6

Mechanics in Hardware Description Languages

6.1 Introduction

The objective of this section is to highlight the most important strategies for obtaining the equations of motion for mechanical components and systems and to clarify the options for their subsequent representation in hardware description languages. Both direct formulations of symbolic equations and indirect formulations based upon the parametric calculation of the system matrices will be considered. The latter is often also called the solution using numerical equations.

The use of hardware description languages for the modelling of mechanics also implies that the solution of the mechanical equation takes place using the solver of the circuit simulator. Naturally, solvers are generally optimised for various domains. For electronics the focus tends to be upon the management of a large number of degrees of freedom, whereas in mechanics numerical problems with a large number of constraints require particular attention. On the other hand, the example of the classical multibody simulator ADAMS shows that this contrast is not irreconcilable, see Orlandea *et al*. [304] and [305]. The numerics of ADAMS is largely based upon procedures that are also used in circuit simulation. We should mention at this point that the equation system is not formulated using a minimum number of equations according to the degrees of freedom. Rather, each individual equation is entered into an overall system. Thus the resulting system matrix is sparse and can be processed using 'sparse matrix' techniques. The numerical integration takes place using the Gear procedure that is also commonly used in circuit simulation.

In mechanics — as in electronics — we can differentiate between various abstractions. Multibody mechanics and continuum mechanics are examples. According to Schiehlen [361] a multibody system is characterised as a collection of rigid and/or elastic bodies with inertia as well as springs, dampers and servo motors without inertia. These are connected together by rigid bearings, joints or suspensions. Friction and contact forces can also be included if necessary. This corresponds with

Mechatronic Systems Georg Pelz
 ISBN: 0-470-84979-7

modelling using concentrated parameters and is thus comparable with a circuit made up of components.

In some cases the abstraction of a continuum to the discrete elements of a multibody system is not suitable for the solving of the envisaged problem. For example, this is the case if the exact deformation of an elastic body contributes significantly to the system behaviour. In this case models need to be created and formulated in hardware description languages that adequately describe the continuum with its distributed parameters. Both multibody mechanics and continuum mechanics will be considered in the following.

6.2 Multibody Mechanics

6.2.1 Introduction

When modelling multibody mechanics using hardware description languages, we first have to raise the question of the perspective from which the system is to be considered. One option focuses upon the system level, the other upon the component level with the system models being generated by connecting component models together. The first method is called system-oriented modelling and the second method object-oriented modelling.

The first option has the advantage that equations of motion can be created using standard engineering methods. Furthermore, we have access to a greater system knowledge during the modelling, which can be beneficial. However, one problem is that this type of consideration opposes one of the most important basic philosophies for the development of electronics. In this field it is generally sufficient to model only the fundamental components and to develop complex systems from these. Further modelling is normally not necessary during the development of electronic systems.

As an alternative to this we can use object-oriented modelling to describe the standard components — for example bodies, springs, dampers, joints, etc. — and put these submodels together into a system model. Information about this system, such as, for example, a favourable selection of generalised coordinates, is in principle not available and thus cannot be used for the simplification or acceleration of the model. However, the building of a system model can be considerably simplified if the basic models that are required are available.

In the following we will consider how it is possible to obtain equations of motion for multibody systems, see for example, Dankert and Dankert [79], Greenwood [125], Hiller [144] or Nikravesh [299] for the basic principles shown. Multibody systems typically include the following components:

- Particles with translational inertia.
- Rigid bodies with translational and rotational inertia.
- Suspensions and joints that limit the movement of individual particles and bodies in relation to one another.
- Coupling elements, e.g. springs, dampers, servo motors, etc.

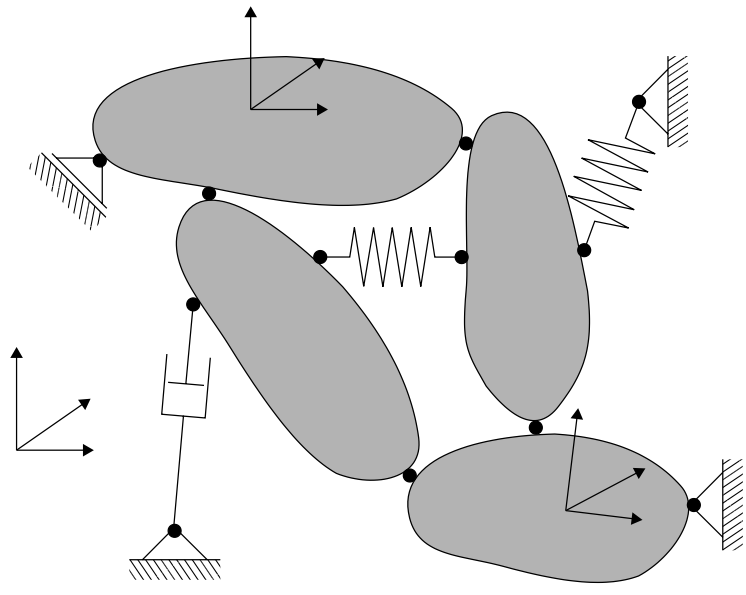

Figure 6.1 Multibody system with four bodies, springs, dampers, suspensions, joints, and inertial and body-related frames of reference

In the consideration of the structure of a multibody system, an abstracted description such as that given in Figure 6.1 is generally sufficient. Decisive factors are the topography of the system and the parameters of the individual elements, such as mass, centre of gravity, moments of inertia with respect to the main axes or the point of application of forces.

For the consideration of point-shaped masses we start from Newton's second law, which identifies the product of mass m and acceleration in the x, y, and z direction a_x, a_y, a_z of a particle with the forces F_x, F_y, F_z acting upon it:

$$F_x = ma_x, \qquad F_y = ma_y, \qquad F_z = ma_z \tag{6.1}$$

Let us now consider a system of N particles. These may be subject to additional limitations to their movement, so-called constraints. This state of affairs can be taken into account by the introduction of the so-called reaction forces, which ensure that the constraints are adhered to. The total force acting upon a body is divided into two components, the force applied from outside F_i^e and the reaction force F_i^r. In total this yields the following equation system:

$$\begin{aligned} m_i a_{ix} &= F_{ix}^e + F_{ix}^r \\ m_i a_{iy} &= F_{iy}^e + F_{iy}^r \qquad (i = 1, 2, \ldots, N) \\ m_i a_{iz} &= F_{iz}^e + F_{iz}^r \end{aligned} \tag{6.2}$$

This can be formulated as a vector equation as follows:

$$m_i \mathbf{a_i} = \mathbf{F_i^e} + \mathbf{F_i^r} \tag{6.3}$$

For the sake of simplicity we can also describe the cartesian coordinates of the first body by (x_1, x_2, x_3), those of the second body by (x_4, x_5, x_6) and so on. If we also term the masses of the k^{th} body $m_{3k-2} = m_{3k-1} = m_{3k}$ and set $a_i = \ddot{x}_i$ for the accelerations, then the equations of motion can be formulated by the following set of equations:

$$m_i \ddot{x}_i = F_i^e + F_i^r \qquad (i = 1, 2, \ldots, 3N) \tag{6.4}$$

If the movement of the particle is not restricted then the reaction forces are negligible. This yields a system of 3N second-order differential equations, which is generally nonlinear. This equation system can in general only be solved numerically, i.e. as part of a simulation.

The constraints between the particles are characterised by a set of M independent constraint equations:

$$f_j(x_1, x_2, \ldots, x_{3N}, t) = 0 \qquad (j = 1, \ldots, M) \tag{6.5}$$

So $3N + M$ equations are available for the solution of the same amount of variables.

However, the use of cartesian coordinates is not always favourable. In many cases cylindrical, spherical, elliptical, parabolic or other coordinates are beneficial. For this reason we will now move to the so-called, generalised coordinates $q_1, \ldots, q_n$. These permit a formulation that is better suited to the problem. Furthermore, under certain conditions the generalised coordinates can be selected so that the constraint equations are dispensed with completely, considerably simplifying the drawing up and calculation of the equations of motion. This is possible if all constraints are holonomous, i.e. they relate exclusively to the possible geometric positions of the bodies or can at least be put into such a form. Regardless of the selection of coordinates, the number of degrees of freedom of the system in principle remains constant. It corresponds with the number of independent coordinates minus the number of independent constraint equations.

A small example, see Greenwood [125], should clarify the relationship between cartesian and generalised coordinates, see Figure 6.2.

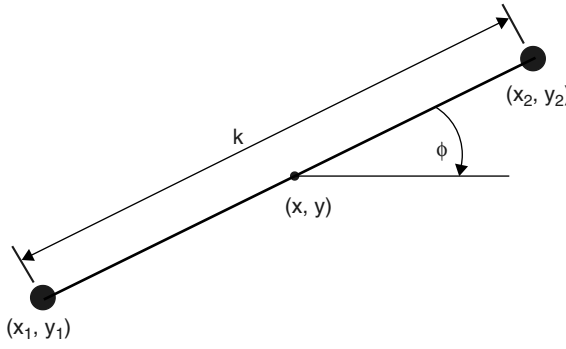

Figure 6.2 Description of the position of two particles joined by a mass-free rod

Two mass points in the plane are rigidly joined together by a mass-free rod. Their position is determined by two pairs of cartesian coordinates (x_1, y_1) and (x_2, y_2). The condition induced by the rod can be described by the following equation

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 - k^2 = 0 \tag{6.6}$$

We therefore have four cartesian coordinates and a bond equation, thus a total of three degrees of freedom. In principle, however, the configuration of the two particles can be described by the following generalised coordinates:

$$\begin{aligned} q_1 &= x \quad \text{coordinate of the mid-point of the rod} \\ q_2 &= y \quad \text{coordinate of the mid-point of the rod} \\ q_3 &= \text{angle } \phi \text{ of the rod.} \end{aligned}$$

The fourth coordinate q_4 would be the length of the rod, which is, however, constant. So the associated bond equation

$$q_4 = k \tag{6.7}$$

is trivial and can be disregarded. There thus remain three coordinates without a further bond equation, i.e. three degrees of freedom. Only by this formulation in generalised coordinates can we thus omit the consideration of constraint equations in holonomous systems. In the following we will consider exclusively holonomous systems.

It is often worthwhile going over to the generalised coordinates, which — as is the case for pure cartesian coordinates — represent the configuration of the system, i.e. the position of all particles. Coordinate transformations permit the conversion between generalised and cartesian coordinates:

$$\begin{aligned} x_1 &= x_1(q_1, q_2, \ldots, q_n, t) \\ x_2 &= x_2(q_1, q_2, \ldots, q_n, t) \\ &\vdots \\ x_{3N} &= x_{3N}(q_1, q_2, \ldots, q_n, t) \end{aligned} \tag{6.8}$$

The transition to generalised coordinates requires that forces acting upon the system from outside are also present in generalised form. Whereas the forces in cartesian coordinates can be simply split up into their x, y and z components, things are more complicated in this case. For example, forces acting upon angular coordinates become moments. The conversion rule for the generalised force Q_i is naturally also based upon the coordinate transformation x_j and looks like this:

$$Q_i = \sum_{j=1}^{3N} F_j \frac{\partial x_j}{\partial q_i} \qquad (i = 1, 2, \ldots, n) \tag{6.9}$$

If we now move on from particles to rigid bodies we now have to consider the moments of inertia in addition to the translational inertia. These are described by the underlying Euler equations for the rigid body K_j:

$$\begin{aligned} I_{j1}\dot{\omega}_{j1} - (I_{j2} - I_{j3})\omega_{j2}\omega_{j3} &= M^e_{j1} + M^r_{j1} \\ I_{j2}\dot{\omega}_{j2} - (I_{j3} - I_{j1})\omega_{j3}\omega_{j1} &= M^e_{j2} + M^r_{j2} \\ I_{j3}\dot{\omega}_{j3} - (I_{j1} - I_{j2})\omega_{j1}\omega_{j2} &= M^e_{j3} + M^r_{j3} \end{aligned} \tag{6.10}$$

In matrix form these equations look like this:

$$\mathbf{I_j}\dot{\boldsymbol{\omega}}_\mathbf{j} + \boldsymbol{\omega}_\mathbf{j} \times (\mathbf{I_j}\boldsymbol{\omega}_\mathbf{j}) = \mathbf{M^e_j} + \mathbf{M^r_j} \tag{6.11}$$

where $\mathbf{M}_j$ represents the applied and reactive torque vectors, $\mathbf{I}_j$ represents the tensors of the moment of inertia and $\boldsymbol{\omega}_i$ represents the angular velocities with respect to the three principal axes of the rigid body K_j.

6.2.2 System-oriented modelling

In system-oriented modelling two classical approaches can be distinguished, the synthetic and the analytical, see for example, Kreuzer [207]. In the synthetic methods we first draw up the Newton and Euler equations for each body. The connections between bodies, e.g. joints, give rise to constraining forces, and the elimination of these converts the Newton/Euler equations into equations of motion. The analytical approach, on the other hand, is associated with the name Lagrange and starts from an energy formulation. This is rearranged directly into equations of motion without the constraining forces being considered.

Both approaches will be described in the following in a formulation using generalised coordinates. In addition to the above-mentioned approaches there is also a range of further options, which are briefly described and compared by Kane and Levinson in [177]. It should not go unmentioned that the equations that result from the various approaches are ultimately the same. However, they are obtained at a different level of complexity. The formulation is also of varying suitability for the subsequent numerical simulation.

Newton–Euler approach

The Newton–Euler approach, see also Kreuzer and Schiehlen [208], should — just like the Lagrange approach described subsequently — be represented in a formulation using the generalised coordinates $q_1, \ldots, q_n$. From these the velocities should be determined for each body K_j in the x, y and z coordinates:

$$\begin{bmatrix} v_{xj} \\ v_{yj} \\ v_{zj} \end{bmatrix} = \begin{bmatrix} \frac{\partial x_j}{\partial q_1} & \frac{\partial x_j}{\partial q_2} & \cdot & \cdot & \frac{\partial x_j}{\partial q_n} \\ \frac{\partial y_j}{\partial q_1} & \frac{\partial y_j}{\partial q_2} & \cdot & \cdot & \frac{\partial y_j}{\partial q_n} \\ \frac{\partial z_j}{\partial q_1} & \frac{\partial z_j}{\partial q_2} & \cdot & \cdot & \frac{\partial z_j}{\partial q_n} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \cdot \\ \cdot \\ \dot{q}_n \end{bmatrix} + \begin{bmatrix} \overline{v}_{xj} \\ \overline{v}_{yj} \\ \overline{v}_{zj} \end{bmatrix} \tag{6.12}$$

In more compact form the same situation can be formulated as follows

$$\mathbf{v_j} = \mathbf{J_{Tj}}\dot{\mathbf{q}} + \overline{\mathbf{v}}_\mathbf{j} \tag{6.13}$$

where both the translational Jacobi matrix $\mathbf{J}_{Tj}$, and the local velocities $\overline{\mathbf{v}}_\mathbf{j}$ depend only upon q and t. The cartesian accelerations $\mathbf{a}_j$ of the j^{th} body are calculated as follows:

$$\mathbf{a_j} = \mathbf{J_{Tj}}\ddot{\mathbf{q}} + \overline{\mathbf{a}}_\mathbf{j} \tag{6.14}$$

For the local accelerations $\overline{\mathbf{a}}_\mathbf{j}$ it is again true that they depend only upon q and t.

In a similar manner we move from the generalised coordinates to the angular velocities $\boldsymbol{\omega}_j$ and angular accelerations $\boldsymbol{\alpha}_j$ of each rigid body K_j:

$$\boldsymbol{\omega_j} = \mathbf{J_{Rj}}\dot{\mathbf{q}} + \overline{\boldsymbol{\omega}}_\mathbf{j} \tag{6.15}$$

$$\boldsymbol{\alpha_j} = \mathbf{J_{Rj}}\ddot{\mathbf{q}} + \overline{\boldsymbol{\alpha}}_\mathbf{j} \tag{6.16}$$

where the rotational Jacobi matrix $\mathbf{J}_{Rj}$ and the local angular velocities $\overline{\boldsymbol{\omega}}_\mathbf{j}$ and angular accelerations $\overline{\boldsymbol{\alpha}}_\mathbf{j}$ again depend exclusively upon q and t.

In the next step we draw upon Newton and Euler equations for each body:

$$m_i\mathbf{a_i} = \mathbf{F_i^e} + \mathbf{F_i^r} \tag{6.17}$$

$$\mathbf{I_j}\dot{\boldsymbol{\omega}}_\mathbf{j} + \boldsymbol{\omega_j} \times (\mathbf{I_j}\boldsymbol{\omega}_j) = \mathbf{M_j^e} + \mathbf{M_j^r} \tag{6.18}$$

Using the transformation of generalised coordinates into (angular) velocities and (angular) accelerations of the individual body, as described above, these equations can now be formulated exclusively in the form of generalised coordinates. However, these are the same for all bodies, which means that the bodies can be linked together in this manner. The resulting system of equations takes the following form:

$$\overline{\mathbf{M}}\overline{\mathbf{J}}\ddot{\mathbf{q}} + \overline{\mathbf{k}} = \overline{\mathbf{p}}^\mathbf{e} + \overline{\mathbf{p}}^\mathbf{r} \tag{6.19}$$

Where the 6k × 6k matrix $\overline{\mathbf{M}}$ takes the form

$$\overline{\mathbf{M}} = \mathrm{diag}(m_1\mathbf{E}, \ldots, m_k\mathbf{E}, \mathbf{I_1}, \ldots, \mathbf{I_k}) \tag{6.20}$$

and forms a block diagonal matrix of masses and inertia tensors. The k denotes the number of bodies. The 6k × n matrix $\overline{\mathbf{J}}$ is the global Jacobi matrix and consists of a stack of k translational and k rotational 3 × n Jacobi matrices of the individual bodies, where n is the number of generalised coordinates:

$$\overline{\mathbf{J}} = [\mathbf{J_{T1}^T}|\cdots|\mathbf{J_{Tk}^T}|\mathbf{J_{R1}^T}|\cdots|\mathbf{J_{Rk}^T}]^T \tag{6.21}$$

Again $\overline{\mathbf{k}}$ is the 6k × 1 vector of gyroscopic and centrifugal forces as well as Coriolis forces. Finally, the applied forces and moments and the reaction forces and moments are located in the 6k × 1 vectors $\overline{\mathbf{p}}^{\mathbf{e}}$ and $\overline{\mathbf{p}}^{\mathbf{r}}$:

$$\overline{\mathbf{p}}^{\mathbf{e}} = [\mathbf{F_1^{eT}}|\cdots|\mathbf{F_k^{eT}}|\mathbf{M_1^{eT}}|\cdots|\mathbf{M_k^{eT}}]^T \tag{6.22}$$

$$\overline{\mathbf{p}}^{\mathbf{r}} = [\mathbf{F_1^{rT}}|\cdots|\mathbf{F_k^{rT}}|\mathbf{M_1^{rT}}|\cdots|\mathbf{M_k^{rT}}]^T \tag{6.23}$$

Finally, we multiply the equation system (6.19) from the left with the transposed, global Jacobi matrix $\overline{\mathbf{J}}^{\mathbf{T}}$, so that it is formulated completely in generalised coordinates. This yields equilibrium of forces in matrix form:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{k} = \mathbf{Q} \tag{6.24}$$

The product $\overline{\mathbf{J}}^{\mathbf{T}}\overline{\mathbf{M}}\overline{\mathbf{J}}$ yields the mass matrix $\mathbf{M}$. Similarly, $\overline{\mathbf{J}}^{\mathbf{T}}\overline{\mathbf{k}}$ yields $\mathbf{k}$, the vector of the generalised gyroscopic forces, and $\overline{\mathbf{J}}^{\mathbf{T}}\mathbf{F}^{\mathbf{e}}$ yields the vector of the generalised forces $\mathbf{Q}$. Here $\mathbf{M}$ is dependent upon the generalised coordinates $\mathbf{q}$ and t, and $\mathbf{k}$ and $\mathbf{Q}$ are dependent upon $\mathbf{q}$, $\dot{\mathbf{q}}$ and t. Last but not least, we should note that the reaction forces are dispensed with as a result of the multiplication by $\overline{\mathbf{J}}^{\mathbf{T}}$. We therefore have a system of ordinary differential equations to solve because the algebraic equations of the constraints have disappeared with the reaction forces.

Lagrange approach

The focus of the Newton–Euler approach described in the previous section was the drawing up of Newton and Euler equations for each body and the conversion of the resulting overall system into generalised coordinates so that the constraint equations are dispensed with. The Lagrange approach takes a different and particularly elegant route. It starts from the premise that the generalised inertial forces and the generalised applied forces cancel each other out. For the formulation of the generalised inertial forces Q_i^I we require the total kinetic energy T of the system, which of course must also be formulated in the form of generalised coordinates:

$$Q_i^I = \frac{\partial T}{\partial q_i} - \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) \tag{6.25}$$

The first subterm represents the generalised inertial forces that arise as a result of the change of position of the system, thus, for example, the Coriolis force. Opposing this part is the second component, which describes the rate of change of the generalised impulses. The above-mentioned premise

$$Q_i^I = -Q_i \tag{6.26}$$

yields the Lagrange equation

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} = Q_i \qquad (i = 1, 2, \ldots, n) \tag{6.27}$$

By drawing up a formula for kinetic energy and the conversion of applied forces into their generalised form we can obtain the equations of motion directly by substituting into equation (6.27). This is particularly simple because kinetic energy is a scalar that contains no higher derivatives with respect to time than the velocities. These are significantly easier to determine than the accelerations.

Formulation in hardware description languages

The primary purpose of analogue hardware description languages is for the modelling of analogue electronic components for a circuit simulator. The variables considered in this application — voltage and current — correspond with the duality of a potential and a flow and can be represented by other quantities in accordance with the analogies described in Section 3.2.2. Although the text-based formulation of the mechanical model is based upon accelerations, velocities, positions, and forces, the underlying calculations take place in accordance with the analogies of the potentials and flows available.

Furthermore, the preceding section has shown that the selection of the considered unknowns of multi-body mechanics is attributed decisive importance. In electronics the unknowns are normally in the form of node voltages, which is because of the nodal analysis that is prevalent in circuit simulation. In the system-oriented modelling of mechanics, on the other hand, it is of decisive importance to specify a suitable set of generalised coordinates. For holonomous systems, which can be described using generalised coordinates, the — sometimes very complex — constraint equations are dispensed with. As was shown by the relatively simple example from Figure 6.2, it is not a question of selecting from a fund of existing coordinates, but one of an independent engineering task.

The methods described supply sets of ordinary differential equations in symbolic form. These can easily be formulated in analogue hardware description languages. This is true under the prerequisite that the size of the equation set remains within limits. In Section 7.2.3 the obtaining and formatting of the equations of motion for an automotive wheel suspension system using the Lagrange approach is illustrated.

6.2.3 Object-oriented modelling

Introduction

The use of generalised coordinates in object-oriented modelling raises two problems. Firstly, it poses the question of how we should determine the generalised coordinates from the very limited perspective of an element. Secondly, the local Jacobi matrices, which describe how the local coordinates arise from the totality of the generalised coordinates, have to be set up. Both questions necessitate the global perspective of mechanics, i.e. the local consideration that has brought so many benefits in electronics is lost. In other words: When generalised coordinates are used the consideration of a multibody system generally results in the completion of the drawing up of the Newton–Euler equation system

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{k} = \mathbf{Q} \tag{6.28}$$

using hardware description languages, based upon models for rigid bodies, springs, dampers etc. A more promising approach would seem to be to add the automated creation of symbolic equations of motion by a suitable programme and thus select system-oriented modelling.

Object-oriented modelling thus cannot be performed directly using generalised coordinates. However, if we free ourselves from the generalised coordinates and in particular permit a greater number of unknowns, then the question is reformulated. The work of Suescun *et al*. [391] provides a first approach to the modelling of multidimensional mechanics in hardware description languages (VHDL-AMS). Here the position of the body is given in natural coordinates, which occur in two forms: Firstly, they are given as cartesian coordinates for certain points on the body. These marked points may be contact points of joints, springs and dampers. Secondly, unit vectors are introduced as natural coordinates, in order to specify axes of rotation. According to Suescun *et al*. the mass matrix $\mathbf{M}$ of a body is constant if a sufficient quantity of natural coordinates are considered. This represents the vector $\mathbf{q}$ of the natural coordinates on the inertial force $\mathbf{Q}_\mathrm{I}$ (with respect to the natural coordinates):

$$\mathbf{Q}_\mathbf{I} = -\mathbf{M}\ddot{\mathbf{q}} \tag{6.29}$$

The natural coordinates are modelled in the hardware description languages as potentials (across), the forces and moments as flows (through). In addition there are algebraic constraint equations in quadratic (for planar mechanics) and cubic (for 3D mechanics) form, which hold constant the constellation of points in relation to each other and the length of the unit vectors. In addition there is a VHDL-AMS module for the gravitation that is suspended on the rigid body model. Also present are models for joints, springs and dampers. The models mentioned are put together using a circuit editor. The corresponding system of DAE is then solved in a circuit simulator. Finally, we should also mention that no simulation results are shown in [391].

In addition to the general principle there is one special case, for which the development of arbitrarily connectable models has been known for a long time. The prerequisite for this is that the movement in the system only takes place in one translational or rotational dimension, or that the movements in the system can be broken down into one-dimensional movements that are independent of each other. Then the generalised coordinates coincide with the cartesian coordinates or the angular coordinates, the Jacobi matrices are trivial and the mechanical forces/moments and velocities/angular velocities can be represented directly by potentials and flows. Applications are any pure translational movements and one-dimensional rotational movements, such as in a drive train with motor, gearbox and mechanical load. In the following suitable models for the basic elements mass, spring, damper, power source and position source[1] will be described.

Basic model

The basic elements mass, spring, and damper can be formulated for both translational and rotational movements. In the following the translational version will be given, with positions (instead of the velocities) being used as potentials. It is formulated in the hardware description language VHDL-AMS. We first begin with the model of mass inertia for translational movements `inertia_trans`, see Hardware description 6.1. This model follows the equation

$$F = -(m \cdot \ddot{x}) - (m \cdot g) \tag{6.30}$$

and thus describes both the inertia and also the acceleration due to gravity. If the acceleration due to gravity does not lie in the direction of the translation, the parameter `GRAVITY`, i.e. g, is set to 0. Otherwise the model follows the convention that an acceleration in the direction of x gives rise to negative forces and vice versa.

```
LIBRARY disciplines;          -- Reference to a package with the
USE disciplines.Kinematic_system.all; -- mechanics declarations
ENTITY inertia_trans IS                -- Interface description
  GENERIC (m, g: REAL);                        -- Mass, gravity
  PORT    (TERMINAL p, n: kinematic);             -- Terminals
end inertia_trans;
ARCHITECTURE simple OF inertia_trans IS          -- Architecture
-- Declaration of potential/flow quantity=deflection x/force F ...
 QUANTITY tdisp ACROSS tforce THROUGH p TO n;
BEGIN
  tforce == -(m*tdisp'DOT'DOT) - (m*g);        -- Basic equation
END simple;
```

Hardware description 6.1 Model of a mass for translational movement in VHDL-AMS

[1] Like a voltage or current source, but supplying a position.

Now to spring and damper models. For the spring model the applied force is dependent upon the position, i.e. upon the distortion of the spring. The damping force, on the other hand, is proportional to the relative velocity of the two terminals of the damper model, and thus primarily describes the Stokes' friction of a viscous fluid, such as for example in an automotive shock absorber. The following equations form the basis:

$$F_{spring} = -k(x_p - x_n - l_0)$$
$$F_{damper} = -b(v_p - v_n) \tag{6.31}$$

The appropriate conversion is found in Hardware descriptions 6.2 and 6.3.

```
LIBRARY disciplines;        -- Reference to a package with the
USE disciplines.Kinematic_system.all;-- mechanics declarations
ENTITY spring_trans IS                -- Interface description
 GENERIC (k, l0: REAL);-- Spring constant, basic spring length
 PORT (TERMINAL p, n: kinematic);                 -- Terminals
end spring_trans;
ARCHITECTURE simple OF spring_trans IS         -- Architecture
       -- Declaration of potential/flow = deflection/force ...
  QUANTITY tdisp ACROSS tforce THROUGH p TO n;
BEGIN
  tforce == -k * (tdisp - l0);               -- Basic equation
END simple;
```

Hardware description 6.2 Spring model for translational movements

In both cases the spring or the damping force is first calculated and correspondingly applied. This force is applied in the negative direction. For the spring this is consistent with the convention that positive forces increase the current positional value. The spring force at terminal p is oriented such that the spring length tends towards the equilibrium l_0. At terminal n the force is correspondingly oriented in the opposite direction. For the damper, the convention applies that positive forces increase the relative distance of the two position terminals. The damping force resists a positive, relative velocity. The descriptions for the application of forces and velocities will not be illustrated here. They correspond with the applicable descriptions of sources for currents and voltages.

```
LIBRARY disciplines          -- Reference to a package with the
USE disciplines.Kinematic_system.all; -- mechanics declarations
ENTITY damper_trans IS                -- Interface description
 GENERIC (b: REAL);                        -- Damper constant
 PORT (TERMINAL p, n: kinematic);                -- Terminals
end damper_trans;
ARCHITECTURE simple OF damper_trans IS -- Architecture 'simple'
        -- Declaration of potential/flow = deflection/force ...
  QUANTITY tdisp ACROSS tforce THROUGH p TO n;
```

```
BEGIN
  tforce == -b * tdisp'DOT;                    -- Basic equation
END simple;
```

Hardware description 6.3 Damper model for translational movements

6.2.4 Example: wheel suspension

Starting from these basic models we can now put together more complex models. A wheel suspension will serve as an example. Let us first of all set up the framework for the consideration of a modelling process. We assume that only the vertical movement of the wheel and the vehicle body is to be considered. Furthermore, the condition is imposed that the centre of gravity of the vehicle is located mainly in the centre of the vehicle and thus the axles are uniformly loaded. In this case the movements of the axles are almost independent of each other, which means that we can restrict ourselves to the consideration of one axle. If we further assume that the road conditions are the same for the left and right-hand wheel, then for reasons of symmetry it is completely adequate to consider only one wheel including half an axle and a quarter of the car body. Using the assumptions described yields a two-mass oscillator, which describes the vertical dynamics very well, see Figure 6.3.

The mass m_a describes the wheel and the associated part of the axle and m_b describes a quarter of the body. Both masses are of course subject to gravity, but also to the forces that are exerted by the adjacent springs and dampers. Shock absorbers and body springs themselves are characterised by the parameters b and k_s respectively. The tyres can also be considered as springs, but with a spring constant k_w that lies around an order of magnitude above that of the body spring. The

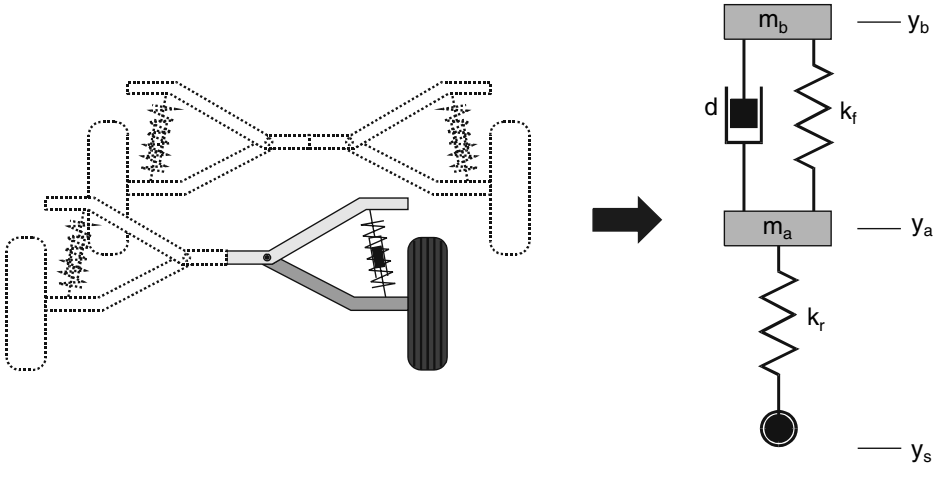

Figure 6.3 Modelling of a wheel suspension by a two-mass oscillator

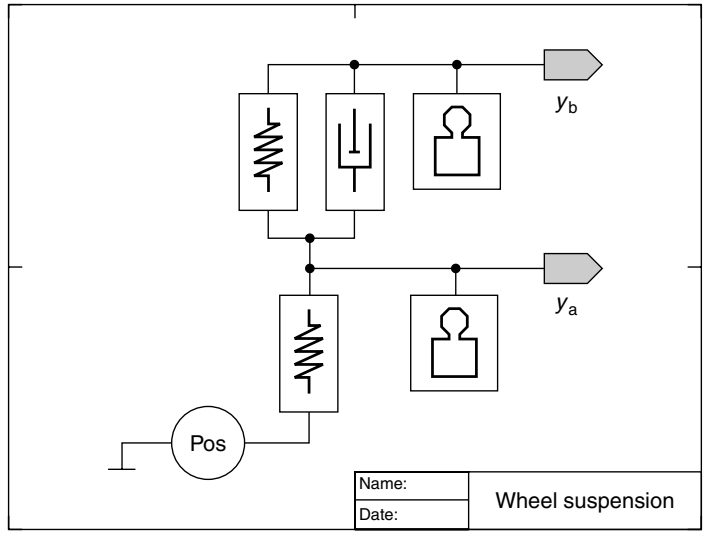

Figure 6.4 Schematic diagram of a wheel suspension

Table 6.1 Parameters

Parameter	Value
m_a	50 kg
m_b	250 kg
k_s	25 500 N/m
k_w	250 000 N/m
B	2000 Ns/m
$l_{0,Spring}$	0.2 m
$l_{0,Tyre}$	0.03 m
G	9.81 m/s^2

damping effect of the tyres can be disregarded here. The system is one-dimensional because only the vertical movement of the masses is being considered. It has two degrees of freedom, the y-positions of the two masses. The y-position of the road serves as the stimulation. Driving over a step of a few centimetres is modelled by imposing a jump of corresponding height. This system can be assembled directly from the basic elements developed above in the form of a schematic diagram, see Figure 6.4.

After suitable parameterisation, simulation can take place without further modelling expense. The parameters in Table 6.1 are used for the simulation shown in Figure 6.5.

The situation considered in the simulation corresponds with driving over a 5 cm high step at a right angle, i.e. the left-hand and right-hand wheel experience the same deflection. Thus the symmetry condition is fulfilled. At the beginning of the journey the spring forces of springs and tyres correspond with the respective weight

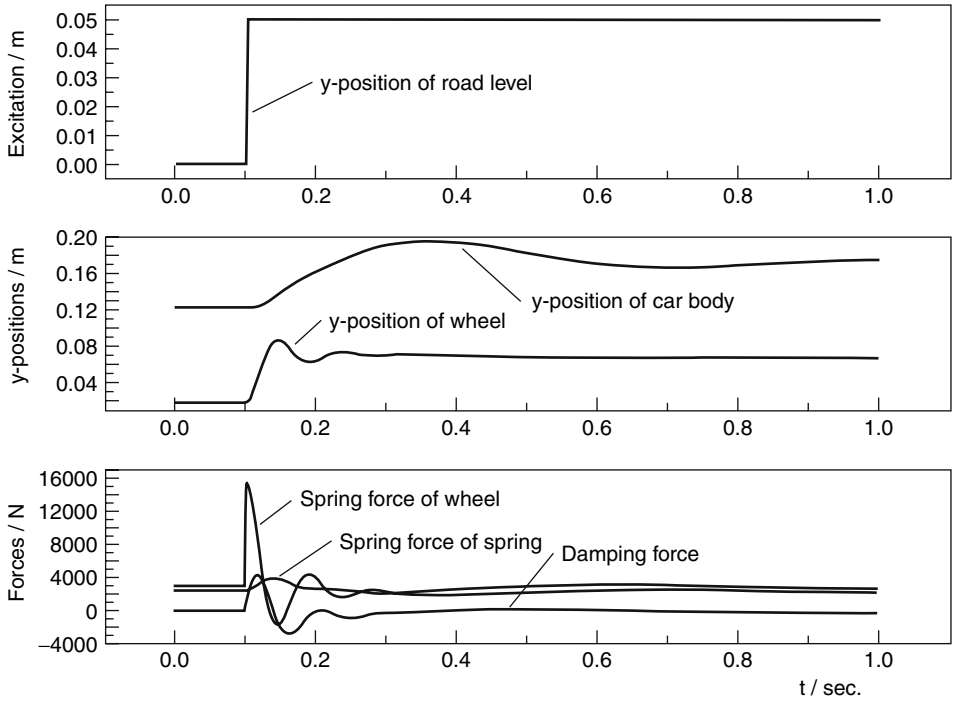

Figure 6.5 Simulation of a wheel suspension

and ensure that body and wheel dip in relation to gravity. Whilst the road level y_s rises suddenly by 5 cm, the tyre springs are correspondingly compressed and quickly build up a force of approximately 15 000 N. As a result, the wheel is pushed upwards, the tyre spring relaxes and the force in question eases. However, the body spring is compressed by the movement of the wheel and corresponding forces are transmitted to the body. In accordance with the mass and spring constants, vibrations are observed in the range of around one hertz. In the case of the wheel the vibrations lie in the range of around ten hertz. Thus the body requires significantly longer to take on its new y-position. Finally, it should be noted that the damping force works to counter the relative movement between wheel and body and thus allows the vibrations to decay. The simulation requires few CPU seconds on a SUN Sparc 20 workstation.

6.2.5 Further applications

Introduction

In the following a few other applications will be presented as examples, thereby illustrating the possibilities of multibody modelling using hardware description languages. The representation takes into account both mechatronic and micromechatronic systems.

Mechatronics

In [254] Makki *et al*. describe an electronically controlled window winder mechanism for cars. On the mechanical side a direct current motor, a gearbox, a rack for the conversion of the rotational motion into a translational movement, a mechanical load — the window pane — and a mechanical stop are envisaged. In addition to this there is a force sensor that allows the drive to be switched off in the event of large counterforces. This typically corresponds with a situation in which objects are trapped by the window-pane whilst the window is raised. In this case movement is restricted to a rotary — or after the rack a translational — dimension. For this reason the system described can be simply assembled from basic models, each of which corresponds with one of the named components.

Other examples can be found in Donnelly *et al*. [84], who describe an electronically controlled hydraulic braking system, or in Mikkola [269], who uses hardware description languages to model and simulate diesel-electric ship drives.

Micromechatronics

For the class of so-called 'suspended' MEMS, Mukherjee and Fedder [282] have developed an approach based upon multibody mechanics. Classical applications for this approach are, for example, seismically suspended masses of acceleration sensors and resonators, see Figure 6.6. The structure of interest is broken down into individual parts such as springs, masses, dampers, etc., for which models are available. Thus a micromechanical model can be assembled from the basic models. This strategy is very well suited to the approach that is also selected here of formulation in hardware description languages, because these continuously support the hierarchical structure of models.

In the NODAS system in [103], Fedder and Jing go beyond multibody systems made up of rigid bodies by including elastic components on the basis of hardware description languages. The following components have been implemented in NODAS as described in [103]: A bending beam, a rigid plate, an electrostatic

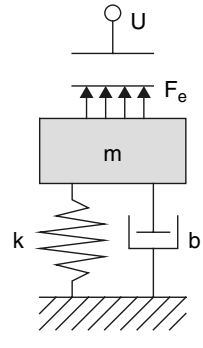

Figure 6.6 Electrically excited resonator in the form of a multibody system

comb actuator, and an anchor (which corresponds with a fixed suspension). Implementation first raises the question of differentiating between a global and local coordinate system. Initially all considerations of an element are local. However, the element can also be given global coordinates, which can be used to solve a calculation of the operating point. Thus the correct values of the global coordinates are set automatically, whereas the actual calculations generally take place using a further set of variables that only give values relative to the operating point.

The model of the bending beam was developed on the basis of a mechanical structural analysis. The equation for the beam takes the form:

$$\mathbf{F_{beam}} = \mathbf{M\ddot{u}} + \mathbf{B\dot{u}} + \mathbf{Ku} \tag{6.32}$$

where $\mathbf{F}_{\text{beam}}$ represents the vector of forces and moments at the beam, **u** represents the vector of the translational and rotational degrees of freedom of the beam, **M** represents the mass matrix, **B** the damping matrix and **K** the stiffness matrix. In principle this follows the beams presented in [34] and in Section 6.3.2, although NODAS is more interested in the global movement and not in the deformation of a continuum. Furthermore, the local mass, damping and stiffness matrices are formulated directly in the hardware description language, which may cause these symbolic equations to explode in the event of more complex elements.

However, in many cases physical modelling, as used in the previous examples, is not possible or would be associated with great expense. In such cases it is often worthwhile to move to experimental modelling. In this approach an experiment does not necessarily consist of measurements on a real system, but often consists of field and continuum simulations, for example based upon finite elements. In this manner, the simulation can be run in advance of manufacture by the use of experimental models. Pure table models, such as for example in Romanowicz *et al*. [350] or Swart *et al*. [394], are an example. However, these table models, with their data list of identification pairs, can be represented by compression into relatively simple equations. This is shown by Teegarten *et al*. [397], who also supply a lovely example of the mixing of physical and experimental modelling based upon a micromechanical gyroscope.

6.3 Continuum Mechanics

6.3.1 Introduction

The previous section dealt with multibody mechanics, the main characteristic of which is the consideration of a collection of bodies connected together by joints and suspensions. The validity of this abstraction depends upon the formulation of the question. In particular, the bending of mechanical components is often not an undesirable side-effect, but is essential to the functioning of the system. Now, if the form of bending plays a significant role in the system behaviour then we

cannot avoid the consideration of the continuum in the modelling. The associated mechanics, and in particular its representation in hardware description languages, are the subject of this section.

We can initially differentiate between whether the consideration is to be performed statically or dynamically. For the static case each mechanical position may be assigned an electrical quantity. Here only the steady state is considered. In the dynamic case velocities and accelerations of mechanical quantities also play a role, so that phenomena such as mechanical resonance are also considered. A further distinction is supplied by the selection of a desired level of abstraction. It is a fundamental truth of continuum mechanics that elasticity and mass spread continuously, thus giving rise to an infinite number of degrees of freedom. As is described in more detail in what follows, we can perform the modelling of mechanical continua on the basis of (geometric) structure, physical equations, and experimental data. In this context the reader is referred to a corresponding classification of modelling approaches in Section 2.4.

6.3.2 Structural modelling

Introduction

Structural modelling traces the generation of a model back to the composition of basic models. In the case of continuum mechanics these basic models may be finite elements, for example. Due to the generality and high degree of adoption of finite elements in the framework of structural modelling, we will deal exclusively with this approach. Bathe [19], Gasch and Knothe [113] and Knothe and Wessels [202] supply a good overview of the methods of finite elements in their works. For the modelling of finite elements, as in the Ritz procedure (see within Section 6.3.3), we work on the basis of interpolation functions. However, these are not formulated globally for the whole structure here, but locally for the finite element. Thus the main difficulties of the Ritz procedure are removed. If the models of the finite elements are available, modelling is a purely geometric task, which primarily represents a breakdown of the continuum. In this context we also speak of a meshing, in which finer resolutions buy more precision at the expense of greater simulation time.

Up until now, finite elements have typically only been used to investigate the component level, disregarding the system context. The following sections will show that finite elements can be drawn into a circuit simulation on the basis of hardware description languages. As we will show in what follows, the differential equation solver of the circuit simulator in question is thus entrusted with the calculation of the equations of the finite elements. The dynamic coupling of electronics and mechanics then takes place automatically. Overall, this opens up a simpler, faster and more secure way of modelling mechanical continua that is compatible with hardware description languages and thus also with circuit simulation.

Finite elements

In principle, finite elements can be used in many fields of engineering science. Our discussion is based upon the field of structural mechanics. Thus the following quantities have to be linked together: displacements, forces, strain, and applied loads, which act as a trigger here. Depending upon the application, different finite elements are used, which vary in structure, number of nodes and degrees of freedom. Figure 6.7 shows a selection of finite elements of structural mechanics.

The degrees of freedom of the finite elements can be of both a translational (u_x, u_y, u_z) and a rotational (r_x, r_y, r_z) nature. The numerical complexity of the calculation increases with their number. Fundamentally, the element selected should fulfil the question formulated with as few superfluous degrees of freedom as possible. In addition, symmetry considerations are used to keep the number of finite elements as low as possible.

In the following, an approach will be presented that allows the finite elements of structural mechanics to be represented in hardware description languages. This is based upon the work of Pelz *et al*. [333] and Bielefeld *et al*. [33] and [34]. Information on the mechanical foundation can be found in [19], [113] or [202]. We should mention at this point that Haase *et al*. [131] to some degree follow a similar approach in a subsequent work by linking system matrices that originate from a commercial FE simulator into a circuit simulator.

The formulation of the finite elements firstly requires that a mass matrix, a stiffness matrix, and a load vector are generated for each element. As in many other works the damping matrix is initially disregarded. Secondly, in a FE simulator, these element matrices are combined into a global equation system, according to

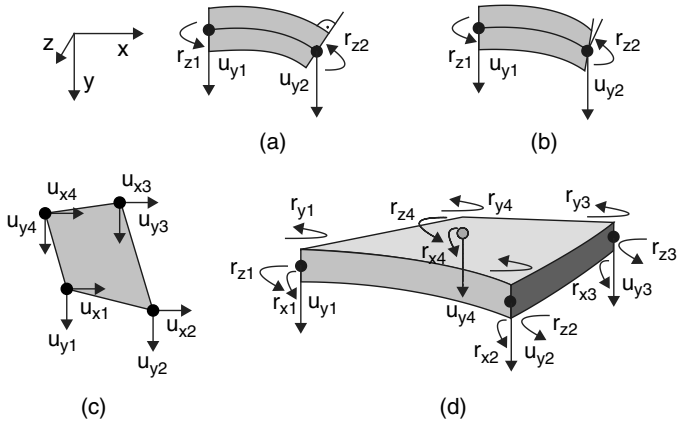

Figure 6.7 Selection of finite elements from structural mechanics: (a) Shear-resistant 1-D beam, two nodes, two degrees of freedom per node (u_y, r_z) (b) Non shear-resistant 1-D, two nodes, two degrees of freedom per node (u_y, r_z) (c) Plane element, four nodes, two degrees of freedom per nodes (u_x, u_y) (d) Plate element, four nodes, four degrees of freedom per node (u_y, r_x, r_y, r_z)

the structure of the mechanics. This must either be completed during the modelling or in the circuit simulator.

A sensible starting point in the drawing up of the element matrices is the principle of virtual displacement. A virtual displacement is a small displacement superimposed upon the actual displacement, which fulfils the geometric boundary conditions and otherwise brings about no gaps or overlapping of the continuum.

The principle of virtual displacements demands that the virtual displacement energy is equal to the virtual work of the external forces for each permitted virtual displacement. This yields the basic equation that is drawn up for the whole continuum. Now the components of the individual elements in the basic equation should be taken into account. This would require knowledge of the continuous displacements over the entire element. However, because we want to operate using only the displacements of the nodes of the finite elements, it is necessary to approximate the continuous displacements from the node displacements. This is done with the aid of interpolation functions that are often created in the form of polynomials. Thus the continuous displacements are approximated from the node displacements, and using the displacement/strain relationship these are transformed into the strains of the element. Using the underlying law of matter we find the stresses from the strains. Using the quantities determined in this manner, the strain energy can be integrated over the element range and summed over all elements. The integration is significantly simplified by the use of interpolation functions, which — as noted before — typically are polynomial.

By contrast, the virtual work of the external forces is based upon the excitation forces, stresses at the edge of the body, and body forces such as weight. The associated proportions of (virtual) work are again calculated from the node displacements by integration over the range in question and summed for all elements. Finally, the total virtual strain energy is equated to the total virtual work of the external forces. In the static case this yields the following equation system:

$$\mathbf{Ku} = \mathbf{p} \tag{6.33}$$

where $\mathbf{K}$ represents the system stiffness matrix, $\mathbf{u}$ the node displacements, and $\mathbf{p}$ the converted body and contact forces at the nodes. The system stiffness matrix is found from the suitable addition of the element stiffness matrices. In the kinetic case there are also inertia forces and the equation is formulated as follows:

$$\mathbf{M\ddot{u}} + \mathbf{Ku} = \mathbf{p} \tag{6.34}$$

where $\mathbf{M}$ represents the system mass matrix, which, in a similar way to the system stiffness matrix, is found by a suitable summing of the element mass matrices. The system mass matrix is linked with the accelerations of the displacements. In this discussion both equation systems correspond with the equilibrium principle.

If we want to represent finite elements in hardware description languages, then it initially appears logical to first draw up the differential equation system resulting

from the collection of finite elements in symbolic form, and then to directly formulate this in a hardware description language. In theory this is correct. However, the handling of the equations causes massive problems. This is firstly the case if we want to parameterise the elements geometrically and not on the basis of the entries in the element matrix. The same applies in the nonlinear consideration if the mass and stiffness matrices of the finite elements are dependent upon the current state of deformation and have to be drawn up afresh depending upon deflection. In both cases the complete rule for the creation of the element matrices must be included in the equation system, as must the conversion from the element matrices into the system matrix. This allows the volume of equations to explode and the resulting equation system is thus beyond any meaningful calculation.

It therefore makes sense to initially consider the finite elements individually and to build the rule for the creation of the element matrices into the model in question. This could, for example, be achieved by embedding a C routine, capable of generating suitable element matrices as required, into the model. This corresponds with the numerical simulation of multibody systems. The question is also raised of how to move from element behaviour to system behaviour. Ideally, the system behaviour would be found by composing the finite elements in a circuit simulator. This first requires a link between electronic quantities and mechanical degrees of freedom. Here mechanical deflections are represented by electrical potentials and mechanical forces and moments by electrical currents. The linking of two finite elements effects a scleronomic[2] constraint between the element degrees of freedom in question, and thus the amalgamation of the degrees of freedom in question to a single system degree of freedom. This is the expected behaviour for voltages and positions. Furthermore, the currents are added at these points, as is also expected of forces and moments.

The generation of element matrices

In what follows the example of the shear-resistant beam element will be used to demonstrate how such a finite element can be formulated in hardware description languages. To achieve this two main problems have to be solved. Firstly the mass and stiffness matrices in question have to be generated. Secondly the element matrices have to be transformed into the system matrices, which represents the behaviour of the entire structure.

The beam element is shown in Figure 6.8 and has two nodes, k and l, each with two degrees of freedom, the deflection in the y-direction u_y and the rotation about the z-axis r_z. Both the shear deflection in the x-direction and the structural damping are disregarded in this model.

The stiffness $B_i = EI_{zz}$ and the mass distribution $\mu_i = \rho A_i$ are assumed to be constant over the length of the beam, where E is the modulus of elasticity, I_{zz} the moment of inertia, ρ the density of the beam material and A_i the cross-section of

[2] Scleronomic constraints are not changeable.

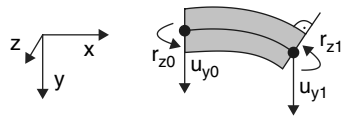

Figure 6.8 Degrees of freedom of the shear-resistant beam element at the nodes k and l: u_y (deflection in y-direction), r_z (rotation about the z-axis)

the beam section i. The beam load is concentrated by p_{i0} and p_{i1} on the nodes 0 and 1 of the beam element. In the shear-resistant case and for small deflections the stiffness matrix $\mathbf{K}_i$, the mass matrix $\mathbf{M}_i$ and the load vector $\mathbf{p}_i$ of the i^{th} beam element are independent of the deflection. If we select the interpolation functions $h_1 \ldots h_4$ in the variables ξ for the approximation of the continuous displacements as follows:

$$\begin{aligned} h_1(\xi) &= 1 - 3\xi^2 + 2\xi^3 \\ h_2(\xi) &= -\xi(1-\xi)^2 l_i \\ h_3(\xi) &= 3\xi^2 - 2\xi^3 \\ h_4(\xi) &= \xi^2(1-\xi) l_i \end{aligned} \tag{6.35}$$

then we find the following element matrices and vectors, see Gasch and Knothe [113]:

$$\begin{aligned} \mathbf{K_i} &= \frac{B_i}{l_i^3} \begin{bmatrix} 12 & -6l_i & -12 & -6l_i \\ -6l & 4l_i^2 & 6l_i & 2l_i^2 \\ -12 & 6l_i & 12 & 6l_i \\ -6l & 2l_i^2 & 6l_i & 4l_i^2 \end{bmatrix} \\ \mathbf{M_i} &= \frac{\mu_i l_i}{420} \begin{bmatrix} 156 & -22l_i & 54 & 13l_i \\ -22l_i & 4l_i^2 & -13l_i & -3l_i^2 \\ 54 & -13l_i & 156 & 22l_i \\ 13l_i & -3l_i^2 & 22l_i & 4l_i^2 \end{bmatrix} \\ \mathbf{p_i} &= p_{i0} l_i \begin{bmatrix} 7/20 \\ -l_i/20 \\ 3/20 \\ l_i/3 \end{bmatrix} + p_{i1} l_i \begin{bmatrix} 3/20 \\ -l_i/30 \\ 7/20 \\ l_i/20 \end{bmatrix} \end{aligned} \tag{6.36}$$

The equation system for such an element thus takes the form:

$$\mathbf{M_i}\ddot{\mathbf{u}}_\mathbf{i} + \mathbf{K_i}\mathbf{u_i} = \mathbf{p_i} \tag{6.37}$$

where

$$\mathbf{u}_i = [u_{y0}, r_{z0}, u_{y1}, r_{z1}]^T$$

where $\mathbf{u}_i$ represents the element displacement vector, and thus the degrees of freedom.

Now, if the behaviour of a mechanical continuum is to be reconstructed in a circuit simulator it is reasonable to keep the modelling close to the actual determination of the simulator. In our case this means that the mechanics model is formulated 'electronically'. For this purpose a network of capacitors, inductors and current sources is drawn up, see Figure 6.9. If we consider the associated admittance matrix we notice that just like the mass and stiffness matrices it is symmetrical and its leading diagonal consists of positive entries.

The task now is to find an LC network, the admittance matrix of which coincides with the mass and stiffness matrix of the mechanics. To a certain degree this corresponds with the drawing up of a type of equivalent circuit. However, we will see later that the formulation in hardware description languages does not rest upon components, but uses the underlying equations. Let us first consider the circuit in Figure 6.9 and draw up Kirchhoff's current law for the four nodes, i.e. four degrees of freedom:

$$\sum_{j=1}^{4} (i_{ij,L} + i_{ij,C}) = i_i \qquad i = 1 \ldots 4, \quad (i_{ii,L} = 0, \quad i_{ii,C} = 0) \tag{6.38}$$

Using the current–voltage relationships this yields the following equations:

$$\sum_{j=1}^{4} \left(\frac{1}{L_{ij}} \int u_{ij}\, dt + C_{ij} \dot{u}_{ij} \right) = i_i \qquad i = 1 \ldots 4, \quad L_{ij} = L_{ji}, \quad C_{ij} = C_{ji} \tag{6.39}$$

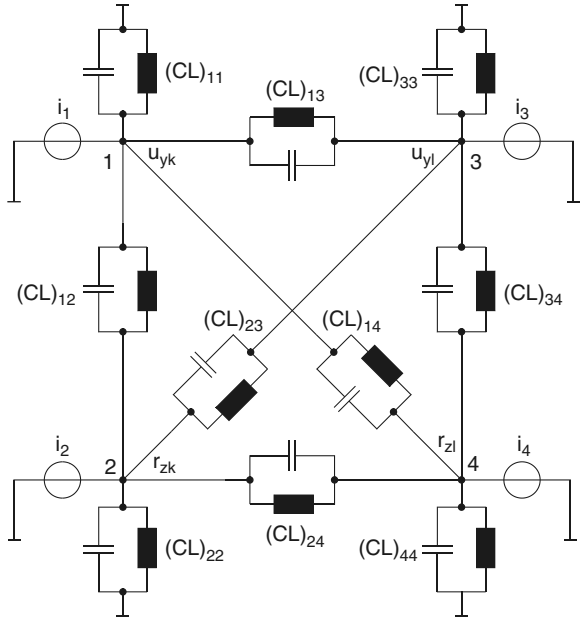

Figure 6.9 LC network with current sources for the modelling of finite beam elements

These four equations are differentiated once with respect to time and then rearranged to give:

$$\sum_{j=1}^{4} C_{ij}\ddot{u}_{ij} + \sum_{j=1}^{4} \frac{1}{L_{ij}} u_{ij} = \dot{i}_i \qquad i = 1 \ldots 4 \tag{6.40}$$

The four degrees of freedom of the beam element u_{yk}, u_{yl}, r_{zk} and r_{zl} should now be represented by the potentials $\varphi_1, \varphi_2, \varphi_3$ and φ_4, which for this reason are used here for the branch voltages u_{ij}. The following is true:

$$\begin{aligned} u_{ij} &= \varphi_i - \varphi_j \qquad (i \neq j) \\ u_{ii} &= \varphi_i \end{aligned} \tag{6.41}$$

Substituting into the above formula yields:

$$C_{ii}\ddot{\varphi}_i + \sum_{i \neq j} C_{ij}(\ddot{\varphi}_i - \ddot{\varphi}_j) + \frac{1}{L_{ii}}\varphi_i + \sum_{i \neq j} \frac{1}{L_{ij}}(\varphi_i - \varphi_j) = \dot{i}_i \qquad i = 1 \ldots 4 \tag{6.42}$$

After rearranging this yields in vector notation:

$$\mathbf{C}\ddot{\varphi} + \mathbf{L}\varphi = \mathbf{i} \tag{6.43}$$

where:

$$\ddot{\boldsymbol{\varphi}} = [\ddot{\varphi}_1, \ddot{\varphi}_2, \ddot{\varphi}_3, \ddot{\varphi}_4]^T$$

$$\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \varphi_3, \varphi_4]^T$$

$$\dot{\mathbf{i}} = [i_1, i_2, i_3, i_4]^T$$

$$\mathbf{C} = \begin{bmatrix} C_{11}+C_{12}+C_{13}+C_{14} & -C_{12} & -C_{13} & -C_{14} \\ -C_{12} & C_{12}+C_{22}+C_{23}+C_{24} & -C_{23} & -C_{24} \\ -C_{13} & -C_{23} & C_{13}+C_{23}+C_{33}+C_{34} & -C_{34} \\ -C_{14} & -C_{24} & -C_{34} & C_{14}+C_{24}+C_{34}+C_{44} \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} \frac{1}{L_{11}}+\frac{1}{L_{12}}+\frac{1}{L_{13}}+\frac{1}{L_{14}} & -\frac{1}{L_{12}} & -\frac{1}{L_{13}} & -\frac{1}{L_{14}} \\ -\frac{1}{L_{12}} & \frac{1}{L_{12}}+\frac{1}{L_{22}}+\frac{1}{L_{23}}+\frac{1}{L_{24}} & -\frac{1}{L_{23}} & -\frac{1}{L_{24}} \\ -\frac{1}{L_{13}} & -\frac{1}{L_{23}} & \frac{1}{L_{13}}+\frac{1}{L_{23}}+\frac{1}{L_{33}}+\frac{1}{L_{34}} & -\frac{1}{L_{34}} \\ -\frac{1}{L_{14}} & -\frac{1}{L_{24}} & -\frac{1}{L_{34}} & \frac{1}{L_{14}}+\frac{1}{L_{24}}+\frac{1}{L_{34}}+\frac{1}{L_{44}} \end{bmatrix}$$

Let us now return to the equations of the i^{th} mechanical, finite beam element:

$$\mathbf{M_i}\ddot{\mathbf{u}}_\mathbf{i} + \mathbf{K_i}\mathbf{u_i} = \mathbf{p_i} \tag{6.37}$$

where

$$\mathbf{u}_i = [u_{y0}, r_{z0}, u_{y1}, r_{z1}]^T$$

This equation system has the same structure as the LC network, see equation (6.43). We now have to identify the individual components of the two matrix equations with each other, i.e.:

$$\begin{aligned} \ddot{\mathbf{u}}_\mathbf{i} &\mathrel{\widehat{=}} \ddot{\boldsymbol{\varphi}} \\ \mathbf{u}_\mathbf{i} &\mathrel{\widehat{=}} \boldsymbol{\varphi} \\ \mathbf{M}_\mathbf{i} &\mathrel{\widehat{=}} \mathbf{C} \\ \mathbf{K}_\mathbf{i} &\mathrel{\widehat{=}} \mathbf{L} \\ \mathbf{p}_\mathbf{i} &\mathrel{\widehat{=}} \dot{\mathbf{i}} \end{aligned} \tag{6.44}$$

The degrees of freedom of the finite beam elements are directly represented by the potentials, i.e. the node voltages. The same applies for the associated accelerations.

In order to balance the matrix entries in question, the negative entries of the mass matrix m_{ij} are used for the capacitance entries in the secondary diagonals, the sum of the involved mass coefficients are used in the leading diagonal:

$$\begin{aligned} C_{ij} &= -m_{ij} \qquad (i \neq j) \\ C_{ii} &= m_{ii} + \sum m_{ij} \quad (i \neq j) \end{aligned} \tag{6.45}$$

In a similar way, the entries for the inductance matrix are formed from the stiffness coefficients k_{ij}:

$$\begin{aligned} L_{ij} &= \frac{1}{k_{ij}} \qquad (i \neq j) \\ L_{ii} &= \frac{1}{k_{ii} + \sum k_{ij}} \qquad (i \neq j) \end{aligned} \tag{6.46}$$

The equations (6.45) and (6.46) ensure that the matrices $\mathbf{M}_i$ and $\mathbf{K}_i$ described by $\mathbf{C}$ and $\mathbf{L}$ are represented with sufficient precision, i.e. there is a good correspondence between equation systems (6.37) and (6.43). Correction terms obtained from the summing term are also added into the leading diagonals of $\mathbf{C}$ and $\mathbf{L}$. These ensure that the LC circuit yielded from the matrices satisfies Kirchhoff's laws and, in particular, that the currents linked by the nodes add up to zero. This corresponds with a variation of the LC branches from the nodes 1 to 4 to the mass, which thus characterises not the relationships between every two degrees of freedom, but only the relationship of the degree of freedom to ground.

Finally, the derivative of the currents $\dot{\mathbf{i}}$ are derived as follows. The loads of the beam element concentrated at the nodes p_{i0} and p_{i1} are converted by equation (6.36) into the element load vector. The components of this are then integrated and, in the form of current, put into the nodes of the associated degree of freedom. This takes place for every time step, so that time-variant loads can also be taken into account.

The finite elements are formulated in the analogue hardware description language MAST of the Saber circuit simulator and this formulation is primarily based

upon introducing two current sources between the nodes i and j for an LC branch, which satisfy the following equations:

$$\begin{aligned} i_{ij,C} &= m_{ij}\dot{u}_{ij} \qquad \text{where } m_{ij} \text{ is from } \mathbf{M} \\ i_{ij,L} &= k_{ij}\int u_{ij}\,dt \qquad \text{where } k_{ij} \text{ is from } \mathbf{K} \end{aligned} \tag{6.47}$$

In addition there are two further current sources for each degree of freedom, which represent the connection to the ground and — as demonstrated above — the external excitations p_{i0} and p_{i1} at each beam element.

Composition of the system matrix

In the previous section an element matrix was put together for the beam element, the four degrees of freedom of which are represented by the potentials at the four terminals of the element. The currents at the nodes in question describe the integral of the associated forces and moments, depending upon whether the degree of freedom is a translational or rotational deflection. In particular, the components of the exciting forces and moments that are assigned to the elements adjoining the nodes are also added to the currents at a node. Thus it is not necessary to explicitly draw up the system matrix. Its solution is found implicitly from the interconnection of the finite elements.

Example: beam with various boundary conditions

Two examples will be considered to illustrate the element model described above, a cantilever beam with and without an additional support point, see Figure 6.10. The second case, in particular, cannot simply be mastered by either analytical equations or finite differences. The beam of length l itself is modelled by 40 finite beam elements. The excitation consists of the pulsed force F_y, which is applied to the beam for eight seconds and then removed again. The outputs are the deflections in the y-direction at $x = 0.25l$, $0.5l$, $0.75l$ and $1.0l$, see Figure 6.11. In the first case there is an oscillation, the amplitude of which is more strongly marked towards the end of the beam, and which is in phase at each point. The additional support in the second case fundamentally alters the behaviour of the beam. Firstly, the natural frequency of the system increases, secondly the node moves downwards at $x = 0.25l$ due to the lever effect of the free end of the beam, although the force is acting upwards. We note that the deflection at $x = 0.5l$ becomes zero.

The same simulation was performed using the ANSYS finite element simulator to verify the results. The differences amount to less than one percent and are in principle attributable to differences in the numerical solution procedure. The simulations were run on a SUN Sparc 20 workstation. The simulation time for the first case amounted to 91 CPU seconds for Saber and 94 CPU seconds for ANSYS.

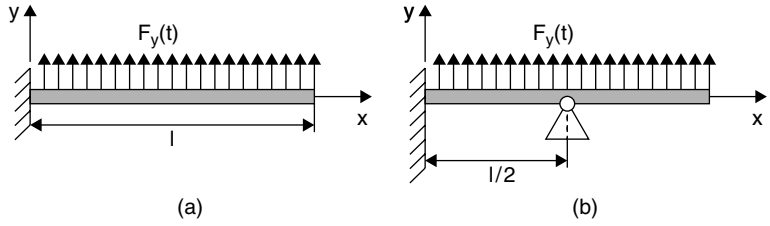

Figure 6.10 Cantilever beam with (a) and without (b) an additional support

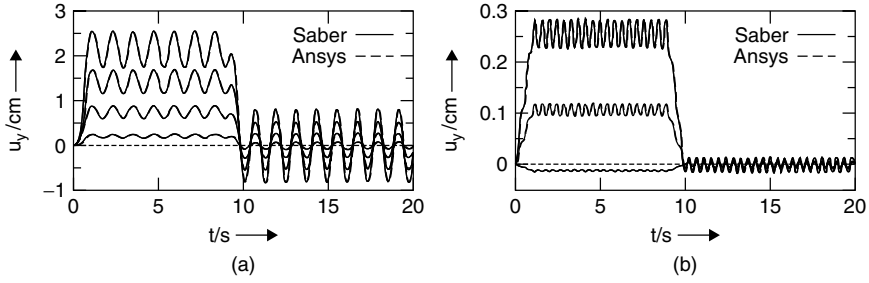

Figure 6.11 Simulation results for the deflection at x = 0.25l, 0.5l, 0.75l and 1.0l of a cantilever beam with (a) and without (b) an additional support

In the second case with the additional support the times are 155 seconds for Saber and 270 seconds for ANSYS.

Based upon the previous example, it was possible to show that finite elements can be formulated in hardware description languages. The same methodology can also be used for the implementation of other finite elements, such as is also shown in Chapter 8. The calculation using the solver of a circuit simulator does not necessarily demand running times that are higher by orders of magnitude. On the other hand, the approach described above does not form a competition to the regular FE-simulators. The main goal of the work described here remains to bring together electronics and mechanics in order to simplify the design of mixed systems.

6.3.3 Physical modelling

Procedures such as the finite element procedure are certainly the most general solution for the envisaged problem. As a result of the high number of degrees of freedom, problems in the simulation speed occasionally occur. In order to achieve improvements here, for certain geometries — for example, round or square plates — we can give formulae that correspond with a physical modelling. The development of such models requires a considerable degree of modelling effort because it calls for an understanding of the physics of the components.

In what follows, four approaches will be considered in this context. The first possibility is to take a partial differential equation for the mechanical continuum

and to represent this using, for example, the method of finite differences on a system of ordinary differential equations, which again can be directly formulated in a hardware description language. The second method relies upon analytical solutions of the partial differential equations in question which are, however, rarely known. Finally, the last two options — the Ritz and Galerkin approaches — attempt to describe bending structures on the basis of a calculus of variations.

Partial differential equations and finite differences

A classical approach to the consideration of the physics of bending structures is to derive a partial differential equation, which can, for example, be represented as a set of ordinary differential equations by the method of finite differences. This step is necessary because analogue hardware description languages cannot in general process partial differential equations directly. The process described was first used by Lee and Wise [224] in order to investigate pressure sensor systems in bulk micromechanics, in which the (quasi-static) solution was built into the respective circuit simulator. In [322], [323] and [324] Pelz *et al*. transferred this solution from the tool level to the model level, where the automatic translation of partial differential equations (in one dimension) into hardware description languages and equivalent Spice net lists was investigated in particular. Consideration was also given to mechanical kinetics. Mrčarica *et al*. [278] also use this approach to consider two-dimensional, partial differential equations, favouring a direct formulation in the in-house hardware description language AleC++. Finally, Klein and Gerlach [195] break up a bending plate into fragments in their approach, and models in an analogue hardware description language are then applied to each of these. These can again be connected to a circuit simulation, thus facilitating the co-simulation of continuum mechanics and electronics. The formulation leads to a system model that is mathematically equivalent to the method of finite differences.

For illustration, the circular plate of a capacitive pressure element will be considered here, see Figure 6.12 and [322], [323] or [324]. A comprehensive description of this example, which will be used frequently in what follows, is found in Section 8.2. The plate is deflected by an external pressure and thus changes the capacitance of the pressure element, which again is detected by a read-out circuit.

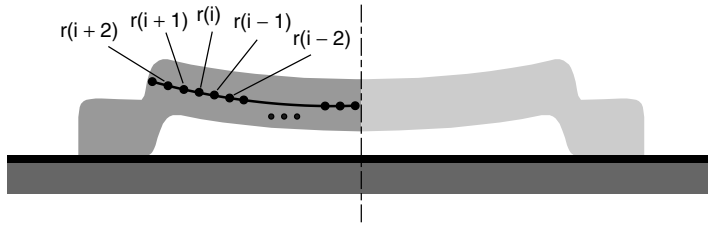

Figure 6.12 Finite differences for a capacitive pressure element

The bending of such a plate can be described by the following partial differential equation, see Gasch and Knothe [113]:

$$\frac{\partial^2 u}{\partial t^2} = -\frac{E t_p^3}{12\rho(1-\nu^2)}\left(\frac{\partial^4 u}{\partial r^4} + \frac{2}{r}\frac{\partial^3 u}{\partial r^3} + \frac{1}{r^2}\frac{\partial^2 u}{\partial r^2}\right) + w \tag{6.48}$$

Where u is the deflection, E the modulus of elasticity, t_p the thickness of the plate, ρ the density of the plate material, ν Poisson's ratio, w the excitation, and r the (radial) position variable. This is then discretised over the range of the plate radius in n nodes, see Figure 6.12. The above equation is used for each of these n points, whereby the positional derivation is replaced according to the following plan:

$$\begin{aligned}
\frac{\partial u}{\partial r} &\approx \frac{1}{12h}(u_{r(i-2)} - 8u_{r(i-1)} + 8u_{r(i+1)} - u_{r(i+2)}) \\
\frac{\partial^2 u}{\partial r^2} &\approx \frac{1}{12h^2}(-u_{r(i-2)} + 16u_{r(i-1)} - 30u_{r(i)} + 16u_{r(i+1)} - u_{r(i+2)}) \\
\frac{\partial^3 u}{\partial r^3} &\approx \frac{1}{2h^3}(-u_{r(i-2)} + 2u_{r(i-1)} - 2u_{r(i+1)} + u_{r(i+2)}) \\
\frac{\partial^4 u}{\partial r^4} &\approx \frac{1}{h^4}(u_{r(i-2)} + 4u_{r(i-1)} + 6u_{r(i)} - 4u_{r(i+1)} + u_{r(i+2)})
\end{aligned} \tag{6.49}$$

As a result of the form of the terms used, the necessity arises to add two further nodes at both ends of the discretised range. These do not describe a real expansion of the plate but can, however, be used in addition to the boundary nodes for the formulation of the boundary conditions. This yields a description with $2n + 4$ degrees of freedom, some of which are dispensed with due to the boundary conditions.

Overall, the drawing up of the equation system and its description in a form compatible with the electronics is definitely specified, but it is very cumbersome to achieve manually. For this reason a model generator is used in [322], [323] or [324], which automatically converts the partial differential equation into a formulation in an analogue hardware description language or a Spice compatible equivalent circuit on the basis of general integrators. The procedure is so general that it can also be used on other partial differential equations such as the heat conduction equation, see for example Bielefeld *et al*. [35] and [36]. However, one remaining limiting factor is the fact that the method is only suitable for relatively simple structures due to the nature of the underlying partial differential equations. Furthermore, in this model only the plate is considered and not its suspension.

Analytical modelling

For some structures, such as square or circular plates, analytical solutions to the partial differential bending equations are known. Models can be created on this basis if the geometric form of a micromechanical structure permits. This is particularly

true for very simple structures. Thus Chau and Wise [67] and Bota *et al*. [41], for example, use analytical equations for the modelling of the square membrane of a pressure sensor. In addition to bending mechanics, torsional mechanics can also be considered analytically, as Gómez-Cama *et al*. [122], for example, demonstrate for a capacitive acceleration sensor and Wetsel and Strozewski [428] demonstrate for a micromirror.

To illustrate analytical modelling, the example of a capacitive pressure sensor, see Figure 6.12, will be considered again in what follows. The bending of the upper plate can be described by the following equation, see Timoshenko and Woinowski-Krieger [401] or Voßkämper *et al*. [417]:

$$\Delta\Delta u = \frac{1}{D}(p + p_{el}), \qquad \text{where } D = \frac{E}{1-\nu^2}\frac{t_p^3}{12} \tag{6.50}$$

where Δ represents the Laplace operator, u the vertical deflection, D the bending resistance, p the external pressure and p_{el} an electrostatic pressure caused by the read-out voltage applied through the plates. The bending resistance is again defined as shown via the modulus of elasticity E, Poisson's ratio ν and the thickness of the plate t_p. The electrostatic pressure can be described as follows using the radius:

$$p_{el}(r) = \frac{1}{2}\varepsilon_0\varepsilon_{r,eff}(r)\left(\frac{U}{t_c + t_i - u(r)}\right)^2 \tag{6.51}$$

with the dielectric constants ε_0 and $\varepsilon_{r,eff}$, the radius r, the read-out voltage U, the thickness of the hollow space t_c and the insulator thickness t_i. A direct execution of the four-fold integration of (6.50) for the solution with respect to deflection is not possible because the electrostatic pressure in (6.51) is itself dependent upon the deflection. A polynomial approximation of p_{el} solves the problem, see [417]:

$$p_{el}(r) \approx \sum_{i=0}^{n} a_i r^i \tag{6.52}$$

The general solution of equation (6.50) is then calculated as:

$$u = \frac{1}{64}\frac{p}{D}r^4 + \frac{1}{4}C_1 r^2\left(\ln\frac{r}{R_0} - 1\right) + \frac{1}{4}C_2 r^2 + C_3 \ln\frac{r}{R_0} + C_4 + \sum_{i=0}^{n}\frac{a_i}{(i+2)^2(i+4)^2}r^{i+4} \tag{6.53}$$

with the radius of the plate R_0 and four constants C_1 to C_4 that have been yielded by the integration, the values of which are to be determined from the boundary conditions. With the aid of the resulting equation, further effects can be built in, such as the restriction of the plate movement through the insulator, the influence of plate suspension, or the dynamics of the movement.

Ritz method

A further procedure for the modelling of strains is the Ritz method, see for example Bathe [19]. In this process the partial differential equation is solved and an attempt is made to approximate an unknown displacement function, e.g. the deflection of a beam over its length by a linear combination of n interpolation functions. These must each correspond with the geometric boundary conditions. The n coefficients of the interpolation functions are yielded by the requirement that the elastic potential must be minimal. From this, n equations are found, which set the partial derivative of the elastic potential with respect to the coefficients equal to zero. So n equations are available for n coefficients. It should also be noted that the interpolation functions are defined over the entire mechanical structure, which makes the consideration of irregular structures considerably more difficult. The same applies for nonhomogeneous distributions of mass and stiffness. For this reason the significance of the Ritz procedure lies not so much in its direct application, but rather in the fact that it forms the basis of the finite elements method. Nonetheless, the direct use of the Ritz procedure can make sense in some cases.

Galerkin method

As in the finite differences approach, this method, see for example Bathe [19], also generates a set of ordinary differential equations from a partial differential equation:

$$L[\phi] = r \tag{6.54}$$

Where L is a linear differential operator, ϕ the sought-after solution, and r the excitation function. The solution of the problem should correspond with the following boundary conditions B_i:

$$B_i[\phi] = q_i|_{\text{at the boundary of } S_i} \tag{6.55}$$

A prerequisite here is that L is both symmetrical (6.56) and positive definite (6.57).

$$\int_D (L[u])v \cdot dD = \int_D (L[v])u \cdot dD \tag{6.56}$$

$$\int_D (L[u])u \, dD > 0 \tag{6.57}$$

Where u and v are arbitrary functions and D is the range of the operator. The solution should now be approximated as a linear combination of weighted interpolation functions h_i:

$$\overline{\phi} = \sum_{i=1}^{n} a_i h_i \tag{6.58}$$

The h_i interpolation functions are selected such that they each fulfil the boundary conditions. Then the residuum R is calculated as follows:

$$R = r - L\left[\sum_{i=1}^{n} a_i h_i\right] \tag{6.59}$$

For the exact solution the residuum is zero and, for the approximation, should at least be sufficiently low at all points of the solution range. Then the weighting factors a_i can be determined during the approximation of the partial differential equation. For the Galerkin method the following equations are used as the basis:

$$\int_D h_i R \, dD = 0 \qquad i = 1, 2, \ldots, n \tag{6.60}$$

Where D is again the solution range.

Hung *et al*. [156] use the Galerkin method to investigate a pressure sensor, which consists primarily of a bending beam that is fixed at both ends. A voltage and consequently an electrostatic force is applied to this. The time that passes before the beam 'snaps into place' as a result of the positive feedback of the electrostatic force, i.e. forcefully rests upon the insulator, is strongly dependent upon the prevailing air pressure. The modelling uses the Euler equation for bending beams and Reynolds' equation for air damping. The authors use the Galerkin method with up to four interpolation functions, which are determined with the aid of a FE simulation. They thereby achieve an acceleration of the simulation by a factor of between 4 and 105 in comparison to FE simulators, with deviations from the FE simulation in the range of 1%–14%.

This method permits the formulation of lower-order models. However, it requires that the system can be considered as a comparatively simple structure, because the starting point, the partial differential equations and boundary equations, either cannot be set or can be set only with great difficulty.

6.3.4 Experimental modelling

Introduction

Experimental modelling dedicates itself to the creation of models on the basis of measured data or FE simulations. The internal physics of the components is disregarded and only the terminal behaviour considered. In this manner we obtain so-called macromodels that can be simply formulated in a hardware description language. We thus obtain efficient and numerically unproblematic models. This method has its advantages if it is difficult or even impossible to derive the physical background of a component. However, its main problem is that the resulting models are only valid for precisely one geometric form of the structure and set of technology parameters. Every change means that a new model must be drawn up.

A whole range of approaches extract the main corner-stones of the behaviour of a component from measurements or simulations using finite elements and use this for simple models consisting of few equations, see for example Ansel *et al*. [11], Hofmann *et al*. [149], [150], Karam *et al*. [179] and Nagel *et al*. [292]. In what follows three approaches will be considered that aim in the aforementioned direction.

Table models

The simplest case of experimental modelling is based upon a list of input and output values, thus arriving at a table model that only considers the static case. In this manner it is possible, for example, to draw up a table listing pressures and the associated capacitance values for the pressure elements described above. Such table models lead to characteristics with kinks that can considerably detract from the convergence of the simulator. This problem can be circumvented by using the present value pair as a support point for the characteristic, e.g. on the basis of splines, which typically removes the numerical problems. In this manner measured values can be very simply integrated into a simulation. More elaborate procedures estimate the structure of the equations and move themselves to the identification of the associated parameter.

Identification of a harmonic oscillator

In [11], Ansel *et al*. consider a seismic acceleration sensor as a harmonic oscillator. For the modelling a linear differential equation is used for the force f and the deflection x:

$$a_0 f + a_1 \frac{df}{dt} + \cdots + a_m \frac{d^m f}{dt^m} = b_0 x + b_1 \frac{dx}{dt} + \cdots + b_n \frac{d^n x}{dt^n} \qquad (6.61)$$

For a spring-mass system, for example, m is set to 0 and n to 2. Here b_0 represents the spring constant, b_1 the viscous damping, and b_2 the seismic mass. For the system currently under consideration the parameters a_i and b_i are automatically obtained from the results of a simulation using finite elements. For this purpose the classical methods for system identification are used. This describes the mechanical section of the system. In addition, there is the conversion of mechanical deflection into capacitance based upon an interlacing comb structure. A table model is used for this, which is also determined on the basis of simulations using finite elements.

General identification

Hofmann *et al*. [149] and [150] propose a general procedure in order to put together the behaviour of a component from functional modules. The modelling is based upon a FE model, the behaviour of which is stored in a macromodel. Thus the complexity and nature of the underlying (partial) differential equations are not

known in advance, so that we have to start from the assumption of the existence of strong couplings and nonlinearities. Furthermore, it is required that inputs and outputs of the FE model can be formulated in an integral manner, i.e. they are not position dependent.

We now start with a basic model, the parameters of which should be identified with the aid of various optimisation procedures. For oscillating systems, for example, equation (6.61) would be a good starting point, whereby the parameters a_1 and b_1 would have to be determined. For the general case these can be determined from the criterion that the resulting model should behave as closely as possible to the FE model. The target function of optimisation is thus the minimisation of the behaviour difference between the predetermined and sought-after model, i.e. [150]:

$$\sum_{j}^{\#outputs} \sum_{i}^{\#timesteps} (f_{FEM_i}(t_j) - f_{MACRO_i}(t_j))^2 \tag{6.62}$$

For optimisation, gradient procedures, simulated annealing, or genetic algorithms can be used and it is also possible to switch between these. The resulting parameters initially apply only for the selected input function. In [150] it is thus proposed to initially define a set of input functions, which represent reality as well as possible. Then optimisation takes place primarily for the input function, the macro model of which exhibits the greatest differences in relation to the FE model. This procedure corresponds with a parameter identification for nonlinear systems.

Now, if it is difficult to arrive at an acceptable solution using parameter optimisation, this raises the question of whether the assumptions with regard to the structure of the solution equations were correct. Once again, the problem lies in the nonlinearities that rule out an analytical solution of the problem. The solution proposed by Hofmann *et al*. consists of setting operators that evaluate the differences between the FE and macromodel, such as for example 'rate of rise too low', 'overshoot too low' and so on. On the basis of this information a fuzzy controller base decides on possible structural changes. So we now go from parameter identification to system identification.

Overall, the procedure supplies efficient and numerically unproblematic models, that can be easily formulated in hardware description languages, e.g. HDL-A [149]. However, a significant computing time has to be expended for model generation. Furthermore, the validation of the generated models remains difficult, since firstly the quality and coverage of the selected input functions is sometimes questionable and secondly the inner physical structure is not available for an investigation into the plausibility. Finally, this type of modelling has to be performed afresh for virtually every variation of the micromechanical geometry or the underlying technology.

6.4 Summary

In this chapter, methods for the modelling of multibody mechanics and continuum mechanics have been highlighted and the representation of the resulting

models shown in hardware description languages. This, along with the results of the previous chapters, facilitates a full, universal modelling of mechatronic and micromechatronic systems in hardware description languages.

Now that the basic technologies have been dealt with in the preceding chapters, the following two chapters on mechatronics and micromechatronics supply a range of demonstrators to illustrate their application.