - \varphi + \frac{2}{3} \pi \right) \]

where \( \varphi \) is the power factor

\[ \varphi = \arctan \frac{\omega L}{R} \]

For synchronous rotation, the torque on the shaft is zero.

When the rotating speed lags behind the synchronous speed \( \omega \), the motor supplies a positive torque to the shaft. The slip \( s \) determines the loading condition of the motor. The rotation angle is given as

\[ \theta = (1 - s) \omega t + \theta_0 \]  \hspace{1cm} (3.29)

Substituting Eq(3.25) and (3.29) into Eq(3.11), we have

\[ u_{d0} = U \cos \alpha \]  \hspace{1cm} (3.30)

\[ u_{q0} = U \sin \alpha \]

where

\[ \alpha = \theta_0 - s \omega t \]  \hspace{1cm} (3.31)

Substituting Eq(3.30) into (3.19) and (3.21) leads to the following equations for the steady state condition of the asynchronous machine,

\[ U \cos \alpha = R i_{d0} + \psi_{d0} + \omega (1 - s) \psi_{q0} \]

\[ U \sin \alpha = R i_{q0} + \psi_{q0} - \omega (1 - s) \psi_{d0} \]  \hspace{1cm} (3.32)

\[ \psi_{d0} + k_1 \psi_{d0} + k_2 \psi_{d0} = L i_{d0} + l_1 i_{d0} + l_2 i_{d0} \]

\[ \psi_{q0} + k_1 \psi_{q0} + k_2 \psi_{q0} = L i_{q0} + l_1 i_{q0} + l_2 i_{q0} \]

\[ T_0 = \psi_{d0} i_{q0} - \psi_{q0} i_{d0} \]

To solve Eq(3.32), we use the following substitution introduced by Besseling (1957), which
simplifies the equations considerably,

\[ i_A = S_1 i_g \] (3.33)

\[ u_A = S_1 u_g \] (3.34)

\[ \psi_A = S_1 \psi_g \] (3.35)

where

\[ S_1 = \begin{bmatrix} \cos \alpha & \sin \alpha \\ \sin \alpha & -\cos \alpha \end{bmatrix} \] (3.36)

\[ i_A = \begin{bmatrix} i_a \\ i_b \end{bmatrix}^T \] (3.37)

\[ u_A = \begin{bmatrix} u_a \\ u_b \end{bmatrix}^T \] (3.38)

\[ \psi_A = \begin{bmatrix} \psi_a \\ \psi_b \end{bmatrix}^T \] (3.39)

\[ i_g = \begin{bmatrix} i_d \\ i_q \end{bmatrix}^T \] (3.40)

\[ u_g = \begin{bmatrix} u_d \\ u_q \end{bmatrix}^T \] (3.41)

\[ \psi_g = \begin{bmatrix} \psi_d \\ \psi_q \end{bmatrix}^T \] (3.42)

Because \( S_1^{-1} = S_1 \), we have the following inverse relation

\[ i_g = S_1 i_A \] (3.43)

\[ u_g = S_1 u_A \] (3.44)

\[ \psi_g = S_1 \psi_A \] (3.45)

From Eq(3.43) and (3.45), we also have

\[ i_g = S_1 i_A + S_1 i_A \] (3.46)
\[ i_g = S_1 i_A + 2 \dot{S}_1 i_A + \ddot{S}_1 i_A \]  
(3.47)

and

\[ \psi_g = S_1 \psi_A + \dot{S}_1 \psi_A \]  
(3.48)

\[ \psi_g = S_1 \psi_A + 2 \dot{S}_1 \psi_A + \ddot{S}_1 \psi_A \]  
(3.49)

where

\[ \dot{S}_1 = -s \omega \begin{bmatrix} -\sin \alpha \cos \alpha \\ \cos \alpha \sin \alpha \end{bmatrix} \]  
(3.50)

and

\[ \ddot{S}_1 = -s^2 \omega \begin{bmatrix} \cos \alpha & \sin \alpha \\ \sin \alpha & -\cos \alpha \end{bmatrix} \]  
(3.51)

Substituting Eq.(3.43)-(3.49) into Eq.(3.32), we get the following equations,

\[ u_a = R i_a + \dot{\psi}_a - \omega \psi_b \]

\[ u_b = R i_b + \dot{\psi}_b + \omega \psi_a \]  
(3.52)

\[ (1 - k_2 s^2 \omega^2) \psi_a + k_1 \dot{\psi}_a + k_2 \ddot{\psi}_a - k_1 s \omega \psi_b - 2 k_2 s \omega \psi_b = \]

\[ (L - l_2 s^2 \omega^2) i_a + l_1 \dot{i}_a + l_2 \ddot{i}_a - l_1 s \omega i_b - 2 l_2 s \omega i_b \]

\[ (1 - k_2 s^2 \omega^2) \psi_b + k_1 \dot{\psi}_b + k_2 \ddot{\psi}_b + k_1 s \omega \psi_a + 2 k_2 s \omega \psi_a = \]

\[ (L - l_2 s^2 \omega^2) i_b + l_1 \dot{i}_b + l_2 \ddot{i}_b + l_1 s \omega i_a + 2 l_2 s \omega i_a \]

\[ T = \psi_b i_a - \psi_a i_b \]

When the motor is connected with an infinite bus with three phase voltages, we have

\[ u_{a0} = U \left[ \cos^2 \alpha + \sin^2 \alpha \right] = U \]

\[ u_{b0} = U \left[ \cos \alpha \sin \alpha - \sin \alpha \cos \alpha \right] = 0 \]
So the steady state solution of the motor equations is

\[ U = Ri_{a0} - \omega \psi_{b0} \]

\[ 0 = Ri_{b0} + \omega \psi_{a0} \]

\[ (1 - k_2 s^2 \omega^2) \psi_{a0} - k_1 s \omega \psi_{b0} = (L - l_2 s^2 \omega^2) i_{a0} - l_1 s \omega i_{b0} \]  \hspace{1cm} (3.53)

\[ (1 - k_2 s^2 \omega^2) \psi_{b0} + k_1 s \omega \psi_{a0} = (L - l_2 s^2 \omega^2) i_{b0} + l_1 s \omega i_{a0} \]

\[ T_0 = \psi_{b0} i_{a0} - \psi_{a0} i_{b0} \]

From Eq (3.53) we get the following solution

\[ \psi_{a0} = \left( \frac{-R}{\omega} \right) i_{b0} \]

\[ \psi_{b0} = \frac{(Ri_{a0} - U)}{\omega} \]  \hspace{1cm} (3.54)

\[ i_{a0} = \frac{U (k_1^2 Rs^2 \omega^2 + k_1 s \omega^2 B + (k_2 s^2 \omega^2 - 1) (k_2 Rs^2 \omega^2 + l_1 s \omega^2 - R))}{A} \]

\[ i_{b0} = \frac{-U \omega (k_1 l_1 s^2 \omega^2 - B (k_2 s^2 \omega^2 - 1))}{A} \]

\[ T_0 = \psi_{b0} i_{a0} - \psi_{a0} i_{b0} \]

where

\[ A = k_1^2 R^2 s^2 \omega^2 + 2k_1 R s \omega^2 B + k_2^2 R^2 s^4 \omega^4 + 2k_2 R s^2 \omega^2 (l_1 s \omega^2 - R) + \]

\[ + L^2 \omega^2 - 2Ll_2 s^2 \omega^4 + l_1^2 s^2 \omega^4 - 2l_1 R s \omega^2 + l_2^2 s^4 \omega^6 + R^2 \]  \hspace{1cm} (3.55)

\[ B = L - l_2 s^2 \omega^2 \]

3.3.3 Equations for small oscillations

When a motor operating under steady state conditions is subjected to small disturbances, the equations can be linearized. The linearized equations can be used for studying steady-state
stability, or for calculating the magnitude of small oscillations which may be superimposed on a condition of steady operation.

For small variations from the steady state condition, we use $\varepsilon(t)$ to represent the change of the angle $\theta$, so $\theta$ is expressed as,

$$\theta = (1 - s) \omega t + \varepsilon(t) + \theta_0$$ (3.56)

The other quantities are

$$u_d = u_{d0} + \Delta u_d$$

$$u_q = u_{q0} + \Delta u_q$$

$$i_d = i_{d0} + \Delta i_d$$

$$i_q = i_{q0} + \Delta i_q$$ (3.57)

$$\psi_d = \psi_{d0} + \Delta \psi_d$$

$$\psi_q = \psi_{q0} + \Delta \psi_q$$

$$T = T_0 + \Delta T$$

For stability analysis, Eq(3.19) and (3.21) can be linearized as

$$\Delta u_d = R\Delta i_d + \Delta \psi_d + (1 - s) \omega \Delta \psi_q + \psi_{q0} \dot{\varepsilon}$$

$$\Delta u_q = R\Delta i_q + \Delta \psi_q - (1 - s) \omega \Delta \psi_d - \psi_{d0} \dot{\varepsilon}$$

$$\Delta \psi_d + k_1 \Delta \psi_d + k_2 \Delta \dot{\psi}_d = L\Delta i_d + l_1 \Delta \ddot{i}_d + l_2 \Delta \dot{i}_d$$ (3.58)

$$\Delta \psi_q + k_1 \Delta \psi_q + k_2 \Delta \dot{\psi}_q = L\Delta i_q + l_1 \Delta \ddot{i}_q + l_2 \Delta \dot{i}_q$$

$$\Delta T = \Delta \psi_d i_{q0} + \psi_{d0} \Delta i_q - \Delta \psi_q i_{d0} - \psi_{q0} \Delta i_d$$

For the variations from the steady state condition, we use the substitution similar to Eq(3.33)-(3.35)

$$i_A = S_2 i_g$$ (3.59)

$$u_A = S_2 u_g$$ (3.60)
\[ \psi_A = S_2\psi_g \]  

but here

\[ S_2 = \begin{bmatrix} \cos \alpha_1 & \sin \alpha_1 \\ \sin \alpha_1 & -\cos \alpha_1 \end{bmatrix} \]  

\[ \dot{S}_2 = (\dot{\epsilon} - s\omega) \begin{bmatrix} -\sin \alpha_1 & \cos \alpha_1 \\ \cos \alpha_1 & \sin \alpha_1 \end{bmatrix} \]  

\[ S_2 = \dot{\epsilon} \begin{bmatrix} -\sin \alpha_1 & \cos \alpha_1 \\ \cos \alpha_1 & \sin \alpha_1 \end{bmatrix} - (\dot{\epsilon} - s\omega)^2 S_2 \]  

and

\[ \alpha_1 = \theta_0 + \epsilon - s\omega t \]  

The inverse relations are

\[ i_g = S_2 i_A \]  

\[ u_g = S_2 u_A \]  

\[ \psi_g = S_2 \psi_A \]  

and

\[ \dot{i}_g = S_2 \dot{i}_A + \dot{S}_2 i_A \]  

\[ \ddot{i}_g = S_2 \ddot{i}_A + 2S_2 \dot{i}_A + \ddot{S}_2 i_A \]  

\[ \dot{\psi}_g = S_2 \dot{\psi}_A + \dot{S}_2 \psi_A \]  

\[ \ddot{\psi}_g = S_2 \ddot{\psi}_A + 2S_2 \dot{\psi}_A + \ddot{S}_2 \psi_A \]  

Substituting Eq(3.65)-(3.35) into Eq(3.19) and (3.21), we have the following equations
\[ u_a = R_i_a + \psi_a - \omega \psi_b \]

\[ u_b = R_i_b + \psi_b + \omega \psi_a \]

\[
\begin{align*}
[1 - k_2 (\ddot{e} - s\omega)^2] \psi_a &+ k_1 \psi_a + k_2 \ddot{\psi}_a + [k_1 (\ddot{e} - s\omega) + k_2 \ddot{e}] \psi_b + 2k_2 (\ddot{e} - s\omega) \psi_b = \\
(L - l_2 (\ddot{e} - s\omega)^2) i_a + l_1 i_a + l_2 i_a &+ [l_1 (\ddot{e} - s\omega) + l_2 \ddot{e}] i_b + 2l_2 (\ddot{e} - s\omega) i_b \\
(3.72)

[1 - k_2 (\ddot{e} - s\omega)^2] \psi_b + k_1 \psi_b + k_2 \ddot{\psi}_b - [k_1 (\ddot{e} - s\omega) + k_2 \ddot{e}] \psi_a - 2k_2 (\ddot{e} - s\omega) \psi_a = \\
[L - l_2 (\ddot{e} - s\omega)^2] i_b + l_1 i_b + l_2 i_b &- [l_1 (\ddot{e} - s\omega) + l_2 \ddot{e}] i_a - 2l_2 (\ddot{e} - s\omega) i_a
\end{align*}
\]

\[ T = \psi_b i_a - \psi_a i_b \]

By using Eq (3.57) and neglecting the non-linear terms, Eq (3.72) falls into the steady state equations and the following equations for small perturbation

\[ \Delta u_a = R\Delta i_a + \Delta \psi_a - \omega \Delta \psi_b \]

\[ \Delta u_b = R\Delta i_b + \Delta \psi_b + \omega \Delta \psi_b \]

\[ (1 - k_2 s^2 \omega^2) \Delta \psi_a + k_1 \Delta \psi_a + k_2 \Delta \psi_a - k_1 s\omega \Delta \psi_b - 2k_2 s\omega \Delta \psi_b = \\
(L - l_2 s^2 \omega^2) \Delta i_a + l_1 \Delta i_a + l_2 \Delta i_a - l_1 s\omega \Delta i_b - 2l_2 s\omega \Delta i_b \] +

\[
\begin{align*}
[-2k_2 s\omega \psi_{a0} - k_1 \psi_{b0} - 2k_2 \psi_{b0} + 2l_2 s\omega i_{a0} + l_1 i_{b0} + 2l_2 \dot{i}_{b0}] \ddot{e} + [-k_2 \psi_{b0} + l_2 i_{b0}] \dddot{e} \\
(1 - k_2 s^2 \omega^2) \Delta \psi_b + k_1 \Delta \psi_b + k_2 \Delta \psi_b + k_1 s\omega \Delta \psi_a + 2k_2 s\omega \Delta \psi_a = \\
(L - l_2 s^2 \omega^2) \Delta i_b + l_1 \Delta i_b + l_2 \Delta i_b + l_1 s\omega \Delta i_a + 2l_2 s\omega \Delta i_a + \\
[-2k_2 s\omega \psi_{b0} + k_1 \psi_{a0} + 2k_2 \psi_{a0} + 2l_2 s\omega i_{a0} - l_1 i_{a0} - 2l_2 \dot{i}_{a0}] \ddot{e} + [k_2 \psi_{a0} - l_2 i_{a0}] \dddot{e} \\
\Delta T = \psi_{b0} \Delta i_a - \psi_{a0} \Delta i_b + i_{a0} \Delta \psi_b - i_{b0} \Delta \psi_a
\end{align*}
\]

where

\[ u_a = u_{a0} + \Delta u_a, \quad u_b = u_{b0} + \Delta u_b \]
3.4 Dynamic equations of alternators

3.4.1 General equations

Generally the rotor of a synchronous alternator carries a field winding in the direct axis and squirrel cage windings in the direct and quadrature axis. So winding q2 of the generalized machine is missing and d2 represents the field winding carrying the direct current. If a machine has more damper windings, it can also be approximated with this generalized machine [Adkins, 1975]. The winding d1 and q1 are short circuited. From Eq(3.1) and (3.7), the equations for the synchronous alternator may be written as

\[ u_d = R_i_d + \psi_d + \theta \frac{\dot{\psi}_d}{\dot{\theta}} \]

\[ u_q = R_i_q + \psi_q - \psi_q \dot{\psi}_q \]

\[ u_0 = R_0 i_0 + \psi_0 \quad \text{(3.74)} \]

\[ u_f = R_f i_f + \psi_f \]

\[ 0 = R_{d1} i_{d1} + \psi_{d1} \]

\[ 0 = R_{q1} i_{q1} + \psi_{q1} \]

\[ \psi_d = L_d i_d + M_{d1} i_{d1} + M_{q1} i_q \]

\[ \psi_q = L_q i_q + M_{q1} i_{q1} \]

\[ \psi_f = L_f i_f + M_f i_d + M_{fd1} i_{d1} \]

\[ \psi_{d1} = L_{d1} i_{d1} + M_{d1} i_d + M_{fd1} i_f \]

\[ \psi_{q1} = L_{q1} i_{q1} + M_{q1} i_q \]
By eliminating the damper currents and the M.M.F.'s of the rotor windings from Eq(3.74), we have the following equations,

\[ u_d = R_i d + \psi_d + \psi_q \dot{\theta} \]
\[ u_q = R_i q + \psi_q - \psi_d \dot{\theta} \]
\[ u_0 = R_i 0 + \psi_0 \]

\[ \psi_d + m_1 \dot{\psi}_d + m_2 \ddot{\psi}_d = L_d \dot{i}_d + n_1 \dot{i}_d + n_2 \dot{i}_d + \frac{M_f}{R_f} u_f + k_u \ddot{u}_f \]
\[ \psi_q + \frac{L_a q}{R_a q} \dot{\psi}_q = L_q q + \frac{L_a q}{R_a q} \dot{\psi}_q \]

where

\[ m_1 = \frac{L_d 1 R_f + L_f R_d 1}{R_d 1 R_f} \]
\[ m_2 = \frac{L_d 1 L_f - M_{fd} 1}{R_d 1 R_f} \]
\[ n_1 = \frac{L_d (L_d 1 R_f + L_f R_d 1) - M^2_{fd} 1 R_f - M_{fd} 1 R_d 1}{R_d 1 R_f} \]
\[ n_2 = \frac{L_d (L_d 1 L_f - M^2_{fd} 1) - L_d 1 M^2_f - M_d 1 (L_f M_{fd} 1 - 2 M_f M_{fd} 1)}{R_d 1 R_f} \]
\[ k_u = \frac{(L_d 1 M_f - M_d 1 M_{fd} 1)}{R_d 1 R_f} \]

With the driving torque \( T_d \) on the shaft, we have

\[ J \ddot{\theta} = T_d - (\psi_d \dot{i}_q - \psi_q \dot{i}_d) \]

(3.77)

For numerical integration, Eq(3.75), similar to Eq(3.17), can be written as

\[ u_d = R_i d + \psi_d + \psi_q \dot{\theta} \]
\[ u_q = R_i q + \psi_q - \psi_q \dot{\theta} \]

\[ u_0 = R_i 0 + \psi_0 \quad (3.78) \]

\[ \psi_d + m_1 \dot{\psi}_d + m_2 \left( \ddot{u}_d - Ri q - \psi_q \dot{\theta} - \psi_q \ddot{\theta} \right) = L_i d + n_1 i_d + n_2 i_d + \frac{M_f}{R_f} u_f + k u f \]

\[ \psi_q + \frac{L_{q1}}{R_{q1}} \dot{\psi}_q = L_q i_q + \frac{L_q L_{q1} - M^2}{R_{q1}} \dot{i}_q \]

### 3.4.2 Steady state equations

When the alternator is connected to an infinite bus with three phase voltages as

\[ u_1 = U \cos \omega t \]

\[ u_2 = U \cos \left( \omega t - \frac{2}{3} \pi \right) \quad (3.79) \]

\[ u_3 = U \cos \left( \omega t + \frac{2}{3} \pi \right) \quad (3.80) \]

and if the values of \( u_f \) and the driving torque are kept constant, the steady state solution for the alternator is

\[ u_{d0} = U \cos \theta_0 = R_i d_0 + \omega \psi_{q0} \]

\[ u_{q0} = U \sin \theta_0 = R_i q_0 - \omega \psi_{d0} \]

\[ u_{f0} = R_i f_0 \quad (3.81) \]

\[ \psi_{d0} = L_d i_{d0} + R_i f_{d0} \]

\[ \psi_{q0} = L_q i_{q0} \]

\[ T_{d0} = \psi_{d0} i_{q0} - \psi_{q0} i_{d0} \]

where

\[ \theta = \theta_0 + \omega t \quad (3.82) \]
3.4.3 Equations for small oscillations

For small variations from the steady state, all quantities can be represented as the steady state value and an additional small time dependent value

\[ \theta = \omega t + \varepsilon + \theta_0 \]

\[ u_d = u_{d0} + \Delta u_d \]

\[ i_d = i_{d0} + \Delta i_d \]

\[ i_q = i_{q0} + \Delta i_q \]

(3.83)

\[ \psi_d = \psi_{d0} + \Delta \psi_d \]

\[ \psi_q = \psi_{q0} + \Delta \psi_q \]

\[ T = T_0 + \Delta T \]

\[ u_f = u_{f0} + \Delta u_f \]

Substituting Eq(3.83) into Eq(3.75) and (3.77) and omitting the high order terms, we have

\[ \Delta u_d = R_d \Delta i_d + \Delta \psi_d + \omega \Delta \psi_q + \psi_{q0} \dot{\varepsilon} \]

\[ \Delta u_q = R_q \Delta i_q + \Delta \psi_q - \omega \Delta \psi_d - \psi_{d0} \dot{\varepsilon} \]  

(3.84)

\[ \Delta \psi_d + m_1 \Delta \dot{\psi}_d + m_2 \Delta \ddot{\psi}_d = L_d \Delta i_d + n_1 \Delta i_d + n_2 \Delta \dot{i}_d + \frac{M_f}{R_f} \Delta u_f + k_u \Delta \dot{u}_f \]

\[ \Delta \psi_q + \frac{L_{q1}}{R_{q1}} \Delta \psi_q = L_q \Delta i_q + \frac{L_q L_{q1} - M_p^2}{R_{q1}} \Delta i_q \]

\[ J \Delta \ddot{\theta} = \Delta T_d - \left( \psi_{d0} \Delta i_d + i_{q0} \Delta \psi_d - \left( \psi_{q0} \Delta i_d + i_{d0} \Delta \psi_q \right) \right) \]
3.5 Dynamics of d.c. motors

3.5.1 Separately excited d.c. motors

The d.c. motors are always started with reduced armature coils and the applied voltage is increased as the back e.m.f. builds up with speed. For separately excited d.c. motors, the dynamic equations can be expressed in terms of the field current $i_f$,

$$ u = R_a i_a + L_a i_a + M_i \dot{\theta} $$  \hspace{1cm} (3.85)

$$ J \ddot{\theta} = M_i i_f - T_l $$

where $T_l$ is the load torque.

For stability analysis and for perturbation control, the non-linear system should be linearized. For small changes from the steady state or from the nominal trajectory in a control process, we define,

$$ u = u_0 + \Delta u $$

$$ \theta = \theta_0 + \Delta \theta $$  \hspace{1cm} (3.86)

$$ i_a = i_{a0} + \Delta i_a $$

$$ i_f = i_{f0} + \Delta i_f $$

If the supplied voltage $u$ is controlled and the field current $i_f$ is constant, the linearized equations of Eq(3.85) are

$$ \Delta u = R_a \Delta i_a + L_a \Delta i_a + M_i \Delta \dot{\theta} $$  \hspace{1cm} (3.87)

$$ J \Delta \ddot{\theta} = M_i \Delta i_f - \Delta T_l $$

If the field current $i_f$ is controlled, we have

$$ \Delta u = R_a \Delta i_a + L_a \Delta i_a + M_i \dot{\theta}_0 \Delta i_f + M_i \dot{f}_0 \Delta \dot{\theta} $$  \hspace{1cm} (3.88)

$$ J \Delta \ddot{\theta} = M_i \dot{a}_0 \Delta i_f + M_i \dot{f}_0 \Delta i_a - \Delta T_l $$

For d.c. motors used in position controlled systems, a simplified model is normally used
[Cliffet and Chirouze, 1983], [Luh, 1983], [Jankowski and Brussel, 1993]. In this model, just the dynamics of the controlled circuit is considered and the uncontrolled circuit is considered constant. For an armature controlled d.c. motor, the field voltage and current is considered constant. The dynamic equations are

\[ u = Ri + Li + k_v \dot{\theta} \]  
(3.89)

\[ T = k_i i \]  
(3.90)

\[ J \ddot{\theta} + k_v \dot{\theta} = T \]  
(3.91)

where \( k_i \) is the torque constant and \( k_v \) is the voltage constant of the motor.

Differentiating Eq(3.90) with respect to time we get

\[ \dot{T} = k_i \dot{i}. \]  
(3.92)

Substituting Eq(3.90) and (3.92) into Eq(3.89), we have the relationship between the controlled voltage and the torque,

\[ k_i u = RT + LT \dot{T} + k_v k_i \dot{\theta}. \]  
(3.93)

Eq(3.91) and (3.93) are used for dynamic analysis of the controlled motor.

In the inverse dynamics, the speed profile of the robot is known and the problem is to determine the controlled voltage. From the known speed profile and the dynamic equations, one can calculate the torque and its rate. So the desired voltage is determined by Eq(3.93) as

\[ u = \frac{1}{k_i} \left( RT + LT \dot{T} + k_v k_i \dot{\theta} \right) \]  
(3.94)

If the field current is controlled, the dynamic equations have the same form as Eq(3.89)-(3.91) with \( u \) representing the field voltage and \( i \) the field current.

3.5.2 Series-excited d.c. machines

In the series-excited d.c. machine, the field coils are in series with the armature coils. The field current \( i_f \) and the armature current \( i_a \) are the same, say \( i \). So we have the following equations for a series motor,

\[ u = Ri + Li + Mi \dot{\theta} \]  
(3.95)
\[ J\ddot{\theta} = Mi^2 - T_i \]

With given voltage and load torque, these equations can be solved numerically for the current \( i \) and the rotor angle \( \theta \).

With the definitions of Eq(3.86), the linearized equations of Eq(3.95) are

\[ \Delta u = R\Delta i + L\Delta \dot{i} + M\dot{\theta}_0 \Delta i + Mi_0 \Delta \dot{\theta} \]  
\[ J\Delta \ddot{\theta} = 2Mi_0 \Delta i - \Delta T_i \]  

(3.96)

In the inverse dynamics, with the speed, acceleration and jerk of the rotor angle and the torque and its rate prescribed, from Eq(3.95), we have the following equations for the supplied voltage \( u \) and the current \( i \),

\[ i = \sqrt{\frac{J\ddot{\theta} + T_i}{M}} \]  
\[ u = (R + M\dot{\theta}) \frac{\sqrt{J\ddot{\theta} + T_i}}{M} + \frac{L}{2\sqrt{M(J\ddot{\theta} + T_i)}} \]  

(3.97)

Here, too, the speed, acceleration and jerk of the rotor angle, and load torque and its rate should be known for the inverse dynamics.

3.6 Controlled electric-mechanical equations

In the finite element representation of mechanisms and robots, the relative rotation between bodies are expressed as deformations of hinge elements. So an electrical machine will also be represented by a hinge element. In the finite element representation of an electrical machine, the expressions for element deformations are the same as for a normal hinge element without an electrical machine attached to it. But more dynamic equations are given as the number of degrees of freedom is increased with the electrical variables added to the system.

To implement the electrical machines in the SPACAR software system, first for every electrical machine and the gear box, we specify a hinge element to represent them. If the rotating angle of the rotor of the electrical machine is different from that of the hinge, the ratio of the rotating speed between the electrical machine and the hinge should be specified. Then the equations of motion for the mechanical system without electrical machines are constructed as
\[ Mq' = f(q, q', t) \]  

(3.98)

where \( q \) is the vector of degrees of freedom of the mechanical system. The torque supplied by or supplying to the electrical machines should be included. The rotor inertial moment of the electrical machines should also be included in Eq(3.98). To do so, an inertial matrix \( J_r \) is specified at the rotating node which is connected with the rotor of the electrical machine. With \( R \) representing the transformation from the inertial frame to the body fixed frame of the rotor, the inertial matrix \( J_r \) is given as [Wittenburg, 1977]

\[ J_r = R^T J_b R \]  

(3.99)

where \( J_b \) is the inertial matrix based on the body fixed frame

\[ J_b = \begin{bmatrix} m^2 J & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \]  

(3.100)

Here \( J \) is the rotor inertial moment and \( m \) the ratio between the rotating speed of the rotor and that of the deformation of the hinge element representing the electrical machine.

From Eqs(3.7)-(3.8), Eq(3.24), Eq(3.78), Eq(3.85) and Eq(3.95), it can be noticed that the electrical variables are not coupled with other mechanical variables except for the rotating speed and acceleration of the rotor. So Eq(3.98) can be uncoupled to calculate the acceleration of the rotors. Then with the dynamical equations of the electrical machines, the whole state equations can be constructed and the system can be integrated for simulation or linearized for stability analysis. An application example of the theory developed in this chapter can be found in section 6.5.2.
Chapter 4

Rigid and Flexible Links and Flexible Joints

4.1 Introduction

As robot manipulators for higher accuracy, higher speed, higher acceleration and higher ratio of payload mass over moved total mass are required in industry, more and more attention has been paid to the flexibility of robot structures. When speed and acceleration increase, there is a tendency to use light-weight structures with inevitably more flexibility in links and joints. For the positioning accuracy, this flexibility should be taken into account.

A specific finite element theory suitable for the analysis of mechanisms and manipulator robots was initiated and developed by van der Werff [112-113], Besseling [9-13], and Jonker [56-61] and its application is found in the analysis and design of many kinds of mechanisms and robots [15-16][37-40][62-63]. This description starts out from the expressions for the deformation modes of the elements as non-linear functions of the coordinates of the element-nodes. This is the main distinguishing point of the approach, which makes it very convenient to analyze and simulate mechanisms and manipulators with a number of flexible links. This finite element description may be looked upon as an algebraic analogue to the continuum description of deformation, with its two, at least piecewise, continuous fields for the displacement vector and the strain tensor. With a proper definition of the deformation modes of the elements the deformations of a flexible body can easily be taken into account or ignored. Systems with flexible bodies are handled by allowing non-zero deformation parameters for these bodies and specifying constitutive equations relating the deformation parameters and the dual stress parameters. Also relative motions between bodies can be described as deformations, e.g., the relative rotation of two beam elements can be described by the deformation of a hinge element which connects these two beam elements. With these characteristics of the finite element method, control actuators can easily be implemented to perform simulation of robots
and manipulators.

Instead of imposing constraint equations for the connection between bodies, permanent contact between bodies or elements is achieved by letting the elements have common degrees of freedom at nodal points. With the finite element description of the mechanical system, transfer functions are constructed and calculated and equations of motion are established.

As shown by Good and et al. (1985), from the two principal sources of mechanical flexibility in a robot, bending and torsion of the arm members and flexibility in the joints and mechanical linkages connecting the drive systems to the arm members, the motions produced by joint and linkage flexibility were dominating. In this chapter, both the flexible links and flexible joints will be introduced using a finite element description. Also the method dealing with some special connections of bodies, e.g. the mostly studied prismatic joints, will be given. Moreover a flexible joint model composed of a driver and a transmission box will be studied. The finite element for flexible links can directly be employed for representing rigid links, however, it takes more computing time. So we have introduced a rigid beam element to represent rigid links.

4.2 Finite element representation for mechanisms and robots

Spatial manipulators and mechanisms composed of links and joints can be modelled by finite elements. The links can be represented by spatial beam elements and the joints by spatial hinge elements. There are nodal points attached to each end of an element. At a node we have a set of coordinates describing the translational or rotational motion of the element end point. The position of an element in space is completely determined by its nodal point coordinates \( \mathbf{x}^k \). The components of \( \mathbf{x}^k \) are determined by the type of the element. For example, in case of a spatial hinge element, \( \mathbf{x}^k \) may be given in Euler parameters as

\[
\mathbf{x}^k = [\lambda_0^T, \lambda_4^T] = [\lambda_0^p, \lambda_1^p, \lambda_2^p, \lambda_3^p, \lambda_0^q, \lambda_1^q, \lambda_2^q, \lambda_3^q]
\] (4.1)

with the following constraint equation at \( p \) and \( q \)

\[
\lambda_0^2 + \lambda_1^2 + \lambda_2^2 + \lambda_3^2 = 1
\] (4.2)

For a spatial beam element, we choose

\[
\mathbf{x}^k = [x^p, y^p, z^p, \lambda_0^p, \lambda_1^p, \lambda_2^p, \lambda_3^p, x^q, y^q, z^q, \lambda_0^q, \lambda_1^q, \lambda_2^q, \lambda_3^q]
\] (4.3)

again with constraint equations Eq(4.2).

The set of all the nodal coordinates construct the nodal coordinate vector \( \mathbf{x} \) for the whole system. The coordinate spaces of the individual element, \( X^k \), can be regarded as sub-spaces of
the system coordinate space, $X$, that is

$$X = \sum_k X^k$$

(4.4)

For every element, a set of deformation parameters $e^k$, ($e^k \in E^k$) are defined as non-linear functions of its nodal coordinates,

$$e^k = D^k (x^k)$$

(4.5)

There are two principles to define the deformation functions for an element:

- The number of deformation parameters for an element should be equal to the number of its nodal coordinates minus the number of the degrees of freedom of the element as a rigid body.

- When the element is considered to be rigid, the deformations $e_i^k = 0$ should exactly give the constraint conditions of the element as a rigid body.

These deformation functions can be used to describe real deformations of the flexible element, or they can be used to represent relative motion between links, e.g., the relative rotation between two links are described by the deformation of a hinge element.

A deformation space of the whole system is defined as the direct sum of the deformation spaces of all the elements

$$E = \bigoplus_k E^k$$

(4.6)

The deformation parameters for the whole system are

$$e = D(x)$$

(4.7)

Both of the system coordinate space and the deformation space can be divided into three sub-spaces as

$$X = X^o \oplus X^c \oplus X^m$$

(4.8)

and

$$E = E^o \oplus E^m \oplus E^c.$$ 

(4.9)

where the superscript $o$ indicates a fixed coordinate or a zero deformation, $m$ an independent coordinate or deformation (generalized coordinates) and $c$ a dependent coordinate or deformation. So the independent generalized coordinates of the system $q$ are
\[ q^T = [x^m T, e^m T] \] (4.10)

In case of dynamic simulation of a manipulator, the independent generalized coordinates are normally given as element deformations, so we have \( q^m = e^m \). On the other hand, in case of inverse kinematics, they are given by the nodal coordinates if the manipulator is required to track a prescribed trajectory. So we have \( q = x^m \).

With the generalized coordinates chosen, the whole system is determined by the so-called geometric transfer functions

\[ x = F^x(q) \] (4.11)

and

\[ e = F^e(q) \] (4.12)

Equations (4.11) and (4.12) can not be solved directly, but the partial derivatives of \( F^x \) and \( F^e \) with respect to \( q \), the first order geometric transfer functions, can be calculated. Then \( x \) and \( e \) can be determined numerically by an iterative procedure.

The derivatives of these geometric transfer functions with respect to \( q \) are calculated by using the deformation functions. Substituting equation (4.11) into (4.7), we have

\[ e = D(F^x(q)) \] (4.13)

By splitting the coordinate space \( \mathbf{X} \) and the deformation space \( \mathbf{E} \) into sub-spaces as shown in equation (4.8) and (4.9), we can calculate the first order geometric transfer functions as

\[ F^{eo}_{\cdot, q} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \] (4.14)

\[ F^{em}_{\cdot, q} = \begin{bmatrix} 0 \\ I \end{bmatrix} \] (4.15)

\[ F^{xo}_{\cdot, q} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \] (4.16)

\[ F^{xm}_{\cdot, q} = \begin{bmatrix} I \\ 0 \end{bmatrix} \] (4.17)

\[ F^{xc}_{\cdot, q} = \begin{bmatrix} D^{o}_{\cdot, xc} \end{bmatrix}^{-1} \begin{bmatrix} -D^{o}_{\cdot, xm} \\ 0 \end{bmatrix} \begin{bmatrix} D^{m}_{\cdot, xc} \\ -D^{m}_{\cdot, xm} \end{bmatrix} \] (4.18)

\[ F^{ee}_{\cdot, q} = D^{c}_{\cdot, xc} F^{xc}_{\cdot, q} + D^{c}_{\cdot, xm} F^{xm}_{\cdot, q} \] (4.19)
where $D_{,x}$ indicates partial differentiation of the deformation functions with respect to the nodal coordinates $x$. These partial differentiations of the deformation functions can be calculated element by element and then constructed for the whole system.

With the calculated first order geometric transfer functions, the geometric transfer functions can be calculated numerically. A similar method may be used to calculate the second order geometric transfer functions. The detailed description is given in reference [Jonker, 1988, 1990].

The unknown velocities and accelerations, that is the velocities and accelerations of the dependent coordinates $x^e$ and deformations $e^e$, can be calculated from the velocities and accelerations of the independent generalized coordinates as

$$\dot{x}^e = F_{,q}^{x} \dot{q}$$ (4.20)

$$\ddot{e}^e = F_{,q}^{e} \ddot{q}$$ (4.21)

and

$$\ddot{x}^e = F_{,q}^{x} \ddot{q} - \left[ D_{,xx}^{0} \right]^{-1} \left[ \begin{array}{c} (D_{,xx}^{0} \dddot{x}) \dot{x} \\ (D_{,xx}^{m} \dddot{x}) \dot{x} \end{array} \right]$$ (4.22)

$$\dddot{e}^e = F_{,q}^{e} \dddot{q} - D_{,xx}^{e} \left[ D_{,xx}^{0} \right]^{-1} \left[ \begin{array}{c} (D_{,xx}^{0} \dddot{x}) \dot{x} \\ (D_{,xx}^{m} \dddot{x}) \dot{x} \end{array} \right] + (D_{,xx}^{e} \dddot{x}) \dot{x}$$ (4.23)

The above analysis gives the kinematic relationship of the whole system with respect to the independent variables (the degrees of freedom) of the system.

The dual of the deformation vector $e^k$ for an element is defined as the vector of generalized stress, $\sigma^k$, of the element. When a linear viscous-elastic deformation is considered, the corresponding vector of the element may be written as

$$\sigma^k = S^k e^k + S_d^k \dot{e}$$ (4.24)

where $S^k$ and $S_d^k$ are the stiffness matrix and the viscous damping matrix.

The equations of motion of the mechanical system are derived with the aid of the principle of virtual power. The inertial properties of a mechanical system are represented by a mass matrix for every nodal point and a mass matrix ($M^k$), a velocity dependent force vectors ($f_{in}^k$) and a velocity dependent stress vector ($\sigma_{in}^k$) for every element. Lumped masses are considered as masses at nodal points. For distributed masses, a consistent mass formulation method is used
to get the mass matrix of an element.

With the contribution of all the nodal points and elements, the global mass matrix $M$, the global nodal force vector $f^e$, the global velocity related forces $f^{in}$ and the global stress vector $\sigma$ dual to $\varepsilon$ can be constructed. According to the principle of virtual power we then obtain

\[
(\delta x^T, \delta \dot{e}^T) \left[ M(x, e) \begin{bmatrix} \ddot{x} \\ \ddot{\varepsilon} \end{bmatrix} - f^{in} + \begin{bmatrix} -f^e \\ \sigma \end{bmatrix} \right] = 0
\] (4.25)

for all the kinematically admissible virtual velocities $(\delta x^T, \delta \dot{e}^T)$. By using the transfer functions $F^x, F^e$ and their derivatives, the acceleration $(\dot{x}^T, \dot{\varepsilon}^T)$ can be expressed as functions of $\dot{q} = (\dot{x}^mT, \dot{\varepsilon}^mT)^T$ and $\ddot{q} = (\ddot{x}^mT, \ddot{\varepsilon}^mT)^T$. This reduces the equations of motions to

\[
\begin{bmatrix}
(F_{,qq})^T M F_{,qq} & (F_{,qq})^T M F_{,qr} \\
(F_{,qr})^T M F_{,qq} & (F_{,qr})^T M F_{,rr} + f
\end{bmatrix} \dot{q}^d = (F_{,qq})^T [-M(F_{,qq} \dot{q} \dot{q}^T + F_{,qr} \dot{q}^r) + f]
\] (4.26)

where

\[
f = [F^x, -\sigma^T]^T + f^{in}
\]

$q^d$ is the vector of independent coordinates and $q^r$ is the vector of coordinates describing rheonomic constraints with

\[
q = \begin{bmatrix} q^r^T \\ q^d^T \end{bmatrix}^T.
\]

The equations of motion, Eq(4.26), can be solved numerically.

### 4.3 Flexible links

The flexible links can be represented by beam elements. We have planar beam elements for planar problems and spatial beam elements for spatial problems. Here detailed information of a spatial beam element is given.
A typical spatial beam element p-q is shown in Fig. 4.1. The configuration of the element is fully determined by the position vectors \( x^p \) and \( x^q \) of the end nodes p and q, and the angular orientation of the orthogonal triads \((n_x, n_y, n_z)\) rigidly attached to the end points. In the undeflected state the triads at both ends coincide with the axis p-q and the principal axes of the cross section. The rotation part of a flexible beam element is determined by the rotation matrices whose components are expressed in terms of Euler parameters \( \lambda = [\lambda_0, \lambda_1, \lambda_2, \lambda_3]^T \). With these parameters the vector of nodal coordinates for the spatial beam element k is as Eq (4.3).

Six deformation parameters are expressed in these nodal coordinates, elongation:

\[
e^k_1 = l^k - l^k_0 + \left[ \frac{1}{30 l^k_0} \right] [2 (e_3^k)^2 + 2 (e_4^k)^2 + 2 (e_5^k)^2 + \frac{1}{2} (e_3^k e_4^k + e_3^k e_5^k + e_4^k e_5^k)]
\] (4.27)

torsion:

\[
e^k_2 = \left[ (R^p n_z, R^q n_y) - (R^q n_y, R^p n_z) \right] (l^k_0 / 2)
\] (4.28)

bending:

\[
e^k_3 = -(R^p n_z, l^k)
\] (4.29)

\[
e^k_4 = (R^q n_y, l^k)
\] (4.30)
\[ e_5^k = (R^p n_y, 1^k) \]  
\[ e_6^k = -(R^q n_y, 1^k) \]

where

\[
R^i = \begin{bmatrix} \lambda_0^2 + \lambda_2^2 - \lambda_3^2 - \lambda_1^2 & 2(\lambda_1 \lambda_2 - \lambda_0 \lambda_3) & 2(\lambda_1 \lambda_3 + \lambda_0 \lambda_2) \\ 2(\lambda_1 \lambda_2 + \lambda_0 \lambda_3) & \lambda_0^2 - \lambda_1^2 + \lambda_2^2 - \lambda_3^2 & 2(\lambda_2 \lambda_3 - \lambda_0 \lambda_1) \\ 2(\lambda_1 \lambda_3 - \lambda_0 \lambda_2) & 2(\lambda_2 \lambda_3 + \lambda_0 \lambda_1) & \lambda_0^2 - \lambda_2^2 - \lambda_3^2 + \lambda_1^2 \end{bmatrix}, \quad i = p, q
\]

and the vector \( l^k \) is defined by \( l^k = ||l^k|| \), the distance between \( p \) and \( q \) in the deformed state, and \( l_0^k = ||l_0^k|| \), the length in the undeformed state. Here fourteen nodal coordinates and six deformation parameters have been introduced. We have in addition two constraint equations \((\kappa^p)^T \kappa^p - 1 = 0\) and \((\kappa^q)^T \kappa^q - 1 = 0\).

The consistent mass formulation of the flexible beam element is derived on the basis of the elastic line concept, which means that the rotational inertia of the cross section is neglected. In order to calculate the distributed inertia forces of the element, the elastic line configuration of the element should be specified by the nodal coordinates and element deformations using cubic polynomial interpolations for the bending deformations. Based on the configuration function introduced by Besseling (1987) and Jonker (1988, 1989), Meijaard (1991) developed a simplified expression as,

\[
r(\xi) = x^p (1 - 3\xi^2 + 2\xi^3) + R^p n_x (l_0^k + e_0^k) (\xi - 2\xi^2 + \xi^3) + x^q (3\xi^2 - 2\xi^3) + R^q n_x (l_0^k + e_0^k) (-\xi^2 + \xi^3)
\]

where \( \xi \) is a normalized material coordinate along the element axis.

Having differentiated expression (4.34) twice with respect to time, we can calculate the virtual power of the distributed mass of the element as

\[
m^k l_0^k \partial^2 \delta \dot{r} / \partial \xi^2 \right] = \delta x^k T \delta \dot{x}^k T \left[ M^k \left( x^k, \dot{e}^k \right) \left[ \dot{x}^k \atop \dot{\sigma}^k \atop \sigma^k \atop \sigma^k \right] + \left[ -\mathbf{f}_{in}^k \atop \sigma_{in}^k \right] \right]
\]

where \( m^k \) is the mass per unit length of the element. The mass matrix \( M^k \) and the quadratic velocity dependent forces \( [\mathbf{f}_{in}^k, \sigma_{in}^k]^T \) for the beam element are,

\[
M^k = \frac{m^k l_0^k}{420} M^k
\]
\[
\begin{bmatrix}
-f_i^k \\
\sigma_{i,m}^k
\end{bmatrix} = \frac{m^k l_i^k}{420} f_i^k
\]

(4.37)

where

\[
M^k = \begin{bmatrix}
156 I & 22I^k A & 5A I & -13 I^k B & (22 R^p n_x - 13 R^q n_x) \\
4 (I^k)^2 A^T A & 13 I^k A^T & -3 (I^k)^2 A^T B & I^k (4 A^T R^p n_x - 3 A^T R^q n_x) \\
156 I & -22 I^k B & (13 R^p n_x - 22 R^q n_x) & \text{symm.} \\
4 (I^k)^2 B^T B & I^k (-3 B^T R^p n_x + 4 B^T R^q n_x) & 8 - 6 (R^p n_x)^T R^q n_x
\end{bmatrix}
\]

\[
f_i^k = \begin{bmatrix}
I^k (22 A^T \dot{\lambda}^p \dot{\lambda}^p - 13 B^T \dot{\lambda}^g \dot{\lambda}^g) + e_i^k(44 A \dot{\lambda}^p - 26 B \dot{\lambda}^q) \\
(I^k)^2 (4 A^T A^T \dot{\lambda}^p \dot{\lambda}^p - 3 A^T B^T \dot{\lambda}^g \dot{\lambda}^g) + I^k e_i^k(8 A^T A \dot{\lambda}^p - 6 A^T B \dot{\lambda}^q) \\
I^k (13 A^T \dot{\lambda}^p \dot{\lambda}^p - 22 B^T \dot{\lambda}^g \dot{\lambda}^g) + e_i^k(26 A \dot{\lambda}^p - 44 B \dot{\lambda}^q) \\
(I^k)^2 (-3 B^T A^T \dot{\lambda}^p \dot{\lambda}^p + 4 B^T B^T \dot{\lambda}^g \dot{\lambda}^g) + I^k e_i^k(-6 B^T A \dot{\lambda}^p + 8 B^T B \dot{\lambda}^q) \\
I^k [4 (R^p n_x)^T - 3 (R^q n_x)^T] A^T \dot{\lambda}^p + I^k [-3 (R^p n_x)^T + 4 (R^q n_x)^T] B^T \dot{\lambda}^g \\
+ 8 e_i^k (R^p n_x)^T A \dot{\lambda}^p - 6 e_i^k (R^p n_x)^T B \dot{\lambda}^g - 6 e_i^k (R^q n_x)^T A \dot{\lambda}^p + 8 e_i^k (R^q n_x)^T B \dot{\lambda}^g
\end{bmatrix}
\]

\[
A = \frac{\partial}{\partial \lambda^p} (R^p n_x), \quad B = \frac{\partial}{\partial \lambda^q} (R^q n_x),
\]

(4.38)

\[
A' = \frac{\partial A}{\partial \lambda^p}, \quad B' = \frac{\partial B}{\partial \lambda^q} = A',
\]

and I is a 3 by 3 unit matrix.

A flexible link may be represented by one or more beam elements depending on the requirement. Normal links and joints, for example, straight line links and on axis joints, can be directly represented by the beam and hinge elements. For complex links and joints, for example, a link with serval members rigidly connected, or a spherical joint, several elements should be used to represented them even they are ones. The finite element representation of many kinds of links and joints can be found in [37].

Much attention has been paid to the dynamics of the flexible beam with a prismatic joint [Chalhoub and Ulsoy, 1986], [Wang and Wei, 1987], [Krishnamurthy, 1989], [Tadikonda and Baruh, 1992]. We give a more detailed analysis of this kind of flexible links when the finite element method is used. For the prismatic joint with a flexible beam as shown in Fig.4.2 (a),
that is commonly used in mechanisms and robots, special attention should be paid. A flexible link in translational motion with a prismatic joint at one end is usually modelled as a beam in flexure with fixed-free end conditions [Wang and Wei, 1987], [Krishnamurthy, 1989], [Tadikonda and Baruh, 1992] as in Fig. 4.2 (a). It consists of a rigid support and a flexible link that slides in and out of the support. Assume that the beam has uniform mass and stiffness properties. It is clear that at any instant, a part of the beam is outside the rigid support and is free to vibrate. As the beam is extended, the length of the vibrating section of the beam increases, while that held inside the rigid support is reduced and vice versa if the beam is retracted., but the total length of the beam remains constant.

![Diagram](image)

Fig. 4.2 A beam with a prismatic end condition (a) and its finite element representation (b).

The finite element representation of the beam is shown in Fig.4.2 (b). Here the translational motion has been considered as the elongation of the beam element, $e_1$. The flexible part of the link is represented by the bending and torsion deformations of the beam element. For the rigid part of the link, there are two ways to represent it, specify a rigid beam element which has a changing length or use a nodal inertia matrix which gives the inertial properties of the rigid part of the link. The later is used in our analysis.

When the translational motion of the beam is represented by the elongation of the beam element, the deformation $e_1$, can not be considered any more, and the inertial properties and the stiffness matrix will have some change. The inertial matrix $M^k$ in Eq now becomes

$$M^k = \frac{m^k l^k}{420} M^k_0$$  \hspace{2cm} (4.39)

This means that the total mass of the beam element is now $m^k l^k$ instead of $m^k l^k_0$. The
velocity related forces are now given as
\[
\begin{bmatrix}
-f^k_{in} \\
\sigma^k_{i_{in}}
\end{bmatrix} = \frac{M^k f^k_{1}}{420 f^k_{1}}
\]  \hspace{1cm} (4.40)

where \( M^k f^k_{1} \) and \( f^k_{1} \) are given in Eq(4.38).

The stiffness matrix of the beam element is not constant any more and it will change with \( e^k_{1} \) as
\[
S^k =
\begin{bmatrix}
S^k_1 & 0 & 0 & 0 & 0 \\
0 & S^k_2 & 0 & 0 & 0 \\
0 & 0 & 4S^k_3 & -2S^k_3 & 0 & 0 \\
0 & 0 & -2S^k_3 & 4S^k_3 & 0 & 0 \\
0 & 0 & 0 & 4S^k_4 & -2S^k_4 \\
0 & 0 & 0 & 0 & -2S^k_4 & 4S^k_4
\end{bmatrix}
\]  \hspace{1cm} (4.41)

where
\[
S^k_1 = \frac{E^k A^k}{l_0 + e^k_{1}}, \quad S^k_2 = \frac{S^k_i}{(l_0 + e^k_{1})^3}
\]  \hspace{1cm} (4.42)
\[
S^k_3 = \frac{E^k I^k_y}{(l_0 + e^k_{1})^3}, \quad S^k_4 = \frac{E^k I^k_z}{(l_0 + e^k_{1})^3}
\]  \hspace{1cm} (4.43)

\( E^k \) is Young’s modulus, \( A^k \) the cross-sectional area of the beam, \( S^k_i \) the torsional stiffness and \( I^k_y, I^k_z \) are second moments of area of the cross-section.

With the changes introduced in Eq(4.39) to (4.43), the mass matrix and stiffness matrix will change at every time step during the dynamic analysis and simulation. In the simulation or in the analysis of dynamic properties of the system, the system is normally linearized. With the above change, the results of the linearization will be different. Comparing with linearization for the normal beam element [Jonker, 1988, 1990] [Meijarda, 1991b], we have the following extra contributions from this beam element with respect to the deformation \( \delta e^k_{1} \)
\[
\frac{M^k f^k_{1}}{l_0} \begin{bmatrix}
x^k \\
e^k_{1}
\end{bmatrix}, \quad S^k e^k_{1} \frac{M^k f^k_{1}}{420 f^k_{1}}.
\]  \hspace{1cm} (4.44)

where \( M_0 \) can be calculated from Eq(4.36).
If the change of the distributed inertial properties is comparatively small with respect to the lumped inertial properties at the end of the beam, we can simply consider the changing stiffness of the beam. Then the calculation can be simplified.

4.4 Flexible joints

A flexible joint is a joint where besides the normal relative motion, there exists relative elastic motion, as described in [Spong, 1987] [Jankowski, 1993]. Here we are going to analyze the relative rotating joint in which relative elastic rotation exists in addition to the relative rotation introduced by the joint. We will use the spatial hinges to represent the flexible joint. So the spatial hinge element will be introduced first.

4.4.1 The spatial hinge element

The spatial hinge element was first developed to introduce relative motion of spatial beam elements [Werff, 1983] [Besseling et al., 1985]. As shown in Fig.4.3, the spatial hinge element consists of two spatial rotational nodes, p and q, located on its axis.

![Spatial hinge element](image)

Fig.4.3. The spatial hinge element

In the initial condition, the vectors $\mathbf{n}_x$ coincide with the hinge axis and the directions of the vectors $\mathbf{n}_y$ and $\mathbf{n}_z$ of both hinge nodes coincide. When the hinge element moves in space, the rotation of the nodes p and q are introduced by the rotation matrices $R^p$ and $R^q$ which are expressed in Euler parameters as in Eq(4.33).

Three deformation functions are defined for the spatial hinge element, the relative rotation about the axis and two orthogonal bending deformations, relative rotation,
4.4 Flexible joints

\[ e^k_i = \arctan \left( \frac{(R^n n_x, R^n n_z)}{(R^n n_x, R^n n_y)} \right) \]  \hspace{1cm} (4.45)

and bending,

\[ e^k_2 = (R^n n_y, R^n n_z) \]  \hspace{1cm} (4.46)

\[ e^k_3 = (R^n n_z, R^n n_x) \]  \hspace{1cm} (4.47)

For a perfect hinge, the two bending deformations are zero.

4.4.2 The flexible joint and its finite element representation

A flexible joint with a driver and a gear transmission is shown in Fig. 4.4. Here a linear elastic model with stiffness \( k \) and damping \( c \) is given for the joint. All the inertial properties of the driver and the gears can be represented by the inertial matrix \( J \).

![Fig.4.4. A flexible joint](image)

A joint may connect the base of the mechanical system and a link or two links. To represent the whole flexible joint shown in Fig.4.4, two spatial hinge elements and three rotation nodes are used. Two relative rotating variables are given, \( e^1_1 \) and \( e^2_1 \). There are two finite element models used to represent this flexible joint.

Model I: As shown in Fig. 4.5, two serially connected hinge elements are used to represent the flexible joint. The relative rotating variable of hinge 1, \( e^1_1 \), is used to give the actuator rotating angle divided by the gear ratio. The other one, \( e^2_1 \), gives the elastic rotation of the joint. The links, rigid or flexible, may be represented by beam elements.

The stress vector \( \sigma^k \) of the spatial hinge element gives the driving force of a driver, the
elastic force of the flexible joint and constraint forces between links. So here, \( \sigma_i^1 \) corresponding \( e_i^1 \) is the driving force of the driver. And \( \sigma_i^2 \) corresponding \( e_i^2 \) is the elastic force calculated as

\[
\sigma_i^2 = k e_i^2 + c e_i^2
\]

(4.48)

where \( k \) and \( c \) are given as in Fig.4.4.

![Diagram of flexible joint model](image)

Fig.4.5. Finite element representation of the flexible joint

Model II: Three hinge elements are used in this model, as shown in Fig.4.5. Hinge 2 is the same as hinge 1 in Model I, while hinge 1 is connected with node 1 and node 3. The relative rotation variable of hinge 1, \( e_i^1 \), represents the relative rotating angle of the two links connected to this flexible joint. Hinge 3, the same as hinge 2 in Model I, is just used to introduce the elastic stresses and its rotating deformation, \( e_i^3 \), is released. Actually

\[
e_i^3 = e_i^1 - e_i^2.
\]

(4.49)

The driving force of the driver in Model II is given by \( \sigma_i^2 \).

Model I is directly related to the physical system and Model II is not. But Model II has somewhat de-coupled the system. For example, if beam j in Fig.4.5 is an inertial base, the dynamic equation for \( e_i^2 \) can be written immediately

\[
J \ddot{e}_i^2 = \sigma_i^3 = c (\dot{e}_i^1 - \dot{e}_i^2) + k (e_i^1 - e_i^2).
\]

(4.50)

This property of Model II will be used in the simulation of manipulator robots in chapter 6.
All the inertial properties of the joint, including the inertial moment of the rotor of the driver and that of the gears, are introduced as a node inertial matrix. The inertial properties of the rotor of an electrical motor and that of the transmission part are transformed to the inertial matrix of the connected node according to the transmission ratio. A detailed description of dynamics of electrical drivers and their finite element representation may be found in [Gong, 1993].

4.5 A rigid beam element

From Eq(4.18), (4.22) and (4.23), we can notice that the inverse of the matrix

\[
A = \begin{bmatrix}
D^0_{r, xc} \\
D^m_{r, xc}
\end{bmatrix}
\]

(4.51)

has to be used very often. The number of rows of this matrix \( A \) is determined by the number of zero deformation parameters plus the number of independent deformation functions. So in view of calculating efficiency, even a zero deformation has some effect. On the other hand, to let a link transfer rotating motion or moment, the spatial beam or planar beam elements are used to represent a rigid body. For a spatial beam element, there are six translational coordinates and eight rotational coordinates with two more constraints when the components are introduced by Euler parameters. The number of deformation functions for a spatial beam element is six. When a spatial beam is considered flexible, there are sufficient coordinate parameters to calculate these deformation functions. But when mechanisms or robots are considered as composed of rigid links or some links are considered as rigid and some as flexible, three more coordinates than necessary are given and that results in three more zero deformation functions for the beam element.

Here we introduce a beam element to represent rigid links. To distinguish it from the normal beam element, we call it a rigid beam element. First the basic theory of the rigid beam element is introduced. Then numerical examples will be given to show the calculating efficiency.

4.5.1 The rigid beam element

One of the basic conditions for the definition of a finite element is that suitable nodal coordinates should be introduced to give the geometric configuration of the element and suitable deformation functions be introduced to give correct constraint conditions among the nodal coordinates connected to the element. The suitable nodal coordinates let the element represent a physical system correctly and have suitable linkages with other elements. After the nodal coordinates are determined, the deformation functions are defined using the principles given in section 4.2.
Unlike a flexible beam element which has two rotational nodes, a rigid beam element needs only one rotational node to give sufficient geometric information of the element, as shown in Fig.4.6.

Fig.4.6 A rigid beam element

With the components of the rotation matrix expressed in Euler parameters, the rigid beam element has ten nodal coordinates, six Cartesian coordinates and four Euler parameters

\[ x^k = [x^p, y^p, z^p, \lambda^0, \lambda^1, \lambda^2, \lambda^3, x^q, y^q, z^q]^T \]  

(4.52)

We define the deformation functions as

\[ e^k_1 = l^k - l^k_0 \]  

(4.53)

\[ e^k_2 = \left( R_{n_z}, I^k \right) \]  

(4.54)

\[ e^k_3 = \left( R_{n_y}, I^k \right) \]  

(4.55)

Where \( I^k \) is defined as

\[ I^k = [x^q - x^p, y^q - y^p, z^q - z^p]^T \]  

(4.56)

and \( l^k_0 \) is the original length of the beam element. The transformation matrix \( R \) is given as Eq(4.34).

Here \( e^k_1 \) can be a real deformation while \( e^k_2 \) and \( e^k_3 \) are virtual deformations.

Similar to the spatial rigid beam element, a planar rigid beam element can also be defined. The geometric position of a planar beam element can be determined by five nodal coordinates,
four Cartesian coordinates and one rotation angle relative to its original position

\[ x^k = [x^p, y^p, \beta, x^q, y^q]^T \] (4.57)

And two deformation functions can be defined as

\[ e^k_1 = l^k - l^k_0 \] (4.58)

\[ e^k_2 = (x^q - x^p) \sin (\beta + \beta_0) - (y^q - y^p) \cos (\beta + \beta_0) \] (4.59)

where \( \beta_0 \) is defined as the angle between \( l_0 \) and the global x axis.

For the rigid beam elements, the elongation deformation, \( e^k_1 \) can be a non-zero deformation and the two bending deformation functions are always zero.

4.5.2 Examples

We give two examples to show the calculating efficiency by using the rigid beam elements compared with using normal flexible beam elements when the deformations are considered to be zero.

![A planar mechanism](image.png)

Fig.4.7 A planar mechanism

The first example (Fig.4.7) is a planar mechanism given by Schiehlen (1990). For kinematic analysis, using the rigid beam element in the SPACAR system can reduce the calculating time.
by nineteen percent. For dynamic analysis, the calculating time can be reduced by twenty-three percent.

Another example is the spatial turbula mechanism as shown in Fig.4.8. For kinematic analysis, the calculating time using the rigid beam element in the SPACAR system is thirty percent less than that of using the normal beam element. The calculating time is reduced by thirty-three percent for dynamic analysis.

![Fig.4.8 The turbula mechanism](image-url)
Chapter 5

Dynamic Analysis of Mechanical Systems

5.1 Introduction

In dynamic analysis, many mechanical systems, whether mechanisms or manipulator robots, and either using a multi-body approach or a finite element approach, can be modelled by the following variational equation

$$\delta x^T [M(x) \ddot{x} - f(x, \dot{x}, t)] = 0$$  \hspace{1cm} (5.1)

with the constraints

$$\Phi(x, t) = 0$$  \hspace{1cm} (5.2)

or with stationary constraints and driving constraints as

$$\Phi(x) = 0,$$  \hspace{1cm} (5.3)

and

$$q^r = q^r(t),$$  \hspace{1cm} (5.4)

$x, \dot{x}, \ddot{x}$ are the displacement, velocity and acceleration vectors of the generalized coordinates, $\delta x$ is a kinematically admissible virtual displacement vector, $M(x)$ is the symmetric inertia matrix, $f(x, \dot{x}, t)$ is the force vector and $q^r$ is a vector of independent coordinates describing the driving constraint conditions.

The condition for a kinematically admissible virtual displacement $\delta x$ is obtained by taking the differential of the constraint equations with time held fixed as
\[ \Phi_x \delta x = 0. \quad (5.5) \]

By using the Lagrangian multiplier technique to Eq.(5.1) and (5.5), there exits a Lagrange multiplier vector \( \lambda \) so that

\[ \delta x^T [M(x) \dot{x} - f(x, \dot{x}, t)] + \delta x^T \Phi^T_x \lambda = 0 \quad (5.6) \]

for arbitrary \( \delta x^T \). So we have

\[ M(x) \dot{x} - f(x, \dot{x}, t) + \Phi^T_x \lambda = 0 \quad (5.7) \]

By differentiating Eq.(5.2) with respect to time, we have

\[ \Phi_x \ddot{x} + \Phi_x \dot{t} = 0, \quad (5.8) \]

\[ \Phi_x \ddot{x} + (\Phi_x \dot{x})_x \dot{x} + 2 \Phi_x \dot{x} \dot{x} + \Phi_x \dot{t} = 0. \quad (5.9) \]

Put Eq.(5.7) and (5.9) together, we get a mixed system of differential-algebraic equations (DAE) as

\[ \begin{bmatrix} M & \Phi^T_x \\ \Phi_x & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ Y \end{bmatrix}, \quad (5.10) \]

where

\[ Y = -\left( \left( \Phi_x \dot{x} \right)_x \dot{x} + 2 \Phi_x \dot{x} \dot{x} + \Phi_x \dot{t} \right) \quad (5.11) \]

The mass matrix \( M \) and the generalized applied force \( f \) are nonlinear functions of the generalized coordinates, their velocities and time. So the system is very complicated and numerical methods are required to get the solutions of Eq.(5.1) or (5.10) with the constraint condition of Eq.(5.2).

In this chapter, we first study the numerical solution of the differential-algebraic equations (DAEs) by using an equation reduction method. Then, the automatic linearization of the system is presented. After that, we discuss the direct determination of periodic solutions of the mechanical system. Finally, we give some numerical examples of mechanical systems.

5.2 Numerical method for equation reduction

Many numerical methods to solve the above equations have been introduced in literature. They are the coordinate reduction method [Nikravesh, 1990], the direct integration method
5.2 Numerical method for equation reduction

[Haug, 1989], the coordinate partitioning method [Besseling, 1979], [Wehage and Haug, 1982], the constraint stabilization method [Baumgarte, 1972] and the hybrid method.

The coordinate reduction method introduced by Nikravesh (1990) use a two-step process to convert the equations of motion for closed-loop systems from a large set of absolute coordinates to a minimal set of relative joint coordinates. First the absolute coordinates are used to define the position of each body, the kinematic joints and the forces acting on the bodies. Then these absolute coordinates are defined as functions of the relative joint coordinates. Before numerical integration of the equations of motion, the equations are converted to a minimal set to gain computational efficiency.

The direct integration method [Haug, 1989] simply solves Eq(5.10) for the acceleration $\ddot{x}$ and integrates for $x$ and $\dot{x}$, with errors in satisfying constraints being ignored. This method suffers from an accumulation of constraint errors and may lead to substantial violation of the position and velocity constraint equations of Eq(5.2) and (5.8). So it may be used only for moderately regular system dynamics applications and small intervals of time.

The coordinate partitioning method given by Besseling (1979) and Wehage and Haug (1982) introduces the automatic partitioning of the coordinates and the reduction of the equations. The method introduced by Besseling (1979) is based on the equations of motion derived by the finite element method, and the method introduced by Wehage (1982) is based on the equations given as Eq(5.10).

5.2.1 Equation reduction method

To solve the equations with time-dependent constraints as Eq(5.1) and (5.2), we can choose a set of $n$ independent generalized variables $q(t)$. Then we have

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = F(q, t) = \begin{bmatrix} u(q, t) \\ q(t) \end{bmatrix},$$

(5.12)

where $u(q, t)$ is the dependent coordinate vector.

If we can find the functions $F(q, t)$ from the constraint equations, Eq(5.2), and substitute them into Eq(5.1), we may get a set of $n$ reduced differential equations and the problem can be solved. But the constraint equations are normally nonlinear and the theoretical expression for $F(q, t)$ would be very difficult if not impossible. However, when we are trying to find a numerical solution, we can use the numerical results instead of the analytical expressions. This is a generalization of the method introduced by Besseling [Besseling, 1979] for solving dynamic problems using the finite element method where the constraint equations are given as element deformation functions.
From Eq (5.8), (5.9) and (5.12), we have

$$\dot{x} = F_q \ddot{q} + \nu$$ \hspace{1cm} (5.13)

and

$$\ddot{x} = F_q \dddot{q} + a$$ \hspace{1cm} (5.14)

where

$$\nu = \begin{bmatrix} -\Phi^{-1}_{.,u} \Phi_{.,.} \\ 0 \end{bmatrix}$$ \hspace{1cm} (5.15)

and

$$a = \begin{bmatrix} -\Phi^{-1}_{.,u} \left( \left( \Phi_{.,x} \dot{x} \right)_{.,.x} + 2 \Phi_{,xx} \dot{x} + \Phi_{,..x} \right) \\ 0 \end{bmatrix}$$ \hspace{1cm} (5.16)

It can be noticed that the accelerations $\ddot{x}$ are linear functions of the acceleration of the generalized coordinates $\ddot{q}$. So if we can find the numerical values of $x$ and $\dot{x}$ via $q$ and $\dot{q}$, the DAEs of Eq (5.1) can be reduced to DEs and be solved numerically.

From Eq (5.2) and (5.12), with the time fixed, we have

$$\Phi_{.,u} u_q + \Phi_{.,q} = 0$$ \hspace{1cm} (5.17)

and

$$(\Phi_{.,xx} F_{.,j}) F_{.,j} + \Phi_{.,u} u_{.,j} = 0.$$ \hspace{1cm} (5.18)

Here the subscript indicates the partial derivatives, e.g., $\Phi_{,x} = \frac{\partial \Phi}{\partial x}$ and, $\Phi_{,xx} = \frac{\partial^2 \Phi}{\partial x^2}$ and so on, $i$ and $j$ indicate the derivatives with respect to the generalized variables $q_i$ and $q_j$ respectively. So we have

$$u_{.,q} = -\Phi^{-1}_{.,u} \Phi_{.,q}.$$ \hspace{1cm} (5.19)

$$u_{.,ij} = -\Phi^{-1}_{.,u} \left( \Phi_{.,xx} F_{.,i} \right) F_{.,j}.$$ \hspace{1cm} (5.20)

Hence
\[ F_{\cdot q} = \begin{bmatrix} u_{\cdot q} \\ I \end{bmatrix} = \begin{bmatrix} -\Phi^{-1} \Phi_{\cdot q} \\ I \end{bmatrix} \]  
(5.21)

and

\[ F_{\cdot qq} = \begin{bmatrix} -\Phi^{-1} \Phi_{xx} F_{\cdot q} F_{\cdot q} \\ 0 \end{bmatrix}. \]  
(5.22)

After the determination of the first order partial derivatives \( F_{\cdot q} \), the values of the dependent variables \( u(q) \) can be calculated by using the Newton-Raphson iteration method. When the iteration method is used, one of the important things is to determine the first guess of the dependent variables. To solve the DAEs of Eq(5.1) and (5.2), we have the following initial conditions at \( t_0 = 0 \)

\[ x_0 = x(t_0), \]  
(5.23)

\[ \dot{x}_0 = \dot{x}(t_0), \]  
(5.24)

and the constraint equations

\[ \Phi_0 = \Phi(x_0) = 0. \]  
(5.25)

After a time step, the generalized variables \( q \) will get new values. The first guess of the dependent variables is given as

\[ u^{(0)} = u_0 + u_{\cdot q} \Delta q + \frac{1}{2} (u_{\cdot qq} \Delta q) \Delta q. \]  
(5.26)

The iteration process is applied in order to guarantee that the constraint equations be ultimately satisfied. In the iteration process, a new approximation of the dependent variables \( u^{(k+1)} \) is determined by

\[ u^{(k+1)} = u^{(k)} + \Delta u^{(k)} \quad k = 0, 1, \ldots, \]  
(5.27)

where the residual vector \( \Delta u^{(k)} \) is calculated from

\[ \Delta u^{(k)} = -\Phi_{\cdot u}^{-1} \Phi(x^{(k)}). \]  
(5.28)

The iteration process continues till the final solution satisfies the following condition

\[ |\Phi(x^{(k)})| \leq \varepsilon, \]  
(5.29)
where \( \varepsilon > 0 \) is the user specified error tolerance.

The Newton-Raphson method has the attractive property of being quadratically convergent, that is the solution error in a given iteration is proportional to the square of the error in the preceding iteration. However, the method may diverge if poor initial estimates are given. And also it is obvious that the matrix \( \Phi_{\delta u} \) needs to be non-singular.

All the velocities can be calculated from the velocities of the generalized variables by Eq(5.13) and the virtual displacement vector is determined as

\[
\delta x = F_{\delta q} \delta q .
\]  
\text{(5.30)}

Substituting Eq(5.13), (5.14) and (5.30) into Eq(5.1), we have

\[
\delta q^T F_{\delta q}^T (MF_{\delta q} \dot{q} + Ma - f) = 0 .
\]  
\text{(5.31)}

So the reduced DEs are as follows

\[
F_{\delta q}^T MF_{\delta q} \ddot{q} = F_{\delta q}^T (f - Ma) .
\]  
\text{(5.32)}

Then the acceleration of the generalized coordinates \( \ddot{q} \) can be evaluated. In order to integrate the equations of motion, the above equations can be written in first order form as

\[
\frac{d}{dt} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ (F_{\delta q}^T MF_{\delta q})^{-1} F_{\delta q}^T (f - Ma) \end{bmatrix}
\]  
\text{(5.33)}

and the system can be integrated numerically for \( q \) and \( \dot{q} \) at the next time step.

For many mechanical systems, the constraint equations can be given as stationary constraints

\[
\Phi (x) = 0
\]  
\text{(5.34)}

and driving constraints

\[
q^T = q^T (t) .
\]  
\text{(5.35)}

Here \( q^T \) is a vector of generalized coordinates describing the driving constraints. So the dependent coordinate vector \( q \) now becomes

\[
q^T = \begin{bmatrix} q^T \n q^d \end{bmatrix}
\]  
\text{(5.36)}

where \( q^d \) is the vector of dynamic degrees of freedom.
5.3 Linearization of the equations of motion

With the constraint equations given as Eq(5.34) and (5.35), we have

\[ \delta x = F_c q \delta q^d \]  

(5.37)

and the reduced equations become

\[ F_{aq}^T MF_{aq} q^d \delta q^d = F_{aq}^T [f - M (F_{aq} \ddot{q} + F_{aq} \dot{q})] \]  

(5.38)

5.2.2 Computational implementation

The computational implementation of the equation reduction method can be carried out as follows:

1. Begin with the an assembled configuration and initial conditions that satisfy Eq(5.2) at the initial time \( t_0 \).

2. Evaluate \( \Phi_{x} \) and \( \Phi_{u} \). If \( \Phi_{u} \) is singular, determine the generalized coordinate vector \( q \) first. Construct \( \Phi_{u} \) by choosing \( m \) columns of \( \Phi_{x} \) so that \( \Phi_{u} \) is regular and the corresponding elements in \( x \) construct the dependent variable vector \( u \) and the rest of the elements in \( x \) construct the generalized coordinate vector \( q \). At the same time, \( \Phi_{q} \) is constructed by the columns of \( \Phi_{x} \) corresponding to the elements in \( q \). The partitioning of the generalized coordinate vector \( x \) into \( u \) and \( q \) will be retained until computational tests indicate that a new partitioning is required. Calculate \( F_{aq} \) and \( F_{aq} \) from Eq(5.21) and (5.22). Calculate \( \dot{u} (t_i) \) from Eq(5.13) and if necessary, calculate \( \dot{u} (t_i) \) from Eq(5.14).

3. Solve Eq(5.32) or (5.38) for \( \ddot{q}. \)

4. Integrate the first-order system of Eq(5.33) from \( t_i \) to \( t_{i+1} \) to obtain \( q (t_{i+1}) \) and \( \dot{q} (t_{i+1}) \).

5. Calculate the first guess of the dependent coordinates \( u (t_{i+1}) \) by Eq(5.26).

6. Solve Eq(5.27) and (5.28) for \( u (t_{i+1}) \) using the Newton-Raphson method.

7. Evaluate \( \Phi_{x} \) and \( \Phi_{u} \). If \( \Phi_{u} \) is singular, return to step (2). Otherwise, continue.

8. Calculate \( F_{aq} \) and \( F_{aq} \) from Eq(5.21) and (5.22).

9. Calculate \( \dot{u} (t_{i+1}) \) from Eq(5.13) and \( \dot{u} (t_{i+1}) \) from Eq(5.14).

10. If \( t_{i+1} \) exceeds the final time, stop. Otherwise, update \( i \) to \( i+1 \) and return to step (3).

5.3 Linearization of the equations of motion

After deriving the minimal set of equations of motion of the mechanical system, one may
like to study the stability of the motion by Liapunov’s first method and the sensitivity of the solution to small disturbances and parameter variations or one may like to determine the main natural frequencies and associated modes based on the linearized equations for the dynamic response analysis of a mechanism or robot with flexible links and flexible joints. So a linearization of the system is needed. Linearization is also needed in the calculation of periodic solutions. The linearized equations of motion enable us to study the stability properties near the linearization point and provide us some necessary conditions for stability of the nonlinear system. Numerical differentiation can be used to get the linearized equations, but this leads to reduction of the accuracy and is less efficient than analytical differentiation.

Consider the dynamic equations for the generalized variables with variations

\[ q = q_0 + \delta q, \quad (5.39) \]

\[ \dot{q} = \dot{q}_0 + \delta \dot{q}, \quad (5.40) \]

\[ \ddot{q} = \ddot{q}_0 + \delta \ddot{q}. \quad (5.41) \]

The linearized equations of Eq(5.32) can be written as

\[ M_0 \delta \ddot{q} + C_0 \delta \dot{q} + K_0 \delta q = F^{T}_{.q_0} \delta f, \quad (5.42) \]

where

\[ M_0 = F^{T}_{.q} M F_{.q}, \quad (5.43) \]

\[ C_0 = F^{T}_{.q} M a_{.q} - F^{T}_{.q} F_{.q} F_{.q}, \quad (5.44) \]

\[ K_0 = F^{T}_{.q} (M F_{.qq} \ddot{q} + M F_{.q} \dddot{q} + M a_{.q} - f_{.q} F_{.q} - f_{.q} (F_{.qq} \dot{q} + v_{.q})) \]

\[ + F^{T}_{.q} (M \dddot{x} - f) \], \quad (5.45) \]

and

\[ M_{.x} = \frac{\partial M}{\partial x}, \]

\[ f_{.x} = \frac{\partial f}{\partial x}, \quad f_{.\dot{x}} = \frac{\partial f}{\partial \dot{x}}. \]
\[ a_{qq} = \begin{bmatrix} \Phi_{u}^{-1} [2 \Phi_{x} x \hat{x} + 2 \Phi_{x} x \hat{x}] F_{x} q \end{bmatrix}, \]
\[ a_{q} = \begin{bmatrix} a_{1, q} \\ 0 \end{bmatrix}, \]
\[ a_{1} = -\Phi_{x}^{-1} (\Phi_{x} x \hat{x} + 2 \Phi_{x} x \hat{x} + \Phi_{x} x), \]
\[ a_{1, q} = -\Phi_{x}^{-1} [\Phi_{x} x F_{x} a_{q} + \Phi_{x x} x F_{x} (\Phi_{x} x q + v_{x})] \]
\[ -\Phi_{x}^{-1} [2 \Phi_{x} x F_{x} q \hat{x} + 2 \Phi_{x} x F_{x} (\Phi_{x} q q + v_{x}) + \Phi_{x x} x F_{x} q], \]
\[ v_{q} = \begin{bmatrix} \Phi_{u}^{-1} [\Phi_{x} x F_{x} \Phi_{x} q] - \Phi_{x} x F_{x} q \end{bmatrix}. \]

Corresponding to Eq(4.26) or (5.38), the linearized equations become
\[ M_{0} \delta q^{d} + C_{0} \delta q^{d} + K_{0} \delta q^{d} = F_{x}^{T} q_{0} \delta f, \]
where
\[ M_{0} = F_{x}^{T} q_{0} M F_{x} q_{0}, \]
\[ C_{0} = F_{x}^{T} q_{0} (2 M F_{x} q_{0} \dot{q} - f_{x} F_{x} q_{0}), \]
\[ K_{0} = F_{x}^{T} q_{0} (M \ddot{x} - f) - F_{x}^{T} q_{0} f_{x} F_{x} q_{0} + \]
\[ + F_{x}^{T} q_{0} (M \ddot{x} F_{x} q_{0} + \dot{f}_{x} F_{x} \dot{q} + M (F_{x} q_{0} \dot{q} + F_{x} q_{0} \ddot{q})), \]
\[ F_{ijk} = \begin{bmatrix} a_{ijk} \\ 0 \end{bmatrix}, \]
with
\[ a_{ijk} = -\Phi_{x}^{-1} [\Phi_{xxx} F_{x} F_{x} F_{x} + \Phi_{x} x F_{x} F_{x} + \Phi_{x x} x F_{x} F_{x} + \Phi_{x x} x F_{x} F_{x}]. \]

We note that in computer implementation, the term \( F_{x} q_{0} \ddot{q} \) in Eq(5.33) can be obtained by calculating \( u_{pp} \ddot{q} \) as
\[ u_{qqq} \ddot{q} = -\Phi^{-1} \left[ \left( (\Phi_{x\dot{x}} \dot{x}) F_{,qq} + 2 (\Phi_{xx} (F_{,qq} \dot{q}) \dot{x} + \Phi_{xx} (F_{,q} \ddot{q} \dot{q}) F_{,q} \right) \right]. \]

This can make the calculation more efficient.

Corresponding to Eq(4.26), the reduced mass matrix \( M_0 \) is the same as Eq(5.51). The velocity sensitivity matrix \( C_0 \) and the stiffness matrix \( K_0 \) are calculated as [Jonker, 1988, 1990] [Meijaard, 1991a-b]

\[
C_0 = F_{,qq}^T (2MF_{,qq} + CF_{,q}) ,
\]

\[
K_0 = F_{,qq}^T \left[ M \begin{bmatrix} x \\ \dot{e} \end{bmatrix} - f \right] + 
+ F_{,qq}^T \left( KF_{,qq} + M(F_{,qq} \dot{q} + F_{,qq} \ddot{q}) + CF_{,qq} \dddot{q} \right),
\]

where the matrices \( C \) and \( K \) are calculated from the contribution of the nodal points and elements as

\[
C^k = \begin{bmatrix}
    \frac{\partial}{\partial x} f^k \\
    \frac{\partial}{\partial \dot{x}} \sigma^k
\end{bmatrix} \begin{bmatrix}
    \frac{\partial}{\partial x} f^k \\
    \frac{\partial}{\partial \dot{x}} \sigma^k
\end{bmatrix},
\]

\[
K^k = \begin{bmatrix}
    \frac{\partial f_{tot}^k}{\partial x^k} \\
    -\sigma^k
\end{bmatrix} \begin{bmatrix}
    \frac{\partial f_{tot}^k}{\partial \dot{e}^k}
\end{bmatrix},
\]

where

\[
f_{tot}^k = \begin{bmatrix}
    f^k \\
    -\sigma^k
\end{bmatrix} - M^k \begin{bmatrix}
    \dot{x}^k \\
    \dot{e}^k
\end{bmatrix}.
\]

### 5.4 Stationary solution of mechanical systems

In the analysis of mechanical systems with flexible bodies, stationary solutions are normally used as initial values. For an autonomous system, described by the following autonomous differential equations,

\[ \dot{q} = g(q) , \]

the stationary solutions are independent of time and can be obtained by solving the algebraic
5.4 Stationary solution of mechanical systems

\[ g(q) = 0. \]  \hspace{1cm} (5.61)

So for the systems given by Eq(5.32), the stationary solutions are given by the solution of the following equations

\[ g(q) = F^T_q (q - Ma) = 0 \]  \hspace{1cm} (5.62)

The Newton-Raphson method can be used to solve the above non-linear equations. By the Newton-Raphson method, first Eq(5.61) is linearized to calculate \( g.q \). Then the new approximation, \( q^{(k+1)} \), is calculated as

\[ q^{(k+1)} = q^{(k)} + \Delta q^{(k)} \quad (k = 0, 1, ...), \]  \hspace{1cm} (5.63)

where the correction \( \Delta q^{(k)} \) is calculated by solving the following linear equations

\[ g.\dot{q}^{(k)} \Delta q^{(k)} = -g(q^{(k)}). \]  \hspace{1cm} (5.64)

using for example Gaussian elimination with partial pivoting [Golub and Van Loan, 1989].

If there are some driving constraints in the mechanical system given as coordinates describing rheonomic constraints \( q^r \), the terms related to the velocities and accelerations of \( q^r \) should be considered for the stationary solutions. In this case, we have

\[ x^T = \begin{bmatrix} u^T & q^r & q^d \\ \end{bmatrix}, \]  \hspace{1cm} (5.65)

\[ \dot{x}^T = \begin{bmatrix} \ddot{u}^T & \dot{q}^r & 0 \\ \end{bmatrix}, \]  \hspace{1cm} (5.66)

\[ \ddot{x}^T = \begin{bmatrix} \dddot{u}^T & \ddot{q}^r & 0 \\ \end{bmatrix}. \]  \hspace{1cm} (5.67)

To find the stationary solutions for the independent dynamic degrees of freedom, \( q^d \), we first need to find \( x \) and \( \dot{x} \). Then we can calculate \( q^d \) by the Newton-Raphson method and also calculate \( u \) from \( q^r \) and \( q^d \).

From Eq(5.38), we have

\[ F^T_{q_d} (f - M (F_{qq} \ddot{q} + F_{q} \dddot{q}^r)) = 0. \]  \hspace{1cm} (5.68)

To find the solutions for \( q^d \) from Eq(5.68) by the Newton-Raphson method, the approximation of \( q^{d(k+1)} \) is calculated from Eq(5.63) and the correction \( \Delta q^{d(k)} \) is calculated by solving the following linear equations
\[ K_0 \Delta q^{d(k)} = -g(q^{d(k)}) , \]  

matrix \( K_0 \) is given as Eq(5.57) and

\[ g(q^{d(k)}) = F_{x}^{T} (x^{(k)}) (f(x^{(k)}) - M(x^{(k)}) (F_{qq} (x^{(k)}) \dot{q}^{(k)} \dot{q}^{(k)} + F_{q} (x^{(k)}) \ddot{q}^{(k)})) \]

where \( x^{(k)} \) is determined by \( q^{d(k)} \) as in section 5.2.

The stability of the stationary solution is determined by the eigenvalues of the Jacobian matrix, \( g_x \) in Eq(5.64) or \( K_0 \) in Eq(5.69). If all eigenvalues of the Jacobian matrix have negative real parts, the solution is stable. If some eigenvalues have a positive real part, the solution is unstable. And if some eigenvalues are purely imaginary or zero, there is a bifurcation point.

### 5.5 Periodic solutions of mechanical systems

Periodically forced systems are often observed in mechanical systems. For this kind of systems, we may like to find their periodic solutions. A periodically forced system with \( n \) variables can be given in state equations as

\[ \dot{x} = g(x, t), \quad g(x, t + T) = g(x, t). \]  

The shooting method [Seydel, 1988] [Meijaard, 1991b] can be used to find the periodic solutions of Eq(5.70). In this method, the initial values of the periodic solution at \( t = t_0 \) are given, for example, by the stationary solution of the system as \( x(t_0) = x_0 \). Then the differential equations Eq(5.70) are integrated over a minimal period of time \( mT \). Here \( m \) is an integer larger than or equal to one. If \( m \) is equal to one, the solution has the same period as the force and if \( m \) is larger than one, the solution is a sub-harmonic one. The integration gives us the result \( x_1 = x(t_1) \) at \( t = t_1 = t_0 + mT \).

Also the linearized equations of Eq(5.70),

\[ \delta \dot{x} = g_x(x, t) \delta x \]  

are integrated \( n \) times with the initial values given by the \( n \) columns of an \( n \) by \( n \) unit matrix and we get an \( n \) by \( n \) amplification matrix \( A \). The growth of small perturbations of the solution is determined by this matrix \( A \).

Our objective is to find an \( x_0 \) so that after the integration we have \( x_1 = x_0 \). If the solution \( x_1 \) is not equal to \( x_0 \), we need to adjust the initial values \( x(t_0) \) as

\[ x(t_0) = x_0 + \Delta x_0 \]
where $\Delta x_0$ is determined by

$$\Delta x_0 = -(A - I)^{-1} (x_1 - x_0).$$

(5.73)

This procedure is repeated until we find that $x_1 - x_0$ is sufficiently small.

The stability of the periodic solution is determined by the so-called characteristic multipliers, the eigenvalues of $A$. The solution is stable if all eigenvalues have a modulus smaller than one. The solution is unstable if some eigenvalues have a modulus larger than one. There are bifurcation points if some eigenvalue have a modulus equal to one. A detailed description of the determination of periodic solutions has been given by Meijaard (1991b).

5.6 Numerical examples of dynamic systems

5.6.1 The four bar mechanism

The first example is a four-bar mechanism as shown in Fig. 5.1. This mechanism was used by Alexander (1975) in his experiment. In the experimental setup, the polar moment of inertia on the shaft which drives the input link was assumed large enough to provide a constant input rotational speed. All links were made of aluminium. Experimental results were obtained by placing strain gauges on the upper and lower surfaces of the coupler and output links at various locations, including midway along their lengths.

![Fig. 5.1 A four-bar mechanism](image)

The properties of this mechanism are listed in Table 5.1. The length of the base link, (the distance between the two supports), is 10.0 in. The mass densities of the coupler and output links have been increased by 8% to account for the additional mass of the wires and strain gauges attached to these links in the experimental setup as recommended by Alexander. The bearing masses at pins joining the input link and coupler, as well as the coupler and output link
are 0.06 lbm.

Table 5.1. Properties of four-bar mechanism

<table>
<thead>
<tr>
<th>Link</th>
<th>Modulus of Elasticity, E (lb/in²)</th>
<th>Mass Density, ρ (lb-sec²/in⁴)</th>
<th>Link Height, h (in)</th>
<th>Link Width, b (in)</th>
<th>Link Length, L (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>10. x 10⁶</td>
<td>2.52 x 10⁻⁴</td>
<td>0.167</td>
<td>1.00</td>
<td>4.00</td>
</tr>
<tr>
<td>Coupler</td>
<td>10. x 10⁶</td>
<td>2.73 x 10⁻⁴</td>
<td>0.063</td>
<td>1.00</td>
<td>11.00</td>
</tr>
<tr>
<td>Output</td>
<td>10. x 10⁶</td>
<td>2.73 x 10⁻⁴</td>
<td>0.063</td>
<td>1.00</td>
<td>10.50</td>
</tr>
</tbody>
</table>

For the analysis of this mechanism, the crank is represented by one beam element, the coupler and the output links are both represented by two beam elements with equal length. The dimensions and properties of the links are as shown in Fig.5.1. At the initial position, the crank is in x-axis direction and the simulation is performed as the crank rotates at a constant speed of 41.88 rad/s. Because this is a periodically forced system, the periodic solution is determined first and the results are shown in Table 5.2. These values are used as the initial values in the simulation. With these initial conditions, the system is simulated for one revolution of the crank.

Table 5.2. Periodic solutions of the four-bar mechanism at initial position

<table>
<thead>
<tr>
<th>Link</th>
<th>ε₂ (in)</th>
<th>ε₃ (in)</th>
<th>̇ε₂ (in/s)</th>
<th>̇ε₃ (in/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam 1 (B1)</td>
<td>-0.0243899</td>
<td>-0.0121394</td>
<td>-14.6685</td>
<td>-7.2793</td>
</tr>
<tr>
<td>Beam 2 (B2)</td>
<td>-0.0605827</td>
<td>-0.0920681</td>
<td>-13.7811</td>
<td>-4.6631</td>
</tr>
<tr>
<td>Beam 3 (B3)</td>
<td>-0.0690702</td>
<td>-0.0378128</td>
<td>45.0027</td>
<td>35.6513</td>
</tr>
<tr>
<td>Beam 4 (B4)</td>
<td>-0.0262803</td>
<td>-0.0390905</td>
<td>84.6188</td>
<td>105.9670</td>
</tr>
<tr>
<td>Beam 5 (B5)</td>
<td>-0.0073998</td>
<td>0.0037284</td>
<td>-5.3304</td>
<td>-27.1435</td>
</tr>
</tbody>
</table>

The elastic displacement of the middle point of the coupler is shown in Fig.5.2 and that of the output link in Fig.5.3.
To compare the results in this analysis with the experimental results given by Alexander, the displacement of the middle point of the coupler is transferred to the strain. When the link is considered as a simply supported beam, the axial strain caused by the bending of the beam about a principal axis of the cross-section is given by the well-known formula

$$\varepsilon_x = \frac{M_y z}{EI_y}$$  \hspace{1cm} (5.74)

where $M_y$ is the moment at the cross section, $E$ is the modulus of elasticity, $I_y$ is the moment of inertia of the cross-sectional area about the neutral axis, and $z$ is the coordinate from the neutral $y$-axis. For a rectangular beam with parameters as shown in Table 5.1, we have
\[ \varepsilon_x = \frac{M_y h}{EI_y^2} \]  

(5.75)

where \( h \) is the height of the beam.

If the beam is subjected to pure bending, the displacement of the middle point is

\[ w = \frac{M_y h}{EI_y^2} \]  

(5.76)

Compare Eq.(5.75) and (5.76), we have

\[ \varepsilon_x = \frac{4h}{I^2} w \]  

(5.77)

The strains at the middle point of the coupler calculated from Eq.(5.77) are shown in Fig.5.4. From Fig.5.4, it can be noticed that the calculated results are in reasonable agreement with those of the experiments.

![Graph showing strain vs. crank angle with simulation and experimental results](image)

**Fig. 5.4** Coupler midpoint strains

To show that the dynamic response of the periodically forced system is also periodic, we give the phase plane representation of Fig. 5.2 and Fig. 5.3. Fig. 5.5 shows the periodic solutions of the middle point of the coupler link and Fig. 5.6 shows the same data for the output link.
5.6 Numerical examples of dynamic systems

Fig. 5.5 Coupler middle point periodic solution, velocities of elastic displacements - elastic displacements

Fig. 5.6 Output link middle point periodic solution, velocities of elastic displacements - elastic displacements

5.6.2 A governor mechanism

A governor mechanism example given by Haug (1989) is shown in Fig. 5.7. It consists of a shaft, two balls connected to the shaft by two arms and a collar and two couplers. The arms
attached to the balls are connected to the shaft by revolute joints with paralleled axes perpendicular to the shaft. The collar can move along the shaft.

The function of the governor is to maintain a nearly constant angular speed \( \omega \) of the shaft under a varying resisting torque from the driven machine. If, due to the increase of the resisting torque of the driven machine, the angular velocity of the shaft decreases, the balls drop and hence lower the collar. A linkage attached to the collar then opens the fuel feed to the engine and the engine generates an increased torque and this leads to a speed-up of the shaft. As a result, the balls rise toward their nominal position as the shaft angular velocity approaches the desired value. The compensate torque, \( \Delta T \), by the engine corresponding to spring deformation \( \Delta l \) (vertical movement of collar) is given as

\[
\Delta T = C\Delta l
\]  

(5.78)

where \( C \) is the torque generated due to fuel fed to the engine by a unit \( \Delta l \).

The inertial properties of the governor are shown in Table 5.3. The length of the arms is 0.16m, that of the couplers is 0.10922m, and the free length of the spring is 0.15m. The stiffness of the spring \( k=1000 \) N/m and its damping rate \( c=30 \) N/s. The couplers are connected to the middle points of the arms.

Fig. 5.7 A governor mechanism
Table 5.3. Inertia properties of governor mechanism

<table>
<thead>
<tr>
<th>Body</th>
<th>Mass (kg)</th>
<th>$I_x$ (kg.m$^2$)</th>
<th>$I_y$ (kg.m$^2$)</th>
<th>$I_z$ (kg.m$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shaft and spindle</td>
<td>200</td>
<td>25</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Ball 1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Ball 2</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Collar</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0.125</td>
</tr>
</tbody>
</table>

The dynamic analysis of the governor is performed with the initial conditions: the angular speed of the shaft $\omega = 11.0174$ rad/s; the slope of the ball arms is $\frac{\pi}{4}$; and the spring is unloaded at this position. The acceleration of gravity is 9.8 m/s$^2$. An external torque change $T_e$ due to the load driven by the shaft is given as Fig. 5.8. The results with $C = 12500N$ in Eq.(5.78) are compared with those of Haug (1989) in Fig. 5.9 and Fig. 5.10. They are nearly the same. Fig. 5.11 and Fig. 5.12 show the results with different values of $C$. It can be noticed that when $C$ equal to 17500 N, the system is unstable.

![Fig. 5.8 External torque on shaft](image-url)
Fig. 5.9  Position of collar ($k=1000 \text{ N/m}$, $c=30 \text{ N/s}$, $C=12500 \text{ N}$)

Fig. 5.10  Angular speed of shaft ($k=1000 \text{ N/m}$, $c=30 \text{ N/s}$, $C=12500 \text{ N}$)

Fig. 5.11  Position of collar with different values of $C$
5.6.3 Spatial slide crank mechanism

The spatial slide crank mechanism [Jonker, 1988] is shown in Fig. 5.13. The crank axis is in the x-z plane and has an angle of π/4 rad with the global x-axis. The sliding block moves along the x-axis. The length of the crank is 0.15 m and that of the coupler is 0.3 m. The crank and the coupler are made of steel with a density of 7.78x10³ kg/m³ and elasticity modulus of 0.2x10¹² N/m². The radii of their uniform circular cross sections are 0.003 m.

The crank is modelled by a rigid beam element and the coupler is modelled by two beam elements, as marked as B1 and B2 in Fig. 5.13. The crank and the coupler are connected by two hinge elements, whose axes are perpendicular to each other and both are perpendicular to the crank. The connection between the coupler and the sliding block is considered as a spherical joint, that is, there are no elements specified between the coupler and sliding block.

The initial condition is that both the crank and coupler are coincident with the x-axis. When the coupler is considered flexible and the crank rigid, the system is simulated for one revolution at the crank speed of 150 rad/s. The initial values obtained from the periodic solution are shown in Table 5.4 and Table 5.7. Table 5.4 shows the initial deformations calculated from the periodic solutions and Table 5.7 shows the corresponding initial deformation rates. The periodic solutions for the coupler middle point deflection are shown in Fig. 5.14 and Fig. 5.15. Fig. 5.14 shows the deflection velocity component \( v_{xy} \) with respect to \( v_{xy} \). Fig. 5.15 shows the deflection velocity component \( v_{xz} \) with respect to \( v_{xz} \).

The initial values are very important for simulation of dynamic systems. In the literature [Jonker, 1988], [Huag1989], [El-Sawy et al., 1993], [Lieh, 1994], two sets of initial values are normally used, zero initial values, that is, all the initial values are given as zero, or initial values
given by the stationary solution at the starting point, that is the solution calculated from Eq. (5.61). To compare the results with different initial conditions, we give the simulation results with different initial values in Fig. 5.16 and Fig. 5.17. It can be noticed from these two figures that the effects of the initial values on the simulation results are significant.

![Spatial slide mechanism](image)

Fig. 5.13 Spatial slide mechanism

**Table 5.4. Initial deformations from periodic solutions of the spatial slide mechanism**

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon_3$ (m)</th>
<th>$\varepsilon_4$ (m)</th>
<th>$\varepsilon_5$ (m)</th>
<th>$\varepsilon_6$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam 1 (B1)</td>
<td>-0.0010655</td>
<td>-0.0016338</td>
<td>0.0000026</td>
<td>0.0000042</td>
</tr>
<tr>
<td>Beam 2 (B2)</td>
<td>-0.0013511</td>
<td>-0.0007495</td>
<td>0.0000038</td>
<td>0.0000022</td>
</tr>
</tbody>
</table>

**Table 5.5. Initial deformation rates from periodic solutions of the spatial slide mechanism**

<table>
<thead>
<tr>
<th></th>
<th>$\dot{\varepsilon}_3$ (m/s)</th>
<th>$\dot{\varepsilon}_4$ (m/s)</th>
<th>$\dot{\varepsilon}_5$ (m/s)</th>
<th>$\dot{\varepsilon}_6$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam 1 (B1)</td>
<td>-0.0002632</td>
<td>-0.0004616</td>
<td>-0.347964</td>
<td>-0.552996</td>
</tr>
<tr>
<td>Beam 2 (B2)</td>
<td>-0.0003731</td>
<td>-0.0002645</td>
<td>-0.502516</td>
<td>-0.292533</td>
</tr>
</tbody>
</table>
Fig. 5.14 Periodic solutions for deflection component $v_{xy}$ of coupler middle point of the spatial slide mechanism.

Fig. 5.15 Periodic solutions for deflection component $v_{xz}$ of coupler middle point of the spatial slide mechanism.
Fig. 5.16 Middle point deflection component $v_{xy}/l$ of the coupler with different initial values

Fig. 5.17 Middle point deflection component $v_{xz}/l$ of the coupler with different initial values
5.6.4 Spatial slide crank mechanism with flexible joint

For the same spatial slide crank mechanism as used in section 5.6.3, we consider the effect of elastic joints. Keeping the dimensions and parameters the same as in the last section, we consider the driving joint in Fig. 5.13 as a flexible one with joint stiffness \( k = 1000 \, \text{Nm/rad} \). The initial values are calculated from the periodic solutions as shown in Table 5.6 and Table 5.7. The initial elastic deflection of the flexible joint is 0.000556509 rad and initial elastic deflection speed of the flexible joint is -5.30655 rad/s. The simulation results are shown in Fig. 5.18-Fig. 5.23. Fig. 5.18 shows the periodic solution for the elastic deflection of the flexible joint and Fig. 5.19 shows its elastic deflection with respect to the input crank angle. Fig. 5.20 and Fig. 5.21 show the components of the periodic solution for the middle point elastic deflection of the coupler. In Fig. 5.22 and Fig. 5.23, the elastic deflection components of the coupler are presented and they are compared with the results when no joint flexibility is considered.

Table 5.6. Initial deformations from the periodic solution of the spatial slide mechanism with a flexible joint

<table>
<thead>
<tr>
<th></th>
<th>( \varepsilon_3 ) (m)</th>
<th>( \varepsilon_4 ) (m)</th>
<th>( \varepsilon_5 ) (m)</th>
<th>( \varepsilon_6 ) (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam 1 (B1)</td>
<td>-0.0009721</td>
<td>-0.0014910</td>
<td>-0.0000017</td>
<td>-0.0000243</td>
</tr>
<tr>
<td>Beam 2 (B2)</td>
<td>-0.0012379</td>
<td>-0.0006873</td>
<td>-0.0000663</td>
<td>-0.0000468</td>
</tr>
</tbody>
</table>

Table 5.7. Initial deformation rates from the periodic solution of the spatial slide mechanism with a flexible joint

<table>
<thead>
<tr>
<th></th>
<th>( \dot{\varepsilon}_3 ) (m/s)</th>
<th>( \dot{\varepsilon}_4 ) (m/s)</th>
<th>( \dot{\varepsilon}_5 ) (m/s)</th>
<th>( \dot{\varepsilon}_6 ) (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam 1 (B1)</td>
<td>-0.0081005</td>
<td>-0.0110417</td>
<td>-0.0201903</td>
<td>-0.235763</td>
</tr>
<tr>
<td>Beam 2 (B2)</td>
<td>-0.0052994</td>
<td>-0.0019514</td>
<td>-0.628971</td>
<td>-0.440818</td>
</tr>
</tbody>
</table>
Fig. 5.18 Periodic solution for the elastic deflection of the flexible joint

Fig. 5.19 Elastic deflection of the flexible joint
5.6 Numerical examples of dynamic systems

Fig. 5.20 Periodic solution for coupler middle point deflection component $v_{xy}$ of the spatial slide mechanism with flexible joint

Fig. 5.21 Periodic solution for coupler middle point deflection component $v_{xz}$ of the spatial slide mechanism with flexible joint
Fig. 5.22 Elastic deflection component $v_{y}/l$ of coupler middle point, with and without joint flexibility.

Fig. 5.23 Elastic deflection component $v_{xz}/l$ of coupler middle point, with and without joint flexibility.
Chapter 6

Simulation of Manipulator Robots

6.1 Introduction

Simulation or off-line control of a robot uses computers to calculate its necessary conditions for executing a desired end-effector trajectory. This process gives us a good idea about the operation and performances of the manipulators. When flexibility is considered in links or joints of the manipulators, simulation is a powerful tool to understand the effect of the flexibility.

A manipulator robot is mainly composed of mechanical components, a computer, and sensors. The main parts of a robot are the arm, the actuators including the gears and the control computer. The arm is made up of several links, connected usually in series by the joints to form an arm. A link is rotary or prismatic depending on the type of motion caused by the actuator attached to its joint. When the actuator of a joint causes rotational motion, the joint is rotary. When the actuator produces translational motion, the joint is called prismatic. An end-effector, or a gripper, is attached to the arm by means of a wrist. When the joints move, the links connected through joints also move, and so does the end-effector. The positions of the joints determine the configuration of the arm, which places the end-effector at a specific location in the environment. The function of the wrist is to orient the end-effector properly. The motion of the joints produced by actuators determines the position and orientation of the end-effector at any time.

As declared by Kopacek and et al. (1988), elastic flexibility in links and joints is one of the features of the next generation robots. Their arms may be made of new materials or with a greater utilization of the material properties and therefore elastic deformations may occur during the performance of the robot as a consequence of the static and dynamic forces. Nevertheless, the end-effector of the robot has to reach each point in the working space without
overshooting. This requires new advanced control algorithms which can be developed only on the basis of an understanding of the dynamic behaviour.

In recent years, much interest has been focused on the modelling and control of manipulators with flexible links and flexible joints. Several experimental results have been presented in the literature for single-link flexible manipulators [Cannon and Schmitz, 1984], [Hastings and Book, 1986], [Barbieri and Oruguner, 1988], [Kotnik et al., 1988]. And also several experimental results on two-links flexible manipulators have been reported [Bayo, 1988], [Lee et al., 1988], [Oakley and Cannon, 1988], [Schmitz, 1989], [Pfeiffer et al., 1989]. In most of these experiments, a rigid-flex manipulator, i.e., first link rigid and second link flexible, has been studied. In the control problem of flexible manipulators, some convincing contributions are given by Khorasani (1985), An and Atkoson (1986), Kalker and Olser (1987), Spong (1987), Siciliano (1988), Jonker (1990) and Wang and Vidyasagar (1991).

The preceding chapters present the methodology of representing mechanisms and manipulators with finite elements, constructing the equations of motion and performing the numerical analysis for mechanical systems with flexible links and flexible joints. These have given us the basis to simulate the motion of controlled manipulator robots.

For many tasks of a manipulator robot, the end-effector is made to move from a given initial position and orientation to a specified terminal position and orientation. For a spatial manipulator, in order to position and orient the end-effector in the three dimensional space, six degrees of freedom are required: three degrees of freedom for positioning and three degrees of freedom for orientation. While for a planar robot, three degrees of freedom are enough: two degrees of freedom for positioning and one degree of freedom for orientation. The curve that the centre of the end-effector traces in moving from the initial point to the terminal state determines a translational path. The curve that describes the motion of the orientation forms a rotational path. The set of points that specifies the translational and rotational paths of the manipulator end-effector as a function of time is referred to as the trajectory.

The trajectory may be specified by either the Cartesian coordinates or by the generalized coordinates in the joint space. The path descriptions in the two different spaces are related by the kinematic equations. A path of an end-effector can easily be visualized in the Cartesian space. But it is usually difficult to think of the shape of the corresponding path in the joint space. So normally the path is given by the Cartesian coordinates of the end-effector and inverse kinematic analysis is required to calculate the desired position and velocity profile of the actuator joints.

The desired trajectory for the motion of a manipulator robot is generated by a path planner. Alternatively the end-effector of the manipulator can be made to move, for example, manually after the brakes of the joints have been released along the desired path, and the positions of the joints are recorded. Controllers need to be designed for the actuators that make the end-effector
follow the specified trajectory as closely as possible when the desired trajectory with the initial and final positions for the end-effector is given.

In the simulation of a manipulator, we normally use finite elements to represent an actuator joint and we do not have rheonomic constraints so \( q^r \) needs not be considered in Eq(4.26). The vector of the dynamic degrees of freedom \( q \) is the vector of actuator displacement variables \( e^m \) and elastic deformation variables \( e^m \),

\[
q = \begin{bmatrix} e^{mT} & e^{mT} \end{bmatrix}^T
\]

(6.1)

The stress vector \( \sigma \) can be separated as \( \sigma^T = \begin{bmatrix} \sigma^cT & \sigma^mT \end{bmatrix} \) and

\[
F^{eT} \dot{\sigma} = \begin{bmatrix} F^{ecT} & F^{eqT} \end{bmatrix} \begin{bmatrix} \sigma^c \\ \sigma^m \end{bmatrix}
\]

(6.2)

As shown in Eq(4.15), \( F^{eqT} \) is an identity matrix, we have

\[
F^{eT} \dot{\sigma} = F^{ecT} \sigma^c + \begin{bmatrix} \sigma^em \\ \sigma^em \end{bmatrix}
\]

(6.3)

Substituting \( q \) and \( F^{eT} \sigma \) into Eq(4.26), the dynamic model of a manipulator robot can be written as a set of first-order differential equations

\[
\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} f_2(z) \\ B_2(z) \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ B_2(z) \end{bmatrix} u(t)
\]

=: \begin{bmatrix} f(z) + G(z) u \end{bmatrix}

(6.4)

where \( u(t) \) is an \( n \)-dimensional input, with \( n \) indicating the number of actuators,

\[
u = -\sigma^em,
\]

(6.5)

\[
z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} q \\ q \end{bmatrix}
\]

(6.6)

\[
f_2(z) = (F^{eT} M F^{eT})^{-1} \begin{bmatrix} -F^{eT} \sigma \left( M \left( F^{eqT} q \right) q + F^{eqT} \sigma^m + F^{eqT} \sigma^e \right) - F^{ecT} \sigma^c - \begin{bmatrix} \sigma^em \\ 0 \end{bmatrix} \end{bmatrix}
\]

(6.7)

and,
\[ B_2(z) = (P^T M F_q)^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \] 

(6.8)

We may also have an output given as

\[ y = T(z, u). \]

(6.9)

In this chapter, we will study the linearization, inverse kinematics and inverse dynamics, and control design of the systems given by Eq.(6.4) and (6.9). We first introduce the linearization method of the dynamic model. Then we give the inverse analysis and the control design. Finally we present some simulation examples.

### 6.2 Linearization for control systems

With the finite element method we can construct the model of a manipulator. In order to make a system response track a desired trajectory and satisfy certain design specifications, controllers are often designed for the system. General procedures to design controllers such as the classical controllers are based on linear time-invariant equations. Since the dynamical models of most manipulators are nonlinear and complex, a possible approach is to design a primary controller that makes the manipulator motion track the nominal trajectory and a secondary controller that compensates for small variations caused by flexibility and other disturbances acting on the system. The primary controller can be specified by solving the dynamical equations for the input while the secondary controller may be determined on the basis of a model resulting from linearizing the nonlinear model. A linearized model, on one hand, is amenable to controller design, and on the other hand, it is useful in studying the local basic properties of the system such as the complete controllability.

There are normally two ways to linearize a nonlinear system for control, linearization accomplished by determining the variational equations about a nominal trajectory, and feedback linearization by cancelling the nonlinear terms by feedback-feedforward loops [Cesareo and Marino, 1984], [Hunt and Meyer, 1983], [Khorasani and Kokotovic, 1985], [Spong, 1987], [Jonker, 1988, 1990]. We introduce both ways when the whole mechanical system is represented by finite elements.

#### 6.2.1 Variational model around a nominal trajectory

To linearize a system about a nominal trajectory, we write the equations of motion of a manipulator as
where the state $z(t)$ of the system is a 2N-dimensional vector with the initial state $z(t_0)$. The input $u(t)$ is N-dimensional, and $f(z,u)$ is a vector valued function containing all the payload, the inertial forces, the velocity related forces and the input forces. From Eq(6.4), we have

$$f = \begin{bmatrix} f_2(z) \\ f_2(z) \end{bmatrix} + \begin{bmatrix} 0 \\ B_2(z) \end{bmatrix} u(t) \quad (6.11)$$

The nominal trajectory $z_0(t)$ is obtained by applying the nominal input $u_0(t)$ to the system. The variables of the nominal path satisfy the dynamic equations:

$$\dot{z}_0 = f(z_0(t), u_0(t)) \quad (6.12)$$

with the initial values:

$$z_0 = z(t_0) . \quad (6.13)$$

The solution of Eq(6.12) with the initial conditions in Eq(6.13) determines the nominal trajectory. When a trajectory is prescribed, Eq(6.12) is used to solve for the input $u_0(t)$. If the manipulator model in Eq(6.10) is sufficiently accurate, the motion follows the trajectory $z_0(t)$ when the input $u_0(t)$ is applied to the system. Because of flexibility, imperfect modelling, and the effect of disturbances, the motion of the manipulator would deviate from the nominal trajectory that is determined in the trajectory planning stage. Then it is necessary to study the variation versus time from the nominal trajectory and to compensate for the perturbations. To do so, the dynamical system is linearized around the nominal trajectory. Then a controller is designed to compensate the variations.

If the motion of a robot manipulator evolving along the nominal trajectory is perturbed, it results in deviations $(\delta z, \delta u)$ from the nominal trajectory. The perturbed variables also satisfy the equations of motion of the manipulator:

$$\frac{d}{dt}(z_0 + \delta z) = f(z_0(t) + \delta z(t), u_0(t) + \delta u(t)) . \quad (6.14)$$

We expand the right hand side of Eq(6.14) into Taylor's series about the nominal path $(z_0(t), u_0(t))$, neglecting the higher order terms and combine it with Eq(6.12). We get the linearized equations about the nominal trajectory as

$$\delta \dot{z} = \frac{\partial f}{\partial z} \bigg|_{(z_0, u_0)} \delta z(t) + \frac{\partial f}{\partial u} \bigg|_{(z_0, u_0)} \delta u(t) \quad (6.15)$$
with $\delta z(t_0) = 0$.

For a dynamic manipulator model, represented by Eq(4.26) or Eq(5.35), the linearization method has been presented in section 5.3. In section 5.3, we have considered the variables corresponding to the dynamic degrees of freedom, and the elements of the input $u_0(t)$ are considered as external forces. The first order geometric transfer functions of the degrees of freedom of the system, $F^q_{.q}$, is an identity matrix. In the simulation of manipulators, we use finite elements to represent actuators. So the input is given as element stresses. The relationship between the element stresses in Eq(5.39) and the input is

$$u = \begin{bmatrix} -\sigma \end{bmatrix}.$$

So Eq(5.39) can be extended to

$$M_0 \delta \dot{q} + C_0 \delta \dot{q} + K_0 \delta q = \delta u,$$

where $M_0$, $C_0$ and $K_0$ are as given in Eq(5.40), (5.41) and (5.42) respectively and are calculated from the nominal trajectory. Writing Eq(6.17) with state variables, we have the linearized system for control

$$\delta z = A(t) \delta z + B(t) \delta u,$$

where

$$A(t) = \begin{bmatrix} 0 & I \\ -M_0^{-1}K_0 & -M_0^{-1}C_0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ B_2 \end{bmatrix}$$

and $B_2(t)$ is defined by Eq(6.8).

Eq(6.18) represents a linearized dynamical model. It is a linear vector differential equation with time-varying parameters since $(z_0, u_0)$ usually varies with time along the trajectory.

### 6.2.2 Feedback linearization

Another method for linearizing a nonlinear system for control systems is the feedback linearization. The feedback linearization method introduces feedback loops, generated by the available information on nonlinear functions appearing in the model. These feedback terms
can be used to cause a total or partial cancellation of the nonlinear functions of the model by feeding them properly into the system. Additional feedbacks and/or feedforward loops with adjustable gains can also be introduced as controllers into the same system.

For a rigid body manipulator, feedback linearization is essentially the inverse dynamics controller [Hemami and Camana, 1976], [Reboulet and Champetier, 1984], [Spong and Vidyasagar, 1989]. The dynamic degrees of freedom vector $q$ is the vector of the independent deformation variables $e^m$,

$$ q = e^m $$

(6.21)

So for a rigid manipulator robot, Eq(6.4) becomes

$$ \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} z_2 \\ f_2(z) \end{bmatrix} + \begin{bmatrix} 0 \\ B_2(z) \end{bmatrix} u(t), $$

(6.22)

where $u(t)$ is an N-dimensional input,

$$ u = -\sigma^m $$

(6.23)

and

$$ f_2(z) = (F_{em}^T M F_{em})^{-1} \left[-F_{em}^T M (F_{em} e^m) \dot{e}^m + F_{em}^T \dot{f}_n + F_{em}^T \dot{f}_c - F_{em}^T \sigma^c \right], $$

(6.24)

and $B_2(z)$ is an N×N matrix representing the inverse of the inertial matrix in the manipulator dynamics,

$$ B_2(z) = (F_{em}^T M F_{em})^{-1}. $$

(6.25)

It can be noticed that the inverse of $B_2$ exists for all values of $z$ and $t$ in the operating domain.

The model Eq(6.22) is to be linearized by generating the input vector $u(t)$ by means of a new input vector $v(t)$ and the nonlinear functions $H(z,t)$ and $h(z,t)$,

$$ u(t) = H(z,t) v(t) + h(z,t) $$

(6.26)

where the matrix $H(z,t)$ and the vector $h(z,t)$ are to be determined. The terms on the right hand side of Eq(6.26) can be implemented by means of feedforward and feedback loops when the state vector $z$ is available from the measurements.

Substituting Eq(6.26) into Eq(6.22) leads to
\[
\begin{bmatrix}
  z_1 \\
  z_2
\end{bmatrix}
= \begin{bmatrix}
  0 & 1 \\
  A_1 & A_2
\end{bmatrix}
\begin{bmatrix}
  z_1 (t) \\
  z_2 (t)
\end{bmatrix}
+ \begin{bmatrix}
  0 \\
  B
\end{bmatrix} \nu (t)
\]  
(6.27)

where the constant sub-matrices \( A_1, A_2 \) and \( B \) can be chosen to meet certain given control design specifications. For example, they may be chosen as diagonal matrices so that Eq(6.27) can be uncoupled. Then the dynamics of one actuator joint does not depend on the variables of the other joints.

Comparing Eq(6.22) and Eq(6.27), we have

\[
f_2 (z) + B_2 (z) h (z, t) = A_1 z_1 (t) + A_2 z_2 (t)
\]  
(6.28)

and

\[
B_2 (z) H (z, t) = B.
\]  
(6.29)

So the unknown functions are determined as

\[
h (z, t) = B_2^{-1} (z) (A_1 z_1 (t) + A_2 z_2 (t) - f_2 (z)) \quad H (z, t) = B_2^{-1} (z) B
\]  
(6.30)

Substitution of Eq(6.29) and (6.30) into Eq(6.26), the original input vector \( u(t) \) becomes

\[
u (t) = B_2^{-1} (z) (B \nu (t) + A_1 z_1 (t) + A_2 z_2 (t) - f_2 (z))
\]  
(6.31)

The system Eq(6.27) is now a linear one and can be easily controlled. One first applies the nonlinear feedback in Eq(6.31) and then designs a controller based on the linear system in Eq(6.27). The sub-matrices \( A_1 \) and \( A_2 \) can be made to correspond to the acceptable step responses of the systems. The sub-matrix \( B \) is used to adjust the gains in the control matrix. The input \( u(t) \) in Eq(6.31) contains two parts: the term that cancels the nonlinearity in the nonlinear dynamic model, the primary controller, and the remaining terms that can be used to specify an additional controller.

For flexible manipulators, some similar studies have been done. The main concern for feedback linearization is the feedback linearity; i.e., whether it is possible to find a nonlinear feedback to linearize the nonlinear system. For a manipulator with flexible joints, Cesaeo and Marino (1984) showed that feedback linearization of the input-state equations fails for some of their models. But Spong (1987) showed in his model that the dynamic equations are input-state feedback linearizable. The use of input-output feedback linearization has been explored for the case of single flexible links [De Luca and Siciliano, 1989] and planar flexible manipulators [Ding et al., 1989]. Wang and Vidyasagar (1991) examined the feedback linearization for the class of three degrees of freedom manipulators with one flexible link.
There are two feedback linearizations defined for flexible manipulators, input-output feedback linearization and input-state feedback.

The input-output feedback linearization [Isidori, 1985] [Wang and Vidyasagar, 1991] is defined as follows. For a nonlinear system given by Eq(6.4) and output given by Eq(6.9), find a nonlinear state feedback given by Eq(6.26) which transforms the nonlinear relation of Eq(6.4) and (6.9) into a linear relationship between the new input \( v \) and the output \( y \).

The input-state feedback linearization [Hunt et al., 1983] uses a feedback of the form (6.26) in addition to a local state diffeomorphism

\[
y = T(z)
\]

so that the transformed state \( y \) satisfies the linear system

\[
y = Ay + Bv,
\]

where \((A, B)\) is a controllable linear system.

For the feedback linearization of manipulators with flexible joints, we use the method given by Spong (1987). In Spong's method, two assumptions have been made:

(A1) The kinetic energy of the actuator rotor is due mainly to its own rotation. So the motion of the pure rotation is with respect to an inertial frame.

(A2) The rotor/gear inertia is symmetric about the rotor axis of rotation so that the gravitational potential of the system and also the velocity of the rotor centre of mass are both independent of the rotor position.

Most existing models of rigid manipulators are derived under precisely these same assumptions [Paul, 1982], [Spong, 1987].

Now we consider a manipulator consisting of \( n \) flexible joints. Using model II (Fig. 4.5) for the finite element dynamic model of a manipulator given as Eq(4.26), we separate the generalized coordinate vector \( q \) into two parts,

\[
q = \begin{bmatrix} q_1^T & q_2^T \end{bmatrix}^T,
\]

where \( q_1 \) represents all the relative rotating angle variables between the base frame and the connected link or between the two connected links, and \( q_2 \) represents the actuator rotating angles, which are the angles between the rotor and its frame. Using the symbols in section 4.4.2, \( q_1 \) and \( q_2 \) are expressed as

\[
q_1 = \begin{bmatrix} (e_1^1) & (e_1^2) & \ldots & (e_1^n) \end{bmatrix},
\]

\[
q_2 = \begin{bmatrix} (e_2^1) & (e_2^2) & \ldots & (e_2^n) \end{bmatrix}.
\]
\[ q_2 = \begin{bmatrix} (e_1^2)_1, (e_1^2)_2, \ldots, (e_1^2)_n \end{bmatrix}, \quad (6.36) \]

where \( e_1 \) and \( e_2 \) are defined in section 4.4.2 and \( n \) indicates the \( n \) flexible joints.

Then the corresponding stress vector \( \sigma^m \) can be separated as

\[
\sigma^m = \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} = \begin{bmatrix} K(q_1 - q_2) \\ -u \end{bmatrix}, \quad (6.37)
\]

where \( K \) is a diagonal matrix representing the stiffness of the flexible joints. With \( k_i \) as the elastic stiffness of joint \( i \), \( K \) is given as

\[
K = \text{diag}[k_1, k_2, \ldots, k_n]. \quad (6.38)
\]

Using assumptions (A1) and (A2) given in this section, the mass matrix of the dynamic equations of the flexible joint manipulator can also be simplified. As a rotor is just connected with the base and has a pure rotation with respect to an inertial frame, the rotating angle \( e_1^2 \) has only the inertial moment \( J_i \) of the rotor and is not coupled with other inertial properties. So the mass matrix has the form

\[
F^{T}_{\cdot q_1} M F_{\cdot q_1} = \begin{bmatrix} F^{T}_{\cdot q_1} M F_{\cdot q_1} \\ J \end{bmatrix} = \begin{bmatrix} D(q_1) \\ J \end{bmatrix}, \quad (6.39)
\]

where \( J \) is a diagonal matrix composed by the rotor inertial moments \( J_i \)

\[
J = \text{diag}[J_1, J_2, \ldots, J_n]. \quad (6.40)
\]

For consideration of feedback linearization of flexible joint manipulators, we do not consider the flexibility of links at this stage and so the velocity related forces from the flexible links \( f^m \) in the equations of motion are zero. Taking this into consideration and substituting Eq(6.34) - (6.40) into Eq(4.26), we have the dynamic equations for the flexible joint manipulator as

\[
\begin{bmatrix} D(q_1) \dot{q}_1 \\ J \dot{q}_2 \end{bmatrix} = \begin{bmatrix} f_1(q_1, q_1) \\ 0 \end{bmatrix} + \begin{bmatrix} -K(q_1 - q_2) \\ K(q_1 - q_2) \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix}. \quad (6.41)
\]

where

\[
f_1(q_1, q_1) = -F^{T}_{\cdot q_1} M F_{\cdot q_1} \dot{q}_1(q_1, q_1) \dot{q}_1 + F_{\cdot q_1}^{ST} F^N - F_{\cdot q_1}^{ec} \sigma^c \quad (6.42)
\]
Using the linear transformation

\[ z_1 = q_1 \quad z_2 = \dot{q}_1 \]

\[ z_3 = q_2 \quad z_4 = \dot{q}_2 \] \hfill (6.43)

the system can be represented in terms of so-called state equations

\[ \dot{z}_1 = z_2 \] \hfill (6.44)

\[ \dot{z}_2 = (D(z_1))^{-1}(f_1(z_1, z_2) - K(z_1 - z_3)) \] \hfill (6.45)

\[ \dot{z}_3 = z_4 \] \hfill (6.46)

\[ \dot{z}_4 = J^{-1}K(z_1 - z_3) + J^{-1}u \] \hfill (6.47)

It is not obvious that the system is linearized by choosing a suitable nonlinear input \( u \) as in the case of the rigid manipulators. To find the feedback linearization for the nonlinear system, the following nonlinear state space change of coordinates is considered,

\[ y_1 = T_1(z) = z_1 \] \hfill (6.48)

\[ y_2 = T_2(z) = \dot{T}_1 = z_2 \] \hfill (6.49)

\[ y_3 = T_3(z) = \dot{T}_2 = (D(z_1))^{-1}(f_1(z_1, z_2) - K(z_1 - z_3)) \] \hfill (6.50)

\[ y_4 = T_4(z) = \dot{T}_3 = \frac{d}{dt}(D(z_1))^{-1}(f_1(z_1, z_2) - K(z_1 - z_3)) \]

\[ + (D(z_1))^{-1}\left\{ \frac{\partial f_1}{\partial z_1}z_2 + \frac{\partial f_1}{\partial z_2}(D(z_1))^{-1}(f_1(z_1, z_2) - K(z_1 - z_3)) - K(z_2 - z_4) \right\} \]

\[ = f_4(z_1, z_2, z_3) + (D(z_1))^{-1}Kz_4 \] \hfill (6.51)

where \( f_4 \) is defined to include all the terms except the term containing \( z_4 \), that is the last term \( (D(z_1))^{-1}Kz_4 \).

The above mapping \( T \) from \( z \) to \( y \) is a diffeomorphism. Its inverse can be found as

\[ z_1 = y_1 \] \hfill (6.52)

\[ z_2 = y_2 \] \hfill (6.53)
\[ z_3 = K^{-1} \left( D(y_1) y_3 - f_1(y_1, y_2) \right) + y_1, \quad (6.54) \]

\[ z_4 = K^{-1} D(y_1) \left( y_4 - f_4(y_1, y_2, y_3) \right). \quad (6.55) \]

With the above transformation, the desired linearized system is given by

\[
y = \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{bmatrix} y + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v, \quad (6.56)
\]

where \( v \) is the new input to be designed with respect to the linear system, Eq(6.56).

To determine the nonlinear input \( u \), we have

\[
v = y_4 = \frac{\partial f_4}{\partial z_1} z_2 + \frac{\partial f_4}{\partial z_2} (D(z_1))^{-1} (f_1 - K(z_1 - z_2)) + \frac{\partial f_4}{\partial z_3} z_4 +
\]

\[
\quad + \left( \frac{d}{dt} \left( D(z_1) \right)^{-1} \right) K z_4 + \left( D(z_1) \right)^{-1} K (J^{-1} K (z_1 - z_3) + J^{-1} u)
\]

\[ : = h(z) + (D(z_1))^{-1} K J^{-1} u \quad (6.57)\]

where \( h(z) \) includes all the terms except the one containing the input \( u \). So \( u \) is determined as

\[ u = J K^{-1} D(z_1) \left( v - h(z) \right) \quad (6.58)\]

The nonlinear system is now linearized by giving the nonlinear input \( u \). The new input \( v \) can be determined according to the new linear system, Eq(6.56).

In the above transformation, the term \( \frac{d}{dt} \left( D(z_1) \right)^{-1} \) appears in several places. It would be difficult to determine this term without symbolic computation. So in the computer implementation, we avoid using this term. Rewriting Eq(6.50) and suppressing function arguments for brevity, we have

\[ Dy_3 = f_1 - K (z_1 - z_2). \quad (6.59)\]

Taking time derivatives for both sides of the above equation and using Eq(6.51) yields

\[ \dot{D}y_3 + Dy_4 = f_1 - K (z_2 - z_4). \quad (6.60)\]

So we have
6.3 Inverse kinematic solutions

\[ y_4 = D^{-1}\left( f_1 - K (z_2 - z_4) - \dot{D} y_3 \right), \]  \hspace{1cm} (6.61)

where \( f_1 \) and \( \dot{D} \) are calculated as

\[ f_1 = \frac{\partial f_1}{\partial z_2} z_2 + \frac{\partial f_1}{\partial z_2} \dot{z}_2. \]  \hspace{1cm} (6.62)

\[ \dot{D} = \frac{\partial D}{\partial z_2} \dot{z}_2. \]  \hspace{1cm} (6.63)

6.3 Inverse kinematic solutions

For a given task, the end-effector of a manipulator is often controlled so that it should follow a desired trajectory determined by the planning in terms of the Cartesian coordinates. Usually, the control of the manipulator is performed in the joint space, so these values of the Cartesian coordinates need to be converted to the values of the joint variables. In particular, when the position and orientation of the end-effector are specified by the base coordinate systems, the values of the joint variables that give rise to the specified location of the end-effector in the work-space must often be determined for control purposes. This is the inverse kinematic problem. The kinematic equations are used in the backward direction.

Given the desired trajectory of the end-effector as \( x_0^p, \dot{x}_0^p \) and \( \ddot{x}_0^p \), the inverse kinematics determine the values of the actuator joints \( q_0, \dot{q}_0 \) and \( \ddot{q}_0 \). From Eq(4.11), we have geometric transfer functions as

\[ x_0^p = F_p (q_0) \]  \hspace{1cm} (6.64)

Differentiating Eq(6.64) with respect to time, we derive the velocity and acceleration vector as

\[ \dot{x}_0^p = F_{p,q} \dot{q}_0 \]  \hspace{1cm} (6.65)

\[ \ddot{x}_0^p = F_{p,q} \ddot{q}_0 + F_{p,qq} \dot{q}_0 \dot{q}_0 \]  \hspace{1cm} (6.66)

When no elastic flexibility is considered in the manipulator robot, the dynamic degrees of freedom of the robot are given as the joint coordinates \( q_0 \). The number of dynamic degrees of freedom should be equal to the number of coordinates of the end-effector. For example, for a spatial robot the trajectory has three translational degrees of freedom and three degrees of freedom describing the orientation of the end-effector, hence there should be six joint coordinates to be used. From Eq(6.65) and Eq(6.66), we can calculate the velocity and acceleration vectors of the joint coordinates as
\[
\dot{q}_0 = (F_{p,q}^p)^{-1} \dot{x}_0^p \\
(6.67)
\]

\[
\dot{q}_0 = (F_{p,q}^p)^{-1} (\ddot{x}_0^p - F_{q,q}^p \dot{q}_0 \dot{q}_0) \\
(6.68)
\]

Here the first and second geometric transfer functions \(F_{p,q}^p\) and \(F_{q,q}^p\) are functions of the joint coordinates. The determination of the values of the joint coordinates \(q_0\) is not so direct. From Eq(6.64), we have

\[
\Delta x_0^p = F_{p,q}^p \Delta q_0 \\
(6.69)
\]

and

\[
\Delta q_0 = (F_{p,q}^p)^{-1} \Delta x_0^p. \\
(6.70)
\]

Though the above expressions for the inverse kinematics are the same as used by Jonker (1988), it can be noticed from Eq(5.18) and (5.19) that they are not restricted to the finite element approach and have a more general application.

With the relationship given by Eq(6.70), the nominal values of the actuators can be calculated by using the Newton-Raphson method as we did in section 5.2.

In practical applications, the position and velocities of the end-effector are not so easily obtained because of the elastic deformations. So one may like to know the compensation needed from the actuator joints, when elastic deformations occur. If we know the deformations of the flexible links or flexible joints, the necessary compensation from the actuator joints can be determined from the above equations. We separate the independent degrees of freedom \(q\) into

\[
q = \begin{bmatrix} e^m \\ \varepsilon^m \end{bmatrix}
\]

with \(e\) representing the displacements of the actuator and \(\varepsilon\) representing the elastic deformations. Eq(6.69) becomes

\[
\Delta x_0^p = F_{p,e}^p \Delta e^m + F_{p,\varepsilon}^p \Delta \varepsilon^m. \\
(6.71)
\]

To compensate the deviation of the end-effector caused by the flexibility, the needed actuator joint values and velocities are

\[
\Delta e^m = -(F_{p,e}^p)^{-1} F_{p,\varepsilon}^p \Delta \varepsilon^m, \\
(6.72)
\]
\[ \Delta \dot{e}^m = -(F_{\theta}^p)^{-1} F_{\theta}^p \Delta \dot{e}^m. \] (6.73)

6.4 Primary and secondary controller design for manipulators

The manipulator control problem for the gross motion is a servo problem in which the end-effector is made to track the reference values representing the desired trajectory. The inputs to the manipulator system can be chosen as the generalized torques produced by the joint actuators, and the outputs to be controlled are the positions, possibly velocities and accelerations of the joints when the control is performed in the joint space. The actual positions and velocities of the joints in a manipulator can be measured and compared with their desired values using feedback loops to form the position and velocity errors for driving the system.

The actuator joints should have appropriate nominal values in order to make the end-effector of a manipulator track the desired nominal trajectory. These values are calculated from the ideal conditions, without consideration of the flexibility and other errors. The controller generating these values is referred to as the primary controller. It compensates for the nonlinear effects in the actuators, and attempts to cancel the nonlinear terms in the model. The errors caused by an un-exact mathematical model, by disturbances, by flexibility and by undesirable deviations can be corrected by means of an additional controller called a secondary controller. The total control system is shown in Fig.6.1.

![Diagram](image)

**Fig. 6.1** Manipulator system with primary and secondary controllers

6.4.1 Primary controller design

With the prescribed trajectory as a function of time, the nominal displacements, their velocities and accelerations of the actuators are known. Then the nominal input \( u_0(t) \) can be
calculated directly from the equations of motion. The force vector $f$ in Eq(4.26) can be separated as

$$ F^T_{\cdot q} f = \begin{bmatrix} F^T_{\cdot x q} & F^T_{\cdot e c q} & F^T_{\cdot e m q} \\ -\sigma^c_{\cdot e} \\ -\sigma^e_{\cdot m} \end{bmatrix} \begin{bmatrix} f^x \\ -\sigma^c \\ -\sigma^e \end{bmatrix} \quad (6.74) $$

Because the first order geometric transfer function of the degrees of freedom is an identity matrix, by using Eq(4.26), Eq((6.5)) and Eq(6.21), we have the nominal input $u_0(t)$ as

$$ u_0 = -F^T_{\cdot x q} (f^x_0 - M(x_0) \dot{x}_0) + F^T_{\cdot e c q} \sigma^c_0 \quad (6.75) $$

where $f^x_0$ includes all the external forces and the velocity dependent forces of all the nodal points. The subscript 0 indicates the values at the nominal trajectory. The accelerations are determined by

$$ \ddot{x}_0 = F^x_{\cdot q 0} \ddot{q}_0 + (F^x_{\cdot q q 0} \dot{q}_0) \dot{q}_0. \quad (6.76) $$

The calculation of the input $u_0$ is an algebraic solution from the nonlinear dynamic equations Eq(6.75) when the desired values of the actuator joints have been calculated by inverse kinematics.

For a rigid manipulator, the input $u_0$ in Eq(6.75) has actually cancelled the nonlinear terms in the nonlinear dynamic model. When the model is accurate and there are no disturbances, the manipulator should follow the prescribed trajectory. Also if a flexible manipulator model is linearized by the variational method around the nominal trajectory, the input $u_0$ in Eq(6.75) is used as the primary controller.

When the feedback linearization method described in Section 6.2.2 is used for the linearization of the nonlinear model, the nonlinear feedback input should be determined by linearization, for example, the primary controller of a manipulator with flexible joints may be given as Eq(6.58). With the prescribed trajectory $z_0$, the primary controller should give the nominal input $u_0$ as

$$ u_0 = JK^{-1} D(z_{10}) (v_0 - h(z_0)), \quad (6.77) $$

where

$$ v_0 = \dot{y}_{40}. \quad (6.78) $$

Under ideal conditions, the primary controller in Eq(6.77) causes the exact cancellation of the nonlinear terms in the nonlinear equations. But because of flexibility and other
disturbances, errors always occur. To compensate for the errors and the effects of other disturbances, a secondary controller is needed.

6.4.2 Secondary controller design

Because the nominal input is calculated by considering the manipulator as a rigid one and because there are other disturbances acting on the system, a secondary controller is necessary to complement the primary controller and to compensate for the errors caused by the flexibility and other disturbances. If the flexibility and the disturbances are small enough so that the disturbed motion can be expected to stay in a small neighborhood of the nominal desired trajectory, the linearized model should suffice to describe this motion for the controller design.

The secondary controller design is based on the linearized systems given by Eq(6.18), (6.56) for flexible manipulator robots, or (6.27) for rigid ones. Eq(6.27) and (6.56) are time-invariant systems while Eq(6.18) is normally time-variant systems.

For the linearized system given by Eq(6.56), to compensate the deviation from the desired trajectory \( y_0(t) \), the compensating input \( \delta v \) can be determined by the following simple linear control law designed to track a desired trajectory \( y_0(t) \)

\[
\delta v = -K(y - y_0),
\]

where the regulator gain matrix \( K \) is determined by the eigenvalues of the matrix \( A - BK \). From Eq(6.56), \( A \) and \( B \) are given as

\[
A = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]

and

\[
B = \begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix}
\]

Then the total linear input \( v \) is

\[
v = v_0 + \delta v,
\]
and the total nonlinear $u$ is calculated from Eq(6.77).

The linearized model Eq(6.18) is a time-variant system. Generally, a time-variant linear system should be controlled by some type of a time-variant linear feedback or by an adaptive controller. But as indicated by Pfeiffer and Kleemann (1989), time-invariant output control realizes a performance which is not so far away from time-variant concepts. There are several ways to determine the constant control gain matrices. We can consider the dynamics of some trajectory points and design the feedback matrices at each point, and take the best one which suits all the trajectory points considered. Or we can choose an averaged matrix from various path points. Another method is to design an output controller for the end point. For a system having elastic flexibility both at links and joints, we used the output controller based on the dynamics of the end point introduced by Pfeiffer and Kleemann (1989). The output feedback is evaluated as

$$
\delta u = -K_1 \delta e^m - K_2 \delta \dot{e}^m - K_3 \delta e^m - K_4 \delta \dot{e}^m.
$$

(6.83)

The gain matrices $K_1$, $K_2$, $K_3$ and $K_4$ are determined in three steps. The diagonal elements of the joint control, $K_1$ and $K_2$, yield stiffness and damping in each joint similar to PD (Proportional and Derivative) control of each axis and have the greatest influence on the system properties [Pfeiffer and Kleemann, 1989]. The first step is to determine these diagonal elements of $K_1$ and $K_2$. Then $K_3$ and $K_4$ are determined by an iterative procedure to feedback the measurements of each elastic deformation to the corresponding joint torque. In the third step, the remaining control parameters are chosen to meet the desired eigenvalues of the linearized system.

When some simplification is performed on the linearized system Eq(6.17), for example, only the mass matrix $M_0$ is considered to be time-variant, the control law can be given as [Jonker, 1988, 1990]

$$
\delta u = -K_p \delta q - K_v \delta \dot{q} - K_{pt} \delta \dot{x}^p - K_{vt} \delta \dot{x}^p
$$

(6.84)

where $\delta q$, $\delta \dot{q}$, $\delta \dot{x}^p$ and $\delta \dot{x}^p$ are the variations of the actuator values and end-effector values and their velocities respectively, and

$$
K_p = M_0 \Omega^2
$$

(6.85)

$$
K_v = 2M_0B\Omega
$$

(6.86)

$$
K_{pt} = M_0\Omega^2 (F_{\dot{q}0})^{-1}
$$

(6.87)

$$
K_{vt} = 2M_0B\Omega (F_{\dot{q}0})^{-1}
$$

(6.88)
\( \Omega, B \) and \( B_i \) are diagonal matrices containing the desired servo loop frequencies and the corresponding active damping ratios. \( M_0 \) is given by Eq(5.43) and \( F_{q0}^{up} \) is the nominal value of the first order geometric transfer function of the end-effector. It can be noticed that the gain matrices \( K_p, K_v \) and \( K_i \) are time-variant as \( M_0 \) is a time-variant matrix.

### 6.5 Simulation examples

In this section, we give some simple simulation examples which have been used in the literature. First a planar two link rigid-flexible manipulator is simulated. Then we simulate a one link manipulator with flexible joint. Finally the simulation of a spatial manipulator will be given.

#### 6.5.1 Two link rigid-flexible manipulator

The planar, two link rigid-flexible manipulator (Fig. 6.2) was used by Yigit (1994) to study the stability of PD (proportional plus derivative) controllers. The first link is assumed to be rigid and the second link is composed of a slender flexible link, cantilevered onto a rigid rotating hub. The length of both link 1 and link 2 is 0.5 m and the length of the hub is 0.04 m. The mass per unit length of link 1 is 2 kg/m and that of link 2 is 0.15 kg/m. The hub has a lumped mass of 0.5 kg and an inertial moment of 0.002 kgm\(^2\) at the joint. The bending stiffness of link 2 is 1 Nm\(^2\).

![Two link rigid-flexible manipulator](image)

We use rigid beam elements to represent the first link and the hub and two flexible beam elements to represent the flexible link. Only the bending flexibility is considered. The two actuators, actuator 1 and actuator 2, are represented by two planar hinge elements, hinge 1 (with relative rotation \( \epsilon_1 \)) and hinge 2 (with relative rotation \( \epsilon_2 \)) respectively. We use a simple independent joint PD controller to simulate the set point tracking, that is, by specifying the final
positions of the manipulator. The input is given as

\[ u = K(z - z_i) \]  \hspace{1cm} (6.89)

where \( z \) is the vector of actuator joint variables and their rates

\[ z = \begin{bmatrix} e^1_i & e^2_i & \dot{e}^1_i & \dot{e}^2_i \end{bmatrix}^T \]  \hspace{1cm} (6.90)

and the elements of \( z_f \) are the final values, given by

\[ z_f = \begin{bmatrix} 1 & 0.5 & 0 & 0 \end{bmatrix} . \]  \hspace{1cm} (6.91)

\( K \) is the feedback gain matrix. For an independent joint feedback control, \( K \) is given as

\[ K = \begin{bmatrix} k_{11} & 0 & k_{13} & 0 \\ 0 & k_{22} & 0 & k_{24} \end{bmatrix} . \]  \hspace{1cm} (6.92)

The components are chosen as

\[ k_{11} = 5, k_{13} = 2, k_{22} = 5, k_{24} = 2, \]  \hspace{1cm} (6.93)

with eigenvalues \( s_1, 2, 3, 4 = -1 \pm 2i \), corresponding to the rigid system.

The simulation of the joint angles of the set point tracking are shown in Fig. 6.3. The actuator joint torque inputs are shown in Fig. 6.4. The elastic deflection of the end-point is shown in Fig. 6.5.

![Fig. 6.3 Joint angles of set point tracking of the two rigid-flexible link manipulator](image-url)
6.5 Simulation examples

Fig. 6.4 Actuator torque inputs of the two rigid-flexible link manipulator

Fig. 6.5 Elastic deflection of the end-point of the two rigid-flexible link manipulator

6.5.2 Flexible joint manipulator

The second simulation example is a simple one link manipulator with a flexible joint. The properties of the manipulator are: length of the link is 1 m, the inertial moments of the rotor and the link are both 1 kgm², a lumped mass of 1 kg is at the end of the link with its gravity force taken into account, and the stiffness of the elastic joint is 100 Nm/rad.

The link is expected to track the trajectory of sin(8t). At the initial position, the link is at the −z direction. We use Model II described in Chapter 4 to represent the flexible joint. With \( e^1 \) representing the rotating angle of the link and \( e^2 \) representing the rotating angle of the actuator,
we have

\[ z = \left[ e_1^1 \ e_1^2 \ e_1^3 \ 0 \right]^T. \]  

(6.94)

Using the feedback linearization given by Eq(6.48)-(6.51) and the input given by Eq(6.58), the system is simulated. For this example, \( A \) in Eq(6.80) becomes

\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]  

(6.95)

and \( B \) is

\[
B = \begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix}^T.
\]  

(6.96)

The regulator gain matrix \( K \) in Eq(6.79) is determined by getting a closed loop characteristic polynomial for the linearized system, Eq(6.79), of \( (s+10)^4 \) (\( s \) is the variable of the Laplace transform), as used by Spong (1987). So \( K \) is determined as

\[
K = \begin{bmatrix}
10000 & 4000 & 600 & 40
\end{bmatrix}.
\]  

(6.97)

The simulation results are shown in Fig. 6.6-Fig. 6.9. It can be noticed from Fig. 6.9 that even though the elastic deflection is comparatively very large, (larger than nominal value), the simulation results follow the desired trajectory very well.
Fig. 6.6  Comparison of desired trajectory and simulation results of flexible joint manipulator

Fig. 6.7  Actuator rotating angle of manipulator with flexible joint

Fig. 6.8  Input torque for the actuator of the manipulator with flexible joint
One step further, we include the dynamics of the electrical machine (separately excited d.c. motor) of the actuator. So there is one more degree of freedom of the system and an additional equation given as Eq(3.93). Now the input is the voltage of the motor. The parameters of the motor are: equivalent inertial moment $I=1 \text{ kgm}^2$, total gear ratio $n=60$, inductance $L=0.0048 \text{ H}$, resistance $R=1.6 \Omega$, voltage constant $k_v=0.19 \text{ Vs/rad}$, torque constant $k_t=0.26 \text{ Nm/A}$. The closed loop characteristic polynomial for the linearized system is $(s + 10)^5$. The feedback gain matrix is now

$$K = \begin{bmatrix} 100000 & 50000 & 10000 & 1000 & 50 \end{bmatrix}.$$  \hspace{1cm} (6.98)

Fig. 6.10 Compare of desired trajectory and simulation results of flexible joint manipulator with motor dynamics considered
The simulation results are shown in Fig. 6.10 to Fig. 6.13. Fig. 6.10 shows the time history of the rotating angle of the robot arm compared with its desired trajectory. The rotating angle of the actuator and the elastic deflection of the flexible joint are shown in Fig. 6.11. The time history of the actuator torque, that is integrated from the input voltage, is shown in Fig. 6.12 and the time history of the input voltage is shown in Fig. 6.13.

![Graph showing actuator rotating angle and elastic deflection](image1)

**Fig. 6.11** Actuator rotating angle and elastic deflection of flexible joint with dynamics of motor considered

![Graph showing actuator torque](image2)

**Fig. 6.12** Actuator torque with dynamics of motor considered
6.5.3 Spatial manipulator with a flexible joint and flexible links

A spatial manipulator is shown in Fig. 6.14. The geometrical configuration of the manipulator is shown in Table 6.1, it is the same as the one analysed by Jonker (1988, 1990). But here we have the flexible joint. The lumped mass at the end-effector is 40 kg and at the joint between link 1 and link 2 there is a mass of 10 kg. The equivalent inertial moment of the rotor of actuator 3 is 72 kgm².
The joint at actuator 3 is considered as a flexible one with a torsional stiffness of 20000 Nm/rad and a damping ratio of 15 Nm/s/rad. The torsional stiffness of link 2 is also 20000 Nm/rad.

<table>
<thead>
<tr>
<th></th>
<th>link 1</th>
<th>link 2</th>
<th>link 3</th>
<th>link 4, 5, 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>length (m)</td>
<td>0.7</td>
<td>0.7</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>mass (kg/m)</td>
<td>2.0</td>
<td>4.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>EI_y, EI_z (Nm²)</td>
<td>20000</td>
<td>20000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EI_x (Nm²)</td>
<td>20000</td>
<td>20000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The desired trajectory for the end-effector is a straight line from point \((1.0, 0, 0)\) to point \((0, 0.6, 0)\) with an acceleration of \(a_0(1 - \cos \omega t)\) and deceleration of \(-a_0(1 - \cos \omega (t - t_1))\), with 0.2 seconds of acceleration and deceleration time, as shown in Fig. 6.15.

![Fig. 6.15 Acceleration of the end-effector coordinates](image)

The simulation is performed by considering the flexibility of the joint at actuator 3, the bending flexibility of link 1, the torsional and bending flexibility of link 2. The control law introduced in Eq(6.84) is used with

\[
\Omega = \text{diag } \begin{bmatrix} 20 & 20 & 20 \end{bmatrix}, \tag{6.99}
\]
\[ B = \text{diag} \begin{bmatrix} 0.5 & 0.5 & 0.05 \end{bmatrix} \]  

(6.100)

\[ B_f = \text{diag} \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix} \]  

(6.101)

The simulation starts with some initial errors from the desired values of 0.01 m, 0.01 m for actuator 1 and actuator 2 respectively and -0.01 radian for actuator 3. The simulation results, compared with their nominal values, are shown in Fig. 6.16 to Fig. 6.25.

---

**Fig. 6.16 Desired and simulation results of end-effector coordinates**

**Fig. 6.17 Desired and simulation of actuator position**
Fig. 6.18 Desired and simulation of actuator position of actuator 3

Fig. 6.19 In-plane elastic deformation, $\varepsilon_3$ and $\varepsilon_4$, of link 1

Fig. 6.20 Out-of-plane elastic deformation, $\varepsilon_5$ and $\varepsilon_6$, of link 1
Fig. 6.21 In-plane elastic deformation, $\varepsilon_3$ and $\varepsilon_4$, of link 2

Fig. 6.22 Out-of-plane elastic deformation, $\varepsilon_5$ and $\varepsilon_6$, of link 2

Fig. 6.23 Elastic deflection of flexible joint and torsional deformation of link 2
Fig. 6.24 Nominal and simulation results of actuator forces of actuator 1 and actuator 2

Fig. 6.25 Nominal and simulation results of torsion moment of actuator 3
Bibliography


[89] Pfeiffer, F., and Kleemann, U., "Elasticity and vibration control for manipulators," in


Numerieke simulatie van mechanismen en robotmanipulatoren met vervormbare schakels en geledingen

Samenvatting


Een geïntegreerd netwerk van programma's en een gegevensbank is toegepast in het programma-systeem SPACAR. De FORTRAN-taal wordt vanwege haar rekendoelmatigheid in de meeste berekeningen gebruikt en de C-taal en X Winbdje worden vanwege hun rijke en samen- gestelde omgeving voor de programmeurs en gebruikers van toepassingsgerichte programma's gebruikt. Het basisidee, de structuur van het ontwerp en de grondslagen waarmee het programma-systeem ontworpen is, worden ingevoerd. Daar de objectgeoriënteerde benadering de jongste tendentie in de programmerpraktijk is, is het objectgeoriënteerde programmeren voor het SPACAR-systeem ook uiteengezet.

De algemene theorie van elektrische machines is toegepast om de dynamische analyse van elektrische machines uit te voeren. Gebruik makend van deze theorie kunnen we allerlei elektrische machines die tot een algemene elektrische machine getransformeerd kunnen worden, analyseren. Het combineren van deze theorie met de eindige-elementenvoorstelling van mechanische systemen resulteert in een krachtig gereedschap voor de computersimulatie van mechanismisch-elektrische systemen. Vereenvoudigde modellen zijn afgeleid voor de analyse van het stationair gedrag van elektrische machines. Twee linearisatiemodellen zijn afgeleid, de een rond de stationaire oplossing en de ander voor de algemene dynamische vergelijkingen voor het onderzoek van de stabilité. Daar dikwerf gelijkstoommotoren in regelsystemen worden gebruikt, zijn de dynamica en inverse dynamica van verschillende gelijkstoommotoren ingevoerd.

De twee modellen voor vervormbare geledingen die in dit proefschrift uitgewerkt worden, kunnen gebruikt worden om vele verschillende soorten vervormbare geledingen voor te stellen. Model I kan als een directe voorstelling van het mechanische systeem beschouwd worden. Model II heeft enige voordelen als het voor de simulatie van robots met vervormbare geledingen gebruikt wordt.

Het invoeren van de starre balkelementen heeft twee voordelen. Enerzijds kan het de rekendoelmatigheid verbeteren. Een merkbare vermindering van rekentijd is gevonden wanneer deze starre balkelementen voor starlichaamssystemen gebruikt worden, zowel in de kinematische als in de dynamische analyses van mechanische systemen. Anderzijds sluit het dichter bij het fysische model van een star lichaam aan, omdat slechts een stel oriëntatiecoördinaten (een rotatieknoop) in het model gebruikt wordt.

De methode van het terugbrengen van het aantal coördinaten die in de eindige-elementenbenadering gebruikt is, is gegeneraliseerd om differentiaal-algebraïsche vergelijkingen op differentiaalvergelijkingen terug te voeren. Het implementeren van de methode in het SPA-CAR-systeem maakt het voor de gebruiker zelf mogelijk om nevenvoorwaarden te gebruiken. Een automatische linearisatie van een niet-lineair systeem is ingevoerd en de gelineariseerde resultaten kunnen gebruikt worden voor het regelaarontwerp, voor de stabiliteitsanalyse, voor een periodieke oplossing en voor het berekenen van de belangrijkste eigenfrequenties. De inschietmethode wordt voor het vinden van de periodieke oplossing van een periodiek aangedreven mechanisch systeem gebruikt. Voorbeelden hebben aangetoond dat het gebruik van de periodieke oplossing als beginvoorwaarden voor de dynamische simulatie voor periodiek aangedreven systemen erg belangrijk is.

Voor de simulatie van robots met vervormbare schakels en vervormbare geledingen zijn twee linearisatiemodellen bestudeerd: de variationele linearisatie en het terugkoppelingsregelingsmodel. Numerieke voorbeelden voor simulatie van robots met vervormbare schakels en vervormbare geledingen zijn getoond om de bevalligheid en de kracht van de ingevoerde methoden en het ontworpen programmasysteem aan te tonen.
Acknowledgements

First of all, I would like to thank my supervisor Professor dr. ir. J.F. Besseling for his invaluable guidance, stimulating discussions and continuous encouragements in the whole research and critical review of the manuscript.

Grateful thanks go to Professor dr. ir. P. Meijers for his continuous support and encouragements from the day I arrived in The Netherlands to date and his meticulous reading of the manuscript.

I owe many debts of gratitude to dr. ir. J.P. Meijaard for his helpful discussions of my research, the careful reading of the manuscript and the translation of the summary of the thesis into Dutch.

I would like to express my gratitude to all the colleagues in the Laboratory for Engineering Mechanics for sharing the facilities, translating Dutch materials and assisting in all aspects.

I am greatly indebted to my wife Xiaode Yu for her understanding, support and encouragements, and helpful discussions on control theory. I am much obliged to my son, with whom I should have spent more time together in the evenings and weekends.
Curriculum Vitae

Diguang GONG was born in Hunan Province, China, on January 28, 1957. In 1982, he received B.Sc. in Engineering from Hunan Agricultural University, Changsha, China. Then he worked in Department of Agricultural Engineering (Farm Machinery) at Hunan Agricultural University from 1982 to 1986. He worked and studied together during this period and received M.Sc. in Engineering from Jiangsu Institute of Technology, Zhenjiang, China. He worked in Department of Mechanical Engineering at Changsha Communications Institute from 1987 to 1989 and was appointed as an assistant professor in 1988. At the fall of 1989, he joined the Laboratory for Engineering Mechanics of Department of Mechanical Engineering and Marine Technology at Delft University of Technology as a visiting research fellow. In July 1989, he embarked on his research on the numerical simulation of mechanical systems towards a Ph.D at the same laboratory under the supervision of Prof. dr. ir. J.F. Besseling.