2
Principles of Modelling and Simulation

2.1 Introduction

The introduction of Information Technology in the last fifty years has allowed modelling and simulation to penetrate the majority of engineering disciplines and natural and social sciences. Regardless of whether the matter under debate is the design of wheel suspension for a car, the metabolism of a bacteria, or the introduction of a new interest formula, models of these real systems are always drawn upon to gain an understanding of the inner relationships of the system and to make predictions about its behaviour. The simulation is often also used as a substitute for experiments on an existing system, which is associated with a range of benefits:

- In comparison to real experiments, virtual experiments often require a significantly lower outlay in financial terms and in terms of time, because it is generally considerably cheaper to model virtual prototypes than it is to build real prototypes.
- Some system states cannot be brought about in the real system, or at least not in a non-destructive manner.
- Normally all aspects of virtual experiments are repeatable, something that either cannot be guaranteed for the real system or would involve considerable cost.
- Simulated models are generally completely controllable. So all input variables and parameters of the system can be predetermined. This is normally not the case for a real system.
- Simulated models are generally fully monitorable. All output variables and internal states are available, whereas in the real system every variable to be monitored involves at least a significant measurement cost. In addition, each measurement taken influences the behaviour of the system.

Mechatronic Systems Georg Pelz
© 2003 John Wiley & Sons, Ltd ISBN: 0-470-84979-7

- In some cases the ‘time constants’ of the experiment and observer are incompatible, such as the investigation of elementary particles or galaxies.
- In some cases an experiment is ruled out for moral reasons, for example experiments on humans in the field of medical technology.

However, these benefits are countered by some disadvantages:

- Each virtual experiment requires a complete, validated and verified modelling of the system.
- The accuracy with which details are reproduced and the simulation speed of the models is limited by the power of the computer used for the simulation.

In many cases the benefits outweigh the disadvantages and virtual experiments can be used advantageously. The repeatability guaranteed by the computer is particularly beneficial if the virtual experiment is systematically planned and performed as part of an optimisation.

In what follows we will define a range of terms relating to modelling and simulation. This will allow us to move from a general consideration to the systems investigated in this work, thus providing a good structure to the discussion. The following representation relates to the work of the SCS Technical Committee on Model Credibility, see [362].

Reality is initially an entity, situation or system to be investigated by simulation. Its modelling can be viewed as a two-stage process, as shown in Figure 2.1. In the first stage, reality is *analysed* and modelled using verbal descriptions, equations, relationships or laws of nature, which initially establishes a *conceptual* model. A field of application then has to be defined for this conceptual model, within which the model should provide an acceptable representation of reality. Furthermore, the degree of correspondence between conceptual model and reality that should be achieved for the selected field of application, has to be defined. A conceptual model is adequately *qualified* for a predetermined field of application if it produces the required degree of correspondence with reality. In the second stage of modelling the conceptual model is transformed into an executable, i.e. *simulatable*, model as part of *implementation*. This primarily consists of a set of instructions that describe the system’s response to external stimuli. The instructions can be processed manually or using a computer. The latter is called *simulation* and permits the processing of significantly greater data quantities, and thus the consideration of significantly more complex problems.

The development of models for simulation is a difficult process, and thus prone to errors. On the other hand, the reliability of a simulation is crucially dependent upon the quality of the model. So methods and tools are required that are capable of validating and verifying the models. Let us now define these two terms, *validation* and *verification*, more closely, see Figure 2.1. Model verification investigates whether the executable model reflects the conceptual model within the specified limits of accuracy. Verification transfers the conceptual model’s field of application

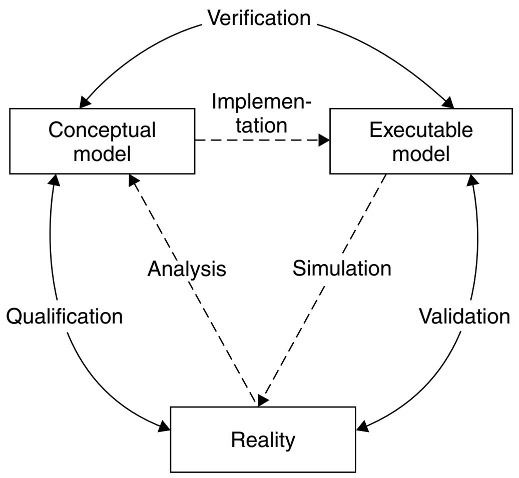

Figure 2.1 Model generation, simulation, validation and verification in context

to the executable model. Model validation, on the other hand, should tell us whether the executable model is suitable for fulfilling the envisaged task within its field of application. In other words: Verification ensures the system is modelled *right*, whereas validation is all about modelling the *right* system. Various degrees of validity can be defined for a model:

Replicative validity

A model is replicatively valid if it moves along tracks that have already been marked out by measurements upon the real system. This is the lowest level of validity. Such models may, for example, be used in the field of training to teach people to use a real system by means of virtual experiments.

Predictive validity

A model is predictively valid if it 'predicts' data that are not extracted from the system until later. So, for example, simulations supply important information on the functionality of a circuit even before it has been constructed in the form of a chip or board. It is also perfectly possible to mix predictively valid component models with replicatively valid models if measurement data is available for the modelling of some components but not for others. A predictively valid model is also replicatively valid.

Structural validity

A model is structurally valid if it not only describes the outward behaviour of a real system accurately enough, but also imitates the internal processes for the generation of the behaviour at the pins. This is the highest level of validity and this level in particular is required in order to understand the real system. A structurally valid system is also predictively valid.

2.2 Model Categories

We can obtain an initial classification of models by considering the range of values of the system variables, see for example Zeigler [435]. These may be continuous or discrete. A range of values is *continuous* if it covers real numbers or an interval of them. For example, a mechanical position has a continuous range of values. In a *discrete* range of values, on the other hand, the system variable takes on a value from a finite (or at least countable) quantity of values, as is the case for digital, electronic signals. The states of the model take on a discrete, continuous or mixed form depending upon the system variables.

Time is explicitly removed from the system variables and investigated in a similar manner with respect to its value range. In the *discrete* case time proceeds in leaps; valid time points are calculated as the product of a whole number and a basic time span. This may, for example, be suitable if a gate simulation is run with unit delays. By contrast, we can also consider models in which time is *continuous*. These can be divided into two categories: *event-oriented* models and *differential equation* models. In the former case each change of state of the model is triggered by an event, so that the trajectory of system states proceeds in leaps. The events themselves can occur at arbitrary points in time; their number in relation to a predetermined time interval is however finite. By contrast, in models based upon differential equations the trajectory of system states is continuous. Changes are described on the basis of the system variables and their rate of change.

A further possibility for differentiating between models is based upon whether the description uses *concentrated* or *distributed* parameters. Examples of the former case are electronic components or the fixed and elastic bodies of the multibody representation of a mechanical system. Distributed parameters should be used in the consideration of a mechanical continuum, for example.

Models may furthermore be of a *static* or *dynamic* nature. In the former case, in electronics for example, when determining the operating point of a circuit it is sufficient to represent capacitors as open circuits and coils as short-circuits. In multibody mechanics stationary systems can be analysed. Dynamic models are required in electronics for transient simulations, i.e. for those over a time range, whereas in mechanics we can differentiate between two application cases: kinematics and kinetics, see for example Nikravesh [299]. Kinematics relates to the investigation of positions, speeds and accelerations without taking into account the forces that cause the movement they describe. Kinetics also considers the acting forces.

In some cases a model cannot be described in a purely deterministic manner, meaning that at least one random variable must be included. As an example, a model may serve to evaluate the power of a computer, which accesses its hard drive with a probability of x% and its tape deck with a probability of y%. Models containing at least one random variable are classified as *stochastic*. All others are called *deterministic*.

A further option for the classification of models is the consideration of the 'outside world' of a model. If the model is isolated from the outside world and thus has no inputs and outputs, then it is called *autonomous*. All other models are called *non-autonomous*. An autonomous model produces a movement in the state space from itself, without taking in and producing data, whereas a non-autonomous model primarily converts values at the inputs into the outputs based upon the current state.

A final option for the classification of models is represented by the question of whether or not time crops up explicitly in the model equations. In the former case the model is *time-variant*, in the latter *time-invariant*.

2.3 Fields of Application

2.3.1 Introduction

If technical systems are to be developed, two main fields of application can be identified for the simulation: The validation of specifications and the verification of designs. In the ideal case the specification or design will be available immediately in model form, so that nothing stands in the way of direct simulation. Hitherto this has mainly been the case in the design of digital electronics using hardware description languages. Otherwise, modelling must take place first to bring about the transition from an arbitrary description to a simulatable model.

The use of modelling and simulation is closely linked to the underlying design processes. These can be roughly divided in accordance with their design direction into top-down and bottom-up design flows. In what follows these will be briefly introduced and characterised by their influence upon modelling.

2.3.2 Bottom-up design

Bottom-up design is the classic method of development of electronics and mechanics, see Figure 2.2. The initial starting point is a specification, which is typically drawn up in natural language. Then the basic components, e.g. transistors, resistors, capacitors or springs, masses, shock absorbers, joints, etc. are added and combined successively to form ever more complex and abstract creations until a complete design emerges. This takes place on a structural level, so that the only thing that is determined each time is which submodules make up a module and how these are

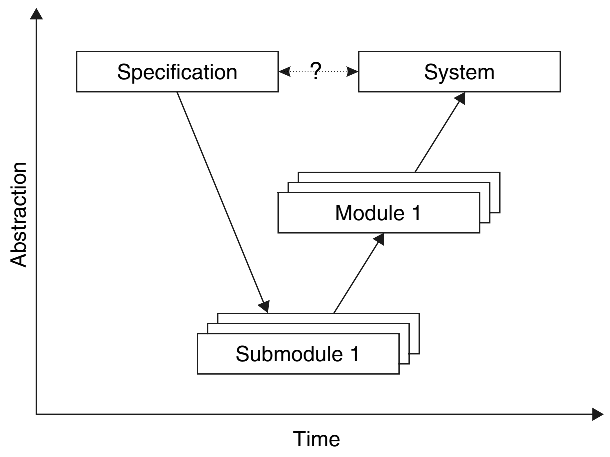

Figure 2.2 Bottom-up design process

to be connected together. Such a design can be performed using a circuit editor or a suitable tool for multibody systems.

The primary advantage of bottom-up design is that the influences of a nonideal implementation can be taken into account at an early stage. For electronics these may be unavoidable parasitic resistances, capacitances and inductances. In the field of mechanics they may be friction effects, for example.

However, one problematic aspect is coming upon the specification for the design, after having had to take a 'diversion' via the submodules and modules from the abstract functional description. This is because, as a result of the structure-oriented modelling, a system can only be simulated when it has been completely implemented. Thus errors and weaknesses in the system design are not noticed until a late stage, which can bring about considerable costs and delays.

2.3.3 Top-down design

A significant characteristic of top-down design is the prevailing design direction from abstract to detailed descriptions, see Figure 2.3. The starting point is a pure behavioural model, the function of which already covers a good part of the specification. The model is successively partitioned and refined until an implementation is obtained. It is necessary to describe a system or module of it in a functional manner. This was first made possible by the introduction of hardware description languages in the field of electronics. Using these the design is directly formulated as a model, so that most of the modelling can be dispensed with.

The top-down design sequence has the following advantages:

- Errors and weaknesses in the design are noticed early, in contrast to the bottom-up approach.

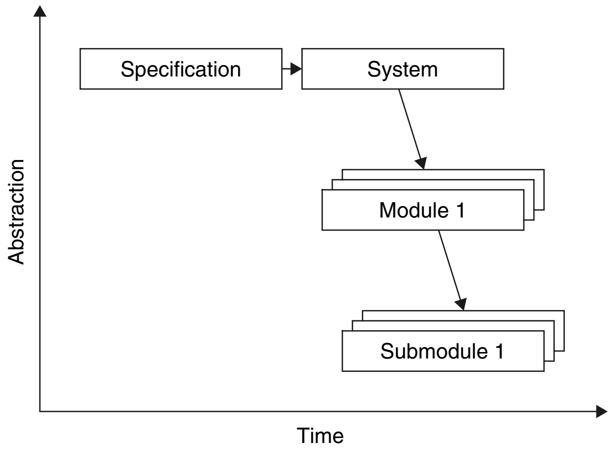

Figure 2.3 Top-down design sequence

- The implementable part of the specification can be validated by simulations.
- The implementable part of the specification is available as a precisely defined reference for the verification of the design.
- The functional part of the specification is unambiguous and complete (in contrast to a specification in natural language). In the event of doubt, a simulation is run.
- The implementable specification and the models of the individual design stages mean that full documentation is available, which however still remains to be supplemented by comprehensive commentary.

In the case of mixed-signal design, the implementable specification can be made available to the test engineers at an early stage as part of a ‘simultaneous engineering’ approach. Using a model for the testing machine a virtual test is created, in which test programmes can be developed on the workstation. This removes the fixed sequence of design $\rightarrow$ production $\rightarrow$ test development and also saves a great deal of time on test development.

However, the disadvantage of the use of implementable specifications is that some technical content can be expressed in a simpler, more compact and more easily understood form in natural language than in a formal modelling language. In addition, there is the question of adhering to the formally correct description of the desired semantics, which incurs an additional cost in relation to a paper specification. Finally, problems in the physical realisation, such as excessive delay times for certain blocks, are not recognised until a relatively late stage.

For mechanics the top-down design sequence is still in the development stage. A significant reason for this is that unified and standardised description methods for mechanical behaviour, with which a design can be taken incrementally from an abstract specification to a detailed implementation, are only now being developed.

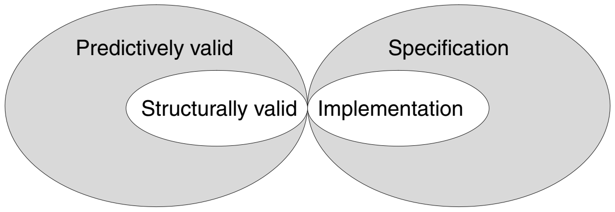

Figure 2.4 Level of validity and its significance for the design of a technical system

2.3.4 Relationship of design strategies to modelling

In the case of the top-down design sequence, modelling is used for the specification of the desired behaviour or for the formulation of designs. In both cases the result can be directly checked through simulation; there is no such thing as modelling exclusively for the purpose of simulation. In this connection, an important classification of such models by their level of validity can be made, see Figure 2.4. For a specification, predictive validity is sufficient — the manner in which the terminal behaviour of the specified systems and modules is individually generated is not relevant. A system design, on the other hand, ideally supplies a structurally valid model that describes both the terminal behaviour and the inner structure.

By contrast, if a technical system is to be developed using a bottom-up design sequence, then simulation can be used for checking the system design or parts of it after the conclusion of the design phase. Modelling is thus not an integral part of the design process; instead it is often performed exclusively for the purpose of the simulation, which raises questions regarding the verification and validation of the model.

Where modelling is used outside a design process we can differentiate between the following two cases: structurally valid modelling in natural and social sciences in order to gain understanding of a system; and replicatively valid modelling in the field of training. The former plays only a lesser role in the consideration of technical systems. The latter is used primarily for the imitation of familiar behaviour. A well-known example is flight simulators that are used for the training of pilots in all feasible operational situations. Such simulators are now available on the market for almost all types of vehicle. But simulators can also be used for other types of training. Preparation for the repair of the Hubble telescope involved a great deal of expenditure on simulation due to the considerable costs and the narrow time frame for such measures in space, see Loftin [237] and [242].

2.3.5 Modelling for the specification

The main purpose of a specification is to describe the desired behaviour of a system to be developed and the associated boundary conditions. Classically, a specification is available on paper, which is associated with a whole range of problems.

First of all it raises the question of its validity, i.e. whether the described system really corresponds with the desired system. Furthermore, it is doubtful whether a given (paper) specification is completely and unambiguously formulated. These questions can only be answered in a systematic manner when the transition is made to an implementable specification, which can then be validated by simulation, for example. A further advantage of this transition lies in the possibility of the verification of the individual design stages against the specification. Furthermore, this opens up the opportunity of performing a formal verification against the specification. In digital electronics, behavioural modelling as a specification is becoming increasingly prevalent, in all other domains it is still at a very early stage.

Modelling for a specification is pure behavioural modelling, which — as is the case for a paper specification — may not anticipate the implementation. For a microprocessor, for example, a specification would describe only the instruction set and the associated actions. The way that the individual operations are realised cannot be the object of the specification. An executable specification for a memory module may consist of a large array for the memory content and some logic for the processing of read and write processes. The specification of an A/D converter could formulate the pure translation of analogue values into digital values and the resulting delay.

2.3.6 Modelling for the design

Modelling for the checking of technical system designs for each simulation is the classic application case. All engineering-science disciplines use simulation beneficially to this end.

This applies particularly in microelectronics. A manufacturing run typically lasts 6–12 weeks and is associated with significant costs. Repairs to manufactured chips are more or less impossible. Under such boundary conditions, one cannot afford to iterate the manufacturing process to rectify design errors. Instead, it is necessary to enter manufacture with a fundamentally error-free design, which — given the complexities that are currently possible, involving some tens of millions of transistors — cannot be achieved without simulation.

If we consider discretely structured printed circuit boards, then it is slightly less critical that the circuit is fully checked in advance by simulation. The etching and fitting of circuit boards is significantly simpler and quicker than chip manufacture. Changes can be performed comparatively easily. The circuits are also less complex by orders of magnitude. So it can be worthwhile to solder a circuit together as a bread-board arrangement and check it by measurement. Nevertheless, the performance of virtual experiments on a computer is generally quicker and cheaper than the real experiment in the laboratory.

For software, things are comparatively simple. The compilation of software can be regarded as rudimentary modelling, as software is executable after this stage, i.e. it is simulatable. The simulation sequence and the simulation result are normally displayed in a debugger that shows the current status of the software, i.e. program line and variable values, plus their outputs on the terminal. Without this type of simulation, software development would be unthinkable.

Like electronics, the construction of mechanical systems in reality is very expensive in terms of time and costs. In many of the industries in question the answer to this problem lies in the increased use of simulation. The automotive industry is particularly advanced in this field. The two main key words here are *digital mock-up* and *virtual prototype*, see for example Paulini *et al.* [317] or Schweer *et al.* [376]. A *digital mock-up* is as complete as possible a description of a single product on the computer and thus represents a limited data quantity. All the various tools check the design on the basis of this data. The digital mock-up thus primarily represents a medium for information exchange, which links together data sources and data sinks in the design process. At regular intervals, for example every two weeks [376], new data are put in and thus are available to all possible users. A *virtual prototype* is extracted from the data of the digital mock-up, which can then be used for experiments on the computer. A classic example of this is the simulation of crash tests. In this application, a finite-element model is obtained from the CAD data of the body by automatic meshing, which can then be subjected to any desired crash situations. Although the simulation requires several hours of processing time even on the fastest computer, it means that the majority of real crash tests can be dispensed with. Furthermore, simulations are also run in virtually all other sectors of the automotive industry, such as for example in the development of running gear, engine, drive train and the associated electronics.

2.4 Model Development

2.4.1 Introduction

The following section provides an overview of the most up-to-date methods for model development in electronics and mechanics, looking at both the common ground and differences. We can make an initial classification by asking whether the model describes the structure or the behaviour of a system.

Taking the first case, in classic modelling the model establishes only which components make up the system and how these are connected together. Alternatively, however, the term structural modelling can also be expanded and, for example, take in the description of the structure of an equation system or a finite state machine. In such cases the following forms of model description may be called structural: electronic circuit diagrams, state graphs, multibody diagrams, meshes of finite elements, block diagrams, bond graphs and Petri nets. The common factor of all these descriptive forms is that they are all graphical in nature.

If, on the other hand, it is the behaviour of a system that is to be described then this can be achieved on the basis of the underlying physics or the measured input/output behaviour. In the former case the development of such models is relatively costly and requires a comprehensive understanding of the system. On the other hand, such models can be adapted to the actual system over a wide range by modifying parameters. If, for example, a system is to be driven by a DC motor, various makes can be included in the simulation by the use of the applicable parameters. These ‘generic’ models thus cover a whole class of components. As an alternative to modelling on the basis of physical behaviour the other option is to take measured data and feed this into models. This is also called experimental modelling and is used if physical modelling is not implementable or the resulting model is too complex for the desired purpose. Typically, however, experimental modelling has to be repeated every time one of the components in question is altered. Both in the case of physical and experimental modelling the models are generally formulated on the basis of equations and assignments, i.e. consequently formulated in the form of text.

In addition to a simulation, an emulation may also come into consideration under certain speed requirements. This has different characteristics for electronics and mechanics. In the field of digital electronics the term emulator is used to mean a device that can take on the function of any desired digital circuit, see for example Bender and Kaiser [25]. This function is based upon a number of programmable chips, for example so-called FPGAs, the logic functions of which are stored in a local RAM and can thus be modified. Currently up to a hundred thousand gate functions can be stored on a single FPGA. With regard to speed, FPGAs, and thus emulators, are generally significantly slower than dedicated hardware, but are, however, faster than a simulation by orders of magnitude. The emulation of analogue electronics and mechanics on the other hand is based upon signal processors, so-called DSPs, that are optimised for analogue signal processing, see for example Huang *et al.* [155] or Georgiew [116]. So differential equation models of mechanical components can again be calculated faster than is the case for a simulator by orders of magnitude.

Since modelling is a difficult process, and prone to errors, in some cases real components are embedded into a simulation, see for example Helldörfer *et al.* [136] or Le *et al.* [219]. This is also called ‘hardware in the loop’. This does not mean that the entire system is constructed as an electronic bread-board assembly or mechanical prototype, instead usually just one component is fitted. Alternatively, the environment of the system to be developed can be included in real form. The rest of the system is modelled in the classical manner, so that simulated and real behaviour are mixed together. The advantage of this is that the modelling and its validation can be dispensed with for the real hardware in the simulation loop. However, the principle disadvantage is that the real components have to be fully installed in the laboratory and adequately fitted with actuators and sensors in order to ensure the main inputs and outputs. Furthermore, the simulation of the remainder of the system must in this case take place in real time, which may involve considerable cost, depending upon the system. Alternatively, this real time simulation can be replaced by an emulation to speed things up.

All the methods described up to this point relate to the description of an error-free system. This is worthwhile if the simulation is to contribute to the actual design. In some cases, however, the aim is to investigate the effect of errors within the system. In this case error modelling is called for. One application for this is the evaluation of measures to increase intrinsic safety; another is the evaluation of test methods for differentiating between functional systems and rejects during production. In both cases, errors that impair the function of the system under consideration are modelled. Here too the modelling represents an abstraction of reality, which in the ideal case covers several error mechanisms. For example, the stuck-at error model in digital electronics describes the permanent presence of a logical 0 or logical 1 at a signal of the circuit. Whether this is caused by a short-circuit with a supply cable or by excessively deep etching of contact holes is of secondary importance. The decisive point is that the circuit no longer functions correctly and that this problem can be detected by the tests developed.

Due to their importance, structural, physical and experimental model development will be considered in more depth in the following. Finally, we note that specialist fields, such as modelling with neural networks, fuzzy techniques or genetic programming, will not be considered.

2.4.2 Structural modelling

Introduction

A structural model is characterised by the basic models used and the connection structure between these basic models. A module can be composed of basic models and can itself be again connected to other modules. This can be performed successively, thus describing complex systems. A structural model can be characterised on the basis of the following terms: Hierarchy, modularity, regularity and locality. The hierarchy of a model is derived from the call structure of basic models and modules. So an operational amplifier (=module) can be put together from MOS transistors (=basic models) and then circuits can be built up from operational amplifiers. Using graph theory, such a hierarchy can be described as a tree, in which the roots represent the system as a whole and the leaves represent the basic models. The number of levels of the hierarchy grow in a logarithmic relationship to the number of basic elements involved. The modularity of the system relates to the question of how simple and reasonable it is to divide the system into modules. Regularity is a measure of how many module types are necessary to represent the entire system. A low number is beneficial here because it indicates a compact representation. Finally, locality is a measure of how well a module can be considered without the context of its installation. Modules with straightforward interfaces to their outside world are particularly beneficial here.

In the following, models are considered in the form of circuit diagrams, state graphs, multibody diagrams and finite elements. Further descriptions with structural aspects are bond graphs, block diagrams and Pr/T networks.[1] As these descriptive forms also permit a modelling of electro-mechanical systems, these are described in detail in Chapter 3 as alternatives to modelling using hardware description languages.

Circuit diagrams

In the case of design using a circuit diagram editor, modelling is primarily used for the derivation of a net list, which is used as a circuit model, incorporating the component or gate models. This procedure is so simple and unproblematic that the process of modelling a circuit is not generally perceived as such. Likewise, there are not normally any problems with the validation of the circuit model. In the most extreme case there may be verification problems with the program for deriving the net list. The field of application is predominantly the development of analogue circuits. Although digital circuits can also be developed using circuit diagrams, a top-down design process is only possible using behavioural modelling based upon hardware description languages.

State graphs

Digital systems can also be represented by state graphs with the system structure then being stored on relatively abstract levels. The selection of the state transitions is precisely specified by conditions. Furthermore, in state graphs only the structure of the connections is necessary in order to characterise the model in question. Such a model can, for example, be used for the specification of digital behaviour, but it can also be translated into a programming or hardware description language and then used directly for the design of software and hardware.

Multibody diagrams

Things are more complicated for multibody mechanics. Although the importance of structural modelling is gaining increasing recognition here too, see for example the work of Panreck [313], when drawing up the model equations it is often the system as a whole that is considered rather than viewing it as a collection of components. Only with the introduction of object-oriented modelling, see Otter [308] or Kecskeméthy [185], does the structural modelling of multibody systems also become more prevalent.

Finite elements

A particularly graphic form of structural modelling is to break down mechanical structures into finite elements for the modelling of continuum mechanics. This is

[1] Predicate/transition network.

also called meshing, and both geometric dimensions and topological information are important. The element matrices of the individual finite elements are found from their material parameters and geometry, whereas the connection structure between the elements, and consequently the system matrix, is derived from the topology. Often the meshing has to be checked manually in order to ensure that the elements have the correct form, the grid is sufficiently fine and available symmetries are exploited.

2.4.3 Physical modelling

Introduction

In physical modelling the laws of physics are used to describe the behaviour and inner action mechanism of a system or a component. The selection of the relevant relationships depending upon suitability and efficiency and the establishment of cause and effect chains, requires a comprehensive understanding of the system and remains an engineering task. Computer support for this form of modelling is at best rudimentary.

In the following, some classifications will be undertaken for the characterisation of the physical modelling based upon various criteria. These consider the perspectives of modelling and the nature of the yielded equations. Otherwise the reader is referred at this point to Chapters 5 and 6 on modelling, and also to Chapters 7 and 8 on applications, which contain a whole range of examples of physical modelling and electro-mechanical systems.

Perspectives of modelling

The perspectives of modelling offer a coarse division of the physical models which, however, runs through all disciplines like a red thread. We should differentiate here between whether the system perspective or the component perspective has been selected. In one case the system-oriented modelling formulates the system in the overall context; in the other case object-oriented modelling describes components, which only form a system by their connection together, i.e. by structural modelling. The decisive factor is that in object-oriented modelling no system knowledge is fed into the component model. This ensures that the components can be used in any desired context, so that modelling work only has to be performed once and not for each system.

Hitherto in electronics, more significance has been attached to object-oriented modelling. The physical models for electronic components provide the classic example of this. These are formulated independently of the circuit in which they are used. The connection structure is determined in a circuit diagram, which forms a structural model. Thus the validation of the circuit model is in principle achieved by a validation of the component model. This is particularly worthwhile if the number of basic models is small. But object-orientation is also becoming increasingly prevalent in digital design using hardware description languages, although in this context it should be regarded more in the context of an increase in efficiency in the development of text-based, software-like models, see for example Ecker and Mrva [93].

In mechanics object-orientation has only recently been implemented in order to make modelling easier, whereby the work of Otter [308] and Kecskeméthy [185] in particular, are worth mentioning. One explanation for this is the fact that the number of basic elements and the associated variation in mechanics is significantly greater than is the case in electronics. Furthermore, the classic modelling methods of mechanical engineering often lead to descriptions in the form of generalised coordinates,[2] which are again incompatible with object-oriented modelling. The advantage of the generalised coordinates is that the resulting equation system has a minimum number of equations and, furthermore, the constraints can be disregarded for holonomous systems. This is attractive from a numerical point of view. However, generalised coordinates can only be specified by drawing upon knowledge of the entire system and not from the mole-hill perspective of a component.

Resulting equations

In this section we will investigate the equations that result from the various modelling forms. From a mathematical point of view, a digital gate or the setting of a digital signal in a hardware description language gives an instruction, which is executed after the passage of a predetermined time period. This period corresponds with the time delay of the described block. If the block is defined without a delay, then a virtual period of time still passes, the delta time, in which although the simulation time does not proceed, a check is made to ensure that the right-hand sides of all assignments have already been evaluated before the new value of the assignment under consideration becomes effective. Otherwise the parallel processing of instructions would not be possible.

In the case of an analogue circuit, the modified node voltage analysis is generally used, see Vlach and Singhal [410] for a good overview. This establishes differential equations for capacitances and inductances. Transistor models can include one or more parasitic capacitances. Otherwise the heart of transistor models, like diode models, is made up of a parallel circuit consisting of a resistor and a current source, the parameters of which have to be set for each new time interval. This corresponds with an arbitrary linear characteristic that can be placed as a tangent at the current working point on the nonlinear characteristic of the transistor. Voltage and current sources each correspond with constraints that are formulated in algebraic equations. Resistors are also expressed in algebraic equations. Overall a differential-algebraic equation system is established that is also known as DAE

[2] See Section 6.2.

(differential-algebraic equation set). The number of equations depends upon the circuit and is very high, typically significantly above the number of degrees of freedom. The resulting system matrices are however only sparsely occupied.

For multibody mechanics, the equations of motion are normally derived by means of the application of a classical principle, e.g. that of Lagrange or D'Alembert. When drawing up the equations it is possible to choose between two extremes. In one case the generalised coordinates, which fully describe the state of a system and which can also be regarded as degrees of freedom, are first determined. For n generalised coordinates (at least for holonomous systems) n equations can be drawn up. The constraints fall away, leaving a system of ordinary differential equations. However, these may turn out to be very complex. Alternatively, it is possible — as in electronics — to permit more unknowns and thereby obtain a system of differential equations for the motion of bodies and algebraic equations for the constraints, which may, for example, be caused by joints. This establishes a system of DAEs, which can be solved using similar methods to those used in the circuit simulation, see for example Orlandea *et al.* [304]. In both cases the number of degrees of freedom is relatively small in comparison to those in electronics. The number of objects under consideration, such as bodies, joints, springs, shock absorbers, etc. is generally significantly below one hundred. However, the numerical problems caused by transitions between static and sliding friction, mechanical impacts, three-dimensional coordinate transformations and other effects, cannot be disregarded.

In the representation of continuum mechanics by means of finite elements the number of degrees of freedom is significantly higher than those in multibody mechanics. The associated system matrices normally have a band shape, which the simulation exploits by suitably customised numerical procedures. Overall, this normally establishes a system of ordinary differential equations, the parameters of which, i.e. the inputs into the mass, damping and stiffness matrix, may however have to be recalculated at runtime.

2.4.4 Experimental modelling

Introduction

Experimental modelling consists of the development of mathematical models of dynamic systems on the basis of measured data or at least providing existing models with parameters. So neither the underlying physics nor the internal life of the system need necessarily play a role in model generation. In contrast to physical modelling there are procedures for experimental modelling in which the modelling can be wholly or partially automated.

Table model

The simplest method of incorporating measured data is by the formulation of table models that lead to a stepped or piece-wise linear characteristic. The problem with the trivial conversion of a table model is the abrupt changes or kinks that are caused by the fact that only a finite number of values are available. The difficulties are numerical in nature since numerical oscillations may occur at abrupt changes and kinks. These are caused by the fact that — as a result of feedback — different sections of the characteristic are approached alternately and this may impair or even prevent the convergence of the simulation. A possible solution is offered by procedures that smooth the characteristic, such as the Chebychev or Spline approximations.

Parameter estimation and system identification

In this connection we can differentiate between two aspects: Parameter estimation and system identification. Parameter estimation requires a model and considers the parameters that belong to it. Some parameters, such as mass or spring constants are generally accessible without parameter estimation, whereas other parameters, e.g. coefficients of friction, can often only be determined within the framework of parameter estimation. The identified parameters then ensure the best possible correspondence between simulation and measurement.

In system identification, on the other hand, a model for the system is created on this basis or selected from a group of candidates. This is generally efficient and numerically unproblematic. The quality criterion here is the degree of correspondence that can be achieved using parameter estimation. The two significant disadvantages of parameter estimation and system identification are that, firstly, a measured result must be available in advance, which means that the system can only be considered after its development and manufacture. Secondly, the results are often not transferable, or at least not in a straightforward manner, to variations of the system or of components.

There are typically four stages to a system identification, see for example, Kramer and Neculau [206] or Unbehauen [405] and Figure 2.5.

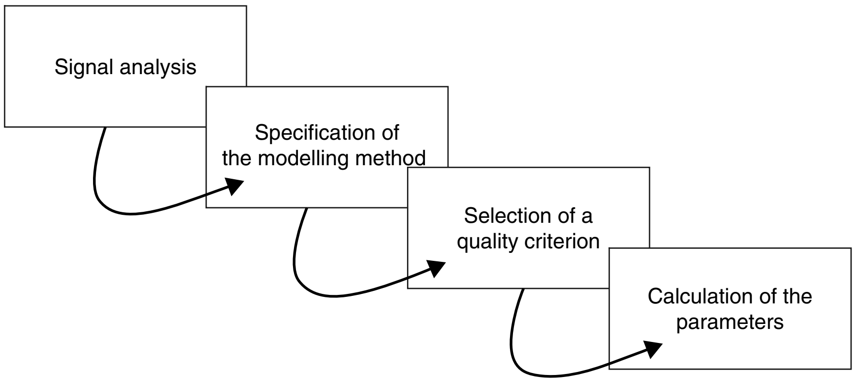
Signal analysis
Specification of the modelling method
Selection of a quality criterion
Calculation of the parameters

Figure 2.5 System identification sequence

The first stage of signal analysis is the establishment of a suitable test signal, which is triggered by the system. Possibilities here are step functions, rectangular pulses, triangular pulses and many more. An inspection in the frequency range facilitates investigations into whether the system to be identified is sufficiently excited over the spectrum of interest. Measurements are generally only made at discrete time points, so that a sampling interval must also be determined. Furthermore, a measurement time must be specified, the lower limit of which is determined by the point at which sufficient data is available for identification. The progressive nature of a real system imposes an upper limit on the measurement time. Then, signal processing procedures may also be used, such as averaging, root mean square calculation, or Fourier analysis, correlation analysis, and spectral analysis. So, for example, statistically dispersive noise signal components can be disposed of by averaging similar measurements which, however, multiplies the measurement time.

In stage two, determining the model approach, we can choose between prefabricated and customised structures, see for example Ljung [234]. The former may, for example, consist of canonical models in the state space and lead to a ‘black-box’ parameterisation, i.e. model structure and parameters have no physical significance, but rather serve merely as a vehicle for reflecting the observed behaviour. Customised equation system structures, on the other hand, are based upon a physical modelling of the system, so that the identified parameters also possess a physical significance. In any case, however, all available information about the system should be fed into this. This applies in particular to the faults that are virtually always present, which in most cases rule out an exact solution.

The identification typically rests upon minimising the discrepancy between measurement and simulated behaviour or a functional of this. Various quality criteria are used for this, one of which is selected in the third stage. Criteria are particularly frequently selected that assess a quadratic function of the measurement error.

To conclude the identification, numerical procedures are used in order to minimise the quality criteria selected in the third stage. These procedures are performed for all model structures proposed in the second stage, so that not only are the parameters in question determined in this stage, the quality of the individual structures in relation to one another are also established. This facilitates a selection of the model structure.

In the simplest case we can, as in Kramer and Neculau [206], quote the following equation for the system under investigation:

$$y_k = a \cdot x_k + n_k \tag{2.1}$$

where x_k denotes an input quantity, y_k an output quantity, n_k a disturbance variable in relation to the measurement and a is the parameter to be estimated. This relationship should be modelled on the basis of the following approach:

$$\hat{y}_k = \hat{a} \cdot x_k \tag{2.2}$$

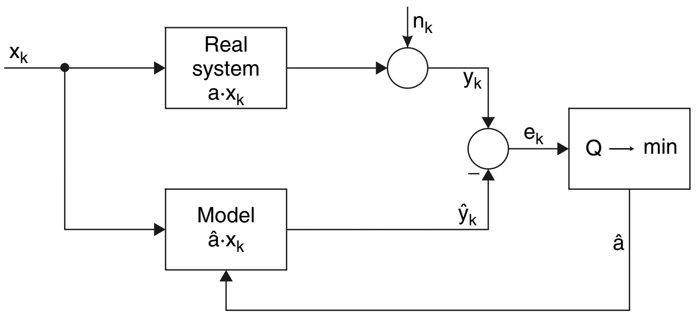

Figure 2.6 Comparison between real system and model for parameter estimation

This can also be graphically represented as shown in Figure 2.6. The aim of this is to minimise the quality function Q, so that the estimated parameter â is optimised in relation to Q.

A common approach for the quality function Q is to find an expression that is proportional to the quadratic average of the error signal e_k:

$$Q = \sum_{k=1}^{n} e_k^2 = \sum_{k=1}^{n} (y_k - \hat{y}_k)^2 = \sum_{k=1}^{n} (y_k - \hat{a} \cdot x_k)^2 \qquad (2.3)$$

where n is the number of measurements. For a compact representation the signals should henceforth be regarded in the form of n-dimensional vectors:

$$\begin{aligned} \mathbf{x}^\mathbf{T} &= [x_1\, x_2 \ldots x_n] \\ \mathbf{y}^\mathbf{T} &= [y_1\, y_2 \ldots y_n] \\ \hat{\mathbf{y}}^\mathbf{T} &= [\hat{y}_1\, \hat{y}_2 \ldots \hat{y}_n] \\ \mathbf{e}^\mathbf{T} &= [e_1\, e_2 \ldots e_n] \end{aligned} \qquad (2.4)$$

Thus the quality function can be described in vector notation as follows:

$$Q = \mathbf{e}^\mathbf{T}\mathbf{e} = (\mathbf{y} - \hat{a}\mathbf{x})^T \cdot (\mathbf{y} - \hat{a}\mathbf{x}) = \mathbf{y}^\mathbf{T}\mathbf{y} - 2\hat{a}\mathbf{y}^\mathbf{T}\mathbf{x} + \hat{a}^2\mathbf{x}^\mathbf{T}\mathbf{x} \qquad (2.5)$$

Now Q should be minimised in relation to â. For this to be achieved the partial derivative of Q in relation to â must become zero, i.e.:

$$\frac{\partial Q}{\partial \hat{a}} = -2\mathbf{y}^\mathbf{T}\mathbf{x} + 2\hat{a}\mathbf{x}^\mathbf{T}\mathbf{x} = 0 \qquad (2.6)$$

Solving this with respect to â finally gives:

$$\hat{a} = \frac{\mathbf{y}^\mathbf{T}\mathbf{x}}{\mathbf{x}^\mathbf{T}\mathbf{x}} \qquad (2.7)$$

Equation (2.7) is also called a regression and represents the solution for the method of least squares [206]. The inclusion of information on the interference process allows us to obtain better parameter estimates, as is the case in the weighted method of least squares.

2.5 Model Verification and Validation

2.5.1 Introduction

As defined in Section 2.1, model verification answers the question of whether the implementable model reflects the conceptual model within the specified boundaries of accuracy, whereas the purpose of model validation is to show whether the implementable model is suitable for fulfilling the envisaged task within its field of application. In what follows the most important methods in this field will be introduced. These originate from a very wide range of fields of application, some of which lie outside the field of engineering sciences. They are, however, general enough to be used in a technical context. Good overviews of the underlying literature can be found in Kleijnen [193], Cobelli *et al.* [72] and Murray-Smith [288], [289].

2.5.2 Model verification

Verification on the basis of the implementation methodology

The most direct form of verification takes place as early as the implementation stage and aims to ensure that, where possible, the errors to be identified by verification do not occur at all. This requires intervention into the methodology of model implementation. In this context, the same boundary conditions often apply as those for the development of software since, in this field too, a formal description based upon syntax and semantics is used for the formulation of a given technical content. Accordingly, most of the mechanisms that are used for software development also come into play here in order to avoid implementation errors. A few key words here, see Kleijnen [193], are: Modular modelling, object-oriented modelling or the ‘chief modeller’ principle, in which the actual implementation is as far as possible performed by a single person, whilst the other colleagues of the ‘chief modeller’ relieve him of all other tasks. In addition, there is the modular testing of submodels, so that modelling errors are recognised as early as possible and at lower levels. A further important aspect of verification lies in the correct definition of the scope of the model and in the ongoing checking to ensure that this scope is adhered to. Extrapolations beyond the guaranteed range should generally be treated with extreme caution.

Plausibility tests

Plausibility tests can also make a contribution to verification (and validation), see also Kramer and Neculau [206]. This is particularly true if they can be performed by means of simple manual calculations. They are based upon analytical considerations or the results of an initial simulation. The following criteria could possibly be drawn upon for plausibility tests:

Causality The cause should precede effect in reality and in the model. Any deviation from this principle indicates serious deficits in the model.

Balance principles The principles of the conservation of energy and matter apply not only to the physical reality, but also for the model itself.

Current/voltage laws Currents, forces and moments at a point add up to zero. Voltages and velocities add up to zero in a closed loop. These relationships apply for any electronic or mechanical system with concentrated parameters.

Value range State and output variables and parameters are normally associated with an applicable range of values. Although this is not necessarily precisely defined, unrealistic values can be recognised very quickly. For example; areas, volumes, energies and entropies can never be negative.

Consistency of units Model equations are generally formulated without units. Nevertheless, it is often worthwhile using the consistency of units as a criterion for verification.

Verification on the basis of alternative models

There are often several methods or tools available for modelling and subsequent simulation. If two approaches are independent of each other in terms of methodology and realisation, then they can be used for mutual verification. This arises because the probability of different errors producing the same effects falls, as the number of independent simulation experiments rises. Still simpler is the case where an approach has already been verified. In this case verification is established directly by means of a sufficient number of experiments, and a comparison between the model that has already been verified and the model to be verified. We see from this that absolute verification remains limited to a very small number of fields of application. In all other cases it is much more a case of deciding how many experiments must be performed before we are prepared to regard a model as having been verified. In this context, moreover, the required degree of correspondence, and consequently the accuracy of the model, has to be defined in advance.

Let us now illustrate this verification procedure on the basis of a few examples. We can use a logic simulator for the simulation of digital circuits, or — when considering the underlying transistor circuit — a circuit simulator can also be used. In principle, both simulators should deliver the same results, with the circuit simulator giving greater accuracy at a higher cost as a result of its analogue consideration method.

For simplified applications it is often possible to put forward analytical solutions that can be used for verification purposes. An example of this is the mechanical deformation of a rectangular or round plate under load, which can be calculated very simply in the form of an analytical equation. The resulting elastic line provides a starting point for the verification of the implementation of finite, mechanical elements.

Verification based upon visual inspection and animation

Another important verification method is the visual inspection ('eyeballing', see Kleijnen [193]) of the sequence of a simulation using a debugger or comparable tool. Simulators for hardware description languages often offer the use of such tools, which permit the representation of sequential modelling code as it is processed. Other forms of visualisation are represented by marks in Petri nets or the current state in state diagrams. However, visualisation can be used not only for the evaluation of the simulation process, but also for the representation of the simulation results. This is also vital because the resulting columns of figures are generally unsuitable for providing an overview of the system behaviour. The simplest and most widespread form is the x/y diagram, the x-axis of which is often time. In the field of electronic circuits this is usually sufficient. However, for the evaluation of mechanical behaviour, this is often not the case. In such cases animation procedures facilitate a better evaluation of the simulation results and thus better verification. It is self-evident that the animation, like any other tool to aid understanding of a model, also makes a contribution to validation, but this is not the subject of the present chapter.

Verification of the runtime behaviour

Occasionally tools are used that identify those parts of a model that contribute significantly to the running time. The classic approach to this is to determine the instruction currently being processed at regular intervals. This sampling allows us to obtain statistical information on the frequency of the execution of instructions and modules. This is entirely sufficient for the given purpose, but does not overload the running time of the programme under investigation. The information extracted can be used to selectively accelerate a model, which is of decisive importance particularly for more complex models which already have considerable running times.

Formal verification

Formal verification will be considered here from the point of view of formal methods for the verification of digital circuits originating from microelectronics. Since the design of digital circuits is increasingly based upon modelling in hardware description languages, we can no longer differentiate the verification methods for the designs from the verification of the corresponding models. Now if the design and simulation models are exactly the same, there is no need for verification. Occasionally, however, models have also been specially prepared for the simulation, which may be necessary for performance reasons. In this case it may be useful to perform a formal verification. This can be divided into two main fields: 'equivalence checking' and 'model checking'.

In the first case we are concerned with the functional comparison of a description with a reference description. One example could be the comparison between a gate net list and a reference model on register-transfer level, which has been intensively simulated during the design process. This largely corresponds with the verification based upon alternative models. However, in this case simulation results are not compared, as is the case for the alternative model verification. Instead formal, mathematical methods are used to find proof of equivalence.

'Model checking', on the other hand, is concerned with using mathematical methods to verify certain predictions about a circuit. So, for example, for a traffic light circuit you could exclude the possibility of all sides showing a green light [211]. This is based upon the automatic construction of a formalised proof for the prediction in question. A similar principle is followed by Damm *et al.* in [77] for the formal verification of state diagrams of automotive systems. 'Model checking' can also be used for the validation of a model.

2.5.3 Model validation

Introduction

The validity of a model is always partially dependent upon the desired applications. This is clearly illustrated by the validation criteria listed below, see also Murray-Smith [288]:

Empirical validity Correspondence between measurements and simulations.

Theoretical validity Consistency of a model with accepted theories.

Pragmatic validity Capability of the model to fulfil the desired purpose, e.g. as part of a regulator.

Heuristic validity Potential for testing hypotheses, for the explanation of phenomena and for the discovery of relationships.

These different validation requirements are the reason for the development of a whole range of validation strategies. In addition to the methods presented in the following sections there are also a few basic strategies that improve the degree to which models can be validated. In general, simpler models are easier to handle, and thus also easier to validate. In some cases it is also a good idea to take the model apart and then validate only the components and their connection together. Finally, it is occasionally worthwhile to selectively improve the quantity and quality of the measured data from the real system, which can, for example, be achieved by a design of the experiment layout that is tailored to the problem.

Direct validation based upon measured data

Validation should ensure the correspondence between the executable model and reality. To achieve this it is necessary to take measurements on real systems in order to compare these with the results of a simulation. Models are often used to obtain predictions about the future behaviour of a system. If this model is predictively valid, it follows that the predictions are correct in relation to reality. However, the reverse is not necessarily true! It is quite possible for faulty models to produce correct predictions by coincidence. So we cannot say that a model is valid on the basis of simulation experiments, but at best that the model is not valid if false predictions are made. In principle a greater number of simulation experiments does not change the situation. Only the probability that the model is predictively valid increases with the number of experiments.

The possibility of performing experiments in reality and recording their results by measurement is limited. Correspondingly, the available data tends to be scarce in some cases. As a result of the lack of support points, this can cause difficulties in validation. But the opposite case can also lead to problems. If plenty of measurement data is available, a great deal of effort is occasionally necessary to extract the relevant content from the data.

An initial clue is provided by the visual comparison of measured data and simulation results in order to ensure that the input data of the model is represented as precisely as possible in the simulation. Furthermore, a whole range of measured variables can be used to check the correspondence between measured data and simulation results. So it is possible, as demonstrated by Murray-Smith in [289], to define various Q functions for the time-discrete case, which represents a degree of correspondence between the measured response z_i and the result of the simulation y_i. The following formula shows the first possibility:

$$Q_1 = \sum_{i=1}^{n} (y_i - z_i) \cdot w_i \cdot (y_i - z_i) \tag{2.8}$$

where w_i denotes weight. This formula can also be viewed as a weighted variant of equation (2.5). Another possibility is to use Q_2 to define a normalised degree of inequality:

$$Q_2 = \sqrt{\sum_{i=1}^{n} (y_i - z_i)^2} \Bigg/ \sqrt{\sum_{i=1}^{n} y_i^2} + \sqrt{\sum_{i=1}^{n} z_i^2} \tag{2.9}$$

The values of Q_2 lie between zero and one, with values close to one indicating a high level of inequality and values close to zero indicating a high level of equality between measurement and simulation. A further approach is recreated in the target functions of simulated annealing and genetic algorithms:

$$Q_3 = \frac{1}{1 + \frac{1}{n}\sum_{i=1}^{n}(y_i - z_i)^2} \qquad (2.10)$$

In this case values close to one indicate a good correspondence and lower values indicate a correspondingly poorer agreement.

Although these measures achieve a significantly better quantification of the correspondence between measurement and simulation than the visual comparison, unresolved problems remain. For example, in some cases it is worthwhile to derive the individual values and draw upon general properties for comparison. One possibility is to make a comparison over the frequency range instead of over time, see Murray-Smith [289].

Validation based upon a system identification

One significant criterion for the validation of a model is how well or badly it can be identified, see previous section on parameter estimation and system identification. Cobelli *et al.* [72] classify the validation methods as identifiable and nonidentifiable models, whereby the former is described as the simpler and the latter as the more complex model. The applications considered stem from the field of physiology and medicine.

If a model is clearly identifiable then the procedure of parameter estimation can be used to validate a predetermined model structure. In the first step the parameters of the model are identified to minimise the difference between measured and simulated data. Then the following information can be obtained about the validity of the model structure:

A high standard deviation of the estimated parameters in the identification for various sets of measured data can indicate an invalid model, but it can also indicate non-negligible measurement errors.

Systematic deficits in the approximation of the measured values by the simulation indicate that the structure of the model does not correctly reflect reality.

Conversely, differences between identified and any known, nominal parameters can be evaluated. This is particularly interesting if the variance of the individual parameter estimates is known.

Furthermore, it is also possible to subject the identified parameters to a plausibility analysis. In this connection, all available information on the system should be used to discover inconsistencies in the identified parameters.

Most procedures and tools for system identification are only suitable for linear models. Furthermore, various aspects of even nonlinear models can be considered if a linearisation is performed.

Validation based upon the 'model distortion' approach

The 'model distortion' approach, see Butterfield [54] and Cameron [58], is similar to validation by identification. The main idea behind this is to calculate the 'distortion' of parameters necessary to obtain simulation results that precisely correspond with the measurements for every point in time. The gap between nominal parameters and the newly determined parameters, which alters from one moment to the next, becomes a measure for the quality of the model. In particular, it is possible to investigate whether these new parameters lie within an accepted variation of the nominal parameters. Once again, measuring precision is a problem in this approach, and this can significantly limit the value of the possible predictions. The 'model distortion' approach was originally used for the validation of models for heavy water reactors.

Validation based upon a sensitivity analysis

It is not generally possible to precisely determine the value of the parameters of a simulation model. However, it is almost always possible to define intervals within which the value of a parameter always lies. The value of a model is questionable if the variation of a parameter within the interval leads to significant variations in the simulation results. This is generally because parameters enter the model behaviour in nonlinear form. In such cases, sensitivity analysis can supply important indications of validity problems, see Kleijnen [193]. In the simplest case, the sensitivity S is determined using the perturbation method for a property of the circuit F and a parameter P, by varying the parameter by ΔP and evaluating the change in the circuit value ΔF:

$$S = \frac{\partial F}{\partial P} \approx \frac{\Delta F}{\Delta P} \tag{2.11}$$

It is often worthwhile to standardise the sensitivity in this connection:

$$S = \frac{\partial F/F}{\partial P/P} \approx \frac{P \cdot \Delta F}{F \cdot \Delta P} \tag{2.12}$$

However, this can lead to problems if F or P are close or equal to zero.

Validation based upon a Monte-Carlo simulation

The sensitivity analysis described in the previous section allows us to investigate the effects of a parameter or possibly to set the individual sensitivities of several parameters off against each other. Now the parameters and their variations are not independent of each other with regard to their effect upon the events of the simulation. On the other hand, for reasons related to the running time it is not possible to itemise all combinations of parameter variations and subject each to a sensitivity analysis. Nevertheless, in order to do justice to these cross-sensitivities to some degree we can predetermine intervals and statistical distributions for the 'suspect' parameters and run a large number of simulations, each with statistically dispersive parameters. However, we cannot prove the validity of the simulation in this manner, we can only say that the check has failed, or has not failed, after a certain number of experiments. In the former case the matter is clear, in the second the risk of the failure of validity has in any case been reduced. For this reason, this method is also called risk analysis by Kleijnen [193]. The methodology described is already built into many circuit simulators. It is generally not used for the validation of models, but for the evaluation of the yield of fabricated circuits taking into consideration the component tolerances.

Validation based upon model hierarchy

This method aims to achieve the validation of a model based upon the validation of its components, whereby the interconnection of the components occurs directly within the model and thus is noncritical in relation to validation.

A simple example of this is the validation of the model of a circuit, where this is described in the form of a net list of components such as transistors, diodes, etc. If we assume that the net list represents the actual connection structure of the circuit, then the validation of the circuit model is transformed into the validation of the component model. If only a few component types are used, which can be individually modified by parameterisation to give the desired component, then the validation of all circuits created from these components requires only one validation of the component model. Thus the validation of circuit models can in principle be considered as having been solved. The only further point of interest is the consideration of macromodels for circuit blocks such as operational amplifiers, which offer advantages in terms of simulation speed due to more abstract modelling.

A similar approach is also followed in the object-oriented modelling of multibody systems or in the creation of block-oriented models for control engineering, although the diversity of basic models is significantly greater in these cases. An example of this is the 'open loop' simulation method described by Gray and Murray-Smith [123], in which a system model is broken down into component models, which are each individually simulated with real measured data at the inputs. An example application for this is the rotor dynamics of a helicopter.

Validation based upon inverse models

In [44] Bradley *et al.* consider the modelling of a helicopter. To validate the developed model, flight trials are performed in which the pilot has to perform a predetermined manoeuvre. His control inputs are used as the stimuli for the simulation. A validation of the model cannot be achieved for certain manoeuvres because the pilot and helicopter form a control loop in which even the smallest deviations quickly accumulate to form large discrepancies between reality and simulation. His measured control movements are correct only for reality. In order to achieve a validation nevertheless, Bradley *et al.* propose to consider also the inverse of the simulation. In this case the desired flight movements are predetermined. An inverse model in the form of an ideal pilot calculates the necessary control of the helicopter. This avoids the accumulation of faults described above. Thus the validity of the helicopter model is demonstrated on the basis of outputs supplied from the inputs generated using the inverse model. The criteria from the previous section on direct validation based upon measured data, can again be applied here.

2.6 Model Simplification

In some cases the precision of some (sub)models is greater than is necessary for the purposes of the simulation. This is not critical as long as the efficiency of the simulation is not a problem. However, if the simulation times become too great then it makes sense to consider the simplification of models, see for example Kortüm and Troch [203] or Zeigler [435]. According to Zeigler the following strategies can be drawn upon to achieve the simplification of a basic model:

- Omission of components, variables and/or interaction rules.
- Replacement of deterministic descriptions by stochastic descriptions.
- Coarsening the value range of variables.
- Grouping of components into blocks and combining the associated variables.

The first method assumes that not all factors are equally important for the determination of the behaviour of a model. Typically, these factors are classified as first and second-order effects. The behaviour of a model usually depends primarily upon a few first-order effects, whilst the second-order effects, although numerous, can generally be neglected without significantly detracting from the validity of the resulting model. Here too the principle applies that the validity of a model is always established from the point of view of the application. A further difficulty is that the omission of components, variables or interaction rules can have side effects for other parts of the model. For example, an eliminated variable may leave behind gaps in various interaction rules, which each need to be carefully closed. This process is not trivial.

The second principle is based upon the observation that in many cases a stochastic formulation is significantly more simple to create than a complete deterministic description. Thus, in the investigation of the performance of a computer, for example, a proportionately weighted mix of instructions is used, instead of considering individual programmes and their sequence.

The third point recommends the coarsening of the value range of variables, such as occurs in electronics in the transition from an analogue to a digital consideration. In this approach, the variables, and of course also the components and interaction rules, are initially retained. But one value now covers a whole value interval in the original model, the individual value of which can no longer be activated. This may lead to changes in the formulation of interaction rules.

Finally, the fourth principle is based upon the combination of components and variables. For example, the distortion of a capacitive pressure element in space can be determined by a large number of positional variables. From an electrical point of view, however, only the resulting capacitance of the structure is of interest and not the strain. The capacitance, on the other hand, represents a single numeric value which is, however, partially determined by the mechanical strain.

All these methods thus serve to obtain a simulatable description from the more theoretical basis of a conceptual model without, in the process, losing the validity of the application cases of interest.

2.7 Simulators and Simulation

2.7.1 Introduction

The models introduced in the previous sections can be automatically evaluated in numerous ways. This is called simulation.[3] Before electronics came into being, attempts were made to construct mechanical equipment that displayed the same relationships between the variables as was the case in the model. Worth mentioning in this context is, for example, the tide prediction device (1879) by Lord Kelvin or the mechanical differential analyzer (1930) by Vannevar Busch. After the second world war the development of electronics resulted in the analogue computer, which was successfully implemented in the aircraft industry, for example. The field of simulation gained new impetus with the introduction of the digital computer, which brought the advantage that adaptation to a new simulation problem did not require changes to the hardware, but only different software. Today we differentiate between a whole range of simulator classes in the field of application of mechatronics and micromechatronics, the most important of which are listed in Table 2.1.

2.7.2 Circuit simulation

A circuit simulation considers networks of components such as transistors, diodes, resistors, capacitors, coils, etc. The variables that are of interest here are generally voltages and currents. These are represented in continuous form. Nonlinear, differential-algebraic equation systems have to be solved, which arise as a result

[3] The word simulation is derived from the Latin verb simulare, which means to feign.

Table 2.1 Classes of simulators for mechatronic systems

Simulator class	Elements considered
Circuit simulator	Circuits made up of electronic components, e.g. transistors, resistors, capacitors, coils, etc. and analogue hardware description languages
Logic simulator	Logic gates, e.g. AND, OR, NAND, NOR, XOR, etc., plus digital hardware description languages
Block diagram simulator	Block diagram of control technology
Multibody simulator	Bodies with mass and inertia moments, joints, springs, dampers, actuators, sensors, etc.
FE simulator	Finite elements for the description of a mechanical continuum
Software simulator	Programs in assembler and in higher programming languages

of the structure of the circuit. The most important procedure in this context is the modified nodal analysis. As the name suggests, nodal analysis considers the node voltages as an unknown. The important point here is that the number of node voltages, and thus the number of equations, is typically significantly higher than the number of degrees of freedom.

The process for drawing up the equation system begins with the generation of the equations for each branch, e.g. for each component in the circuit. Then there is the adjacency matrix, which describes the connection structure of the circuit and thus the relationship between branch and node voltages. Furthermore, capacitances and inductances have to be taken into account in the form of a numeric integration. The procedures of Gear, trapezoidal or backward Euler integration are often used here. Finally, the nonlinear components, such as transistors and diodes must be taken into account by bringing about a linearisation at the working point typically using the Newton–Raphson procedure.

2.7.3 Logic simulation

It is often not possible to perform circuit simulation for larger circuits[4] as a result of the associated cost. If we still want to analyse these circuits by simulation, sacrifices must be made in accuracy. For digital circuits it is generally possible to use a number of logical values e.g. (0,1,X,Z) instead of the continuous potentials used previously. Here X represents an unknown and Z a high-ohmic state. Nonideal signal changes are represented in the digital world by signal transitions, which are, however, subject to a time delay. Furthermore, time is no longer continuous, but is considered as discrete or event-oriented depending upon the simulator. Only in the latter case is a precise consideration of gate and block delays possible,

[4] $>10\,000$ components.

which means that virtually all logic simulators on the market currently work in an event-oriented manner.

The two main strategies for simulation are the compiled and the interpreted simulation. In the compiled simulation, the circuit is translated prior to simulation into a programme, the processing of which brings about the simulation. In the simplest case the circuit contains a few gates and no feedback loops. In this case an instruction can be provided for every gate, which applies the function of the gate at the gate's inputs and stores the result as a variable, which represents the output signal of the gate. These instructions are sorted topologically, so that all values are already calculated if they are needed in a calculation. If the circuit contains back-couplings this principle can no longer be maintained. In this case we work with two sets of variables — one for the values drawn into the calculation and one for the newly calculated values. Then we iterate until no more changes occur in the circuit.

By contrast, in the interpreted simulation, information is available for every gate about which other gates are connected to it. The idea is to not recalculate all gates afresh for each step, but only to calculate those for which the logical value of the input has changed. For all other gates nothing has changed. We start with the inputs to the circuit and evaluate the gates connected to it. In this manner, future events are generated at the gate outputs in question, which are stored in a chronologically sorted list. Based upon this list the next event in chronological terms can be determined and the associated gate calculated, which often triggers further events. In addition, further events may occur at the circuit inputs. The simulation ends when the event list is empty or a predetermined time period has passed.

2.7.4 Multibody simulation

In this context we can differentiate between two main types of mechanical simulators, see Leister and Schielen [233]. Firstly, there are the simulators that formulate the mechanics as a symbolic equation system, which can then be processed using numerical standard solution procedures. The other option is to consider the mechanics as a linear system with mass, damping and stiffness matrix. In this case, the individual coefficients of the matrices have to be determined afresh for every time increment. Both approaches have their advantages and disadvantages.

Depending upon the application under consideration, the symbolic equations may explode in size, putting them beyond any simulation. On the other hand, there are occasionally numerical advantages, because, in the case of symbolic equations, the equation solver is in possession of all relevant information about the system. This is not the case for the numerical variant because the calculation of system matrices typically tends to be independent of the differential equation solver. Finally, symbolic equations can also be used in another context, for example, optimisation.

2.7.5 Block diagram simulation

A block diagram[5] describes the structure of a system of mathematical equations in graphical form. All connections between blocks are set up so that the causality of the system can be determined in advance. This facilitates the creation of a simulator for block diagrams that explicitly builds up system equations. Thus a sequence of instructions can be processed during the simulation. This is more efficient than an implicit formulation using ‘genuine’ equations. However, the causality must be determined in advance and must not change during the course of the simulation, which may well occur, depending upon the system.

2.7.6 Finite element simulation

Every finite element is characterised by its mass, damping and elasticity matrices. These matrices are square; the number of rows and columns corresponds with the number of degrees of freedom of the element. For small deflections the movement of the mechanics can still be considered as linear. In this case it is sufficient to establish the element matrices once at the beginning of the simulation. Otherwise, the element matrices must be calculated afresh for every time increment.

In order to determine the behaviour of the entire structure, the element matrices are now converted to system matrices. If two suitable degrees of freedom of two neighbouring finite elements are linked together, then on the system level they come together into one degree of freedom. In this manner, various degrees of freedom are dispensed with on the element level. The resulting system matrices have a band structure, for which certain solution procedures, e.g. the Cholesky method, are particularly suitable.

2.7.7 Software simulation

The most obvious form of software simulation is performing it on a computer. A debugger is generally available for this, which displays the processing of programme instructions and the current variable value and outputs. The timing of processing naturally varies according to the computer used, so that at this level only functional investigations tend to be performed.

Furthermore, there are also so-called command set simulators, which consider the software processing for a certain processor at assembler level. Timing can be determined on the basis of the timing cycles that have elapsed. This is only the case, however, if access to external resources, e.g. to a hard disk, can also be precisely specified in the timing, which is rarely the case. For embedded processors such resources are often not available, which means that in this case precise values can often be obtained for the timing.

[5] See also Section 3.4.2.

2.8 Summary

This chapter has presented a cross-section of the methods for the modelling and simulation of electronics and mechanics that are currently prevalent. It has listed categories and fields of application of models. It has also taken into consideration methods for the verification, validation and simplification of models. This forms the basis for the consideration of electro-mechanical systems in the next chapter.