Analysis and Design Environment for Flexible Manipulators

Ravn, Ole; Poulsen, Niels Kjølstad

Published in:
Flexible Robot Manipulators: Modelling, Simulation and Control

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Chapter 19

Analysis and Design Environment for Flexible Manipulators

Ole Ravn[1] and Niels Kjølstad Poulsen[2]

[1] *Automation, Ørsted.DTU, the Technical University of Denmark*

[2] *Informatics and Mathematical Modelling, the Technical University of Denmark*

In this chapter we will focus on a design environment for modelling and control of flexible link robots. The environment consists of some physical equipments and a software environment for design and analysis. The physical equipments are two computer platforms, a design platform for analysis and design and, a real time (LINUX) platform for controlling a laboratorial rig equipped with actuators and measuring systems. The rig has several configurations including a horizontal and a vertical mounting of the robot. In the horizontal mounting the gravity plays a minor role while it is an important issue in the vertical mounting. The actual rig has both a simple 1DOF and a more advanced 2DOF configuration. The experimental rig is controlled from a real time (LINUX) platform. The software environment for design, analysis and simulation is located on a design platform in terms of a MATLAB/SIMULINK work bench extended with a mechatronic SIMULINK library (MSL). Here the different principle for modelling and control can be combined resulting in various control designs and strategies. A simple switch can change the configuration from simulation to real time application. The real time code can be generated using Real Time Workshop (RTW) and transferred to the real time platform in order to perform the experiments. The collected data can also be transmitted back to the design platform for further analysis.

19.1 Introduction

The desire for high-performance manipulators and the benefits offered by a lightweight flexible arm have lead to analysis in which flexibility is the essential issue. This is especially the case for manoeuvring of large pay-loads. The high-performance requirements will inevitably produce designs that during operation will excite vibrations in the manipulator structure.

The flexibility generates a severe problem in controlling the motion due to the inevitably excitation of structural vibrations which affect the accuracy of the manipulator. Therefore a successful controller implementation of a flexible manipulator system is contingent on achieving acceptable performance taking into account variations in e.g. payload and environmental disturbances.

The aim of the controller is to suppress the structural vibration while in addition to minimize the cycle time of the manipulator system. For flexible manipulator systems, it is necessary to use a model-based controller in order to mitigate the first harmonics. However, changes in

payload degrade the model and consequently the performance of the control system, unless some sort of adaptation or gain scheduling is taken into account of estimate these effects.

In order to investigate different aspects of control of flexible links robot configurations an experimental setup has been made. This experimental setup form the basis for the work described in this chapter. The setup (see Figure 19.1 a photo and Figure 19.2 a schematic view) consists of the design platform, a real time system and the actual experimental rig. The rig has several configurations and it can be mounted in a horizontal and a vertical position. In the horizontal position the gravity plays ignorable role. The rig has a 1DOF and 2DOF configuration. The rig consists (in its 2DOF configuration) of two very flexible links with two actuators located in the joints. The geometry of the links makes the predominant bending take place in this plane making it possible to ignore torsion. The actuators are DC-motors with a sufficient gear ratio and tachometers making an analog velocity feedback feasible, this suppresses the friction and other non-linearities in the actuators. Apart from the tachometers there are two kinds of sensors on the setup, a potentiometer in each joint enabling a measurement of the (angular) position of the joint and a number of strain gauges located on each link enabling the measurement of the bending of the link.
When dealing with real lab rigs it is very important to have a Control System Design environment that supports the design process in order to be able to put emphasis on the controller design and not on practical details.

Figure 19.1: The robot consists of hub where different type of arms can be mounted. Notice the wall mount enabling the robot to move in the vertical plane instead of horizontally. The arm mounted is a 1DOF arm; on the table are another 1DOF arm and an arm with two flexible links. It is equipped with tachometers and strain gauges in order to measure the link angles and deflection, respectively.

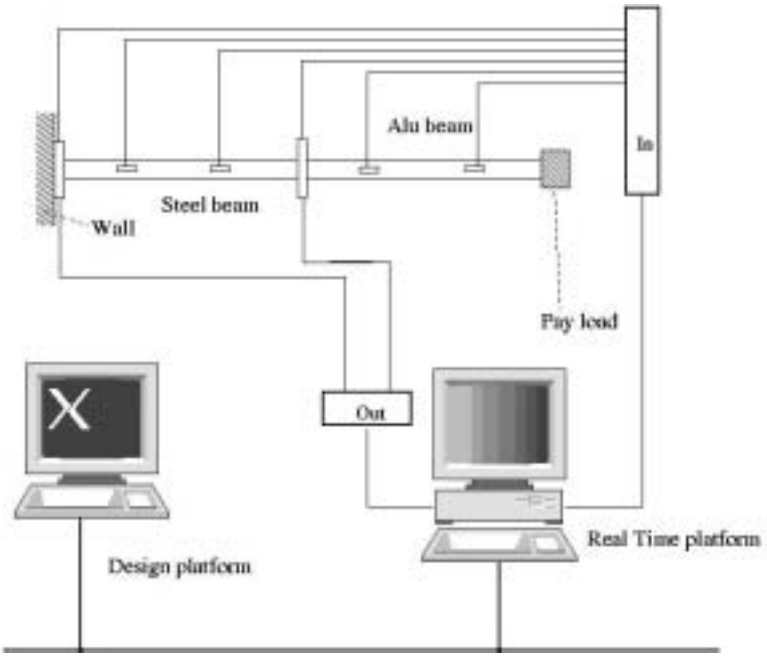


Figure 19.2: The experimental setup consists of a design platform, a real time system and the experimental rig. The actuators a two (or one) DC motor, two (or one) potentiometers measuring the angular positions of the joints and a number (two or four) of strain gauges measuring the deflection of the beam(s). On the design platform there is a MATLAB environment for design, simulation and evaluation. The real time platform executes the automatically generated code and collects data for analysis and validation.

19.2 Computer Aided Control System (CACE) design paradigm

In this section some background on the Computer Aided Control Engineering (CACE) is given in order to motivate the design of the software for controlling the flexible robot rig.

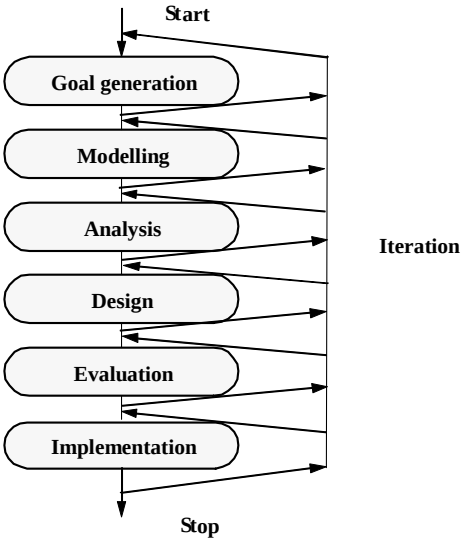


Figure 19.4 Diagram of the classical design model.

In the classical model of the control development process presented in [Ravn and Szymkat, 1992] shown in figure 19.4 the following general phases have been distinguished:

- **Goal Generation**. This phase initiates the design process. The problem and the desired features of the solution are determined. This phase is normally done in cooperation with the customer, other engineers etc. Normally no formalized tools or methods are used here.
- **Modelling.** The modelling phase is used to determine a model of the system to be controlled. This is normally a mathematical model, which can be used by the tools of the following phases. Models of different complexity may be derived such as linear plant models for the design of linear controllers and then non-linear plant models in the evaluation phase. Many CACE tools exist for assisting the user during this phase.
- **Analysis**. The derived model is analysed in order to gain an understanding of the system and the potential problems. The analysis results are used as a basis for choosing a controller structure. Not just the normal numerical tools are applicable in this phase; the potential benefits of using symbolic manipulation tools are becoming more and more evident and many of the numerical packages have built-in symbolic tools or interface to them.
- **Design**. A possible controller structure is selected and the parameters are chosen in order to match the design goals. It may be useful to consider more controller structures and compare their performance in parallel. Many tools for designing standard LQ, LQG etc. controllers exist.
- **Evaluation**. The different controllers are considered in this phase and compared with respect to the features of the desired solution set up in the first phase of the design process. The degree of compliance with the goals is determined and the best controller selected. The evaluation phase may use simulation of the system or use partially the real-time interface in order to select the best controller. More models may be used in order to gain insight into what features of the system and the controller limit the performance.
- **Implementation.** The chosen mathematical description of the controller is implemented. More and more tools are emerging in this field. The standard packages have C-code generation tools and offer hardware, which can be used for testing the controller in a laboratory environment. The main problem here is the balance between code efficiency, hardware dependency and the degree of automation of the phase.

Another element of the design process model is the iteration, which is its fundamental property. The iteration can be performed manually, semi-automatically or automatically. The iterative nature of the design process is also an important element. An overall evaluation of the design phases indicates that most CACE tools are available for the Modelling, Analysis and Design phases. Some tools are also available for the Implementation phase. However there is a lack of tools for the rest of the phases and the iteration. Many alternative design process models have been discussed in numerous papers see [MacFarlane et al., 1989], [Ravn and Szymkat, 1992], [Barker et al., 1993].

The proposed design models emphasize the importance of the simulation most commonly becoming central and integrating phase of the design cycle. The simulation centred model of the design process presented in this section is based on the fact that different users have

different approaches to the design problem. Two major categories have been found. The application engineer, how is focused on getting his application controlled in the optimal way and the algorithm designer, whose main goal is to develop and validate the control algorithm on some object. The model is shown in Figure 19.5

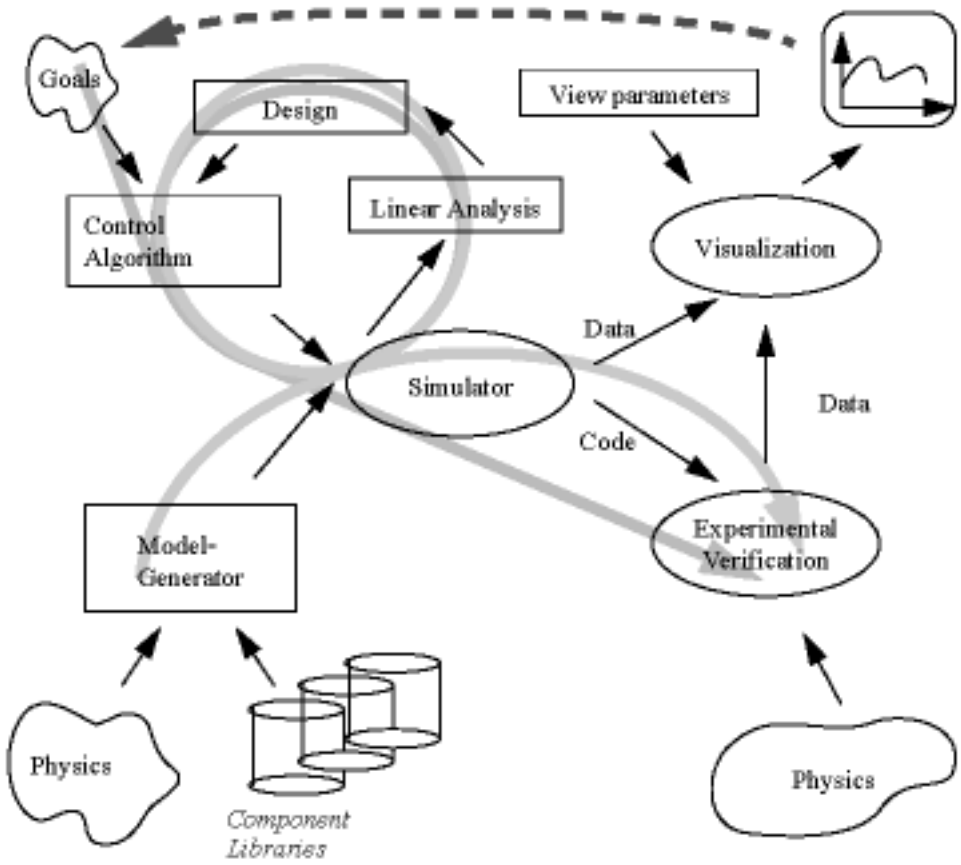


Figure 19.5: A simulation centred design process model seen from the algorithm and application designers points of view

in the first instance i.e. from the application engineers point of view. The model of the design process is centred on the simulator; this is not a simulator in the limited sense of the term but includes the ability of making real-time code, simulation data and linearizing the non-linear model of the system. The simulator takes a control algorithm and a system model as inputs and produces results in terms of data and real-time code. The system model is generated by a model generator based on the physics of the system drawing components from a component library. The control algorithm is found in an iterative way as described earlier. The results of the simulation are visualized in different ways, as plot of variables or as a visualization of the actual components of the system in an animation. The result is then evaluated against the goals of the system performance. When the goals are met by the simulated results the real-time code is generated in the simulation block and the experimental verification is done. The visualization of the results obtained here is validated against the simulation results using the same visualization module and compared with the goals. Finally the design should be implemented for production purposes, which quite different from the implementation for experimental verification and rapid prototyping.

The application engineer starts by modelling the system to be control and ends with the experiments. The starting point of the algorithms developer is different as shown in the figure. The algorithm developer start by determining the goals having the control algorithms to be validated some model is chosen for the validation. Iterations are done modifying the parameters or structure of the control algorithm until the goals are met. The control algorithm is then possible validated through experiments.

The two points of view are quite similar in structure but the focus is different. The result of the process should not just be the control algorithms but also some measure on the sensitivity to imperfect initial conditions in the design. The validation results should be presented in a form suitable for documentation purposes.

19.3 Mechatronic Simulink Library (MSL)

In the design of Mechatronic Simulink Library (MSL) [Ravn and Szymkat, 1995], [Szymkat et al., 1995], [Ravn et al., 1996] a number of important observations was made. In the modelling phase it is very important to look at the following aspects:

- Component based modelling so that the simulation model structure closely resembles the physical model structure to facilitate the easy exchange of components.
- Consistent definition of the input/output of components
- Handling of parameters through a database and/or tuneable through an interface.
- Tuneable granularity/complexity as shown below.

In the prototype implementation of MSL a number of components often used in the laboratory have been made. In figure 19.6 the current groups of components are shown. It should be noted that MSL is designed in such a way that the addition of new component types is simple and the user should use this facility, as only the most basic component types are included from the beginning.

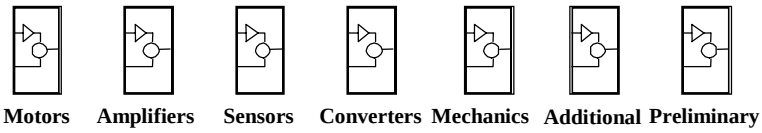


Figure 19.6 The Mechatronic Simulink Library

The DC motor model is shown in figure 19.7. In the figure the dialogue box is shown, in which type of DC motor is input as well as if the parameters of friction, stiction and fluid friction should be taken from the database (default) or the dialogue box. Furthermore it can be chosen if friction should be simulated or not and the level of extra dynamics included in this case the electrical time constant of the drive. As also shown in the figure there are two variants of the drive, a normal and a triggered version. The triggered blocks is a feature of Simulink enabling the user to only simulate certain blocks in the diagram. This is utilized in MSL to enhance simulation performance of the simplest complexity level significantly. In the traditional version all parts of a component is simulated even though the results from some blocks are not used.

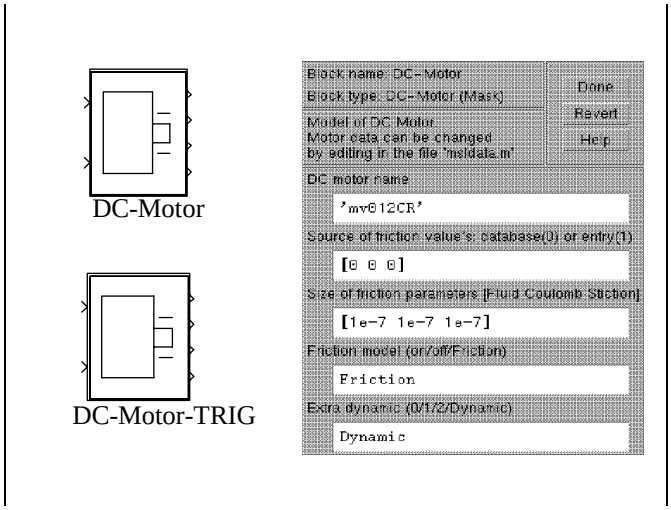


Figure 19.7 DC motor symbols and dialogue box.

To demonstrate possibility to make simple changes in the complexity of the MSL models the simple servomechanism of the Automation, Ørsted•DTU has been modelled and calibrated see figure 19.8.

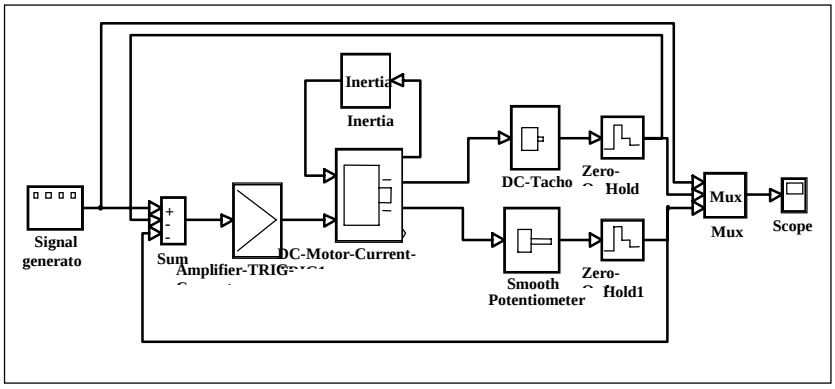


Figure 19.8 MSL Position servomechanism

In figure 19.9 the shaft position of the servomechanism is shown when the global variable '`Friction`' is respectively '`on`' and '`off`'.

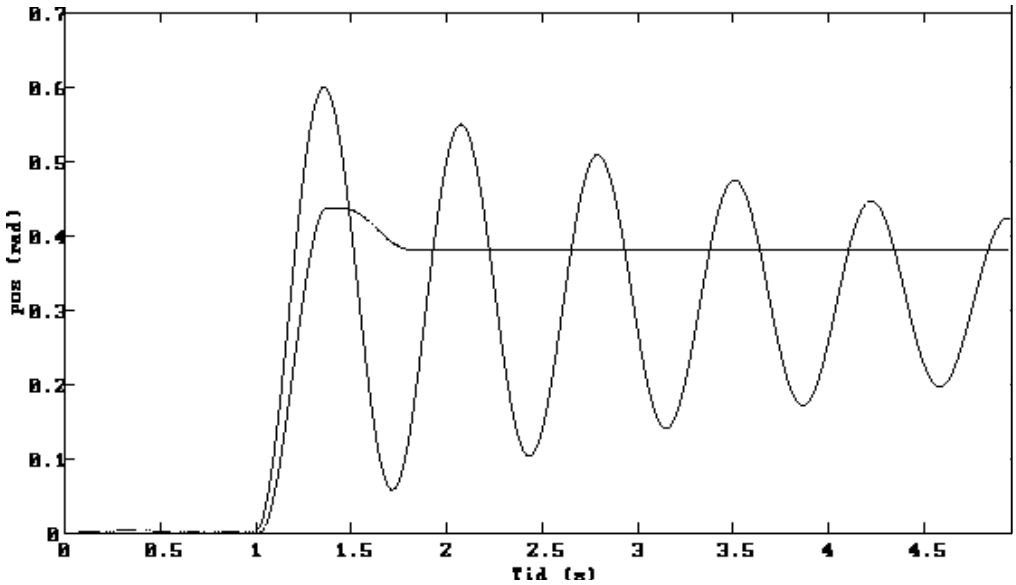


Figure 19.9 Step response for the simple servomechanism with and without friction.

Another system example is a simple 1 DOF flexible robot arm. The arm is a 1 joint robot where is arm is made from i very flexible material. The bending of arm is measured using two strain gauges. A drawing is viewed in figure 19.10.

Data and more details relating to the flexible robot arm and the modelling in MSL is found in [Rostgaard, 1995], [Lund, 1994], [Andersen, 1993] and [Andersen and Baungaard, 1992].

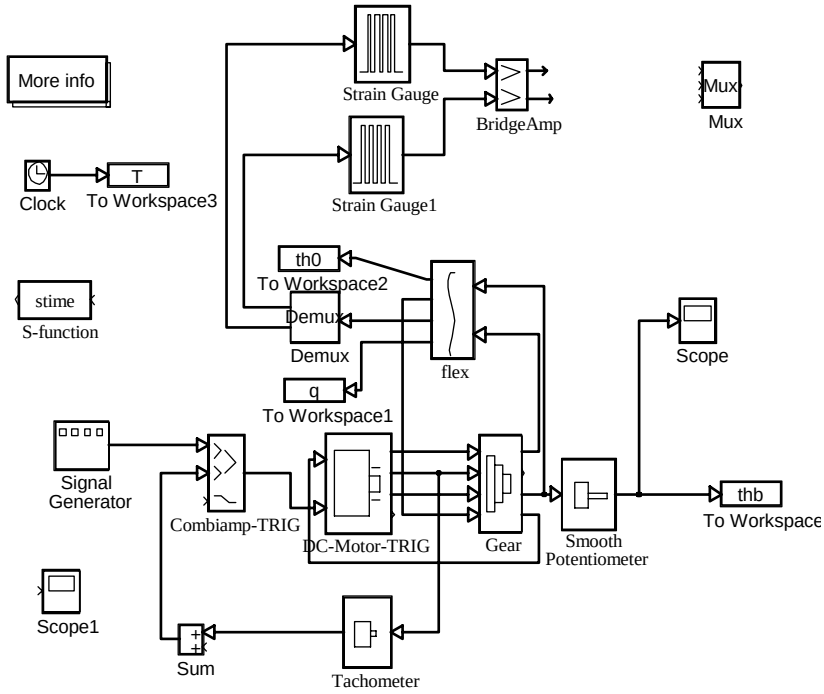


Figure 19.11 Mechatronic Simulink Library model of the flexible robot

In figure 19.11 the finished MSL model of the flexible robot is shown. Below the plot of the end point angle and the motor angle and the harmonic time functions q are shown for a step input in motor position using a simple position control and without taking the flexibility into account. The simulation is shown with one and two modes respectively. The change is done by inputting the number of modes into the dialogue box of the 'flex' block.

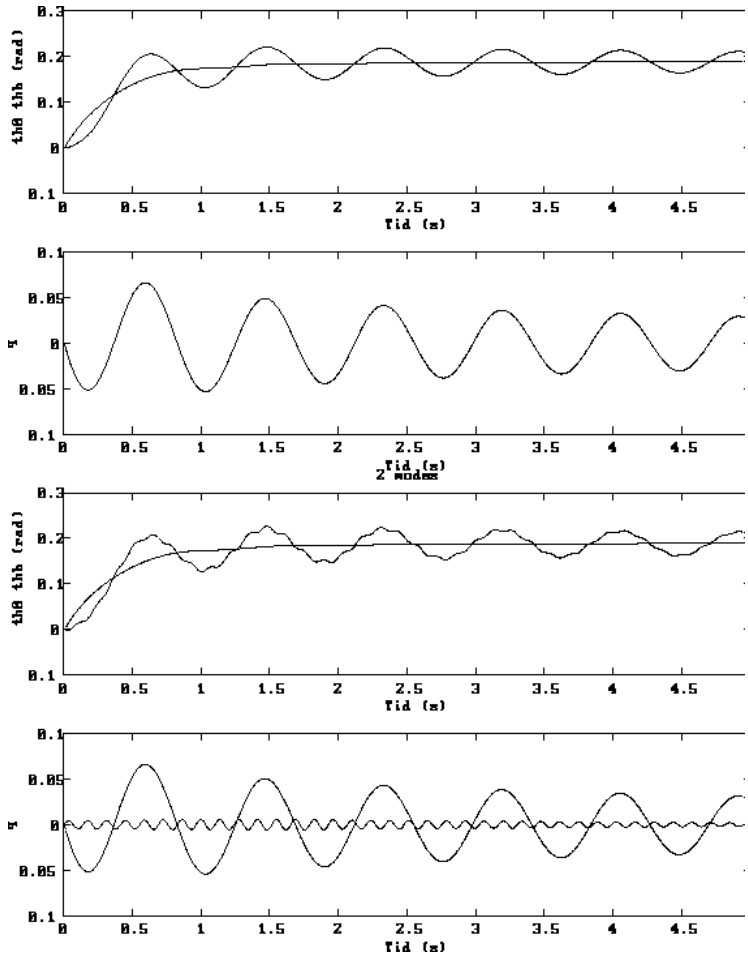


Figure 19.12 Step responses for the flexible robot modelled with one and two modes, respectively.

The MSL based model is thus an excellent alternative to modelling in plain Simulink. As mentioned it supports

- Component based modelling so that the simulation model structure closely resembles the physical model structure to facilitate the easy exchange of components.
- Consistent definition of the input/output of components
- Handling of parameters through a database and/or tuneable through an interface.
- Tuneable granularity/complexity as shown below.

More information on MSL can be found on `http://www.oersted.dtu.dk/personal/or/MSL`

19.4 Design models

The experimental setup exists in a horizon and a vertical configuration in which gravity plays an ignorable role in the horizontal configuration. The control design has in this work been based on physical models. These models can be based on a number of different approaches including Modal or Eigenvalue Method (see e.g [Kruise, 1990]) and Finite Elements Method (see e.g. [Sakawa, 1985]). Example on a FEM model wil in the following only be given for a 1DOF configuration.

The flexible manipulator system studied here (see Figure 19.13) carries a pay-load, m_p at its tip and moves in the horizontal or the vertical plane. For the 2 DOF flexible link robot he active degrees of freedom are the two rotational angles θ_{b1} and θ_{b2} (see Figure 19.14).

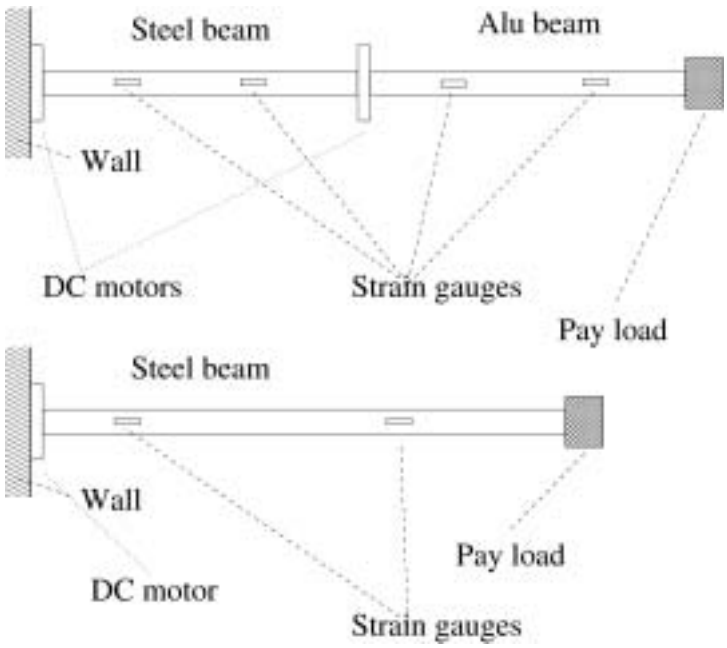


Figure 19.13: The experimental rig has a 1DOF and a 2DOF version. The 2DOF version consists of two flexible links, two actuators (DC motors) and four strain gauges for measuring the deflection of the links. The 1DOF version have one actuator and two strain gauge

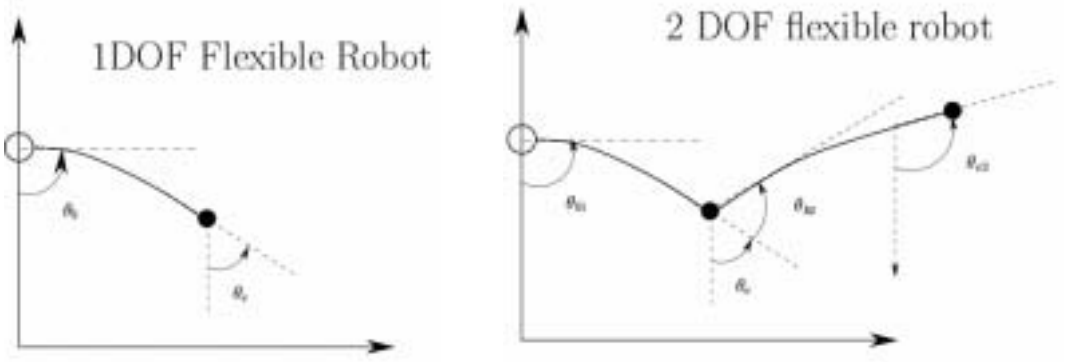


Figure 19.14 Definitions of angles for the 1DOF and 2DOF configuration of the flexible link robot.

19.4.1 Dynamics of the actuators

The actuators consist mainly of a DC motor supplied with a gear and a tacho feedback. Consider a DC motor, which can be described (to a reasonable degree) by a first order model as depicted in Figure 19.15.

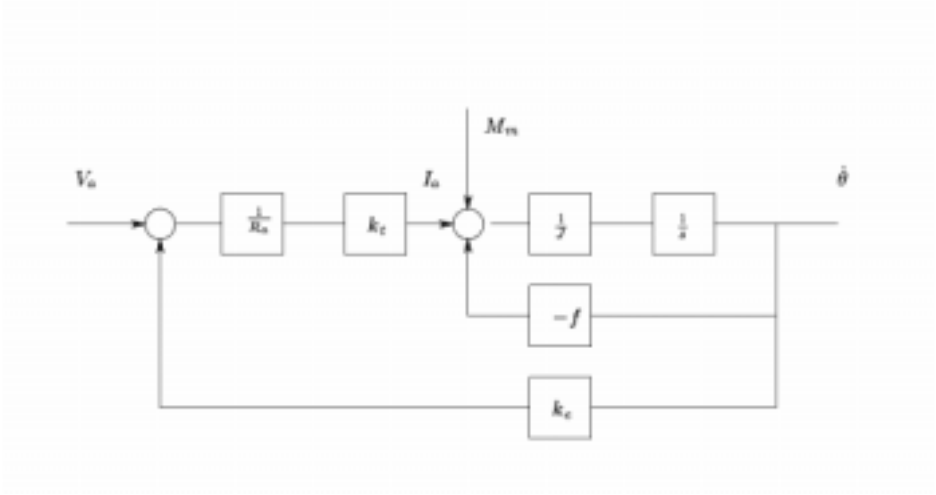


Figure 19.15: Block diagram for a DC-motor.

Let V_a denote the voltage input, T_m is the resulting external torque (i.e. torque not included in the model) and $\dot{\theta}$ is the angular velocity of the motor shaft. Then using Newton second law the mode can be given as

$$J\ddot{\theta}_m = T_m - f\dot{\theta}_m + \frac{k_t}{R_a}(V_a - k_e\dot{\theta}_m)$$

where $k_e = k_t$ represent motor constants, f the total viscous friction of the motor, R_a the electrical resistance and J the total inertia of motor.

In order to reduce the influence from disturbances, non-linearities and other imperfects, the DC-motor is included in a tacho loop, in which the difference between the reference voltage u and the tacho voltage $V_{tg} = k_{tg}\dot{\theta}_m$ is amplified (gain k_g) and feed into the motor, ie

$$V_a = k_p(u - k_{tg}\dot{\theta}_m)$$

A gear is introduced between the motor shaft (θ_m) and the manipulator (θ_b). The gear is considered stiff and is described by the model

$$\theta_b = \frac{1}{N}\theta_m$$

If the external momentum $T_b = NT_m$ is introduced the actuator can be given as

$$\ddot{\theta}_b = k_1 u + k_2 T_b + k_3\dot{\theta}_b \qquad (19.1)$$

Here k_1, k_2 and k_3 are constants. Notice this is a second order model.

Due to the span in time constants for the actuator and for the rest of the robot the fast part of the actuator dynamics is often neglected. In that case the actuators are modelled by:

$k_1 u + k_2 T_b + k_3 \dot{\theta}_b = 0$
which is a first order model.

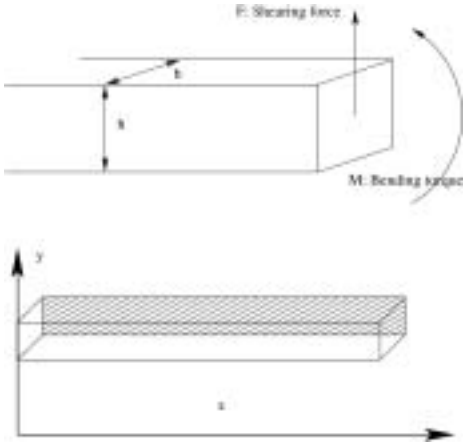
19.5.2 Modal models

The model of the flexible link robot consists of four parts; namely the models for the two actuators and the two arms. The dynamics of the flexible arms can be described by a PDE, which can be transferred into an ODE by using the a finite element method or the method of separation of variable. In that case the resulting model becomes a modal model in which the deflection, $w_j(x,t) \quad j = 1,2$ of the arms is written as

$$w_j(x,t) = \sum_{i=1}^{\infty} \varphi_{ji}(x) q_{ji}(t)$$

where $\varphi_{ji(x)}$ and $q_{ji}(t)$ are the normal and harmonic function of mode i and arm j, respectively. In the development of the model

Consider the beam segment in Figure 19.16. Let x denote the distance along the beam and $x(x,t)$ the deflection of the beam. Here a is the cross section area of the beam. The beam is physically described by b, h and L representing the width, height and length.



19.16: Forces and torque in a cross section of a beam

Since transversal vibrations appear athwart to the beam the cross section area and inertia are

$$a = bh \qquad I = \int_0^h (2 \int_0^{b/2} r^2 dr) dy = \frac{b^3 h}{12}$$

Taking the resulting shearing forces and torque we obtain the following relations

$$\frac{\partial F}{\partial x} + \rho a \frac{\partial^2 w(x,t)}{\partial^2 t} = 0 \qquad F = \frac{\partial M}{\partial x}$$

From the elementary flexural theory [Timoshenko, Yuong and Weaver, 1974] we have

$$M = EI \frac{\partial^2 w(x,t)}{\partial^2 x}$$

These three last equations give us the Euler-Bernoulli equation for the beam

$$EI \frac{\partial^4 w(x,t)}{\partial x^4} = -\rho a \frac{\partial^2 w(x,t)}{\partial t^2}$$

The solution to this equation with boundary conditions can be found by using separation of variable. This means

$$w(x,t) = \sum_{i=1}^{\infty} \varphi_i(x) q_i(t)$$

which is equivalent to expand the deflection of the beam in modes. The functions, $\varphi_i(x)$, defines the shape of the natural modes of vibration and are called principal functions or normal functions. It can be shown (see e.g. [Rostgaard, 1995]) for details) that the normal functions are orthogonal and posses other interesting properties. The functions, $q_i(t)$, describes the time dependence of the modal deflection. With this expansion we have

$$M = EI \sum_{i=0}^{\infty} q_i(t) \frac{\partial^2 \varphi_i(x)}{\partial x^2} \qquad F = EI \sum_{i=0}^{\infty} q_i(t) \frac{\partial^3 \varphi_i(x)}{\partial x^3}$$

For each mode we have the following time dependence

$$\frac{\partial^2 q_i(t)}{\partial t^2} + \omega_i^2 q_i(t) = 0 \qquad \text{or} \qquad \ddot{q}_i(t) + \omega_i^2 q_i(t) = 0 \tag{19.2}$$

The modal function is given by:

$$\frac{\partial^4 \varphi_i(x)}{\partial x^4} - \gamma_i^2 \varphi_i(x) = 0 \tag{19.3}$$

where

$$\omega_i^2 = \frac{EI}{\rho a} \gamma_i^4$$

The general solution to the mode shape equation, (19.3), can be found to

$$\varphi_i(x) = c_{1i} Cosh(\gamma_i x) + c_{2i} Sinh(\gamma_i x) + c_{3i} Cos(\gamma_i x) + c_{4i} Sin(\gamma_i x)$$

The constant can be determined from the end point constraints. If we consider a clamped free beam then we the following 4 conditions (to determine the 4 constants).

$$\varphi_i(0) = 0 \quad \frac{\partial \varphi_i(0)}{\partial x} = 0 \qquad \frac{\partial^2 \varphi_i(L)}{\partial x^2} = 0 \quad \frac{\partial^3 \varphi_i(L)}{\partial x^3} = 0$$

The first two conditions are due to the clamped end ($x = 0$) and the last two are caused by the free end ($F(L) = 0$, $M(L) = 0$). In order to obtain nontrivial solution the *frequency equation*

$$Cosh(\gamma_i L)Cos(\gamma_i L) = -1$$

has to fulfilled, i.e. determine the of γ_i (and ω_i). The first four normal functions (mode functions) are plotted in Figure 19.17.

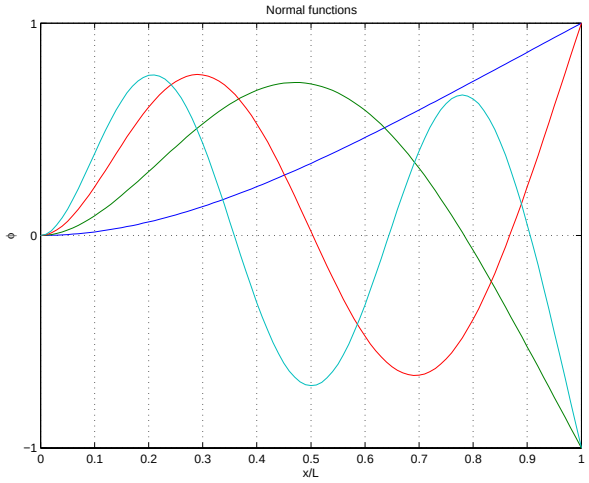


Figure 19.17: The first four mode shape functions

Let us now focus on a modal model of the flexible link robot. The robot exists in a 1 DOF and 2DOF configuration. Here only the 2DOF version is summarized. The robot consists of two actuators, two flexible links, two tacho encoders and four strain gauges. The four basic differential equations can be summarized (see e.g. [Rostgaard, 1995] for details).

For the shoulder-actuator:

$$k_{13}\dot{\theta}_{b1} + k_{11}u_1 + k_{12}E_1I_1\sum_{i=0}^{\infty}\varphi''_{1i}(0)q_{1i}(t) = \ddot{\theta}_{b1} \tag{19.4}$$

where k_{11}, k_{12} and k_{13} are constants related to the shoulder actuator, I_1 is the beam inertia for the upper arm, E_1 is the Young module for the beam and u_1 is the input (control) voltage. For the elbow actuator we have in a similar manner:

$$k_{23}\dot{\theta}_2 + k_{21}u_2 + k_{22}E_2I_2\sum_{i=0}^{\infty}\varphi''_{12i}(0)q_{2i}(t) = \ddot{\theta}_{b2} \tag{19.5}$$

For the lower arm we have the ODE equation for each mode ($i = 1,2,...,n$):

$$\varpi_{2i}^2 q_{2i}(t) + 2\zeta_{2i}\omega_{2i}\dot{q}_{2i}(t) + \ddot{q}_{2i}(t) = \sum_{j=1}^{\infty}\kappa_{2ij}^{*}\ddot{q}_{2j}(t) + \alpha_{2i}^{*}\left[\ddot{\theta}_{b2} + \ddot{\theta}_{b1} + \sum_{j=1}^{\infty}\varphi_{1j}'(L_1)\ddot{q}_{1j}(t)\right] + \beta_{2i}^{*}\left[L_1\ddot{\theta}_{b1} + \sum_{j=1}^{\infty}\varphi_{1j}(L_1)\ddot{q}_{1j}\right]\cos(\theta_{b2})$$

Here ω_{2i} and ζ_{2i} are the harmonic frequency and damping for mode I (and the second or lower arm). L_1 is the length of arm and the modal parameters for the lower arm are linearly depending of the payload, i.e.

$$\begin{aligned}
\alpha_{2i}^{*} &= \alpha_{2i} - \frac{m_p}{\mu_2}L_2\varphi_{2i}(L_2) \\
\beta_{2i}^{*} &= \beta_{2i} - \frac{m_p}{\mu_2}\ \varphi_{2i}(L_2) \\
\kappa_{2ij}^{*} &= -\frac{m_p}{\mu_2}\ \varphi_{2i}(L_2)\varphi_{2i}(L_2)
\end{aligned} \tag{19.6}$$

Here the payload free modal parameters, α_{2i} and β_{2i} depends on the geometry and the normal functions. The parameter μ_2 is one quarter of the mass of the link, i.e. $\mu_2 = \frac{1}{4}m_{l2}$

For the upper arm the situation becomes a little more complicated. This is due to the coupling between elbow-actuator and the deflection of the upper arm. Here the modal equations are:

$$\omega_{1i}^2 q_{1i}(t) + 2\zeta_{1i}\dot{q}_{1i}(t) + \ddot{q}_{1i}(t) = \sum_{j=1}^{\infty}\kappa_{1ij}^{*}\ddot{q}_{1j} + \ddot{\theta}_{b1}\left[\alpha_{1i}^{*} + \frac{J_h\varphi_{1i}'(L_1)}{\mu_1}\right] + \frac{J_h\varphi_{1i}'(L\ )}{\mu_1}\sum_{j=1}^{\infty}\varphi_{1j}'(L_1)\ddot{q}_{1j} - \frac{F_{ye}^{(1)}}{\mu_1}\varphi_{1i}(L_1) + \frac{J_2N_2\varphi_{1i}'(L_1)}{\mu_1}\left[k_{21}u_2 + k_{23}\dot{\theta}_{b2}\right]$$

where J_h, J_2 are hub and rotor inertia of actuator 2. Furthermore

$$\begin{aligned}
F_{ye}^{(1)} &= F_{b2}\cos(\theta_{b2}) + F_{x2}\sin(\theta_{b2}) \\
F_{b2} &= EI_2\sum_{j=1}^{\infty}\varphi_{2j}'''(0)q_{2j}(t) \\
F_{x2} &= (m_{l2} + m_p)\sin(\theta_{b2})\left[L_1\ddot{\theta}_{b1} + \sum_{j=1}^{\infty}\varphi_{1j}(L_1)\ddot{q}_{1j}(t)\right]
\end{aligned} \tag{19.7}$$

Here m_{l2} is the mass of arm 2 and the modal parameters α_{1i}^{*} and β_{1i}^{*} are independent of the payload mass (but do depend on the mass of actuator 2). Notice, the linear dependence on the pay load mass, m_p enters through (19.7).

If the actuator equations, (19.3) and (19.4), are used for obtaining the angular accelerations in (19.5) and (19.6) the four main equations can be written in a more compact form. Let us truncate the sums involved and introduce the notation:

$$z = \begin{bmatrix} \theta_{b1} \\ \theta_{b2} \\ \underline{q_1} \\ \underline{q_2} \end{bmatrix} \qquad \underline{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

where

$$\underline{q_1} = \begin{bmatrix} q_{11} \\ \vdots \\ q_{1n} \end{bmatrix} \qquad \underline{q_2} = \begin{bmatrix} q_{21} \\ \vdots \\ q_{2n} \end{bmatrix}$$

Then the description of the flexibility, (19.5) and (19.6), can be linearized and brought into the following compact form

$$\ddot{z} = M_1 z + M_2 \dot{z} + M_3 \ddot{z} + M_4 \underline{u} \tag{19.8}$$

where the matrices, $M_k, k = 1,...,4$ are affine in m_p. Notice, the matrices depend on the point of linearization. In this case the matrices depend only on θ_{b2}. Also, notice the angular acceleration, $\ddot{z}$ occurs on both side of the equation.

Now the compact description in (19.8) and is to be transformed into a state space description. The algebraic loop (related to $\ddot{z}$ in (19.8) can be solved if the following matrix inverse exists,

$$\Delta = (I - M_3)^{-1}$$

Notice M_3 depend linearly on the payload mass, m_p. Using the following definition of the state vector

$$x = \begin{bmatrix} z \\ \dot{z} \end{bmatrix}$$

the state space description

$$\dot{x} = \underline{\underline{A}} x + \underline{\underline{B}} \underline{u}$$

is obtained where

$$\underline{\underline{A}} = \begin{bmatrix} 0 & I \\ \Delta M_1 & \Delta M_2 \end{bmatrix} \quad \underline{\underline{B}} = \begin{bmatrix} 0 \\ \Delta M_4 \end{bmatrix}$$

Here the state transition matrix is of order $2+4n$ with n as the number of considered modes. For an arbitrary linearization angle, this representation is a linear approximation to the dynamics, i.e. M_k, $k = 1,...,4$ are functions of the linearization angle. Thus, one can use the measurements of θ_2 to obtain a running linear description around the actual orientation.

The measurement system consists of two tachometers and four strain gauges. The tachometers give measurements of the link angles θ_{b1} and θ_{b2} whereas the strain gauges are located tactically on the links in order to give measurements of the deflections $\bar{q}$.

The measurement are connected to the state of the description through

$$y_m(t) = C_m x(t) \qquad C_m = \begin{bmatrix} C_{tg} & 0 & 0 & \underline{0} \\ 0 & C_{sg1} & 0 & \underline{0} \\ 0 & 0 & C_{sg2} & \underline{0} \end{bmatrix}$$

Here C_{tg}, C_{sg1} and C_{sg2} are observation matrices for the two tachometers and the strain gauges located on the two links. These are:

$$C_{tg} = \begin{bmatrix} k_{tg1} & 0 \\ 0 & k_{tg2} \end{bmatrix} \qquad C_{sg1} = \begin{bmatrix} k_{sg11}\varphi''_{11}(l_{11}) & k_{sg11}\varphi''_{12}(l_{11})\ldots \\ k_{sg12}\varphi''_{11}(l_{12}) & k_{sg12}\varphi''_{12}(l_{12})\ldots \end{bmatrix}$$

The output matrix, C_{sg2}, is defined in a similar manner. The constants k_{tgj} and k_{sgji} are constants characterizing the tachometers and the strain gauge, whereas l_{ji} are the location (no i) on the links (link no. j).

The control objective is to control the end point position as well as the velocity of the end point. The controlled quantities are related to the system state according to:

$$y_c(t) = C_c x(t) \qquad C_c = \begin{bmatrix} C_p & 0 \\ 0 & C_p \end{bmatrix} \qquad C_p = \begin{bmatrix} 1 & 1 & \frac{\varphi_{11}(L_1)}{L_1} \cdots \frac{\varphi_{1n}(L_1)}{L_1} & \frac{\varphi_{21}(L_2)}{L_2} \cdots \frac{\varphi_{2n}(L_2)}{L_2} \end{bmatrix}$$

This clearly shows the dependency of the link angles, θ_{b1} and θ_{b2}, the deflection of the beams and their velocities.

19.4.3 FEM Model

In the previous section a modal presentation of the flexible robot arm was presented. The displacement of the link was described by a weighted infinite sum series of mode shape functions defined on the entire special domain and subsequently, the system was modally truncated to obtain a finite order model. Instead of this modal discretization and a subsequent reduction to a certain number of modes, the Finite Element Method (FEM) makes use of a spatial discrete model by cutting the robot arm into a finite number of elements and assume

that the arm displacement for each element can be represented by a certain specified well-known type of function. The model can be made arbitrarily accurate by choosing a suitable number of elements and suitable placements of the corresponding nodal points.

The robot arm is formulated in spatial discrete model by dividing it into $n-1$ elements which results in n nodal points as illustrated in Figure 4.18. The n nodal points are arbitrarily placed along the link. However, the first nodal point is fixed because it has to separate the link from the hub and the final point is fixed too at the tip to separate a possible payload from the link.

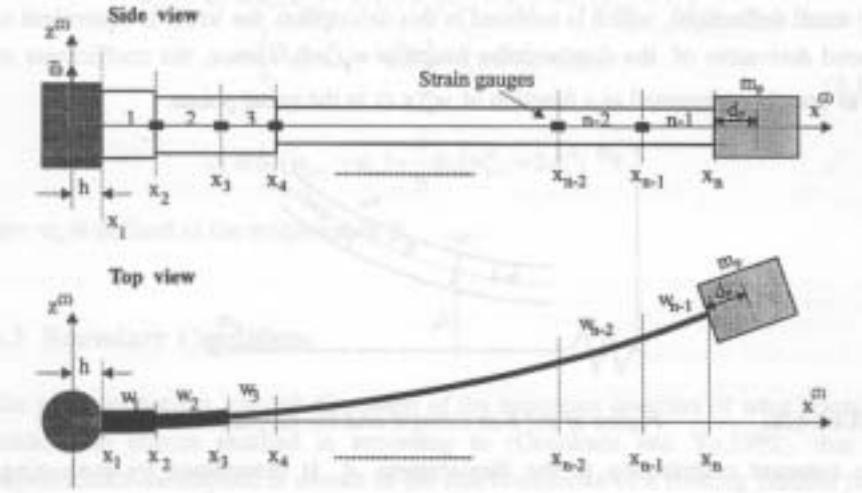


Figure 4.18: Segmentation of the flexible robot link in $n-1$ elements. (From Baungaard, 1996)

The displacement function of the k'th element $w_k(x,t)$ defined in the interval $x_k \le x < x_{k+1}$ is an approximated cubic spline function, which is the must often function approach for one degree of freedom flexible structures. Furthermore, the degree of the chosen polynomial is the lowest possible to choose for the purpose of describing the vibrations in the link while these are described by the forth order Euler-Bernoulli equation

$$w_k(x,t) = a_k(x-x_k)^3 + b_k(x-x_k)^2 + c_k(x-x_k) + d_k \quad \text{for} \quad x_k \le x < x_{k+1}$$

x_k is the position of the k'th nodal point. The time dependence is embedded in the coefficients which is related to the curvature, w_k'', and the displacement, w_k in the nodal points through

$$\begin{aligned} a_k &= \tfrac{1}{6}\,\beta_k(w''_{k+1} - w''_k) \\ b_k &= \tfrac{1}{2}\,w''_k \\ c_k &= \beta_k(w_{k+1} - w_k) - \tfrac{1}{6}\,\alpha_k(w''_{k+1} - w''_k) \\ d_k &= w_k \end{aligned}$$

Let $\overline{w} = (w_1, \ldots w_k, \ldots w_n)^T$ and $\overline{w}'' = (w_1'', \ldots w_k'', \ldots w_n'')^T$. Due to the internal smoothness it is possible to express the displacements $\overline{w}$ as a simple function,

$$\overline{w} = N\overline{w}''$$

of the curvature, $\overline{w}''$.Consequently it is a natural choice (several possible) to use the curvature $\overline{w}''$ and the angular position θ_{b1} as state variable, i.e.

$$q = \begin{bmatrix} \theta_{b1} \\ \overline{w}'' \end{bmatrix}$$

It is possible to derive the following description of the system

$$T = M\ddot{q} + D\dot{q} + Kq \tag{19.9}$$

where

$$M = \begin{bmatrix} J_t & \gamma^T \\ \gamma & M_b \end{bmatrix} \qquad K = \begin{bmatrix} 0 & 0 \\ 0 & K_b \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & \zeta I \end{bmatrix}$$

Here M is the mass/inertia matrix, D is the system damping and K the stiffness. The external forces are due to the actuator and

$$T = \begin{bmatrix} T_{ext} \\ 0 \end{bmatrix}$$

In this section we will only consider the 1 DOF version of the robot. If we combine the FEM model for the flexible link described in the previous section with the actuator model (see section 19.4.1) we can obtain a state space model for the flexible link robot.

Here we use the full actuator model, which can be written as

$$T_{act} = -T_b = \frac{k_1}{k_2}u + \frac{k_3}{k_2}\dot{\theta}_b - \frac{1}{k_2}\ddot{\theta}$$

If this equation is combined with the FEM model we obtain the following model

$$0 = M^*\ddot{q} + D^*\dot{q} + Kq + Vu + G(q)$$

where

$$M^* = \begin{bmatrix} J_t + \frac{1}{k_2} & \gamma^T \\ \gamma & M_b \end{bmatrix} \qquad D^* = \begin{bmatrix} -\frac{k_3}{k_2} & 0 \\ 0 & K_b \end{bmatrix} \qquad V = \begin{bmatrix} -\frac{k_1}{k_2} \\ 0 \end{bmatrix}$$

If the following definition of the states

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

is introduced, the linear state space model

$$\dot{x} = Ax + Bu$$

is obtained, where

$$A = \begin{bmatrix} 0 & I \\ -(M^*)^{-1}K & -(M^*)^{-1}D^* \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ -(M^*)^{-1}V \end{bmatrix}$$

For the FEM model the measured quantities are tacho measurement and the strain gauge voltage, i.e.

$$y_m(t) = C_m x(t) \qquad C_m = \begin{bmatrix} K_{tg} & 0 & 0 \\ 0 & K_{sg} & 0 \end{bmatrix}$$

The position and the velocity of the end point, which are the controlled variable, are given as

$$y_c(t) = C_c x(t) \qquad C_c = \begin{bmatrix} C_p & 0 \\ 0 & C_p \end{bmatrix} \qquad C_p = \begin{bmatrix} 1 & \underline{0} & \dfrac{1}{L+h}N \end{bmatrix}$$

and are influenced by the link angle and the curvature of the beam.

19.5 Control design

The controllers used in this workbench are standard discrete time LQG controllers eventually augmented with an integral state. The designs of the controllers are based on a linear model (linearized in a specific point of operation), which is sampled. The measurement system (tacho and strain gauge sensors) is introduced in order to obtain reasonable values for the system state. In order to reduce the effect from the measurement noise the state are estimated via a Kalman filter. The estimated states are used in the controller and are fed into the actuator(s).

The state estimator is based on a stationary version of the Kalman filter for either the fem model or the modal model. Assume that A_d and B_d are system matrices in the discrete time description of the system. The stationary Kalman filter consists of two sets of recursions. It consists of the time update

$$\hat{x}(t+1 \mid t) = A_d \hat{x}(t \mid t) + B_d u(t)$$

and on the data update

$$\hat{x}(t \mid t) = \hat{x}(t \mid t-1) + K_{est}\left[y(t) - C_m \hat{x}(t \mid t-1)\right]$$

Here K_{est} is determined as the optimal gain (and from system information such as system matrices and variance of process and measurement noise).

The controllers, which are implemented in the workbench, are variants of LQG controllers. They exist in a standard version and in a version which includes an integral action in order to cancel stationary errors, which might occur due to model mismatch or non-zero DC components in the disturbances. In this case the state vector is augmented with an integral state.

Let $\tilde{x}_i = x_i - x_0$ be the deviation away from the stationary point equivalent to reference (or set point). In order to introduce the integral action the state vector is augmented with the integral state z_i. The total design model can then be stated as:

$$\begin{bmatrix} \tilde{x} \\ z \end{bmatrix}_{i+1} = \begin{bmatrix} A_d & 0 \\ -C_c & 1 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ z \end{bmatrix}_i + \begin{bmatrix} B_d \\ 0 \end{bmatrix} u_i \qquad \tilde{y}_i = \begin{bmatrix} C_c & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ z \end{bmatrix}_i$$

The objective is to control the end point position and its velocity with due respect to the control effort. In the LQG framework this is formulated as a minimization of the performance index:

$$J = \sum_{i=0}^{\infty} \|\tilde{y}\|_Q^2 + \|u_i\|_R^2$$

where the weight matrices Q and R should compromise between performance and control effort. In the work bench the performance weight matrix

$$Q = \begin{bmatrix} q_p & 0 & 0 \\ 0 & q_v & 0 \\ 0 & 0 & q_i \end{bmatrix}$$

which contains the weights q_p, q_v and q_i accounting for the error in position, velocity and integral state, respectively. The result is a well-known feedback

$$u_i = -K_c \tilde{x}_i - K_z z$$

from the deviation away from the stationary point and from the integral state.

19.6 CACE Environment

The main panel of the CACE environment used for controlling the flexible link robot is shown in figure 19.19. It is programmed using the graphical interface in MATLAB and uses the MATLAB Control System Toolbox functions for computing the controllers from the appropriate model description. The main panel has 5 sections, of which 2 is related to modelling and design. One section is related to specification of the reference signal. In the last 2 sections it is possible to impose different plotting and other types of commands.

The models can, as previously mentioned, be based on a modal description (section 19.4.2) or on a FEM description (section 19.4.3). The positions of the strain gauges can be specified as well as which of them are currently used. The payload and sampling interval is also specified in this section of the main panel. In the design part of the main panel the weights q_p, q_v and q_i are entered as well as a switch indication whatever the integral action is required. The reference signal is specified and the corresponding Simulink model is generated and opened. A plot menu exists for controlling which plots should be made based on the simulation results from the simulation. The unified plot menu ensures that the resulting plots from the simulation and the experiments are easily comparable.

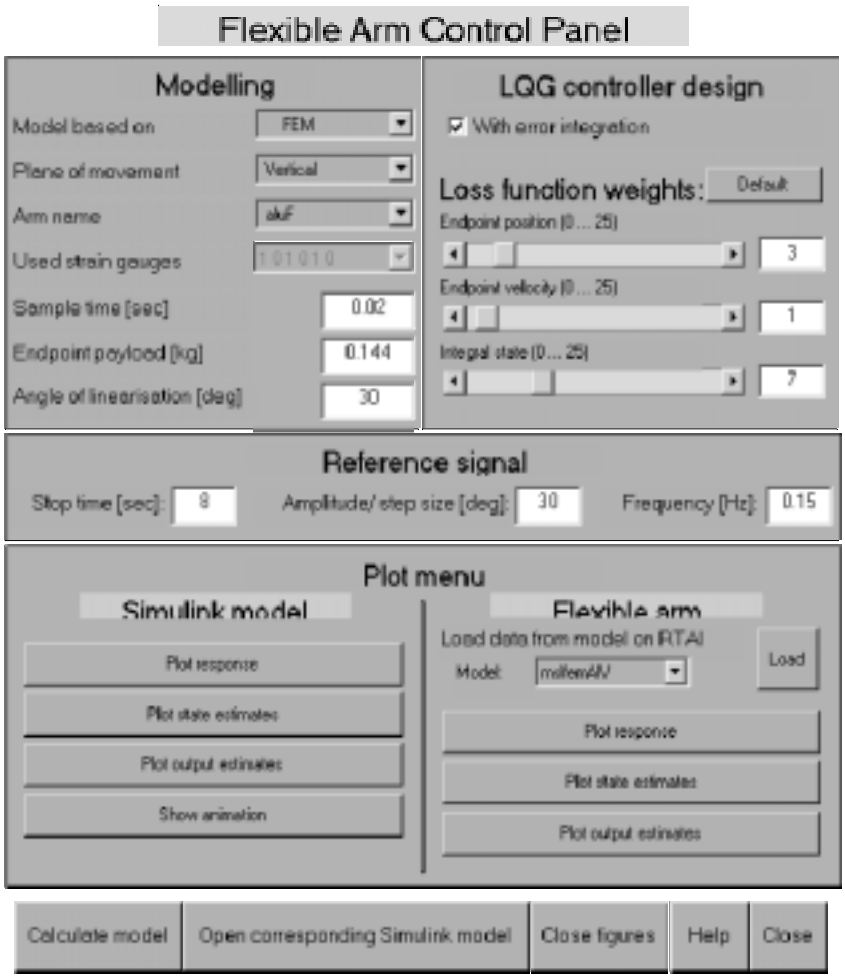


Figure 19.19 Main panel for the software system

The corresponding Simulink model is shown in Figure 19.20. Notice, the two blocks realizing the flexible link system. One block, *mlsflexo,* is the simulation model (shown in Figure 19.21) and another block, *flexrt,* is the real time system interface (shown In Figure 19.22). Switching the two blocks convert the Simulink system from a simulation model to a system from which real time code can be generated using Real Time Workshop (RTW). The main advantage of this approach is that the same blocks and codes are used for simulation and for real-time experiments. Furthermore the transition to real-time code is automated eliminating the source of errors that manual translation means and enhancing the design process, as it is easy to go back and forth between the simulation and experimentation phases. Safety is also addressed, as it is simple to test new controllers by simulation before applying to the real-time experiments.

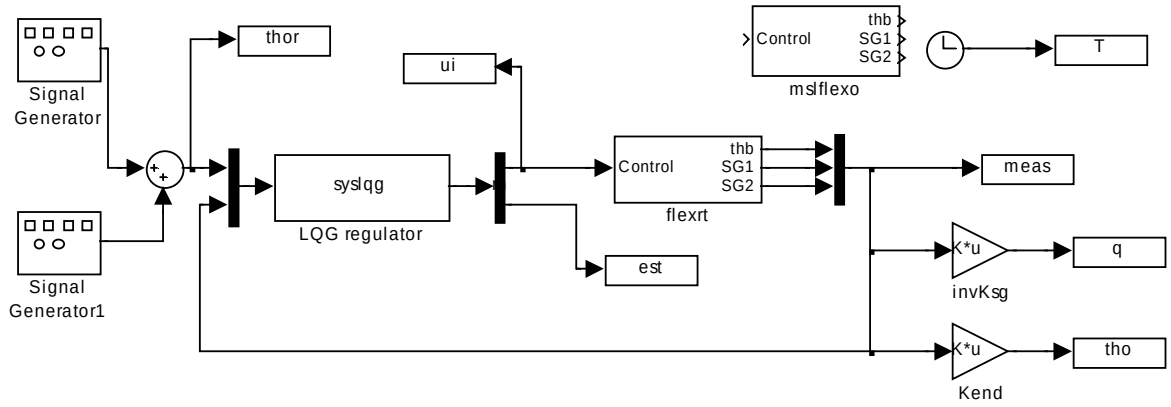


Figure 19.20 Simulink diagram of the flexible link system. Notice, the two blocks (flexrt and mslflexo) realizing the flexible link system.

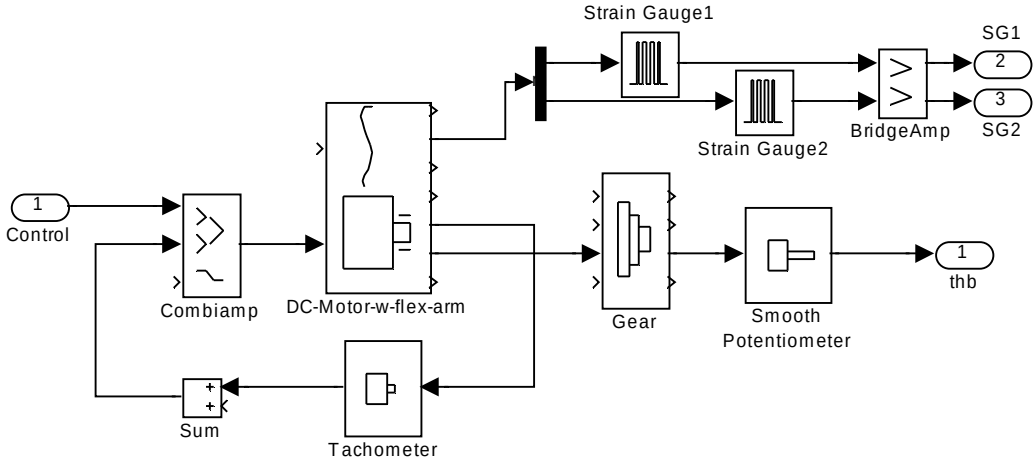


Figure 19.21 The block realizing the simulation version of the flexible link system.

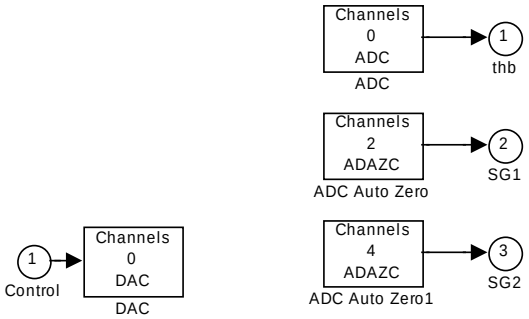


Figure 19.22 The block realizing the real time version of the flexible link system. Notice, the interface to the experimental laboratorial rig.

The real time code is automatically generated, compiled and downloaded to the real time system connected to the actual physical model through process interface boards. The computer executing the real time code is an ordinary PC running RTAI - the Realtime Linux Application Interface for Linux. [Quaranta, G. and Mantegazza, P (2001)]: The data collected during the experiment is automatically collected and can be plotted from the main window menu. The system has proven quite flexible and easy to use especially for no experienced users and students.

The controller module used in the work bench (see Figure 19.20) is based on the LQG block of SIMULINK. The main challenge is to convert the system model into a form suitable for calculating the parameter of the LQG controller using the *dlqr* command in MATLAB. The first step in this direction is to choose between the full simulation model or a reduced order version. This model is then linearized in a specific point of operation and sampled. Based on this description the Kalman filter and the feed back control law is determined using the *dlqe* and the *dlqr* commands. The controller module realize a state space representation of the combined Kalman filter and feed back control law.

19.7 Conclusions

This chapter focuses on design software for modelling and control using an experimental platform. The modelling is carried out for both the horizontal and the vertical case. An LQG control design that suits both types of models is designed and analysed. The aim of the control is to dampen the vibrations of the arm as well as position the end-point of the flexible link according to the reference signal. With regards to the vertical movement the controller must allow and compensate for the static deflection and the curvatures the links have in a given angle position. At the same time the controller must damp the vibrations and position the end point accurately. The model and the control strategy depend on the mass of a pay-load at the end point. Methods for estimation of the payload are examined. These methods are incorporated into the control algorithm.
The requirements for a computer aided control system design environment are outlined and a model of the design process described. Furthermore a block library for Simulink (MSL) is described enabling simulations and experiments with scalable granularity in the simulations.
The controllers are implemented and illustrated within simulations and experimental set-ups, using an environment based on Matlab/Simulink using Real Time Workshop (RTW) to generate real-time code for a real-time (RTAI) Linux based platform.

19.9 References

Barker, H. A., Jobling, C. P., Ravn, O., and Szymkat, M. (1993). A requirements analysis of future environments for computer aided control engineering. In *Proceedings of the 12th IFAC World Conference*, Sydney, Australia.

Baungaard, J.R. (1996): Modelling and Control of Flexible Robot Links. Ph.D. thesis. Ph.D. thesis. Department of Automation, The Technical University of Denmark.

Caspersen, Morten Keller (2000): Control of a Flexible Link Robot. Master Thesis, Department of Automation, The Technical University of Denmark.

Kruise L. (1990): Modelling and control of a flexible Beam and Robot Arm. Ph.D. Thesis, University of Twente

Luca, A.D. and Panzieri, S. (1994): An iterative scheme for learning gravity compensations in flexible robot arms. *Automatica*, **30**(6), 993-1002.

MacFarlane, A. G. J., Grübel, G., and Ackermann, J. (1989). Future design environments for control engineering. *Automatica*, **25**(2):165–176.

M'Saad, M, Dugard, L. and Hammand, S. (1993): A suitable generalized predictive adaptive controller case study of a flexible arm. *Automatica*, **29**(3), 589-608.

Quaranta, G. and Mantegazza, P (2001): Using MATLAB-Simulink RTW to Build Real Time Control Applications in User Space with RTAI-LXRT, Realtime Linux Workshop Milano 2001

Ravn, O. and Szymkat, M. (1992). The evolution of CACSD tools — a software engineering perspective. In *Proceedings of IEEE Control Systems Society Symposium on Computer-Aided Control System Design*, pages 225–231, Napa, CA, USA.

Ravn, O. and Szymkat, M. (1994). Requirements for user interaction support in future CACE environments. In *Proceedings of IEEE Control Systems Society Symposium on Computer-Aided Control System Design*, Tucson, Arizona, USA.

Ravn, O. and Szymkat, M. (1995). Mechatronics approach to robotic modelling. software engineering perspective. In *Proceeding of the 2nd International Symposium on Methods and Models in Automation and Robotics — MMAR'95*, Miedzyzdroje, Poland. Plenary Paper.

Ravn, O., Szymkat, M., Uhl, T., Betemps, M., Pjetursson, A., and Rod, J. (1996). Mechatronic blockset for simulink, - concept and implementation. In *Proceedings of CACSD'96*, Dearborn, MI, USA.

Rostgaard M (1995): Modelling, Estimation and Control of Fast Sampled Dynamic Systems. Ph.D. theses, Department of Mathematical Modelling, The Technical University of Denmark

Sakawa, Y., Matsuno F. and Fukushima S. (1985): Modelling and control of a Flexible arm. Journal of Robotic Systems, 2, 453-472.

Szymkat, M., Ravn, O., Turnau, A., Kolek, K., and Pjetursson, A. (1995). Integrated mechatronic modelling environments. In *International Conference on Recent Advances in Mechatronics*, pages vol. II pp. 767–772, Istanbul, Turkey.

Timoshenko, S., Young, D.H. and Weaver, J.W. (1974). Vibration problems in Engineering. John Wiley and Sons.

.