Modelling and Validation of Robot Manipulators

A thesis in fulfillment of the requirements of the degree of M.Sc

Xiufeng Li

Faculty of Engineering
Department of Mechanical Engineering
University of Glasgow

April 1989

Acknowledgements

I wish to express my gratitude to my supervisor, Professor P.J. Gawthrop, for the thesis subject suggestion, for the invaluable support and guidance over the course of this research, and for the superior environment and the distinctive working method that made me quickly evolve into the role.

I would also like to thank Dr. A.R. Whittaker for the experimental data and inspiring ideas.

I also thank Dr. J. Howell for showing me how to use ACSL (Advanced Continuous Simulation Language).

My appreciation is extended to the staff of the Department of Mechanical Engineering, especially Mrs. L. McCormick, for the convenient working conditions.

I would like to express my appreciation to my fellow students of the Department of Mechanical Engineering, especially Mr. H. Demircioglu and Mr. H. Mirab.

I wish to express my thanks to Higher Education Bureau of GuangDong province and Professor Henry H. Wong for the opportunity of studying at the University of Glasgow.

Finally, I wish to express my thanks to my parents for unfailing support and encouragement to seek a higher education.

This thesis is dedicated to my wife.

Summary

There are many methods to describe manipulator dynamics, the iterative Newton-Euler dynamic formulation and the Lagrange-Euler formulation are two of them. Between these two well known methods, the former has been regarded as computationally efficient, and the latter as understandable in representing manipulator dynamics. It is hard and dull to generate robot manipulator dynamic equations manually from either the iterative Newton-Euler dynamic formulation or the Lagrange-Euler formulation. Therefore, the two general programmes, which are based on these two formulations respectively and suited to rotary joint manipulators, have been written in REDUCE. After running the programmes, we find that the calculation time for generating the dynamic equations of a rotary joint manipulator by the programme based on the Lagrange-Euler formulation is much shorter than the one by the programme based on the other.

Robot manipulator dynamic equations are a set of differential ones. Therefore, the simulation of the motion of a rigid manipulator belongs to mathematical modelling problems. It requires numerical integration.

Robot validation is a new area of research. To model the general dynamic characteristics of flexible links and to examine the friction effects on DC motors are a contribution of this work.

We have presented four different analytic methods to set up the dynamic model of one flexible link manipulators. The method based on vibration theories to generate the exact model, which is described by transfer functions, is efficient. There are a number of simplifications used in deriving the dynamic equations of one-link flexible manipulators by three other approximate methods: one is derived in time domain, one is derived by using ‘complete set of functions’ and the Lagrange-Euler formulation, and one is derived by adding same stiffness springs to connect equal length rigid parts to approximate a flexible link. However, the corresponding methods are found to be reasonable by comparing the frequency responses of the models obtained by them with the one obtained by experimental test.

The friction effects on DC motors include two parts: one is the static effect which causes the system to have a positive or a negative constant resistant torque which depends on the rotation direction of the motors shafts, and the other is the dynamic effect which causes the system to have a friction torque which is proportional to the rotation speed. These friction effects have been verified by the system parameters identified by Least-squares method.

In summary, the manipulator dynamics generation programmes, the simulation method and the robot validation work are shown to be satisfactory. Computer simulation is efficient for the preliminary research on robot manipulators.

Contents

Table of Contents

Chapter 1. Introduction

1. Introduction

1.1. Motivation

An industrial robot manipulator is a mechanical device whose purpose is to enable its end point — equipped with a gripper or a tool — to follow a desired trajectory in order to perform a given task. The manipulator can generally be thought as a chain of structurally rigid links interconnected by rotary or sliding joints, each joint can be controlled by its own actuator.

It is an efficient way for the researchers who study in the field of mechanical manipulators to work with a simulated manipulator by using computers. Craig's book [1] and Fu's book [2] describe a basic approach to set up the dynamic equations of a manipulator by using the iterative Newton-Euler dynamic formulation. Nicosia [3] gives a method to obtain manipulator dynamic equations by using the Lagrange-Euler formulation. To generate manipulator dynamic equations from both of these two formulations by hand is dull and arduous. Is it possible to solve the problem by machines? The SAM (System Algebraic Manipulation) language MACSYMA [4] and REDUCE [5] can give a positive answer.

For today's manipulators, the position of the end point is controlled by "dead-reckoning" that is, by commanding the appropriate joint-angles derived through a real-time inverse manipulator kinematics, and then assuming that the links are stiff enough so that the end point will be automatically in the intended location.

Instead of this conventional dead-reckoning method, a new concept of end point position feedback is developed resently for the control of flexible manipulators. The new concept is that the position of the end point is sampled directly by a sensor whose output is then fed back to the joint actuators with a proper servo compensation. End point sensing has two main advantages over the dead-reckoning.

First, it improves the static and dynamic position accuracy of the end point through feeding back the quantity to be controlled to the actuators.

Second, with end point sensing, the links do not need rigid any more. The manipulator can be built with light links. The moments of inertia at each joint are smaller so that smaller actuators can be used or higher performing speed can be achieved. Also it becomes possible to use direct drives instead of gear driven motors, with the advantages of manufacturing simplicity, actuator linearity and lower cost.

Lighter links will cause the system to be more flexible, even to vibrate. Therefore, to realise the characteristics of flexible links is placed on agenda.

Robot validation is a newly developed research aspect. It deals with the problems of searching and arranging theoretical methods to explain practical phenomena. The existed do not satisfy robot developments. One example is that, according to classical dynamics, manipulator designers must face the problem of faster and faster performance of its end point demanded by working environments against the heavier and heavier devices chosed by safety requirements. To realise the characteristics of flexible links and to search the friction effects on DC-motors are based on this idea.

1.2. Literature

Most research in the field of flexible manipulators has focused on the dynamic modelling aspects. Relatively fewer references are on the control design aspects.

Much of the literature has been devoted to producing algorithms to model open-loop chains of rigid and elastic bodies. This thesis first deals with the problem of modelling rigid link manipulators, later develops the idea of adding springs into rigid links to approximate the real flexible link of manipulators.

Book [6] [7] applies a transfer matrix method to describe the elastic bending motion of a two-link planar elastic arm in the frequency domain, for a given relative configuration of the link and for small angular velocities.

Schmits [8] and Skaar [9] use a fourth order partial differential equation to develop the transfer functions for one flexible link manipulators. The bound conditions are chosen as the same as the ones of cantilevered beams'. They all only give the definitions of the transfer functions.

Nicosia [3] provides the Lagrange energy method to derive multi-link flexible manipulator dynamic equations in time domain. He establishes two generalised (or Lagrangian) coordinates: one denotes the relative displacement between connected links when the elastic deformation is neglected, the other characterises the shapes of the links. The "complete set of function" is used to approximate the dynamic equations.

To our knowledge, most of the authors have considered proportional and derivative (PD) joint angle feedback. With a PD feedback for each joint, Book [6] shows that the maximum attainable closed loop bandwidth for a two-link planar manipulator is 0.5 Ω , where Ω is the system first vibration frequency with both joints locked.

Maizza-Neto [10] discusses the use of a pole placement algorithm to obtain full-state feedback gains for a 12th order linear model of the same two-link planar manipulator as Book's.

More detailed references are given in the appropriate chapters.

1.3. Thesis outline

This thesis is about the establishment of dynamic equations of rotary joint manipulators, the simulation of the dynamic equations and the validation of robot manipulators.

The first part of the thesis is on the aspects of using REDUCE to set up manipulator dynamic equations. Both the iterative Newton-Euler dynamic formulation and the Lagrange-Euler formulation are described and applied to illustrative examples. A big part of the thesis is on the research of robot validation. Some vibration theories are used for obtaining the transfer functions of one-link flexible manipulators. The simulated one-link elastic beams' natural frequencies correspond to the theoretical calculated ones and the ones from the experiment. After that, the basic idea of adding springs to approximate real flexible links is developed. Another validation work is to realise the friction effects on DC motors. In the discussion and conclusion part, we discuss the possibility of the application of the validation work to the control of manipulators.

To review the Chapters briefly, the programmes in REDUCE for generating the dynamic equations of rotary joints manipulators are presented in chapter 2. Chapter 3 describes a method to simulate the motion of rigid manipulators using SIMNON language. The characteristics of flexible links are presented and simulated in chapter 4 and chapter 5. Chapter 6 presents the friction effects on DC-motors. The final Chapter, Chapter 7, gives the discussion and the conclusion of the thesis.

Chapter 2. Dynamic Equations of Rigid Manipulators

2. Dynamic Equations of Rigid Manipulators

2.1. Introduction

Manipulator dynamics deals with the mathematical formulations of equations of robot arm motion. The dynamic equations of motion of a manipulator are a set of mathematical equations describing the dynamic behaviour of the manipulator. Such equations are useful for computer simulation , the design of suitable control strategies, and the evaluation of robot arm kinematic design and structures.

Formulation of manipulator dynamics in relation to computational efficiency and control analysis has been an active research topic. Between the two well-known formulations, the iterative Newton-Euler dynamic formulation and the Lagrange-Euler formulation, the former has been regarded as computationally efficient, and the latter as perceptible in representing manipulation dynamics. However, the comparison of computational efficiency has been based on the premise that the dynamic equations have been expressed in vector / matrix form and a numerical approach used to solve the joint forces (or torques). If the vector / matrix equations were expanded symbolically to scalar form, the expanded scalar equations from these two formulation would be equivalent. The expanded scalar equations would not only provide insight into understanding of the system dynamics, but also result in faster computation than the numerical approach based on either of the formulations. The saving computation cost would be quite substantial if most of the matrices in the system equations were sparse [11]. The matrices in manipulator dynamic equations are usually sparse.

The objective of this chapter is to provide a systematic methodology for the dynamic equations of rigid manipulators. It is divided into four sections, Introduction, Dynamics, Programmes, and Conclusion.

In the dynamic section, two formulations are used to present the equations of motion for a manipulator. It is shown how the equations for position, velocity, acceleration of the link coordinates can be used in the Newton-Euler forward and backward recursive equations of motion of a free rigid body to obtain the model of an open-chain manipulator. It also gives the Lagrange-Euler formulation (or the Lagrange energy method) to get the torque and motion equations.

The programmes section gives two general programmes, which are based on the two formulations described in the following section, for calculating the torques acting on rotary manipulators.

2.2. Dynamics

There are two aspects which relate to manipulator dynamics. In the first, the trajectory of each joint and its first and second derivatives are given, the required torque on the joint can be found. This dynamic formulation is useful for manipulator controls. The second deals with the approach to obtain the motion of the joints forced by given torques. This is useful for manipulator dynamic simulations.

2.2.1. Iterative Newton-Euler dynamic formulation

All manipulators can be classified into one of two categories: those that contain closed kinematic loops and those that do not (open-chain mechanisms). In this section, we only discuss the cases of open-chain type manipulators.

The complete algorithm for computing joint torques from the motion of the joints is composed of two parts. First, link velocities and accelerations are iteratively computed from link 1 out to link n and the Newton-Euler equations are applied to each link. Second, forces and torques of interaction and joint actuator torques are computed recursively from link n back to link 1. The equations are summarised below for general rigid manipulators.

Considering the problem of computing the torques that correspond to a given trajectory of a manipulator, we assume the position, velocity, and acceleration of the joints, q, $\dot{q}$, $\ddot{q}$, are known. With this knowledge and the one of the kinematics and mass distribution information of the manipulator, we can calculate the joint torques which cause the particular motion [1] [2].

Forward equations: i = 1, 2, ..., n, to compute the link motions (kinematics).

If link i is rotational

$$\omega^i{}_i = R^i{}_{i-1}(\omega^{i-1}{}_{i-1} + Z_0 \dot{q}_i), \tag{2-1}$$

$$\dot{\omega}^i{}_i = R^i{}_{i-1}[\dot{\omega}^{i-1}{}_{i-1} + Z_0 \ddot{q}_i + \omega^{i-1}{}_{i-1} \times Z_0 \dot{q}_i], \tag{2-2}$$

$$\dot{v}^i{}_i = \dot{\omega}^i{}_i \times p^{i*}{}_i + \omega^i{}_i \times (\omega^i{}_i \times p^{i*}{}_i) + R^i{}_{i-1}\dot{v}^{i-1}{}_{i-1}, \tag{2-3}$$

$$a^i{}_i = \dot{\omega}^i{}_i \times s^i{}_i + \omega^i{}_i \times (\omega^i{}_i \times s^i{}_i) + \dot{v}^i{}_i. \tag{2-4}$$

If link i is translational

$$\omega^i{}_i = R^i{}_{i-1}\omega^{i-1}{}_{i-1}, \tag{2-5}$$

$$\dot{\omega}^i{}_i = R^i{}_{i-1}\dot{\omega}^{i-1}{}_{i-1}, \tag{2-6}$$

$$\dot{v}^i{}_i = R^i{}_{i-1}(Z_0\ddot{q}_i + \dot{v}^{i-1}{}_{i-1}) + \dot{\omega}^i{}_i p^{i*}{}_i + 2\omega^i{}_i \times (R^i{}_{i-1}Z_0\dot{q}_i) +$$

$$\omega^i{}_i \times (\omega^i{}_i \times p^{i*}{}_i), \tag{2-7}$$

$$a^i{}_i = \dot{\omega}^i{}_i \times s^i{}_i + \omega^i{}_i \times (\omega^i{}_i \times s^i{}_i) + \dot{v}^i{}_i. \tag{2-8}$$

Backward equations i = n, n-1, ..., 1, to compute the joint torques corresponding to link motions (dynamics).

$$F^i{}_i = m_i a^i{}_i, \tag{2-9}$$

$$f^i{}_i = R^i{}_{i+1}f^{i+1}{}_{i+1} + F^i{}_i, \tag{2-10}$$

$$n^i{}_i = R^i{}_{i+1}(n^{i+1}{}_{i+1} + p^{i+1*}{}_i \times f^{i+1}{}_{i+1}) + (p^{i*}{}_i + s^i{}_i) \times F^i{}_i +$$

$$I^i{}_i\omega^i{}_i + \omega^i{}_i \times (I^i{}_i\omega^i{}_i), \tag{2-11}$$

If link i is rotational

$$\tau = (n^i{}_i)^T (R^i{}_{i-1} Z_0) + b_i \dot{q}_i \tag{2-12}$$

If link i is translational

$$\tau = (f^i{}_i)^T (R^i{}_{i-1} Z_0) + b_i \dot{q}_i \tag{2-13}$$

Terminology

n = number of degrees of the manipulator

q = n × 1 vector of joint variable positions

$\dot{q}$ = n × 1 vector of joint variable velocities

$\ddot{q}$ = n × 1 vector of joint variable accelerations

$R^i{}_j$ = 3 × 3 transformation matrix for link jth coordinates reference to link ith coordinates

$\omega^i{}_i$ = 3 × 1 vector, angular velocity of link ith coordinates

$\dot{\omega}^i{}_i$ = 3 × 1 vector, angular acceleration of link

$v^i{}_i$ = 3 × 1 vector, linear velocity of link ith coordinates

$\dot{v}^i{}_i$ = 3 × 1 vector, linear acceleration of link ith coordinates

$a^i{}_i$ = 3 × 1 vector, linear acceleration of link ith mass center

$f^i{}_i$ = 3 × 1 vector, force exerted on link i by link i-1

$n^i{}_i$ = 3 × 1 vector, moment exerted on link i by link i-1

$F^i{}_i$ = 3 × 1 vector, total force exerted on link i

$N^i{}_i$ = 3 × 1 vector, total moment exerted on link i

m_i = total mass of link i

$s^i{}_i$ = position vector from the link ith mass center to the origin of the coordinate system (x_i, y_i, z_i)

$p^{i*}{}_i$ = the origin of the ith coordinate system with respect to the (i-1)th coordinate system

$I^i{}_i$ = inertia matrix of link i about its mass center with reference to base coordinate system (x_0, y_0, z_0)

Z_i = vector $(0\ 0\ 1)^T$

The effect of gravity loading on the links can be included quite simply by setting $\dot{v}^0_0 = G$, where G is the gravity vector. This is equivalent to saying that the base of the robot is accelerating upward with 1G acceleration. This fictitious upward acceleration causes exactly the same effect on the links as gravity would. So, with no extra computational expense, the gravity effect is calculated.

It is often convenient to express the dynamic equations of a manipulator in a single equation which hides the details, but shows some of the structure of the equations.

When the Newton-Euler equations are evaluated symbolically for any manipulator$, they yield the dynamic equations which can be written in the form:

$$\tau = M(q)\,\ddot{q} \;+\; Q(q,\dot{q}), \tag{2-14}$$

where M(q) is the $n \times n$ mass matrix of the manipulator, $Q(q,\dot{q})$ is an $n \times 1$ vector of centrifugal, Coriolis and gravity terms.

2.2.2. Lagrange-Euler formulation

The general motion equations of a manipulator can conveniently be expressed through the direct application of the Lagrange-Euler formulation to nonconservative systems. Many investigators utilise the Denavit-Hartenberg [12] matrix representation to describe the special displacement between the neighbouring link coordinate frames to obtain the link kinematics information, and they employ the Lagrangian dynamics technique to derive the dynamic equation of a manipulator. The direct application of the Lagrangian dynamics formulation, together with the Denavit-Hartenberg link coordinate representation, results in a convenient and compact algorithmic description of the manipulator equations of motion. The algorithm derived from the Lagrange-Euler equation is expressed by matrix operations and facilitates both analysis and computer implementation.

The derivation of the dynamic equations of an n degrees of freedom manipulator is based on the understanding of the Lagrange-Euler equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i}\right) = \tau_i \qquad i = 1, 2, ..., n \tag{2-15}$$

where

L = Lagrange function = kinetic energy K - potential energy P

K = total kinetic energy of the robot arm

P = total potential energy of the robot arm

q_i = generalised coordinates of the robot arm

$\dot{q}_i$ = first derivative of the generalised coordinates, q_i

τ_i = generalised force (or torque) applied to the system at joint i to drive link i

From the above Lagrange-Euler equation, one is required to properly choose a set of *generalised coordinates* to describe the system. Generalised coordinates are used as a convenient set of coordinates which completely describe the location (position and orientation) of a system with respect to a reference coordinate frame. For a simple manipulator with rotary joints, since the angular positions of the joints are ready available because they can be measured by potentiometers or encoders or other sensing devices, they provide a natural correspondence with the generalised coordinates. This in effect, corresponds to the generalised coordinates with the joint variable defined in each of the 4×4 link coordinate transformation matrices.

According to the book [2], the total kinetic and potential energy of a robot manipulator are:

$$K = \sum_{i=1}^{n}\sum_{p=1}^{i}\sum_{r=1}^{i} Kc_{ipr}(q)\dot{q}_p\dot{q}_r \qquad (2\text{-}16)$$

and

$$P = \sum_{i=1}^{n} - m_i g\bar{r}^0{}_i \qquad (2\text{-}17)$$

where $Kc_{ipr}(q)$ is a function of q, n is the link number, m_i is the ith link mass, g is a gravity row vector expressed in the base coordinate system, $\bar{r}^0{}_i$ is a vector from the origin of the base coordinate frame to the ith link mass center and expressed in the base coordinate system.

From the Lagrange-Euler equation and the kinetic and potential energy equations, we derive the motion equation in the form

$$D(q)\ddot{q} + H(q,\dot{q})\dot{q} + C(q,\dot{q}) = \tau \qquad (2\text{-}18)$$

where

$D(q)$ is an $n\times n$ inertial acceleration-related symmetric matrix whose elements are

$$d(i,j) = d(j,i) = \frac{\partial^2 K}{\partial\dot{q}_i\partial\dot{q}_j} \qquad (2\text{-}19)$$

$H(q,\dot{q})$ is an $n\times n$ nonlinear centrifugal and Coriolis force vector-related matrix whose elements are

$$h(i,j) = \frac{\partial^2 K}{\partial\dot{q}_i\partial q_j} \neq \frac{\partial^2 K}{\partial q_i\partial\dot{q}_j} = h(j,i) \qquad (2\text{-}20)$$

$C(q,\dot{q})$ is an $n\times 1$ gravity loading force vector whose elements are

$$c(i) = -\frac{\partial(K - P)}{\partial q_i} \qquad (2\text{-}21)$$

Generally, the total manipulator's kinetic and potential energies can be symbolically evaluated

$$K = \sum_{i=1}^{N} K_i; \qquad P = \sum_{i=1}^{N} P_i.$$

where

$$K_i = \frac{1}{2}\int_0^{L(i)} \rho_i (\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) dl$$

$$P_i = g \int_0^{L(i)} \rho_i Z_i dl$$

ρ_i = *mass per unit length of the ith link*

$L(i)$ = *length of ith link*

So far, the each link's torque of a manipulator can be calculated in symbolic form from the equations described above. It takes long time to find the torque equations by hand. Moreover, the model we chose may change. In order to simplify this stage work, we write programmes in REDUCE to generate the torque equations symbolically.

2.3. Programmes

Manual symbolic expansion of manipulator matrix equations is tedious, time-consuming, and error-prone. The equations generation process consists of many vector / matrix manipulations, and the generated equations may consist of hundreds of terms. Automatic derivation of equations using a computer is desirable even for simple manipulators. The expense of computation is justifiable considering the elimination of the manual derivation process and the saving of computation costs in the later numerical computation phase.

Various computer programmes for deriving manipulator dynamic equations have appeared in the literature. Some of them are written in general programming languages such as FORTRAN [13] or PL/I [14]; other are written in Lisp-based symbolic algebra languages such as MACSYMA [15], Both the Newton-Euler and the Lagrange-Euler formulations have been used for the equation derivation.

2.3.1. Introduction to REDUCE

REDUCE is a system for carrying out symbolic algebraic operations accurately, no mater how complicated the expressions become. It can manipulate polynomials in a variety of forms, both expanding and extracting various parts of them as required. There are many other functions in REDUCE such as MATRIX CALCULATIONS, PROCEDURES.

REDUCE is designed to be an interactive system, so that the user can input an algebraic expression and see its value before moving on to the next calculation. REDUCE can also be used in batch mode by inputing a sequence of calculations and getting results without the necessity of interaction during the calculations.

Now we introduce the programmes for calculating the torques of rotational manipulators using the iterative Newton-Euler dynamic formulation and the Lagrange-Euler formulation respectively.

2.3.2. The general programmes for calculating the torques acting on rotational manipulators

After considering the structure of the iterative Newton-Euler dynamic formulation, the Lagrange-Euler formulation and the functions provided by REDUCE, we have written the general programmes for calculating the torques of rotational manipulators.

2.3.2.1. Link coordinate position and orientation

The link coordinate is the position of the joint variable. For rotational joint, its position is measured in radians. To each link of the manipulator is attached a right-handed coordinate system composed of three orthogonal unit vectors. These coordinate systems are called link coordinates, and their position and orientation are defined in terms of 4×4 link coordinate transformation matrices.

One particularly suitable method for assigning link coordinates is attributed to Hartenberg and Denavit [11]. In this method four parameters are used to describe the position of successive link coordinates, Figure 2.1. The parameters are a, α, d and θ. The definitions of these parameters are:

a_i = *the shortest distance between* Z_i and Z_{i-1}

α_i = *the angle between* Z_i and Z_{i-1}

d_i = *the shortest distance between* X_i and X_{i-1}

θ_i = *the angle between* X_i and X_{i-1}

Only one of these four parameters is variable and is denoted by q_i. For rotational joints manipulators, θ_i is the joint variable and d_i, a_i, and α_i are constants.

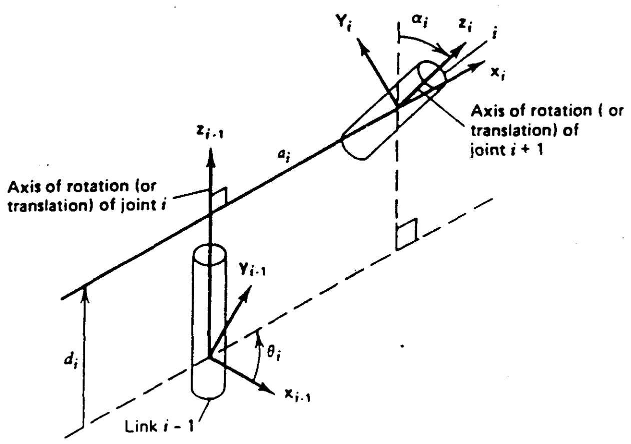

Y_i
α_i
z_i
i
x_i
Axis of rotation (or translation) of joint $i+1$
z_{i-1}
a_i
Axis of rotation (or translation) of joint i
Y_{i-1}
θ_i
d_i
x_{i-1}
Link $i-1$

Fig.2.1 Parameters relating adjacent link coordinate systems.

2.3.2.2. Notes for the two programmes

Firstly, we introduce the programme based on the iterative Newton-Euler dynamic formulation. This programme is written in REDUCE. It can calculate all sorts of rotational manipulators' torques. We write it in the form of procedure. The variables are all in matrix form.

In the programme, everything from the symbol % to the end of the line on which it appears is ignored as comments. The programme can be divided into three parts: the part which defines and inputs variables before the procedure, the outward recursive loop part which calculates kinematics, and the inward recursive loop part which computes dynamics in the procedure. In order to correspond with the symbols in the Newton-Euler equations, we choose the variables symbolically.

Secondly, the programme based on the Lagrange-Euler formulation has some differences from the one above. The main programme in PROCEDURE form is not valid for IBM 3090. Therefore, the main programme is written in the form a sequence of commands.

Trying to write the programme efficiently and clearly, we write the Vector-CrossProduct function and Transformation function in PROCEDURE form. The variables are all meaningful words.

It is difficult to say which programme is better. The Newton-Euler one has a clear outline. One can check any calculation steps when the programme is running. Another advantage is that it is conducive to the design of a manipulator because of knowing the kinematics. The Lagrange-Euler one is more efficient to generate the dynamic equations, because it does not need to calculate the acceleration and the force of each link.

2.3.2.3. Advantages of symbolic expansions of dynamic equations

By expanding the vector / matrix dynamic equations symbolically, insights on the dynamics of a manipulator can be generated in two ways:

(1) examining directly the terms of the dynamic equations, and

(2) using the dynamic equations to simulate individual force components.

Another merit of expanding the vector / matrix equations of motion is that, if most manipulator links are symmetric in geometry, the resultant equations are more computationally efficient even than the efficient iterative Newton-Euler dynamic formulation of manipulator dynamics in vector / matrix form.

This section has presented the techniques and programmes for deriving the scalar form of manipulator dynamic equations by symbolically expanding the iterative Newton-Euler dynamic formulation and the Lagrange-Euler formulation using REDUCE. The automatic equation derivation process is highly desirable because it not only eliminates the time-consuming, error-prone manual derivation process, but also generates equations which are both perceptible and more computational efficient than the numerical approach.

2.3.3. An example of closed form dynamic equations

Now we adopt the equations which are described in 2.2 to generate the torque equations in two ways: one is by hand, the other is by running the programmes.

2.3.3.1. The torques are calculated by hand

Here we first compute the closed form dynamic equations for the 2-link planar manipulator shown in Figure 2.2. For simplicity, we assume that the mass distribution is extremely simple: all mass exists as a point mass at the distal end of each link. These masses are m1 and m2.

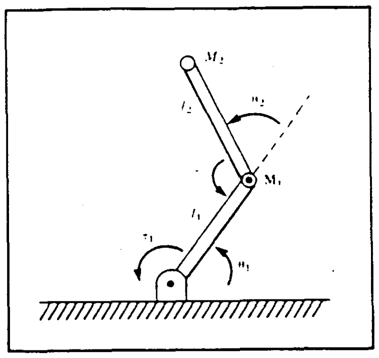


FIGURE 2.2 Two-link with point masses at distal end of links.

By using Newton-Euler dynamic formulation, the joint torques are calculated:

$$\tau_1 = m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 C_2(2\ddot{\theta}_1 + \ddot{\theta}_2) + (m_1 + m_2) l_1^2 \ddot{\theta}_1 - m_2 l_1 l_2 S_2 \dot{\theta}_2^2$$

$$- 2m_2 l_1 l_2 S_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 l_2 g C_{12} + (m_1 + m_2) l_1 g C_1 \qquad (2\text{-}22)$$

$$\tau_2 = m_2 l_1 l_2 C_2 \ddot{\theta}_1 + m_2 l_1 l_2 S_2 \dot{\theta}_1^2 + m_2 l_2 g C_{12} + m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) \qquad (2\text{-}23)$$

From these and equation (2.1) we can get M and Q:

$$M(\theta) = \begin{bmatrix} l_2^2 m_2 + 2l_1 l_2 m_2 C_2 + l_1^2(m_1 + m_2) & l_2^2 m_2 + l_1 l_2 m_2 C_2 \\ l_2^2 m_2 + l_1 l_2 m_2 C_2 & l_2^2 m_2 \end{bmatrix}$$

$$Q(\theta,\dot{\theta}) = \begin{bmatrix} - m_2 l_1 l_2 S_2 \dot{q}_2^2 - 2m_2 l_1 l_2 S_2 \dot{q}_1 \dot{q}_2 + m_2 l_2 g C_{12} + (m_1 + m_2) l_1 g C_1 \\ m_2 l_1 l_2 S_2 \dot{q}_1^2 + m_2 l_2 g C_{12} \end{bmatrix}$$

This example is taken from the book "Introduction to Robotics Mechanics & Control" by John J. Craig. The purpose of doing this is to check the results of the programmes.

2.3.3.2. The torques obtained by running the programmes

The torques of the manipulator which is described in 2.3.3.1 can also be obtained by running the programmes. The form $M(\theta)$ and $Q(\theta,\dot{\theta})$ are:

$M(1,1) = 2m_2l_1l_2\cos\theta_2 + m_1l_1^2 + m_2(l_1^2 + l_2^2)$

$M(1,2) = m_2l_1l_2\cos\theta_2 + m_2l_2^2$

$M(2,1) = m_2l_1l_2\cos\theta_2 + m_2l_2^2$

$M(2,2) = m_2l_2^2$

$Q(1) = -m_2l_1l_2(2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2\sin\theta_2 + m_2l_2g\cos(\theta_1 + \theta_2) + (m_1 + m_2)l_1g\cos\theta_1$

$Q(2) = m_2l_1l_2\dot{\theta}_1^2\sin\theta_2 + m_2l_2g\cos(\theta_1 + \theta_2)$

and

$d(1,1) = 2m_2l_1l_2\cos\theta_2 + m_1l_1^2 + m_2(l_1^2 + l_2^2)$

$d(1,2) = m_2l_1l_2\cos\theta_2 + m_2l_2^2$

$d(2,1) = m_2l_1l_2\cos\theta_2 + m_2l_2^2$

$d(2,2) = m_2l_2^2$

$h(1,1) = -2m_2l_1l_2\dot{\theta}_2\sin\theta_2$

$h(1,2) = -m_2l_1l_2\dot{\theta}_2\sin\theta_2$

$h(2,1) = m_2l_1l_2\dot{\theta}_1\sin\theta_2$

$h(2,2) = 0$

$c(1) = m_2l_2g\cos(\theta_1 + \theta_2) + (m_1 + m_2)l_1g\cos\theta_1$

$c(2) = m_2l_2g\cos(\theta_1 + \theta_2)$

After having compared the results of M and Q with the ones in 2.3.3.1, we find the results are correct. Moreover, the programmes are very efficient. It only takes several minutes to obtain the results.

We have also verified the examples described in the book "Robotics: control, sensing, vision, and intelligence" by K.S. Fu, R.C. Gonzales and C.S.G. Lee.

2.4. Conclusion

Two different formulations and the related REDUCE programmes for robot arm dynamics have been presented. The iterative Newton-Euler dynamic formulation is very efficient, and the Lagrange-Euler formulation has a well structured form. After running the two programmes, we find the one derived from the Lagrange-Euler formulation is more efficient because it does not need to calculate the acceleration and the force of each link.

Chapter 3. Simulation of Rigid Manipulators

3. Simulation of Rigid Manipulators

3.1. Introduction

To model a system is to replace it by something which is (a) simpler and / or easier to study, and (b) equivalent to the original in all important respects. If the real system interacts with the outside world in some way, that interaction must be reflected in the model. The (simplified) logical equivalent is subjected to the same, or similar, external stimuli as the original. It then produces outputs which may be interpreted as the system's reaction to the stimuli (see figure 3.1). Thus, by varying the model inputs and examining the corresponding outputs, one attempts to study the behaviour of the real system.

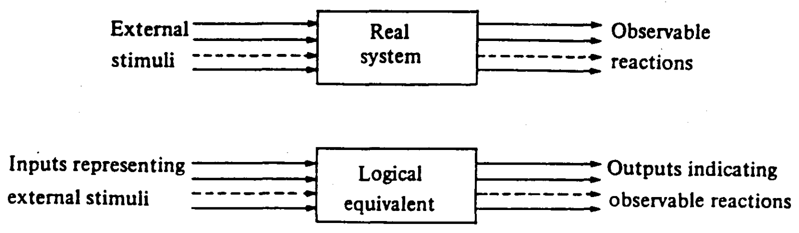

External stimuli
Real system
Observable reactions
Inputs representing external stimuli
Logical equivalent
Outputs indicating observable reactions

Fig. 3.1 Simulation process

The first and most basic distinguishing feature of a model is the nature of the logical equivalent used. That may be a physical system, in which case we talk of physical modelling, or it may be a set of abstract variables whose behaviour is controlled by a number of assumptions and equations; then the model is said to be mathematical.

Physical modelling belongs primarily, although not exclusively, to the engineering domain: applications rang from car and ship design, through aircraft testing in wind tunnels, to the training of astronauts in centrifuges. The mathematical modelling methods can be divided into 'analytical' and 'numerical', depending on the approach to the solution. An analytical solution provides a closed-form expression for the desired system characteristics in terms of the defining parameters. The numerical methods are divided further into 'deterministic' and 'stochastic'. The terms 'deterministic' and 'stochastic' refer, respectively, to the absence or presence of random variables in the model. These may, or may not, reflect the absence or presence of random phenomena in the system being modelled [16].

The robot manipulator dynamics described in chapter 2 are structured in differential equations. The simulation of these equations belongs to mathematical modelling.

SIMNON is a special language for solving difference and differential equations and for simulating dynamical systems. The systems may be described as interconnection of subsystems whose behaviour are characterised by differential equations. Models of this type are common in mathematics, biology, economics and in many branches of engineering, especially in robot manipulators. SIMNON has an interactive implementation which makes it easy for a user to work with the system. The user interacts with the system by typing commands. Parameters, initial conditions and system descriptions can be modified interactively. The results are displayed as curves on the screen. The layout can be easily modified and the results can be documented using a hard copy facility.

The main characteristic of SIMNON is that it can be used in a very simple way to find solutions to difference and differential equations. So it can be used as a tool to simulate robot manipulator dynamic equations.

This chapter presents a method which specially suits robot manipulator dynamics simulation. The dynamics of a three-link manipulator, which is calculated by running the programmes in chapter 2, is simulated by using SIMNON. Several different cases are discussed in the simulation.

3.2. Simulation method

In order to design a controller, one needs to model a system first. Robot dynamic simulation should enable the real performance of a robot to be reproduced. We use the simulation method to simulate a simplified three-link manipulator which is described in Fig. 3.3.

To simulate the motion of a manipulator, for example the first three links of MA3000, we could make use of the dynamic model which can be obtained by running the programmes. Given the dynamics written in closed form as in equation (2-14), the most common way of simulating the motion is to solve for the acceleration (which involves inverting $M(q)$):

$$\ddot{q} = M(q)^{-1} [\ \tau - Q(q,\dot{q})\ ] \tag{3-1}$$

We may then apply any known numerical integration techniques to integrate the equations forward in time. SIMNON provides a software for solving difference and differential equations. By using SIMNON, we can calculate the joint output q, $\dot{q}$, and $\ddot{q}$ when we know the torque input τ and initial conditions. Usually the initial conditions are:

$$q(0) = q_0, \quad \dot{q}(0) = 0, \quad \ddot{q}(0) = 0 \tag{3-2}$$

The above process can be summarised in the block diagram:

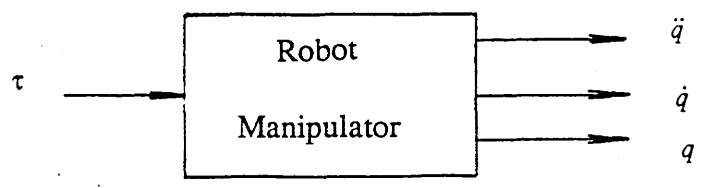


Fig. 3.2 Input τ, output $q, \dot{q}, \ddot{q}$.

The method to perform numerical integration is fixed in the SIMNON software. Integration is a discrete numerical calculation process. The accuracy is depend on the size of time interval Δt. It should be sufficiently small that breaking continuous time into these small increments is a reasonable approximation. SIMNON can choose Δt automatically and manually during the simulation process. So, the simulation accuracy can be controlled by setting a reasonable small time interval Δt.

3.3. The model of three-link manipulator

The example of the two-link planar manipulator shown in figure 2.2 is too simple to imply the power and convenience of the method for the simulation of rigid manipulators. So, we apply this method to simulate a three-link manipulator which is shown in figure 3.3 by using SIMNON language.

3.3.1. Dynamic equations

We have run the programme which is based on the iterative Newton-Euler dynamic formulation to generate the dynamic equations of the three-link manipulator.

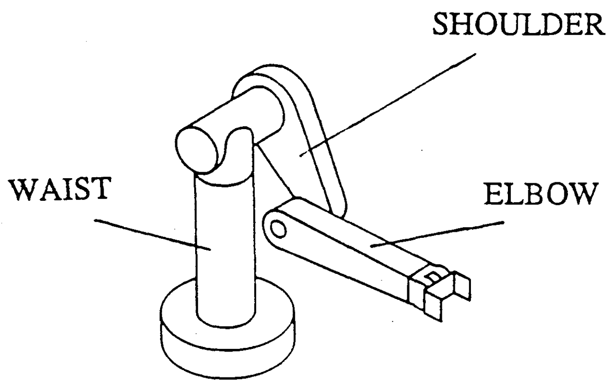


Fig. 3.3 Three-link manipulator

A feature of REDUCE is that it can handle a mixture of symbolic and numerical values. This is illustrated by running the procedure 'NewtonEuler' firstly with symbolic values for the masses and link lengths, and secondly with numerical values.

a) Using symbolic values:

In our programme, the variables are in the general symbolic form except the alpha, alpha = $(\frac{\pi}{2}, 0, 0)^T$, because we know it.

The torques of each link's joint which are in the form of $M(q)$, $Q(q,\dot{q})$ can be obtained, where $q = (\theta_1, \theta_2, \theta_3)^T$, $\dot{q} = (\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)^T$ are vectors of θ and $\dot{\theta}$.

$M(1,1) = ((m_3 l_3^2 - 4I_{3x} + 4I_{3y})\cos 2(\theta_2 + \theta_3) + (m_2 l_2^2 + 4m_3 l_2^2 - 4I_{2x} + 4I_{2y})\cos 2\theta_2 + 4m_3 l_2 l_3 \cos\theta_3 + 4m_3 l_2 l_3 \cos(2\theta_2 + \theta_3) + m_2 l_2^2 + 4m_3 l_2^2 + m_3 l_3^2 + 8I_{1y} + 4I_{2x} + 4I_{2y} + 4I_{3x} + 4I_{3y}) / 8$

$M(1,2) = 0$

$M(1,3) = 0$

$M(2,1) = 0$

$M(2,2) = (4m_3l_2l_3\cos\theta_3 + m_2l_2^2 + 4m_3l_2^2 + m_3l_3^2 + 4I_{2z} + 4I_{3z}) / 4$

$M(2,3) = (2m_3l_2l_3\cos\theta_3 + m_3l_3^2 + 4I_{3z}) / 4$

$M(3,1) = 0$

$M(3,2) = (2m_3l_2l_3\cos\theta_3 + m_3l_3^2 + 4I_{3z}) / 4$

$M(3,3) = (m_3l_3^2 + 4I_{3z}) / 4$

$Q(1,1) = -((m_3l_3^2 - 4I_{3x} + 4I_{3y})(\dot\theta_2 + \dot\theta_3)\dot\theta_1\cos2(\theta_2 + \theta_3) + 2m_3l_2l_3(2\dot\theta_2 + \dot\theta_3)\dot\theta_1\sin(2\theta_2 + \theta_3)$
$+ (m_2l_2^2 + 4m_3l_2^2 - 4I_{2x} + 4I_{2y})\dot\theta_1\dot\theta_2\sin2\theta_2 + 2m_3l_2l_3\dot\theta_1\dot\theta_3\sin\theta_3) / 4$

$Q(2,1) = ((m_3l_3^2 - 4I_{3x} + 4I_{3y})\dot\theta_1^2\sin2(\theta_2 + \theta_3) + 4m_3l_2l_3\dot\theta_1^2\sin(2\theta_2 + \theta_3) -$
$4m_3l_2l_3(2\dot\theta_2 + \dot\theta_3)\dot\theta_3\sin\theta_3 + 4m_3l_3g\cos(\theta_2 + \theta_3) + 4(m_2 + 2m_3)l_2g\cos\theta_2) / 8$

$Q(3,1) = ((m_3l_3^2 - 4I_{3x} + 4I_{3y})\dot\theta_1^2\sin2(\theta_2 + \theta_3) + 2m_3l_2l_3\dot\theta_1^2\sin(2\theta_2 + \theta_3) +$
$2m_3l_2l_3(\dot\theta_1^2 + 2\dot\theta_2^2)\sin\theta_3 + 4m_3l_3g\cos(\theta_2 + \theta_3) / 8$

b) Using numerical values:

Let $m_1 = 25kg$, $m_2 = 5kg$, $m_3 = 10kg$, $l_1 = 0.3m$, $l_2 = 0.5m$, $l_3 = 0.4m$, $I_{1x} = I_{1z} = \frac{m_1l_1^2}{12}$, $I_{1y} = I_{2x} = I_{3x} = 0$, $I_{2y} = I_{2z} = \frac{m_2l_2^2}{12}$, $I_{3y} = I_{3z} = \frac{m_3l_3^2}{12}$, run the programme again, then the torques in the form of $M(q)$ and $Q(q,\dot q)$ are:

$M(1,1) = (4000\cos(2\theta_2 + \theta_3) + 1066\cos2(\theta_2 + \theta_3) + 5833\cos2\theta_2 + 4000\cos\theta_3 + 6899) / 4000$

$M(1,2) = 0$

$M(1,3) = 0$

$M(2,1) = 0$

$M(2,2) = (4000\cos\theta_3 + 6899) / 2000$

$M(2,3) = (1000\cos\theta_3 + 533) / 1000$

$M(3,1) = 0$

$M(3,2) = (1000\cos\theta_3 + 533) / 1000$

$M(3,3) = 533 / 1000$

$Q(1,1) = -((4000\dot{\theta}_2 + 2000\dot{\theta}3)\dot{\theta}_1 \sin(2\theta_2 + \theta_3) + 1066(\dot{\theta}_2 + \dot{\theta}_3)\dot{\theta}_1\sin2(\theta_2 + \theta_3) + 5833\dot{\theta}_1\dot{\theta}_2\sin2\theta_2 + 2000\dot{\theta}_1\dot{\theta}_3\sin\theta_3) / 2000$

$Q(2,1) = ((4000\sin(2\theta_2 + \theta_3) + 1066\sin2(\theta_2 + \theta_3) + 5833\sin2\theta_2)\dot{\theta}_1^2 - 8000\dot{\theta}_2\dot{\theta}_3\sin\theta_3 - 4000\dot{\theta}_3^2\sin\theta_3 + 80000g\cos(\theta_2 + \theta_3) + 25000g\cos\theta_2) / 4000$

$Q(3,1) = ((1000\sin(2\theta_2 + \theta_3) + 533\sin2(\theta_2 + \theta_3) + 1000\sin\theta_3)\dot{\theta}_1^2 + 2000\dot{\theta}_2^2\sin\theta_3 + 4000g\cos(\theta_2 + \theta_3) / 2000$

Note that the resulting equations are much simplified compared with the symbolic form, but this is at the expense of not being able to change masses in the simulation itself.

3.3.2. Simulation process

Using the summarised block described in figure 3.2 to simulate the three-link manipulator, we have the structure block for the simulation of equation (3-1):

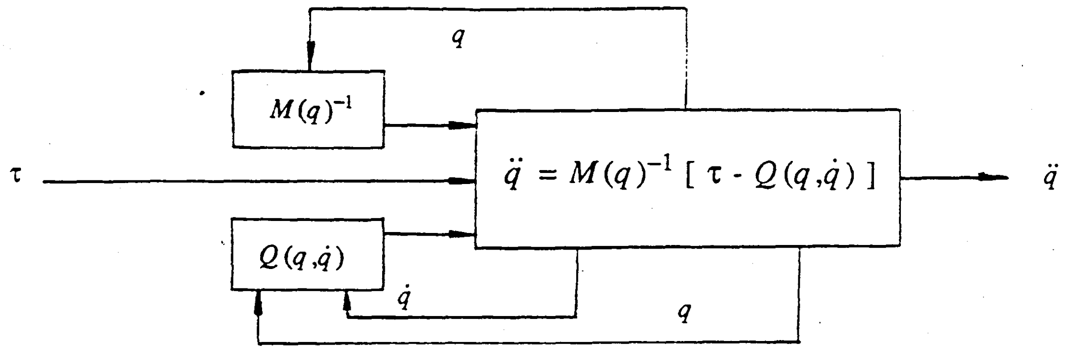


Fig. 3.4 Structure block of simulation process

Three programmes are used for calculating the three-link manipulator dynamic equations in the form of equation (3-1). Under given initial conditions, the initial values of $M(q)$ and $Q(q,\dot{q})$ are obtained in the second programme (CONTINUOUS SYSTEM CalculateMQ). Under input torque τ, which is a vector of $(\tau_1,\tau_2,\tau_3)^T$, the output, $q, \dot{q}, \ddot{q}$, are calculated in the first programme (CONTINUOUS SYSTEM ThreeLink) and then transferred to the second programme by the third connecting programme (CONNECTING SYSTEM ConTLandMQ); and versus the output $M(q)$ and $Q(q,\dot{q})$ are transferred back to the first one.

3.3.3. Figures

Now we run the simulation programmes to check the correctness of the dynamic equations obtained by the programme in REDUCE.

To simplify the problem and to show it easily, we suppose there is no any friction in our system.

First, let the torques which act on the links equal to zero, that is $\tau_1 = 0$, $\tau_2 = 0$, $\tau_3 = 0$. In this case, the manipulator works under the effect of gravity force. Therefore, the angular position of the first link which is called WAIST should keep its original position; the second link (SHOULDER) and the third link (ELBOW) must move up and down in a certain way.

Let $\tau_1 = 0$, $\tau_2 = 0$, $\tau_3 = 0$, $m_1 = 5kg$, $m_2 = 1kg$, $m_3 = 0.5kg$, and assume the mass distribution is the same along the axis of the link, the results are shown in figure 3.5. Note that the angular position of the shoulder changes between 0 and $-\pi$, and the movement of the elbow is a little complicated, but the value keeps around 0.

Figure 3.5 shows us the case of under zero initial position conditions, that is the shoulder and the elbow are at the horizontal position. If we keep them nearly along the vertical line position, for example the initial value of $\theta_{2_0} = -1.55$, other conditions are the same as that in the case of figure 3.5, the amplitude of the elbow's movement should be much smaller.

Figure 3.6 shows that the value of the elbow's position is much smaller compared with that in figure 3.5, but the movement of the shoulder is not exactly in the shape of the cosine function because of the existence of elbow.

If we let $m_3 \to 0$, the effect of the elbow to the shoulder can be ignored. This means that, in this case, the movement of the shoulder should be in the shape of the cosine function. The results are shown in figure 3.7. Note: we can not let $m_3 = 0$. If $m_3 = 0$, there is no inverse form of $M(q)$.

Now let us see the case of the waist having an initial angular velocity, for example, $\dot{\theta}_{1_0} = 0.1$ 1/s, the other initial conditions are the same as that in figure 3.5. From figure 3.8, we find that when the tangents of θ_2 and θ_3 are zero, the tangent of θ_1 equals to 0.1. This means that the initial value of $\dot{\theta}_1$ keeps to effect on the system during the whole time. This can be obviously shown by setting the gravity vector G = 0. The results are shown in figure 3.9. The curve of θ_1 is a line. Its tangent equals to $\dot{\theta}_1 = 0.1$.

Secondly, let $\tau_1 = 0.1$Nm, $\tau_2 = \tau_3 = 0$, the initial condition $\theta_{2_0} = -1$, and the other initial conditions are the same as that in figure 3.5. In this case, the shoulder and the elbow just like two flying sticks under no weight condition. When time $t \to \infty$, they should keep in the horizon position. The results are shown in figure 3.10.

Thirdly, let $\tau_1 = 0.1$Nm, $\tau_2 = 5.88$Nm, $\tau_3 = 0.98$Nm, G = 9.8kgm/s^2, and the initial conditions are the same as that in figure 3.5. In this case, τ_2 and τ_3 balance the effect of the gravity. The results should be the same as that of the case with G = 0. See figure 3.11.

Now let τ_3 increase, for example $\tau_3 = 1$Nm, the other conditions are the same as the one in figure 3.11. The results should be much complicated. It is shown in figure 3.12.

Change the input torques, such as $\tau_1 = 0.1$Nm, $\tau_2 = 2$Nm, $\tau_3 = 0.1$Nm, then the results are shown in figure 3.13.

3.4. Conclusion

The simulation method is efficient for rigid manipulator simulation. All figures obtained from running SIMNON programmes show that the simulation of the three-link manipulator is desirable.

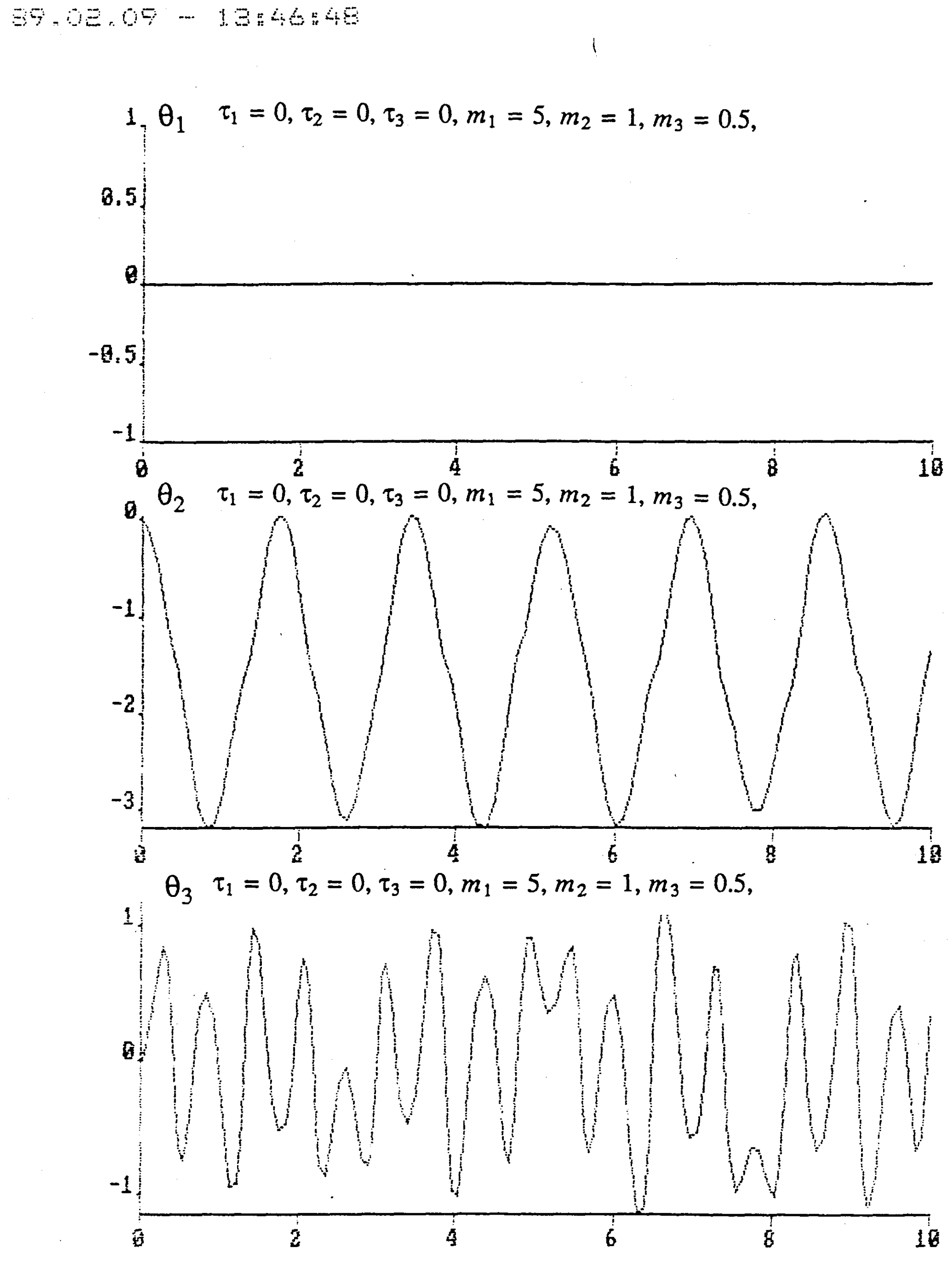


Fig. 3.5 The angular positions, $G = 9.8$

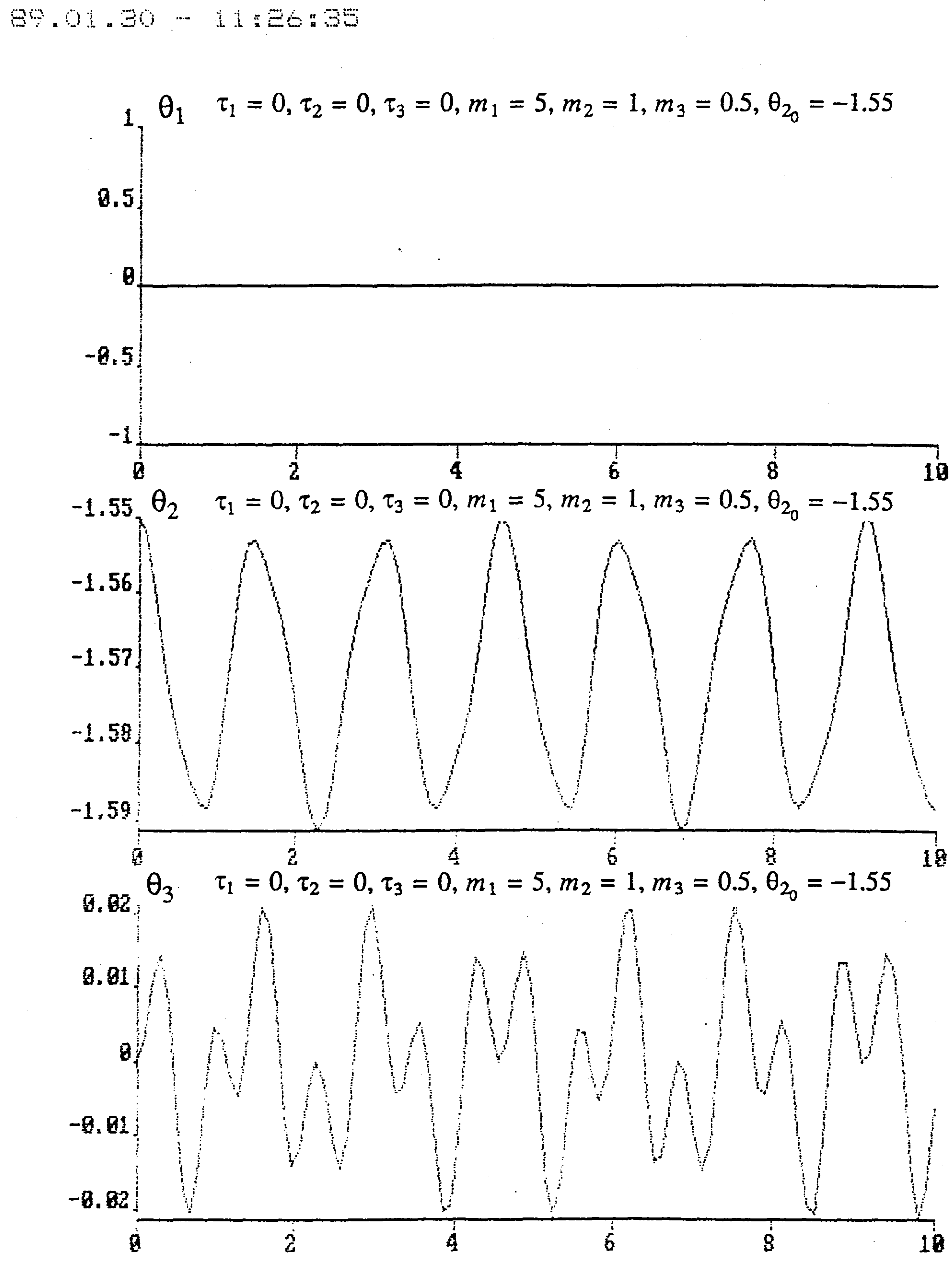


Fig. 3.6 The angular positions, $G = 9.8$, $th2_0 = -1.55$

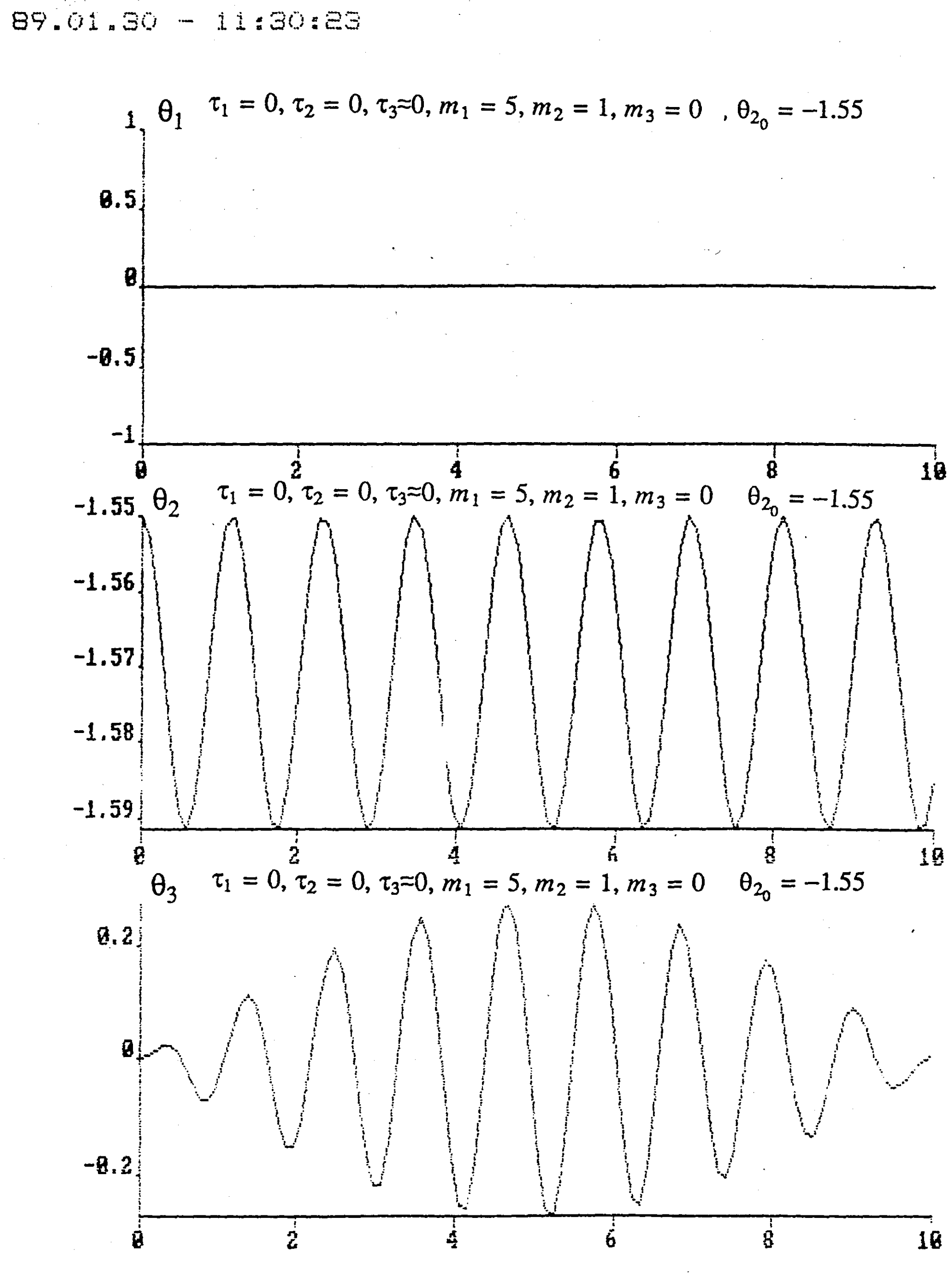


Fig. 3.7 The angular positions, $G = 9.8$, $th2_0 = -1.55$, $m3 \approx 0$

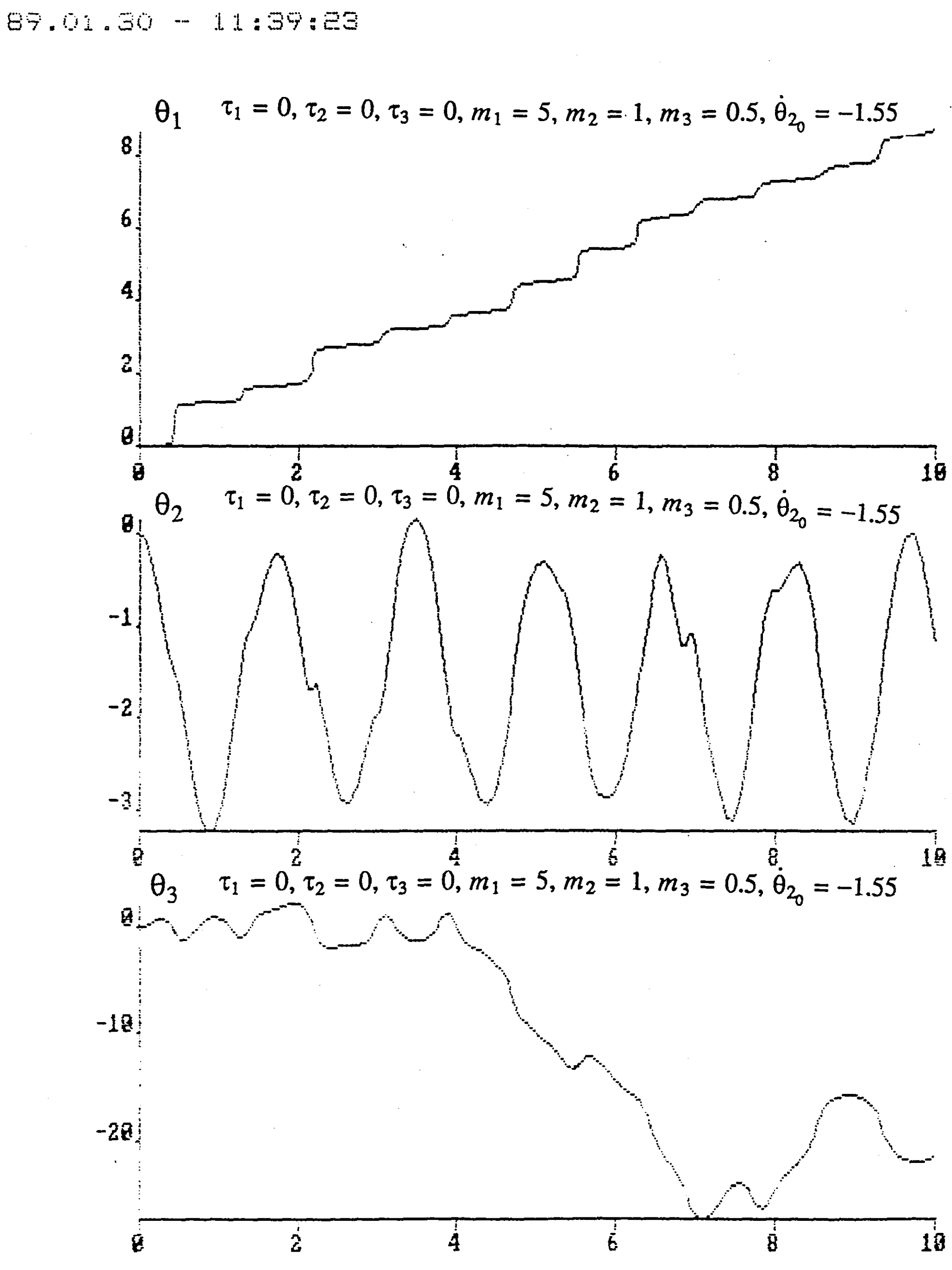


Fig. 3.8 The angular positions, $G = 9.8$, $thd1_0 = 0.11/s$

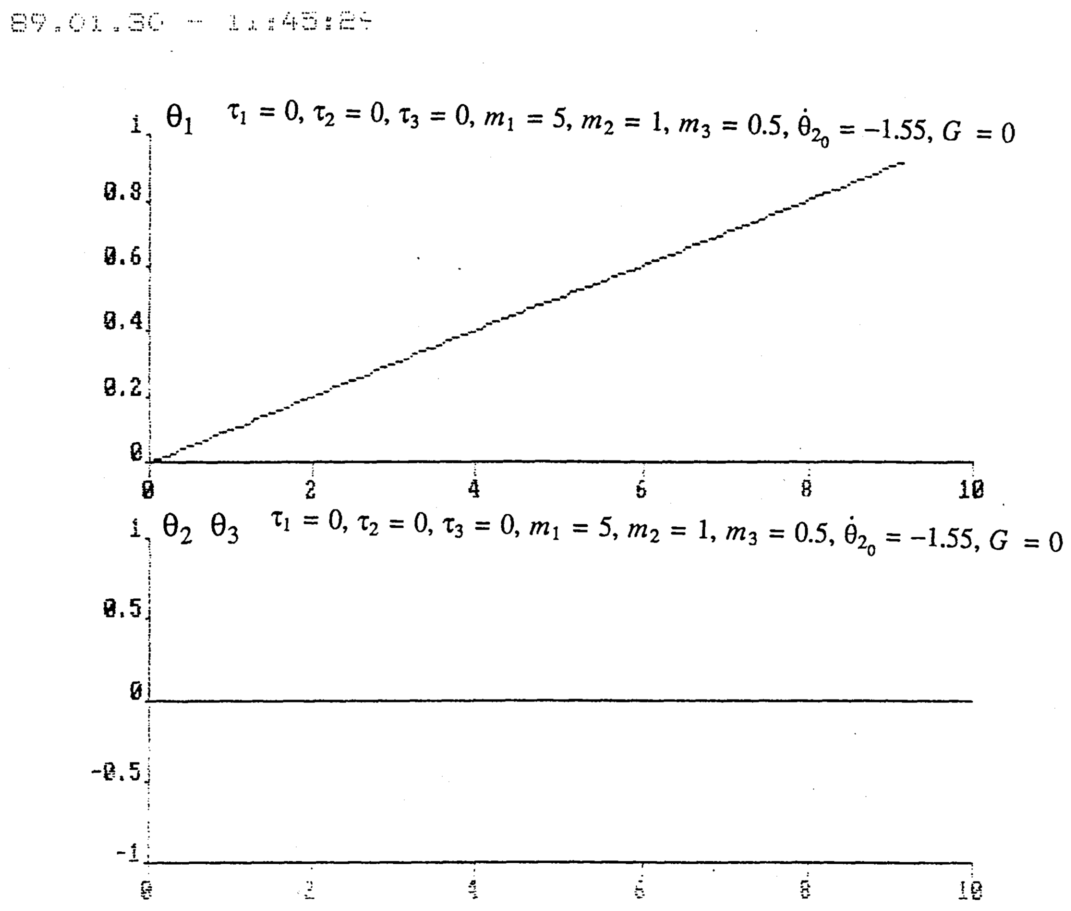


Fig. 3.9 The angular positions, $G = 0$, $thd1_0 = 0.11/s$

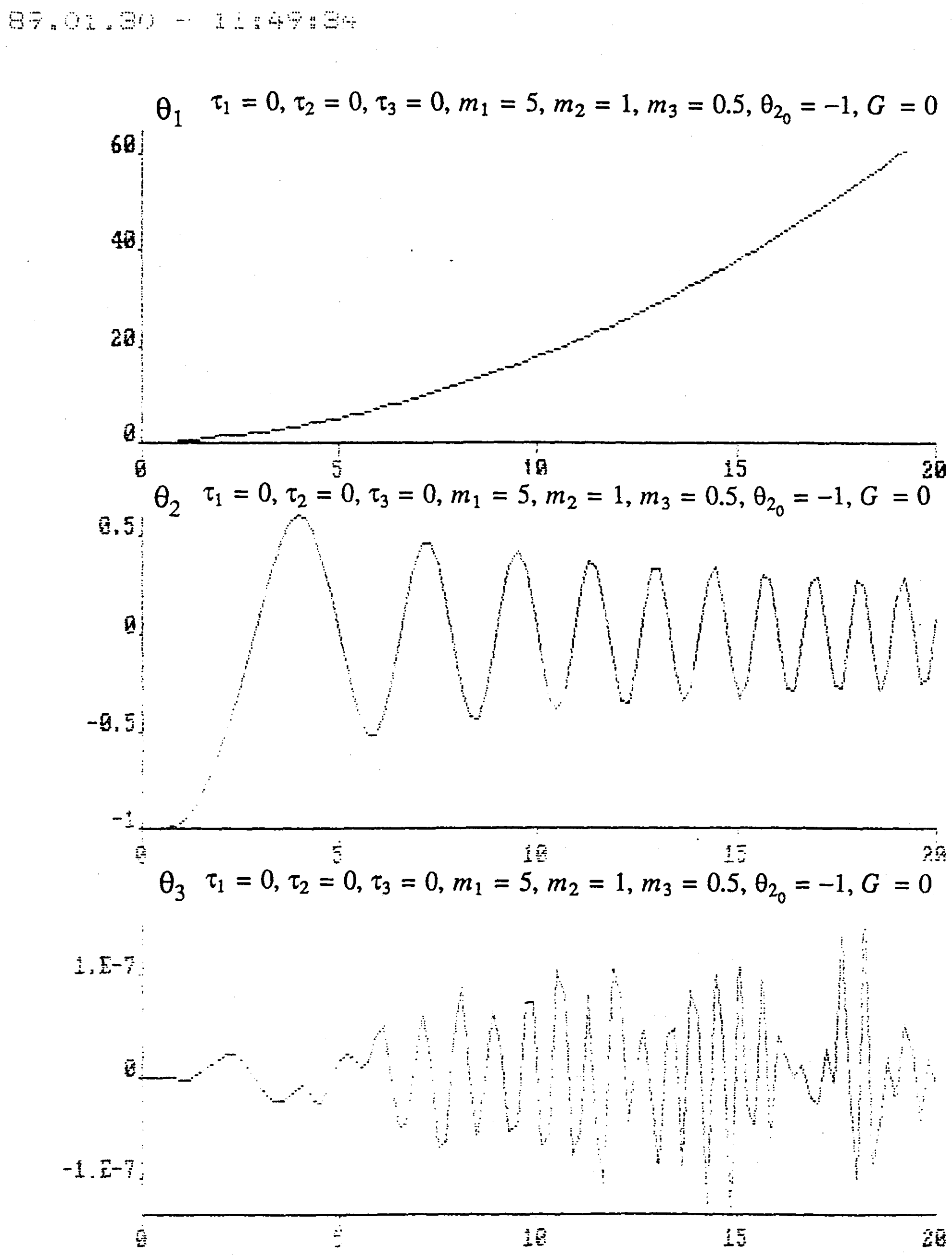


Fig. 3.10 The angular positions, $G = 0, \tau_1 = 0.1Nm, thd2_0 = -1$

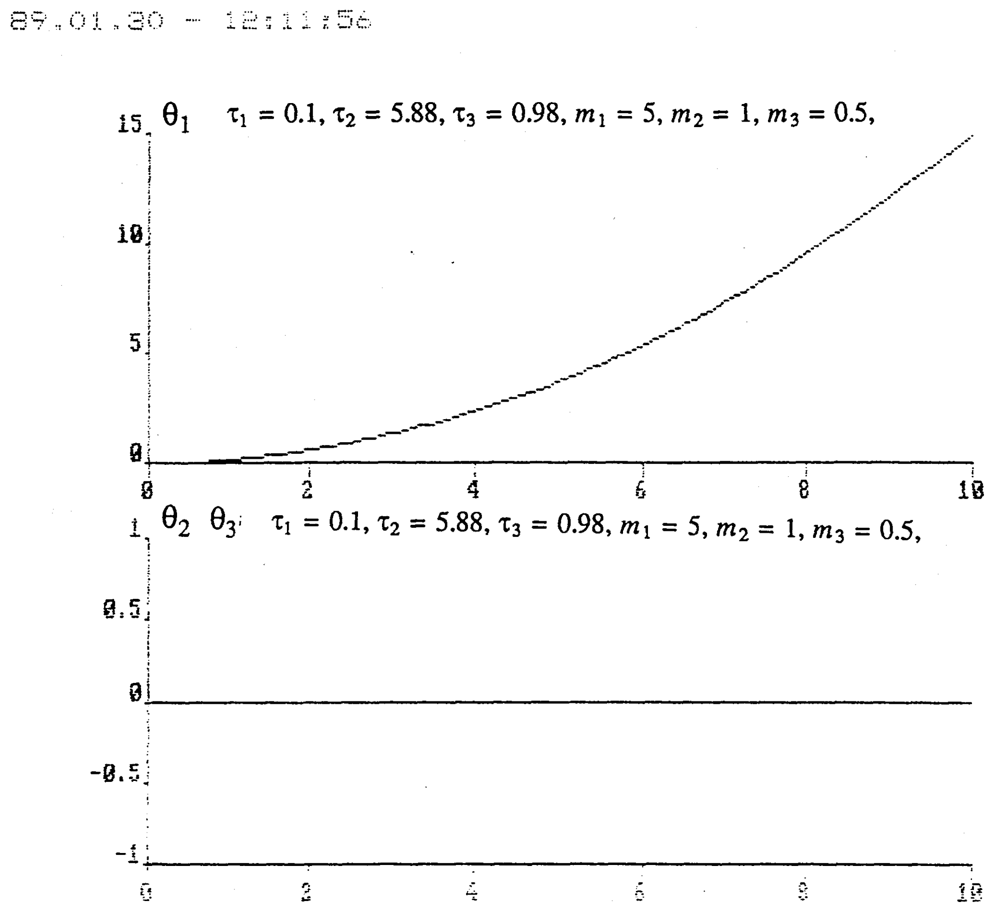


Fig. 3.11 The angular positions, $G = 9.8$, $\tau_1 = 0.1Nm$, $\tau_2 = 5.88Nm$, $\tau_3 = 0.98Nm$

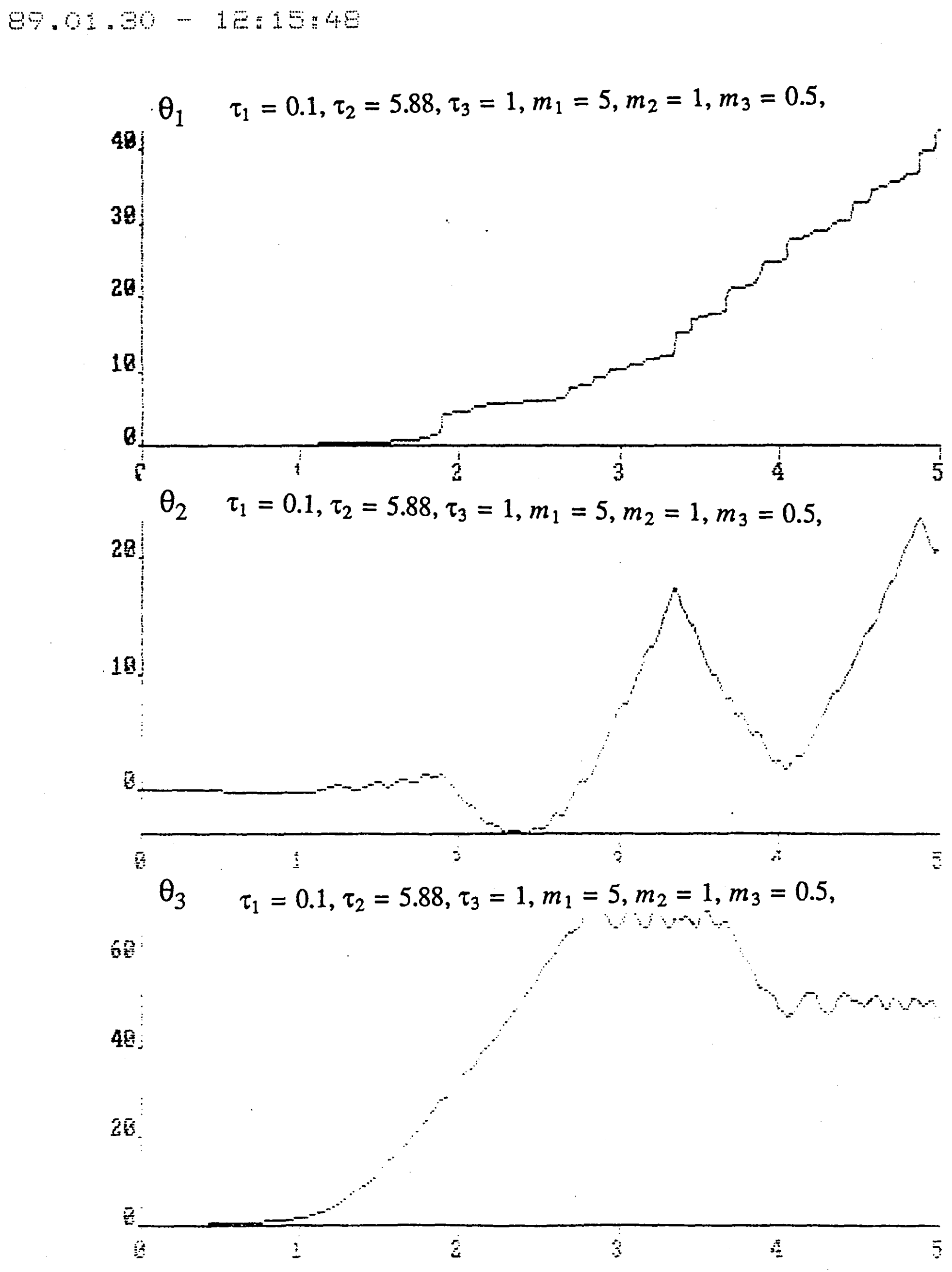


Fig. 3.12 The angular positions, $G = 9.8, \tau_1 = 0.1Nm, \tau_2 = 5.88Nm, \tau_3 = 1Nm$

89.02.09 — 13:54:43

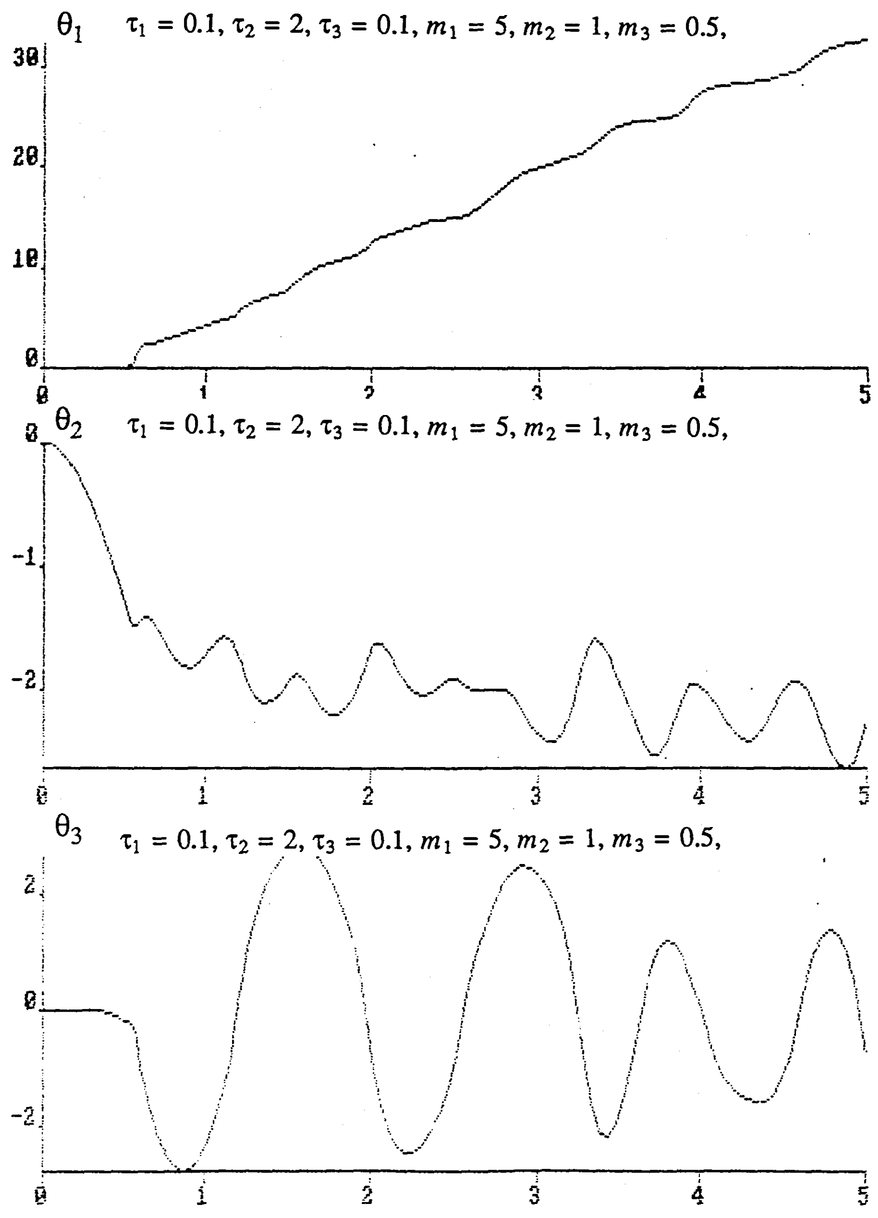


Fig. 3.13 The angular positions, $G = 9.8, \tau_1 = 0.1Nm, \tau_2 = 2Nm, \tau_3 = 0.1Nm$

Chapter 4. Analytic Model for One-Link Flexible Manipulators

4. Analytic Model for One-Link Flexible Manipulators

4.1. Introduction

A model is a description of some system intended to predict what happens if certain actions are taken. Virtually any useful model simplifies and idealises. Often the boundaries of the system and of the model are rather arbitrarily defined. Most forces that impinge on the system must be neglected on a priori grounds to keep the model tractable, even when there is no rigorous proof that such neglect is justified. Inevitably, the model is better defined than the real system. For a model to be useful, it is essential that, given a reasonably limited set of descriptors, all its relevant behaviour and properties can be determined in a practical way: analytically, numerically, or by driving the model with certain (typically random) inputs and observing the corresponding outputs.

An analytical model gives us a mathematical formula into which we substitute the characteristics of the system in question. It can then be quickly evaluated to give a performance number for the system. The formula is obtained by some sort of analysis: probability theory, queuing theory, or differential equation theory, for example. The mathematical sophistication required to derive the formula is usually substantially higher than that needed to develop a simulation model; however once derived, a formula is much easier to use.

In this situation, the simulation model may be more credible: perhaps its behaviour has been compared to that of the real system or perhaps it requires fewer simplifying assumptions and hence inevitably captures more of a hypothetical real system. However, the analytic model may give more insight into which policies are likely to be good.

Whether the model and the programme implementing it accurately represent the real system can be checked in two stages [17]:

Verification. Checking that the simulation programme operates in the way that the model implementation thinks it does; that is, is the programme free of bugs and consistent with the model? Such checks are rarely exhaustive.

Validation. Checking that the simulation model, correctly implemented, is a sufficiently close approximation to reality for the intended application. Due to approximations made in the model, we know in advance that the model and the real system do not have identical output distributions; thus statistical tests and theoretical analysis of model validity have to be use.

The validation problem arises because various approximations to reality are made in creating the model. We always restrict the boundary of the model, ignoring everything outside that is not an explicit input, and neglect factors believed to be unimportant.

Industrial robots are required to have light structures, because of the needs of high-speed performance and low energy consumption. Flexible manipulator systems exhibit many advantages over their traditional (rigid-arm) counterparts: they require less material, have less (arm) weight, consume less power, are more maneuverable, require smaller actuators, and are more transportable. However, they have not been much favoured in production industries due in part to the fact that manipulators are required to have a reasonable accuracy in the response of the arm's end-point to the joint control system input commands and this is severely deteriorated by structural deformation, especially in the case of flexible links where the deformation is oscillatory. Traditionally, these vibrations have been eliminated by increasing the rigidity of the arms, but this solution is not available in the case of flexible manipulators; therefore, it is important to realise (or validate) the characteristics of flexible links from theoretical point of view.

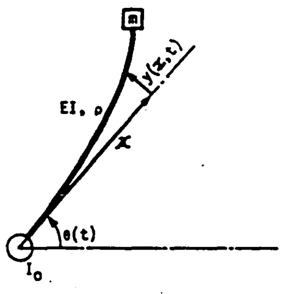
The problem of modelling flexible mechanical systems has been studied only partially. In the papers of Balas [18] and Karkkainen and Halme [19] a model approach to the problem of approximating a general flexible mechanical system is used. Book, Maizza-Neto and Whitney [7] directly approximate a two link flexible robot with a linear model derived from a nonlinear distributed parameter model. Book [20] also uses a special technique called lumping approximation to analyse flexible mechanical system assuming that the links bend is in a first mode vibration; Judd and Falkenburg [21] apply this method to nonrigid articulated robots; the same technique is adopted by Sunada and Dubowsky [22], and modified in such a way that more vibration modes are allowed. Chassiakos and Bekey [23] approximate the distributed parameter system response. This chapter provides a method of using transfer functions to model one-link horizontal planar flexible manipulators.

In this chapter, we set up a model by using transfer functions which are the responses of the two end points of a flexible distributed parameter system versus the input torque. These transfer functions which are purely based on vibration theories described by many people [24] [25] are symbolically calculated by REDUCE. Four different methods are presented to obtain the model of one-link flexible manipulators. The poles and zeros of the open-loop transfer functions, the frequency response and impulse response have been obtained by MATLAB. Also, the frequency response of a hinged-free beam has been tested by experiment. The correspondence between the four different methods and the experiment shows that the established model is reasonable.

4.2. Exact model

Consider the system of Fig. 4.1. Along the length of the arm ($0 \le x \le L$), the Young's modulus of elasticity (E), the transverse area moment of inertia (I), and the mass per unit length (ρ) are constant. Although the radius of the base axis of rotation is assumed for convenience to be zero, a motor armature and gear box are modeled by way of nonzero rigid mass moment of inertia, I_0, located at this base axis: The end mass, m, (located at the opposite end of the arm) is considered to occupy a point. The control torque, τ, is continuously variable.

The variable $y(x,t)$ is the deflection of the arm at a point located a distance x from the torqued end, measured relative to the undeformed position of the arm. The angular displacement, $\theta(t)$, is the angular position of the base measured from its original or reference position.



L =	MANIPULATOR ARM LENGTH
EI =	" " BENDING STIFFNESS
ρ =	" " MASS/LENGTH
I_0 =	RIGID BASE MASS MOMENT OF INERTIA
m =	MASS OF END LOAD
θ =	ANGULAR POSITION OF BASE
y =	DEFLECTION FROM EQUILIBRIUM
z =	POSITION LOCATION ALONG ARM
t =	TIME (MEASURED FROM BEGINNING OF MANEUVER)

Fig.4.1 Manipulator arm model

4.2.1. Transfer functions

Trying to find the transfer functions (G(s), G1(s)) of the tip's angle and the joint angle versus the input torque of a one flexible link, we can think it is the case of a cantilevered beam forced by adding inertial forces shown as below:

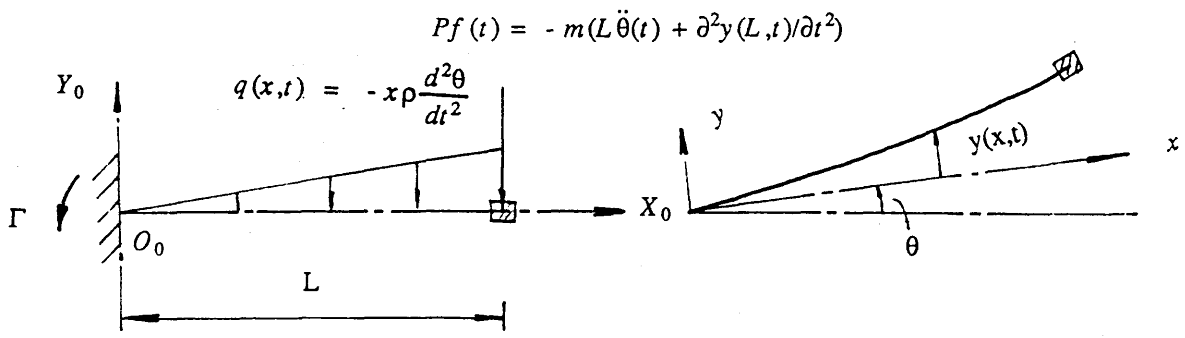

$Pf(t) = -m(L\ddot{\theta}(t) + \partial^2 y(L,t)/\partial t^2)$
$q(x,t) = -x\rho\frac{d^2\theta}{dt^2}$
Y_0
Γ
O_0
L
X_0
y
y(x,t)
x
θ

Fig.4.2. Inertial forces of the manipulator arm model

The method for solving the forced vibration of finite beams has been developed in Chen's book [24]. The inhomogeneous differential equation is

$$EI\frac{\partial^4 y}{\partial x^4} + \rho\frac{\partial^2 y}{\partial t^2} = q(x,t). \tag{4-1}$$

that is

$$EI\frac{\partial^4 y}{\partial x^4} + \rho\left(\frac{\partial^2 y}{\partial t^2} + x\frac{d^2\theta}{dt^2}\right) = 0. \tag{4-2}$$

For the torque, we have

$$I_t\frac{d^2\theta}{dt^2} + mL\frac{d^2 y(L,t)}{dt^2} + \rho\int_0^L x\frac{\partial^2 y}{\partial t^2}dx = \tau. \tag{4-3}$$

where

$$I_t = I_0 + \rho\frac{L^3}{3} + mL^2$$

The geometric boundary conditions at the torqued end are:

$$y(0,t) = 0, \tag{4-4}$$

$$\frac{\partial y}{\partial x}\Big|_{x=0} = 0. \tag{4-5}$$

The natural boundary conditions at the free end are:

$$\frac{\partial^2 y}{\partial x^2}\Big|_{x=L} = 0, \tag{4-6}$$

$$EI\frac{\partial^3 y}{\partial x^3} - m\left(L\frac{d^2\theta}{dt^2} + \frac{\partial^2 y}{\partial t^2}\right)\Big|_{x=L} = 0. \tag{4-7}$$

Using the Laplace transform method to solve the wave equation, it is possible to obtain a solution in terms of standing or travelling waves. The type of solution obtained depends on the manner in which the inverse transformation is carried out.

The Laplace transforms of equation (4-2) and (4-3) in view of zero initial conditions are

$$EI\,\frac{d^4Y(x,s)}{dx^4} + \rho\, s^2\,(\,Y(x,s) + x\Theta(s)\,) = 0 \tag{4-8}$$

$$I_t\, s^2\Theta + mLs^2Y(L) + \rho\, s^2\int_0^L x\, Y\, dx = \Gamma \tag{4-9}$$

where Y(x), Θ, and Γ are the Laplace transforms of y, θ, and τ, respectively. In writing equation (4-8) it was assumed that

$$\mathcal{L}\;\frac{\partial^2 y(x,t)}{\partial x^2} = \int_0^\infty e^{-st}\frac{\partial^2 y(x,t)}{\partial x^2}dt = \frac{\partial^2}{\partial x^2}\int_0^\infty e^{-st}y(x,t)dt = \frac{d^2Y(x,s)}{dx^2}$$

which implies that the function $e^{-st}y(x,t)$ is such that interchange of the order of differentiation with respect to x and integration with respect to t is possible [26]. The function $Y(x,s)$ is subjected to the transformed boundary conditions

$$Y(0) = 0 \tag{4-10}$$

$$\frac{dY}{dx}\Big|_{x=0} = 0 \tag{4-11}$$

$$\frac{d^2Y}{dx^2}\Big|_{x=L} = 0 \tag{4-12}$$

$$ms^2Y(L) + mLs^2\Theta - EI\frac{d^3Y}{dx^3}\Big|_{x=L} = 0 \tag{4-13}$$

where Y(x), Θ, and Γ are the Laplace transforms of y, θ, and τ, respectively. A general solution to equation (4-8) is

$$Y(x) = exp(\beta x)\,[\,A\cos\beta x + B\sin\beta x\,] + exp(-\beta x)\,[\,C\cos\beta x + D\sin\beta x\,] - \Theta x \tag{4-14}$$

where

$$\beta^4 = \frac{\rho s^2}{4EI} \tag{4-15}$$

The constants A, B, C, and D are evaluated using equations (4-10) - (4-13). The resulting solution for Y is then substituted into the definite integral of equation (4-9), which is evaluated analytically. From this result and equation (4-14), the transfer functions G(s), G1(s) are found, where

$$G = \frac{Y/L + \Theta}{\Gamma} \tag{4-16}$$

$$G1 = \frac{\Theta}{\Gamma} \tag{4-17}$$

The above process can be done by using REDUCE. The REDUCE code and the expending equations of G and G1 are listed in appendix 3.

4.2.2. Open-loop responses

MATLAB has a rich collection of functions immediately useful to the control engineers. Complex arithmetic, root-finding, and FFT's are just a few examples of important numerical tools. Moreover, the most important is the tools are not found in the toolbox can be created by writing new M-files. The results in this report are all got by running written M-files.

Figures 4.3 - 4.6

In order to simplify the model, suppose the length of the link is $L = 1$ M, the mass per unit length is $\rho = 1$ Kg/M, the base moment of inertia is $I_0 = 0$, the end mass is $m = 0$. Figures 4.3 - 4.6 show the poles and zeros of G near the origin of axes. They are:

The Poles and Zeros of G

EI	Poles	Zeros
50	0.0000+-0.0000*i, 0.0000+-109.02*i, 0.0000+-355.30*i.	+-079.10+0.0000*i, +-427.46+0.0000*i.
100	0.0000+-0.0000*i, 0.0000+-154.18*i, 0.0000+-499.65*i.	+-111.87+0.0000*i, +-604.52+0.0000*i.
200	0.0000+-0.0000*i, 0.0000+-218.05*i, 0.0000+-706.61*i.	+-158.20+0.0000*i, +-854.90+0.0000*i.
300	0.0000+-0.0000*i, 0.0000+-267.05*i, 0.0000+-865.42*i.	+-193.80+0.0000*i, +-1047.1+0.0000*i.

According to Thomson's book [25], for any kind of beams, the natural frequencies of vibration are found by the equation:

$$\omega_n = n^2 \sqrt{\frac{EI}{\rho}} \tag{4-18}$$

where the number n depends on the boundary conditions of the problem. Here the beam configuration is the hinged-free kind. So

$$(n_i L)^2 = 0, \; 15.4, \; 50.0, \; ..$$

Back to the equation (4.18)

$$EI = 50: \quad \omega_1 = 0, \; \omega_2 = 108.89, \; \omega_3 = 353.55.$$

$$EI = 100: \quad \omega_1 = 0, \; \omega_2 = 154.00, \; \omega_3 = 500.00.$$

$$EI = 200: \quad \omega_1 = 0, \; \omega_2 = 217.79, \; \omega_3 = 707.11.$$

$$EI = 300: \quad \omega_1 = 0, \; \omega_2 = 266.74, \; \omega_3 = 866.03.$$

The values of the natural frequencies are the same of the poles value of G.

Figures 4.7 - 4.10

Figures 4.7 - 4.10 show the poles of the transfer function (G1) which is the joint angle versus the input torque. The poles are the ones near the origin of the axes. Because the ones far from the origin have small effects on the system, they can be ignored. G1 has no zeros. The poles' values are the same as the ones of G shown in the table above.

Figures 4.11 - 4.12

Figures 4.11 - 4.12 are the frequency responses of G when different values of EI are chosen. The peaks occur at the exact points of the natural frequencies.

Figures 4.13 - 4.14

Figures 4.13 - 4.14 show the impulse responses of G. The larger stiffness causes larger vibrational frequencies.

The results of figures 4.11 - 4.14 correspond with the results in Gawthrop's report [27].

4.3. Approximated model derived in time domain

In equation (4-1), the assumed force $q(x,t)$ varies with time in the same way for all points on the beam, that is

$$q(x,t) = p(x)\, f(t) \tag{4-19}$$

where

$$p(x) = -x\rho, \quad f(t) = \ddot{\theta}(t)$$

equation (4-1) becomes

$$EI\frac{\partial^4 y}{\partial x^4} + \rho\frac{\partial^2 y}{\partial t^2} = p(x)\, f(t) \tag{4-20}$$

The general solution to equation (4-20) will be the form

$$y = \sum_r \Phi_r(x)\, q_r(t) \tag{4-21}$$

where $\Phi(x)$ is a normal mode which has the form:

$$\Phi_r = B_1\cosh\lambda x + B_2\sinh\lambda x + B_3\cos\lambda x + B_4\sin\lambda x \tag{4-22}$$

$$\int_0^L \rho\, [\Phi_r(x)]^2\, dx = M \tag{4-23}$$

where $\lambda^4 = \rho\frac{\omega^2}{EI}$, M is the mass of the beam, $q_r(t)$, a principal coordinate, is a function of time.

After some tedious manipulations [28], we obtain

$$\ddot{q}_r + \omega_r^2 q_r = \frac{f(t)\int_0^L p(x) PHI_r(x)\, dx}{M} \tag{4-24}$$

For a known distribution of applied force $p(x)$ and known mode shapes the integrals in equation (4-24) can be evaluated; if the ratio is K_r, then

$$\ddot{q}_r + \omega_r^2 q_r = K_r f(t) \tag{4-25}$$

4.3.1. Transfer functions

For the system of figure 4.1, there are two kinds of virtual forces acted on the cantilevered flexible beam: one is a distributed force $q(x,t) = -\rho x\ddot{\theta}(t)$, the other is a concentrated force acted at the free end $Pf(t) = -m(L\ddot{\theta}(t) + \partial^2 y(L,t)/\partial t^2)$ as shown in figure 4.2.

A concentrated force can be thought as $Pf(t) = p(x)\Delta x f(t)$ as $\Delta x \to 0$. Equation (4.24) becomes

$$\ddot{q}_r + \omega_r^2 q_r = \frac{P\Phi_r(L)f(t)}{M} \tag{4-26}$$

So, the differential equation of our system's coordinate $q_r(t)$ is

$$\ddot{q}_r + \omega_r^2 q_r = \frac{\int_0^L -\rho x\ddot{\theta}(t)\Phi_r(x)\,dx}{M} + \frac{P\Phi_r(L)f(t)}{M}$$

$$= K_{qr}\ddot{\theta} + K_{pr}\,(L\ddot{\theta} + \ddot{y}\,|_{x=L}) \tag{4-27}$$

where

$$K_{qr} = \frac{\int_0^L -\rho x\Phi_r\,dx}{M}, \quad K_{pr} = \frac{-m\Phi_r(L)}{M}$$

The normal mode of cantilevered beam has the boundary conditions:

$$\Phi(0,t) = 0, \quad \frac{\partial\Phi}{\partial x}\,|_{x=0} = 0 \tag{4-28}$$

$$\frac{\partial^2\Phi}{\partial x^2}\,|_{x=L} = 0, \quad \frac{\partial^3\Phi}{\partial x^3}\,|_{x=L} = 0 \tag{4-29}$$

With this four boundary conditions, the frequency equation is

$$\cos\lambda L\,\cosh\lambda L + 1 = 0 \tag{4-30}$$

With equation (4-23), the constants B_1, B_2, B_3, B_4 can be obtained. Take these constants back to equation (4-27), K_{qr} and K_{pr} can be calculated.

Take Laplace transform of equations (4-3), (4-21), and (4-27), the transfer functions G(s), G1(s) which are defined by equations (4-16) and (4-17) respectively can be obtained. This process also has been done by using REDUCE. The REDUCE code and the expanding equations of G and G1 are listed in appendix 3.

4.3.2. Simulation results

We take the same system which has been discussed in section 4.2. The transfer function G of the uniformed beam's manipulator is listed in appendix 3. Having obtained the transfer function, it is easy to see its frequency response, and to compare the results with that of the exact model.

Figures 4.15 - 4.16 are the frequency responses of G derived in time domain with different stiffness of EI=50, 100, 200, 300 as the same cases in section 4.2. Figures 4.17 - 4.18 show the comparison of the frequency responses of exact model's transfer function G and of the transfer function derived in time domain approximated by eight orders. The first seven peaks of the two different model's frequency responses happen at the same frequency points. They have the same frequency responses except the part after the seventh peak.

4.4. Approximated model of using Lagrange-Euler formulation

For the same system of figure 4.1, we can also get its approximated model by using the Lagrange-Euler formulation [3].

The application of the Lagrange-Euler formulation in deriving flexible links' manipulator dynamic equations is based on the approximation method known as the Ritz-Kantorovitch method which is on account of a function series expansion and uses a so called ' complete set of functions '. A set of function { $f_k(x)$ with $f_k : [0,L]\rightarrow R$ and $f_k \in C^2([0,L])$} is said complete if $\forall\ y\in C^2([t_i,t_f]\times[0,L])$ and $\forall\ \varepsilon>0$ an index δ exists and, δ time functions $\beta_1(t)$, $\beta_2(t)$, ...,$\beta_\delta(t)$ exist such that

$$|y(t,x) - \sum_{k=0}^{\delta} \beta_k(t)\, f_k(x)| \le \varepsilon \tag{4-31}$$

Given a complete function set each function $y \in C^2([t_i,t_f])$ can be expanded in a function series

$$y(t,x) = \sum_{k=0}^{\infty} \beta_k(t)\, f_k(x) \tag{4-32}$$

In particular, if an approximate description is sufficient to the purpose of the analysis, the expansion can be truncated at a finite order term

$$y(t,x) = \sum_{k=0}^{n} \beta_k(t)\, f_k(x) \tag{4-33}$$

In this way our problem is reduced to the classical formulation of Lagrangian discrete mechanics and, the motion equation of the system can be given by the Lagrange-Euler formulation described in chapter 2.

For one-link flexible manipulator shown in figure 4.1, we apply the Ritz-Kantorovitch method. Given a set of functions { $f_{(x)}$ } y can be approximated by $y(t,x) = \sum_{k=0}^{n} \beta_k(t)\, f_k(x)$ with an error depending on the order n. We choose $f_k(x) = x^k$, consequently in frame (X_0, Y_0) of figure 4.2 the parameter equations of the curve are

$$X_0(x,t) = x\cos\theta - \sum_{k=0}^{n} \beta_k(t)\, x^k \sin\theta \tag{4-34}$$

$$Y_0(x,t) = x\sin\theta + \sum_{k=0}^{n} \beta_k(t)\, x^k \cos\theta \tag{4-35}$$

in which, in order to respect the geometrical constraints, $y(t,0) = 0$, $\dot{y}(t,0) = 0$, $\beta_0 = \beta_1 = 0$.

Introduce the new variables $q_1 = \theta$ and $q_i = \beta_i$, $(i \neq 1)$. From which, with some tedious manipulations, we obtain the following approximate model which has the form

$$D\ddot{q} + H\dot{q} + C = \tau \tag{4-36}$$

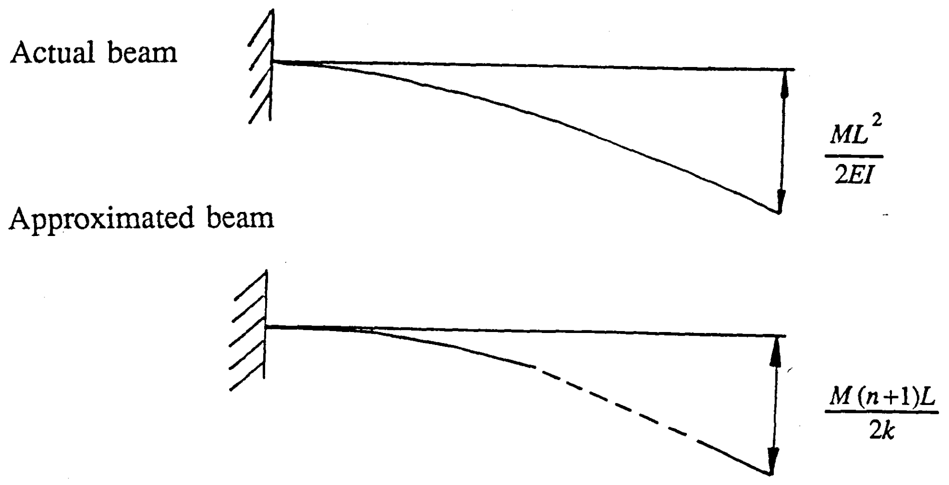
We have done this process by REDUCE. The REDUCE code and the expanding equations of D, H and C for the model approximated by six order are listed in appendix 3.

This model is described by a set of non-linear differential equations, we cannot directly give its transfer functions. So, we just can check the correctness by the simulation in time domain. Figure 4.19 shows the response of the differential equations of equation (4-36) using ACSL (Advanced Continuous Simulation Language) [28] under unit input torque. This result also has been compared by the step response of exact model's transfer function G shown in figure 4.20.

4.5. Approximated model of adding springs

In this section, we investigate another approximate model whereby a flexible link can be replaced by several rigid links connected by springs. Assume that the number of added springs is n. The distribution of the springs is that the first one is at the fixed end and the others divide the beam into equal parts. According to the elastic theory, they have the same stiffness k. There are a number of ways to choose the spring constant k. Two possible methods are given.

First, in static state, if the free end of the link is acted with a moment M, the displacements of the end of the two links are the same.



For the actual beam, the displacement of the end point of the beam is

$$\Delta y_1 = \frac{ML^2}{2EI}$$

where L is the length of the beam, E is the Young's modulus of elasticity, I is the transverse area moment of inertia.

For the approximated beam, the displacement of the end point of the beam is:

$$\Delta y_2 = \frac{ML}{k}(1/n + 2/n + \cdots + n/n) = \frac{ML(n+1)}{2k}$$

where k is the springs' stiffness.

That is

$$\frac{ML^2}{2EI} = \frac{M(\,n+1\,)L}{2k} \tag{4-37}$$

$$k = \frac{(\,n+1\,)EI}{L} \tag{4-38}$$

Second, another way of choosing the stiffness is that the angles of the end of the two links are the same. In this case, we can similarly get the stiffness k

$$k = \frac{n\ EI}{L} \tag{4-38a}$$

After some simulation (described in chapter 5), it is better to use equation (4-38) than equation (4-38a) to choose the springs' stiffness.

We have only considered the static state condition to derive the ways of choosing the springs' stiffness described by equation (4-38) and (4-38a) . The more rigid parts are used to approximate a flexible link, the better correspondence with the real response of the dynamic characteristics of the link can be obtained.

With these assumption, the differential equations can be obtained by using the programmes described in chapter 2. In order to use the approximated method to simulate the flexible link, We must use the free joint linked parts to replace the real link. If a manipulator has three real links, we use three rigid parts to simulate each link, we will equal to calculate a manipulator with twelve links. It must be a real problem to get the dynamic equations although the programme in REDUCE can calculate a manipulator with any links. Even though we have obtained the dynamic equation, to simulate it is a problem.

In real systems, the added joints' angle is small. The effect of the angles is in the form of cosine and sine function. So we can let the angles equal to zero to get the approximated dynamic equations. This approximation will cause the dynamic equation to be much simplified. Another advantage of with this assumption is that the dynamic equations of one-link flexible manipulators is a set of linear differential equations. This means that we can obtain the transfer functions of the system by introducing the equation

$$\theta_{end} = \theta + \theta_1 + \frac{n-1}{n}\theta_2 + \cdots + \frac{1}{n}\theta_n \tag{4-39}$$

The model has approximately been obtained by adding eight springs using REDUCE and MATLAB. The MATLAB programme is listed in appendix 3.

Figures 4.21 - 4.22 show the comparison between the frequency responses of exact model's transfer function G and the ones of the transfer function of the model approximated by adding seven springs. The first four peaks of the two different models' frequency responses happen at the same frequency points. They have the same frequency responses except the part after the fourth peak.

4.6. Experimental results

The four different methods to model one-link flexible manipulators are fully developed in frequency domain. Do they correspond with real flexible beams? We should do some experiments to verify the simulation results.

The experiment is based on the idea of measuring the end point linear acceleration of a vertical hinged-free beam excited by a force shown in figure 4.23.

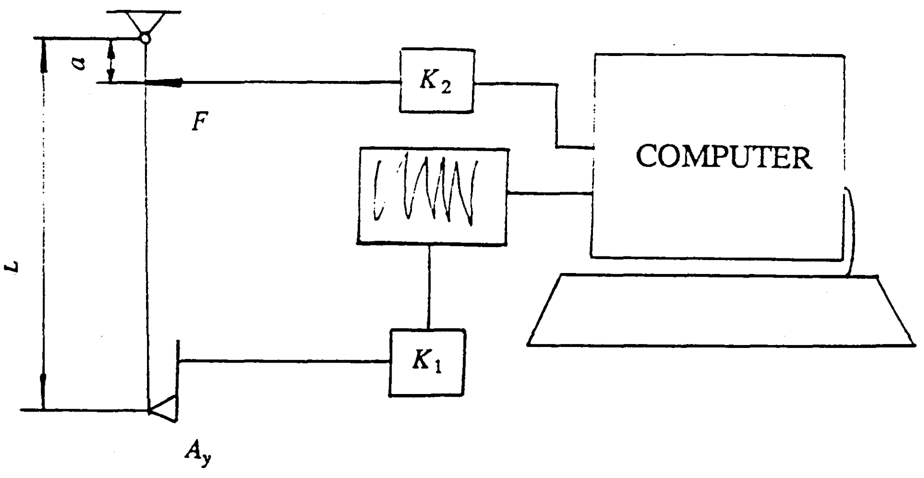


Fig.4.23. Experiment diagram

The force $F(t)$ is a sine function of time t, that is

$$F(t) = F_0 \sin(\omega t + \phi_0) \tag{4.40}$$

where F_0 is the force amplitude which can be preset manually, ω is the frequency of the exciting force $F(t)$, ϕ_0 is the initial phase of $F(t)$.

The measured end-point linear acceleration's amplitude A_y can be read by accelerometers. Consequently, the correlated frequency response of the end-point angle's acceleration, $|\ddot{\theta}_{end}| = \frac{A_y}{L}$, versus input torque $|\tau| = F_0 \times a$ can be calculated by the computer. The comparison between the frequency response of the experiment results and the one of the exact model is shown in figure 4.24. The beam's data are EI = 842.285, L = 0.727 M, a = 0.005 M, ρ = 2.45 Kg/M.

We must note that the excited force is not the same as the real system's input torque. But, when the distance *a* is much small, the force can approximately be thought as the real system's input torque. This has been verified by the simulation of the forced beam system described in figure 4.23. The simulation method is similar to the one described in section 4.3. The simulation results are shown in figures 4.25 - 4.26.

4.7. Conclusion

The work of one-link flexible manipulator validation is important and efficient. Any of the four different analytic methods can lead to obtain the analytic model of a one-link flexible manipulator. The exact model is efficient. The other three approximate methods are reasonable. The experiment result gives us much more confidence to the analytic models. One big advantage of the method of adding springs to approximate our model is that it is easy to be used for modelling multi-link flexible manipulators.

Fig4.3. The poles and zeros of G, EI=50

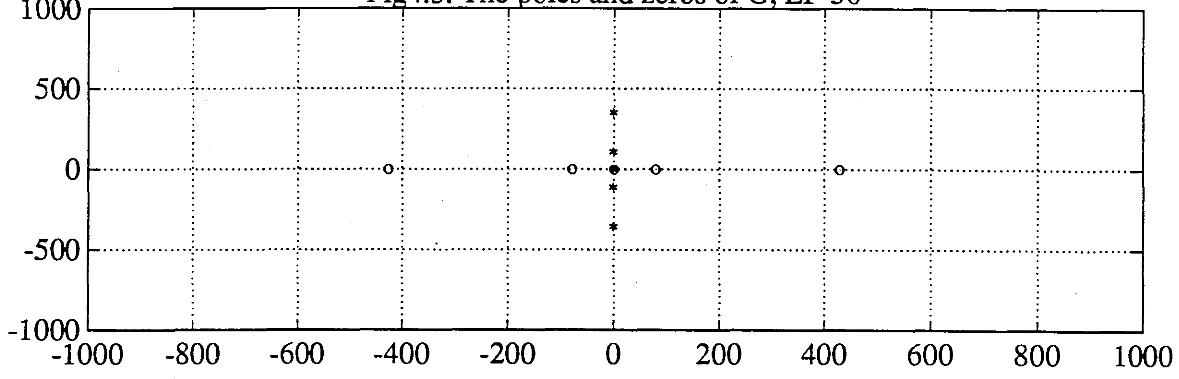


Fig4.4. The poles and zeros of G, EI=100

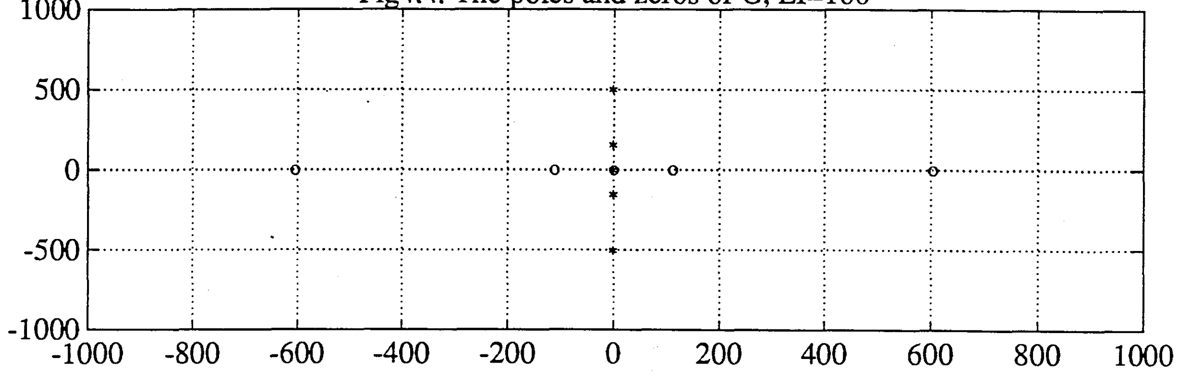


Fig4.5. The poles and zeros of G, EI=200

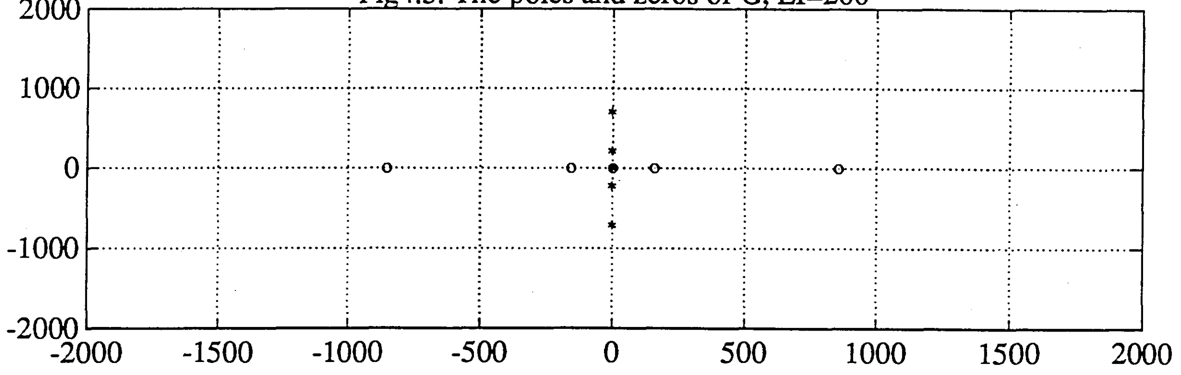


Fig4.6. The poles and zeros of G, EI=300

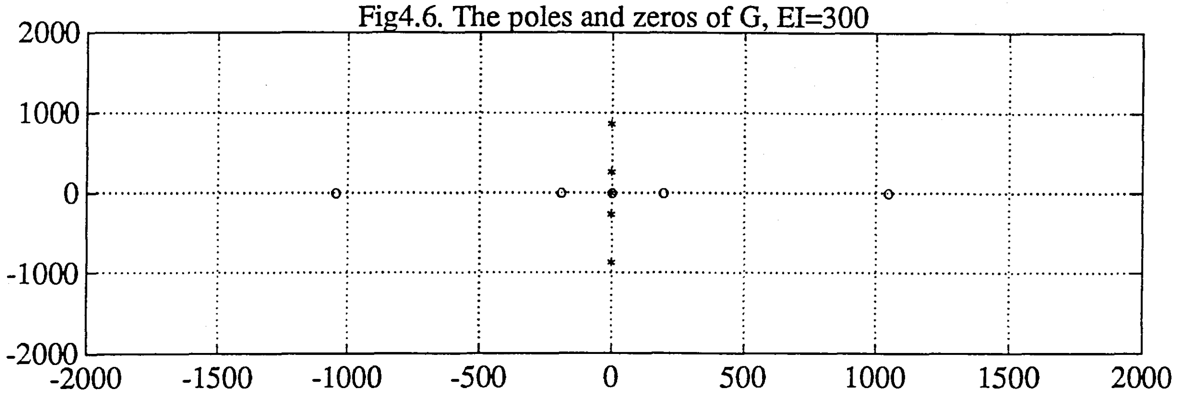


Fig4.7. The poles of G1, EI=50

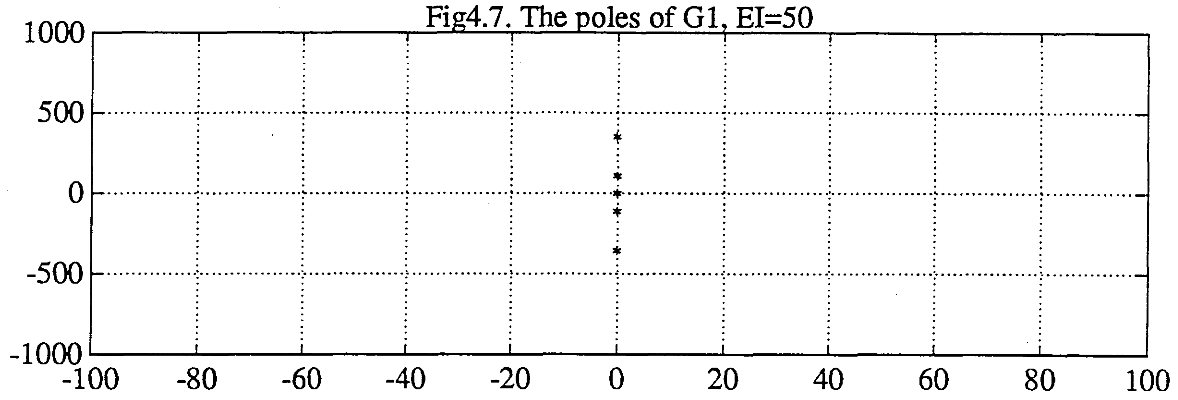


Fig4.8. The poles of G1, EI=100

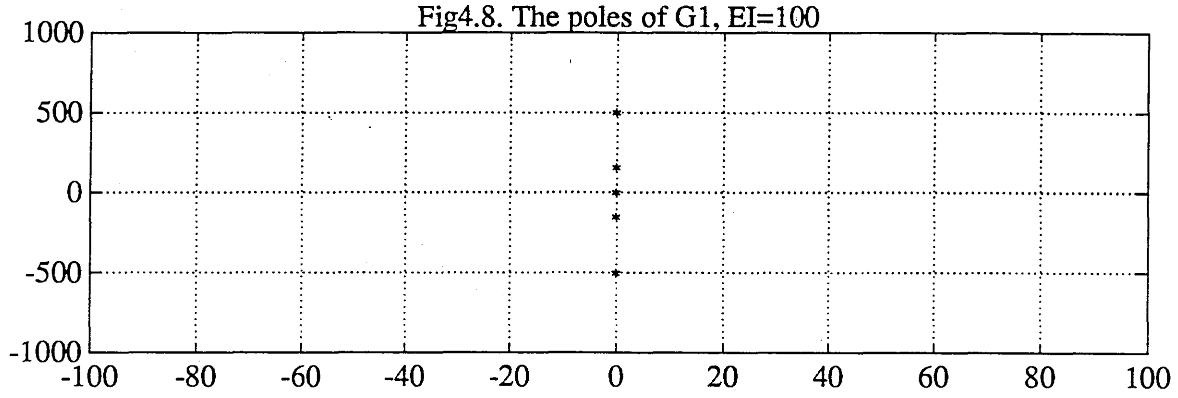


Fig4.9. The poles of G1, EI=200

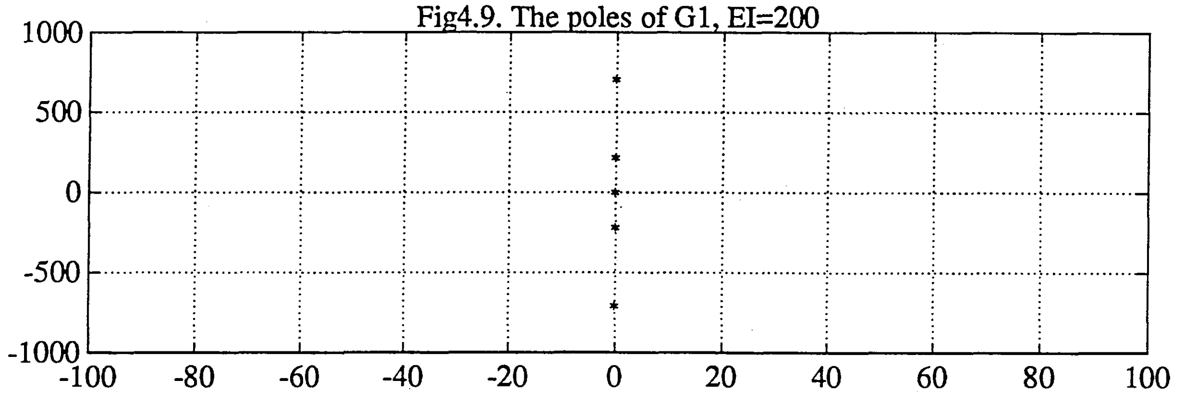


Fig4.10. The poles of G1, EI=300

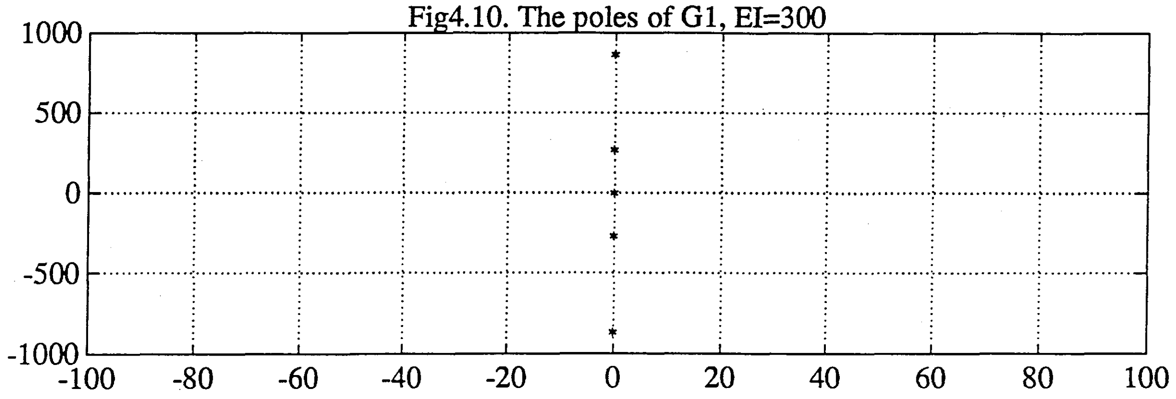


Fig.4.11. The frequency response of G, EI=50, 100

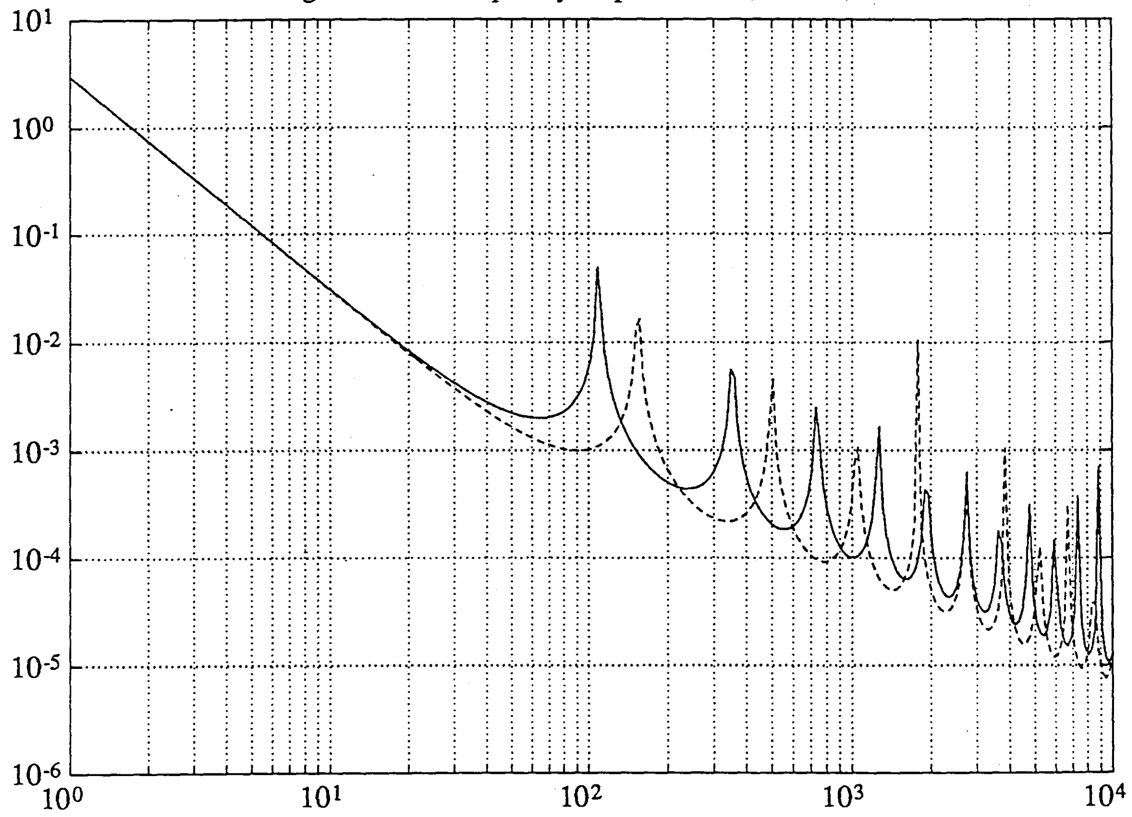


Fig.4.12. The frequency response of G, EI=200, 300

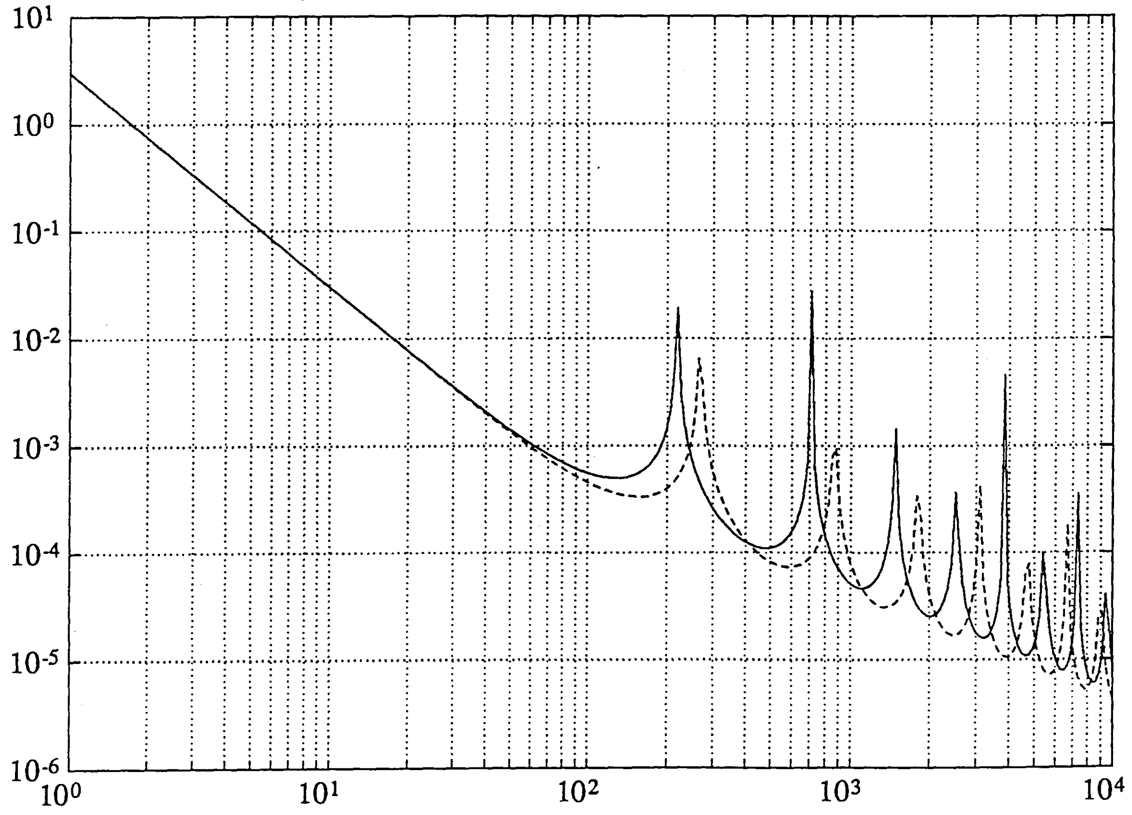


Fig.4.13. The impulse response of G, EI=50, 100

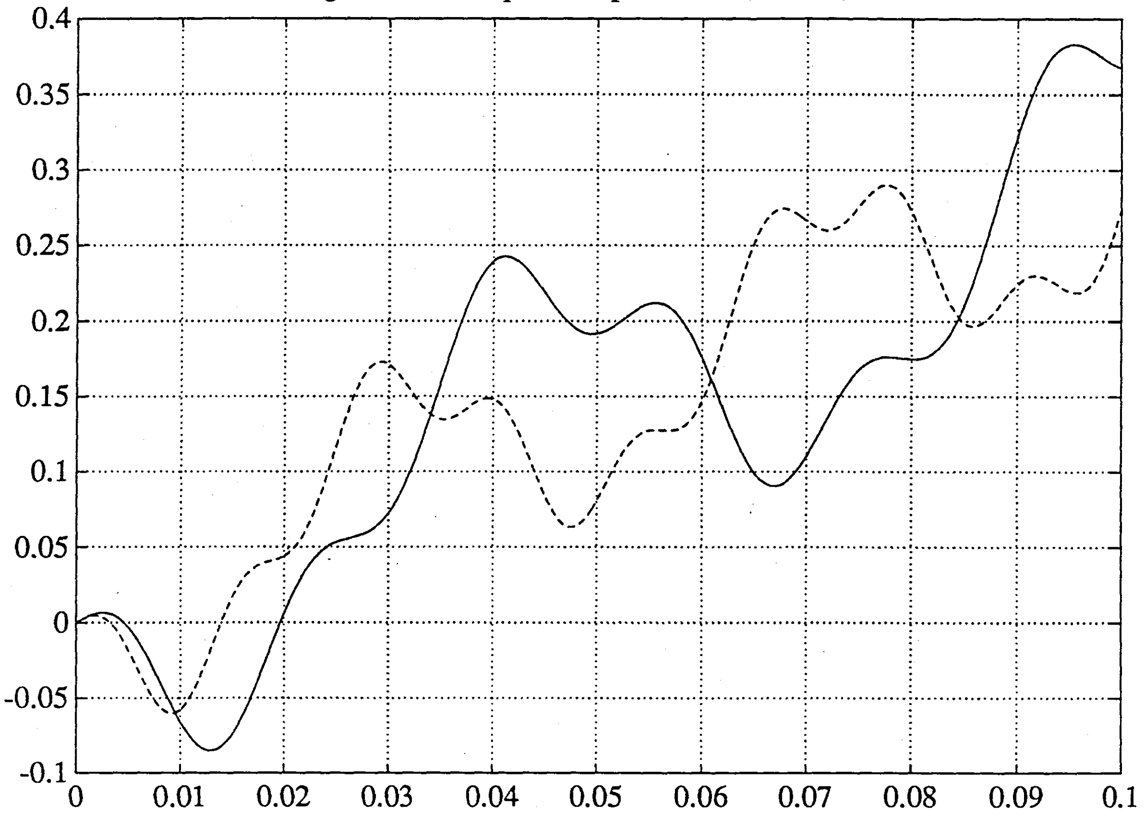


Fig.4.14. The impulse response of G, EI=200, 300

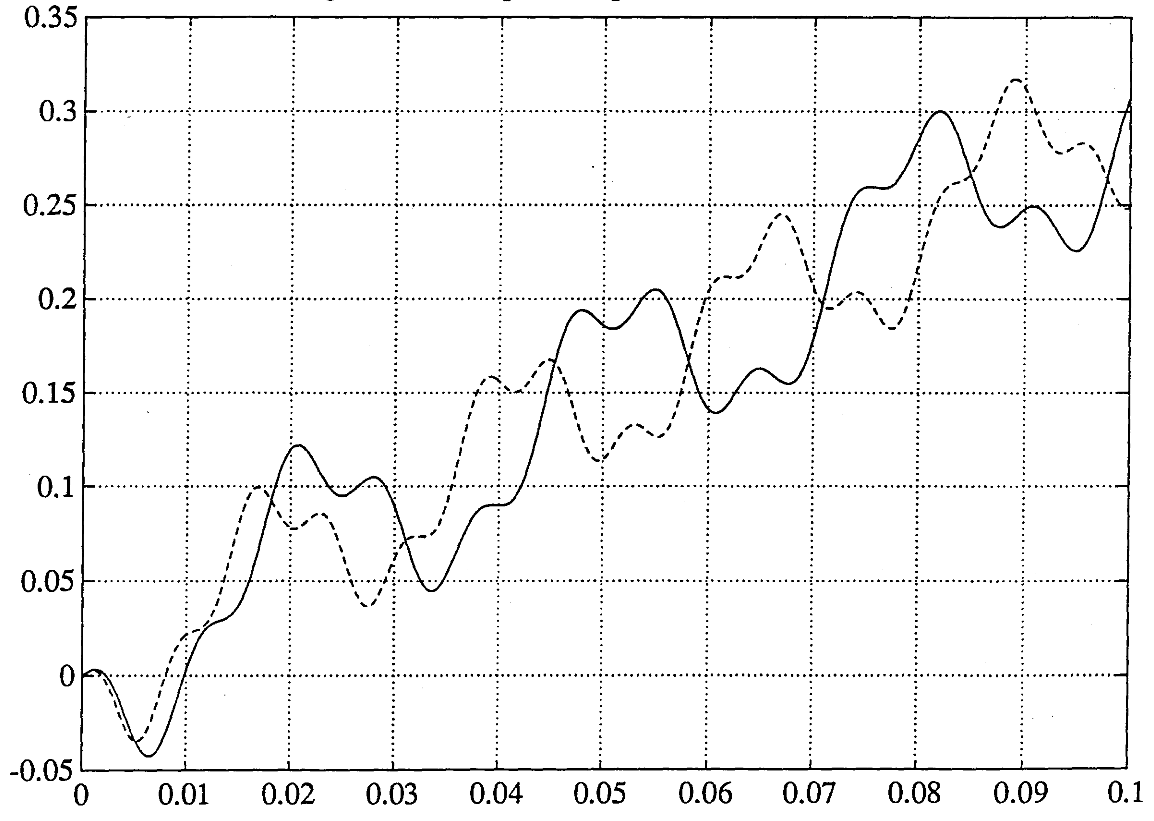


Fig.4.15. The F-R of G derived in time domain, EI=50, 100

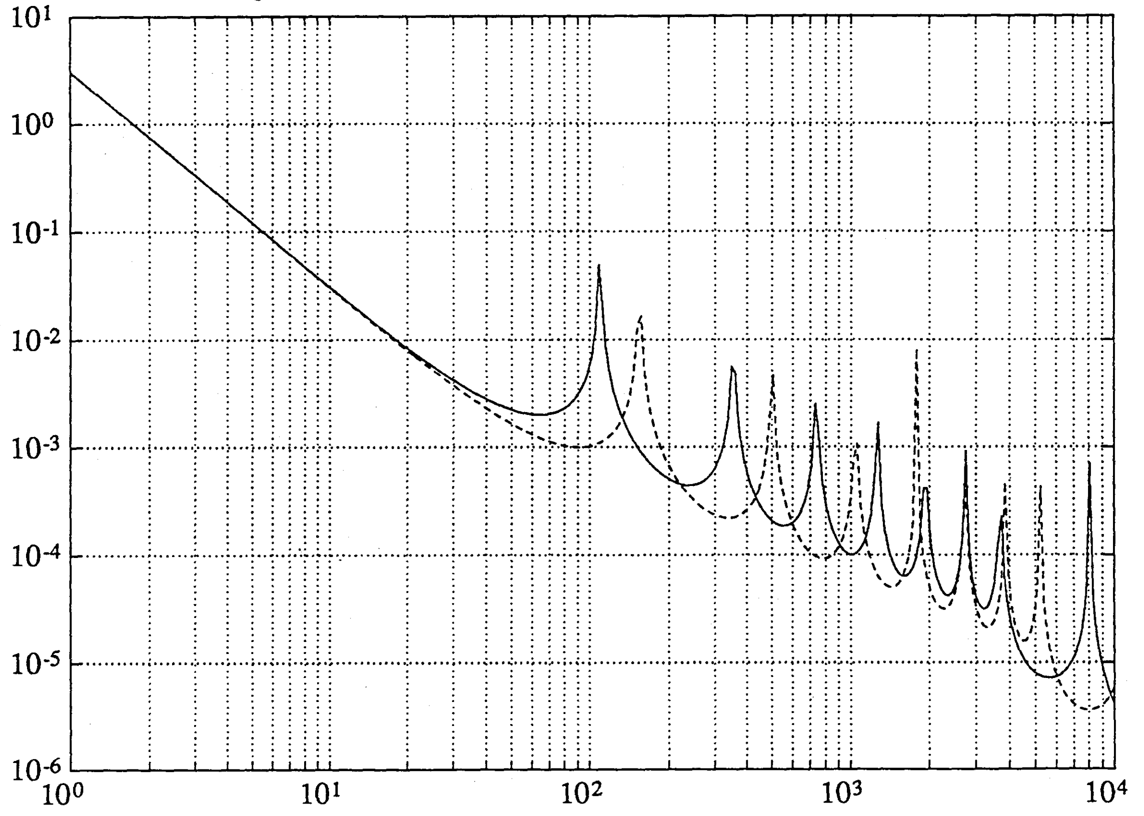


Fig.4.16. The F-R of G derived in time domain, EI=200, 300

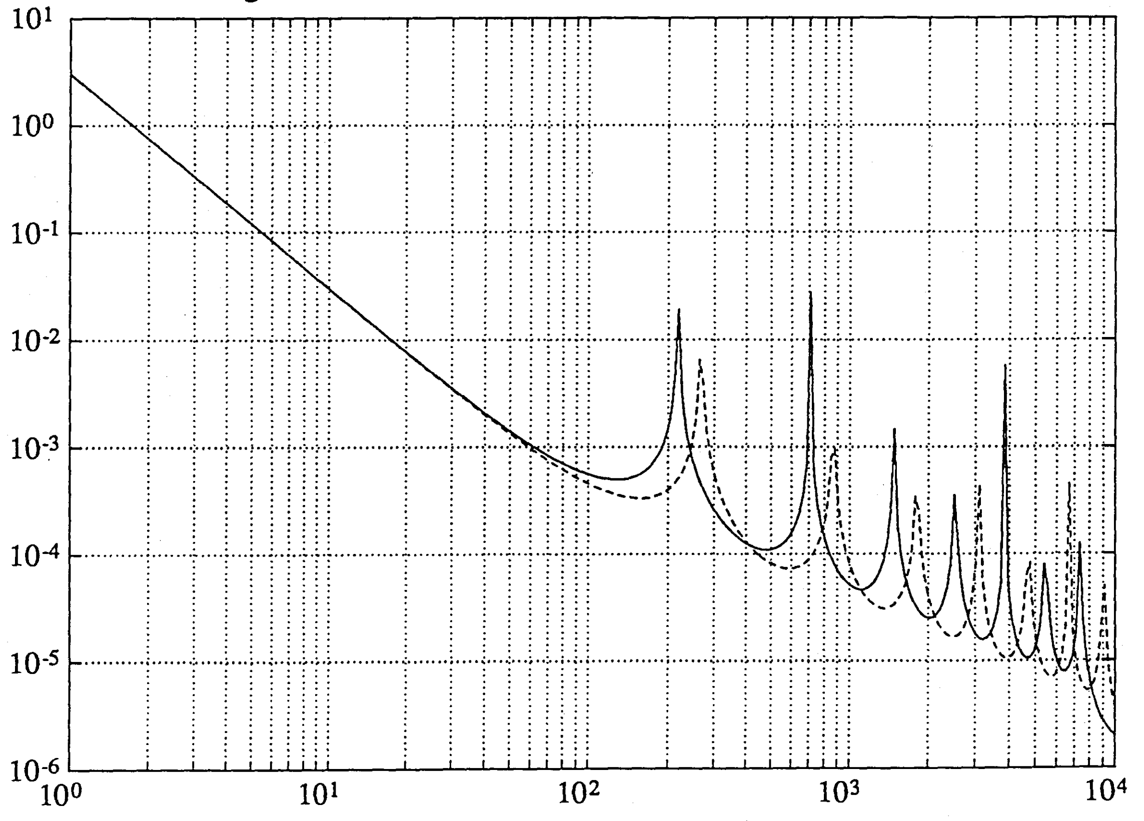


Fig.4.17. Compare of exact model and time domain model, EI=50

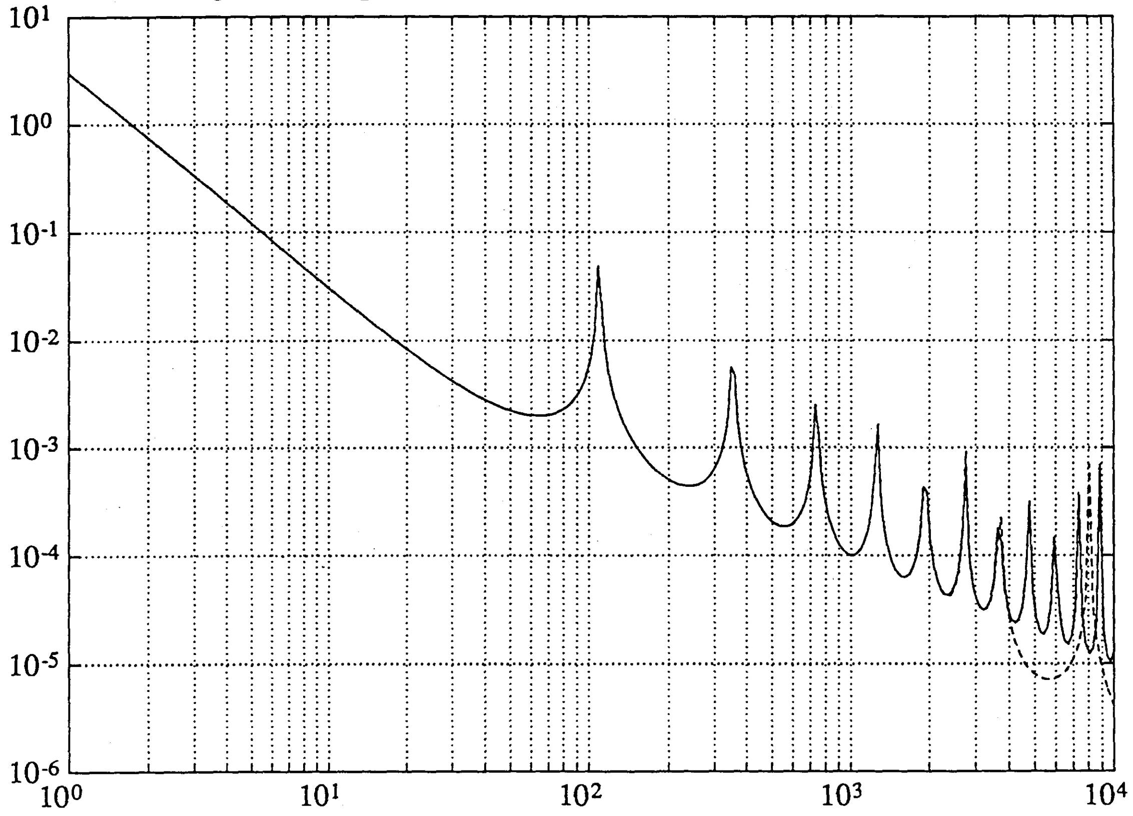


Fig.4.18. Compare of exact model and time domain model, EI=200

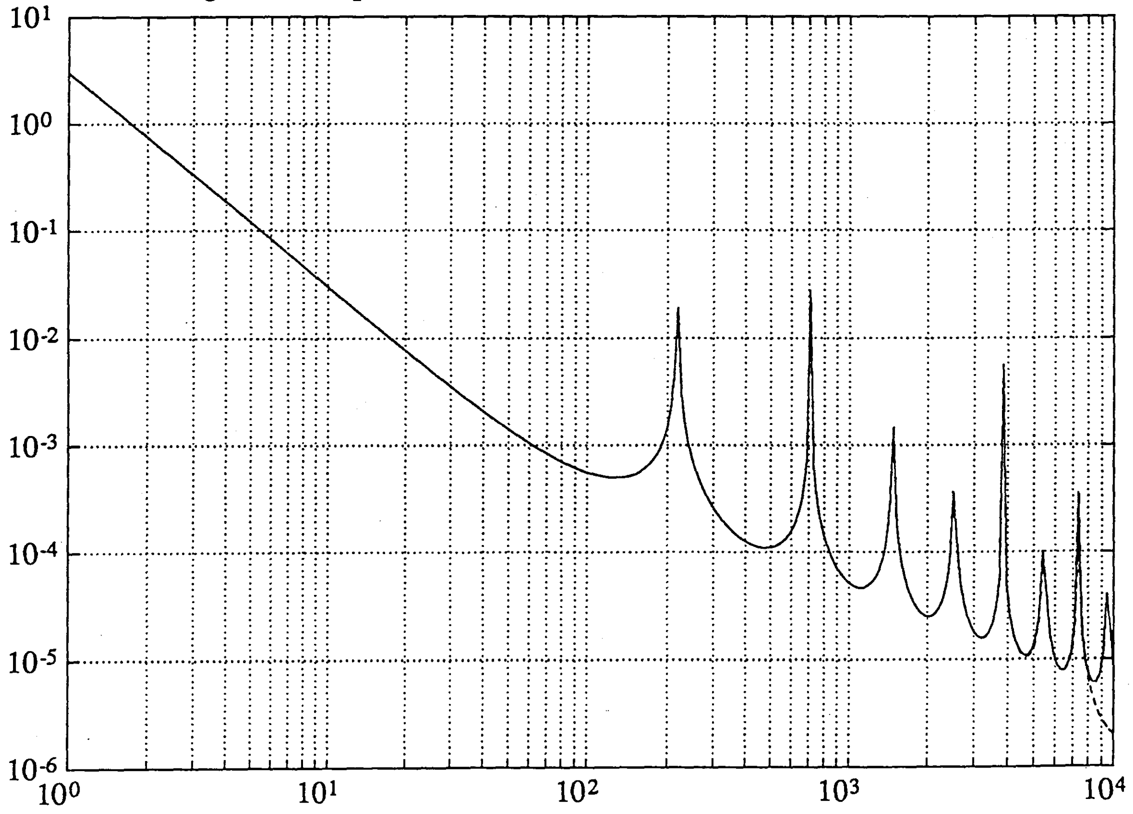


Fig.4.20. Step response of exact model G, EI=50

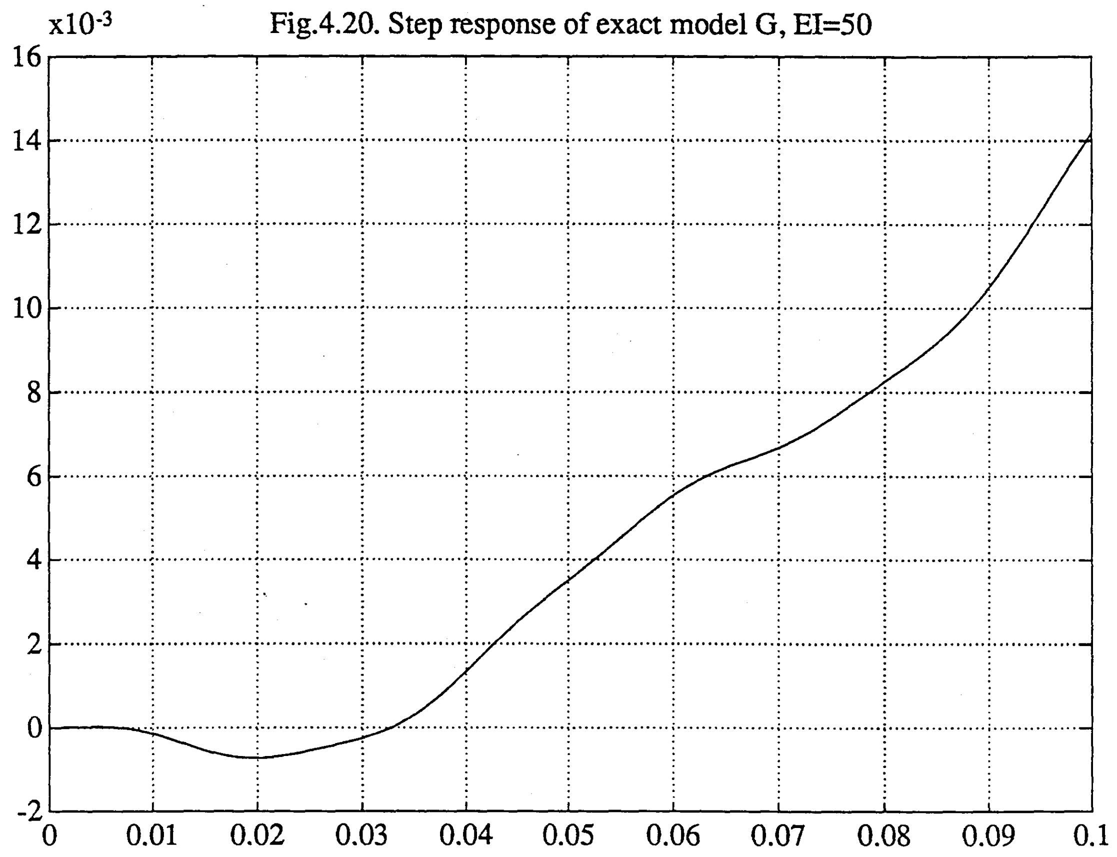


Fig.4.19. The response of the Lagrange model under unit torque input

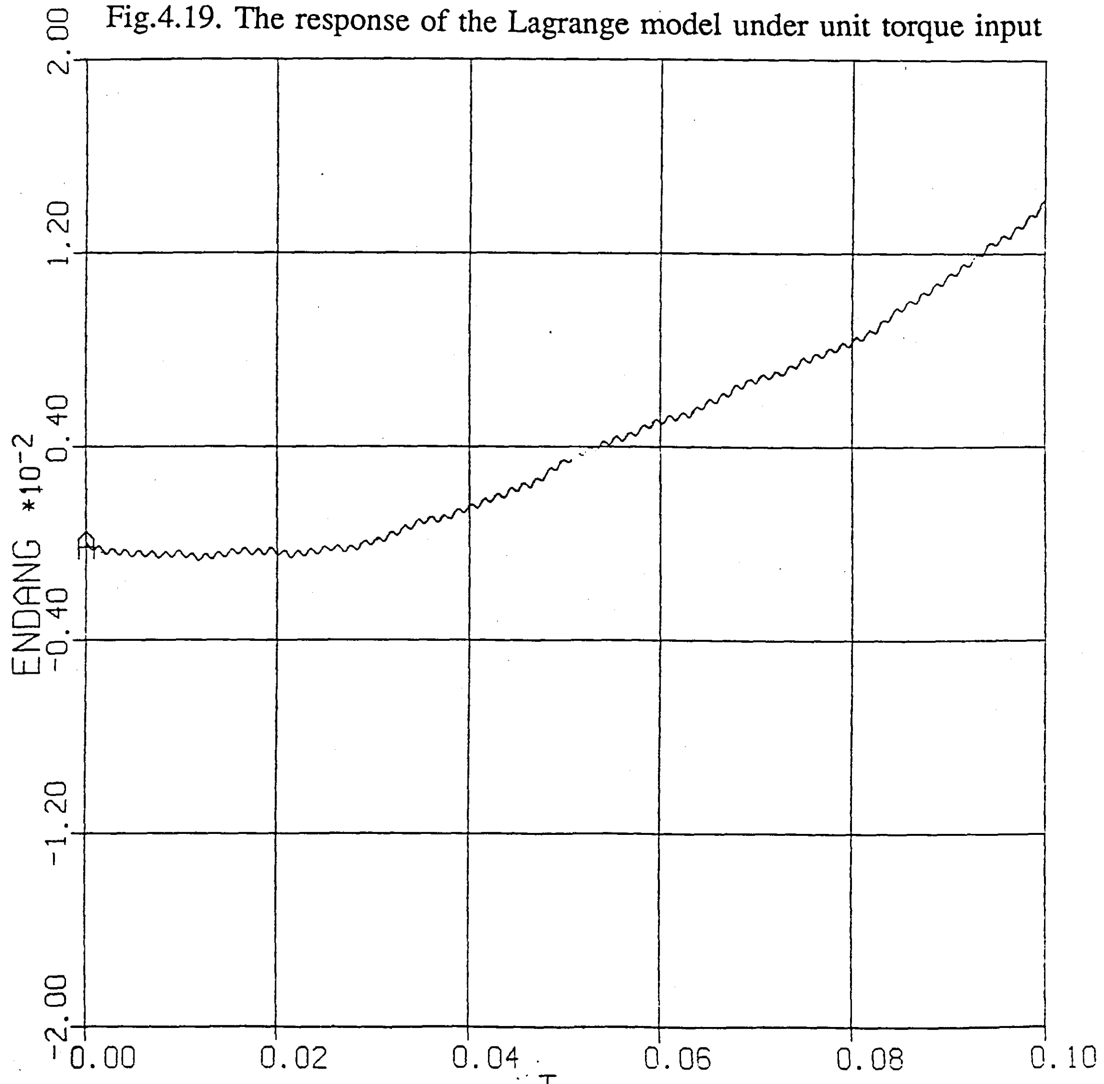


Fig.4.21. Comparison of exact model and adding springs model, EI=50

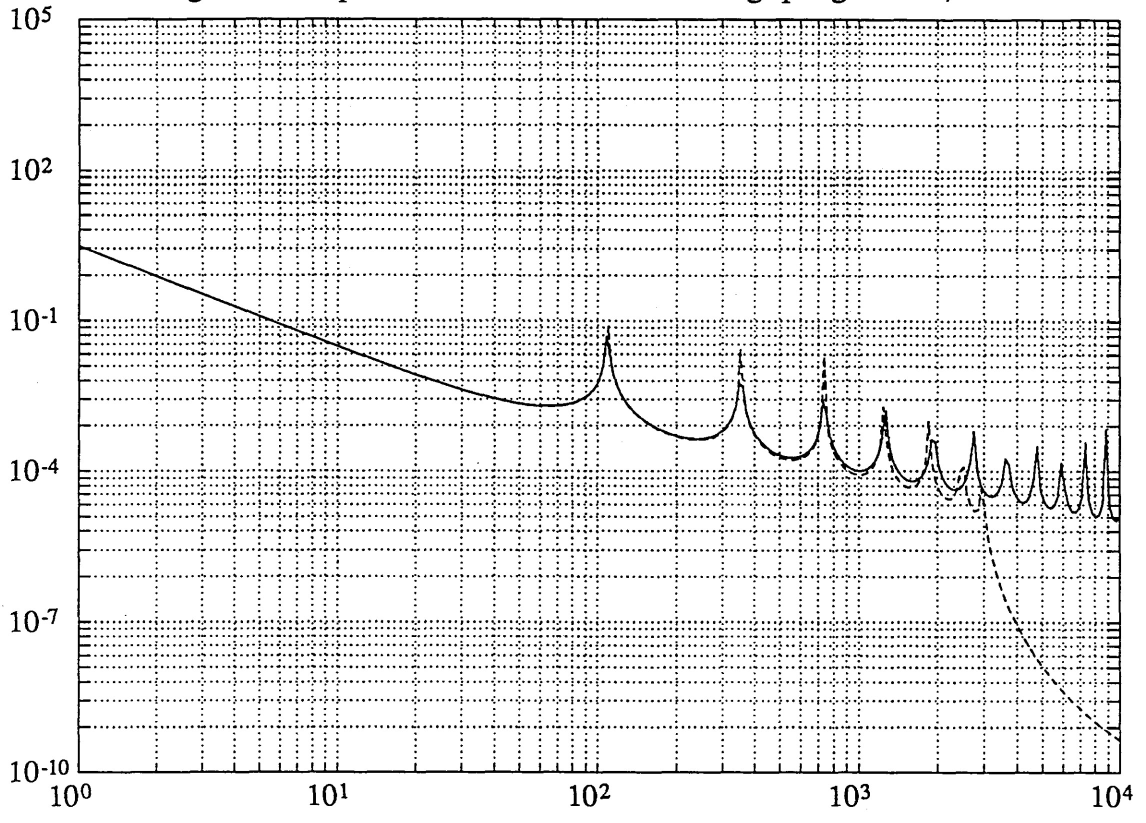


Fig.4.22. Comparison of exact model and adding springs model, EI=200

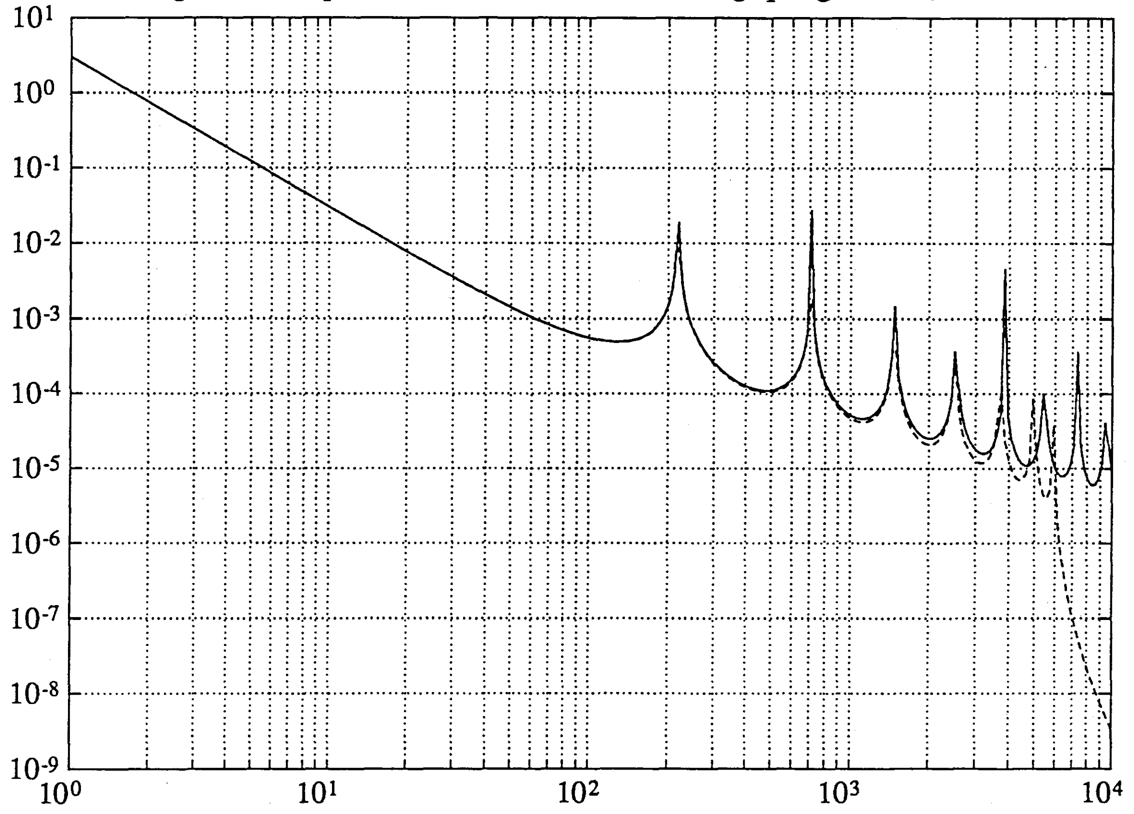


Fig.4.24. Comparison of experiment results and exact model

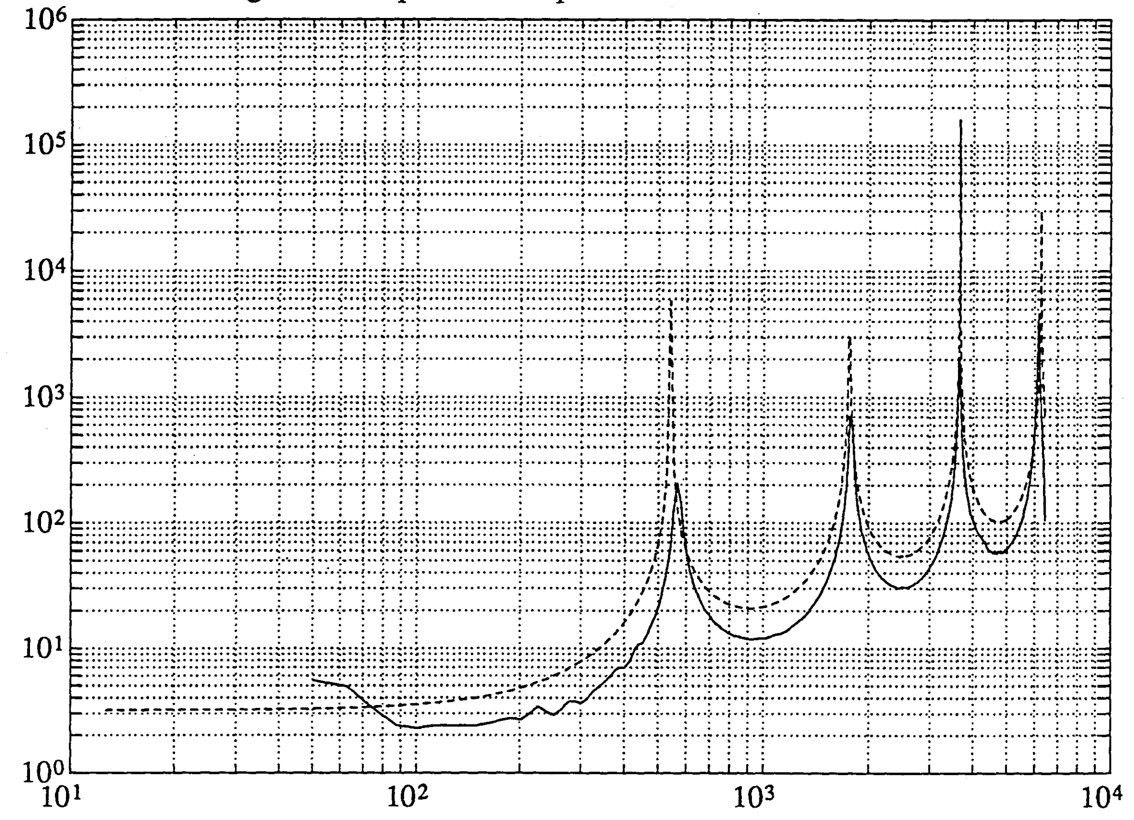


Fig.2.25. Comparison of forced model and exact model, a=0.05 M

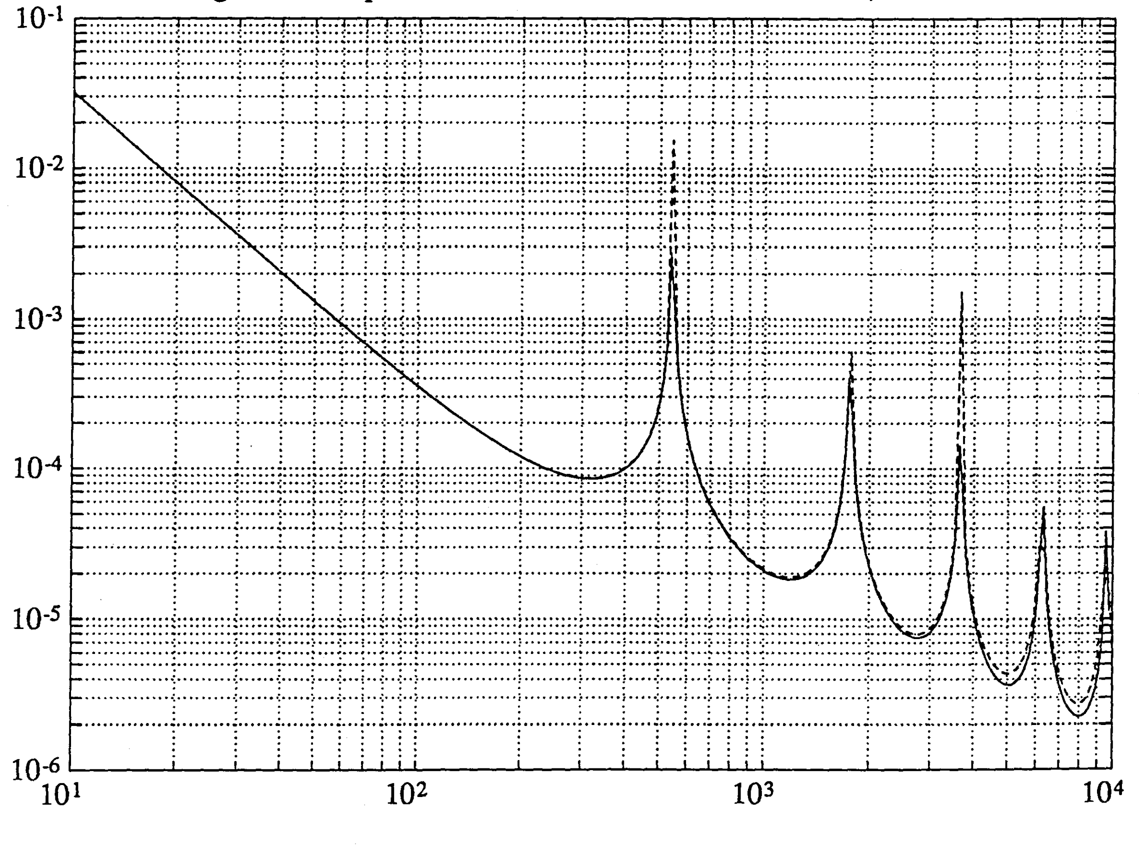
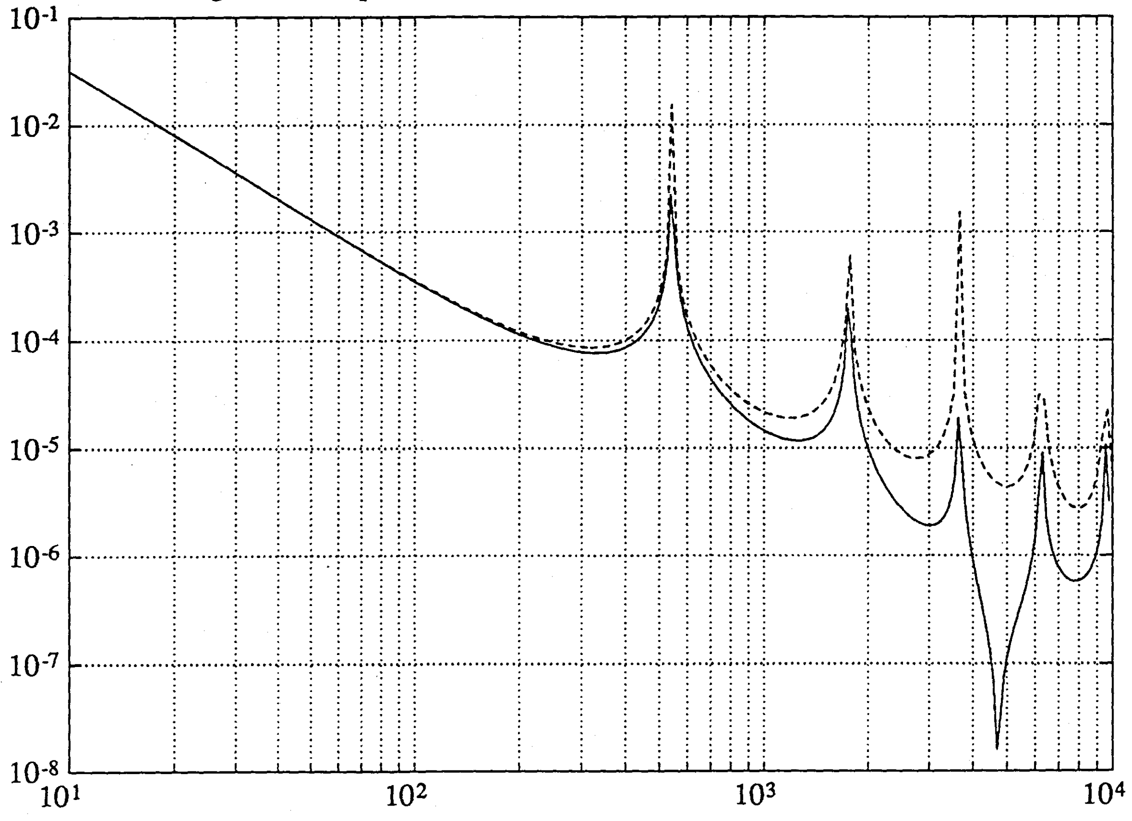


Fig.2.26. Comparison of forced model and exact model, a=0.20 M



Chapter 5. Simulation of Flexible Links of Manipulators

5. Simulation of Flexible Links of Manipulators

5.1. Introduction

The analytic models of one-link flexible manipulators in the previous chapter are quite satisfied. We use the method of adding springs to approximate the model. This approximation has been verified in frequency domain. Because the relative simplicity of the dynamic model obtained for a one-link flexible manipulator disappears altogether when considering a multi-link flexible manipulator, we should also verify the approximation in time domain.

This chapter continues to describe the validation work about flexible beams. It concentrates on the method of adding springs to simulate flexible links of manipulators. The basic method to verify the simulation results is to compare the natural frequencies of a cantilevered beam with the ones of the approximated model.

5.2. Bending vibration of a uniform beam

In order to simulate flexible links, we first analyse the transverse vibration of a beam. Referring to any standard text on strength of materials, for a beam, we find that the bending moment M is given by

$$- M = EI \frac{\partial^2 y}{\partial x^2} \tag{5-1}$$

while the corresponding transverse force on the beam per unit length q is given by

$$-q = \frac{\partial^2 M}{\partial x^2} = \frac{\partial^2}{\partial x^2}\left[EI\,\frac{\partial^2 y}{\partial x^2}\right] \tag{5-2}$$

and when the beam is uniform

$$-q = EI\,\frac{\partial^4 y}{\partial x^4} \tag{5-3}$$

If the beam is vibrating with a periodic motion we can always represent the vibration as a sum of a set of Fourier components which are all harmonic functions of time, and whose amplitudes are functions of x. For any mode of vibration the motion y(t) will be represented by $y_0(x)e^{j\omega t}$, where ω is the natural frequency of the mode and $y_0(x)$ is the mode shape. Assuming that the inertia and elastic forces are independent, the equation of motion in each mode is expressed by

$$\rho\frac{\partial^2 y}{\partial t^2} + EI\frac{\partial^4 y}{\partial x^4} = 0, \tag{5-4}$$

where ρ is the mass density of the link along length. Since y is a function of x and t, this is a form of wave equation with a wave velocity $\pm\sqrt{EI/\rho}$, and each mode can be represented by a standing wave. We can express the displacement as $y = \phi_1(x)\phi_2(t)$, where ϕ_1 and ϕ_2 are independent, so that $\frac{\partial y}{\partial x} = \frac{\partial \phi_1}{\partial x}\phi_2$ and $\frac{\partial y}{\partial t} = \frac{\partial \phi_2}{\partial t}\,\phi_1$. Thus for each mode we can replace $\frac{\partial^2 y}{\partial t^2}$ by $-\omega^2 y$ in equation (5-4), resulting in the pair of ordinary differential equations

$$\frac{d^2 y}{dt^2} + \omega^2 y = 0 \tag{5-5}$$

and

$$\frac{d^4\Phi(x)}{dx_4} - \frac{\rho\omega^2\Phi(x)}{EI} = 0, \tag{5-6}$$

where equation (5-6) gives the shape of each mode $\Phi(x)$ corresponding to the value of ω.

To solve the equation (5-6) we need to find a function whose fourth derivative is equal to itself. There are now four possibilities, namely $e^{\lambda x}$,$e^{-\lambda x}$, $e^{j\lambda x}$, and $e^{-j\lambda x}$. Since the general solution to a fourth order differential equation will require four arbitrary constants of integration, the general solution to equation (5-6) will be

$$y_0(x) = B_1 e^{\lambda x} + B_2 e^{-\lambda x} + B_3 e^{j\lambda x} + B_4^{-j\lambda x}, \qquad (5\text{-}7)$$

or alternatively

$$y_0(x) = B_1 \cosh \lambda x + B_2 \sinh \lambda x + B_3 \cos \lambda x + B_4 \sin \lambda x \qquad (5\text{-}7a)$$

where

$$\lambda^4 = \frac{\rho\omega^2}{EI}, \qquad (5\text{-}8)$$

while the general solution to equation (5-5) is given by

$$y = \Phi(x)\, e^{j\omega t}. \qquad (5\text{-}9)$$

We now have the problem of finding the constants in equation (5-7), for which we require four boundary conditions. These can be determined from the constraints placed at the ends or other parts of the beam when the problem is originally defined.

5.3. The natural frequency of a uniform cantilevered beam

Assuming the beam's length is L, the constraint at the fixed end is such that the beam has zero deflection and slope there. At the free end there is zero moment and shear force so the boundary conditions for this problem are

$$\Phi(0) = 0, \dot{\Phi}(0) = 0, \ddot{\Phi}(L) = 0, \Phi^{(3)}(L) = 0.$$

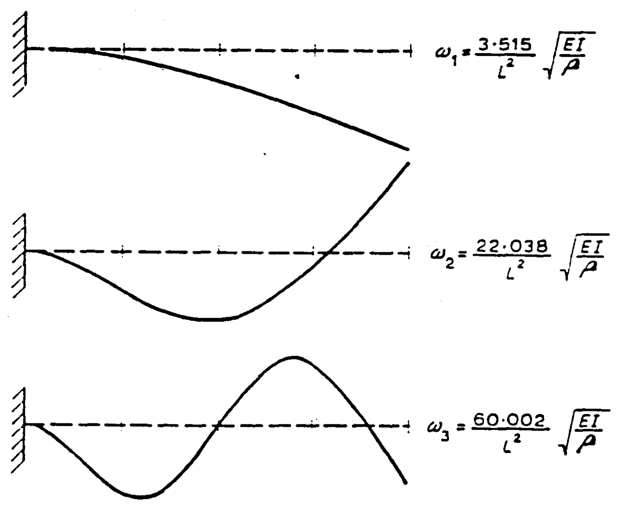

$\omega_1 = \frac{3.515}{L^2}\sqrt{\frac{EI}{\rho}}$
$\omega_2 = \frac{22.038}{L^2}\sqrt{\frac{EI}{\rho}}$
$\omega_3 = \frac{60.002}{L^2}\sqrt{\frac{EI}{\rho}}$

Fig.5.1 The natural frequencies of the first three modes

The first two of these conditions gives

$$B_1 + B_3 = 0, \text{ and } B_2 + B_4 = 0,$$

while the other pair of conditions gives

$$B_1 \cosh \lambda L + B_2 \sinh \lambda L - B_3 \cos \lambda L - B_4 \sin \lambda L = 0,$$

$$B_1 \sinh \lambda L + B_2 \cosh \lambda L + B_3 \sin \lambda L - B_4 \cos \lambda L = 0,$$

from which it may be shown that the frequency equation is given by

$$\cos \lambda L \cosh \lambda L = -1 \tag{5-10}$$

and the first three wave numbers are then $\lambda_1 = 1.875/L, \lambda_2 = 4.694/L$ and $\lambda_3 = 7.855/L$ corresponding to the first three modes of vibration of the cantilevered beam.

The natural frequencies for the first 3 modes which are illustrated on figure 5.1 are therefore

$$\omega_1 = \frac{3.515}{L^2}\sqrt{\frac{EI}{\rho}},\ \omega_2 = \frac{22.038}{L^2}\sqrt{\frac{EI}{\rho}} \text{ and } \omega_3 = \frac{62.002}{L^2}\sqrt{\frac{EI}{\rho}}.$$

5.4. The response of adding springs in the cantilevered beam

We think a flexible link can be replaced by several rigid parts connected with springs, if only their natural frequencies are more or less the same. The problem of accomplishing this is that how many rigid links we should need and how large the stiffness of the springs we must choose.

Assume the number of the added spring is n. The distribution of the springs is the same as described in section 4.5 of chapter 4. The springs' stiffness can be chosen by equation (4-38) or (4-38a).

5.4.1. The model approximated by two rigid parts.

In order to simplify the model, we assume the shape of the flexible link is shown below

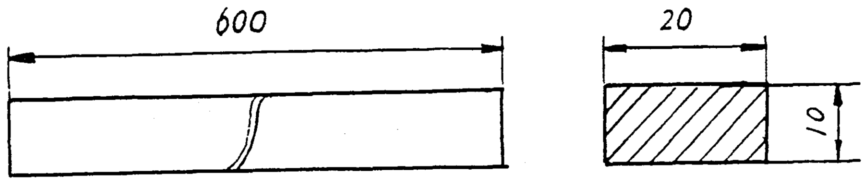


Fig.5.2 The shape of the flexible link

and the material of the link is steel, then we can choose the Young's modulus $E = 200\ GPa$, and the second moment of area $I = \frac{h^3 b}{12} = \frac{0.01^3 \times 0.02}{12} = 1.667\times10^{-9}\ m^4$, mass density along length $\rho = \frac{7800\times0.02\times0.01\times0.6}{0.6} = 1.56\ kg/m$. Then,

$$\omega_{n1} = \frac{3.515}{L^2}\sqrt{\frac{EI}{\rho}} \approx 142.7 \ \ rad/s, \ \ \omega_{n2} = \frac{22.038}{L^2}\sqrt{\frac{EI}{\rho}} \approx 894.8 \ \ rad/s.$$

According to equation (4-38), the stiffness of the springs is $k = 1666\,Nm$. By using REDUCE programmes described in chapter 2, we can obtain the dynamic equations of the model in vertical plane. The generated equations have the general form of equation (2-14).

Because each joint is connected with a spring, the torque acted on the joint is $\tau = -k\theta$, where θ is the generalised joint coordinate vector. By using ACSL, we can get the transient response of the end point under the effect of gravity.

Figure 5.3 shows us that the frequency of the transient response of the end point is

$$\omega = 114.8 \ rad/s$$

the difference between ω and the first mode natural frequency ω_{n1} is

$$\eta = \frac{|\omega_{n1} - \omega|}{\omega_{n1}} = 19.58 \ \%$$

This result shows that the model approximated by two rigid parts with two springs is quite satisfied. The difference is caused by the less rigid parts used for the model.

According to equation (4-38a), the stiffness of the springs is $k = 1111\,Nm$. The transient response of the end point of the model is shown in figure 5.4. The frequency of the end point is

$$\omega = 91.4 \ rad/s$$

the difference between ω and ω_{n1} is

$$\eta = \frac{|\omega_{n1} - \omega|}{\omega_{n1}} = 35.95 \ \%$$

having compared this result with the one in figure 5.3, we find that it is better to use equation (4-38) than equation (4-38a) to calculate the springs' stiffness.

5.4.2. The model approximated by three rigid parts

The transient response of the end point of the model approximated by three rigid parts is shown in figure 5.5. The frequency of the end point is is

$$\omega = 120.7 \; rad/s$$

the difference between ω and ω_{n1} is

$$\eta = \frac{|\omega_{n1} - \omega|}{\omega_{n1}} = 15.41 \;\%$$

Compared with the case in section 5.4.1, it is a little better. From figure 5.3 and figure 5.5, we notice that their end point response' shapes are almost the same. That is these two cases show a same characteristic which is the real flexible link's.

5.4.3. The model approximated by four rigid parts

The transient response the end point of the model approximated by four rigid parts is shown in figure 5.6. From figure 5.6, the frequency of the transient response is $\omega = 123.8 \; rad/s$. In this case, the difference η is

$$\eta = \frac{|\omega_{n1} - \omega|}{\omega_{n1}} = 13.23 \;\%$$

The more rigid parts are used, the more approximated model is obtained. However, it is satisfied to simulate the real flexible link by using four rigid parts.

5.5. Conclusion

From the results of the simulation in real time domain, we can conclude that the approximate method by adding springs to connect equal length rigid parts to simulate real flexible links is reasonable. It can be used as a reference to the simulation of multi-link flexible manipulators.

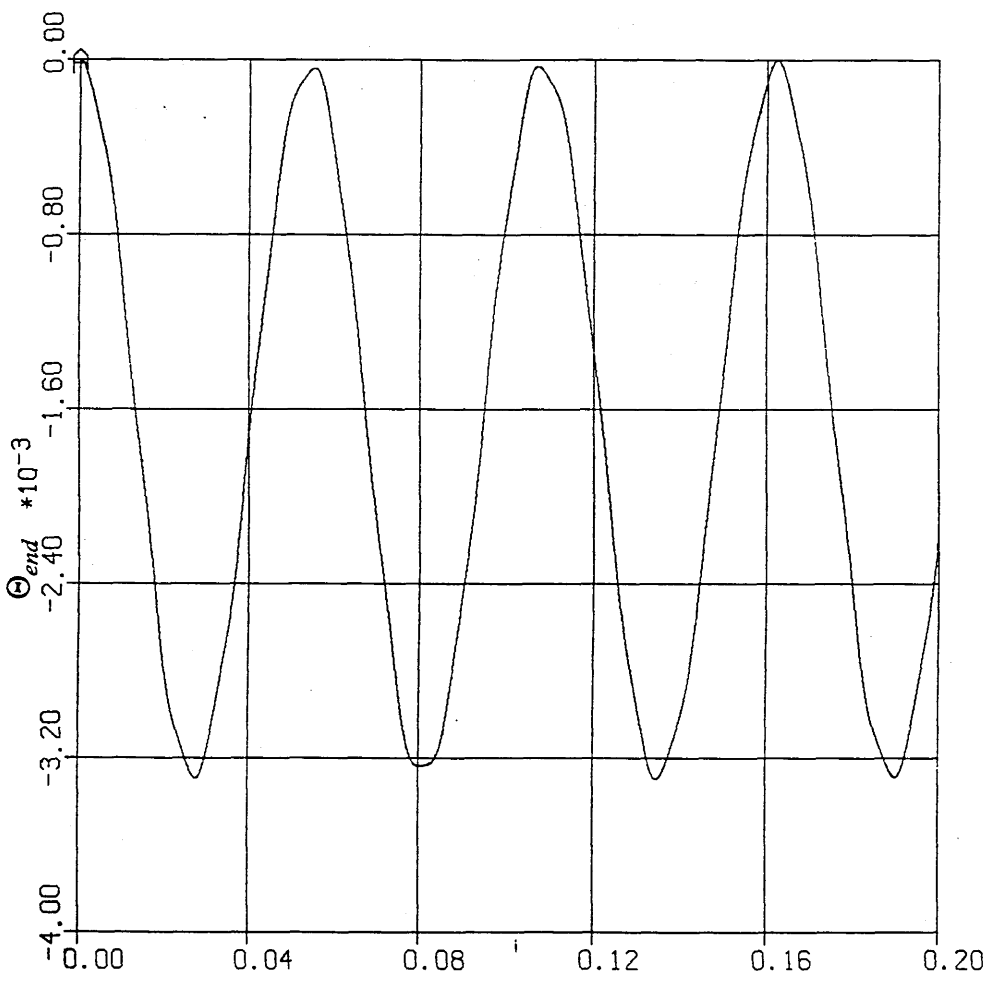


Fig.5.3 The end-point transient response of adding two springs

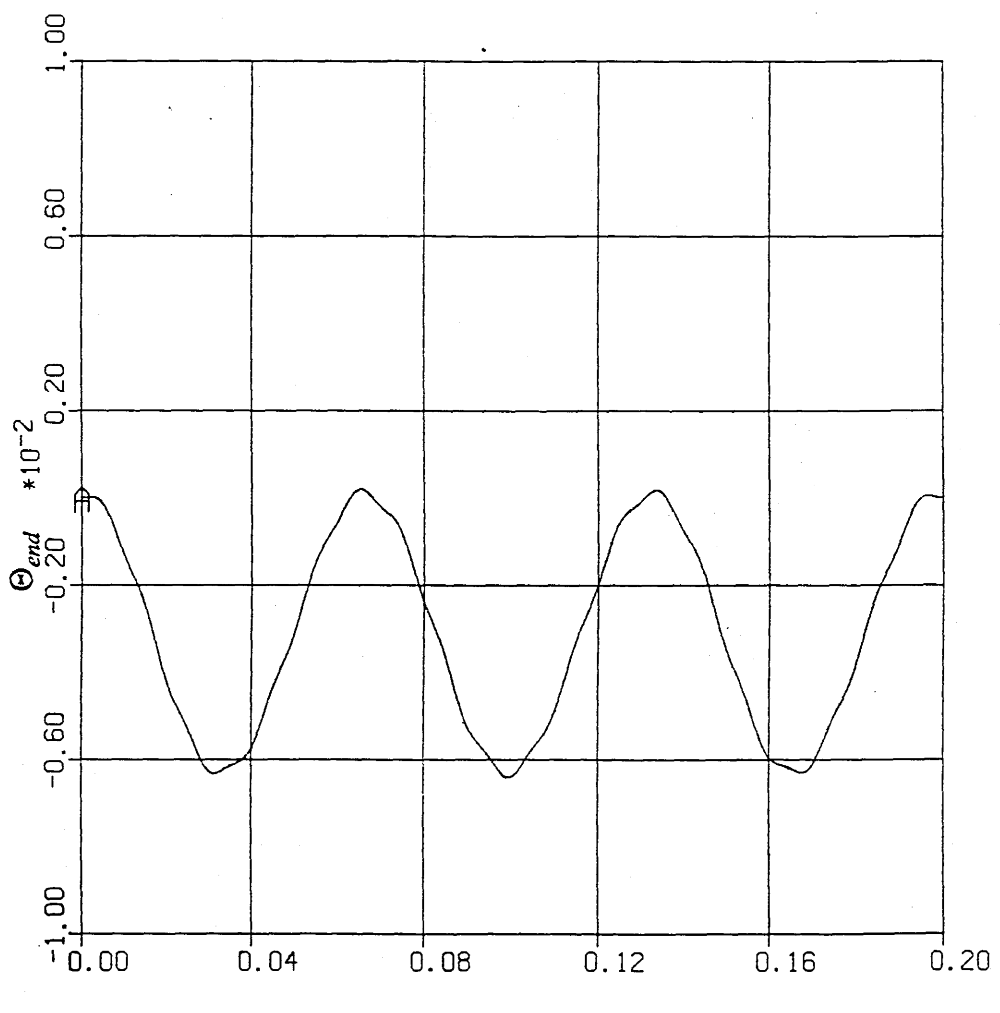


Fig.5.4 The end-point transient response of adding two springs

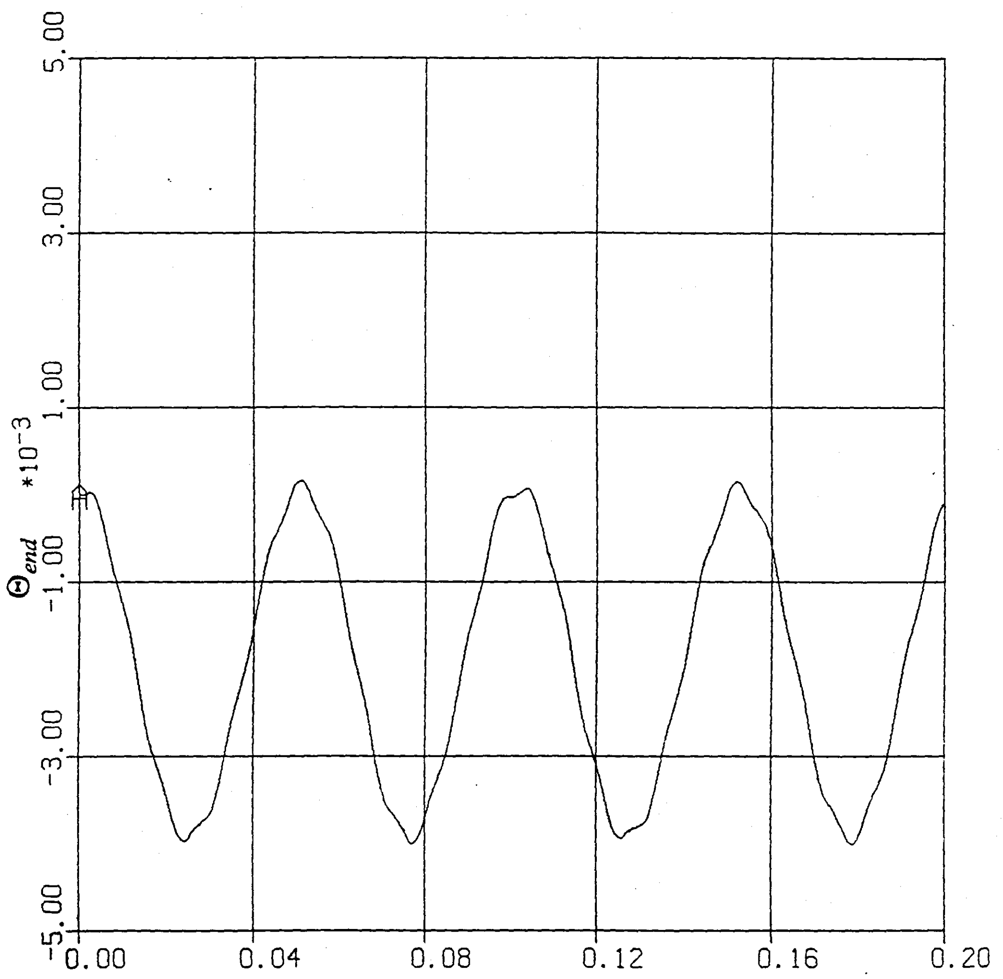


Fig.5.5 The end-point transient response of adding three springs

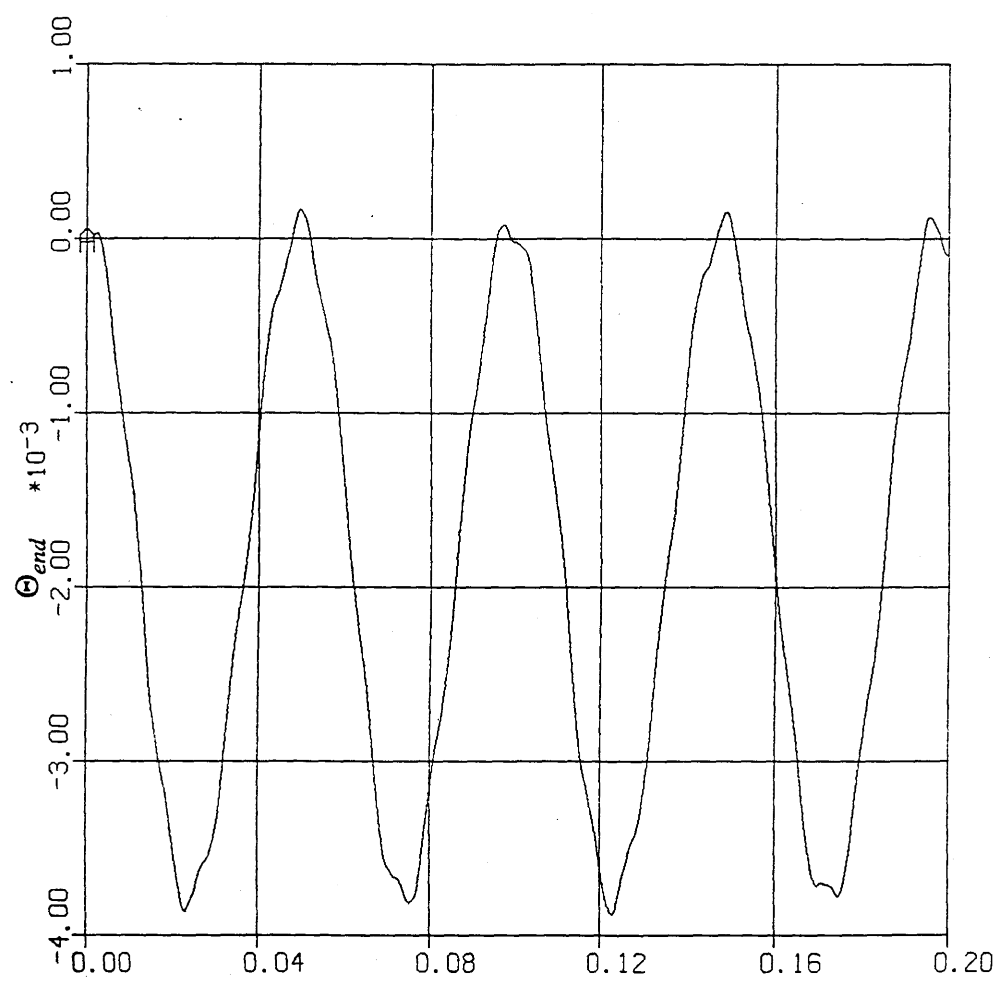


Fig.5.6 The end-point transient response of adding four springs

Chapter 6 Friction Effects on DC Motors

6. Friction Effects on DC Motors

6.1. Introduction

Robot validation involves a wide range of research topics. Chapter 4 and chapter 5 focus on the realisation of characteristics of flexible links. This chapter focuses on actuating systems.

Actuating systems provide a robot with muscle power. They are energy conversion devices, converting electrical, hydraulic or pneumatic power to mechanical power. With respect to robot applications, electrical power offers several advantages. One is that electric actuators are easy to control. In order to identify the parameters of an electric actuator system, we must know the friction effects on motors. This chapter changes its point to the verification of friction effects on DC motors.

The Quin DSC-1 digital device provides a convenient control system which is essentially described in a continuous-time form. This chapter is based on the report [30] to identify the friction effects on the DC motor used in the system.

6.2. System identification

6.2.1. Least-squares method

The friction effects on a DC motor are assumed in the form as described by the figure below

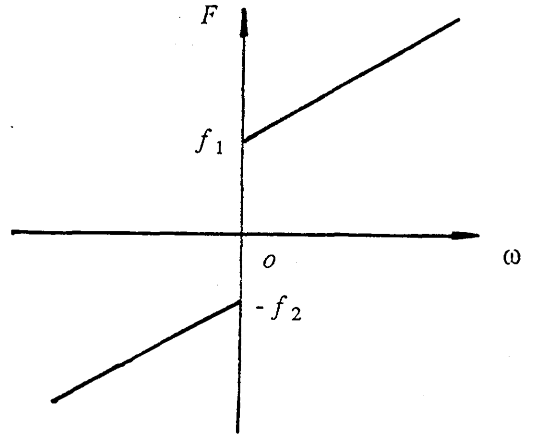


Fig.6.1 Friction effects of DC motors

where F is the resistant torque which includes static friction (f_1, f_2) and dynamic friction (the slope of the lines), ω is the rotation speed of the motor.

If we add a fixed inertia load, the system can be described by

$$y(s) = \frac{1}{a_0 s^2 + a_1 s + a_2} \left(u(s) - f_1 sign(\omega + |\omega|) - f_2 sign(\omega - |\omega|) \right) \quad (6\text{-}1)$$

where $y(s)$ is the (Laplace transformed) motor's rotation angle, $u(s)$ is the (Laplace transformed) control signal input, a_0 is the sum of the load inertia and the motor shaft inertia, a_1 is the slope of the dynamic friction, a_2 is the term proportional to y (a_2 usually equals to zero. We assume that there is a zero order term to complete a second order system. From system parameters identifications, we can find that the value of a_2 approximates to zero), f_1 and f_2 are constants of resistant torques. With this assumption, there are five unknown parameters : $a_0, a_1, a_2, f_1, f_2,$. To perform system identification, the system is rewritten as:

$$\frac{1}{Cs(s)}u(s) = \frac{a_0 s^2}{Cs(s)}y(s) + \frac{a_1 s}{Cs(s)}y(s) + \frac{a_2}{Cs(s)}y(s) +$$

$$\frac{f_1}{Cs(s)}sign(\omega + |\omega|) + \frac{f_2}{Cs(s)}sign(\omega - |\omega|) \tag{6-2}$$

where $Cs(s)$ is a second-order polynomial.

This can be converted into the time domain and written in the standard form for Least-Squares estimation as:

$$\phi(t) = X^T(t)\theta \tag{6-3}$$

where the filtered output $\phi(s)$ is

$$\phi(s) = \frac{1}{Cs(s)}u(s) \tag{6-4}$$

the Laplace transformed data vector X(s) is

$$X(s) = [\frac{s^2 y(s)}{Cs(s)}\ \frac{sy(s)}{Cs(s)}\ \frac{y(s)}{Cs(s)}\ \frac{sign(\omega + |\omega|)}{Cs(s)}\ \frac{sign(\omega - |\omega|)}{Cs(s)}]^T \tag{6-5}$$

and the parameter vector θ is

$$\theta = [a_0\ a_1\ a_2\ f_1\ f_2]^T \tag{6-6}$$

The unknown parameter vector θ can be estimated from the data using the standard least-squares solution which has the form

$$\theta = [\sum_{t=\Delta}^{N\Delta} X(t)X^T(t)]^{-1} \sum_{t=\Delta}^{N\Delta} X(t)\phi(t) \tag{6-7}$$

where N is the number of applied data, Δ is the sampling time interval.

6.2.2. Control signal in static state

The motor control system operates by sampling the position of the motor at regular intervals which is $\Delta = 1/256$ S, and calculating a motor control signal according to some control algorithm. The algorithm used is of the following form:

$$u_i = K_p e_i + K_i \sum_{j=1}^{i} e_i + K_d (e_i - e_{i-1}) - K_v (y_i - y_{i-1}) + K_f (w_i - w_{i-1}) \qquad (6\text{-}8)$$

where u_i is the control signal at time i, y_i is the system output at time i, and w_i is the demand signal at time i, $e_i = w_i - y_i$ is the error signal.

The electric voltage output of the amplifier used in the DSC-1 module usually has an initial value V_{out_0} when the input signal is zero. The value of initial electric voltage output V_{out_0} can be measured in static state. First, set K_i, K_d, K_v, K_f to zero. Second, choose a proper value of K_p, for example, K_p = 100,200,300 respectively. Third, add a moment, which is done by hand, that is to cause the shaft to have a deviation angle, this will cause the amplifier to have an output voltage which is read from a voltage meter, and the voltage value is proportional to the control signal. Repeat these three steps, we can obtain another point of voltage value versus control signal value. Therefore, we can draw the curves of static state control signals versus electric voltage outputs.

Three curves of the static state control signals versus electric voltage outputs are shown in figure 6.2 when $K_p = 100, K_p = 200, K_p = 300$ respectively. From figure 6.2, we find $V_{out_0} = -0.73$ Volt. These curves should not depend on the value of K_p, that is they should be a coincident slope line. The differences of the curves are caused by the low precision of the measuring method.

6.2.3. Simulation results

If the assumed friction effects which are described in figure 6.1 are right, we can write MATLAB files to identify the unknown parameters θ. These system identification files can be tested by using simulated data.

Assuming a system has the parameters $\theta = [0.8\ 3\ 0.5\ 26\ 10]^T$. The control signal is chosen from any measured data of the real system, for example, from the the data file when $K_p = 100$. After running the MATLAB files, the identified system parameters are $\theta = [0.7751\ 2.8477\ 0.4992\ 26.0115\ 9.9955]^T$. These simulation results which are shown in figure 6.3 imply that the MATLAB files can be applied to the system identification.

6.2.4. Experimental results

After running the data files when $K_p = 100, 200, 300$, we get the results which are shown in figures 6.4 - 6.9:

$$K_p = 100: \quad \theta = [0.6489\ 1.8741\ 0.1182\ 18.7031\ 11.7938]^T$$

$$K_p = 200: \quad \theta = [0.6579\ 1.7145\ 0.2375\ 19.0903\ 13.5122]^T$$

$$K_p = 300: \quad \theta = [0.6026\ 1.8979\ -0.4106\ 21.0599\ 4.0251]^T$$

Figure 6.4, 6.6, 6.8 are the simulation and actual system outputs when $K_p = 100, 200, 300$ respectively, and the related control signals are shown in figures 6.5, 6.7, 6.9.

The identified parameters under the different values of K_p should have same values. We are quite happy with the results except the values of a_2 and the value of f_2 when $K_p = 300$.

The major differences between f_2 when $K_p = 300$ and the one when $K_p = 200$, or $K_p = 100$ can be explained from the errors between the real control signals input to the system and the estimated control signals used for the identification. Figure 6.9 shows the system control signals when $K_p = 300$. They are bounded in a certain scope because of the capacity of the amplifier. The capacity of the amplifier is in the range from -12 Volts to +12 Volts, and the related scope of the control signals is in the range from -125 to +150. They are shown in figure 6.2. When the control signal is beyond the restricted scope, the related amplifier voltage is not strictly bounded at the point of +12 (or -12) Volts, it can reach as high as +12.5 (or -12.5) Volts. Therefore, the relationship between the control signal and the electric voltage output which is proportional to the torque input is nonlinear when the control signal is beyond the scope. The nonlinear output of the amplifier causes the estimated control signal unreliable.

In figure 6.9, when the rotation speed of the motor is positive, the control signals are below the restricted scope, therefore, the value of f_1 is reasonable compared with the other values of f_1 when $K_p = 200, 100$. However, when the rotation speed is negative the control signals are bounded by the capacity of the amplifier, therefore, the wrong identified value of f_2 (compared with the other values of f_2) is not strange.

We know that the system of a motor with a fixed rotation inertia load has a zero term of a_2. Therefore, we can assume $a_2 = 0$ In this case, we run the MATLAB files with the same data files as above, we get the identified parameters values:

$$K_p = 100: \quad \theta = [0.6411\ 1.8911\ 0\ 19.1033\ 10.9335]^T$$

$$K_p = 200: \quad \theta = [0.6410\ 1.7029\ 0\ 20.4705\ 12.6031]^T$$

$$K_p = 300: \quad \theta = [0.6326\ 1.8991\ 0\ 18.8158\ 6.0287]^T$$

What we must note at here is that the simulation results are all obtained by open-loop simulation process. The errors will be accumulated during the whole process. Therefore, the simulation results may have more and more errors compared with the real system shaft angle outputs when time becomes larger and larger.

6.3. Conclusion

The differences of the identified system parameters from different data files can be explained by (1) poor quality power amplifier being used, (2) low precision measuring method being applied and (3) some unknown factors in the Quin DSC-1 digital device. Even so, the results show that the friction effects on DC motors are describes by figure 6.1.

Fig6.2. Output volts versus Control signals, Kp=100, 200, 300

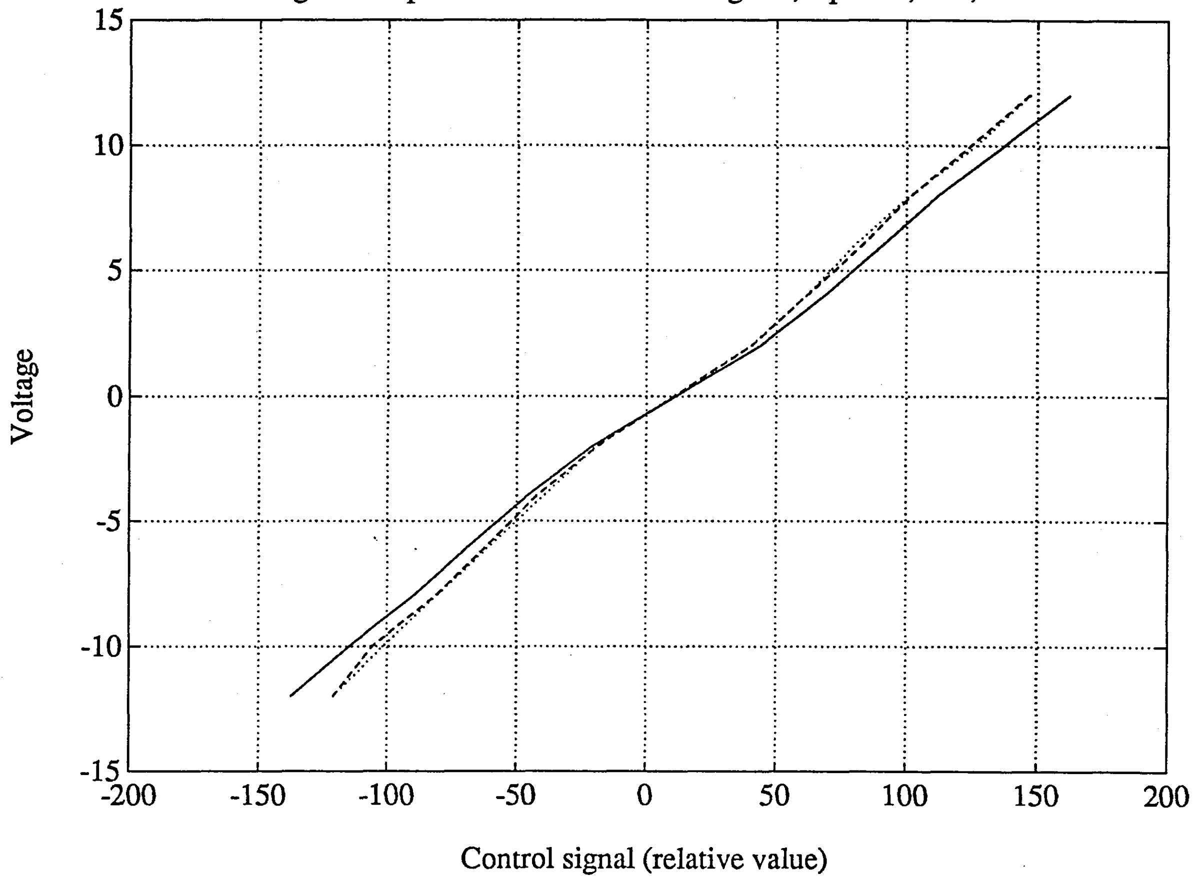


Fig6.3. Simulation results, Assumed and Identified system outputs

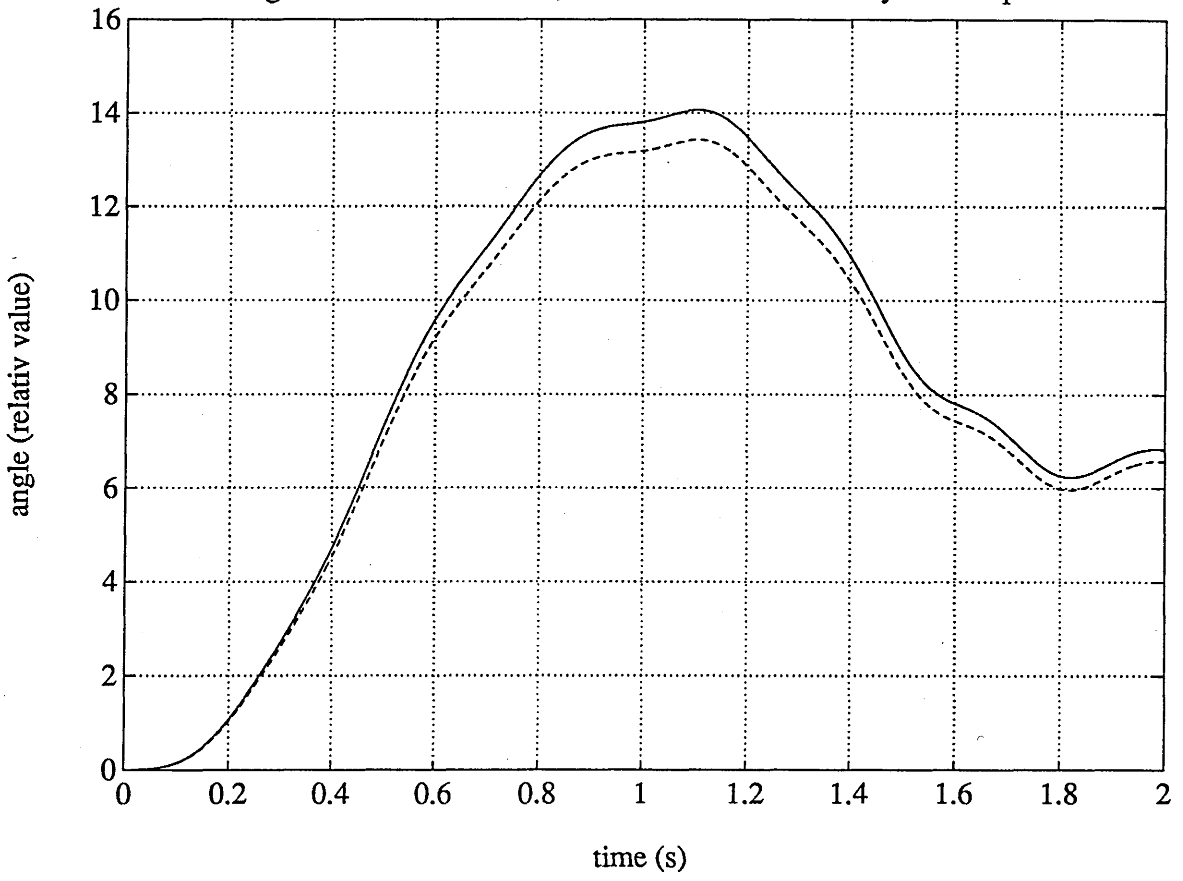


Fig6.4. Simulation and actual system outputs, Kp=100

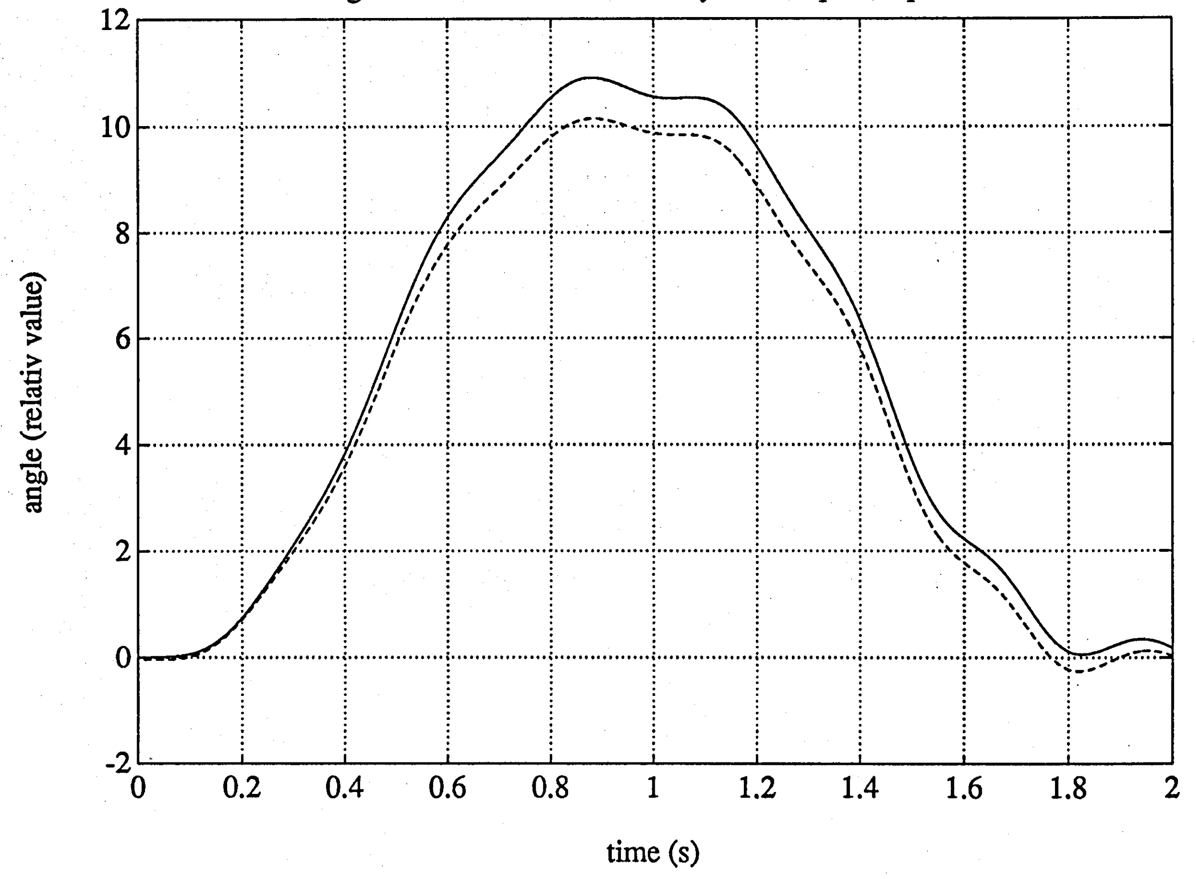


Fig6.5. Control signals, Kp=100

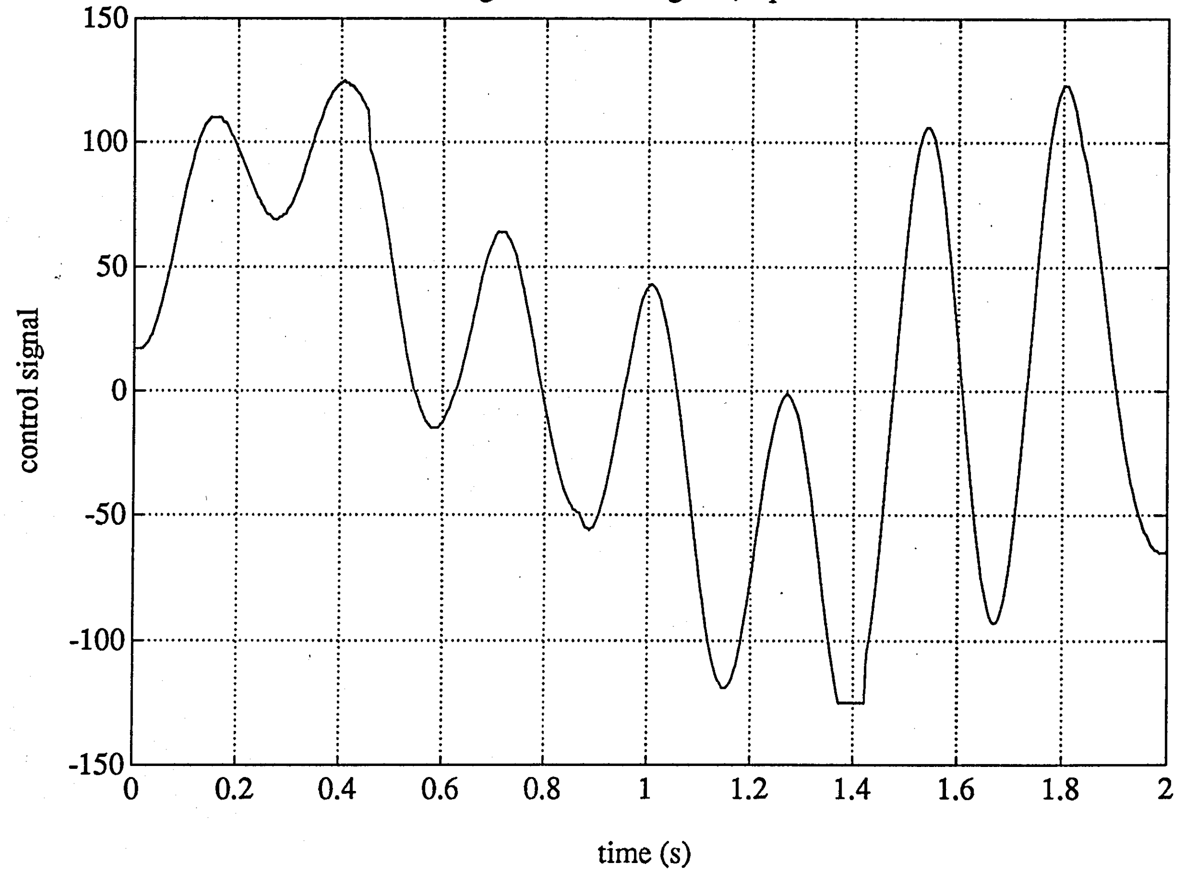


Fig6.6. Simulation and actual system outputs, Kp=200

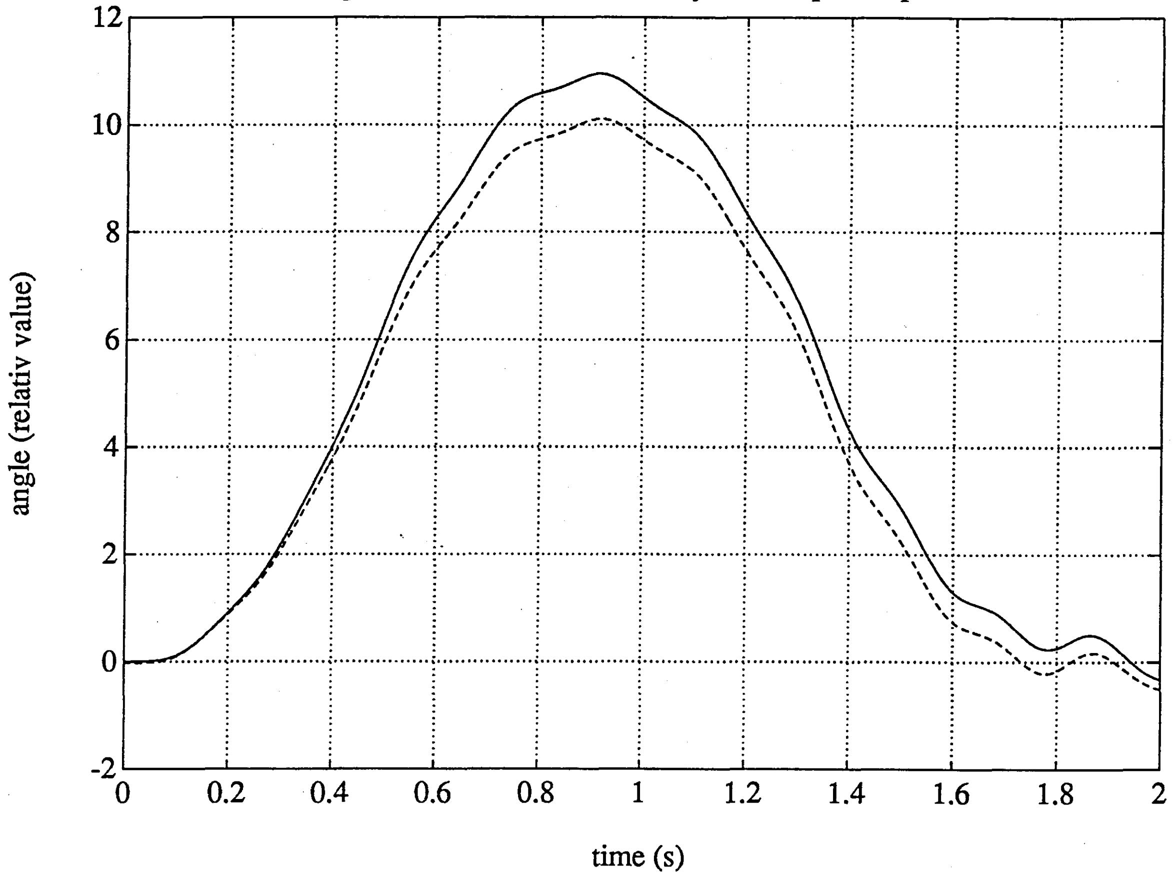


Fig6.7. Control signals, Kp=200

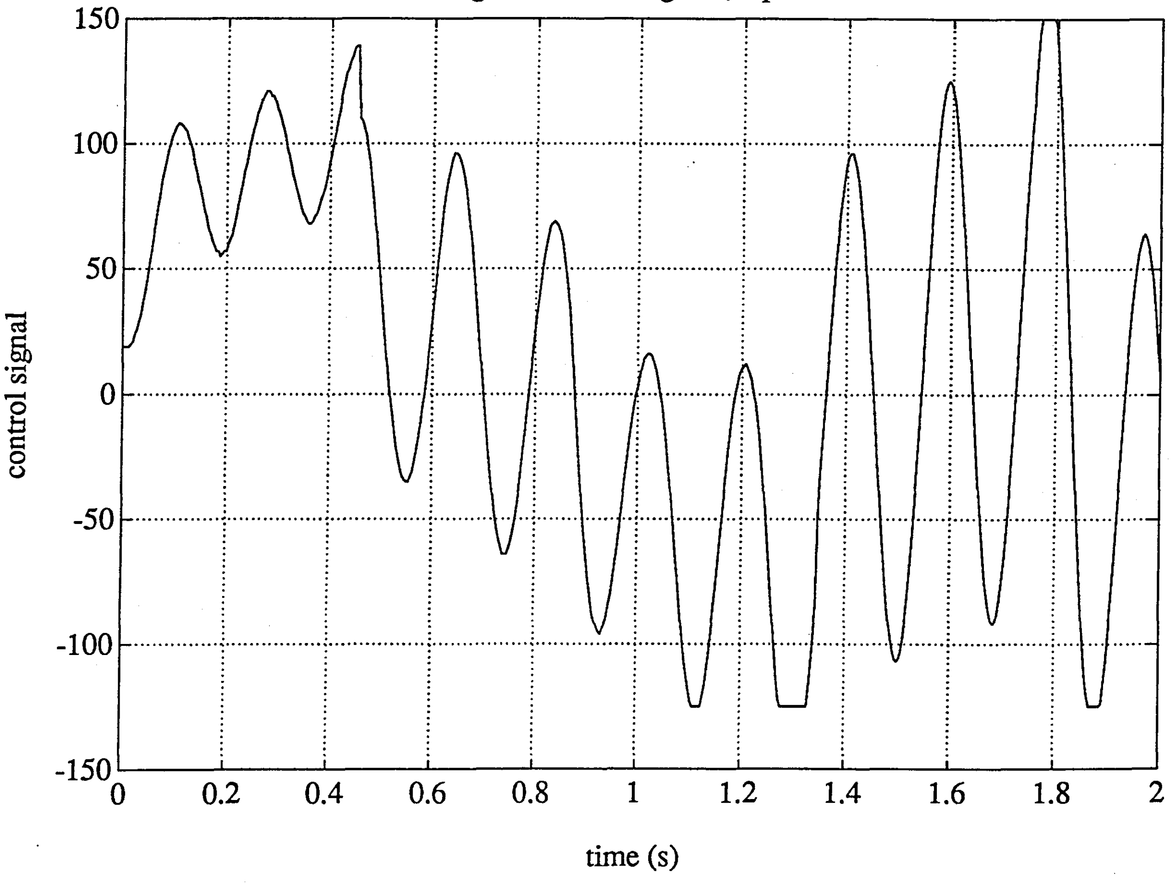


Fig6.8. Simulation and actual system outputs, Kp=300

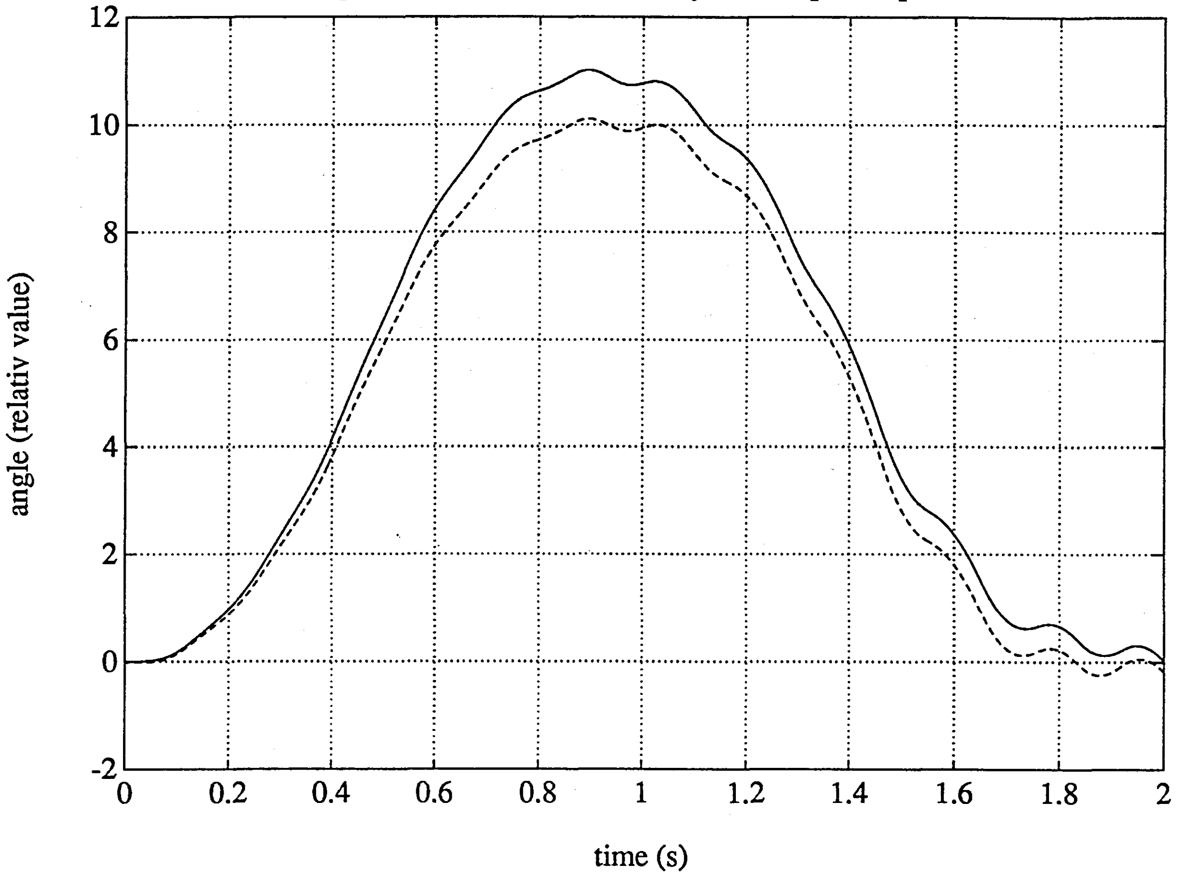
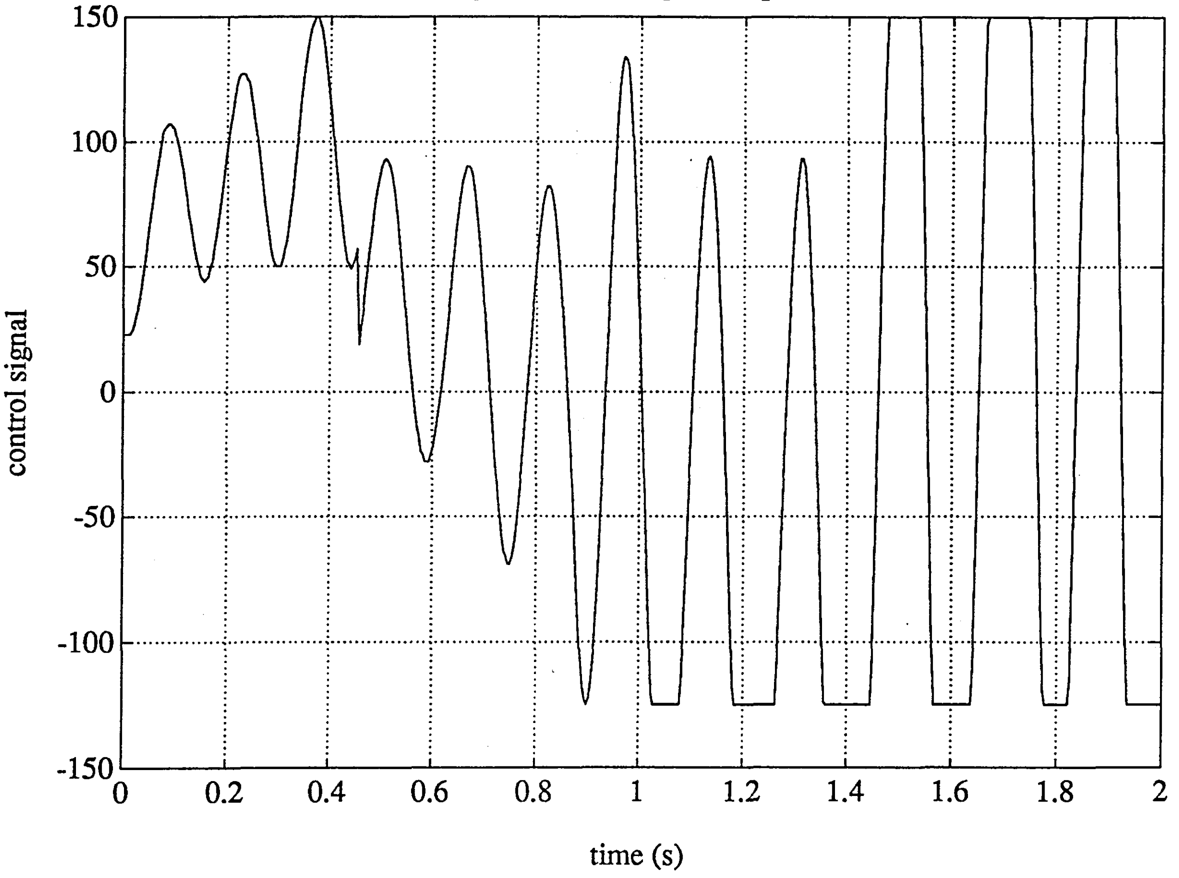


Fig6.9. Control signals, Kp=300



Chapter 7. Discussion and Conclusion

7. Simulation of Rigid Manipulators

The establishment of robot manipulator dynamic equations is the basis of robot validation. There are many methods to obtain the manipulator dynamics, the iterative Newton-Euler dynamic formulation and the Lagrange-Euler formulation are two of them. By applying these two methods, we write two programmes to generate the codes which can be read by both machine and man.

For today's manipulator, the environments require it having light component devices and fast speed responses. This causes the manipulator to have flexible links. The concise way to realise the characteristics of flexible manipulators is to study the case of one-link flexible manipulators at the beginning stage. This thesis has described four methods to set up the dynamic equations of a one hinged-free flexible link manipulator.

Robot validation is a recently developed research area. To realise the general dynamic characteristics of flexible manipulators and to verify the friction effects on DC motors are the research in this area.

The modelling and the validation of robot manipulators are developed quite smoothly in the thesis, but there still exists limitations.

In the first part of the thesis, we have created two programmes to set up the dynamic equations of rigid manipulators. We know that either the iterative Newton-Euler dynamic formulation or the Lagrange-Euler formulation suits the manipulators with both rotational and translational joints, but the programmes can only be applied for the manipulators with rotational joints. However, this limitation can easily be solved by adding some condition loops into the programmes. Moreover the robot manipulator MA3000 in our laboratory is the one with rotational joints. Another limitation is that it can not be directly for the programmes to generate the dynamic equations of a manipulator with a constant mass load at its end point. This problem can be prevented by adding a zero length link with the load mass to the end point of the manipulator. Because a manipulator with a constant mass load is an ordinary case during its working time, this should have been considered when writing the programmes. Lastly, because the whole modelling and the validation work is done by computer simulations, we are not quite certain how much it is coincident with the real cases.

In the second part, four analytic methods are developed to set up the model of a one-link flexible manipulator, and the static and dynamic friction effects on DC motors are verified. There are also three main limitations.

Firstly, the transfer functions which describe the model of one-link flexible manipulators are very complex, especially when we consider the real ones which have rigid base mass moment inertia, end point load mass and nonlinear mass distributions. This limitation can be overcome by substituting the transfer functions with the ones described by zeros and poles in S plane. The disadvantage of this substitution is that different system has its own zeros and poles.

Secondly, because of the limitation of the experimental devices, the applied beam in the frequency response experiment is not flexible at all (the beam's elastic constant coefficient is $EI = 842.285Nm^2$, its length is $L = 0.727m$). If time and experiment devices permit, we can sample data from other more flexible beams to test their frequency responses.

Thirdly, the robot validation work about flexible links and actuating systems in this thesis is only at its beginning stage. The contents of robot validation are rather widespread. Taking transmission system alone for example, we can consider many factors: the nonlinear area relationship between the electric voltage inputs and the mechanical rotational torque outputs for electric motors, the backlash between gears when the rotation direction changes, the elastic slipping problem in belt transmissions and so on. Therefore, it is not enough for a real system to consider link flexibilities and friction effects on motors. Anyway, the link flexibility is a major factor to be considered in the modelling of flexible manipulators.

Robot manipulators tend to have light and sensitive devices and components. This tendency will cause the manipulators' links to be flexible. In order to solve these problems, we should investigate the all-round factors of robot validation. The validation work in this thesis is partial, but essential to the establishment of dynamic models.

A proper control strategy for either rigid or flexible manipulators is not difficult to be found by computer simulation, if a suitable dynamic model has been set up. As described in chapter 1, there are some existed control laws for the manipulator control. Compute torque technique and PID controller are two of them. We have used these two control strategies for the control of the established dynamic models of one-link flexible manipulators. Because there is not any new ideas in this part of research, we have not included it in the thesis.

In summary, the manipulator dynamics generation programmes, the simulation methods, and the robot validation work are satisfied from simulation point of view. Computer simulation is efficient for the preliminary research on robot manipulators.

References

[1] Craig, John.J., " Introduction to Robotics Mechanical & Control, " Addison-Wesley Publishing Company, Inc. 1986

[2] Fu, K.S., Gonzalez, R.C., Lee, C.S.G., " Robotics: Control, Sensing, Vision, and Intelligence, " McGraw-Hill, Inc. 1987

[3] Nicosia, S., Tomei, P., Tornambe, A., " Dynamic Modelling of Flexible Robot Manipulators, " Rep. CH2282-2/86/0000/0365 1986

[4] Richard, P., Michael, R., John, F., " Computer Algebra, " Scientific American, Vol. 245, No. 6, December 1981, PP. 136-152

[5] Anthony, C.H., " Reduce User's Manual, "

[6] Book, W.J., " Modelling, Design and Control of Flexible Manipulator Arms, " Ph.D. thesis, Dept of Mechanical Engineering, Massachusetts Institute of Technology, 1974

[7] Book, W.J., Maizza-Neto, O., Whitney, D.E., " Feedback Control of Two Beam, Two Joint Systems with Distributed Flexibility, " ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 97, No. 4, December 1975

[8] Schmits, E., " Experimental on the End-Point Position Control of a Very Flexible One-Link Manipulator, " Ph.D. thesis, Stanford University, 1985

[9] Skaar, S.B., Tucker, D., " Point Control of a One-Link Flexible Manipulator, .IP

[10] Maizza-Neto, O., " Model Analysis and Control of Flexible Manipulator Arms, Technology, 1974

[11] Dubowsky, S., Grant, J.L., " Application of Symbolic Manipulation to Time Domain Analysis of Nonlinear Dynamic Systems, " ASME Journal of Dynamic

Systems, Measurement, and Control, March 1975, PP. 60-80

[12] Denavit, J., Hartenberg, R.S., " A kinematic Notation for Lower-Pair Mechanisms Based on Matrices, " ASME Journal of Applied Mechanics, June 1965, PP. 215-221

[13] Schiehlen, W.O., Kreuser, E.J., " Symbolic Computerized Derivation of Equations of Motion, " Proceedings of IUTAM Symposium, Munich, Germany, August 1977

[14] Liegois, A., Khalil, W., Dumas, J.M., Renand, M., " Mathematical and Computer Models of Interconnected Mechanical Systems, " Proceedings of 2nd International CISM-IFTOMM Symposium, Warsaw, September 1976, PP.5-17

[15] Leu, M.C., Hemati, N., " Automated Symbolic Derivation of Dynamic Equations of Motion for Robot Manipulators, " ASME Journal of Dynamic Systems, Measurement, and Control, September 1986, PP. 172-179

[16] Mitrani, I., " Simulation Techniques for Discrete System" Cambridge University Press, 1982

[17] Bratley, P., Fox, B.L., Schrabe, L.E., " A Guide to Simulation, " 2nd ed. Springer-Verlag New York Inc. 1987

[18] Balas, M.J., " Feedback Control of Flexible Systems, " IEEE Trans. Automatic Control, Vol. 23, PP. 673-679, 1978

[19] Karkkainen, P., Halme, A., " Model Space Control of Manipulator Vibrational Motion, " SYROCO '85, Barcellona, PP. 101-105, November 1985

[20] Book, W.J., " Analysis of Massless Elastic Chains with Rervo Controller Joints, " ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 101, 1979, PP. 187-192

[21] Judd, R.P., Falkenburg, D.R., " Dynamics of Nonrigid Articulated Robot Linkages, " IEEE Trans. Automatic Control, Vol. 30, PP. 499-502, 1985

[22] Sunada, W.H., Dubowsky, S., " On the Dynamic Analysis and Behaviour of Industrial Robotic Manipulator with Elastic Members, " Journal of Mechanical Design, Vol. 105, PP. 42-51, 1983

[23] Chassiakos, A.G., Bekey, G.A., " Pointwise Control of Flexible Manipulator Arm, " SYROCO '85, Barcellona, PP. 113-117, November 1985

[24] Chen, Y., " Vibration Theoretical Methods, " Addison-Wesley Publishing Company Inc., Reading, Massachusttes, U.S.A., 1966

[25] Thomson, W.T., " Theory of Vibration with Applications, " 3rd ed., Geoge Allen & Unwin Ltd., London, 1988

[26] Leonard, M., " Analytical Methods in Vibrations, " Collier-Maacmillon Ltd., 1967

[27] Gawthrop, P.J., " Metamodelling of Robots: A Feasibility Study, " Preliminary Report, Dept. of Mechanical Engineering, Glasgaw University, July 1988

[28] Warburton, G.B., " The Dynamical Behaviour of Structures, " 2nd ed., Pergamon Press Ltd., 1976

[29] ACSL User's Manual

[30] Gawthrop, P.J., " Automatic Tuning of a Motion Controller, " Preliminary Report, Dept. of Mechanical Engineering, Glasgaw University, August 1988

The Programmes for Calculating the Torques Acting on Rotational Manipulators

First, the Newton-Euler recursive equations programme is presented below:

```
% Put in the link's number of the manipulaor,
 PAUSE;
% Define the variables in matrix,
% MA(I,1) The mass of ith link,
% IL        The inertials of all links,
% THE(I,1)       The ith joint's variable position,
% D1(I,1)  The ith joint's variable velocity,
% D2(I,1)  The ith joint's variable acceleration,
% ALPHA(I,1)  The angle between Zi and Zi-1,
% DX(I,1) The shortest distance between Zi and Zi-1,
% DZ(I,1) The shortest distance between Xi and Xi-1,
% POS(I,3)       The ith link mass center position from the origin of,
%           the ith coordinate system respect to its system,
%ENDN   The moment load of the end link,
%ENDF          The force load of the end link,
%W0            The angular velocity of the base coordinate system,
%E0     The angular acceleration of the base coordinate system,
%A0     The linear acceleration of the base coordinate system,

MATRIX
    MA(N,1), IL(3*N,3), THE(1+N,1), D1(1+N,1), D2(1+N,1), ALPHA(1+N,1),
        DX(N,1), DZ(N,1), POS(N,3), ENDN(3,1), ENDF(3,1), W0(3,1), E0(3,1),
        A0(3,1);
     PAUSE;


   % The procedure of calculating the torques,


   PROCEDURE
        NewtonEuler(MA,IL,THE,D1,D2,ALPHA,DX,DZ,POS,ENDN,ENDF,W0,E0,A0);


   % e1 e2 e3 h1 h2 h3 u1 u2 u3 o1 o2 o3 are uesd for creating a cross product,
   % m is the matrix M(n,n) in the general torque equations,
   % q  is the vector QI(q, q̇ )in the general torque equations,


     BEGIN
        MATRIX          e1(3,3), e2(3,3), e3(3,3), h1(1,3), h2(1,3), h3(1,3),
            u1(1,1), u2(1,1), u3(1,1), z(3,1), m(N,N), q(N,1);


   % Define the variables in array form,
   % rr(i)    The transformation of link i to i-1,
   % r(i)          The transformation of link i-1 to i,
   % w(i)          The angular velocity of link ith coordinate system,
   % ww(i)    The cross product form of w(i),
   % e(i)     The angular acceleration of link ith coordinate system,
   % ee(i)    The cross product form of e(i),
   % a(i)     The linear acceleration of link ith coordinate system,
   % ac(i)    The linear acceleration of link ith mass center,
   % f(i)     Force exerted on linki by link i-1,
% rp rp1   Used as the function of cross product,
% v(i)     The moment exerted on link i by link i-1,
% t(i)     Input torque acting on joint i,
% i(i)     Used as the array form of IL,
% p(i)     Used as the array form of DX and DZ,
% d(i)     USsed as the array form of POS,
      ARRAY
           r(1+n), rr(1+n), w(1+n), ww(1+n), e(n+1), ee(n+1), a(n+1),
           ac(n+1), f(1+n), rp(1+n), rp1(1+n), v(1+n), t(n+1), i(n),
           p(n), d(n);

% Assignment of the basic data,

      w(0):=w0;
      e(0):=e0;
      a(0):=a0;
      e1:=mat((0,0,0),(0,0,-1),(0,1,0));
      e2:=mat((0,0,1),(0,0,0),(-1,0,0));
      e3:=mat((0,-1,0),(1,0,0),(0,0,0));
      h1:=mat((1,0,0)); h2:=mat((0,1,0)); h3:=mat((0,0,1));
      z:=mat((0),(0),(1));
      f(n+1):=endf; v(n+1):=endn;

% The loop for calculating transformations, w and e,
% Outward recursion,

      FOR j:=1 STEP 1 UNTIL n+1 DO
```

```
<<
rr(j):=mat
        (
         (cos(the(j,1)),
            -sin(the(j,1))*cos(alpha(j,1)),
               sin(the(j,1))*sin(alpha(j,1))),
         (sin(the(j,1)),
            cos(the(j,1))*cos(alpha(j,1)),
               -cos(the(j,1))*sin(alpha(j,1))),
         (0,
            sin(alpha(j,1)),
               cos(alpha(j,1)))
        ),
r(j):=tp(rr(j)),
w(j):=r(j)*(w(j-1)+z*d1(j,1)),
u1:=h1*w(j-1); o1:=u1(1,1),
u2:=h2*w(j-1); o2:=u2(1,1),
u3:=h3*w(j-1); o3:=u3(1,1),
ww(j):=o1*e1+o2*e2+o3*e3,
e(j):=r(j)*(e(j-1)+z*d2(j,1)+ww(j)*(z*d1(j,1))),
u1:=h1*e(j); o1:=u1(1,1),
u2:=h2*e(j); o2:=u2(1,1),
u3:=h3*e(j); o3:=u3(1,1),
ee(j):=o1*e1+o2*e2+o3*e3
>>;
```

```
% The loop for calculating the accelerations a and ac,
    FOR j:=1 STEP 1 UNTIL n DO
      <<
      p(j):=r(j)*mat(
                    (cos(the(j,1))*dx(j,1)),
                     (sin(the(j,1))*dx(j,1)),
                      (dz(j,1))),
      d(j):=mat((pos(j,1)),(pos(j,2)),(pos(j,3))),
      a(j):=ee(j)*p(j)+ww(j+1)*(ww(j+1)*p(j))+r(j)*a(j-1),
      ac(j):=ee(j)*d(j)+ww(j+1)*(ww(j+1)*d(j))+a(j)
      >>;

% The loop for calculating the torque t,
% Inward recursion,

    FOR j:=n STEP -1 UNTIL 1 DO
      <<
      f(j):=rr(j+1)*f(j+1)+ma(j,1)*ac(j),
      u1:=h1*(p(j)+d(j)); o1:=u1(1,1);
      u2:=h2*(p(j)+d(j)); o2:=u2(1,1);
      u3:=h3*(p(j)+d(j)); o3:=u3(1,1);
      rp(j):=o1*e1+o2*e2+o3*e3,
      u1:=h1*(r(j+1)*p(j)); o1:=u1(1,1);
      u2:=h2*(r(j+1)*p(j)); o2:=u2(1,1);
      u3:=h3*(r(j+1)*p(j)); o3:=u3(1,1);
      rp1(j):=o1*e1+o2*e2+o3*e3;
      i(j):=mat
           (
                (il(j*3-2,1),il(j*3-2,2),il(j*3-2,3)),
                (il(j*3-1,1),il(j*3-1,2),il(j*3-1,3)),
                (il(j*3,1),il(j*3,2),il(j*3,3))
               ),
          v(j):=rr(j+1)*(v(j+1)+rp1(j)*f(j+1))+rp(j)*(ma(j,1)*ac(j))+
          i(j)*e(j)+ww(j+1)*(i(j)*w(j)),
          t(j):=tp(v(j))*(r(j)*z),
          u1:=t(j);o4:=u1(1,1);

     % The loop for calculating M and Q,

          FOR k:=1 STEP 1 UNTIL n DO
            <<
            m(j,k):=df(o4,D2(k,1))
            >>;
         q(j,1):=FOR k:=1 STEP 1 UNTIL n SUM o4/n-m(j,k);
         >>;
  END;
 END;
```

Second, the programme of applying Lagrange-Euler formulation is presented below:

```
%Number of links entered
%For example N:=3;
N:=3;
```

```
%Or this can be done before running the programme
MATRIX
      Masses(N,1),Il(3*N,3),Thetas(N,1),Dthetas(N,1),Alpha(N,1),
      Disalongxes(N,1),Disalongzs(N,1),Masscenters(N,3),
      Frameacceleration(3,1),B(N,N),C(N,N),D(N,1);

% N              Number of links
% Masses N X 1 vector masses of links
% Il      Links' inertials
% Thetas  Joints' variable positions
% DthetasJoints' variable velocities
% Alpha   Angles between Zs
% Disalongxes   Shortest distances between Zs
% Disalongzs    Shortest distances between Xes
% Masscenters   Centers of masses under their own coordinates
%
% Wait for entering basic parameters
% Type "N" then "return"
% In a file including the above parameters
% Type "cont;"
PAUSE;
% Unit and zero vectors

MATRIX Xunit,Yunit,Zunit,unit,Zero;

Xunit:=MAT((1),(0),(0));
Yunit:=MAT((0),(1),(0));
```

```
Zunit:=MAT((0),(0),(1));
Unit :=MAT((1,0,0),(0,1,0),(0,0,1));
Zero :=MAT((0),(0),(0));

PROCEDURE Vectorcrossproduct(X,Y);

%INPUT      X,Y
%OUTPUT     VCP

BEGIN
  XX:=TP(Xunit)*X; XY:=TP(Yunit)*X; XZ:=TP(Zunit)*X;
  YX:=TP(Xunit)*Y; YY:=TP(Yunit)*Y; YZ:=TP(Zunit)*Y;
  VCP:=MAT(
        (XY(1,1)*YZ(1,1)-YY(1,1)*XZ(1,1)),
        (XZ(1,1)*YX(1,1)-YZ(1,1)*XX(1,1)),
        (XX(1,1)*YY(1,1)-YX(1,1)*XY(1,1))
        );
END;

PROCEDURE Transfermation(Theta,Alpha);

%Input          Theta,Alpha
%Output   R

BEGIN
  R:=MAT(
      (   COS(Theta)      ,     SIN(Theta)      ,    0    ),
        (-SIN(Theta)*COS(Alpha), COS(Theta)*COS(Alpha) , SIN(Alpha)),
        (SIN(Theta)*SIN(Alpha) , -COS(Theta)*SIN(Alpha), COS(Alpha))
       );
END;

%       The Main Programme of Lagrange Method for the Dynamic Equations
%                   of Rotational Joints Manipulators

  MATRIX
        R1,Omiga,Omiga1,Linkposition,Linkposition1,Masscenter,
        Linkvelocity,Masscentervelocity,Inertial,Middlemax,
        Middlem1;

  R1:=unit;                   %Base coordinate system
  Omiga:=Zero;          %Base angular speed
  Linkposition1:=Zero;%Base origin
  Linkvelocity:=Zero;         %Base origin's speed
  Kineticenergy:=0;
  Potentialenergy:=0;

  FOR J:=1 STEP 1 UNTIL N DO
  BEGIN
     Transfermation(Thetas(J,1),Alpha(J,1));
     R1:=R1*TP(R);            %R1  Transfform to base
     Omiga:=R*(Omiga+Zunit*Dthetas(J,1));
                         %Omiga    Rotation speed of link
     Omiga1:=R1*Omiga;   %Omiga1   Compare to base
```

```
    Linkposition:=R1*R*MAT(
                        (COS(Thetas(J,1))*Disalongxes(J,1)),
                        (SIN(Thetas(J,1))*Disalongxes(J,1)),
                        (        Disalongzs(J,1)         )
                       );
                     %Linkposition
                        %     Link length vector compare to base
  Linkposition1:=Linkposition1+Linkposition;
                     %Linkposition1
                     %     Compare to base
    Masscenter:=R1*MAT(
                  (Masscenters(J,1)),
                  (Masscenters(J,2)),
                  (Masscenters(J,3))
                 );
                     %Masscenter
                     %     Compare to base
    Vectorcrossproduct(Omiga1,Linkposition);
    Linkvelocity:=Linkvelocity+VCP;
                     %Linkvelocity
    Vectorcrossproduct(Omiga1,Masscenter);
    Masscentervelocity:=Linkvelocity+VCP;
                        %Masscentervelocity
    Velocitysquare:=Masscentervelocity(1,1)**2+
                Masscentervelocity(2,1)**2+
                Masscentervelocity(3,1)**2;
                     %Velocitysquare
```

```
     Inertial:=MAT(
        (IL(3*J-2,1),IL(3*J-2,2),IL(3*J-2,3)),
        (IL(3*J-1,1),IL(3*J-1,2),IL(3*J-1,3)),
        (IL(3*J,1),IL(3*J,2),IL(3*J,3)));
                        %Inertial
     Middlem1:=TP(Omiga)*Inertial*Omiga;
                        %Middlemax
     InertialOmigasquare:=Middlem1(1,1);
                        %InertialOmigasquare
     Kineticenergy:=Kineticenergy+Masses(J,1)*Velocitysquare/2+
                              InertialOmigasquare/2;
                        %Kineticenergy
     Middlemax:=Masses(J,1)*TP(Frameacceleration)*
              (Linkposition1+Masscenter);
     Potentialenergy:=Potentialenergy+Middlemax(1,1);
                        %Potentialenergy
  END;

  FOR J:=1 STEP 1 UNTIL N DO
  BEGIN
     FOR K:=1 STEP 1 UNTIL N DO
     BEGIN
        D(J,K):=DF(DF(Kineticenergy,Dthetas(J,1)),Dthetas(K,1));
        H(J,K):=DF(DF(Kineticenergy,Dthetas(J,1)),Thetas(K,1));
     END;
     C(J,1):=DF(Kineticenergy-Potentialenergy,Thetas(J,1));
  END;
```

```
END; ; END;
```

The Programmes for the Simulation of a Three Rigid Link Manipulator

The first one is the programme for obtaining θ, $\dot{\theta}$, $\ddot{\theta}$.

```
CONTINUOUS SYSTEM ThreeLinkSys
"
"THIS IS THE SYSTEM FOR SOLVING THE EQUATION (3-1).
"INPUT THE TERMS OF M(θ)^-1, AND Q(θ,θ̇),
"MMij IS THE TERM OF iTH ROW jTH COLUMN OF M(θ)^-1,
"Qi IS THE iTH TERM OF Q(θ,θ̇).
input MM11 MM12 MM13 MM21 MM22 MM23 MM31 MM32 MM33 Q1 Q2 Q3
"DEFINE STATES ADN DERIVATIVES.
state theta1 theta2 theta3 thetadot1 thetadot2 thetadot3
der dtheta1 dtheta2 dtheta3 dthetadot1 dthetadot2 dthetadot3
"DEFINE THE SYSTEM'S TIME t.
time t
"THE EQUATION OF (3-1).
dtheta1 = thetadot1
dtheta2 = thetadot2
dtheta3 = thetadot3
dthetadot1 = MM11*(torque1-Q1)+MM12*(torque2-Q2)+MM13*(torque3-Q3)
dthetadot2 = MM21*(torque1-Q1)+MM22*(torque2-Q2)+MM23*(torque3-Q3)
dthetadot3 = MM31*(torque1-Q1)+MM32*(torque2-Q2)+MM33*(torque3-Q3)
"GIVE THE VALUES OF ALL CONSTANTS.
t1:0
```

```
t2:0

t3:0

end

"END OF THE PROGRAMME.
```

The second programme is for obtaining the values of $M(\theta)^{-1}$, AND $Q(\theta,\dot{\theta})$.

```
CONTINUOUS SYSTEM MQValue
"
"THIS SYSTEM CALCULATES THE TERMS OF M(θ)^-1, AND Q(θ,θ̇).
"INPUT THE ANGLES AND THEIR FIRST DERIVATIVES.
input theta1 theta2 theta3 thetadot1 thetadot2 thetadot3
"OUTPUT THE TERMS OF M(θ)^-1, AND Q(θ,θ̇).
output MM11 MM12 MM13 MM21 MM22 MM23 MM31 MM32 MM33 Q1 Q2 Q3
"WHERE MMij IS THE TERM OF iTH ROW jTH COLUMN OF M(θ)^-1.
"Qi IS iTH TERM OF Q(θ,θ̇).
"DEFINE THE SYSTEM'S TIME t.
time t
"THE ASSIGNMENTS BELOW ARE THE TERMS OF M(θ) AND Q(θ,θ̇).
M111= cos(2*(theta2+theta3))*(m3*l3*l3-4*I3X+4*I3Y)
M112 = cos(2*theta2)*(m2*l2*l2+4*m3*l2*l2-4*I2X+4*I2Y)
M113 = 4*cos(theta3)*m3*l2*l3+4*cos(2*theta2+theta3)*m3*l2*l3+m2*l2*l2
M114 = 4*m3*l2*l2+m3*l3*l3+8*I1Y+4*I2X+4*I2Y+4*I3X+4*I3Y
M11 = (M111+M112+M113+M114)/8
M12 = 0
M13 = 0
M21 = 0
M22 = (cos(theta3)*l3+l2)*m3*l2+(m2*l2*l2+m3*l3*l3)/4+I2Z+I3Z
M23 = (2*cos(theta3)*m3*l2*l3+m3*l3*l3+4*I3Z)/4
M31 = 0
M32 = (2*cos(theta3)*m3*l2*l3+m3*l3*l3+4*I3Z)/4
M33 = (m3*l3*l3+4*I3Z)/4
```

```
DETM = M11*(M22*M33-M23*M32)+M21*(M32*M13-
M12*M33)+M31*(M12*M23-M22*M13)
MM11 = (M22*M33-M23*M32)/DETM
MM12 = (M32*M13-M12*M33)/DETM
MM13 = (M12*M23-M22*M13)/DETM
MM21 = (M23*M31-M21*M33)/DETM
MM22 = (M11*M33-M31*M13)/DETM
MM23 = (M21*M13-M11*M23)/DETM
MM31 = (M21*M32-M22*M31)/DETM
MM32 = (M12*M31-M11*M32)/DETM
MM33 = (M11*M22-M21*M12)/DETM
Q11 = -sin(2*(theta2+theta3))*thetadot1*m3*l3*l3*(thetadot2+thetadot3)
Q12 = sin(2*(theta2+theta3))*thetadot1*4*(I3X-I3Y)*(thetadot2+thetadot3)
Q13 = 2*sin(2*theta2+theta3)*m3*thetadot1*l2*l3*(-2*thetadot2-thetadot3)
Q14 = sin(2*theta2)*thetadot1*thetadot2*(-l2*l2*(m2+4*m3)+4*I2X-4*I2Y)
Q15 = -2*sin(theta3)*m3*thetadot1*thetadot3*l2*l3
Q1 = (Q11+Q12+Q13+Q14+Q15)/4
Q21 = sin(2*(theta2+theta3))*thetadot1*thetadot1*(m3*l3*l3-4*I3X+4*I3Y)
Q22 = 4*sin(2*theta2+theta3)*m3*thetadot1*thetadot1*l2*l3
Q23 = sin(2*theta2)*thetadot1*thetadot1*((m2+4*m3)*l2*l2-4*I2X+4*I2Y)
Q24 = 4*sin(theta3)*m3*thetadot3*l2*l3*(-2*thetadot2-thetadot3)
Q25 = 4*cos(theta2)*G*l2*(m2+2*m3)+4*cos(theta2+theta3)*G*m3*l3
Q2 = (Q21+Q22+Q23+Q24+Q25)/8
Q31 = sin(2*(theta2+theta3))*thetadot1*thetadot1*m3*l3*l3-4*I3X+4*I3Y)
Q32 =
m3*l3*(2*sin(2*theta2+theta3)*thetadot1*thetadot1*l2+4*cos(theta2+theta3)*G)
Q33=2*sin(theta3)*M3*l2*l3*(thetadot1*thetadot1+2*thetadot2*thetadot2)
```

```
Q3 = (Q31+Q32+Q33)/8

"GIVE THE VALUES OF ALL CONSTANTS.

l1:0.3

l2:0.5

l3:0.4

m1:25

m2:5

m3:10

G:9.801

I1X:0.1875

I1Y:0

I1Z:0.1875

I2X:0

I2Y:0.104

I2Z:0.104

I3X:0

I3Y:0.133

I3Z:0.133

end

"END OF THE PROGRAMME.
```

Third, the programme for connecting the two programmes above is listed below:

```
CONNECTING SYSTEM ConnectingSys
"
"THIS IS THE CONNECTING PROGRAMME FOR CONNECTING THE PRO-
GRAMS OF MQValue AND ThreeLinkSys.
"DEFINE THE SYSTEM'S TIME t.
time t
"THE CONNECTING ASSIGNMENTS.
q1[ThreeLinkSys] = q1[MQValue]
q2[ThreeLinkSys] = q2[MQValue]
q3[ThreeLinkSys] = q3[MQValue]
theta1[MQValue] = theta1[ThreeLinkSys]
theta2[MQValue] = theta2[ThreeLinkSys]
theta3[MQValue] = theta3[ThreeLinkSys]
thetadot1[MQValue] = thetadot1[ThreeLinkSys]
thetadot2[MQValue] = thetadot2[ThreeLinkSys]
thetadot3[MQValue] = thetadot3[ThreeLinkSys]
MM11[ThreeLinkSys] = MM11[MQValue]
MM12[ThreeLinkSys] = MM12[MQValue]
MM13[ThreeLinkSys] = MM13[MQValue]
MM21[ThreeLinkSys] = MM21[MQValue]
MM22[ThreeLinkSys] = MM22[MQValue]
MM23[ThreeLinkSys] = MM23[MQValue]
MM31[ThreeLinkSys] = MM31[MQValue]
MM32[ThreeLinkSys] = MM32[MQValue]
MM33[ThreeLinkSys] = MM33[MQValue]
```

```
end
"END OF THE PROGRAMME.
```

The Computer Codes for Chapter 4

The REDUCE codes for obtaining the exact model described by the transfer functions of one flexible link manipulators.

```
%Simplify equations
FOR ALL X,Y LET
sin(X)*cos(Y) = (sin(X+Y)+sin(X-Y))/2,
sin(X)*sin(Y) = (cos(X-Y)-cos(X+Y))/2,
cos(X)*cos(Y) = (cos(X-Y)+cos(X+Y))/2;
FOR ALL X LET
sin(X)**2 = (1-cos(2*X))/2,
cos(X)**2 = (1+cos(2*X))/2,
sin(X)*cos(X) = sin(2*X)/2,
sinh(X)**2 = (cosh(2*X)-1)/2,
cosh(X)**2 = (cosh(2*X)+1)/2,
cosh(X)*sinh(X) = sinh(2*X)/2;
%Define matrix. A is the coefficients of A,B,C,and D. K1 is the substitute of
A,B,C,D.
%U1 is the other part of equations (4-4) - (4-7). K2 used for saving the result of
A,B,C, and D.
MATRIX A(4,4),K1(4,1),U1(4,1),K2(4,1);
%Let K1 have symbolic values.
K1 := MAT((A1),(A2),(A3),(A4));
%Define array.
```

```
ARRAY YY(4);
%Assignment of equation (4-14).
Y := EXP(B*X) * ( K1(1,1)*cos(B*X)+K1(2,1)*sin(B*X) ) + EXP(-B*X) * (
K1(3,1)* cos(B*X)+K1(4,1)*sin(B*X) ) - theta*X;
Integration1 := INT(X*Y,X);
Y1 := DF(Y,X);
Y2 := DF(Y1,X);
Y3 := DF(Y2,X);
LET X = 0;
YY(1) := Y;
YY(2) := Y1;
ValueIntegration10 := Integration1;
LET X = L;
YY(3) := Y2;
ValueIntegration1L := Integration1;
YY(4) := M*S**2*Y + M*L*S**2*theta - EI*Y3;
%Torque is the input torque's transform.
Torque := (I0+P*L**3/3+M*L**2) *S**2*theta + P*S**2* (ValueIntegration1L-
ValueIntegration10) + M*L*S**2*Y;
FOR J := 1 STEP 1 UNTIL 4 DO
  <<FOR K := 1 STEP 1 UNTIL 4 DO
    <<A(J,K) := DF(YY(J),K1(K,1)); YY(J) := YY(J) - A(J,K)*K1(K,1)>>;
  U1(J,1) := -YY(J)>>;
K2 := 1/A*U1;
A1 := K2(1,1);
A2 := K2(2,1);
A3 := K2(3,1);
```

```
A4 := K2(4,1);
```

%G is the transfer function of the tip-point's angle versus input torque. B is the β in equation (4-16).

```
G := (Y/L+theta)/Torque;
G1 := theta/Torque;
;
END;
```

%End of the main codes.

The transfer functions of G, and $G1$ of the one flexible link manipulator with rigid bass mass moment of inertia and end-point mass is listed below:

```
G = (8*E**(B*L)*B**5*EI* (2*E**(6*B*L)*cos(B*L)*B**3*EI + E**(6*B*L)*
cos(B*L) * M*S**2 + 2*E**(6*B*L)*sin(B*L)*B**3*EI
+E**(6*B*L)*sin(B*L)*M*S**2 + 8*E**(4*B*L)*cos(B*L)*B**3*EI -
2*E**(4*B*L)*cos(B*L)*M*S**2 + 2*E**(4*B*L)* cos(3*B*L)*B**3*EI +
E**(4*B*L)*cos(3*B*L) * M*S**2 + 8*E**(4*B*L) * sin(B*L)*B**3* EI +
2*E**(4*B*L)*sin(3*B*L) * B**3*EI - E**(4*B*L)*sin(3*B*L) * M*S**2 -
8*E**(2*B*L) * cos(B*L)*B**3*EI - 2*E**(2*B*L) * cos(B*L)*M*S**2-
2*E**(2*B*L)* cos(3*B*L)*B**3*EI + E**(2*B*L) * cos(3*B*L)*M*S**2 +
8*E**(2*B*L)* sin(B*L)*B**3*EI + 2*E**(2*B*L) * sin(3*B*L)*B**3*EI +
E**(2*B*L) * sin(3*B*L)* M*S**2 - 2*cos(B*L)*B**3*EI + cos(B*L)*M*S**2 +
2*sin(B*L)*B**3*EI - sin(B*L)*M*S**2)) / (L*S **2*
(8*E**(8*B*L)*B**9*I0*EI**2 + 8*E**(8*B*L) * B**6*M*S**2*I0*EI +
4*E**(8*B*L) * B**6*P*EI**2 + 2*E**(8*B*L) * B**3*M**2*S**4*I0 +
4*E**(8*B*L) * B**3*M*P*S**2*EI + E**(8*B*L)*M**2 * P*S**4 + 16
*E**(7*B*L)* cos(B*L)* B**8*L*M*EI**2 + 8*E**(7*B*L)* cos(B*L)*B **5*L*
M**2*S**2*EI - 4*E**(7*B*L)* cos(B*L)*B**4*L*M*P* S**2*EI -
2*E**(7*B*L)* cos(B*L)*B*L*M**2* P*S**4 + 16*E**(7*B*L)* sin(B*
L)*B**8*L*M*EI**2 + 8*E**(7*B*L)* sin(B*L)*B**5*L* M**2*S**2*EI -
4*E**(7*B*L)*sin(B*L)*B**4*L*M*P* S**2*EI - 2*E**(7*B*L)* sin( B*L)*B*L*
M**2*P*S**4 + 32*E**(6*B*L)* cos(2*B*L)*B**9*I0* EI**2 + 16* E**(6*B*L)*
cos(2*B*L)*B**6*M* S**2*I0*EI + 8* E**(6*B*L)*cos(2*B*L)*B**6*P*EI**2 -
2*E**(6*B*L)* cos(2*B*L)*M**2*P* S**4- 16*E**(6*B*L)* sin(2*B*L)*B**6*M*
S**2*I0*EI - 8* E**(6*B*L)* sin (2*B*L)*B**6* P*EI**2 - 8*E**(6*B*L)*
sin(2*B*L)*B**3* M**2*S** 4*I0 - 8* E**(6*B*L)* sin(2*B*L)*B**3*M*
```

P*S**2*EI - 2*E** (6*B*L)* sin(2*B*L)*M**2* P*S**4 + 64*E**(6*B*L)*
B**9*I0*EI**2 + 32*E** (6*B*L)*B**6*M* S**2*I0*EI + 16*E**(6*B*L)*
B**6*P*EI**2 + 8*E** (6*B*L)* B**3*M* P*S**2*EI + 64*E** (5*B*L)*
cos(B*L)*B**8*L* M*EI **2 - 16* E**(5*B*L)*cos(B*L)* B**5*L*M**2*S**2*EI
- 16*E**(5*B*L)* cos(B*L)* B**4*L*M* P*S**2*EI + 4*E**(5*B*L)*
cos(B*L)*B*L* M**2*P*S**4 + 16*E**(5*B*L)* cos(3*B*L)* B**8*L*M*EI**2 +
8*E**(5*B*L)* cos(3*B*L)*B**5*L* M**2*S**2*EI - 4*E**(5*B*L)*
cos(3*B*L)*B**4*L*M* P*S**2*EI - 2*E**(5*B*L)* cos(3*B*L)*B*L* M**2*
P*S**4+64*E**(5*B*L)* sin(B*L)*B**8*L* M*EI**2 - 16*E**(5*B*L)* sin(B*L)*
B**4*L*M*P* S**2*EI + 16*E**(5*B*L)* sin(3*B*L)*B**8*L* M*EI**2-
8*E**(5*B*L)* sin(3*B*L)* B**5*L*M**2* S**2*EI - 4*E**(5*B*L)*
sin(3*B*L)*B**4*L*M*P* S**2*EI + 2*E**(5*B*L)* sin(3*B*L)*B*L*
M**2*P*S**4 + 16*E**(4*B*L)* cos(4*B*L)*B**9*I0* EI**2 - 4*E**(4*B*L)*
cos(4*B*L)* B**3*M**2*S**4*I0 - 8*E**(4*B*L)* cos(4*B*L)*B**
3*M*P*S**2*EI + 128*E**(4*B*L)* cos(2*B*L)*B**9* I0*EI**2 - 16*
E**(4*B*L)*cos(2*B*L)*B**3*M* P*S**2*EI - 16*E**(4*B*L)* sin(4*B *L)*
B**6*M* S**2*I0*EI - 8*E**(4*B*L)* sin(4*B*L)*B**6* P*EI**2+
2*E**(4*B*L)* sin(4*B*L)* M**2*P*S**4 - 64*E**(4*B*L)* sin(2*B*L)
*B**6*M* S**2*I0*EI - 32* E**(4*B*L)* sin(2*B*L)* B**6*P*EI**2 + 160*
E**(4*B*L)* B**9*I0*EI**2 - 64* E**(3*B*L)* cos(B*L)* B**8*L*M *EI**2 -
16* E**(3*B*L)* cos(B*L)*B**5* L*M**2* S**2*EI + 16* E**(3 *B*L)*
cos(B*L)* B**4*L*M* P*S**2*EI + 4* E**(3*B*L)* cos(B*L)* B*L* M**2*P*
S**4 - 16*E**(3*B*L) *cos(3*B*L) *B**8*L* M*EI**2 + 8*E**(3*B*L)*
cos(3*B*L)* B**5*L*M**2* S**2*EI + 4*E**(3*B*L)* cos(3*B*L)*B**4*
L*M*P* S**2*EI - 2*E**(3*B*L)* cos(3*B*L)* B*L*M**2* P*S**4+
64*E**(3*B*L)* sin(B*L)* B**8*L* M*EI**2 - 16*E**(3*B*L)* sin(B*L)*B**4*
L*M*P* S**2*EI + 16* E**(3*B*L)* sin(3*B*L)*B**8* L*M*EI**2 + 8*

```
E**(3*B*L)* sin(3*B*L)* B**5*L* M**2* S**2*EI - 4*E**(3*B*L)* sin(3*B*L)*
B**4*L*M*P* S**2*EI - 2*E**(3*B*L)* sin(3*B*L)* B*L*M**2* P*S**4 + 32*
E**(2*B*L)* cos(2*B*L)* B**9* I0*EI**2 - 16* E**(2*B*L)* cos(2*B*L)*B**6*
M*S**2* I0*EI - 8* E**(2*B*L)* cos(2*B*L)* B**6*P*EI**2 + 2*E**(2*B*L)*
cos(2*B*L)* M**2*P*S**4 - 16* E**(2*B*L)* sin(2*B*L)* B**6*M* S**2*I0*EI -
8*E**(2*B*L)* sin(2*B*L)* B**6*P*EI**2 + 8*E**(2*B*L)* sin(2*B*L)*B**3*
M**2*S**4*I0 + 8*E**(2*B*L)* sin(2*B*L)*B**3*M*P* S**2*EI - 2*
E**(2*B*L)* sin(2*B*L)* M**2*P*S**4 + 64* E**(2*B*L)* B**9*I0*EI**2 - 32*
E**(2*B*L)*B**6*M* S**2*I0*EI - 16* E**(2*B*L)*B**6* P*EI**2 + 8*
E**(2*B*L)* B** 3*M* P*S**2*EI - 16*E**(B*L)* cos(B*L)* B**8* L*M*EI**2 +
8*E**(B*L)* cos(B*L)* B**5*L*M**2* S**2*EI + 4* E**(B*L)* cos(B*L)*
B**4*L*M* P*S**2*EI - 2* E**(B*L)* cos(B*L)*B* L*M**2*P*S**4 +
16*E**(B*L)* sin(B*L)*B**8*L* M*EI**2 - 8*E**(B*L)* s in(B*L)*
B**5*L*M**2* S**2*EI - 4* E**(B*L)* sin(B*L)* B**4*L*M*P* S**2*EI + 2*
E**(B*L)* sin(B*L)* B*L*M**2* P*S**4 + 8*B**9* I0*EI**2 - 8*B**6*
M*S**2*I0*EI - 4*B**6* P*EI**2 + 2*B**3*M**2* S**4*I0 + 4*B**3*M*
P*S**2*EI- M**2* P*S**4));
G1 = (2*B**3*(2* E**(4*B*L)* B**3*EI + E**(4*B* L)*M*S**2 +
4*E**(2*B*L)* cos(2*B*L)* B**3*EI - 2*E**(2*B*L)* sin (2*B*L)*M*S**2 +
8*E**(2*B*L)*B**3*EI + 2*B**3*EI - M*S**2)) / (S**2*(4*E**(4*B*L)*
B**6*I0*EI + 2*E**(4*B*L)* B**3*M* S**2*I0 + 2*E**(4*B*L)* B**3*P*EI +
E**(4*B*L)* M*P*S**2 + 8* E**(3*B*L) * cos(B*L)* B**5*L*M*EI -
2*E**(3*B*L)* cos(B*L)*B*L* M*P*S**2 + 8 * E**(3*B*L)* sin(B*L)*
B**5*L*M*EI - 2*E**(3*B*L)* sin(B*L)* B*L*M* P*S**2 + 8*E**(2*B*L)*
cos(2*B*L)* B**6*I0*EI - 2*E**(2*B*L)*cos (2*B*L)*M*P*S**2 - 4*E**(2*B*L)*
sin(2*B*L)*B**3* M*S**2*I0 - 4*E**(2*B*L)* sin(2*B*L)* B**3*P*EI +
16*E**(2*B*L)* B**6*I0*EI - 8*E**(B*L)* cos(B*L)* B**5*L*M*EI +
2*E**(B*L)* cos(B*L)* B*L*M*P *S**2 + 8*E**(B*L)* sin(B*L)* B**5*L*M*EI -
2*E**(B*L)* sin(B*L)* B*L*M*P*S**2 + 4*B**6*I0*EI - 2*B**3*M* S**2*I0 -
2*B**3* P*EI + M*P* S**2));
```

The equations above is very complecated. If we let the rigid bass mass of inertia and the end-point mass be zeros, the equations are:

```
M := 0;
I0 := 0;

G = (4*E**(B*L)*B**2*( E**(6*B*L)* cos (B*L) + E**(6*B*L)*sin(B*L) +
4*E**(4*B*L)*cos(B*L) + E**(4 * B*L)*cos(3*B*L) + 4*E**(4*B*L)*sin(B*L) +
E**(4*B*L)*sin(3*B* L ) - 4*E**(2*B*L)*cos(B*L) - E**(2*B*L)*cos(3*B*L) + 4*
E**(2*B * L)*sin(B*L) + E**(2*B*L)*sin(3*B*L) - cos(B*L) + sin(B*L)))/( L
*P*S**2*(E**(8*B*L) + 2*E**(6*B*L)*cos(2*B*L) - 2* E**(6*B*L)* sin (2*B*L)
+ 4*E**(6*B*L) - 2*E**(4*B*L)*sin(4*B*L) - 8* E**(4* B *L)*sin(2*B*L) -
2*E**(2*B*L)*cos(2*B*L) - 2*E**(2*B*L)*sin(2 * B*L) - 4*E**(2*B*L) - 1));

G1 = (2*B**3*(E**(4*B*L) + 2* E**(2 * B*L)*cos(2*B*L) + 4*E**(2*B*L) +
1))/(P*S**2*(E**(4*B*L) - 2* E **(2*B*L)*sin(2*B*L) - 1));
```

The MATLAB files for generating the transfer functions in section 4.3 are listed below:

```
F.m:
function y = F(x)
%Function of y used for obtaining the results of λ in equation (4-30).
y = (exp(x) + exp(-x))/2*cos(x) + 1;

ValueB1B2B3B4.m:
function [B1,B2,B3,B4] = ValueB1B2B3B4(lambda,L)
%Function [B1,B2,B3,B4] = ValueB1B2B3B4(lambda,L):
%    To find the value of B1, B2, B3, B4 in the function of phi(x) of equation (4-22)
from equation (4-23).
%Output:
%    A, B, C, D
%Input:
%    lambda
%    L
%
%    1/89

ans1 = 2*sinh(lambda*L)*sinh(2*lambda*L)*sin(lambda*L)-..
    2*sinh(lambda*L)*cosh(2*lambda*L)*cos(lambda*L)+..
    10*sinh(lambda*L)*cos(lambda*L)+8*sinh(lambda*L)*sin(lambda*L)*lambda*L;
ans1 = ans1+2*sinh(2*lambda*L)*cosh(lambda*L)*cos(lambda*L)+..
    3*sinh(2*lambda*L)*cos(2*lambda*L);
ans1                    =                    ans1+3*sinh(2*lambda*L)-
```

```
2*cosh(lambda*L)*cosh(2*lambda*L)*sin(lambda*L)-..
        10*cosh(lambda*L)*sin(lambda*L)-3*cosh(2*lambda*L)*sin(2*lambda*L)+..
        2*cosh(2*lambda*L)*lambda*L-2*cos(2*lambda*L)*lambda*L-
3*sin(2*lambda*L);
ans2 = 2*lambda*L*(4*sinh(lambda*L)*sin(lambda*L)+cosh(2*lambda*L)-..
        cos(2*lambda*L));
%ans1 and ans2 can be obtained by REDUCE fron the integration equation (4-23).
B1 = sqrt(ans2/ans1);
B2 = -(cosh(lambda*L)+cos(lambda*L))/(sinh(lambda*L)+sin(lambda*L))*B1;
B3 = -B1;
B4 = -B2;


Timemethod.m
function [G,G1] = Timemethod(w,EI,rho,L,N)
%Function [G,G1] = Timemethod,(w,EI,rho,L);
%The frequency response of approximated method by transfer functions derived
%    in time domain
%Output:
%       G       The T-F of angle of the tip point versus input torque
%       G1 The T-F of angle of the torqued point versus input torque
%Input:
%       w       Frequency
%       EI      Stiffness
%       rho     Mass per unit
%       L       Length
%       N   Approximated number
%
```

```
%       1/89.
for n = 1:N
x = (n-0.5)*pi;
al = zeroin('F',x)/L;
lambda(n) = al;
[B1,B2,B3,B4] = ValueB1B2B3B4(al,L);
phi(n) = B1*cosh(al*L) + B2*sinh(al*L) + B3*cos(al*L) + B4*sin(al*L);
Ks(n) = - L*(B1*(al*L*sinh(al*L) - cosh(al*L) + 1) + B2*(al*L*cosh(al*L) -
sinh(al*L))+..
      B3*(al*L*sin(al*L) + cos(al*L) - 1) - B4*(al*L*cos(al*L) - sin(al*L))) /
(al*L)^2;
end
omegaSquare= lambda.^4*EI/rho;
s=sqrt(-1).*w;
for n = 1:N
qs(n,:) = (omegaSquare(n) + s.^2).^2;
qq(n,:) = Ks(n)*qs(n,:);
end
Y = phi*qq;
%torque equation (4-3)
torque = 1/3*rho*L^3.*s.^2 - rho*L*Ks*qq.*s.^2;
G = torque.\(1+Y/L);
G1 = torque.\1;
```

The REDUCE codes and the expansion form of B, H, an C of section 4.4 is listed below:

```
%N is the order of the approximation.
%B, H, C are the terms of equation (4-36).
%Q is the general virables.
%M is the mass of the flexible link.
%L is the length of the link.
%G is the gravity acceleration constant.
%DTH is the rotation speed of the link.
%EI is the stiffness of the link.
%The calculation process is based on [3].

N := 6;
MATRIX B(N,N),H(N,N),Q(N,1),C(N,1);
Q:=MAT((q1),(q2),(q3),(q4),(q5),(q6));
FOR J:=1 STEP 1 UNTIL N DO
<<FOR K:=1 STEP 1 UNTIL N DO
 <<B(J,K):=M*L**(J+K)/(J+K+1)>>;
 F(J,1):=M*G*SIN(Q1)*L**J/(J+1)>>;
B(1,1):=M*L**2/3;
FOR K:=2 STEP 1 UNTIL N DO
<<FOR J:=2 STEP 1 UNTIL N DO
 <<B(1,1):=B(1,1)+Q(K,1)*Q(J,1)*L**(K+J)/(K+J+1)*M;
   H(1,J):=2*M*DTH*Q(K,1)*L**(K+J)/(K+J+1)+H(1,J);
   C(J,1):=M*DTH**2*Q(K,1)*L**(K+J)/(K+J+1)-
        EI*(K**2-K)*(J**2-J)*Q(K,1)*L**(K+J-3)/(K+J-3)+C(J,1)>>;
```

```
C(1,1):=C(1,1)+M*G*Q(K,1)*L**K/(K+1)*COS(Q1)>>;
;
END;
%End of the codes.
```

The results of B, H, C are:

B(1,1) = (L**2*M*(13860*L**10* Q6**2 + 30030*L**9* Q6*Q5 + 32760*L**8* Q6*Q4 + 16380*L**8 *Q5**2 + 36036*L**7* Q6*Q3 + 36036*L**7 *Q5*Q4 + 40040*L**6* Q6*Q2 + 40040*L**6* Q5*Q3 + 20020*L**6* Q4**2 + 45045*L**5* Q5*Q2 + 45045*L**5* Q4* Q3 + 51480*L**4* Q4*Q2 + 25740*L**4* Q3**2 + 60060*L**3 *Q3*Q2 + 36036*L**2* Q2**2 + 60060)) / 180180

B(1,2) = L**3*M/4 B(1,3) = L**4*M/5 B(1,4) = L**5*M/6

B(1,5) = L**6*M/7 B(1,6) = L**7*M/8

B(2,1) = L**3*M/4 B(2,2) = L**4*M/5 B(2,3) = L**5*M/6

B(2,4) = L**6*M/7 B(2,5) = L**7*M/8 B(2,6) = L**8*M/9

B(3,1) = L**4*M/5 B(3,2) = L**5*M/6 B(3,3) = L**6*M/7

B(3,4) = L**7*M/8 B(3,5) = L**8*M/9 B(3,6) = L**9*M/10

B(4,1) = L**5*M/6 B(4,2) = L**6*M/7 B(4,3) = L**7*M/8

B(4,4) = L**8*M/9 B(4,5) = L**9*M/10 B(4,6) = L**10*M/11

B(5,1) = L**6*M/7 B(5,2) = L**7*M/8 B(5,3) = L**8*M/9

B(5,4) = L**9*M/10 B(5,5) = L**10*M/11 B(5,6) = L**11*M/12

B(6,1) = L**7*M/8 B(6,2) = L**8*M/9 B(6,3) = L**9*M/10

B(6,4) = L**10*M/11 B(6,5) = L**11*M/12 B(6,6) = L**12*M/13

H(1,1) = 0

H(1,2) = (L**4*M*DTH* (280*L**4* Q6 + 315*L**3* Q5 + 360*L**2* Q4 + 420 *L*Q3 + 504* Q2))/1260

H(1,3) = (L**5*M*DTH* (252*L**4* Q6 + 280*L**3* Q5 + 315*L**2* Q4 + 360 *L*Q3 + 420* Q2))/1260

H(1,4) = (L**6*M*DTH* (2520*L**4* Q6 + 2772*L**3* Q5 + 3080*L**2* Q4 + 3465*L*Q3 + 3960* Q2))/13860

H(1,5) = (L**7*M*DTH* (330*L**4* Q6 + 360*L**3* Q5 + 396*L**2* Q4 + 440 *L*Q3 + 495* Q2))/1980

H(1,6) = (L**8*M*DTH* (1980*L**4* Q6 + 2145*L**3* Q5 + 2340*L**2* Q4 + 2574*L* Q3 + 2860* Q2))/12870

H(2,1) = 0 H(2,2) = 0 H(2,3) = 0 H(2,4) = 0 H(2,5) = 0 H(2,6) = 0

H(3,1) = 0 H(3,2) = 0 H(3,3) = 0 H(3,4) = 0 H(3,5) = 0 H(3,6) = 0

H(4,1) = 0 H(4,2) = 0 H(4,3) = 0 H(4,4) = 0 H(4,5) = 0 H(4,6) = 0

H(5,1) = 0 H(5,2) = 0 H(5,3) = 0 H(5,4) = 0 H(5,5) = 0 H(5,6) = 0

H(6,1) = 0 H(6,2) = 0 H(6,3) = 0 H(6,4) = 0 H(6,5) = 0 H(6,6) = 0

C(1,1) = (G*L*M* (60*COS(Q1)*L**5* Q6 + 70*COS(Q1)*L**4* Q5 + 84* COS(Q1)*L**3* Q4 + 105* COS(Q1)*L**2* Q3 + 140* COS(Q1)*L*Q2 + 210* SIN(Q1)))/420

C(2,1) = (L*(840* SIN(Q1)*G*L*M + 280*L**7*M* DTH**2*Q6 + 315*L**6* M* DTH**2*Q5 + 360*L**5*M* DTH**2*Q4 + 420*L**4*M* DTH**2* Q3 - 30240*L**4* EI*Q6 + 504*L**3*M* DTH**2*Q2 - 25200 *L**3*EI*Q5 - 20160*L**2* EI*Q4 - 15120*L*EI*Q3 - 10080* EI*Q2))/2520

C(3,1) = (L**2*(630* SIN(Q1)*G*L*M + 252*L**7*M* DTH**2*Q6 + 280*L**6 *M* DTH**2*Q5 + 315*L**5*M* DTH**2*Q4 + 360*L**4*M* DTH**2 *Q3 - 75600*L**4* EI*Q6 + 420*L**3*M* DTH**2*Q2 - 60480*L**3* EI*Q5 - 45360*L**2* EI*Q4 - 30240*L*EI* Q3 - 15120* EI*Q2))/2520

C(4,1) = (L**3*(5544*SIN(Q1)*G*L*M + 2520*L**7*M* DTH**2*Q6 + 2772 *L**6*M* DTH**2*Q5 + 3080*L**5*M* DTH**2*Q4 + 3465*L**4*M * DTH**2*Q3 - 1425600*L**4* EI*Q6 + 3960*L**3*M* DTH**2* Q2 - 1108800*L**3* EI*Q5 - 798336*L**2* EI*Q4 - 498960*L* EI*Q3 - 221760* EI*Q2))/27720

```
C(5,1) = (L**4*(4620*SIN(Q1)*G*L*M + 2310*L**7*M* DTH**2*Q6 + 2520
*L**6*M* DTH**2*Q5 + 2772*L**5*M* DTH**2*Q4 + 3080*L**4*M *
DTH**2*Q3 - 2079000*L**4* EI*Q6 + 3465*L**3*M* DTH**2* Q2 -
1584000*L**3* EI*Q5 - 1108800*L**2* EI*Q4 - 665280*L* EI*Q3 - 277200*
EI*Q2))/27720
C(6,1) = (L**5*(25740*SIN(Q1)*G*L*M + 13860*L**7*M* DTH**2*Q6 +
15015*L**6*M* DTH**2*Q5 + 16380*L**5*M* DTH**2*Q4 + 18018*L**4*M*
DTH**2*Q3 - 18018000*L**4* EI*Q6 + 20020*L**3*M* DTH**2*Q2 -
13513500*L**3* EI*Q5 - 9266400*L**2* EI*Q4 - 5405400*L* EI*Q3 - 2162160*
EI*Q2))/180180
```

The MATLAB programmes for section 4.5 is listed below:

```
SpringMethod.m
function [theta,W] = SpringMethod(k)
% The transfer function obtained by using adding 8 springs
% The one-link manipulator is a uniform beam.
% k is the stiffness of the springs.
% M1 is the rigid link's mass, M1 = 1/8Kg. L is the length, L = 1/8m. II is the rotation inertia  along the axis through its centre.
M1=0.125;
L=0.125;
II=M1*L^2/12;
%The symbolic value of M matrix is obtained by running the programmes in chapter 2.
    M(1,1)=2.*(85.*L^2*M1+4.*II);
    M(1,2)=(7.*(79.*L^2*M1+4.*II))/4.;
    M(1,3)=(215.*L^2*M1+12.*II)/2.;
    M(1,4)=(5.*(63.*L^2*M1+4.*II))/4.;
    M(1,5)=53.*L^2*M1+4.*II;
    M(1,6)=(125.*L^2*M1+12.*II)/4.;
    M(1,7)=(29.*L^2*M1+4.*II)/2.;
    M(1,8)=(15.*L^2*M1+4.*II)/4.;
    M(2,1)=(7.*(79.*L^2*M1+4.*II))/4.;
    M(2,2)=(7.*(65.*L^2*M1+4.*II))/4.;
    M(2,3)=(179.*L^2*M1+12.*II)/2.;
    M(2,4)=(5.*(53.*L^2*M1+4.*II))/4.;
    M(2,5)=45.*L^2*M1+4.*II;
```

```
M(2,6)=(107.*L^2*M1+12.*II)/4.;
M(2,7)=(25.*L^2*M1+4.*II)/2.;
M(2,8)=(13.*L^2*M1+4.*II)/4.;
M(3,1)=(215.*L^2*M1+12.*II)/2.;
M(3,2)=(179.*L^2*M1+12.*II)/2.;
M(3,3)=(143.*L^2*M1+12.*II)/2.;
M(3,4)=(5.*(43.*L^2*M1+4.*II))/4.;
M(3,5)=37.*L^2*M1+4.*II;
M(3,6)=(89.*L^2*M1+12.*II)/4.;
M(3,7)=(21.*L^2*M1+4.*II)/2.;
M(3,8)=(11.*L^2*M1+4.*II)/4.;
M(4,1)=(5.*(63.*L^2*M1+4.*II))/4.;
M(4,2)=(5.*(53.*L^2*M1+4.*II))/4.;
M(4,3)=(5.*(43.*L^2*M1+4.*II))/4.;
M(4,4)=(5.*(33.*L^2*M1+4.*II))/4.;
M(4,5)=29.*L^2*M1+4.*II;
M(4,6)=(71.*L^2*M1+12.*II)/4.;
M(4,7)=(17.*L^2*M1+4.*II)/2.;
M(4,8)=(9.*L^2*M1+4.*II)/4.;
M(5,1)=53.*L^2*M1+4.*II;
M(5,2)=45.*L^2*M1+4.*II;
M(5,3)=37.*L^2*M1+4.*II;
M(5,4)=29.*L^2*M1+4.*II;
M(5,5)=21.*L^2*M1+4.*II;
M(5,6)=(53.*L^2*M1+12.*II)/4.;
M(5,7)=(13.*L^2*M1+4.*II)/2.;
M(5,8)=(7.*L^2*M1+4.*II)/4.;
```

```
    M(6,1)=(125.*L^2*M1+12.*II)/4.;
    M(6,2)=(107.*L^2*M1+12.*II)/4.;
    M(6,3)=(89.*L^2*M1+12.*II)/4.;
    M(6,4)=(71.*L^2*M1+12.*II)/4.;
    M(6,5)=(53.*L^2*M1+12.*II)/4.;
    M(6,6)=(35.*L^2*M1+12.*II)/4.;
    M(6,7)=(9.*L^2*M1+4.*II)/2.;
    M(6,8)=(5.*L^2*M1+4.*II)/4.;
    M(7,1)=(29.*L^2*M1+4.*II)/2.;
    M(7,2)=(25.*L^2*M1+4.*II)/2.;
    M(7,3)=(21.*L^2*M1+4.*II)/2.;
    M(7,4)=(17.*L^2*M1+4.*II)/2.;
    M(7,5)=(13.*L^2*M1+4.*II)/2.;
    M(7,6)=(9.*L^2*M1+4.*II)/2.;
    M(7,7)=(5.*L^2*M1+4.*II)/2.;
    M(7,8)=(3.*L^2*M1+4.*II)/4.;
    M(8,1)=(15.*L^2*M1+4.*II)/4.;
    M(8,2)=(13.*L^2*M1+4.*II)/4.;
    M(8,3)=(11.*L^2*M1+4.*II)/4.;
    M(8,4)=(9.*L^2*M1+4.*II)/4.;
    M(8,5)=(7.*L^2*M1+4.*II)/4.;
    M(8,6)=(5.*L^2*M1+4.*II)/4.;
    M(8,7)=(3.*L^2*M1+4.*II)/4.;
    M(8,8)=(L^2*M1+4.*II)/4.;

unit=[0 0 0 0 0 0 0 0;0 1 0 0 0 0 0 0;0 0 1 0 0 0 0 0;0 0 0 1 0 0 0 0;
    0 0 0 0 1 0 0 0;0 0 0 0 0 1 0 0;0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1];
```

```
torque=[1;0;0;0;0;0;0;0];
theta = []; W = [];
n=0;
for j=0:4/300:4;
w = 10^j;
n=n+1;
s=sqrt(-1)*w;
mm=M*s*s;
mm=mm+k*unit;
thi=mmorque;
theta(n)=[1 7/8 6/8 5/8 4/8 3/8 2/8 1/8]*thi;
W(n)=w;
end
%End of the programme.
```