

2 Elements of Modeling

Background Abstract *dynamic models* play a central role in the design process for mechatronic systems. The *dynamic* behaviors and desired (or undesired) interactions of system components fundamentally define favorable (and unfavorable) product properties. The primary challenge in modeling mechatronic systems lies in their *multi-domain nature*. To the extent that *heterogeneous* physical components are interconnected in a *homogeneously* operating functional unit, models of the components must naturally also be expressed in a *domain-independent abstract structure*. Naturally, when creating an abstract model, relevant physical and dynamic properties must be correctly depicted, and the assignment of real component properties to model parameters should remain sufficiently transparent.

Contents of Chapter 2 In this chapter, a number of selected, fundamental *methodological* approaches to the *modeling* of mechatronic systems with *lumped system elements* are discussed, which, in the estimation of the author, belong to the basic tool chest of the system engineer, and which go beyond the standard subject matter of individual disciplines. *Structured analysis* delivers *qualitative* system models with a clear functional structure. For *quantitative* physical multi-domain modeling, the *LAGRANGE formalism* in a generalized formulation (an energy-based method) and a generalized *network-based approach* (a power-flow-based method) are presented, and their strengths and weaknesses are discussed. Particular attention is paid to the *modularization* of models, to which *multi-port models* are particularly suited. In addition, these models form the basis for modern *object-oriented* modeling tools. The resulting mathematical models are generally systems of *differential-algebraic equations*, the non-trivial handling of which is discussed in detail. Discontinuous behaviors of mechatronic systems can be represented using *hybrid* descriptions, for which *net-state models* are introduced as a practical standard structure. To round out the toolbox of methods, a summary review of *linearization* of nonlinear system models is included. A methodological and practical approach to the *experimental* determination of *frequency responses* concludes the theoretical concepts of modeling. ■

2.1 Systems Engineering Context

System models Abstract, mathematical models of a mechatronic system as a depiction of the real world play a central role in systems design. As a rule, these models are developed and manipulated long before the actual components of the system are available. It is on the basis of such abstract models that robust predictions of the capabilities of the (possibly yet to be produced) real system must already be made.

Experimenting with models: simulation In this context, the construction of these models—the *modeling task*—is a key task of systems design which should be carried out with the greatest degree of diligence and care. Ultimately, product capabilities of interest are always stated in terms of statistics concerning system performance at a *specified time* or over a defined *time interval* (e.g. the nominal operating period). Such time-based performance properties can be determined via *simulation*, that is, through experiments on available models. As a part of this process, each experiment should be set up in a clearly verifiable manner with an *experiment frame* ε (the experimental conditions) and assessable system responses $y(t)$. Such experiments can be performed on the real system (ε_S, y_S) or a model of the real system (ε_M, y_M) via simulation, see Fig. 2.1.

To perform a simulation experiment, the mathematical model must be “animated” in such a way that it becomes possible to calculate the time history of all outputs that are of interest (Fig. 2.2).

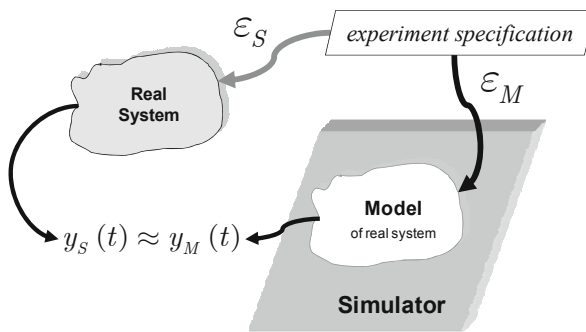


Fig. 2.1. Simulation as experimentation on models

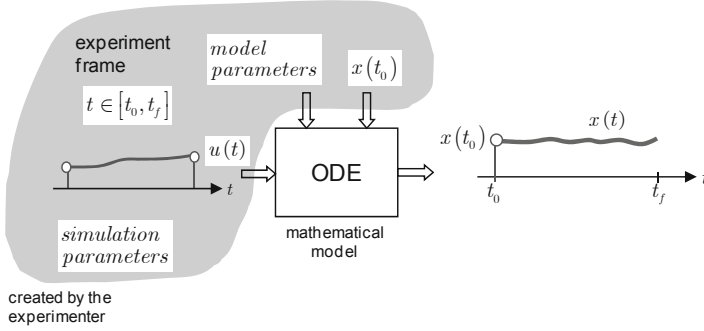


Fig. 2.2. Simulation experiment on an abstract mathematical model of an illustrative system with one input $u(t)$, one output $y(t)$, and internal states $\mathbf{x}(t)$ with initial values $\mathbf{x}(t_0)$

In general, this proceeds via the solution of a system of differential equations using appropriate numerical methods (numerical integration algorithms) which are implemented on a computational platform (digital *simulator*, Fig. 2.1). The methods used to implement simulation models inside of simulators are termed *simulation techniques*.

Validation vs. verification

Simulation experiments By performing a simulation experiment, it becomes possible to predict the behavior $y_S^i(t)$ of the real system for *one* specific experiment ε_S^i , using the simulation solution $y_M^i(t)$ as the result of *one* equivalent simulation experiment ε_M^i . The following must be kept in mind in this context: the comparability of $y_S^i(t)$ and $y_M^i(t)$ depends both on the model accuracy and on the concrete computational implementation of the mathematical model. The predictive capability of the simulation results should thus always be *critically* scrutinized: “Does my model consider all properties important to me?”, “How are my model equations actually implemented in the simulator?”, “Which method-dependent approximation errors are entailed by the solution algorithms employed?”, “What numerical errors result from the concrete implementation on the chosen computational platform?”. All of these questions *predictably* influence the accuracy of the simulation, i.e. the best possible equivalence $y_S^i(t) \approx y_M^i(t)$ under the chosen simulation boundary conditions.

In confirming the correctness of models, a distinction is made—depending on the types of the models—between the terms *verification* and *validation*¹.

Definition 2.1. *validation* – (IEEE 1997) “Validation is the process of determining the degree to which a simulation is an accurate representation of the real world from the perspective of the intended use(s) as defined by the requirements.”

Definition 2.2. *verification* – (IEEE 1997) “Verification is the process of determining that an implementation of a simulation accurately represents the developers conceptual description and specifications.”

In summary, the definitions can be expressed as follows:

- *validation* = “Have I created the *correct model*?”
- *verification* = “Have I *correctly created* (implemented) the model?”

The three fundamental steps of verification and validation (V&V) in the context of model creation are further discussed below.

Experimental model validation If sufficiently meaningful results regarding the output $y_s^i(t)$ of the real system are available for comparison, *experimental validation of mathematical models* is possible based on simulation (Fig. 2.3). Non-trivial questions in this context include: “Where do the comparison results come from, given an as yet non-existent system?”, “How many simulation experiments are really sufficient for validation?”.

Verification of simulation models For model validation, an implied premise is that the mathematical models, including the accompanying experiment frame, have been correctly implemented in the employed simulation model. Assessing the correctness of the implementation is termed *verification of the simulation model* (Fig. 2.3). Verification proceeds by comparing simulation results $y_M^i(t)$ with significant reference data $y_s^i(t)$. Predictions of system behavior which can be obtained through *analytical* consideration of the underlying mathematical models are particularly well-suited to this purpose, e.g. steady-state values from limiting values of a

¹ The terms “verification” and “validation” are unfortunately understood quite differently in different technical communities. The interpretations and definitions presented here follow international standards which have generally proven valuable in industrial fields designing complex (mechatronic) systems, e.g. aerospace, for many years (ESA 1995), (IEEE 1998).

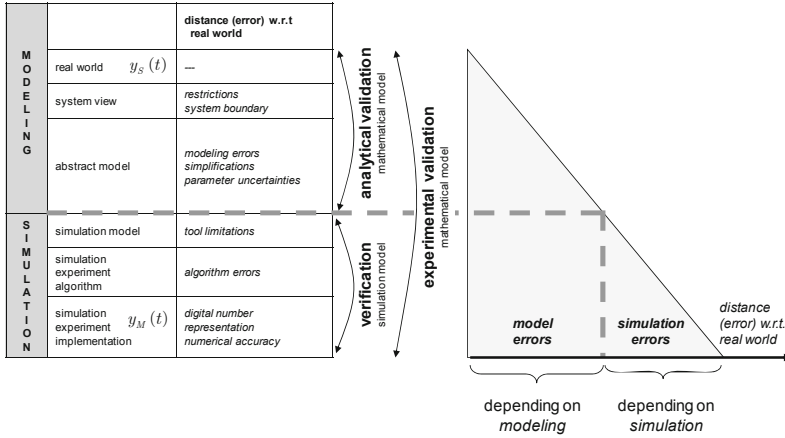


Fig. 2.3. Verification & validation vs. modeling & simulation

LAPLACE transform, oscillatory dynamics under harmonic excitation from the frequency response, angular momentum from conservation principles in mechanical systems. For any such analytical prediction, appropriate test cases (experimental frames) should be created. The deeper the theoretical understanding of the system incorporated at this point, the greater the chance of generating an accurate simulation model².

Analytical model validation Analytical predictions of system behavior bring forth the possibility of *analytical validation* of mathematical system models, i.e. a direct comparison with real system data. This can happen independently of, or complementary to, experimental validation (Fig. 2.3).

A further strength of *analytically*-based predictions is that, in the best case, properties with general validity can be derived for a large class of experiment frames.

A good example is provided by *stability* predictions for mechatronic systems. Experimental confirmation of system linearity and time-invariance within a few characteristic experiment frames permits general predictions for arbitrary inputs in terms of *bounded-input bounded-output (BIBO)* stability within a particular operational regime. Conversely, *one single* simulation experiment only allows inference about that *specific, unique* experimental frame. Using analytical predictions can thus signifi-

² The converse holds equally, i.e. carelessness and ignorance are the enemies of reliable systems design!

cantly reduce the costs and complexity of verification and validation (directly affecting design time and cost).

Model variants

Model hierarchy In the domain of systems design, a variety of types of models are employed, representing differing perspectives on the mechatronic system under consideration (see the model hierarchy in Fig. 2.4). Each perspective describes certain differing properties of the same mechatronic system, similar to how a cuboid can be viewed from different sides.

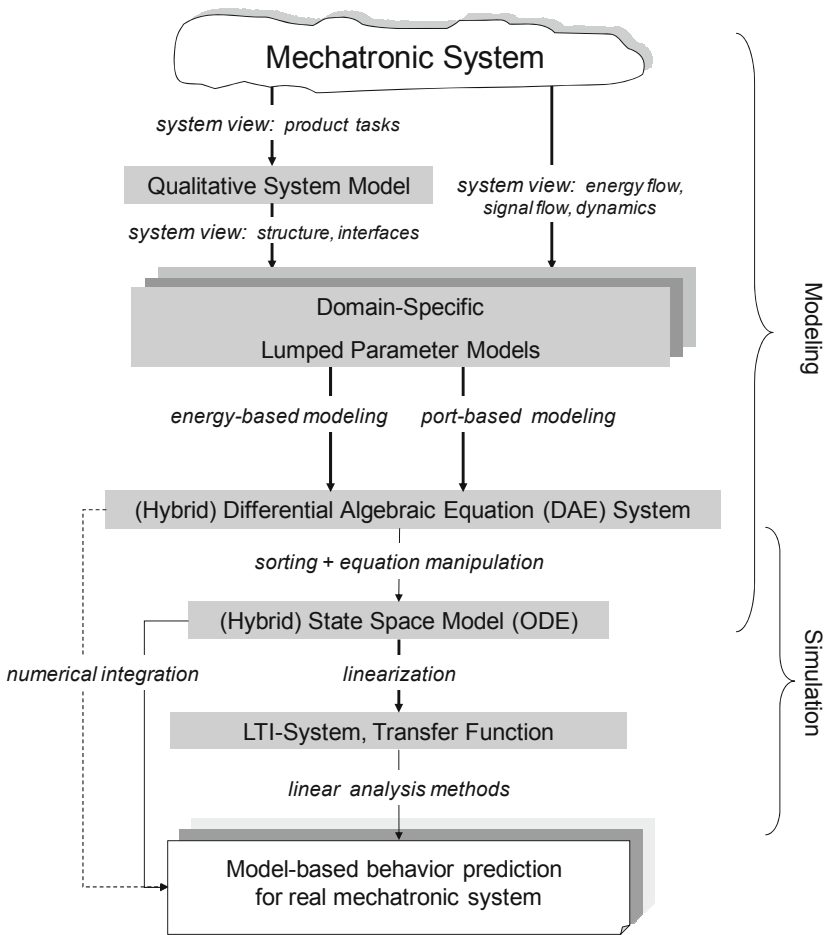


Fig. 2.4. Model hierarchy for the design of mechatronic systems

Qualitative system model At the most abstract level, a system can be described using purely *qualitative attributes*, resulting in a qualitative system model (Fig. 2.4. upper left, see also introductory example Fig. 1.17). In a mechatronic system, important aspects of the model include defining the behavior of the system with respect to the environment (user) and assigning product tasks to realizing “functions” (in the sense of “accomplish task”). At the same time, a preliminary functional system structure and important functional interfaces should be defined (see Sec. 2.2).

Quantitative models For quantitative predictions of system behavior, mathematical models suitable for computation must be generated. To accomplish this, attention should be paid to energy flow, signal flow, and the dynamics within and between the functions defined in the qualitative model. The challenge in mechatronic systems lies in the variety of physical domains involved (electrical, mechanical, hydraulic, thermal, etc.). As a result, a broad technical understanding of the different domains must be present to form a firm basis for modeling.

When creating a model, attention must be paid to the fact that interactions between system elements from different physical domains always take place via energy flows with power back-effects. The resulting *heterogeneously-coupled system behavior* should be captured in *domain-independent models*.

Modeling paradigms For modeling in multiple physical domains, there exist only a few suitable modeling methodologies. This book considers system descriptions with *lumped system elements*, and two methodological approaches particularly well-suited to this formulation are presented in more detail. First, *energy-based* modeling using the *LAGRANGE formalism* is well-suited to smaller systems with nonlinear equations, and can be easily worked out by hand. *Multi-port* modeling methods, on the other hand, are particularly appropriate for very large systems and for computer modeling. Combined with *object-oriented* concepts, multi-port methods nowadays offer an attractive array of efficient computational tools for multi-domain modeling and simulation.

Mathematical system model The end results of the various modeling methodologies are so-called *differential-algebraic equations (DAE)*. These are systems of (generally nonlinear) differential and algebraic equations involving relevant system variables (Fig. 2.4 center). They are the final,

actual mathematical model formulation—a DAE system presents a domain-independent mathematical model in which all physical phenomena are represented. Predictions of system behavior are obtained via appropriate experimentation on the DAE system model. Solving such DAE systems in the general case is, however, (very) difficult, so that a DAE system model must generally be manipulated so as to enable computations based on it (e.g. converting it to state-space representation, possibly linearization). Note that certain special phenomena—such as switching operations, mechanical contact problems, stiction, and discrete-time (primarily in the context of information processing) and discrete-event phenomena—should be modeled by extending the model into a *hybrid system*. Working with DAEs and hybrid systems is elaborated upon in the following sections.

Model variants

Model purpose, model accuracy One significant task in systems design is to incorporate the “correct” abstraction and simplification of the real physical behavior of a system under examination into a mathematical model. There are no set rules for this process; rather, it requires experience and engineering judgment. In the end, the idea is to separate significant aspects from the negligible. For example, the question of whether to consider parasitic effects such as electrical line resistance or mechanical friction in the model always depends on the purpose of the model. For this reason, in the context of design, there will generally be several mathematical models with *differing model accuracy* and *fidelity* for a single mechatronic system.

Low-fidelity models For controller design, stochastic performance predictions, or a rough design (feasibility studies), *simplified* or *low-fidelity models* are typically employed. This type of model is generally a linear time-invariant model (LTI system) of low order. In the case of multibody systems, these consider only a select subset of the mechanical structural frequencies (eigenmodes). Implementing such models to produce output time histories in simulation presents hardly any problems using standard algorithms, and is thus not further discussed here.

High-fidelity models On the other hand, for design verification and validation (e.g. sensitivity analysis with differing experiment frames, statistical analysis via Monte-Carlo simulation), *detailed, high-fidelity models* with the smallest possible modeling errors (Fig. 2.3) should be used to the

greatest extent possible. Such models generally take into account all relevant nonlinearities, broadband dynamic system behavior, and, in particular, high-frequency structural modes. This results in models of (very) high order and considerable complexity. Additional issues in the treatment of models, e.g. the modularity of simulation models, play a key role here. *High-fidelity* models pose some of the greatest challenges to simulation technology, and can often only be *accurately* simulated by taking special measures. For this reason, Ch. 3 is devoted to certain specialized methods and solution approaches for simulating mechatronic systems.

2.2 System Modeling with Structured Analysis

Goal This section introduces several fundamental elements of *systems analysis and modeling*. These formal tools permit the *structuring* of a system in such a way that it becomes manageable and that its *inner relationships* become visible to the engineer. Exposing system structures is a task which is generally performed without requiring mathematical formulae, but which should nonetheless not be underestimated.

Qualitative system models Due to the complexity of some systems, developing a system model is generally a non-trivial task with an uncertain outcome (there are infinite solutions, depending on which system properties are considered). Though *qualitative system models* do not allow for numerical computation, they do enable—in addition to the recognition of fundamental system relations (causality loops, dynamics)—the preliminary establishment of a foundation for quantitative (mathematical) models by setting up clearly-delineated and manageable subsystems. On the basis of such structured system models, design variants of the device can be developed. Only once such device variants are available is it even possible to engage in the creation of a physical (i.e. quantitative) model. Indeed, it is only at this point that the candidate devices can start to be considered (for example, electro-mechanical vs. servo-hydraulic actuation). It is only in textbooks that standard problem formulations of the type “given:, find:” exist.

Top-down modeling The workflow described above belongs to the so-called *top-down* system modeling paradigm, and always takes place at the beginning of the product design process. During the definition of requirements, the description of properties of the product under development is made more and more detailed following to the above procedure. Today,

techniques used during this process often come from the realm of software development and can be divided into *function-oriented* and *object-oriented* modeling methods (Vogel-Heuser 2003).

Function-oriented models *Function-oriented* modeling methods offer a natural approach for the design of *mechatronic systems*, as they are centered on workflows and input/output relations in a conventional way. Thus, following a few fundamental definitions, several elements of modeling via *structured analysis (SA)*—which are practical for qualitative modeling due to their intuitive simplicity—are presented below.



Fig. 2.5. Colorful definition of the term “system” (adopted from (Yourdon 1989) with kind permission of Ed Yourdon)

2.2.1 Definitions

Definition 2.3. *System* – (Cellier 1991) “A system is characterized by the fact that we can say what belongs to it and what does not, and by the fact that we can specify how it interacts with its environment. System definitions can furthermore be hierarchical. We can take the piece from before, cut out a yet smaller part of it, and we have a new ‘system’.” See Fig. 2.5.

Definition 2.4. *System* – (Schnieder 1999) “A system is marked by the presence of certain properties and is characterized by the following four *axioms*:

- **Principle of Structure** The system consists of a quantity of parts, which have mutual relationships to each other and the (system) environment. The system has reciprocal influences with its environment via physical quantities describing the energy, mass and information state of the system.
- **Principle of Decomposition** The system consists of a quantity of parts, which can further be decomposed into a number of mutually influencing sub-parts. When examined in detail, the sub-parts in turn exhibit a certain complexity or general system characteristics.

- **Principle of Causality** The system consist of a quantity of parts, whose mutual relationships and own variations are clearly defined in themselves. Following a causal interrelationship, later states can only depend on previous ones. Causality is understood as the logic of events.
- **Principle of Temporality** The system consists of a quantity of parts, whose structure and state to a greater or lesser extent determine changes occurring over time. Temporality is the sequence of events and variations over time.”

These two definitions are, in the end, equivalent, though the second definition also brings into play the temporal aspect important for physical systems. In the context of model creation, the principle of causality is unfortunately sometimes cited in the wrong setting; Sec. 2.3.8 further clarifies this terminology.

2.2.2 Ordering principles

Complex systems One significant task of system modeling is the ordering of the various components of a system. Complex systems are understood to be those consisting of a great number of components. Experience shows that humans are limited in the number of elements (graphic symbols) they are capable of simultaneously recognizing and understanding the contents (the semantics) of in a depiction. The important design principles of *structuring*, *decomposition*, *aggregation*, and *hierarchy* allow the designer to limit the number of system components under consideration at any time in the face of increasing model detail; these principles are described below.

Structuring

- establishing the relationships between entities in a system according to given criteria,
- deconstructing a given system according to given criteria so that its relationships become recognizable.

Decomposition

- “breaking into fundamental elements”,
- systems are *decomposed* into subsystems,
- more details are revealed in an existing model,
- refinement of a structure.

Aggregation³

- “layering together of individual elements”,
- subsystems are *aggregated* into a system.

³ Aggregation is the opposite of decomposition.

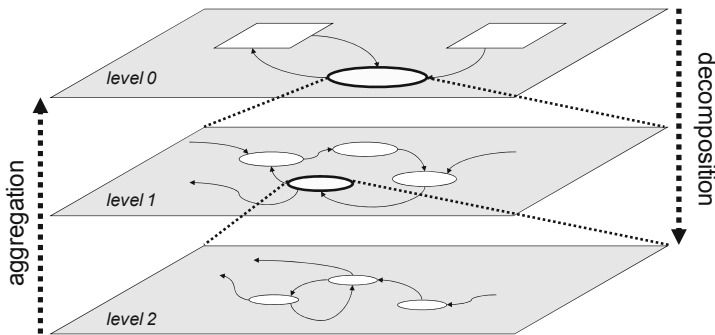


Fig. 2.6. Hierarchical levels of a system model: *decomposition* from level 0 \rightarrow level 1, *aggregation* from level 1 \rightarrow level 0

Hierarchy

- “(pyramidal) ranking”,
- system hierarchy: system definitions are *hierarchical*, i.e. one piece of the system can be extracted from the whole and be considered as a new system (see subsystems),
- hierarchical level: a particular level of consideration of a system, generally representing a subsystem,
- top level: a global view of the system
- lower level: a detailed view of the system (view of the interior).

2.2.3 Modeling elements of structured analysis

Introductory considerations The method of *structured analysis* (SA) known from the field of software development offers a very natural approach for system modeling for the design of mechatronic systems.

This section presents a simplified variation of the YOURDON SA approach (Yourdon 1989). In the experience of the author, the modeling elements presented here completely suffice for the creation of manageable functional descriptions of mechatronic products in a short time and a clear manner, without requiring extensive methodological knowledge or specialized tools⁴.

⁴ The practical procedure presented here has been successfully employed in the author’s teaching for over 10 years as a requirement in all student projects (final projects and master’s theses).



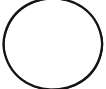


An expanded formulation—*structured analysis with real-time extension (SA/RT)* (Hatley and Pirbhai 1987; Hatley and Pirbhai 1993), (Vogel-Heuser 2003)—permits a strictly formal specification of temporal-causal system dynamics and is recommended for larger (more complex) tasks.

Function-oriented modeling Using structured analysis results in a primarily *function-oriented model* as it takes as a starting point product functions and then considers their *logically causal* interconnectedness via data and signals. Note further, that for a complete system description, structured analysis additionally models *temporally causal* relationships using state machines.

Model elements

The most important elements of function-oriented modeling are shown in Table 2.1.

Table 2.1. Model elements for structured analysis

Symbol	Property	Description
	data flow	A <i>data flow</i> represents the transport of abstract data between processes (functions)
	control flow	A <i>control flow</i> represents the transport of abstract <i>control data</i> (events) between processes (functions)
	process, function	A <i>process (function)</i> transforms input data to output data using specified rules; e.g. <i>measure speed</i> (<i>predicate</i> + object)
	store	A <i>data store</i> preserves data elements for a certain duration of time (non-volatile)
	terminator	A <i>terminator</i> represents an entity (function, device, person) which exists <u>outside</u> the system under consideration and exchanges data with it

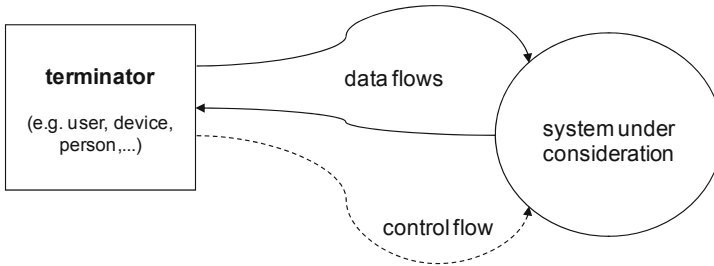


Fig. 2.7. System delineation via a data context diagram

Context diagram

Data context diagram (DCD) Context diagrams describe the system under development from the point of view of a user. The purpose of the system is summarized as a single system process. This process converts inputs from terminators/end users into outputs to terminators/end users (Fig. 2.7). The context diagram describes the interaction of the system with its environment.

Data flow diagram

Data flow diagram (DFD) Data flow diagrams are the primary tool for determining the functional properties of a system. They make structures of the system concrete by defining component functions (processes) which are tied together with data flows. A DFD contains processes, data flows and data storage locations, but no terminators (Fig. 2.8).

Process A process (also called a function, activity, or task) creates an output from an input by performing an operation. Processes have names and numbers.

Data flow Data flows represent all possible types of generalized information (signals, action flows) and can be further decomposed. Data flows can be binary, digital, or analog.

Leveling The decomposition of a parent DFD into child DFDs with an increased level of detail. The decomposition levels can have various depths.

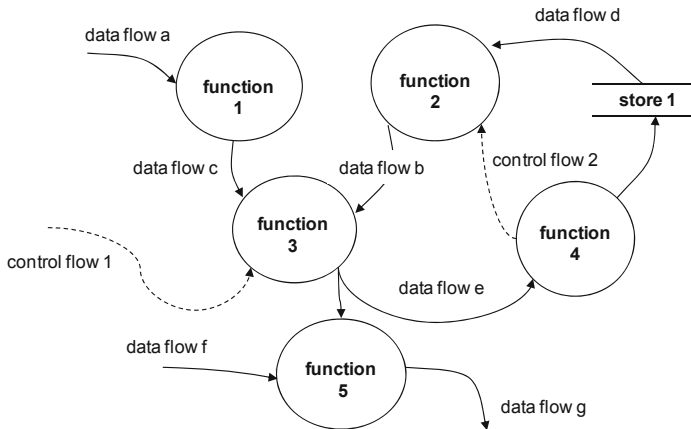


Fig. 2.8. Functional decomposition via a data flow diagram

Balancing A test for inconsistent data flows. Input and output data flows from parent and child processes must match up in a consistent manner. Data flows without a source or sink create inconsistencies which can be tested for with automated or manual procedures. For each level, preferably *five to seven* (at most ten) processes (functions) should appear (a greater number becomes unwieldy and encumbers any clear overview for the designer).

Process specification (PSPEC) A process is continuously decomposed until a short and unambiguous component description becomes possible. Possible methods for describing the process include anything elucidating its content, e.g. tables, prose description, equations, control theory transfer functions. PSPECs can appear at all levels of refinement (Fig. 2.9).

PSPEC function x.y

- textual specification
- pseudocode
- mathematical equations, etc.

Fig. 2.9. Function definition via process specification

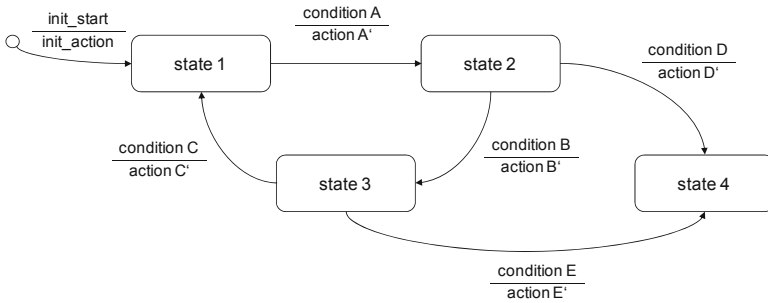


Fig. 2.10. Specification of the temporal/logical implementation of functions via a state transition diagram (here in MEALY machine notation)

Control specification

State transition diagram (STD) Control specifications describe the processing of control flows. Usually, these flows trigger state transitions or are combined with other signals to form new control signals. Typical means of description include state transition diagrams, decision tables, or prose descriptions. In general, each process has *its own* control specification. Fig. 2.10 shows an example of a state transitions diagram in the form of a *MEALY machine* (Litz 2005). When *State 1* is active, fulfillment of *Condition A* results in the activation of *State 2* (with simultaneous deactivation of *State 1*) and the carrying out of *Action A\'*.

Data handling

Data dictionary (DD) Every data and control flow—as well as all storage locations—must be defined in a data dictionary. Flows are either primitives or non-primitives, the latter consist of groupings of primitives. A dictionary is usually laid out in a computer-readable format, and can be formulated in written form, table form, or as a database.

Architecture diagram

Implementation structure An architecture diagram describes an implementation structure which realizes the functional relationships indicated in a DFD (device elements and their linkages, the *device architecture*, see Fig. 2.11). In addition, the distribution of functions among device elements and the distribution of data flows among device interfaces are documented. This gives a *semantic* description of devices, i.e. “which task(s) should the device fulfill”. Further, clear justification for the existence of any particular device in the system becomes evident within a device architecture.

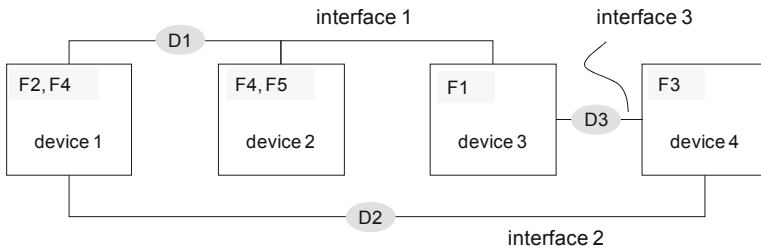


Fig. 2.11. Assignment of functions and data flows to device components in an architecture diagram

System model

A complete qualitative *structured analysis system model* is presented Fig. 2.12. In a concise, semi-formal form, this model describes different views of the system (logically causal = functional vs. temporally causal = dynamic), possesses different hierarchical levels (context, DFD level 1, DFD level 2...), and contains one or more possible device architectures (design variants) along with the assignment of device elements to their functions.

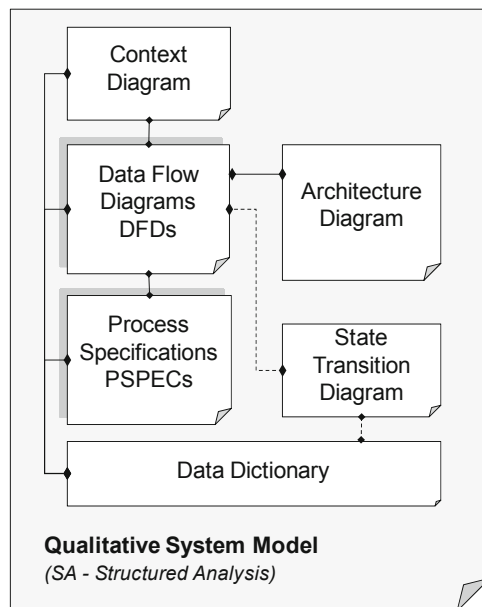


Fig. 2.12. Complete system model obtained from structured analysis

2.2.4 Example product: autofocus camera

Task definition The goal is a qualitative design model for a simple *autofocus camera*. In particular, the mechatronic aspects which enable optimal picture taking should be made evident. In addition, the assignment of functions to proposed device elements in possible physical realizations (design variants) should be highlighted. This example thus represents the typical first steps in the design of a new product.

Written product specification “The autofocus camera should be very user-friendly (target user is a photographic layperson) and have as few operational and display functions as possible. Sharp pictures should be produced without special manual intervention (industry-standard autofocus functionality). The camera should work with standard rolls of film (black-and-white, color). It should be possible to use standard rechargeable batteries. The camera should fit into the low-cost market, preferably be light weight, and enable as long an operational time as possible for every full battery charge.”

Typically, user requirements (e.g. from marketing) are given in purely verbal form, and at this point, still allow for great freedom in the design. To begin the design process, this freedom should—in cooperation with the customer (e.g. with marketing)—be further constrained using formal qualitative system models.

Context diagram Here, the most significant outside view of the product, i.e. how the user sees the camera, should already be recognizable (Fig. 2.13). Basically, the available data flows and a PSPEC of the Primary Function F0 can already be used to write a first draft of the user manual. The data flows are coded alphanumerically, in order to simplify later reference (D0.x = Level 0).

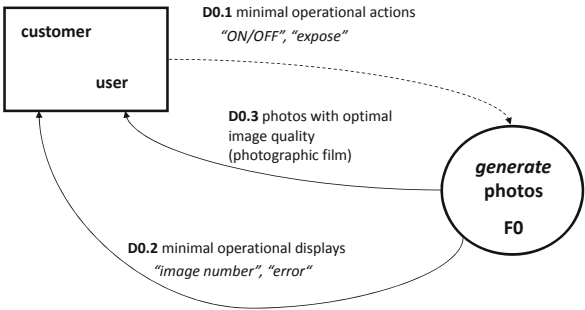


Fig. 2.13. Level 0: data context diagram for an autofocus camera

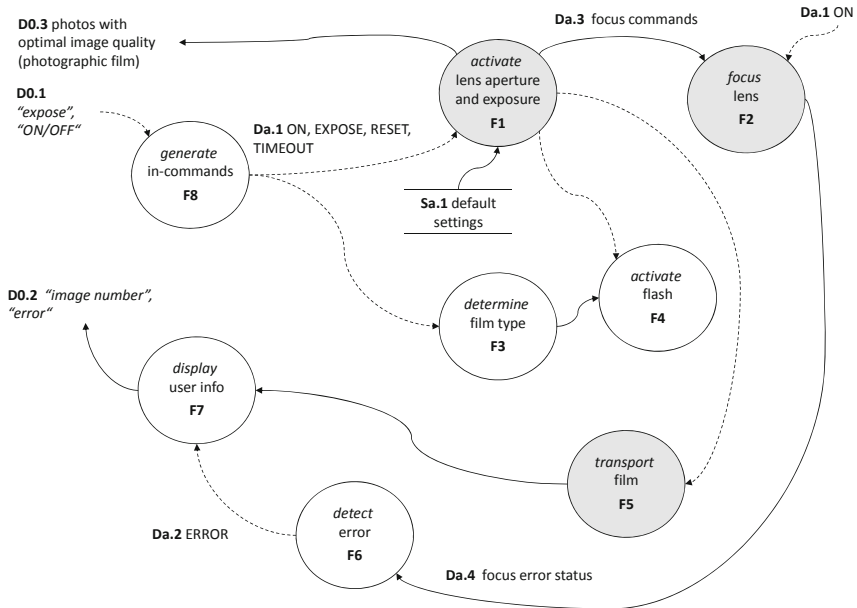


Fig. 2.14. Level a: data flow diagram for Primary Function F0 *generate_photos*; mechatronic functions in gray (labels excerpted)

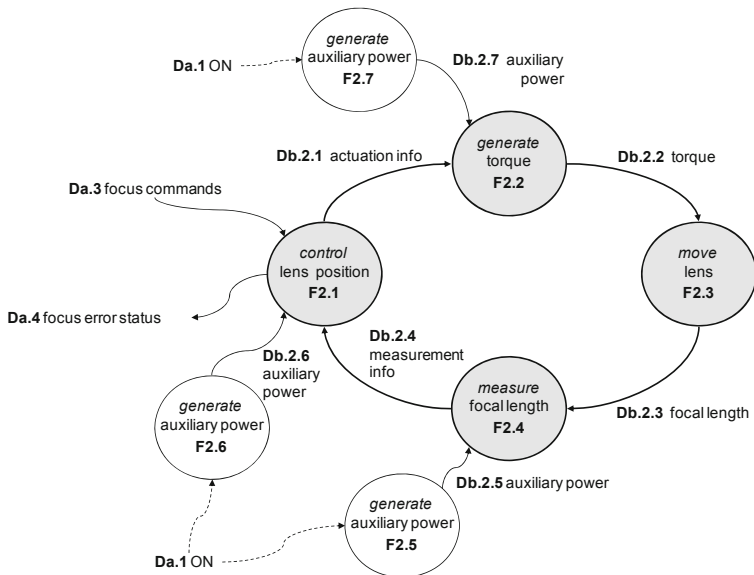


Fig. 2.15. Level b: data flow diagram for function F2 *focus_lens*; mechatronic functions in gray

Data flow diagrams Already at Level *a* (Fig. 2.14)—the *first* below the context diagram—the first candidate mechatronic functions (F1, F2, F5, i.e. those in which it is obvious that masses must be made to move in a precise manner) become evident. With eight functions, the DFD remains just readable; a larger number would render it rather unwieldy.

Of course, this decomposition is anything but unique. It is the result of a subjective perspective on the system by the design engineer, and should always be coordinated with the customer. The textual representation of data flows and functions in this model obviously eases such discussions across professional boundaries (all the way down to the layperson). The functions and data flows are coded alphanumerically, in order to simplify later references ($Da.x = \text{Level } a, Fi = \text{Function } i$).

Function F7 *focus_lens* is further decomposed at the *second* level, Level *b* (Fig. 2.15). The presence of a closed functional chain is already clearly discernable here. The coding of functions and data flows is as before ($Db.x = \text{Level } b, F2.j = \text{Subfunction } j \text{ of } F2$).

PSPECs Though the functional descriptions of the DFD elements are already readily interpretable in clear text, it is advisable to further specify the contents of the functions.

In the present case (Fig. 2.16), Function F2 *focus_lens* has a detailed written specification, which is supplemented with important numerical performance parameters. However, at this point, concrete values for the parameters are unknown, indicated by *TBD = to be defined*. Such concrete values must then be specified in subsequent steps of the design process.

State transition diagram In this diagram (Fig. 2.17), a model of the flow between operational modes becomes visible. Note that the transition conditions of the state machine must involve only available control flow signals.

PSPEC F2 *focus_lens*
The lens shall be moved automatically such that sharp images are always produced. It shall be possible to select an object in front of the lens as the distance reference.
The focal length shall be positioned with an accuracy of *TBD* [mm].
The automatic positioning shall complete in not more than *TBD* [sec].

Fig. 2.16. PSPEC for Subfunction F2 *focus_lens* (*TBD = to be defined*)

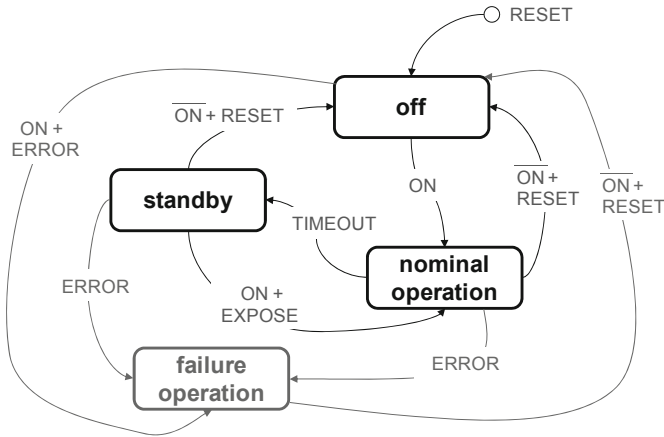


Fig. 2.17. State transition diagram for Primary Function F0 *generate_photos* (specification of the actions of the MEALY machine definition shown in Fig. 2.10 are omitted here for clarity)

Level	Data flow	Type	Description
0	D0.1 Minimal user intervention	C	operation by user
	D0.2 Minimal operation feedback	D	feedback to user
	D0.3 Optimal picture quality	D	output: exposed, safely rewound roll of film
a	Da.1 ON, SHUTTER, RESET, TIMEOUT	C	internal control signals
	Da.2 ERROR	C	internal control signal
	Da.3 Focus commands	D	default system parameters
:	:	:	:
b	Db.2.1 Set point information	D	computed set point signal
	Db.2.2 Torque	D	rotational moment on lens
	Db.2.3 Focal length	D	physical focal length
	Db.2.4 Measurement information	D	measured focal length
	Db.2.5 Auxiliary energy	D	auxiliary energy for measurement
:	:	:	:

Fig. 2.18. Data dictionary for Primary Function F0 *generate_photos* (excerpt)

Data dictionary In this concise representation (Fig. 2.18), all data flows are described in more detail (*D* = data flow, *C* = control flow).

Design variants Fig. 2.19 and Fig. 2.20 show *two* different *design variants* (alternatives) for Function F2 *focus_lens*. In the two cases, the same functional characteristics are realized using different technologies. Conceptualization of the physical implementation is completely up to the design engineer. This fact can be used to justify, for example, the design decision to use two microcontrollers.

The important aspect here is the *assignment of functions and data flows* of the DFD to *device elements*. This makes clear which tasks are fulfilled by these elements. Additionally, this directly specifies the *hardware interfaces* along with their data contents. Note that it is *only at this point* that *physical modeling* can begin (see Sec. 2.3) as it is here that *candidates for concrete device technologies*—e.g. a DC motor or piezoelectric motor—become known. The specification of particular device elements in this example is clearly based on the defined functions as a concretization of user requirements. Thus, a device architecture derived in this manner possesses a clear, justifiable basis; none of the elements “appears out of thin air”. Naturally, though, it is still necessary to factually justify any particular concrete choice. In this particular case, the DC and piezoelectric motor variants result in completely *different device properties*, e.g. device complexity (fewer elements for piezoelectric), power requirements (high-voltage for piezoelectric), motion properties, mass, cost, etc. The final choice can only take place following analysis of all performance aspects and device capabilities. However, the starting point for all of these further investigations is the qualitative functional system model.

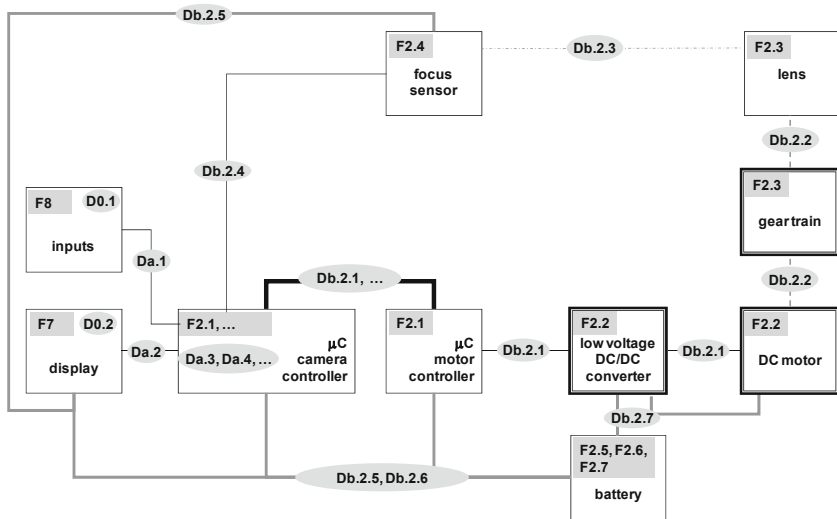


Fig. 2.19. Design Variant A for Function F2 *focus_lens* with a *DC motor and gear train*; architecture diagram with assignment of functions/data flows ↔ device technology

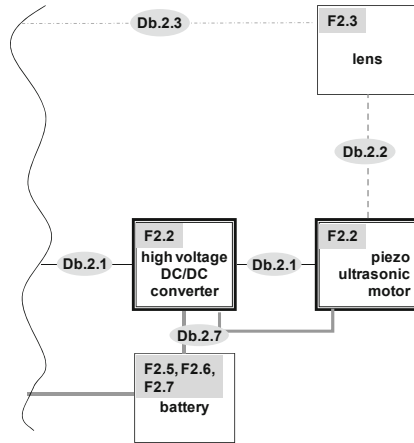


Fig. 2.20. Design Variant B for Function F2 *focus_lens* with a *direct drive piezo-electric ultrasonic motor*; architecture diagram (detail) with assignment of functions/data flows ↔ device technology

Nomenclature Structured analysis does not have a standardized nomenclature; at most in the context of computer-aided tools are there strict syntax rules which must be followed. In this example, a very simple and pragmatic nomenclature was employed, which has demonstrated its usefulness in the author's many years of experience in industry and teaching. The most important syntactic element has proven to be *alphanumeric coding*, which enables simple, concise, and unambiguous naming of model elements. This example also demonstrates quite nicely that structured analysis can be easily applied *without specialized tools*—commonly used word processing tools are thoroughly sufficient.

2.2.5 Alternative modeling methods

Object-oriented system modeling with UML

Unified Modeling Language (UML) In addition to the above-mentioned *structured analysis with real-time extensions (SA/RT)* following *Hatley* and *Pirbhai*, the only real alternative—due to its engineering-oriented approach and wide international acceptance—for *comprehensive* system modeling in the *object-oriented* paradigm is the *Unified Modeling Language (UML)* (Oestereich 2006), (Vogel-Heuser 2003).

UML model elements The semi-formal modeling elements of UML follow *object-oriented* paradigms and offer the possibility of depicting *structural* and *dynamic* system layers in the following diagram types:

- *Structure Diagrams*: Class Diagram, Object Diagram, Profile Diagram, Package Diagram,
- *Architecture Diagrams* (derived subgroup of structure diagrams): Composite Structure Diagram, Component Diagram, Subsystem Diagram, Deployment Diagram,
- *Behavior Diagrams*: Activity Diagram, Use Case Diagram, State Machine Diagram (a special type of State Machine Diagram is the so-called Protocol State Machine),
- *Interaction Diagrams* (derived from Behavior Diagrams): Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram.

UML is particularly well-suited to modeling and specification of *complex systems* and is primarily employed in software development. Since 1997, UML has been an *international standard*⁵ and is supported with an excellent selection of computer-aided tools (primarily for software design, though equally suited to general systems design). Newer extensions permit considerably improved options to specify requirements, system blocks, and allocations (see SysML⁶), as well as timing properties and performance parameters, see *Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)*⁷.

Model verification UML itself is—as are SA and SA/RT—a *semi-formal* definition, whose model descriptions are not strictly formally (analytically) verifiable, nor do they lead directly to executable models (simulation models). Though computer-based UML models can be checked for syntactic integrity with automated tools, no semantic integrity tests of any substance are possible. Thus, while such system models provide structured information about the system, they can still exhibit arbitrary inconsistencies in the logical and temporal flow. In the case of quantitative models (complex nonlinear dynamic systems, see Sec. 2.3), such a lack of comprehensive analytical verification can be circumvented by performing simulation experiments. This type of procedure has recently been more intensively stud-

⁵ www.omg.org

⁶ <http://www.sysml.org/>, September 2009

⁷ <http://www.omgmarTE.org/Specification.htm>, September 2009

ied for these more qualitative models, giving rise to so-called *executable UML models*, e.g. (Koycheva and Janschek 2007), so that in the future, the use of UML system models in simulations may become more feasible.

Application From the viewpoint of mechatronics, UML certainly places greater demands on the formal abstractive capabilities of the design engineer than does structured analysis. The function-oriented mindset of SA coincides very nicely with control theoretical viewpoints, and is thus directly usable even without further training. In order to truly be able to fully apply UML, however, training in object-oriented paradigms and practical design exercises are absolutely recommended. Familiarity with the modeling syntax and the mindset of object-oriented methods is necessary in order to meet formal requirements in the models. Thus, for an *introduction* to systematic and structured system modeling, structured analysis is to be preferred, and, in the author's experience, when applied to problems with low to moderate complexity, this latter method quickly gives very useable results.

Model-based systems design

An alternative approach—which is increasingly experiencing great popularity among mechatronics users (particularly in the automotive industry)—is so-called *model-based systems design* (Rau 2002), (Conrad et al. 2005), (Short and Pont 2008). In general, this approach does *not* encompass qualitative system modeling in the above sense, but rather quantitative *hybrid* system models, with mixed *discrete event* and *continuous* dynamics (see Sec. 2.5). For the modeling of flow-oriented properties, generalized state machines are used in combination with descriptive languages employing block diagrams (e.g. STATEFLOW / SIMULINK (Angermann et al. 2005)). As a result of the hierarchical structure of these tools, structural characteristics can be depicted with hierarchical subsystems (SIMULINK) and can be coupled with flow-oriented model components (STATEFLOW). Given consistent modeling, the descriptive elements of structured analysis (the data flow diagram and the state transition diagram) can, to a certain extent, be transformed into executable models. They thus result in executable specifications, which can be continually refined throughout the design process. For example, purely written PSPECs can be used in the form of text comments in SIMULINK blocks, and can be replaced step by step with mathematical functions (transfer functions, state machines, etc.) dur-

ing the course of the design process in an *evolution of system models*. A particularly attractive aspect of this latter concept is the possibility of directly generating code and immediately testing models in the context of *rapid prototyping*.

2.3 Modeling Paradigms for Mechatronic Systems

Modeling goals The physical modeling of mechatronic systems is predominantly characterized by the multidisciplinary (multi-domain) character of the different component systems. Naturally, the goal is a complete abstract model, representing overall system behavior—a *domain-independent model*. In this and following sections, this will be accomplished by transferring and simplifying fundamental physical model equations (differential equations, algebraic equations) into linear time-invariant (LTI) models in a frequency-domain representation (transfer functions). Using such models, a series of significant analyses of system behavior can be efficiently worked through using established (commercial) computational tools. However, to deal with more complex *high-fidelity* models, further modeling must be undertaken.

Modeling approaches Starting from the view of a system using *lumped system elements* and generally valid statements of energy conservation, there are fundamentally two modeling approaches which emerge (Fig. 2.21):

- *energy-based modeling* employing *scalar energy functions* (LAGRANGE formalism, HAMILTON's equations)
- *multi-port modeling* employing *component-based* system models with *power-conserving network rules* (KIRCHHOFF networks, bond graphs).

In both modeling approaches, the aspects of *back-effect* arising from mutual power exchanges between interacting system components is taken into account in different ways. In the case of *unidirectional* KIRCHHOFF networks, modeling can be simplified using *signal-coupled networks* (e.g. control system signal-oriented diagrams). The port-HAMILTONian formulation—an interesting, relatively new approach—combines properties of energy-based and multi-port modeling and enables a specific type of mathematical model which is particularly useful for nonlinear controller design.

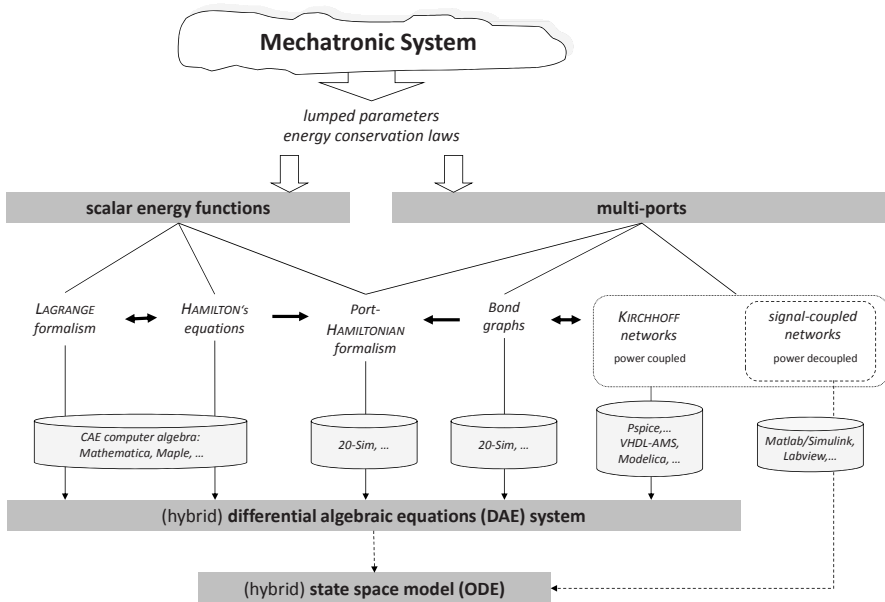


Fig. 2.21. Paradigms and commercial computational tools for multi-domain modeling of mechatronic systems using lumped system elements

Common model basis: DAE systems All of the modeling paradigms presented here accomplish the task of describing the physical dynamics of a *coupled heterogeneous* mechatronic system using a *domain-independent* mathematical model in the form of a system of differential-algebraic equations (DAE system) or a state space model. This mathematical model is then the starting point for all further analyses of system behavior.

A thorough discussion of these approaches is far beyond the scope of this book; for such, respective standards and specialized references should be consulted. However, in order to sharpen understanding of the scope of the problem and to ease classification of simulation approaches and tools, the remainder of this section presents a concise summary of the paradigms introduced above for multi-domain modeling of mechatronic systems. Due to their foundational importance and ease of application, the LAGRANGE formalism and the KIRCHHOFF network approach are described in somewhat more detail. In addition, these two approaches are applied in subsequent chapters for modeling physical phenomena implementing mechatronic system functions (Chs. 5 through 8).

2.3.1 Generalized power and energy

Axiomatic foundation All of the modeling paradigms presented here are based on generally valid statements of energy conservation. In order to clarify the relationship between these different approaches, this section introduces domain-independent generalized variables which describe the transfer of energy between component systems. This permits a largely domain-independent, generic, axiomatic foundation for modeling to be established, which is only connected to domain-specific physical laws—the so-called *constitutive equations*—at very precise points to realize a functional instrument for computation.

The following definitions and descriptions are conceptually largely aligned with (Wellstead 1979), though to facilitate the presentation sometimes employing different symbols. For a deeper background in modeling approaches utilizing this axiomatic foundation, the interested reader is referred to the eminently readable monograph (Wellstead 1979).

Definition 2.5. *Generalized energy variables.* For the modeling of mechatronic systems, the following generalized energy variables are defined:

- Generalized *potential, effort* e
- Generalized *velocity, flow* f
- Generalized *momentum*

$$p(t) := \int_{t_0}^t e(\tau) \cdot d\tau + p(t_0) \quad \text{or} \quad dp = e \cdot dt, \quad e = \dot{p}$$

- Generalized *coordinate, displacement*

$$q(t) := \int_{t_0}^t f(\tau) \cdot d\tau + q(t_0) \quad \text{or} \quad dq = f \cdot dt, \quad f = \dot{q}$$

- Generalized *power*

$$P(t) := \frac{dE(t)}{dt} := f(t) \cdot e(t) = \frac{dq}{dt} \frac{dp}{dt}$$

- Generalized *energy*

$$dE(t) = P(t) \cdot dt = f(t) \cdot e(t) \cdot dt = \frac{dq}{dt} e \cdot dt = e \cdot dq = f \frac{dp}{dt} dt = f \cdot dp$$

$$\Rightarrow E(t) = \int_{t_0}^t f(\tau) \cdot e(\tau) d\tau$$

- Generalized potential energy

$$V(q) = \int_{q_0}^q e(q) \cdot dq \quad (2.1)$$

- Generalized potential co-energy

$$V^*(e) = \int_{e_0}^e q(e) \cdot de$$

- Generalized kinetic energy

$$T(p) = \int_{p_0}^p f(p) \cdot dp$$

- Generalized kinetic co-energy

$$T^*(f) = \int_{f_0}^f p(f) \cdot df \quad (2.2)$$

A summary representation of the relationships created by these definitions is presented in Fig. 2.22 and Fig. 2.23.

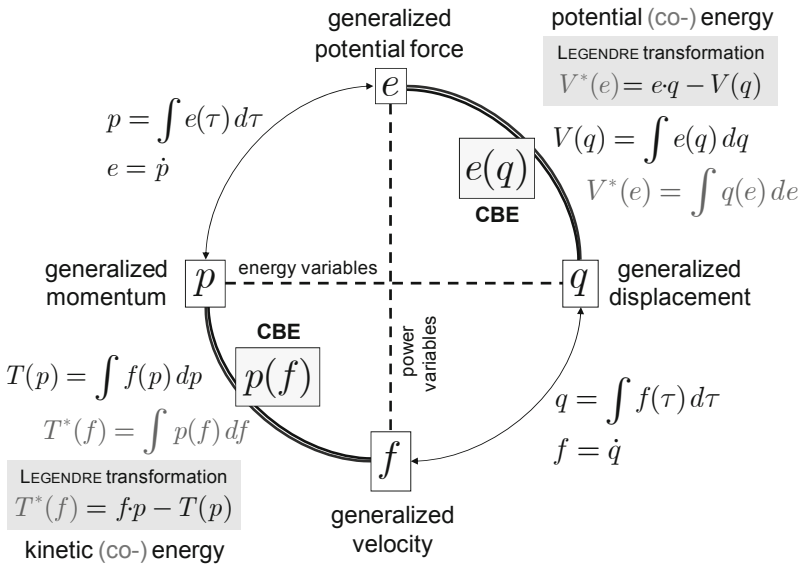


Fig. 2.22. Generalized energy variables: Definitional Relationships I (CBE = constitutive basic equations)

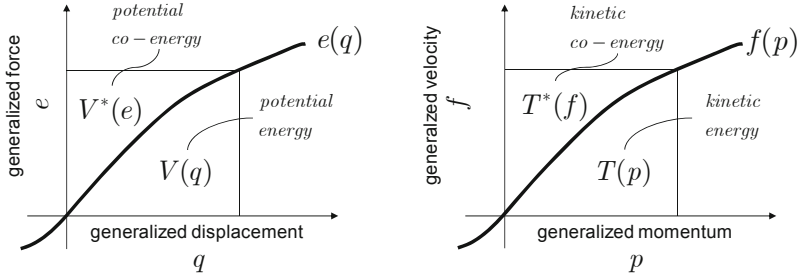


Fig. 2.23. Generalized energy variables: Definitional Relationships II

Conjugate variables The tuples (p, q) and (e, f) are respectively termed *conjugate energy variables* and *conjugate power variables*.

Constitutive equations Among the equations involving the generalized energy variables, the so-called *constitutive equations* $e = e(q)$ and its inverse relation $q = q(e)$, as well as $\dot{q} = \dot{q}(p)$ and $p = p(\dot{q})$, represent the *domain-specific physical laws*, which relate energy variables to one another.

For the majority of applications, the generalized displacement coordinates q and \dot{q} along with the constitutive equations $e = e(q)$ and $p = p(\dot{q})$ offer a convenient starting point for model creation.

Energy and co-energy The energy and co-energy variables are only equal in the case where there is a *linear* relationship between the energy and power variables (Fig. 2.23), i.e.

$$e = \alpha \cdot q \quad \text{or} \quad f = \dot{q} = \frac{1}{\beta} \cdot p.$$

The energy and co-energy are generally linked via a so-called **LEGENDRE transformation**

$$\begin{aligned} V^*(e) &= e \cdot q - V(q), \\ T^*(f) &= f \cdot p - T(p). \end{aligned}$$

NEWTONian mechanics In the context of **NEWTONian** mechanics with lumped parameters, there exists a *linear* relationship between momentum and velocity, i.e. momentum $p = M(x) \cdot v$, and angular momentum $h = I(\theta) \cdot \omega$. Thus the kinetic energy T and co-energy T^* are equal.

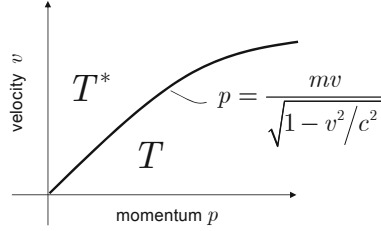


Fig. 2.24. Mechanical kinetic (co-)energy in the presence of relativistic effects

It is only with *relativistic* effects that kinetic energy T and co-energy T^* begin to differ (Fig. 2.24) due to

$$p = \frac{mv}{\sqrt{1 - v^2/c^2}}, \quad c = \text{speed of light.}$$

For the applications considered in this book, however, relativistic effects do not play a role.

There is however one more important aspect to consider. Typically when working in the context of NEWTONian mechanics, the kinetic *co-energy* is employed

$$T^*(\dot{q}) = \int_0^{\dot{q}} p(\dot{q}) \cdot d\dot{q} = \int_0^{\dot{q}} m \cdot \dot{q} \cdot d\dot{q} = \frac{1}{2} m \dot{q}^2 = \frac{1}{2} m v^2.$$

Due to the equality of T and T^* , this is not an issue in most applications (as long as NEWTONian conditions are maintained!). Care should however be taken when applying the LAGRANGE formalism (see Sec. 2.3), as there the distinction between the energy and co-energy functions must be carefully maintained.

Domain-specific relations

Table 2.2 shows examples of a few selected physical domains with their domain-specific energy variables and the corresponding constitutive equations. Corresponding elementary mechanical and electrical system configurations with linear *energy storage elements* are depicted in Fig. 2.25.

Table 2.2. Power and energy variables for selected physical domains and linear energy storage elements (SI units, generalized coordinates = domain-specific generalized displacements)

System Type	Gen. Coordinate (displacement)	Gen. Velocity (flow)	Constitutive Equations			Kinetic Co-energy	Potential Energy
			Gen. Momentum	Gen. Restoring Force (effort)	Other Gen. Force (excitation)		
	q	\dot{q}	$p(\dot{q})$	$e(q)$	$e(q)$	$T^*(q, \dot{q})$	$V(q)$
Mechanical Translation	Position [m]	Velocity [m/s]	Momentum [Ns]	Force [N]	Force [N]	$\frac{1}{2} M \dot{x}^2$	$\frac{1}{2} K x^2$
	x	$\dot{x} = v$	$p = M \dot{x}$	$F = Kx$	F		
Mechanical Rotation	Angle [rad]	Angular Velocity [rad/s]	Angular Momentum [Nms]	Torque [Nm]	Torque [Nm]	$\frac{1}{2} J \dot{\theta}^2$	$\frac{1}{2} K \theta^2$
	θ	$\dot{\theta} = \omega$	$h = J \dot{\theta}$	$\tau = K \theta$	τ		
Electrical	Charge [C=As]	Current [A=C/s]	Flux Linkage [Vs]	Voltage [V]	Voltage Source [V]	$\frac{1}{2} L \dot{q}^2$	$\frac{1}{2C} q^2$
	q	$\dot{q} = i$	$\psi = L \dot{q}$	$u = \frac{1}{C} q$	u		
Hydraulic	Volume [m ³]	Volume Flow Rate [m ³ /s]	Pressure Impulse [Ns/m ²]	Pressure [N/m ²]	Pressure [N/m ²]	$\frac{1}{2} I_h \dot{Q}^2$	$\frac{1}{2C_h} V^2$
	V	Q	$p_P = I_h \dot{Q}$	$P = \frac{1}{C_h} V$	P		

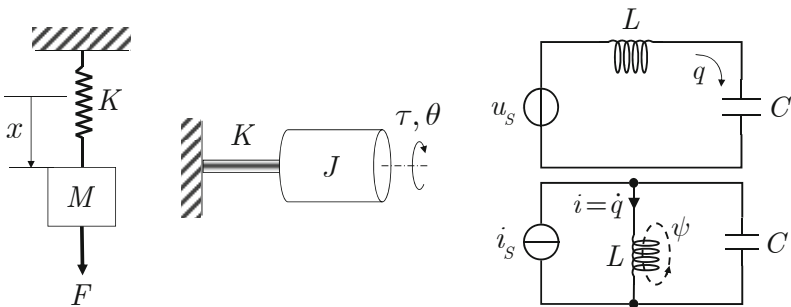


Fig. 2.25. Elementary physical systems (mechanical, electrical)

Table 2.3. Network-based power and energy variables for mechanical systems (SI units)

System Type	Constitutive Equations			
	Integrated Flow Variable	Flow Variable	Integrated Effort Variable	Effort Variable
	q	f	$p(f)$	$e(q)$
Mechanical Translation	Momentum [Ns]	Force [N]	Position [m]	Velocity [m/s]
	p	F	$x = \frac{1}{K} F$	$v = \frac{1}{M} p$
Mechanical Rotation	Angular Momentum [Nms]	Torque [Nm]	Angle [rad]	Angular Velocity [rad/s]
	h	τ	$\theta = \frac{1}{K} \tau$	$\omega = \frac{1}{J} h$

Mechanical power variable in KIRCHHOFF networks Interestingly, in the domain of KIRCHHOFF network models for mechanical systems, slightly different definitions are often used for the flow and effort variables (Lenk et al. 2011), (Reinschke and Schwarz 1976). In this formulation, the conjugate power variables are switched $e \rightleftharpoons f$ while the remainder of the definitional relations are retained (see Table 2.3).

Note that under this alternate assignment of variables, the power $P = e \cdot f$ does not change. However, as is easy to verify, the definitional assignment of potential and kinetic (co-)energy does change. For this reason, when applying the energy-based LAGRANGian formulation, it is better to use the power and energy variables defined in Table 2.2 (see Sec. 2.3.2).

2.3.2 Energy-based modeling: LAGRANGE formalism

Background An exceedingly elegant method for domain-independent modeling is offered by the *LAGRANGE formalism* (Schultz and Melsa 1967), (Goldstein et al. 2001), (Wellstead 1979), based on the calculus of variations. Using the generalized energy and power variables (Table 2.2), not

only can the equations of motion for *mechanical* systems be derived (as is commonly taught in courses on mechanics), but in a very natural way, the dynamic equations for *heterogeneous physical* systems—in particular for *mechatronic* systems—can also be obtained.

A basic knowledge of the LAGRANGE formalism is assumed here; interested readers are directed to one of the above-cited monographs to refresh or update their knowledge. The remainder of this section describes the most important steps and requirements for modeling, and demonstrates them with an example.

The LAGRANGE formalism is further used in Chapter 5: Generic Mechatronic Transducer as a fundamental basis for deriving a generalized dynamic transducer model.

Generalized coordinates One important modeling decision is the choice of domain-specific (physical) generalized coordinates. Candidates include two of the energy variables: the generalized *displacement* q or the generalized *momentum* p (see Fig. 2.22). Both coordinates are equally valid; note however the differing definitions of the energy functions depending on the choice of variables (energy vs. co-energy).

In the case of *mechanical* systems, the mechanical *displacement* x is commonly chosen as the generalized coordinate, so that the usual kinetic *mechanical co-energy* $T^*(x, \dot{x})$ ⁸ and potential mechanical energy $V(x)$ are employed, see Table 2.2.

In the case of *electrical* systems, there is a choice of whether to choose the electrical displacement (*charge*) q_{el} or the electrical momentum (*flux linkage*) ψ as the generalized coordinate. A detailed discussion of this aspect of modeling is undertaken in Chapter 5: Generic Mechatronic Transducer, so that this topic is not further pursued here.

For compatibility with mechanical systems, the following discussion considers the generalized coordinate to be the respective *domain-specific displacement coordinate*, in other words, *electric charge* q_{el} for electrical components and *volume* V_{fluid} for hydraulic components. Under these assumptions, it is thus always the kinetic co-energy function and the potential energy function which apply.

⁸ In certain cases, the kinetic energy depends not only on the generalized velocity, but also on the generalized displacement (position), e.g. in a multi-link manipulator, the mass matrix depends on the joint angles.

Configuration space: redundant coordinates If every energy storage element $\nu = 1, \dots, N$ is assigned a generalized coordinate \tilde{q}_ν according to Table 2.2, the result is a vector of generalized coordinates

$$\tilde{\mathbf{q}} = (\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_N)^T \quad (2.3)$$

defining the *configuration space* of the system under consideration. Since these coordinates are generally not independent due to configuration constraints, they are termed *redundant* coordinates.

Holonomic constraints: minimal coordinates In general, energy storage elements are coupled via constraints. These manifest themselves as dependencies between the generalized coordinates, which, in the simplest case, can be described via *algebraic* relations between the coordinates. These are termed *holonomic* constraints⁹.

With N_C such *holonomic constraints*

$$h_j(\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_N) = 0, \quad j = 1, \dots, N_C \quad (2.4)$$

elimination of N_C components of the redundant coordinates is possible (by expressing them in terms of other coordinates) resulting in a set of *independent* coordinates, the *minimal coordinates*

$$\mathbf{q} = (q_1, q_2, \dots, q_{N_{FG}})^T, \quad N_{DOF} = N - N_C \quad (2.5)$$

where N_{DOF} is termed the number of *degrees of freedom* of the system.

Nonholonomic constraints If the configuration constraints between redundant generalized coordinates cannot be described via algebraic equations, they are termed *nonholonomic*¹⁰ constraints. Examples include non-integrable differential equations (e.g. a rolling wheel) or inequality constraints.

Differential constraints: PFAFFian form The manipulation of holonomic and nonholonomic constraints is simplified by examining their *differential* form, i.e. the relationship between derivatives of the generalized coordi-

⁹ A more detailed discussion of *holonomic* and *nonholonomic* conditions is included in Chapter 4 (Sec. 4.3.3) and is thus not further pursued here.

¹⁰ See Sec. 4.3.3.

nates. This results in first-order differential equations in a particular form—the *PFAFFian form*:

$$\sum_{i=1}^N a_{ji}(\tilde{\mathbf{q}}) \cdot \dot{\tilde{q}}_i = 0, \quad j = 1, \dots, N_C. \quad (2.6)$$

If a differential constraint following Eq. (2.6) has been derived from a holonomic constraint (Eq. (2.4)), it then holds that

$$a_{ji} := \frac{\partial h_j(\tilde{\mathbf{q}}, t)}{\partial \tilde{q}_i}, \quad i = 1, \dots, N; \quad j = 1, \dots, N_C. \quad (2.7)$$

EULER-LAGRANGE equations of the second kind Under the condition that the chosen *minimal coordinates* are not only *independent*, but also unconstrained, component-wise construction of the kinetic co-energy $T^*(\mathbf{q}, \dot{\mathbf{q}})$ and the potential energy $V(\mathbf{q})$ enables construction of *EULER-LAGRANGE equations of the second kind*:

$$\frac{d}{dt} \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_i} - \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial q_i} = f_i - D_i, \quad i = 1, \dots, N_{DOF} \quad (2.8)$$

with the *LAGRANGian*¹¹

$$L(\mathbf{q}, \dot{\mathbf{q}}) := T^*(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q}). \quad (2.9)$$

The LAGRANGian (2.9) contains, at a consistent, abstract level, all *conservative* dynamic components of the modeled heterogeneous physical (mechatronic) system (see Table 2.2). Domain-specific generalized *forcing* terms f_i (see Table 2.2, column 6) and *dissipative* forces D_i can be accounted for on an equally consistent, abstract level within the EULER-LAGRANGE equations (2.8) (right-hand side) (Schultz and Melsa 1967), (Goldstein et al. 2001).

¹¹ Note: the often used form of the LAGRANGE function “kinetic co-energy minus potential energy” is only valid when displacement coordinates are chosen as the generalized coordinates (this is very practical for modeling of mechanical systems). In general, “LAGRANGE function = total co-energy minus total energy”.

The system of equations (2.8) leads to a set of nonlinear second-order differential equations in terms of the generalized minimal coordinates $\mathbf{q} = (q_1, q_2, \dots, q_{N_{\text{DOF}}})^T$. Using the state definition $x_j := q_i$, $x_{j+1} := \dot{q}_i$ and $u_i := f_i$, this can be easily transformed into a tractable *state space model* of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t). \quad (2.10)$$

EULER-LAGRANGE equations of the first kind In the case of *non-holonomic* constraints between redundant generalized coordinates $\tilde{\mathbf{q}} = (\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_N)^T$ or when *constraint forces* are called upon to maintain system constraints, the EULER-LAGRANGE equations can be extended with LAGRANGE multipliers λ_j to form *EULER-LAGRANGE equations of the first kind*

$$\frac{d}{dt} \frac{\partial L(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}})}{\partial \dot{\tilde{q}}_i} - \frac{\partial L(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}})}{\partial \tilde{q}_i} = f_i - D_i + \sum_{j=1}^{N_c} \lambda_j a_{ji}(\tilde{\mathbf{q}}), \quad i = 1, \dots, N \quad (2.11)$$

with differential-algebraic *constraints* (holonomic and/or nonholonomic, the PFAFFian form) following Eq. (2.6).

The second-order differential system of equations (2.11) and the first-order differential system of equations (2.6) give a total of $(N + N_c)$ equations for computation of the $(N + N_c)$ unknowns $\tilde{q}_1, \dots, \tilde{q}_N$ and $\lambda_1, \dots, \lambda_{N_c}$.

Here, the LAGRANGE multipliers $\lambda_j(t)$ represent the *constraint forces* required to maintain the constraints in Eq. (2.6) (Goldstein et al. 2001). To this extent, the EULER-LAGRANGE equations of the first kind also commend themselves in the case of holonomic constraints if the constraint forces are of interest.

In the general case, using the state definition $x_j := \tilde{q}_i$, $x_{j+1} := \dot{\tilde{q}}_i$ or the algebraic variables $z_j := \lambda_j$ results in a system of *differential-algebraic equations* (DAE) in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}, t) \quad (2.12)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{z}, t). \quad (2.13)$$

The dimension of the state vector \mathbf{x} can vary from N to $2N$.

Example 2.1 *Electrostatic saddle bearing: LAGRANGE formalism.*

System configuration Gyroscopes are used to measure rotation rates of moving objects (airplanes, satellites, etc.). An electrostatic suspension in a gyroscope enables friction-free rotation of the inertial mass spinning at a constant rate. A full two-sided electrostatic suspension (electrostatic bearing) is treated in detail in Ch. 6 (Sec. 6.6.4, Example 6.2).

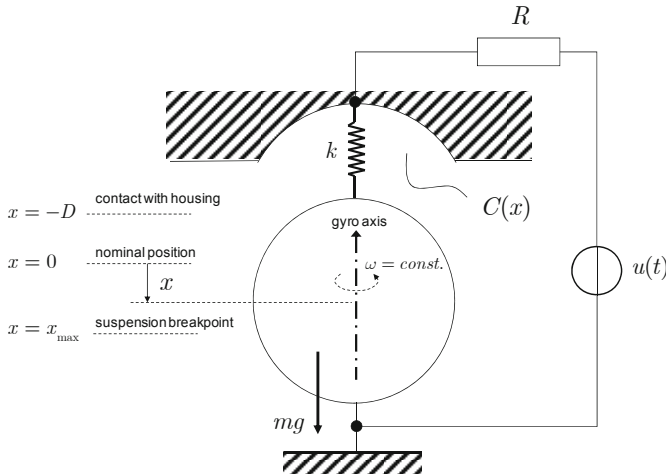


Fig. 2.26. Simplified configuration for an electrostatic saddle bearing (e.g. a test assembly for functional tests of bearings for an electrostatic gyro)

In this example, however, the *equations of motion* of the *elastically suspended* inertial mass m are to be derived for the greatly *simplified model* of an *electrostatic saddle bearing* shown in Fig. 2.26 (from (Schultz and Melsa 1967)) with help of the LAGRANGE formalism. This configuration can be considered to be a test assembly for functional tests of a full electrostatic bearing. Here, instead of the lower electrostatic bearing, an elastic suspension is inserted in order to prevent the sphere from sticking to the upper housing.

Model validity A few remarks regarding restrictions on the model validity: in this example, only the vertical motion of the sphere is of interest. The vertical translational motion is decoupled from the rotational motion of the sphere. The rotational energy of the sphere is a conserved variable (i.e. a constant physical quantity), and does not affect the translational motion. Thus, the sphere rotation can be ignored when creating the model.

Two *nonholonomic* constraints are apparent:

$$x < x_{\max} : \text{when violated} \rightarrow \text{detachment of suspension}$$

$$x > -D : \text{when violated} \rightarrow \text{contact with housing.}$$

In the remainder of this example, it is assumed that the sphere only moves within its free range of motion. Thus, the nonholonomic constraints do not bind, and all relevant generalized coordinates remain *unconstrained* (an important prerequisite for EULER-LAGRANGE equations of the second kind).

Model creation To start, a generalized coordinate is chosen for each of the three energy storage elements (see Table 2.2):

$$C : q_1 := \text{charge } Q$$

$$k : q_2 := \text{displacement } x$$

$$m : q_3 := \text{displacement } x$$

Due to the *holonomic* condition $q_2 = q_3$, there remain only two degrees of freedom with *independent, unconstrained* (in the sense of the above assumption!) generalized coordinates q_1, q_2 .

The energy functions obtained for this case are (see also Sec. 5.3.1):

$$V(\mathbf{q}) = \frac{1}{2C(q_2)} q_1^2 + \frac{1}{2} k q_2^2 - m g q_2,$$

$$T^*(\dot{\mathbf{q}}) = \frac{1}{2} m \dot{q}_2^2,$$

giving the LAGRANGian $L(\mathbf{q}, \dot{\mathbf{q}}) := T^*(\dot{\mathbf{q}}) - V(\mathbf{q})$.

An approximation for the computation of the gap-dependent capacitance $C(x)$ is a simplified *plate capacitor* where

$$C(q_2) = \frac{A \varepsilon_0}{D + q_2} \quad \begin{array}{l} A \dots \text{capacitor area} \\ \varepsilon_0 \dots \text{permittivity of free space} \\ D \dots \text{see nonholonomic condition.} \end{array}$$

The only *dissipative* term which requires consideration is the resistive contribution $R \dot{q}_1$ for the electrical coordinate q_1 . The generalized *forcing* term is to be found in the voltage source $u(t)$ (in the q_1 -direction).

By evaluating the EULER-LAGRANGE equations of the second kind (2.8), the (nonlinear) *equations of motion* are then obtained in *problem coordinates*:

$$\begin{aligned} m \ddot{x} + kx + \frac{Q^2}{2\varepsilon_0 A} &= mg, \\ R \dot{q}_C + \frac{D+x}{\varepsilon_0 A} Q &= u. \end{aligned} \quad (2.14)$$

Using the *state definition* $x_1 := x$, $x_2 := \dot{x}$, $x_3 := Q$, in Eq. (2.14), the *state space model* is then

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{k}{m}x_1 - \frac{1}{2\varepsilon_0 Am}x_3^2 + g \\ \dot{x}_3 &= -\frac{D}{\varepsilon_0 AR}x_3 - \frac{1}{\varepsilon_0 AR}x_1x_3 + \frac{1}{R}u.\end{aligned}\tag{2.15}$$

Discussion As is apparent, Eq. (2.15) represents an ordinary state space model without algebraic variables or equations. This is not surprising, as a set of independent generalized coordinates q_1, q_2 without constraints was already present.

One other particularity is worth noting in this example: as is apparent, the EULER-LAGRANGE equations—despite representing a system of second-order differential equations with *two degrees of freedom*—do not (as in the case of purely mechanical systems) result in a state space model with *four* states. Rather, only *three* states appear in Eq. (2.15). This is due to the fact that in the electrical subsystem including the capacitor, only one potential energy storage element is present, so that the electrical coordinate makes no contribution to the total kinetic energy. As a result, the typically-expected second time derivative of the electrical coordinate—which would require an additional state variable—is missing. ■

Ease of use The charm of the LAGRANGian formulation lies in both the universal applicability of its definition of energy functions and in its automatable, formal procedure. The actual creative modeling task consists of establishing generalized coordinates and constraints, and the energy functions. Derivation of the systems of equations in (2.8) and (2.11) can be transferred to a computer algebra program. The LAGRANGE formalism is particularly well-suited to problem statements of sufficiently *low order* with *smooth nonlinearities*. For higher-order systems, however, the analytic derivation of EULER-LAGRANGE equations very quickly becomes intractable.

Multi-domain properties Example 2.1 nicely demonstrates the strengths of the LAGRANGE formalism. The simplicity of combining differing domains at the model level is noteworthy. Starting from the elementary constitutive relations for each energy storage element in the system, and the

geometric and algebraic dependencies between coordinates in the problem, the derivation of the EULER-LAGRANGE equations so-to-speak automatically reproduces the multi-domain *interactions* between the electrical and mechanical component systems, i.e., in this case, the *charge-dependent* and *gap-dependent* mechanical *COULOMB force*

$$F_{Coul}(t) = \frac{Q(t)^2}{2\epsilon_0 A} \quad (2.16)$$

and the *gap-dependent capacitor voltage*

$$u_C(t) = \frac{D + x(t)}{\epsilon_0 A} Q(t). \quad (2.17)$$

2.3.3 Energy-based modeling: HAMILTON's equations

Special state definition: canonical momentum Transforming the system of second-order differential EULER-LAGRANGE equations into an equivalent system of first-order differential equations can be effected via a very specialized *state definition*, motivated by analytical mechanics (Reinschke 2006).

Define as the state variable the existing generalized position coordinates q_i as well as a set of new variables, their *conjugate* coordinates p_i :

$$p_i := \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_i}, \quad i = 1, \dots, N_{DOF}. \quad (2.18)$$

The variables p_i are also termed the *canonical momentum* variables¹². It is easy to see that, from a physical point of view, the p_i actually correspond to generalized momenta by simply evaluating Eq. (2.18) while accounting for $L = T^* - V$ and relation (2.2) with $f = \dot{q}$.

Due to Eq. (2.8) (and assuming an autonomous system), the canonical momentum variables p_i satisfy the following system of first-order equations

$$\dot{p}_i = \frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial q_i}, \quad i = 1, \dots, N_{DOF}.$$

¹² In physics, the term “canonical” has the meaning of *natural* or *regular*.

Employing a manipulation with various justifications (Goldstein et al. 2001), (Reinschke 2006), and with help of Eq. (2.18), the scalar *HAMILTONian* $\mathcal{H}(\mathbf{q}, \mathbf{p})$ can be introduced and derived from the LAGRANGian via the following LEGENDRE transformation

$$\mathcal{H}(\mathbf{q}, \mathbf{p}) := \mathbf{p}^T \dot{\mathbf{q}} - L(\mathbf{q}, \dot{\mathbf{q}}). \quad (2.19)$$

With the resulting HAMILTONian and the canonical momentum, the EULER-LAGRANGE equations, after a few intermediate steps, can be written in the following, equivalent form:

$$\dot{q}_i = \frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial p_i} \quad \dot{p}_i = -\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial q_i}, \quad i = 1, \dots, N_{DOF}. \quad (2.20)$$

The $2N_{DOF}$ differential equations (2.20) are called *HAMILTON's equations* or also the *canonical equations of motion* (due to their formal simplicity and symmetric structure). The $2N_{DOF}$ state variables q_i, p_i are termed the *canonical variables*. In general, systems which can be described via a function $\mathcal{H}(\mathbf{q}, \mathbf{p})$ and a system of differential equations (2.20) are called *HAMILTONian systems*.

In any concrete case, the computation of the HAMILTONian (2.19) presents a certain obstacle. Note that it is a function of \mathbf{q} and \mathbf{p} . However, the right-hand side of Eq. (2.19) still contains the generalized velocities $\dot{\mathbf{q}}$. In order to eliminate the velocities, Eq. (2.18) is evaluated, giving $\dot{q}_i = \dot{q}_i(\mathbf{q}, \mathbf{p})$.

Special case In the *special case* of scleronomic (time-independent), holonomic constraints, inertial coordinates, and conservative forces, this laborious computation can, however, be avoided, and the fact that in this case (only with the indicated prerequisite conditions!), the HAMILTONian describes the *total energy* of the system under consideration (Goldstein et al. 2001) can be used to advantage, i.e.

$$\mathcal{H} = T^* + V = E_{tot}.$$

Significance The HAMILTON's equations (2.20) are completely equivalent to the LAGRANGian equations. Thus, all properties discussed there hold here as well. The HAMILTONian procedure seems, in the general case, somewhat less tractable (and is also much less taught). However, it offers certain advantages in terms of a geometric interpretation of the motion in

the coordinate/momentum state space (also called the *phase space*). The HAMILTONian approach in the elementary form presented here has no noteworthy significance in the modeling of mechatronic systems. However, it forms the basis for the port-HAMILTONian formalism presented in Sec. 2.3.6¹³.

2.3.4 Multi-port modeling: KIRCHHOFF networks

Motivation The abstract depiction of physical systems as *networks* permits an additional, very powerful approach to multi-domain modeling for mechatronic systems. Methods and procedures for network modeling and analysis are especially developed in the field of electronics, and have also been made broadly useful in industry with powerful computational tools for highly complex applications in the field of microelectronics. It is noteworthy that mechanical, acoustic, fluidic, and thermal systems can also be neatly abstracted as networks; generalized KIRCHHOFF's Laws suffice for their modeling and analysis. Such networks are termed *KIRCHHOFF networks*. Due to the existence of well-developed methods and tools for electrical networks, it is convenient to describe network models in the other domains as equivalent electrical networks. These are then called "*analogous electrical*" networks (Lenk et al. 2011). By finding these analogs, it is thus possible to depict networks from a variety of domains at a common, abstract level in a unified system model.

Lumped element networks The following discussion examines networks having *lumped elements*. Such spatially and functionally delineated elements can mutually exchange energy with other coupled elements (the *network*) via interfaces (*terminals* or *poles*, with a pair of terminals forming a *port*). Thus network elements are termed two-terminal elements (bi-poles), three-terminal (tripoles), four-terminal (quadripoles), or one-port or multi-port (Fig. 2.27). Spatially lumped implies that spatial delay effects (wave propagation) play no role at the element level.

¹³ In the field of mechanics, however, HAMILTON's equations are of great importance to statistical mechanics and quantum mechanics, see (Goldstein 2001).

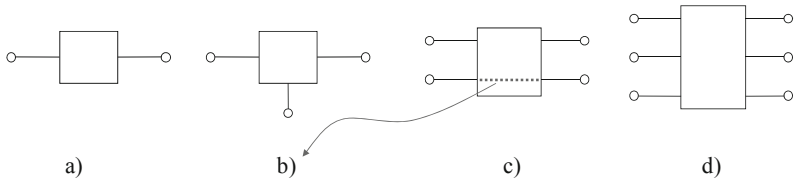


Fig. 2.27. Lumped network elements: a) two-terminal = one-port, b) three-terminal, c) four-terminal = two-port, d) multi-port

Effort and flow variables The behavior of an element can be described using two generic terminal variables (Fig. 2.28):

- *Flow f* . Describes variables which transit through the network element, i.e. flow into and out of the element (e.g. electric current). Such a variable can be measured at any single point (terminal) in the network.
- *Effort e* . Describes variables which are applied between terminals of the network element (e.g. electric voltage). Such a variable can be measured between two different points (terminals) in the network.

In the international literature, the synonymous terms for flow and effort variables shown in Table 2.4 have become commonplace.

Domain-specific flow and effort variables A straightforward assignment of flow and effort variables to concrete physical quantities is possible via simple energy considerations. The resulting *power flow* (= energy flow per unit time) can be unambiguously described for each network element by the product of two system variables—termed *conjugate variables*. If these are chosen to be flow and effort variables, then the domain-specific

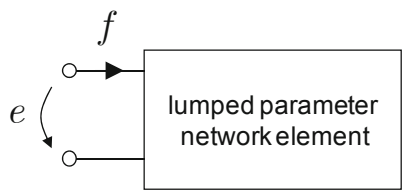


Fig. 2.28. Effort variable e and flow variable f for a network element

Table 2.4. Different terms for flow and effort variables

<i>effort variable “e”</i>	<i>flow variable “f”</i>
across variable	through variable
potential	

choice of these variables is, in effect, unrestricted (and thus unfortunately non-unique and somewhat ambiguous), as it is only the *product* = *effort* \times *flow* which must have the physical units of power.

A few examples of domain-specific power variables can be found in Tables 2.2 and 2.3. Note the different conventions for mechanical systems: $e := \text{force}$, $f := \text{velocity}$ vs. $e := \text{velocity}$, $f := \text{force}$. In both cases, the product is power, but with quite differing physical interpretations of flow and effort variables. However, this is not the only ambiguity resulting from the usage of conjugate variables, as discussed below.

For reasons of convenience, flow and effort can also be chosen deviating from the power convention presented above. For some mechanical systems, e.g. servo drives, it is the axis position which is of primary interest. In this case, without further consideration, $f := \text{force}$ and $e := \text{position}$ can be chosen. In this case, the product is energy, i.e. the work expended. Regardless of the specific choice of flow and effort variables, the product should always result in an energy-related quantity.

Circuit laws Combining elements into a network proceeds by considering the following elementary conservation laws (Reinschke and Schwarz 1976), (Fig. 2.29):

- *Junction rule for flow variables* (KIRCHHOFF's node rule)
The algebraic sum of all flow variables (inflows and outflows) at any arbitrary network node is zero at every point in time.
- *Loop rule for effort variables* (KIRCHHOFF's mesh rule)
The algebraic sum of all effort variable differences around any arbitrary closed network path is zero at every point in time.

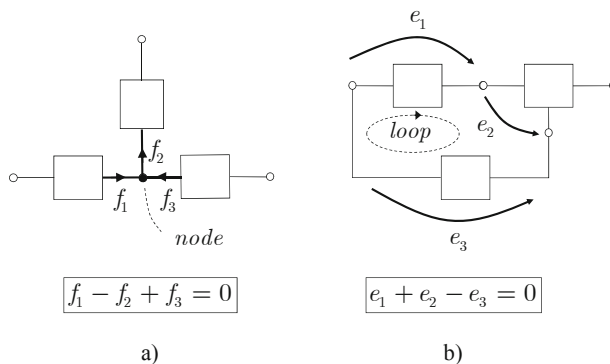


Fig. 2.29. Generalized KIRCHHOFF's laws: a) junction rule, b) loop rule

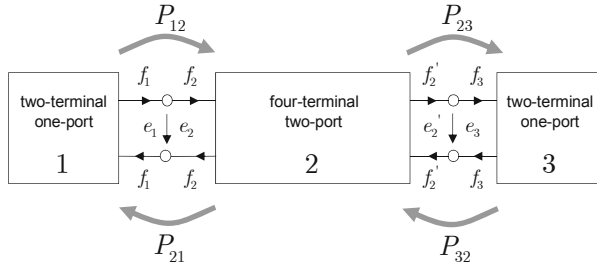


Fig. 2.30. Port-based network

Port-based networks So-called *two-terminal* or *one-port elements* and *four-terminal* or *two-port elements* are the most important network elements. Networks resulting from the interconnection of such elements are called *port-based networks* (Fig. 2.30). When creating networks, it should be ensured that the same types of flow and effort variables connect at circuit nodes.

Elementary two-terminal element: the one-port Characteristic *constitutive* relations between effort and flow can be described using the following elementary two-terminal or one-port network elements:

- *Load*

$$e(t) = \alpha \cdot f(t) \quad (2.21)$$

- *Flow accumulator*

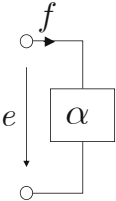
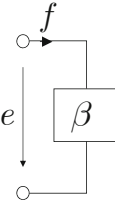
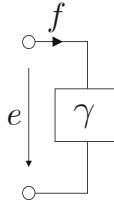
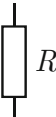




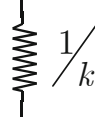
$$f(t) = \beta \frac{d}{dt} e(t) \quad \text{or} \quad e(t) = e(t_0) + \frac{1}{\beta} \int_{t_0}^t f(\tau) d\tau \quad (2.22)$$

- *Effort accumulator*

$$e(t) = \gamma \frac{d}{dt} f(t) \quad \text{or} \quad f(t) = f(t_0) + \frac{1}{\gamma} \int_{t_0}^t e(\tau) d\tau. \quad (2.23)$$

The proportionality constants α, β, γ used in relations (2.21) through (2.23) represent the parameters of the corresponding lumped network elements. In the form presented, these parameters are specified as constant, so that *linear time-invariant* relations between the power variables e and f result. In the general case, however, time varying and nonlinear relationships are also possible (see also the constitutive equations in Sec. 2.3.1 and Fig. 2.23).

Table 2.5. Elementary two-terminal (one-port) network elements (general, electric, mechanical)

	consumer	flow accumulator	effort accumulator
general			
electrical $e = \text{voltage}$ $f = \text{current}$			
mechanical $e = \text{velocity}$ $f = \text{force}$			

Analogous relations In any particular domain, the parameters α, β, γ are typically assigned individual names and symbols (Tables 2.2, 2.3). To the great chagrin of the engineer, however, this assignment is rather inconsistent, particularly the assignments for *mechanical* systems. In the field of *network theory* (Thomas et al. 2009), (Reinschke and Schwarz 1976), (Lenk et al. 2011), the assignment of effort and flow shown in Table 2.3 has become the norm. The reader can verify that as a result, the assignment of electrical analogs shown in Table 2.5 and Table 2.6 is then appropriate for mechanical network parameters.

Table 2.6. Analogous electrical assignment of mechanical network parameters

Flow f	Force or torque F or τ
Effort e	Velocity or angular velocity v or ω
Load R	1/friction (damping) coefficient $h = 1/b$
Capacitance C	Mass or moment of inertia m or J
Inductance L	Compliance (= 1/stiffness) $n = 1/k$

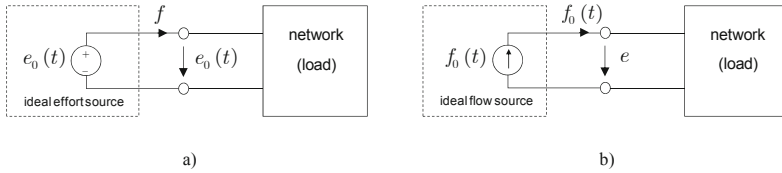


Fig. 2.31. Ideal network sources: a) potential source, b) flow source

Independence of power variables One hallmark of port-based networks is that only one of the two conjugate power variables can be independently controlled at any one time. An important example is seen in ideal (lossless) sources for effort and flow (Fig. 2.31). In the case of a *potential source* (source for effort variables), the output effort $e_0(t)$ can be controlled independently of the applied load, whereas the flow $f(t)$ is load-dependent. The equivalent holds for *flow sources*, where the flow $f_0(t)$ can be controlled independently of the load.

Transducers The domain-specific two-terminal network elements introduced so far can only be connected within a single physical domain, i.e. only the same types of effort and flow variables can be coupled to the network port. In order to model the interaction of variables from different domains—e.g. electrical-mechanical, mechanical-hydraulic, translation-rotation—the *four-terminal* or *two-port* transducer elements shown in Fig. 2.30 (element 2) are required¹⁴.

Transformer vs. gyrator These two transducer elements describe a generalized power flow between different ports. In particular, differing physical domains can be present at the two ports, e.g. $(e_1, f_1) \triangleq$ electrical and $(e_2, f_2) \triangleq$ mechanical. Fig. 2.32 shows two elementary *two-port elements* (*quadripoles*): an ideal *transformer* (*transducer*) and an ideal *gyrator*.

The two-port elements represent a *lossless* transfer of power, i.e. regardless of the particular power variables, $e_1 \cdot f_1 = e_2 \cdot f_2$.

In the case of a *transformer*, the transducer constant n determines the relation between pairs of the same power variables (e_1, e_2) or (f_1, f_2) of the two ports. In the case of a *gyrator*, the transducer constant r is used to relate pairs of differing power variables (e_1, f_2) or (e_2, f_1) of the two ports.

¹⁴ Using so-called *multi-ports*, the simultaneous interactions between more than two domains can be described in an expanded form.

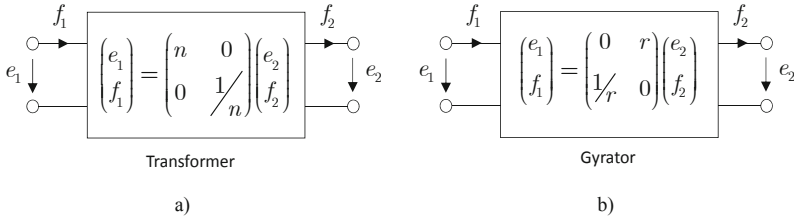


Fig. 2.32. Ideal transducers (lossless): a) transformer, b) gyrator

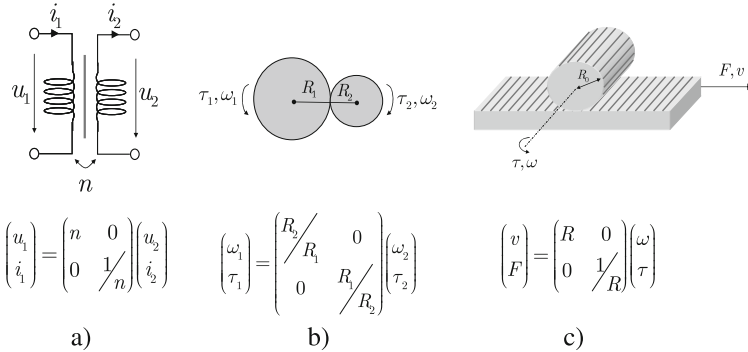


Fig. 2.33. Elementary mechatronic transformer transducers (ideal, lossless): a) electrical transformer, b) mechanical transmission, c) rack and pinion



Fig. 2.34. Elementary mechatronic gyrator transducer (ideal, lossless): hydraulic transducer

The physical units of the transducer constants n , r are determined by the domain-specific power variables. Fig. 2.33 shows physical examples of *transformer* type transducers and Fig. 2.34 shows a physical example of a *gyrator* type transducer.

In the matrix form shown, the transfer matrix represents the so-called chain matrix of two-port network theory.

Mechanical transducers: variable definitions Recall that, due to the arbitrary assignment of domain-specific power variables, the description of physical transducer implementations is not unique. Particular attention is due in the case of *mechanical transducers*. Fig. 2.35 shows a lossless DC motor as an example of an ideal electromechanical transducer. Depending

on which physical quantities are assigned to the power variables *effort* (e) and *flow* (f), the same system can be described as a *transformer* or a *gyrator*. Particular attention should be paid to this fact when such systems are modeled using computer-aided tools and pre-made component libraries.

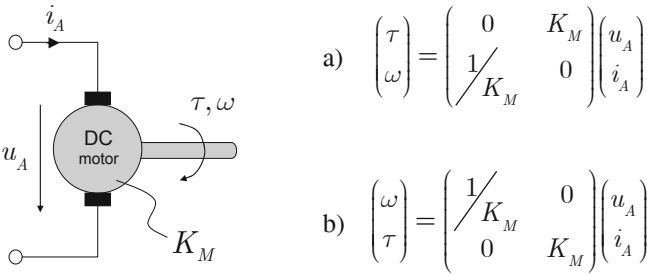


Fig. 2.35. Lossless DC motor as an ideal electromechanical transducer: a) mechanical power variables $\text{effort} := \tau$, $\text{flow} := \omega \rightarrow$ gyrator, b) mechanical power variables $\text{effort} := \omega$, $\text{flow} := \tau \rightarrow$ transformer, K_M designates the motor constant

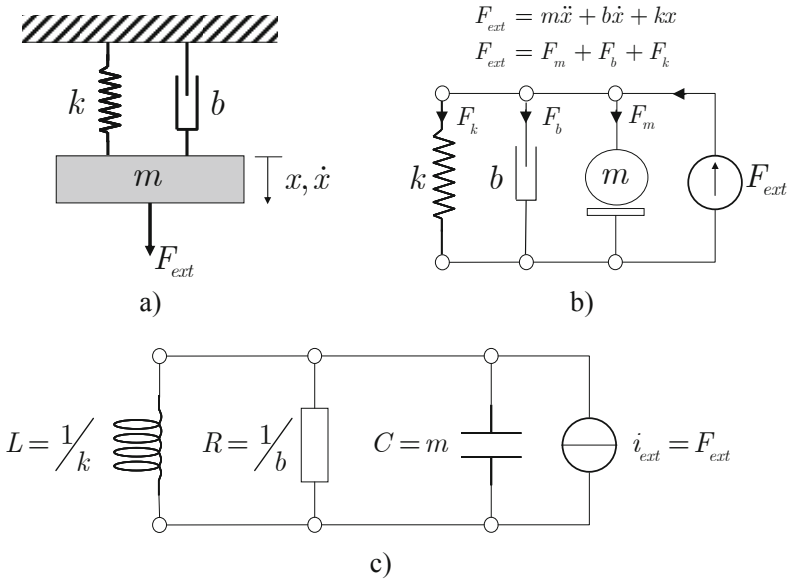


Fig. 2.36. Example of topological rules of construction: a) physical configuration of a mechanical system, b) mechanical network, c) analogous electrical network

Topological rules of construction Starting with the physical topological structure of network elements, an abstract topological network model can be constructed as an undirected graph including standardized network elements (e.g. Table 2.5) using the following elementary rules (Fig. 2.36):

- *Common potential*: Elements with a common potential are connected *in parallel* in the abstract network graph,
- *Common flow*: Elements with a common flow are connected *in series* in the abstract network graph.

Connection rule for inertial masses One pole of the mass symbol (the one with the *bar*) must always be connected to an *inertial system* (corresponding to “ground” in an electrical network).

This is due to the applicability of NEWTON’s second law of motion and thus the inertia $m\ddot{x}$ relative to inertial space. Note that in many cases, it suffices to consider a *non-accelerating* reference coordinate system as a “virtual” inertial coordinate system, e.g. a coordinate system attached to the Earth and with a constant inertial orientation, or a coordinate system attached to a vehicle if the vehicle is moving with constant velocity (i.e. is not accelerating) (see Sec. 4.3.2, Example 4.1).

KIRCHHOFF networks are defined by

- flow variables f
- effort variables e
- a conservation rule for flow (junction rule, node rule)
- a conservation rule for effort (loop rule, mesh rule)
- constitutive equations $f = f(e)$ or $e = e(f)$

Example 2.2 *Electrostatic saddle bearing – multi-port model.*

System configuration A multi-port system model is to be derived for the electrostatic saddle bearing already described in Example 2.1 (Fig. 2.26).

From the physical configuration, a locally limited electrical and a mechanical network can be directly extracted. These two networks are depicted as (for the time being) separate networks in Fig. 2.37.

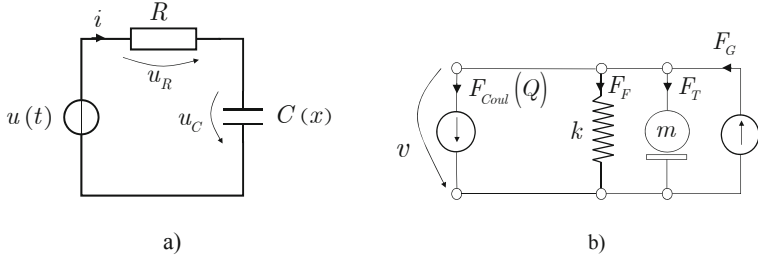


Fig. 2.37. Electrostatic saddle bearing: separate domain-specific networks: a) electrical network, b) mechanical network

Mathematical model: DAE system For each of these networks, the elementary circuit rules can now be applied, and the constitutive equations for the network elements can be constructed.

(a) Electrical network

Loop rule for effort variables (electrical voltages, KIRCHHOFF's mesh rule)

$$\sum_i u_i = 0 : \quad u = u_R + u_C \quad (2.24)$$

Constitutive equations

$$\begin{aligned} i &= \dot{Q} \\ u_R &= R \cdot \dot{Q} \\ u_C &= \frac{1}{C(x)} Q = \frac{D + x}{\varepsilon_0 A} Q \end{aligned} \quad (2.25)$$

(b) Mechanical network

Junction rule for flow variables (forces, NEWTON's third law)

$$\sum_i F_i = 0 : \quad F_T = F_G - F_F - F_{Coul} \quad (2.26)$$

Constitutive equations

$$\begin{aligned} F_T &= m\ddot{x} \\ F_G &= mg \quad F_{Coul}(Q) = \frac{1}{2\varepsilon_0 A} Q^2 \\ F_F &= kx \end{aligned} \quad (2.27)$$

The system of equations (2.24) through (2.27) fully describes the system behavior already modeled in Example 2.1, though here it is in the form of a general DAE system integrating all involved physical domains (a domain-independent system model). Via suitable manipulation, this DAE system can, without much difficulty, be transformed into the state space form (2.15) (this is left as an exercise to the reader).

Thus, network-based modeling results in equations equivalent to those produced following the LAGRANGE formalism (which should not surprise the astute reader).

Nonlinear network representation (multi-domain) Network-based modeling offers additional advantages which have so far not been called upon in this exposition. Though the DAE system (2.24) through (2.27) represents a complete computational specification of the system behavior, it obscures topological relationships between the networks. The electro-mechanical coupling between the two physical domains occurs via the charge-dependent COULOMB force and the displacement-dependent capacitance. This coupling, which is only indirectly apparent in the DAE system and in Fig. 2.37, can, however, be made more visible in an extended network representation, where a *two-port element* (transducer) is introduced as a *coupling element* between the two networks (Fig. 2.38). Due to the nonlinear characteristics of the coupling, however, none of the simple transducer elements introduced above can be employed.

On the mechanical side, the configuration can also be described using domain-specific mechanical network elements. The resulting network is then directly coupled, clearly exposing the physical system topology and the coupling mechanism, but possesses a *heterogeneous* structure as relates to the *physical domains*. It is self-evident that applying the junction rule or mesh rule will lead to the same DAE equations.

Linear network representation The strengths of a network representation can be particularly well exploited in linear networks, e.g. the two-port representation, and graphical and analytical network analysis (Reinschke and Schwarz 1976). A linear description is possible in the present case, for example, if *small deviations* about a resting position of the system are considered. The linearization of system models is considered more closely in Sec. 2.6, so that only the results of this process are presented here.

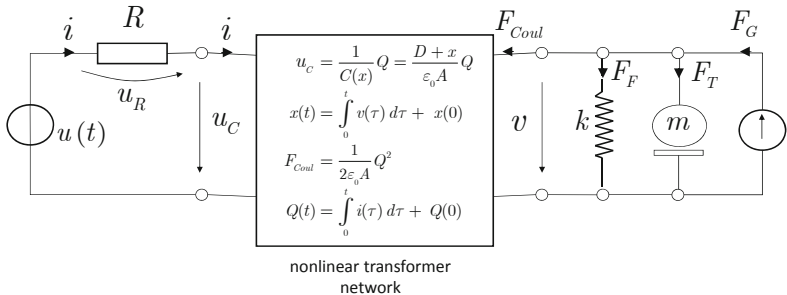


Fig. 2.38. Electrostatic saddle bearing: coupled domain-specific networks (nonlinear)

Resting position (compensation of the weight of the sphere with a charge Q^0 or equivalently a voltage u^0):

$$x^0 = \dot{x}^0 = 0, \quad Q^0 = \sqrt{2\varepsilon_0 A m g}, \quad u^0 = D \sqrt{\frac{2mg}{\varepsilon_0 A}}. \quad (2.28)$$

Linearized constitutive equations of the transducer ($\Delta i, \Delta u_c, \Delta F, \Delta v$ are displacements about the resting position):

$$\begin{aligned} \Delta F &= \sqrt{\frac{2mg}{\varepsilon_0 A}} \Delta Q \\ \Delta u_c &= \sqrt{\frac{2mg}{\varepsilon_0 A}} \Delta x + \frac{1}{C_0} \Delta Q \end{aligned} \quad \text{where } C_0 := \frac{\varepsilon_0 A}{D}. \quad (2.29)$$

Linear analogous electrical network For the network representation (Fig. 2.38), consistent network elements are now to be used—for example, the *electrical* elements from Table 2.5. For the mechanical elements, the corresponding analogs in Table 2.6 are to be used. An appropriate representation of the transducer network is somewhat trickier. Indeed, the constitutive relations (2.29) do not relate $\Delta F \leftrightarrow \Delta i$ as is needed, but rather $\Delta F \leftrightarrow \Delta Q = \int \Delta i \, d\tau$. This relationship can be represented in a symbolic, elegant manner via the LAPLACE transform of the integral. This results in an algebraic relationship $\Delta F \leftrightarrow \left(\frac{1}{s}\right) \Delta i$ and further, using a “complex” transducer constant N/s , a *transformer* transducer network (see Fig. 2.39).

The resulting network now exhibits a *homogeneous* structure in the *physical domains*. It can either be interpreted as a *generalized* network with abstract generalized components, or as an *analogous electrical* network with *electrical* network elements. The latter interpretation is particularly apt in the case where a corresponding computer-aided tool for electrical networks is available for further analysis and simulation tasks.

With some skill and experience, the above linear network can be converted into a (simpler) equivalent linear network with a real *lossless gyra-tor* (Fig. 2.40)¹⁵.

¹⁵ The configuration of the electrostatic saddle bearing examined here is equivalent to an *electrostatic plate transducer*, as is discussed in depth in, e.g., (Lenk et al. 2011) or (Senturia 2001). Further discussion of the system behavior (negative k_C !) follows in Ch. 6.

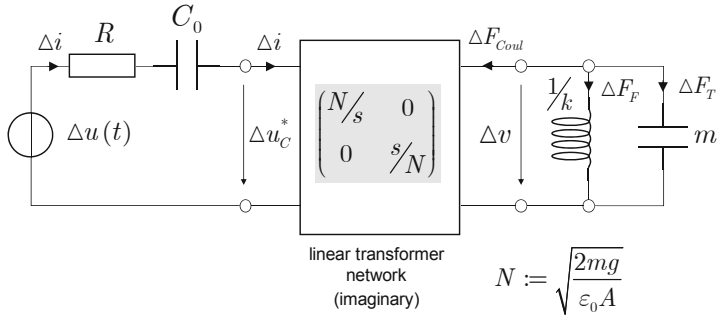


Fig. 2.39. Electrostatic saddle bearing: analogous electrical linear network (small deviations about equilibrium position); Variant 1 with imaginary *transformer network*

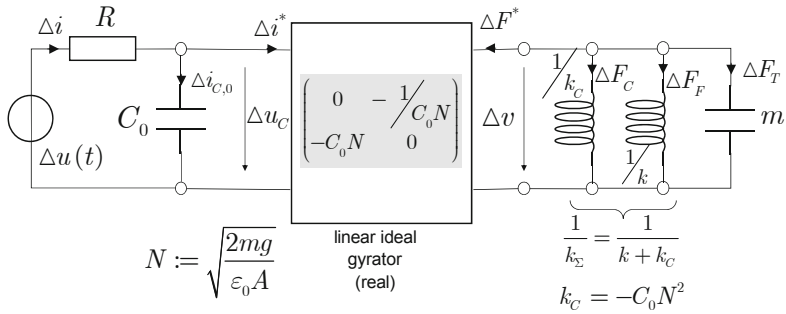


Fig. 2.40. Electrostatic saddle bearing: analogous electrical linear network (small deviations about equilibrium position); Variant 2 with real *gyrator network*

Ease of use and multi-domain properties The example discussed above very nicely demonstrates the strengths and weaknesses of network-based modeling. From the network topology, elementary circuit rules (junction rule, mesh rule), and constitutive equations with (lumped) network parameters, a mathematical model in the form of a (high-dimensional) DAE system can be systematically constructed. Via appropriate (automatable) manipulation, this can then be brought into a more concise form, e.g. a state-space representation (though in some cases, due to certain restrictions, this is only possible to a limited extent, see Sec. 2.4).

Among the advantages is certainly also the structure of the network model, which replicates the physical topology. This presents an excellent opportunity to *modularize* physical models by partitioning them into com-

ponent networks, with a clear assignment of components to physical subsystems. For this reason, network paradigms are the foundation for modern *object-oriented* modeling and simulation tools (see Sec. 2.3.9).

For linear network models, a well-researched methodological mechanism for *network analysis* is additionally available (e.g. while also exploiting topological properties (Reinschke and Schwarz 1976)). For electrical networks in particular, there is excellent support in the form of available tools, with which complex mechatronic systems can also be very efficiently modeled as *analogous electrical* networks in a domain-independent manner.

It must however be counted as a disadvantage that it is not always completely trivial to employ domain-independent analogs in a proper manner. This is made particularly difficult by the inconsistent physical assignment of conjugate power variables (especially for mechanical elements!). In addition, it is occasionally tricky—and thus creates a barrier for access to the “network world” for non-electrical engineers—to recast components into standard structures (see Fig. 2.40), for example to be able to use component libraries for computer-aided tools.

2.3.5 Multi-port modeling: bond graphs

Power bonds One particular network-oriented modeling approach is offered by so-called *bond graphs*. This approach has gained popularity lately—in particular in the field of mechatronics—though often its close relationship to the significantly older network modeling approach is obscured. Bond graphs employ a more concise, graph-oriented representation of network elements and their links via power variables (*power bonds*, see Fig. 2.41, left).

Graph-oriented model In place of two directed edges (arrows) for flow and effort, only a single weighted, directed edge is employed, on which symbolic identification of the flow and effort and their defined direction of effect are indicated in a concise form. The network parameters appear as weighted end nodes in the bond graph. Further, specialized nodes (*junctions*) are defined. Via a suitable assignment of definitions, a 1:1 depiction of heterogeneous networks as bond graphs on a homogeneous domain is more or less straightforwardly possible.

In order to give an impression of this type of modeling, Fig. 2.41 shows the bond graph model (Geitner 2008) equivalent to the network model in Fig. 2.38 of the electrostatic saddle bearing of Fig. 2.26.

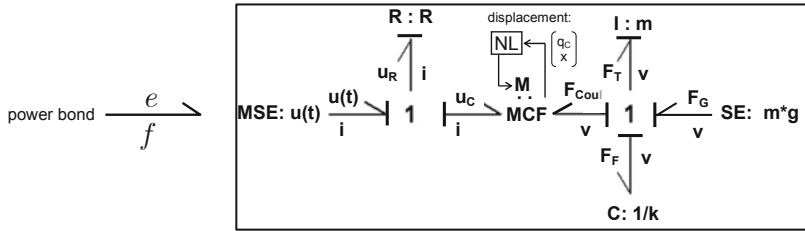


Fig. 2.41. Bond graph model of the electrostatic saddle bearing (Fig. 2.26), equivalent model to the network model in Fig. 2.38, from (Geitner 2008)

Ease of use Depending on the degree of familiarity with the various modeling formulations, one or the other of these representations may seem clearer to the user. Since modeling with bond graphs does not offer anything methodologically new as compared to network modeling, this approach will not be further discussed here. The interested reader is directed to the copious literature on this subject, e.g. (Paynter 1961), (Cellier 1991), (Damic and Montgomery 2003), (Karnopp et al. 2006).

Notation for power variables One remark relevant to applications will however not be dispensed with, in case the reader wishes to adapt *mechanical bond graph models* from the literature for her own purposes. In contrast to the network world, in the bond graph community, it is customary to use the physical assignments for power variables listed in Table 2.2, i.e. *effort:= force/torque*, *flow:= velocity/angular velocity* (though here too, the exception unfortunately proves the rule!). As a consequence, the assignment of network parameters also changes. Thus, when working with bond graph models, careful attention should be paid to the assignment of domain-specific quantities.

2.3.6 Energy / multi-port modeling: port-HAMILTONian systems

Foundations The advantages of universal scalar energy functions and the excellent modularizability of KIRCHHOFF networks are combined in the still rather young modeling approach of *port-HAMILTONian systems* (Maschke and van der Schaft 1992), (Cervera et al. 2007), (Duindam et al. 2009).

A *port-HAMILTONian* ($p\mathcal{H}$) system can be thought of as being constructed in the following way: take a (for now purely conservative) physical component system with HAMILTONian $\mathcal{H}(\mathbf{q}, \mathbf{p})$ which is excited by external generalized forces \mathbf{f} . The dynamics of the component system are described by HAMILTON's equations

$$\begin{aligned}\dot{\mathbf{q}} &= \frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial \mathbf{p}}, \\ \dot{\mathbf{p}} &= -\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial \mathbf{q}} + \mathbf{f}.\end{aligned}\tag{2.30}$$

The power used can be described as the product of conjugated power variables, i.e. in a concrete case, by the scalar product

$$P = \dot{\mathbf{q}}^T \mathbf{f}.$$

Power port If Eq. (2.30) is interpreted as a network element with power variables $\mathbf{u} = \mathbf{f}$ and $\mathbf{y} = \dot{\mathbf{q}}$ at its terminals¹⁶, then the system effects on the outside (at the *interface*) are described by the *vector* terminal variables $\langle \mathbf{u}, \mathbf{y} \rangle$ and its internal structure (*functionality*) is described by the *scalar* HAMILTONian $\mathcal{H}(\mathbf{q}, \mathbf{p})$ (Fig. 2.42).

It is easy to verify that the following energy conservation relation holds:

$$\frac{d}{dt} \mathcal{H} = \frac{\partial^T \mathcal{H}}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial^T \mathcal{H}}{\partial \mathbf{p}} \dot{\mathbf{p}} = \frac{\partial^T \mathcal{H}}{\partial \mathbf{p}} \mathbf{u} = \dot{\mathbf{q}}^T \mathbf{u} = \mathbf{y}^T \mathbf{u},$$

i.e. the increase in energy of the system equals the power applied to the port $\langle \mathbf{u}, \mathbf{y} \rangle$.

In Fig. 2.42, a concise, alternative representation of a generalized network element with a *power port* is recognizable. This can be linked in the normal manner with other components according to the circuit rules of KIRCHHOFF networks. It can be shown that connecting $p\mathcal{H}$ systems results in a $p\mathcal{H}$ system (van der Schaft and Maschke 1995).

¹⁶ The signals \mathbf{u} and \mathbf{y} are termed *collocated* in- and outputs of the $p\mathcal{H}$ system.

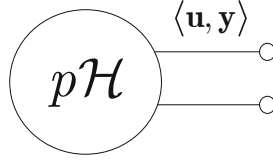


Fig. 2.42. Port-HAMILTONian system with power port

Generalized $p\mathcal{H}$ system HAMILTON's equations (2.30) can be represented in the following way:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{J} \frac{\partial \mathcal{H}(\mathbf{x})}{\partial \mathbf{x}} + \mathbf{G} \mathbf{u} & \mathbf{x} &= \begin{pmatrix} \mathbf{q} & \mathbf{p} \end{pmatrix}^T \\ \mathbf{y} &= \mathbf{G}^T \frac{\partial \mathcal{H}(\mathbf{x})}{\partial \mathbf{x}} & \text{where} & \mathbf{J} = \begin{pmatrix} \mathbf{0} & \mathbf{E} \\ -\mathbf{E} & \mathbf{0} \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} \mathbf{0} \\ \mathbf{E} \end{pmatrix}, \quad \mathbf{J} = -\mathbf{J}^T, \end{aligned} \quad (2.31)$$

and the skew-symmetric matrix \mathbf{J} describes the *internal structure* of the system (in the sense of a generalized geometric structure). This matrix is also called the POISSON structure matrix, and the system thus described (2.31) is thus said to have POISSON structure (van der Schaft and Maschke 1995).

The mathematical description introduced in Eq. (2.31) can be further generalized and extended with the introduction of dissipative elements, giving a *generalized port-HAMILTONian system* with dissipation:

$$p\mathcal{H} := \begin{cases} \dot{\mathbf{x}} = [\mathbf{J}(\mathbf{x}) - \mathbf{R}(\mathbf{x})] \frac{\partial \mathcal{H}(\mathbf{x})}{\partial \mathbf{x}} + \mathbf{G}(\mathbf{x}) \mathbf{u} \\ \mathbf{y} = \mathbf{G}^T(\mathbf{x}) \frac{\partial \mathcal{H}(\mathbf{x})}{\partial \mathbf{x}} \end{cases} \quad (2.32)$$

$$\begin{aligned} \text{where} \quad & \mathbf{J}(\mathbf{x}) = -\mathbf{J}^T(\mathbf{x}), \\ & \mathbf{R}(\mathbf{x}) \geq 0 \end{aligned}$$

i.e. the structure matrix \mathbf{J} is skew-symmetric and the dissipation matrix \mathbf{R} is symmetric positive-definite.

Significance From this intentionally short presentation of the port-HAMILTONian formulation it should have become clear where its strengths lie with respect to model creation. Via the power ports, the $p\mathcal{H}$ formulation supports modular modeling at the physical level and permits the direct linking of physical structures (equivalent to KIRCHHOFF networks and

bond graphs). An additional recognizable advantage is the concise description of internal functionality via the scalar HAMILTONian, as well as the clear depiction of the internal working structure of the system in the POISSON structure and dissipation matrices. The greatest challenge, however, consists of the construction a suitable HAMILTONian.

As was mentioned in the beginning, the port-HAMILTONian formulation is still relatively young and is the subject of current research. In this context, it should be mentioned that the significance of the $p\mathcal{H}$ formulation appreciably exceeds the narrower task of modeling. For example, the formal description of a system as a POISSON structure shown in Eq. (2.32) presents valuable properties which directly support the construction of nonlinear control algorithms and stability proofs (Ortega et al. 2002), (Kugi and Schlacher 2002), (Kugi and Schlacher 2001), (Fuchshumer et al. 2003). One further focus of current research concerns the extension to infinite-dimensional system descriptions (via partial differential equations) (van der Schaft and Maschke 2002).

2.3.7 Signal-coupled networks

One important set of questions to be asked in the context of model creation deals with the modularization of dynamic models. This is particularly of interest in the case when the models are to be used libraries as part of computer-aided simulation tools.

The following questions are appropriate in this context:

- When and under what conditions can models of physical (component) systems be directly connected to each other?
- Why are signal-oriented models of physical systems only conditionally modularizable?

Both questions are fundamentally connected to the problem of *back-effects* between systems.

Power flow The physical coupling between system components always occurs via a power flow (Fig. 2.43). This means that for system models, at every interface, the corresponding conjugate power variables effort e and flow f should be examined: it is already known that the transferred energy is equal to the product of the power variables. In the general case, inter-

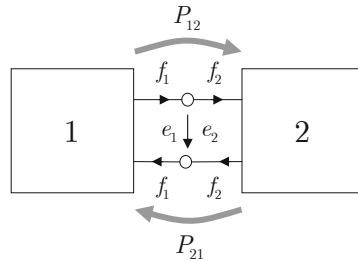


Fig. 2.43. Coupling of physical system models

faces between system models must thus always contain a *pair of conjugate power variables* (e, f) . One additional condition which must be met is that *both* interface variables must be compatible with respect to the *physical domains* considered and the *physical units* used, i.e. $[e_1] = [e_2]$ and $[f_1] = [f_2]$ in Fig. 2.43. Thus, for example, only electrical power variables from the same physical domain (volt, ampere) may be connected to one another; the direct coupling of electrical and mechanical interface variables is, in contrast, trivially impossible.

Power back-effect Why must both power variables always be taken into account?

In a one-port element, at most one of the two power variables can be freely controlled, the other power variable is determined by the coupling. If, for example, in the connected system in Fig. 2.43, the effort e_1 can be arbitrarily set via an ideal potential source, the arising flow f_1 —and as a consequence, due to the coupling conditions, also the flow f_2 —depends on the structure of System 2, which serves as the load for System 1. From the perspective of the power flow P_{12} , in this sense there is a *back-effect* from System 2 to System 1. The flow f_1 can thus only be computed if the inner structure or the behavior at the terminals of System 2 is known.

System coupling via *power flows* is always subject to power *back-effects*.

Signal coupling: absence of back-effects How, then, do models behave in which no appreciable power flow occurs?

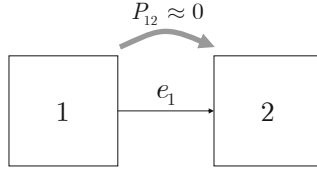


Fig. 2.44. Signal-coupled model for vanishing power flow between Systems 1 and 2 (back-effect-free linking)

This is the case, for example, when one of the two power variables possesses a very small value. In a system configuration as shown in Fig. 2.43, given a very large input resistance in System 2, only vanishingly small flow f_1 (or f_2) appears. The transferred power $P_{12} = e_1 \cdot f_1$ is thus approximately zero. Thus, the interface between Systems 1 and 2 becomes dependent on *only one single* interface variable, in this case the effort e_1 . The particular characteristics of System 2 have no influence whatsoever on the effort e_1 . In this context, the relationship between the two Systems 1 and 2 is called *back-effect-free* and the connection is termed *signal-coupled* using a single interface variable (the *signal*).

Definition 2.6. *Signal coupling* – A signal flow between two physical systems implies an information flow with negligible power flow, i.e. one of the two conjugate power variables at the system interface is approximately zero. The non-zero power variable is termed the connecting *signal* between the two systems. Under these conditions, the interface between two systems is described solely by the connecting signal, the two systems are said to be *signal-coupled*.

Signal-coupled model The type of back-effect-free action relationship between Systems 1 and 2 in Fig. 2.43 is commonly depicted in the form of a *signal-coupled model* (Fig. 2.44). Well-known examples of signal-coupled models are, e.g., *control system block diagrams*.

In signal-coupled networks (in general), all *flows equal zero*; the *efforts* (measured with respect to a common reference) serve as *signal quantities* and alone describe the flow of action.

In the case of system connection via *signals*, *back-effect-free* interaction of the components is always presumed.

Example 2.3 Back-effect in an RC network.

System configuration Fig. 2.45 shows an RC network whose input is excited via an ideal potential source u_Q and whose output is connected to a one-port with impedance Z_L as a load.

Model creation The mathematical model of the RC network is formally given in Eqs. (2.24), (2.25) (assuming a constant capacitance C). Given the constitutive load relation (LAPLACE-transformed variables)

$$U_L(s) = Z_L \cdot I_L(s),$$

it follows for the input interface variables that¹⁷

$$U_s(s) = \left(1 + \frac{R}{Z_L} + RCs \right) U_L(s),$$

$$I_s(s) = \left(\frac{1}{Z_L} + Cs \right) U_L(s),$$

and for the *load-dependent* input impedance

$$Z_s(s) = \frac{U_s(s)}{I_s(s)} = \frac{1 + \frac{R}{Z_L} + RCs}{\frac{1}{Z_L} + Cs}. \quad (2.33)$$

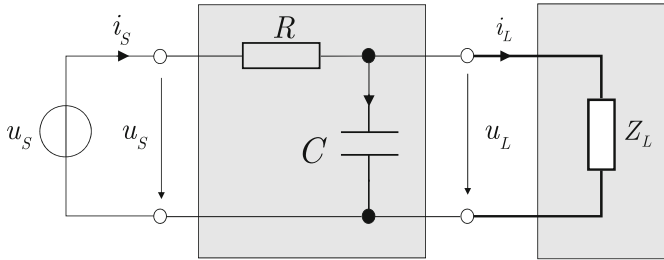


Fig. 2.45. Loaded RC network

¹⁷ A model with equivalent computational structure taking the form of *linear differential equations* can be obtained in the well-known manner using the inverse LAPLACE transform.

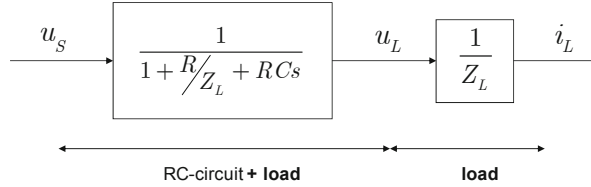


Fig. 2.46. RC network with back-effect as a signal-oriented block diagram obscuring the physical system topology

Signal-coupled model Even in the signal-oriented block-diagram representation (using transfer functions), representing the dependence on the load in the system model is ungainly. It is certainly possible to represent the impedance load in its own block. However, the model of the RC network itself also depends on the impedance load Z_L . In this case, then, using a *signal-oriented* graphical model (Fig. 2.46) hides the original physical structure, and thus a modular signal-coupled model at the level of physical components is not possible. ■

Example 2.4 Back-effect-free connection for an RC network.

System configuration Fig. 2.47 depicts the same RC network as in Example 2.3, though here, the impedance load is connected via a so-called *isolation amplifier*. The isolation amplifier possesses a high-resistance input stage (input current $i_{in} \approx 0$). At the output, a low-resistance voltage source $u_{out}(u_{in}) = V \cdot u_{in}$ is controlled by the input voltage u_{in} . The input power draw is vanishingly small, and the output power (determined by the impedance load Z_L) is provided by the controlled voltage source. Note, however, that this requires an external auxiliary energy source. In the back-effect case in Example 2.3, the power for the load Z_L was provided by the voltage source u_S .

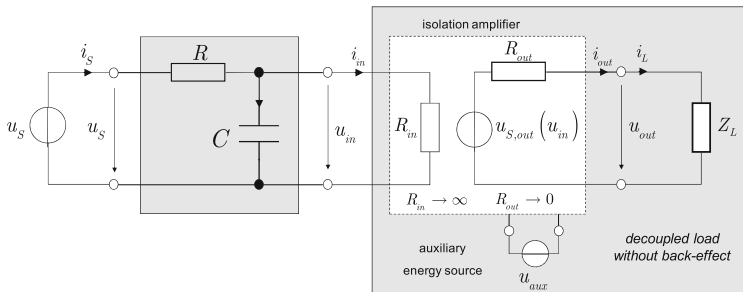


Fig. 2.47. Loaded RC network with isolation amplifier

Model creation The mathematical model of the network is once again formally given in Eqs. (2.24), (2.25), and, since $i_m \approx 0$, it holds that

$$\begin{pmatrix} U_s(s) \\ I_s(s) \end{pmatrix} = \begin{pmatrix} 1 + RCs & R \\ Cs & 1 \end{pmatrix} \begin{pmatrix} U_m(s) \\ 0 \end{pmatrix}.$$

Noting that $u_{out}(u_m) = V \cdot u_m$, it follows that

$$\begin{aligned} U_{out}(s) &= \frac{V}{1 + RCs} U_s(s), \\ I_s(s) &= \frac{Cs}{1 + RCs} U_s(s). \end{aligned} \quad (2.34)$$

In this case, the impedance of the networks at the exciting input is independent of the load (compare Eqs. (2.33), (2.34)). The voltage u_{out} at the load can be controlled without back-effect (independent of the actual load) by the voltage source $u_{out}(u_m)$.

Signal-coupled model This state of affairs can also be discerned from the signal-oriented block diagram (transfer function, Fig. 2.48). In addition, the block diagram allows the separation of the RC network and the load via the isolation amplifier to be distinguished, reflecting the physical topology. In this case, then, the original physical structure of the system can be easily reproduced in the signal-oriented graphical model. It is only under the conditions given here that *signal-oriented* modeling at the level of physical components is possible.

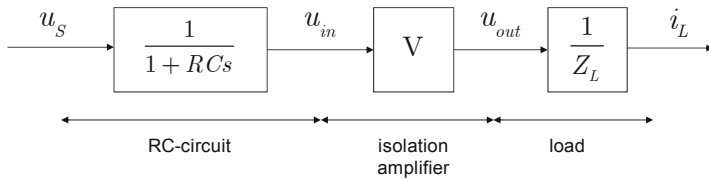


Fig. 2.48. Back-effect-free RC network as a signal-oriented block diagram with clear replication of the physical system topology

Further examples of back-effect-free system interfaces—and thus signal-oriented models—are *measurement amplifiers*, *power stages* for actuators, *information processing* systems, and *analog* and *digital signal processing* elements. The latter can be used with familiar block diagrams from control theory (transfer functions) in the accustomed manner.

2.3.8 Model causality

Causal vs. acausal models Within the context of modeling of physical systems, the terms *causal* and *acausal model* often appear, usually in connection with certain modeling paradigms or simulation tools. This section attempts to place these imprecise and sometime incorrectly used designations into a consistent framework, and to explain their meanings.

Causality in system theory In the context of the design of physical systems, the term *causality* should be used in accord with the established meaning in system theory.

Definition 2.7. *Causality:* A system with input $u(t)$ and output $y(t)$ ¹⁸ is called *causal* if at any time t_1 , the output $y(t_1)$ is only affected by the evolution of the input $u(t)$ up to time t_1 (Fig. 2.49). A system is called *acausal* if it does not meet the above condition.

At its core, this definition states that in a realizable, physical system, no future inputs can affect the current output. In this sense, naturally all mechatronic systems are *causal by definition*, and the same should consequently also be expected of their dynamic models¹⁹.

Causal structure Given the above definition, it is quite clear that system dynamics must exhibit a *cause-and-effect* behavior (input-output, excitation-response). In an unconnected multi-port model (with open terminals), any particular system variable can be defined as a cause or an effect by asking the question of how a particular *experiment* will be conducted (e.g. simulation, frequency response, theoretical noise analysis). Only with a clearly defined experiment frame is a definite *causal structure* established within the model in the form of unambiguous *cause-and-effect relationships*. It is thus completely incorrect to speak of a “causal” or “acausal” model given an unconnected multi-port model²⁰.

¹⁸ The limited formulation here of single-input single-output (SISO) systems can easily be extended to systems with an arbitrary number of inputs and outputs.

¹⁹ One exception, however, is so-called *prediction models*, which are *acausal* in the above sense.

²⁰ Models referred to as “acausal” (e.g. networks, bond graphs, object-oriented models) are often pointed to as having the most beneficial of properties. However, such a paradoxical labeling unnecessarily obscures the actual strengths of such models.

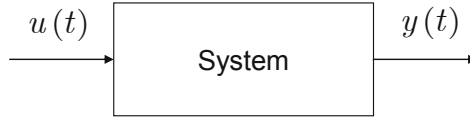


Fig. 2.49. Causal structure (cause → effect) of a system

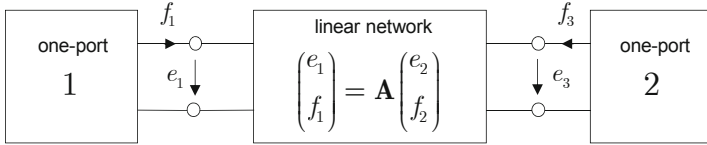


Fig. 2.50. Linear network with indeterminate causal structure

What is actually meant by such statements is whether the experiment-dependent cause-and-effect relationship—and thus the casual computational structure—of an abstract model is determined or still open. To be correct, it is thus better to speak of models with *determinate* or *indeterminate causal structure*.

Causal structures in network models Without more detailed specification of the input or output connections, network models generally have an *indeterminate causal structure*. Fig. 2.50 shows a linear network model, described by its chain matrix \mathbf{A} ²¹

$$\begin{pmatrix} e_1 \\ f_1 \end{pmatrix} = \mathbf{A} \begin{pmatrix} e_2 \\ f_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} e_2 \\ f_2 \end{pmatrix}. \quad (2.35)$$

The actual model equations are described by the matrix \mathbf{A} . The behavior of the network in response to external “excitations” depends on the external connections of the network to One-ports 1 and 2. Each of these one-ports can represent a passive load or an active network, e.g. a source.

The input, or cause, of a causal structure can only be an independent source, i.e. either a potential or flow source. The description given in Eq. (2.35) does not, however, answer the questions of (a) which of the two one-ports will act as the source and (b) which type of source is being considered, i.e.

$$\begin{aligned} &\text{potential source } e_i(t), \quad i = 1, 2 \\ &\text{flow source } f_i(t), \quad i = 1, 2. \end{aligned}$$

²¹ In general, complex elements $a_{ij}(s)$ can also be incorporated, i.e. arbitrary passive (generalized) network elements R, L, C can be included in the circuit.

Thus, the model (2.35) can be considered complete, yet *causally indeterminate*. However, in order to carry out an experiment, a causal structure with an excitation and one or more responses must be identified. If, for example, a potential source $e_1(t)$ is chosen as the excitation (One-port 1) and the effort $e_2(t)$ is taken as the output with an open circuit (One-port 2 has infinite input resistance), then an *unambiguous casual structure* has been defined and the experiment (determining temporal evolution, frequency response, noise response, etc.) can be carried out. It is in this sense that the model can be said to have *determinate causal structure*.

Example 2.5 *Causal structures for an RC network.*

Indeterminate external connection Consider the RC network with *indeterminate external connections* shown in Fig. 2.51. A mathematical dynamic model can be directly constructed by applying the modeling methodology introduced in Sec. 2.3.3 (using the junction rule mesh rule, and constitutive equations with lumped, constant parameters). Using LAPLACE-transformed system variables and the complex chain matrix, this model can be concisely represented as

$$\begin{pmatrix} U_1(s) \\ I_1(s) \end{pmatrix} = \begin{pmatrix} 1 + RCs & R \\ Cs & 1 \end{pmatrix} \begin{pmatrix} U_2(s) \\ I_2(s) \end{pmatrix} = \mathbf{A}(s) \begin{pmatrix} U_2(s) \\ I_2(s) \end{pmatrix}. \quad (2.36)$$

As is readily apparent, cause-and-effect relationships in Eq. (2.36) can be seen in either direction. Without more detailed specification of the external connections—i.e. constraints on the input and output power variables u_1, i_1, u_2, i_2 —the *causal structure* of the model (2.36) remains *indeterminate*.

Load For the first concrete experiment, let One-port 2 be further specified: let it be a passive network with infinite input resistance, i.e. no current should be able to flow into One-port 2 (an open circuit at terminal 2 of the RC network).

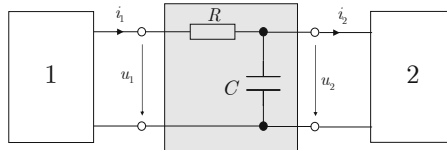


Fig. 2.51. RC network with indeterminate external connections

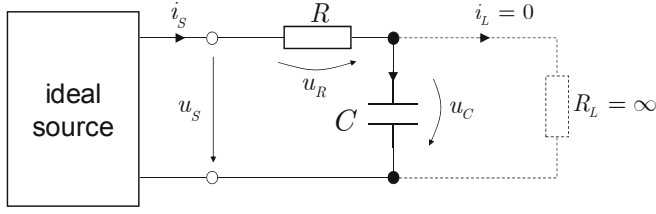


Fig. 2.52. Load-free RC network with indeterminate source

Source This makes it clear that One-port 1 must be regarded as a source (Fig. 2.52). However, the causal structure has still in no way been unambiguously determined. The question of which of the two power variables at the source are to be independently forced—serving as the *excitation* for the network (or *input of the causal structure*)—still remains open. Which of the system variables should be regarded as *response variables* (or *outputs of the causal structure*) also remains to be defined.

A mathematical model of the load-free RC network with an indeterminate source—in the form of a DAE system—thus has the following form:

$$\begin{aligned} u_s - u_R - u_C &= 0, \\ u_R &= R i_s, \\ \dot{u}_C &= \frac{1}{C} i_s, \\ u_s(i_s) &= 0. \end{aligned} \tag{2.37}$$

As can be easily recognized, a self-contained, complete description of the dynamics of this system is available from the model in Eq. (2.37). All that remains to be determined is the algebraic relation between source voltage and source current.

Experiments In order to be able to investigate concrete system behaviors (experiments), suitable, independently controllable excitations must be defined. For this purpose, in the present case, the only possible candidates are the two source power variables .

Case A: *ideal voltage source* \rightarrow excitation $u_s = u_s(t)$, no constraints on $i_s(t)$, the output is considered to be all time-varying system variables (Fig. 2.53a).

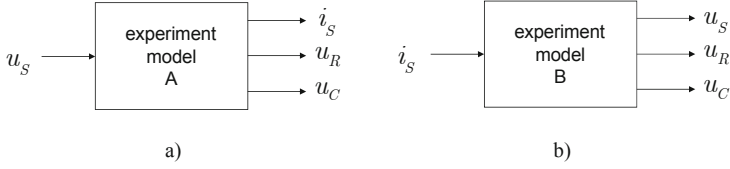


Fig. 2.53. Causal structures for possible experimental models of the RC network: a) voltage excitation, b) current excitation

Mathematical model for *Experiment A* (via simple manipulation of the model in Eq. (2.37)):

$$\begin{aligned}
 \dot{u}_C &= -\frac{1}{RC} u_C + \frac{1}{RC} u_s , \\
 i_s &= -\frac{1}{R} u_C + \frac{1}{R} u_s , \\
 u_R &= u_s - u_C .
 \end{aligned} \tag{2.38}$$

Case B: *ideal current source* \rightarrow excitation $i_s = i_s(t)$, no constraints on $u_s(t)$, the output is considered to be all time-varying system variables (Fig. 2.53b).

Mathematical model for *Experiment B* (also via simple manipulation of the model in Eq. (2.37)):

$$\begin{aligned}
 u_s &= Ri_s + \frac{1}{C} \int i_s(\tau) d\tau , \\
 u_R &= Ri_s , \\
 \dot{u}_C &= \frac{1}{C} i_s .
 \end{aligned} \tag{2.39}$$

Now both experiments exhibit clear causal structures in their models (2.38), (2.39), which additionally present a causal computational structure, i.e. all unknown quantities (outputs, system responses) can be computed explicitly using the DAE systems (2.38) and (2.39) given known input functions $u_s(t)$ or $i_s(t)$ (this is a determining property for simulation experiments).



Computational causality A different causality concept is connected with the technical solution of a DAE system for simulation—that is, carrying out a (simulation) experiment. The term *computational causality* describes a suitable *structure* and *sequencing* of the equations of the DAE model permitting the *sequential numerical* solution of the DAE system.

For example, the system of equations (2.37)—even given a defined source—does not permit sequential computation. In contrast, the system of equations (2.38) presents a causal computational structure. In the first equation, given the source voltage u_Q , the capacitor voltage u_C can be computed. In the next step, using u_Q and u_C , the source current i_Q is obtained, and in the third step, the voltage across the resistor is determined (steps 2 and 3 can also be interchanged). In modern simulation tools, this manipulation of equations is automated using symbolic computation.

An extended interpretation of computational causality additionally requires *explicit* representation of the equations in the form $a = f(b, c)$ with known variables b and c , as opposed to an implicit representation $a = f(a, b, c)$. Implicit equations actually pose no extraordinary problems for modern simulation tools, so that nowadays, explicit representation of the equations is not a particular requirement.

Causal structure vs. modeling approaches

Multi-ports It should be noted that, in a way, multi-port modeling automatically results in models in the form of DAE systems with indeterminate causal structure. This in turn means that the causal structure only needs to be established at the time of experimentation by fixing the assignment of inputs and outputs. The DAE model is thus universally reusable once generated (see Eq. (2.37)). Of course, *before* carrying out the actual experiments, the equations must be suitably manipulated to achieve a causal computational structure. This task can, however, be delegated to a computer-aided tool, just as can be done for determining the DAE system equations from the network topology. This approach is followed by modern equation-oriented modeling and simulation tools, e.g. object-oriented modeling with MODELICA (Tiller 2001) and network-based modeling with VHDL-AMS (Schwarz et al. 2001).

LAGRANGE formalism In comparison, in the modeling approach using the LAGRANGE formalism, specification of the causal structure must occur significantly earlier in the modeling process. The generalized coordinates

must already be established during the process of setting up the LAGRANGIAN equations. As previously mentioned, these must be independent and unconstrained when dealing with *EULER-LAGRANGE* equations of the second kind. For example, in Example 2.5, given a voltage excitation, charge would be used as a generalized coordinate; on the other hand, given a current excitation, flux linkage would be used (see also Table 2.2). Furthermore, the forcing function must be defined in the form of an external generalized force or potential function. As a result, each input case directly gives rise to its own particular model with a determinate causal structure—though naturally, this is ultimately identical to the equivalent network-based model (see Eqs. (2.38), (2.39)).

Evaluation Given the perspectives presented here, the network-based modeling approach is justifiably favored within the technical community. Due particularly to the substantial available tool set, it is eminently suited to (very) large, complex systems. Certainly, it should also be pointed out again that the advantages of multi-port physical modeling come from its equation-oriented description with indeterminate causal structure, and not, mistakenly, from “acausal” models.

2.3.9 Modular modeling of mechatronic systems

DAE systems as dynamic models When speaking of the modular modeling of mechatronic systems, it is first and foremost desirable that the *physical topology* of real systems be equivalently replicated in their dynamic models. Such computable dynamic models (see also Fig. 2.4) are given in a general form by a system of *differential-algebraic equations* (DAE system)²² (here in semi-explicit form, see Sec. 2.4):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}, t), \quad (2.40)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{z}, t). \quad (2.41)$$

As a typical example of a mechatronic system, consider the *servo-controlled drive shaft* (e.g. in a robot or machine tool) depicted in Fig. 2.54.

²² This applies in any case for most physical system components which can be described via *continuous-time models*. Switching, mechanical contact issues, stick-slip behavior, and discrete-time and discrete-event phenomena (mostly in the context of information processing) can be described using specialized model extensions termed *hybrid models* (see Sec. 2.5).

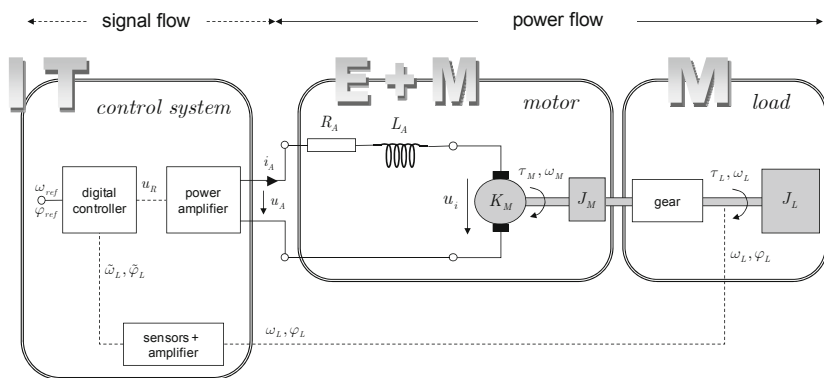


Fig. 2.54. Example of a multi-domain model: servo-controlled drive shaft (e.g. robot, machine tool)

Requirements A useable *modular dynamic model* for a mechatronic system should fulfill the following general *requirements*:

1. The *physical system topology* should remain visible in the system of equations (2.40), (2.41), e.g. the armature inductance should have a physical relation to the armature resistance and motor, the transmission should have a physical relation to the motor and load axes.
2. Models of individual system elements should be *interchangeable* while maintaining the power relation at interfaces (i.e., power-conserving), e.g. a rigid transmission interchangeable with an elastic transmission, a rigid load interchangeable with a multibody system.
3. Models of system elements should support *hierarchical* operations (decomposition, aggregation), e.g. aggregate motor model \rightarrow new model “*Motor*”, decomposed transmission model \rightarrow detailed component model.
4. System components from *different physical domains* (multi-domain components) should be described using *consistent models*; this requirement is automatically met at the computational level using a DAE system (2.40), (2.41), but is equally desirable for abstract model precursors (graph-oriented models).
5. *Coupling* should be possible for *power-flow-* and *signal-oriented* models, e.g. control \rightarrow motor, load \rightarrow control.

Requirements 2 and 3 generally suggest the use of *model libraries*; Requirement 4 can be implemented with *domain-specific* model libraries.

Computer-aided tools For the practical manipulation of modular system models, the use of *computer-aided tools* is of particular interest (for managing large systems, design automation, etc.). In the process, the distinction between actual *model creation*, and *experimentation* on these models (simulation, analysis) should be kept in mind (cf. Sec. 2.1). In this section, it is model creation (Eqs. (2.40), (2.41)) which is of interest; particular aspects of technical implementations for simulation are discussed in Ch. 3.

The remainder of this section concisely discusses the strengths and weakness of the various modeling approaches presented so far in terms of their suitability for computational implementation while retaining the form of modular physical models.

Energy-based modular modeling

Strengths and weaknesses The LAGRANGE formalism presented in Sec. 2.3.2 is actually not suited to modular modeling. Requirements 2 and 4 can be implemented with uncoupled coordinates at the energy level using suitable model libraries. In the same way, conversion into a DAE system can, in principle, be automated using computer algebra systems (for differentiation of the LAGRANGian). System coupling is represented by equating coordinates in the problem. However, the coupling of power flows behind these coordinates is only indirectly visible via differentiation of the LAGRANGian, and is thus not evident in the DAE system. This is likely the reason that no multi-domain tools employing this paradigm have penetrated the market.

Multi-port modular modeling

Strengths and weaknesses From the material in the preceding sections, it should already be clear that essentially only *multi-port* modeling approaches can fulfill all of the modularity requirements presented above. They inherently exhibit all necessary properties, such as *power-conserving interfaces*, *open causal structures*, *multi-domain properties* via the analogous substitutions, and *generalized network elements* (see Examples 2.2 through 2.4).

The *hierarchy requirements* can also—at least in the case of linear models—easily be met using concise *two-port descriptions* with complex elements (see Eq. (2.36)).

At this point, the particular assignment of power variables to mechanical quantities used in network theory can finally be clearly justified (see Table 2.3).

The assignment is as follows

flow := *force or torque*

effort := *velocity or angular velocity*.

For example, examining the *rigid coupling* between motor axle and transmission (τ_M, ω_M) in the servo-controlled drive shaft in Fig. 2.54 from a physical point of view shows that the angular velocities—not the torques—must be identical at the interface. However, following the rules of network theory, such a condition is only possible for effort variables, from which it follows that the (*angular*) *velocity* must be used as the effort variable (see Example 2.7).

Note that in the bond graph world, the opposite assignment (that in Table 2.2) is typically used, complicating the representation of rigid couplings (see Example 2.6).

Exploiting the advantageous properties of port-based modeling, a series of powerful modeling and simulation tools have been developed, which in particular support *domain-specific model libraries* at the physical level, and which appreciably lighten the burden of the modeling task, e.g. (Schwarz et al. 2001), (Geitner 2006).

Example 2.6 *Port-based model (bond graph) of a drive shaft.*

System configuration Consider the servo-controlled drive shaft in Fig. 2.54. Of particular interest is the rigid coupling between the motor arbor and the transmission to the load. This physical interface is to be replicated with as little modification as possible in a modular dynamic model.

Bond-graph-based dynamic model As laid out in the preceding paragraphs, most bond graph tools employ forces/torques as mechanical effort variables. As explained above, this convention brings with it certain complication for the case of rigidly-coupled arbors. Fig. 2.55a shows a bond graph model (Geitner 2008) in which such a rigid coupling is in fact directly implemented, though with an accompanying loss of correspondence between physical and model modules. The coupling is only realizable by accounting for the combined moment of inertia $J_{ges} = J_M + J_L$ at the driving input (the motor).

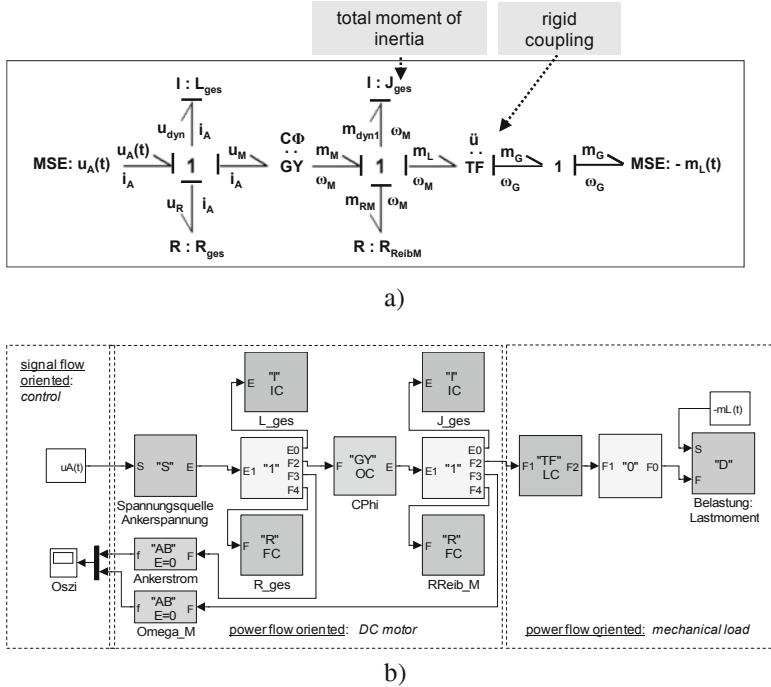


Fig. 2.55. Bond graph model for the servo-controlled drive shaft in Fig. 2.54 with a *rigid* motor-transmission coupling: a) bond graph, b) bond graph computer model embedded in SIMULINK, from (Geitner 2008)

In contrast, Fig. 2.56a shows a modular bond graph model which strictly maintains correspondence to physical modules, so that the motor and load each retain their individual inertia parameters (Geitner 2008). Admittedly, in order to maintain consistency at the interfaces—i.e. the coupling of the velocities as flows—a virtual *elastic coupling* employing a high-stiffness spring had to be introduced. In this way, relative motions between the two arbors are allowed, though they remain small only with very high spring stiffness. In addition to this thoroughly problematic *model falsification*²³, a new problem has been introduced which impacts the simulation of the system. The new model is a stiff differential system of equations and is hence more difficult to solve numerically when performing simulation experiments (see Ch. 3).

²³ It is a question of the system point of view—i.e. of what the modeling objective is—whether this model modification is treated as a *falsification* of the rigid coupling assumption, or as an *enhancement* of the modeling of physical system properties (strictly speaking, there are no real “rigid” bodies).

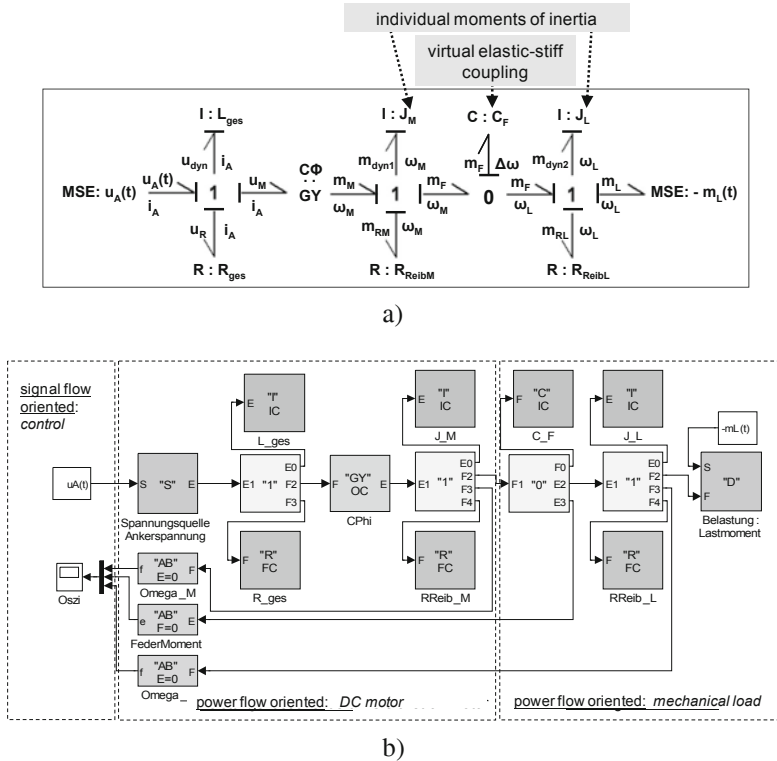


Fig. 2.56. Bond graph model for the servo-controlled drive shaft in Fig. 2.54 with a *virtual stiff elastic* motor-transmission coupling: a) bond graph, b) bond graph computer model embedded in SIMULINK, from (Geitner 2008)

Bond-graph-based computer models are shown in Fig. 2.55b and Fig. 2.56b for both cases considered (embedded here in the computer-based tool SIMULINK, (Geitner 2006)).

Signal-oriented modular modeling

Strengths and weaknesses Modularization using signal-oriented models—while meeting the Requirements 1 through 4 established above—is only meaningfully possible for *functional physical device groups* having *back-effect-free* coupling.

For example, in Fig. 2.54, this would include independent modules labeled “sensor measurement amplifier”, “controller”, “power amplifier”, and a closed (aggregate) module “motor + load”.

For fundamental reasons, library modules for elementary physical elements and components which have a power flow cannot be constructed in a meaningfully usable fashion (due to the back-effect problem, see Sec. 2.3.5).

However, if connections are sufficiently back-effect-free, working with *signal-coupled models* is quite uncomplicated. This type of modeling is most suitably introduced from a control theoretical viewpoint. In addition to being employed to perform simulation experiments, linear system models can be directly used to great advantage in analysis tasks (control loop structuring, aggregated transfer functions, etc.).

Working at the signal level, libraries of modules can be easily created, modified, and used. Suitable elementary models include linear and nonlinear *algebraic operators* (e.g. summation, multiplication by constant parameters), as well as modules for *integration* (integrators). Using such component, models in *state-space form*²⁴

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \quad (2.42)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t), \quad (2.43)$$

are easily modularized.

Note however, that connecting several state-space models (2.42), (2.43) is only straightforward if no *algebraic loops* appear (corresponding to a index-0 DAE system, see Sec. 2.4).

Due to this methodologically simpler process, computer-aided tools for signal-oriented modeling, simulation, and analysis (MATLAB/SIMULINK, LABVIEW) are significantly more widely distributed in practice than those which are network based. Thus, in many cases, a signal-oriented modeling approach is preferred intuitively, despite the fact that this entails considerable *limitations* on physical modular models incorporated into libraries.

²⁴ Here, as is usual, \mathbf{u} represents the vector of inputs, \mathbf{y} the vector of outputs, and \mathbf{x} the state vector (see also Sec. 2.6).

A characteristic example highlighting the inherent obstacles to modularizability was already presented in Sec. 2.3.7 (see Example 2.3). The problem of rigid coupling in mechanical systems introduced there is thoroughly discussed in Sec. 2.4 (DAE systems, index-3 systems). For the case of signal-oriented modeling (as with the bond graph model), only two solutions come into question, neither of which is fully satisfactory: a *common* model including both masses (the correspondence to physical objects is lost), or *elastic coupling* with sufficient stiffness (resulting in a stiff system of differential equations).

Object-oriented modeling of physical systems

Dual meaning of “object-oriented” The advantageous aspects of *network-based* modeling discussed above can be efficiently combined with *object-oriented* concepts from software design to create computer-based modeling implementations. The resulting paradigm—termed *object-oriented modeling of physical systems*—employs the attribute “object-oriented” in two ways:

- object-oriented in the *conceptual* sense: using *encapsulated dynamic models* of physical components (e.g. capacitors, transmissions, masses, motors) while maintaining power-based interrelationships,
- object-oriented in the *software* sense: using *encapsulated software entities* with classic object-oriented software properties (polymorphism, inheritance, hierarchical classes, etc.).

Object-oriented computer tools *Computer-based tools* created following the object-oriented physical modeling paradigm enable—alongside a *user-friendly model representation*—automated model creation and computer-based simulation and analysis (set point computation, linearization, frequency response calculation, etc.). Fundamentally, this can be viewed as creating a class of generalized network analyzers having a multi-domain man-machine interface.

Fig. 2.57 depicts the generalized model hierarchy for object-oriented physical modeling, cf. the generalized model hierarchy in Fig. 2.4.

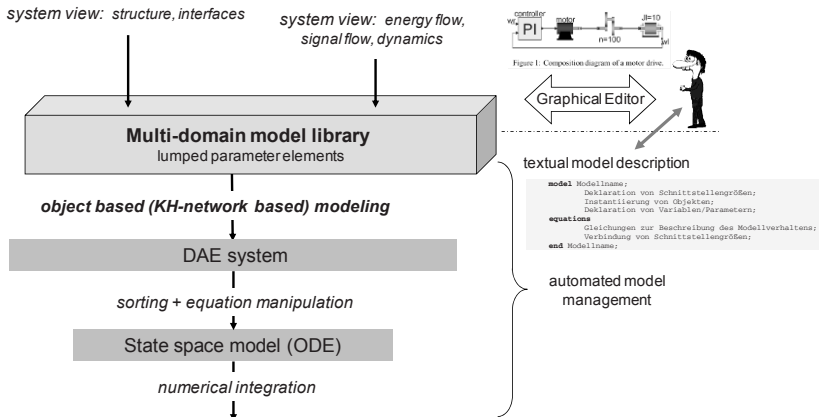


Fig. 2.57. Model hierarchy for object-oriented physical modeling

Model description languages At the core of object-oriented modeling approaches lie *textual model description languages*. It is here that the concepts of network modeling and of object-oriented software design are united, including:

- the mathematical description of real processes in encapsulated objects,
- models implemented as (object) classes;
- hierarchical class structures: the parent-child concept \Rightarrow specialization at lower levels of the hierarchy;
- inheritance of properties from superordinate classes in the hierarchy;
- polymorphism: inherited properties can be locally overridden or, in the case of multiple definitions, the property of the closest ancestor applies;
- the use of models in the form of (object) instances (predefined in the form of classes);
- simplified reuse of models and components.

MODELICA An important example worth mentioning is the language MODELICA²⁵ (Fritzson 2011), (Tiller 2001), which offers several specialized attributes:

- a model-specific rather than mathematically-oriented description,
- application neutrality,
- is a quasi-standard for object-oriented modeling languages,

²⁵ www.modelica.org

- supports mixed continuous-discrete systems (hybrid systems),
- offers tracking and accounting of physical units.

The translation of textual models into a DAE system along with subsequent model manipulation in computer-based tools takes place in a largely automated fashion using powerful *transformation algorithms* (Otter 1999; Otter and Bachmann 1999; Otter and Bachmann 1999).

Graphical editors Though not methodologically relevant for modeling, an additional helpful (and attractive!) user option offered by several tools is a powerful *graphical editor*, where—using domain-specific *object diagrams*—heterogeneous (e.g. mechatronic) systems can be graphically composed and automatically translated into the textual description language.

Advantages for the user The particular *advantages* of object-oriented modeling tools lie in their accessibility for the user (particularly if a graphical editor is available), the straightforward reusability of previously-constructed models (evolving model libraries), inherently physically accurate model composition using the network approach, and efficient, automated management of models.

A superficial consideration of the model hierarchy shown in Fig. 2.57 would give the impression that a user has only to open the right toolkit, arrange and connect the components in a physically relevant manner and voila! an accurate system model is generated.

It is, however, precisely in such simplistic user interventions (lacking the requisite technical understanding), in combination with concealed, automated model management processes, that, in the experience of the author, lie potential dangers which should not be underestimated.

Problems during use Even if it is assumed that computer-based tools contain validated software (but what software is really error-free?), sufficient opportunity still exists to create an incorrect or fundamentally numerically ill-conditioned model, which will hit (often mostly hidden) snags only during the course of experimentation (simulation). It should be noted that *all* possible sources of difficulty discussed in *Chapter 3: Simulation Issues* can also appear precisely in object-oriented models. In many cases, these problems can be countered with a suitable parameterization of the simulation algorithms; in other cases, however, they can only be overcome with changes to the model. Knowledge of these aspects of modeling is indispensable even—and precisely—when using object-oriented computational tools.

Example 2.7 *Object-oriented model of a drive shaft.*

System configuration Consider again the servo-controlled drive shaft in Fig. 2.54. In the example models so far (bond graphs, signal-oriented), the rigid coupling between the motor arbor and the transmission to the load could only be represented in a physical, modular model in a very limited manner. The following modeling approach using the modeling language MODELICA permits *freely configurable, port-based* model interfaces, and thus enables the much more straightforward generation of general physical, modular models and model libraries.

MODELICA dynamic model The object-oriented dynamic model presented below is based on (Schwarz and Zaiczek 2008) and is characterized by:

- on the mechanical side: a port for torque and angular rotation,
- on the electrical side: two terminals, which combine to form an electrical port,
- a “ground” (at the “top-level schematic”) which also should not be omitted from hand-drawn system descriptions,
- the load: damping (with value “d”) proportional to the angular velocity,
- “*protected*” variables which cannot be used outside of their scope of validity (encapsulation, a typifying property of object orientation).

The models are annotated in such a way as to be understandable to the unpracticed eye, so that the model hierarchy and the interface concept should be self-evident to the knowledgeable reader. In order to simulate these models using a computational tool, they must first be suitably prepared (e.g. modules must be arranged as *files* in *directories* or compiled into *packages*). These latter issues are however questions of implementation, exceeding the scope of modeling methodology of interest here. For a detailed understanding of the modeling language MODELICA, the interested reader is referred to e.g. (Tiller 2001).

Rigid mechanical coupling The strengths of the *port-based* approach of a modeling language like MODELICA now become apparent. Due to the *user-specific* choice of flow and effort, suitable, consistent variables can be defined at the mechanical ports of the drive and load shafts for the case presented here, directly enabling their rigid coupling. In this example, the most obvious choice is to select motion variables as the effort, that is, the *angular velocity* or the *rotation angle* of the shaft. Both quantities are equally permissible (see remarks in Sec. 2.3.4 in the paragraph “Domain-specific flow and effort variables”) and must be—both as efforts and as physical quantities—identically equal for rigid arbor cou-

pling. For convenience, for servo control (as is the case here), the *rotation angle* will be chosen as the effort variable. Naturally, constitutive equations should be matched to the choice of port variables.

MODELICA program module (complete model, from (Schwarz and Zaiczek 2008))

```
model Motor_Multiport_complete "consists of motor + voltage source + load +
ground"
// Description of the overall system, "top-most" model level
// Motor connected to voltage source and load
// List of all models used
Motor_MultiportModel MM;
SourceOnePort source;
Load_damping load;
Ground ground;
// Linking of models
equation
  connect( source.p, MM.p );
  connect( source.n, MM.n );
  connect( source.n, erde.p );
  connect( MM.portM, load.port );
end Motor_Multiport_complete;
```

```
model Motor_MultiportModel "motor model as reusable multi-port"
//Model parameters: normalized, dimensionless
parameter Real RA = 1;
parameter Real LA = 1;
parameter Real JM = 2;
parameter Real KM = 3;
//electrical and mechanical terminals
Pole p " together form " ;
Pole n " an electral port " ;
MechanicalPort portM;
//"protected" encapsulates internal data (object-oriented!)
protected
Real tauE;
Real uA;
Real iA;
Real ui;
Real tauM;
Real omM;
Real omE;
Real phiE;

equation
  //internal model
  uA = RA*iA + LA*der(iA) + ui;
  tauM = tauE + JM*der(omM);
  tauM = KM*iA;
  ui = KM*omM;
  omE = omM;
  der(phiE)=omE;

  // communication with the environment, interface
  p.v-n.v = uA;
  p.i = iA;
  p.i+n.i = 0 "electrical port condition";
  portM.tau = tauE;
  portM.phi = phiE;
end Motor_MultiportModel;
```

```
connector MechanicalPort
  Real phi;
  flow Real tau;
end MechanicalPort;
```

```

model Load_Damping
  MechanicalPort port;
  parameter Real d=1;
  equation
    port.tau = -d*der(port.phi);
  end Load_Damping;

```

```

model SourceOnePort
  parameter Real val=1.0;
  Pole p;
  Pole n;
  equation
    p.v-n.v = if time >= 0 then val else 0.0 "step function as example" ;
    p.i+n.i = 0 "electrical port condition";
  end SourceOnePort;

```

```

model Ground "required for definition of reference node"
  Pole p;
  equation
    p.v=0;
  end Ground;

```

```

connector Pole
  Real v "potential at terminal";
  flow Real i "current flow into the terminal";
  end Pole;

```

Dynamic model as object diagram The benefits of an object-oriented modeling language like MODELICA can be taken advantage of in user-friendly computer-aided modeling and simulation tools, such as DYMOLA²⁶ and SIMULATIONX²⁷. These also employ so-called *object diagrams*, easing model creation (for the possible hazards, see above).

Fig. 2.58 shows such an object diagram created with the modeling tool DYMOLA (Schwarz and Zaiczek 2008). This diagram uses basic blocks from the MODELICA standard library. Here, no load has been connected on the mechanical side (free running motors).

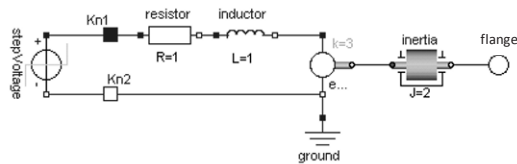


Fig. 2.58. DYMOLA object diagram for the motor of the servo-controlled drive shaft in Fig. 2.53, from (Schwarz and Zaiczek 2008)

²⁶ www.dymola.com

²⁷ www.simulationx.com

The object diagram has a *textual* description generated automatically by the simulator, but which could also be entered manually (cf. the tool architecture in Fig. 2.57). For each model component (Resistor, Inductor, EMF, Inertia, etc.), the encapsulated program modules (see above) and functional and interface reference descriptions (e.g. the meanings of the parameters R , L , k , J) are stored in a model library. An example of the automatically-generated text description is shown in excerpt below. The functional description of the model modules is equivalent to the MODELICA models presented above.

DYMOLA / MODELICA Structural description (excerpt)

```
model Motor_with_voltage_step
// List of all components
Modelica.Electrical.Analog.Basic.Resistor resistor(R=1) ;
Modelica.Electrical.Analog.Basic.Inductor inductor(L=1) ;
Modelica.Electrical.Analog.Basic.EMF eMF(k=3) ;
Modelica.Mechanics.Rotational.Inertia inertia(J=2) ;
Modelica.Electrical.Analog.Interfaces.NegativePin Kn2 ;
Modelica.Electrical.Analog.Interfaces.PositivePin Kn1 ;
Modelica.Mechanics.Rotational.Interfaces.Flange_b Flange ;
Modelica.Electrical.Analog.Sources.StepVoltage stepVoltage ;
Modelica.Electrical.Analog.Basic.Ground ground ;
// component connections
equation
connect(resistor.n, inductor.p) ;
connect(inertia.flange_a, eMF.flange_b) ;
connect(eMF.n, Kn2) ;
connect(inertia.flange_b, Flange) ;
connect(inductor.n, eMF.p) ;
connect(resistor.p, Kn1) ;
connect(stepVoltage.p, Kn1) ;
connect(stepVoltage.n, Kn2) ;
connect(eMF.n, ground.p) ;
end Motor_with_voltage_step;
```

2.4 Systems of Differential-Algebraic Equations

2.4.1 Introduction to DAE systems

DAE systems In the course of modeling the dynamics of mechatronic systems, *systems of differential equations* appear which are subject to *algebraic constraints*. Multi-port modeling using KIRCHHOFF networks discussed in Sec. 2.3.4 naturally leads to this form of model, as does the application of the EULER-LAGRANGE equations of the first kind given holonomic and nonholonomic constraints²⁸.

²⁸ These models consist of a system of second-order differential equations and can easily be transformed into the standard form of first-order differential equations used here.

Such systems of equations are called *differential-algebraic equations* (DAEs). As will be shown in Ch. 3, the numeric solution of such models requires the application of special methods.

For reasons of space, only a short introduction to the important terms and relationships fundamental to understanding DAE systems as applied to system models can be given here. For a deeper understanding, the reader is referred to the very readable and extensive monographs (Cellier and Kofman 2006) and (Brenan et al. 1996).

Representations

- *implicit* DAE system

$$\mathbf{0} = \tilde{\mathbf{f}}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{z}, \mathbf{u}, t)$$

$\mathbf{x}(t) \in \mathbb{R}^n$ state variables, appear with differentials

$\mathbf{z}(t) \in \mathbb{R}^m$ algebraic variables

$\mathbf{u}(t) \in \mathbb{R}^r$ inputs (externally imposed)

$\tilde{\mathbf{f}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \times \mathbb{R} \rightarrow \mathbb{R}^{n+m}$... set of differential and algebraic equations

Note: a total of $(n + m)$ equations is required for the $(n + m)$ unknowns $\mathbf{x}(t)$, $\mathbf{z}(t)$.

- *semi-explicit* DAE system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}, t) \quad (2.44)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{z}, t) \quad (2.45)$$

$$\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \times \mathbb{R} \rightarrow \mathbb{R}^n, \mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^m$$

Index of a DAE system

Difficulty of solving a DAE The classification of DAE systems commonly takes place using a special measure, called the *index* i of the DAE system. In a certain manner, this measure describes the difficulty of solving the DAE system. A system of ordinary differential equations has index $i = 0$, DAE systems have $i > 0$. A DAE system is called *high index* if $i \geq 2$.

However, there are varying definitions of the index, so that in some cases, different orderings are possible. In this book, the most commonly-used definition will be referenced: the *differential index*.

Definition 2.8. *Differential index:* The *differential index* is the minimum number of differentiations required for the equations of a DAE system to arrive at a system of explicit ordinary differential equations.

2.4.2 DAE index tests

Goal: Determination of the *differential index*, i.e. the minimal number of differentiations d/dt of Eq. (2.45) required so that—given Eq. (2.44)—a system of explicit ordinary differential equations ensues.

Index-1 systems

$$\frac{d}{dt} \mathbf{g}(\mathbf{x}, \mathbf{z}) = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial \mathbf{g}}{\partial \mathbf{z}} \dot{\mathbf{z}} = \mathbf{0}$$

$$\text{substituting Eq. (2.44): } \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{f} + \frac{\partial \mathbf{g}}{\partial \mathbf{z}} \dot{\mathbf{z}} = \mathbf{0}$$

- *Index-1 condition*

$$\det \left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}} \right) = \det \begin{pmatrix} \frac{\partial g_1}{\partial z_1} & \dots & \frac{\partial g_1}{\partial z_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial z_1} & \dots & \frac{\partial g_m}{\partial z_m} \end{pmatrix} \neq 0 \quad (2.46)$$

If Eq. (2.46) holds, then the algebraic constraint can be written as a system of first-order differential equations

$$\dot{\mathbf{z}}(\mathbf{x}, \mathbf{z}, \mathbf{u}) = - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}} \right)^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{f}.$$

Example 2.8 *RC network.*

System configuration Consider an unloaded RC network following Fig. 2.52, with an ideal (lossless) voltage source $u_s(t)$ and constant components R, C .

Modeling as a KIRCHHOFF network gives the mathematical model

$$\text{Conservation equation: } u_s - u_R - u_C = 0$$

$$u_R = Ri_s$$

$$\text{Constitutive equations: } \dot{u}_C = \frac{1}{C} i_s$$

and, along with the definitional relations

$$x := u_C, \quad z_1 := i, \quad z_2 := u_R, \quad u := u_s,$$

this gives the DAE system in semi-explicit standard form:

$$\begin{aligned} \dot{x} &= f(\mathbf{z}) = \frac{1}{C} z_1 \\ 0 &= g_1(\mathbf{z}) = Rz_1 - z_2 \\ 0 &= g_2(x, \mathbf{z}, u) = x + z_2 - u. \end{aligned}$$

The index-1 condition

$$\det \left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}} \right) = \det \begin{pmatrix} \frac{\partial g_1}{\partial z_1} & \frac{\partial g_1}{\partial z_2} \\ \frac{\partial g_2}{\partial z_1} & \frac{\partial g_2}{\partial z_2} \end{pmatrix} = \det \begin{pmatrix} R & -1 \\ 0 & 1 \end{pmatrix} = R \neq 0$$

holds for all $R > 0$, so that this represents a DAE system with differential index $i = 1$.



Index-2 systems²⁹

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{x})\end{aligned}\quad (2.47)$$

$$\frac{d}{dt} \left[\frac{d}{dt} \mathbf{g}(\mathbf{x}) \right] = \frac{d}{dt} \left[\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \dot{\mathbf{x}} \right] = \frac{d}{dt} \left[\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}) \right] = \mathbf{0} \Rightarrow \dots + \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{f}}{\partial \mathbf{z}} \dot{\mathbf{z}} = \mathbf{0}$$

 • **Index-2 condition³⁰**

$$\det \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{f}}{\partial \mathbf{z}} \right) = \det \left(\begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \dots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \dots & \frac{\partial g_m}{\partial x_n} \end{pmatrix} \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \dots & \frac{\partial f_1}{\partial z_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial z_1} & \dots & \frac{\partial f_n}{\partial z_m} \end{pmatrix} \right) \neq 0 \quad (2.48)$$

If Eq. (2.48) holds, then the algebraic constraint (2.47) can again be written as a system of first-order differential equations: $\dot{\mathbf{z}} = \dot{\mathbf{z}}(\mathbf{x}, \mathbf{z}, \mathbf{u})$.

Example 2.19 *Linear index-2 system.*

Problem statement Determine the differential index of the following DAE system ($\phi(t)$ is a given function of time):

$$\dot{x} = f(x, z) = (z - x)a,$$

$$0 = g(x, t) = x - \phi(t).$$

Solution

Test index-1 condition: $\frac{\partial g}{\partial z} = 0$ condition violated \rightarrow index > 1 .

Test index-2 condition: $\frac{\partial g}{\partial x} \frac{\partial f}{\partial z} = 1 \cdot a \neq 0 \rightarrow$ index $i = 2$.



²⁹ To simplify the presentation, only index-2 candidates are considered for which the index-1 condition Eq. (2.46) is *not* met, i.e. the algebraic constraints are *independent* of the algebraic variables.

³⁰ Note: This condition only holds for the case $\mathbf{g} = \mathbf{g}(\mathbf{x})$, i.e. *independence* from the algebraic variables \mathbf{z} .

Index-3 systems

Negative index test If Eq. (2.48) does not hold, then the DAE system possesses an index $i > 2$. In most cases, this means it is an index-3 system. This should be considered a very difficult problem, as will be elucidated in the following typical example.

Example 2.10 *Rigidly-coupled two-mass system.*

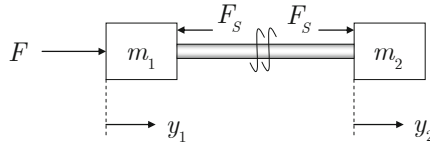


Fig. 2.59. Rigidly-coupled two-mass system

System configuration Let two masses be coupled via a rigid connection (Fig. 2.59). Find the equations of motion and the constraining force in the connecting rod, as well as the classification of the resulting DAE system.

Model Using LAGRANGIAN equations of the first kind gives the *equations of motion*³¹

$$\begin{aligned} m_1 \cdot \ddot{y}_1 &= F - F_s, \\ m_2 \cdot \ddot{y}_2 &= F_s, \end{aligned}$$

and the *holonomic* constraint (rigid coupling)

$$y_2 - y_1 = 0.$$

Using the definitions $x_1 := y_1$, $x_2 := \dot{y}_1$, $x_3 := y_2$, $x_4 := \dot{y}_2$, $u := F$, $z := F_s$, gives the equivalent DAE system in *semi-explicit* form:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{1}{m_1}z + \frac{1}{m_1}u \\ x_4 \\ \frac{1}{m_2}z \end{pmatrix} = \mathbf{f}(\mathbf{x}, z, u), \quad (2.49)$$

³¹ The constraint force F_s represents the LAGRANGE multiplier in Eq. (2.11).

$$0 = x_3 - x_1 = g(\mathbf{x}). \quad (2.50)$$

Note: $m = 1$, i.e. there is *one* algebraic variable and *one* algebraic equation.

Index test

$$\text{Index-1 condition } \frac{\partial g}{\partial z} = 0 \rightarrow \text{violated}$$

$$\text{Index-2 condition } \frac{\partial g}{\partial \mathbf{x}} \frac{\partial \mathbf{f}}{\partial z} = \begin{pmatrix} -1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -1/m_1 \\ 0 \\ 1/m_2 \end{pmatrix} = 0 \rightarrow \text{Index} > 2 !$$

Index determination What is the index of the DAE system (2.49), (2.50)?

Solution: Applying the definition of the differential index, the minimal number of differentiations d/dt of Eq. (2.50) which—given Eq. (2.49)—result in a system of explicit ordinary differential equations can be found.

$$(1) \frac{d}{dt} \text{Eq. (2.50)} \rightarrow \dot{x}_3 - \dot{x}_1 = 0$$

$$\text{with Eq. (2.49):} \quad x_4 - x_2 = 0 \quad (2.51)$$

$$(2) \frac{d}{dt} \text{Eq. (2.51)} \rightarrow \dot{x}_4 - \dot{x}_2 = 0$$

$$\text{with Eq. (2.49):} \quad \left(\frac{1}{m_1} + \frac{1}{m_2} \right) z - \frac{1}{m_1} u = 0 \quad (2.52)$$

$$(3) \frac{d}{dt} \text{Eq. (2.52)} \rightarrow \left(\frac{1}{m_1} + \frac{1}{m_2} \right) \dot{z} - \frac{1}{m_1} \dot{u} = 0$$

$$\text{ODE} \quad \boxed{\dot{z} = \frac{m_2}{m_1 + m_2} \dot{u}} \quad (2.53)$$

\rightarrow Index $i = 3$ since the original algebraic condition (2.50) had to be differentiated three times in order to arrive at an ODE (2.53) in the algebraic variable z .

This state of affairs applies *in general* for mechanical systems with *rigid coupling*.

2.4.3 DAE index reduction

Goal Solution of a DAE system (2.44), (2.45) via *explicit* numerical integration (an *ordinary differential equation (ODE) solver*, e.g. RUNGE-KUTTA).

Solution Using *index reduction*, the DAE system is re-formed into a system of ordinary differential equations, i.e. the algebraic variables also become defined by a system of differential equations. One fundamental hurdle in the process is the determination of consistent initial values $\mathbf{x}(0)$, $\mathbf{z}(0)$ for the state and algebraic variables.

Index reduction by a factor k

- *differentiate* the algebraic equations (2.45) k times: $\frac{d^k}{dt^k}(\mathbf{g}(\mathbf{x}, \mathbf{z}, t)) = \mathbf{0}$,
i.e. since $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^m$, there are m new differential equations
- k statements of consistent *initial values* from the nonlinear system of equations:

$$\left. \begin{array}{l} \mathbf{g}(\mathbf{x}(0), \mathbf{z}(0), t=0) = \mathbf{0} \quad \dots \text{algebraic equations} \\ \frac{d}{dt} [\mathbf{g}(\mathbf{x}, \mathbf{z}, t)]_{\mathbf{x}=\mathbf{x}(0), \mathbf{z}=\mathbf{z}(0), t=0} = \mathbf{0} \\ \vdots \\ \frac{d^{k-1}}{dt^{k-1}} [\mathbf{g}(\mathbf{x}, \mathbf{z}, t)]_{\mathbf{x}=\mathbf{x}(0), \mathbf{z}=\mathbf{z}(0), t=0} = \mathbf{0} \end{array} \right\} \dots (k-1) \text{ derivatives}$$

- $\mathbf{x}(0)$, $\mathbf{z}(0)$ must satisfy the above system of equations, i.e. they generally can not be chosen independently of one another.

Further index reduction procedures Index reduction of DAE systems is the key to successfully employing object-oriented physical models. As will be shown in Ch. 3, only DAE systems with sufficiently low index can be solved in a numerically stable, usable form. For this reason, index reduction should always be attempted for DAE systems with a higher index to bring them into a tractable form. In addition to the further readings presented above, the interested reader is directed to three core original publications: (Pantelides 1988), (Cellier and Elmqvist 1993), (Mattsson and Söderlind 1993).

Example 2.11 *Index reduction for a rigidly-coupled two-mass system.*

Algebraic conditions: consistent initial conditions The calculations in Example 2.10 lead to the DAE system (2.49), (2.50). Derived from the thrice-differentiated algebraic constraints, the algebraic conditions below hold for the entire trajectory (cf. Eqs. (2.51) through (2.53)) and thus for $t = 0$, from which follow *consistent* initial conditions:

$$\left. \begin{array}{l} x_1(t) = x_3(t) \\ x_2(t) = x_4(t) \\ z(t) = \frac{m_2}{m_1 + m_2} u(t) \end{array} \right\} \forall t \Rightarrow t = 0 : \quad \begin{array}{l} x_1(0) = x_3(0) \\ x_2(0) = x_4(0) \\ z(0) = \frac{m_2}{m_1 + m_2} u(0) \end{array} \quad (2.54)$$

Given the initial conditions (2.54) and the differential equations (2.49), (2.53), a causal block diagram simulation model can now be constructed, where the differentiation of the input value (Fig. 2.60a) can be eliminated by integrating Eq. (2.53) (Fig. 2.60b).

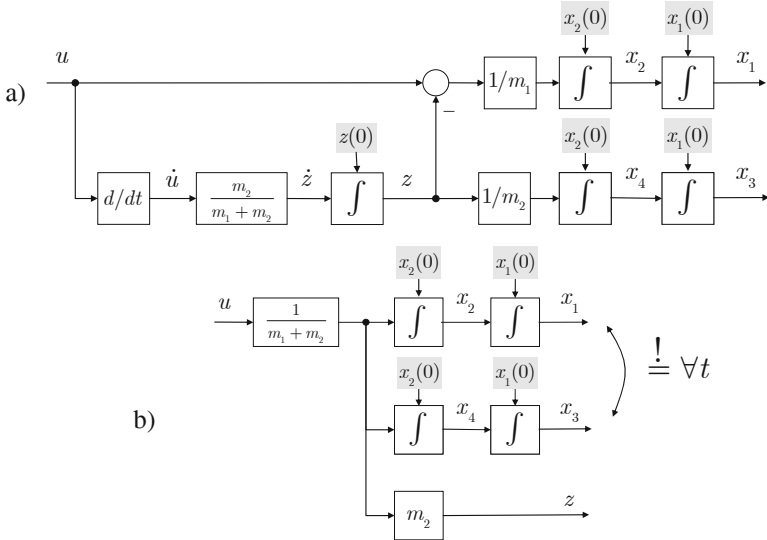


Fig. 2.60. Signal-oriented simulation model of the rigidly-coupled two-mass system: a) with integrator for the algebraic variable z , b) equivalent representation with integrator for the algebraic variable z eliminated

2.5 Hybrid Systems

2.5.1 General structure of a hybrid system

Hybrid systems: terminology In many technical systems, especially if the boundaries of the system are sufficiently broadly defined (see Sec. 2.1) or if the system dynamics are modeled with sufficient accuracy, the simultaneous appearance of continuous-time and discrete-event phenomena will be observed. Such a system model is then termed a *hybrid dynamic system*, referred to concisely as a “*hybrid system*” below³².

For example, in mechatronic systems, a hybrid description is called for when, during the course of operation, the governing mathematical model changes due to a deterministic external influence (e.g. user intervention) or state-dependent conditions (e.g. mechanical contact effects, state-dependent parameter changes). Such hybrid dynamics resulting from continuous-time and discrete-event changes—and particularly their resulting interactions—must be appropriately represented in a *combined* system model (Engell et al. 2002), (Buss 2002).

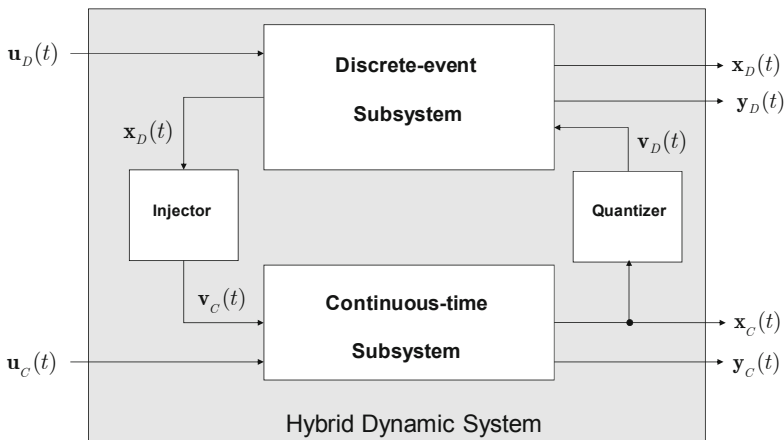


Fig. 2.61. General structure of a hybrid dynamic system

³² Often mixed *continuous-time*, *discrete-time* models are termed *hybrid*. At the simulation implementation level, such models should be considered special cases of hybrid discrete-event, continuous-time systems, in which the events occur at well-defined, predetermined, periodic times. The principle of energy-based adaptive step size (Sec. 3.7) can be correspondingly applied in a greatly simplified form.

Hybrid system structure The general structure of a *hybrid discrete-event, continuous-time* system (or *hybrid* system) is represented in Fig. 2.60 (Lunze 2002). A hybrid system comprises the following components and system variables:

$\mathbf{x}_c, \mathbf{u}_c, \mathbf{y}_c$	continuous states/inputs/outputs
$\mathbf{x}_d, \mathbf{u}_d, \mathbf{y}_d$	discrete states/inputs/outputs
<i>Continuous-time subsystem</i>	the principal, continuous-time system dynamics
<i>Discrete-event subsystem</i>	varying operating states of the system
<i>Injector</i>	unique map from a discrete-valued signal defined over a finite set of symbols to a real-valued signal
<i>Quantizer</i>	discrete-valued map of a real-valued signal

Research in this area is still relatively young, and as yet no generally-valid, easily-applicable methods particular to the design and analysis of hybrid systems have been established. The situation is somewhat better in the area of methodological support and toolset implementation for modeling and simulation. This is of rather more practical significance, as such tools enable—at least at the model level—the prediction of complex hybrid system dynamics via simulation (design verification). The remainder of this section introduces the most significant hybrid phenomena in mathematical notation (Lunze 2002); presents a specialized, practical approach—*net-state models* (Nenninger et al. 1999)—and discusses particulars of the implementation of simulations.

2.5.2 Hybrid phenomena

Hybrid system Consider as given the following description of a *continuous subsystem*³³

$$\dot{\mathbf{x}} = \mathbf{f}^1(\mathbf{x}, \mathbf{u}, t) \quad \text{where } \mathbf{x} \in M_1, \quad (2.55)$$

$$\dot{\mathbf{x}} = \mathbf{f}^2(\mathbf{x}, \mathbf{u}, t) \quad \text{where } \mathbf{x} \in M_2, \quad (2.56)$$

$$\dot{\mathbf{x}} = \mathbf{f}^\delta(m^\delta(\mathbf{x}, \mathbf{u}, t)), \quad (2.57)$$

³³ This representation can also be directly applied to DAE systems, i.e. the conditions presented for the vector field of the state variable derivatives (= right-hand side of the ODE) hold equivalently for the vector field of the algebraic constraints.

with *discontinuity hypersurface*

$$m(\mathbf{x}, \mathbf{u}, t) = 0. \quad (2.58)$$

Discontinuous vector field One characteristic of hybrid systems is the existence of discontinuities in the vector field of derivatives of the state variables (= right-hand side of the ordinary differential equations (2.55) through (2.57)). In such cases, the smoothness of the vector field is violated, i.e. there exist states \mathbf{x}_s , for which *no* LIPSCHITZ condition of the type

$$\begin{aligned} \|\mathbf{f}^i(\mathbf{x}_s, \mathbf{u}, t) - \mathbf{f}^j(\tilde{\mathbf{x}}, \mathbf{u}, t)\| &\leq L \cdot \|\mathbf{x}_s - \tilde{\mathbf{x}}\|, \quad i, j \in \{1, 2, \delta\} \\ \|\bullet\| &\text{arbitrary vector norm, } L \in \mathbb{R}^+ \text{ finite} \end{aligned} \quad (2.59)$$

exists (holds similarly for \mathbf{u} if its components $u_j(t)$ exhibit step-wise changes).

Sets of states The sets M_1, M_2 describe regions in the n -dimensional state space \mathbb{R}^n in which the corresponding vector fields $\mathbf{f}^1(\mathbf{x}, \mathbf{u}, t), \mathbf{f}^2(\mathbf{x}, \mathbf{u}, t)$ define the dynamics of the continuous subsystem. The boundary between M_1 and M_2 is described by the *discontinuity hypersurface* (2.58). Systems with a description following Eq. (2.55), (2.56) are distinguished by the fact that the *states* themselves remain *continuous* at the discontinuity hypersurfaces, whereas in the case of a system following Eq. (2.57) so-called *state discontinuities* appear (Fig. 2.62).

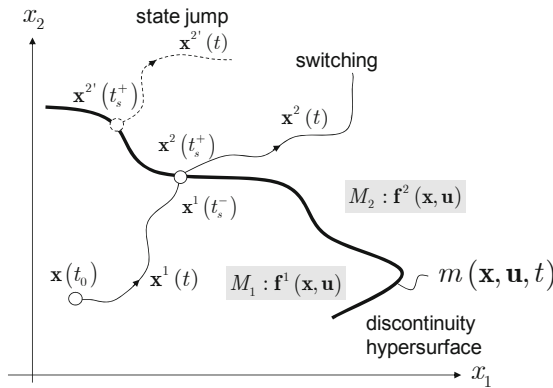


Fig. 2.62. Discontinuity properties of hybrid phenomena

Hybrid dynamics of systems can be systematically described by the following four *hybrid phenomena*. In concrete cases, an arbitrary combination of these phenomena can be present.

Autonomous switching

Entry of the trajectory of the continuous system into a particular subset of the continuous state space which induces a change in the discrete state, which in turn modifies the continuous system dynamics (\Rightarrow Eqs. (2.55), (2.56), (2.58); states remain continuous).

Controlled switching

An *external* intervention triggers a change in the discrete state, which in turn modifies the continuous system dynamics (\Rightarrow Eqs. (2.55), (2.56), (2.58), discontinuous change in $\mathbf{u}(t)$; states remain continuous).

Autonomous state jumps

Entry of the trajectory of the continuous system into a particular subset of the continuous state space which induces a change in the discrete state, which directly triggers a jump in the continuous state (or subset of the state variables) (\Rightarrow Eq. (2.57)).

This behavior can be illustrated with a *first-order system*:

$$\left. \begin{aligned} \dot{x}(t) &= a \cdot \delta(x(t) - x_s) \\ x(t_s) &= x_s \end{aligned} \right\}$$

When the threshold x_s is exceeded, the state immediately jumps by an amount a , i.e. the derivative becomes infinite, as modeled by the DIRAC delta function³⁴ $\delta(\bullet)$.

$$\Rightarrow \quad x(t_s + 0) = x(t_s - 0) + a \quad \text{and} \quad x(t_s) = x(t_s^-) + a = x_s + a.$$

³⁴ More precisely, this is a DIRAC distribution, as the argument is a function of time.

Controlled state jumps

An external intervention triggers a change in the discrete state, which in turn directly triggers a jump in the continuous state (\Rightarrow Eq. (2.57)).

To illustrate, once again consider a *first-order system*:

$$\left. \begin{aligned} \dot{x}(t) &= a \cdot \delta(u(t) - u_s) \\ u(t_s) &= u_s \end{aligned} \right\}$$

$$\Rightarrow x(t_s + 0) = x(t_s - 0) + a \quad \text{and} \quad x(t_s) = x(t_s^-) + a$$

Note: $u(t)$ is continuous, i.e. does not have a jump/DIRAC character.

Switching occurs most commonly (e.g. structure-variable control), *discontinuities* are more rare, e.g. in the case of collisions.

2.5.3 Net-state models

Model structure The general structure of a *net-state model (NSM)* is shown in Fig. 2.63. Here, the discrete-event (DE) subsystem is realized as an *interpreted PETRI net (IPN)* (with inputs and outputs) (Litz 2005). When combined with a continuity condition on states in the continuous-time subsystem (*extended state space model (ESM)*), all above-mentioned hybrid phenomena can be modeled using a net-state model (Nenninger et al. 1999).

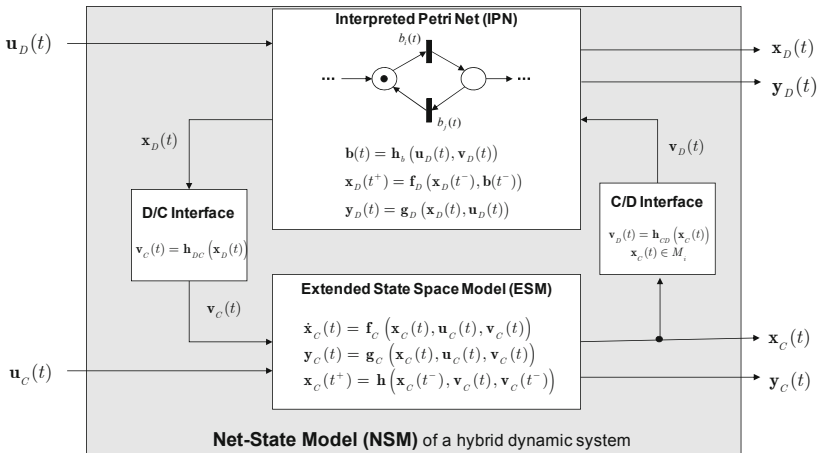


Fig. 2.63. General net-state model (NSM)

Properties of net-state models

- *Discrete-event subsystem*

By using interpreted PETRI nets (IPNs), a great diversity of models is achievable, as both purely sequential and parallel processes can be modeled (state machines vs. synchronization graphs); interpretation is conventional: switching conditions at the transitions represent inputs to the DE subsystem, and output places represent outputs of the DE subsystem.

- *Hybrid model state*

$$\mathbf{x}_H(t) = \left(\mathbf{x}_D(t), \mathbf{x}_C(t) \right)^T.$$

- *Discrete model state*

Describes the current marking vector of the interpreted PETRI net (IPN token distribution).

- Dynamics of the discrete model variables $\mathbf{x}_D(t)$, $\mathbf{u}_D(t)$

Piecewise-constant, until the occurrence of a switching event at time \bar{t} .

- *D/C interface, injection*

Dynamics of the continuous model variable $\mathbf{v}_C(t)$; piecewise-constant, until the occurrence of a switching event at time \bar{t} .

- *Firing requirement for transition j*

If the transition j is activated according to the marking, and the Boolean switching expression $b_j(t)$ is true, then transition j fires immediately; if several parallel transitions are ready to fire, then all fire simultaneously.

- Reinitialization of the continuous model state \mathbf{x}_C

Upon an external/autonomous jump in the state, the continuous state vector $\mathbf{x}_C(t^+)$ is updated according to the discontinuity equation.

- *C/D interface, quantization*

Threshold violation by the continuous model state \mathbf{x}_C . When the continuous state vector $\mathbf{x}_C(t)$ enters the set Φ_i , the corresponding internal model variable $v_{D_i}(t)$ is set to 1 (i.e. $v_{D_i}(t) = 0$ for $\mathbf{x}_C(t) \notin M_i$).

Modeling the discrete-event subsystem A detailed description of discrete-event modeling paradigms is beyond the scope of this book. For the reader unacquainted with this area of modeling, the very readable monograph (Litz 2005) is recommended, where, in particular, the *interpreted PETRI net* employed here is described in detail.

PETRI nets are not required to describe net-state models. However, in comparison to sequential *automaton models*, they do appear more suitable, since parallel PETRI nets produce a clearer DE system structure when several discrete system variables are present. Note, however, that for a comprehensive analysis of the dynamics of a DE subsystem, the entire *reachability graph* of the PETRI net must be examined. This graph is itself a sequential automaton (with a potentially very large number of states). In this respect, there is no difference in the amount of effort required for these two representations. The use of one over the other is rather a matter of taste for the designer.

Computational implementation with STATECHARTS The implementation of net-state models for computation requires a common platform for continuous-time and discrete-event models. For both automaton models and PETRI net models, there exists only a limited number of publicly-available simulation platforms which allow for the incorporation of continuous-time models. A practicable solution is offered by the *STATECHARTS* modeling paradigm (Harel 1987), with which hierarchical and parallel automaton structures can be very efficiently modeled. However, when representing parallel, structurally limited PETRI net models using STATECHARTS, particular transformation limitations must be observed (Schnabel et al. 1999).

The attraction of STATECHARTS for discrete-event modeling in hybrid systems is not least due to the fact that this modeling paradigm has been successfully implemented in commercial simulation tools³⁵ so that easy-to-use computational platforms for efficient simulation of hybrid systems are available.

Example 2.12 *Single-joint manipulator with collision.*

System configuration An elastically suspended single-joint manipulator (massless arm of length l , end effector mass m , spring constant k , motor torque τ , equilibrium position at $\theta = 0$) moves horizontally on a level surface with position-dependent, velocity-proportional coefficients of friction μ_1, μ_2 (Fig. 2.64). Along the x-axis shown, collisions with a hard boundary are possible (assuming elastic collisions). For this system, a *net-state model* is to be created.

³⁵ E.g. STATEFLOW, a component of MATLAB / SIMULINK.

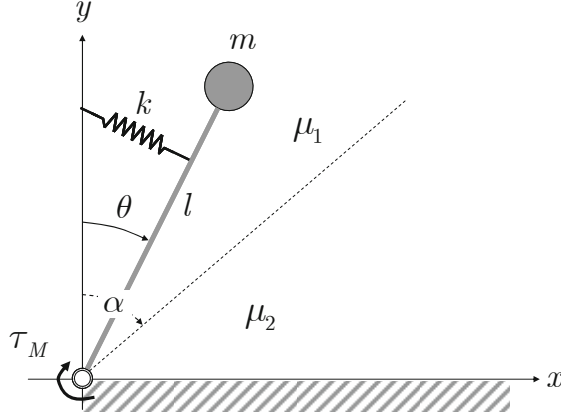


Fig. 2.64. Single-joint manipulator with collision

Model creation In this system, two *hybrid phenomena* appear:

- *autonomous switching*: due to the position-dependent friction μ_1, μ_2 , the vector field of the equations of motion changes when the threshold on the arm angle (state) $\theta > \alpha$ is exceeded,
- *autonomous state jump*: in the case of contact at $\theta = 90^\circ$ under the assumption of an elastic collision, the state θ remains continuous; however, there is a jump in the state $\dot{\theta}$: the angular velocity changes sign and its magnitude decreases according to the *coefficient of restitution* ρ , $0 \leq \rho \leq 1$.

In the state plane $\theta, \dot{\theta}$, three disjoint sets of states $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ can be defined. These sets are in turn assigned (binary) variables $v_{D_i} \in \{0, 1\}$, $i = 1, 2, 3$ in the C/D interface.

$$v_{D_i} \begin{cases} 1, & \mathbf{x}_c \in \mathcal{M}_i \\ 0, & \mathbf{x}_c \notin \mathcal{M}_i \end{cases}, i = 1, 2, 3 \quad \begin{array}{ll} v_{D1} : & \mathcal{M}_1 = \{\theta | 0 \leq \theta < \alpha\} \\ v_{D2} : & \mathcal{M}_2 = \{\theta | \alpha \leq \theta < 90^\circ\} \\ v_{D3} : & \mathcal{M}_3 = \{\theta | \theta \geq 90^\circ\} \end{array}$$

The alternation of the trajectory $\mathbf{x}_c(t)$ between the subsets can be described via a simple, structurally constrained PETRI net of the *state machine* type (Fig. 2.65). The discrete states $x_{D_i} \in \{0, 1\}$, $i = 1, 2, 3$ describe the current subset occupied by $\mathbf{x}_c(t)$.

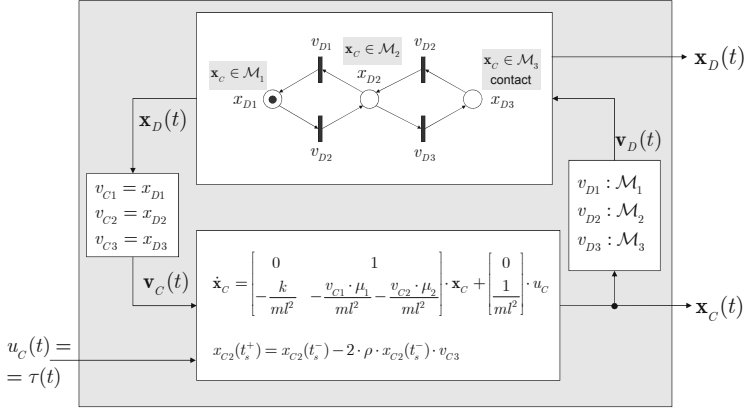


Fig. 2.65. Net-state model for single-joint manipulator with collision

In the D/C interface, the discrete states are used to generate continuous-time switching variables $v_{Ci}(t)$, for which

$$v_{Ci}(t) = \begin{cases} 0, & x_{Di} = 0 \\ 1, & x_{Di} = 1 \end{cases}$$

holds.

Using the switching variables $v_{Ci}(t)$, the vector field in the state model can then be modified in a timely manner or the state discontinuity can be modeled. The complete net-state model is presented in Fig. 2.65. ■

2.6 Linear System Models

Linear dynamic analysis As shown in the previous sections, the class of mechatronic systems with lumped elements considered in this book can be described in a general form by a system of *nonlinear* differential-algebraic equations. The methods used throughout this book for dynamic analysis and controller design are, however, based on *linear time invariant (LTI)* models. This section thus presents a primer on *local linearization* (or *JACOBI linearization*) of nonlinear dynamic systems. This type of linearization is well known in many technical disciplines and is discussed here to complete the methodological toolbox.

Local linearization As a precondition for local linearization, the dynamics of the nonlinear system in the neighborhood of certain solutions (trajectories) are examined. Even with a completely known (nonlinear) model, the result is always an *approximation* of the actual system behavior in the vicinity of a certain trajectory and it is no longer representative if “larger” deviations from the reference solution are considered. Results based on such linearized models should thus always be evaluated with all due caution, and the observance of “sufficiently small deviations” should be carefully verified in every particular case.

Exact linearization Attention of the interested reader is directed to one extended form of linearization, which has in the last two decades significantly extended control system theory in particular. In *exact linearization*, *input-output linearization*, or *feedback linearization*—e.g. (Isidori 2006)—a nonlinear transformation (e.g. feedback) is inserted in the system to generate a “linear” system (i.e. the system is not simply “linearized”, but in fact “linear”) to which linear control laws can then be applied (for examples in robot controllers see (Siciliano et al. 2009)). In such cases, then, given exact knowledge of the original nonlinear system, an *exactly linear system* is generated. However, since this type of linear model is of only limited use for general analysis of system dynamics, it is *local linearization* which is pursued below.

2.6.1 Local linearization of nonlinear state space models

System description Consider the following nonlinear state space model as a special case of a DAE system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad (2.60)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}), \quad (2.61)$$

$$\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^r \rightarrow \mathbb{R}^n, \mathbf{g} : \mathbb{R}^n \times \mathbb{R}^r \rightarrow \mathbb{R}^m.$$

For a given input $\mathbf{u}_*(t)$ and the resulting solution $\mathbf{x}_*(t)$ of the differential equation (2.60), it holds by definition that

$$\begin{aligned} \dot{\mathbf{x}}_* &= \mathbf{f}(\mathbf{x}_*, \mathbf{u}_*) \\ \mathbf{y}_* &= \mathbf{g}(\mathbf{x}_*, \mathbf{u}_*) \end{aligned} \quad (2.62)$$

Now, considering (arbitrary) deviations $\mathbf{x}(t) = \mathbf{x}_*(t) + \Delta\mathbf{x}(t)$ from the *reference solution* resulting from input deviations $\mathbf{u}(t) = \mathbf{u}_*(t) + \Delta\mathbf{u}(t)$, it then follows from Eqs. (2.60), (2.61) that

$$\begin{aligned}\dot{\mathbf{x}}_* + \Delta\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}_* + \Delta\mathbf{x}, \mathbf{u}_* + \Delta\mathbf{u}) \\ \mathbf{y}_* + \Delta\mathbf{y} &= \mathbf{g}(\mathbf{x}_* + \Delta\mathbf{x}, \mathbf{u}_* + \Delta\mathbf{u})\end{aligned}\tag{2.63}$$

where the substitution $\mathbf{y} = \mathbf{y}_* + \Delta\mathbf{y}$ was applied to the output vector.

Linear approximation Replacing the vector fields \mathbf{f} and \mathbf{g} in Eq. (2.63) by their *linear approximations* (Taylor expansion) and considering *small deviations* $\Delta\mathbf{x}$, $\Delta\mathbf{u}$, higher-order terms can be neglected, so that

$$\begin{aligned}\dot{\mathbf{x}}_* + \Delta\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}_*, \mathbf{u}_*) + \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \\ \mathbf{y}_* + \Delta\mathbf{y} &= \mathbf{g}(\mathbf{x}_*, \mathbf{u}_*) + \mathbf{C}\Delta\mathbf{x} + \mathbf{D}\Delta\mathbf{u}\end{aligned}\tag{2.64}$$

where

$$\begin{aligned}A_{ij} &= \frac{\partial f_i}{\partial x_j}(x_*, u_*), \quad \mathbf{A} \in \mathbb{R}^{n \times n} \quad C_{ij} = \frac{\partial g_i}{\partial x_j}(x_*, u_*), \quad \mathbf{C} \in \mathbb{R}^{m \times n} \\ B_{ij} &= \frac{\partial f_i}{\partial u_j}(x_*, u_*), \quad \mathbf{B} \in \mathbb{R}^{n \times r} \quad D_{ij} = \frac{\partial g_i}{\partial u_j}(x_*, u_*), \quad \mathbf{D} \in \mathbb{R}^{m \times r}.\end{aligned}\tag{2.65}$$

Due to (2.62), Eq. (2.64) can be simplified to give the standard representation of a *linear state space model*

$$\begin{aligned}\Delta\dot{\mathbf{x}} &= \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \\ \Delta\mathbf{y} &= \mathbf{C}\Delta\mathbf{x} + \mathbf{D}\Delta\mathbf{u}.\end{aligned}\tag{2.66}$$

The system matrices (2.65) are the well-known *JACOBIAN matrices* of the vector fields \mathbf{f} and \mathbf{g} , e.g.

$$\mathbf{A} := \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_*, \mathbf{u}_*) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}_*, \mathbf{u}_*) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}_*, \mathbf{u}_*) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}_*, \mathbf{u}_*) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}_*, \mathbf{u}_*) \end{bmatrix}.\tag{2.67}$$

Thus, the state space system (2.66) is called the *local linearization* or *JACOBI linearization* of (2.60), (2.61).

Eq. (2.67) predicts that at every time \tilde{t} , the partial derivatives of the respective vector fields should be calculated and subsequently the variables x_i, u_j should be replaced by the corresponding values $x_{i*}(\tilde{t}), u_{j*}(\tilde{t})$ from the reference solution. This results in the following two characteristic cases.

Linearization about an equilibrium For constant inputs $\mathbf{u}_*(t) = \mathbf{u}_{*0} = \text{const.}$, the *equilibria* $\mathbf{x}_{*0} = \text{const.}$ of the system in Eq. (2.60) are computed from the algebraic system of equations

$$\mathbf{0} = \mathbf{f}(\mathbf{x}_{*0}, \mathbf{u}_{*0}). \quad (2.68)$$

With the (constant) solutions $\mathbf{u}_{*0}, \mathbf{x}_{*0}$ of Eq. (2.68), *constant* system matrices (Eq. (2.65)) follow, and thus a linear *time-invariant* (LTI) state space model

$$\begin{aligned} \Delta \dot{\mathbf{x}} &= \mathbf{A}_0 \Delta \mathbf{x} + \mathbf{B}_0 \Delta \mathbf{u}, \\ \Delta \mathbf{y} &= \mathbf{C}_0 \Delta \mathbf{x} + \mathbf{D}_0 \Delta \mathbf{u}. \end{aligned} \quad (2.69)$$

The tuple $(\mathbf{x}_{*0}, \mathbf{u}_{*0})$ is also called the *operating point* of the system so that Eq. (2.69) is referred to as a *linearization* about the operating point $(\mathbf{x}_{*0}, \mathbf{u}_{*0})$.

Linearization about a trajectory For general, non-constant inputs $\mathbf{u}_*(t)$, the time-varying *solution trajectory* $\mathbf{x}_*(t)$ must also be incorporated into the computation of the system matrices (2.65)³⁶, resulting in time-varying system matrices and, overall, a *linear time-varying* (LTV) *state space system*

$$\begin{aligned} \Delta \dot{\mathbf{x}} &= \mathbf{A}(t) \cdot \Delta \mathbf{x} + \mathbf{B}(t) \cdot \Delta \mathbf{u}, \\ \Delta \mathbf{y} &= \mathbf{C}(t) \cdot \Delta \mathbf{x} + \mathbf{D}(t) \cdot \Delta \mathbf{u}. \end{aligned} \quad (2.70)$$

As usual, the eigenvalues of the system matrix \mathbf{A} determine the dynamics and stability of the locally linearized system.

³⁶ This solution trajectory comes from the solution of the nonlinear system model (2.60). Only in exceptional cases is this analytically possible. For simulation experiments, the value for $\mathbf{x}_*(t)$ is taken to be the approximate value from the numerical solution. For stochastic dynamic analysis (Ch. 11), pre-calculated (numerical) nominal trajectories can be used.

Example 2.13 *Particle motion with viscous friction.*

System configuration Consider a point mass m moving weightlessly in a plane subject to viscous friction (coefficient of friction μ) and an applied force \vec{F} . The current distance r from the point mass to the origin can be measured with a suitable measuring device (*range measurements*) (Fig. 2.66).

Find a linear state space model given a reference trajectory $\vec{s}_*(t)$, $\vec{v}_*(t)$, $\vec{F}_*(t)$.

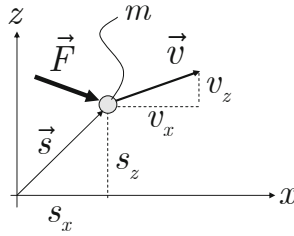


Fig. 2.66. Linearization of a nonlinear state space model

Model creation

- Equation of motion for the point mass:

$$m\dot{\vec{v}} = \vec{F} - \mu |\vec{v}|^2 \frac{\vec{v}}{|\vec{v}|}$$

- Measurement equation:

$$r = |\vec{s}|$$

- Nonlinear state space model:

$$x_1 := s_x, x_2 := v_x, x_3 := s_z, x_4 := v_z$$

$$u_1 := F_x, u_2 := F_z, y := r$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{\mu}{m} x_2 \sqrt{x_2^2 + x_4^2} + \frac{1}{m} u_1$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = -\frac{\mu}{m} x_4 \sqrt{x_2^2 + x_4^2} + \frac{1}{m} u_2$$

$$y = \sqrt{x_1^2 + x_3^2}$$

- Linearized state space model (time-varying due to $x_{i*}(t)$):

$$\begin{aligned}
 \Delta \dot{x}_1 &= \Delta x_2 \\
 \Delta \dot{x}_2 &= -\frac{\mu}{m} \frac{2x_{2*}^2 + x_{4*}^2}{\sqrt{x_{2*}^2 + x_{4*}^2}} \Delta x_2 - \frac{\mu}{m} \frac{2x_{2*}x_{4*}}{\sqrt{x_{2*}^2 + x_{4*}^2}} \Delta x_4 + \frac{1}{m} \Delta u_1 \\
 \Delta \dot{x}_3 &= \Delta x_4 \\
 \Delta \dot{x}_4 &= -\frac{\mu}{m} \frac{2x_{2*}x_{4*}}{\sqrt{x_{2*}^2 + x_{4*}^2}} \Delta x_2 - \frac{\mu}{m} \frac{2x_{4*}^2 + x_{2*}^2}{\sqrt{x_{2*}^2 + x_{4*}^2}} \Delta x_4 + \frac{1}{m} \Delta u_2 \\
 \Delta y &= \frac{2x_{1*}}{\sqrt{x_{1*}^2 + x_{3*}^2}} \Delta x_1 + \frac{2x_{3*}}{\sqrt{x_{1*}^2 + x_{3*}^2}} \Delta x_3
 \end{aligned}$$

2.6.2 Local linearization of nonlinear DAE systems

Consider the *semi-explicit* time-invariant DAE system

$$\begin{aligned}
 \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}), \\
 \mathbf{0} &= \mathbf{g}(\mathbf{x}, \mathbf{z}),
 \end{aligned} \tag{2.71}$$

$$\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \rightarrow \mathbb{R}^n, \quad \mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^m.$$

Generalizing the results in Sec. 2.6.1 for small deviations $\Delta \mathbf{x}$, $\Delta \mathbf{z}$, $\Delta \mathbf{u}$ from a reference solution \mathbf{x}_* , \mathbf{z}_* , \mathbf{u}_* (i.e. $\mathbf{x} = \mathbf{x}_* + \Delta \mathbf{x}$, $\mathbf{z} = \mathbf{z}_* + \Delta \mathbf{z}$, $\mathbf{u} = \mathbf{u}_* + \Delta \mathbf{u}$) gives the *locally linearized DAE system*

$$\begin{aligned}
 \Delta \dot{\mathbf{x}} &= \mathbf{A} \Delta \mathbf{x} + \mathbf{B} \Delta \mathbf{u} + \mathbf{F} \Delta \mathbf{z} \\
 \mathbf{0} &= \mathbf{G} \Delta \mathbf{x} + \mathbf{H} \Delta \mathbf{z}
 \end{aligned}$$

with system matrices

$$\begin{aligned}
 A_{ij} &= \frac{\partial f_i}{\partial x_j}(x_*, z_*, u_*), & \mathbf{A} &\in \mathbb{R}^{n \times n} \\
 B_{ij} &= \frac{\partial f_i}{\partial u_j}(x_*, z_*, u_*), & \mathbf{B} &\in \mathbb{R}^{n \times r}, \\
 F_{ij} &= \frac{\partial f_i}{\partial z_j}(x_*, z_*, u_*), & \mathbf{F} &\in \mathbb{R}^{n \times m} \\
 G_{ij} &= \frac{\partial g_i}{\partial x_j}(x_*, z_*), & \mathbf{G} &\in \mathbb{R}^{m \times n} \\
 H_{ij} &= \frac{\partial g_i}{\partial z_j}(x_*, z_*), & \mathbf{H} &\in \mathbb{R}^{m \times m}.
 \end{aligned}$$

Note that depending on the index of the DAE system, the matrices \mathbf{G} and \mathbf{H} can also be singular.

2.6.3 LTI systems: transfer function, frequency response

Frequency domain representation Starting with mathematical models of a mechatronic system in DAE form (2.71) or state space form (2.60), (2.61), a coupled dynamic analysis is particularly clear if undertaken with *linearized models* in the *frequency domain*. The most important tool presented in this book for evaluating system dynamics is thus the use of *transfer functions* of the system elements under consideration. As usual, the linear (LTI) state space model (2.69) can be used to compute the transfer function via the *LAPLACE transform* (Ogata 1992).

As an important, representative case, the rest of this section considers the *single-input single-output (SISO)* LTI system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A} \cdot \mathbf{x} + \mathbf{b} \cdot u & \mathbf{x}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{n \times n} \\ y &= \mathbf{c}^T \cdot \mathbf{x} + d \cdot u & d, u, y \in \mathbb{R} \end{aligned} \quad (2.72)$$

Transfer function

Procedure By applying the LAPLACE transform to Eq. (2.72) and with straightforward manipulation of the resulting algebraic equations in the complex variable s , the *transfer function* $G(s)$ is obtained:

$$G(s) = \frac{Y(s)}{U(s)} = \mathbf{c}^T (s\mathbf{E} - \mathbf{A})^{-1} \mathbf{b} + d, \quad (2.73)$$

where $U(s)$ and $Y(s)$ represent the LAPLACE transforms of the input $u(t)$ and the output $y(t)$, respectively (Fig. 2.67, transform direction from left to right).

The *transfer function* $G(s)$ is a rational function of the form

$$\begin{aligned} G(s) &= \frac{Y(s)}{U(s)} = \frac{b_m \cdot s^m + \cdots + b_1 \cdot s + b_0}{s^n + a_{n-1} \cdot s^{n-1} + \cdots + a_1 \cdot s + a_0} \\ &= \frac{N_G(s)}{D_G(s)} = b_m \frac{\prod_{j=1}^m (s - n_j)}{\prod_{i=1}^n (s - p_i)}, \quad m \leq n, \end{aligned} \quad (2.74)$$

with *numerator polynomial* $N_G(s)$, *denominator polynomial* $D_G(s)$, *zeros* n_i , and *poles* p_j . If the degree m of the numerator equals the degree n of the denominator, there is a direct feedthrough of the input u to the output y —equivalent to $d \neq 0$ in Eq. (2.72).

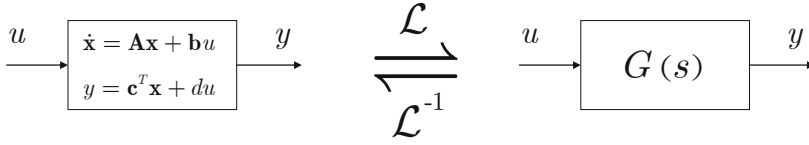


Fig. 2.67. LTI system: transfer function (single-input single-output (SISO))

Poles and zeroes The dynamic behavior, and in particular the stability, of the system is determined by the *poles* of the transfer function $G(s)$, which as a rule are identical to the *eigenvalues* of the system matrix \mathbf{A} ³⁷. The *zeroes* of the transfer function depend on the *input vector* \mathbf{b} and the *output vector* \mathbf{c} of the state space model in a complicated manner which can be expressed analytically only to a limited extent³⁸.

Inverse transformation to state space model Given a transfer function $G(s)$ (2.74), an infinite number of state space representations is possible (Fig. 2.67, transform direction from right to left). One possible form is the so-called *control canonical form* (Ogata 2010): (for $m = n$)

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \cdot u, \quad (2.75)$$

$$y = \left((b_0 - b_n a_0) \quad (b_1 - b_n a_1) \quad \cdots \quad (b_{n-1} - b_n a_{n-1}) \right) \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + b_n u.$$

³⁷ This is *never* the case when the state space model (2.72) is *uncontrollable* or *unobservable*. In these cases, certain pole and zero terms cancel in Eq. (2.74), i.e. the number of poles in the transfer function is then *smaller* than the number of eigenvalues of \mathbf{A} .

³⁸ If the state space system (2.73) describes the “plant” in a mechatronic system—e.g. a mechanical structure (multibody system)—then the zeroes of the transfer function between the actuator and sensor depend significantly on the locations of the actuator and sensor in the multibody system. This fundamental behavior is discussed in detail in Ch. 4.

The transformation *state space model* \leftrightarrow *transfer function* is thus only valid in one direction (left \rightarrow right in Fig. 2.67). Thus, when back-converting a transfer function into a state space model (right \rightarrow left in Fig. 2.67), the choice of state variables can result in more or less suitable state representations for numerical integration (see Ch. 3).

Frequency response

The transfer function $G(s)$ is used to derive the *frequency response* $G(j\omega)$ in the usual manner (Ogata 2010):

$$G(j\omega) = G(s)|_{s=j\omega} = |G(j\omega)| \cdot e^{j \arg G(j\omega)}. \quad (2.76)$$

The frequency response (2.76) represents a *non-parametric* description of the LTI system (2.72) in the form of the frequency-dependent *amplitude response* $|G(j\omega)|$ and *phase response* $\arg G(j\omega)$, and can also be very efficiently determined *experimentally* (see Sec. 2.7).

Model properties Such a non-parametric description is particularly useful when the LTI system exhibits a high system order and contains transcendental components (delay terms in the form of exponential functions in $j\omega$). In such cases, greater model complexity is simply reflected in a greater degree of detail in $G(j\omega)$. As will be shown below, the dynamics of *multibody systems* and the influences of significant physical parameters are, in particular, very clearly revealed in the frequency response.

Graphical representation The ordinary representation of the frequency response in the form of a *frequency response locus* $G(j\omega)$ in the complex plane proves to be rather impractical when working with more complex systems. For this reason, this book prefers the well-known representation in the form of *logarithmic frequency curves* or *BODE diagrams* (Ogata 2010). This representation depicts the amplitude and phase curves as a function of frequency in separate plots. By using a semi-logarithmic representation, construction can be simplified, so that even hand sketches are possible, which can then be used for quick checks (in the sense of “verification”, see Sec. 2.1) on computer-generated results.

For more specialized applications in controller design, an additional representation of the frequency response in the phase-amplitude plane—the so-called NICHOLS plot—is of particular use. Its use is explained in detail in the context of robust controller design using the NYQUIST criterion in the *intersection form* in Ch. 10.

Controller design In addition to advantageous properties for modeling, the frequency response approach also offers excellent opportunities for *controller design*. There exists an extensive methodological toolbox for control based on the frequency response which can answer questions of closed-loop *stability* (NYQUIST criterion) and synthesize *robust control algorithms*. Practical methodological approaches for both of these areas are presented in detail for mechatronic systems in Ch. 10.

Example 2.14 *Two-mass oscillator with force excitation.*

System configuration and model creation Any of the methods for physical modeling shown in Sec. 2.3 leads to the following *equations of motion* for the coupled two-mass system in Fig. 2.68 (with assumed negligible damping):

$$\begin{aligned} m\ddot{y}_1 + 2ky_1 - ky_2 &= F, \\ m\ddot{y}_2 - ky_1 + 2ky_2 &= 0. \end{aligned} \quad (2.77)$$

Using the state definition $x_1 := y_1$, $x_2 := y_2$, $x_3 := \dot{y}_1$, $x_4 := \dot{y}_2$ and $u := F$, the *state space model* is

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{2k}{m} & \frac{k}{m} & 0 & 0 \\ \frac{k}{m} & -\frac{2k}{m} & 0 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m} \\ 0 \end{pmatrix} u, \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \mathbf{x}. \end{aligned} \quad (2.78)$$

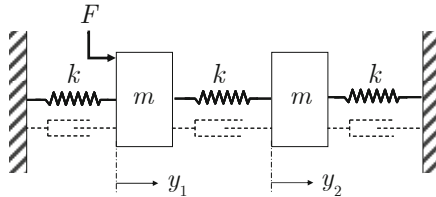


Fig. 2.68. Two-mass oscillator with very low damping

Via a LAPLACE transform of Eq. (2.77), or using relation (2.73), Eq. (2.78) yields the *transfer function* between the excitation force F and the mass displacement y_I :

$$G(s) = \frac{Y_I(s)}{F(s)} = V \frac{1 + \frac{s^2}{\omega_z^2}}{\left(1 + \frac{s^2}{\omega_{p1}^2}\right) \left(1 + \frac{s^2}{\omega_{p2}^2}\right)}, \quad (2.79)$$

$$\omega_{p1} = \sqrt{\frac{k}{m}}, \quad \omega_{p2} = \sqrt{\frac{3k}{m}}, \quad \omega_z = \sqrt{\frac{2k}{m}}, \quad V = \frac{2}{3k}.$$

The values ω_{p1}, ω_{p2} are termed the *natural frequencies* of the multi-body system. For a more detailed physical interpretation of these natural frequencies, as well as the numerator term containing ω_z (antiresonant frequency), refer to Ch. 4.

The *amplitude response* component of the frequency response $G(j\omega)$ is shown in Fig. 2.69 in the form of a BODE diagram (in the graphical representation, a finite, very small damping is assumed). The natural frequencies and the anti-resonant frequency are clearly discernable.

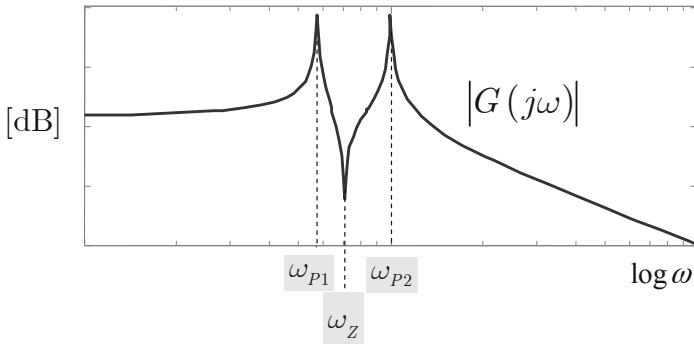


Fig. 2.69. BODE diagram (amplitude response) of the two-mass oscillator

Elastic structures: harmonic oscillators Example 2.14 exemplifies typical dynamics of mechatronic systems. In the task “purposeful motion of mass-bearing bodies”, *elastically-connected multibody systems (MBS)* are, as a rule, involved. Either the mechanical structure is elastically joined, or force application happens not via rigid, but rather elastic structures. These particular dynamics—in the form of *resonant natural frequencies* (eigenmodes) of the elastic multibody systems—are significantly easier to recognize in the transfer function (*complex conjugate pairs* of poles) than in the state space model. The natural frequencies are particularly recognizable in the amplitude responses of the BODE diagram (as spikes). If the possibility of designing controllers based on the frequency response of the *open* loop is taken into account, BODE diagrams of the (open) loop consisting of the *actuator, mechanical structure (MBS), and sensor* represent central analysis and design aids.

Notation For a concise and expressive representation of transfer functions of mechatronic systems, later chapters will, wherever useful, employ the following *shorthand* for linear and quadratic factors appearing there:

$$\boxed{\begin{aligned} [\omega_i] &:= 1 + \frac{s}{\omega_i} \\ \{d_i, \omega_i\} &:= 1 + 2d_i \frac{s}{\omega_i} + \frac{s^2}{\omega_i^2} \\ \{\omega_i\} &:= 1 + \frac{s^2}{\omega_i^2} \end{aligned}} \quad (2.80)$$

In particular, the representation in Eq. (2.80) enables the concise specification of eigenmodes with *natural frequency* ω_i and, if applicable, damping d_i .

The transfer function (2.79) in Example 2.14 would thus be represented as follows

$$G(s) = K \frac{\{\omega_z\}}{\{\omega_{P1}\}\{\omega_{P2}\}}.$$

2.7 Experimental Determination of the Frequency Response

2.7.1 General considerations

As a complement to the creation of system models by theoretical and analytical means, experimental modeling methods and procedures present significant advantages. For models derived from physical experiments, the following distinctions regarding the form of model are made:

- *parametric* models, e.g. transfer functions with a finite number of parameters → *parameter estimation*,
- *non-parametric* models, e.g. impulse response, frequency response → *signal-oriented estimation*.

Parametric models Parameter estimation is well-suited to determining models of sufficiently *low* order, and possibly even the direct estimation of physical parameters, while also accounting for the most *salient* static and dynamic system properties. However, mechatronic systems with pronounced multibody system properties (many resonant frequencies) and complex dynamics generally require a very high order model.

Non-parametric models: frequency response As is shown in following chapters, properties of complex mechatronic systems can be very efficiently and comprehensively described with transfer functions or *frequency responses*. Using the procedures presented in Ch. 10 for *robust controller design* based on frequency responses, it is in principle possible to carry out controller design without knowledge of the physical system parameters and using only knowledge of the complete dynamic response between actuator and sensor (including delays!). Since, as is shown below, the measurement of the frequency response can be carried out very efficiently, experimentally-determined *non-parametric* models in the form of frequency responses offer themselves as ideal complements to theoretically-based models for mechatronic systems.

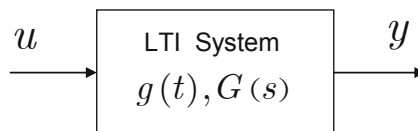


Fig. 2.70. Linear time-invariant system

2.7.2 Methodological approach

System configuration To facilitate further discussion, consider the linear time-invariant system shown in Fig. 2.70. By definition, the frequency response

$$G(j\omega) = \frac{Y(j\omega)}{U(j\omega)}$$

represents the relationship between the LAPLACE or FOURIER transforms of the input and output of the systems. A convenient arrangement for the experimental determination of $G(j\omega)$ for an example mechatronic system is shown in Fig. 2.71.

The following methodological approaches have proven valuable in practice, and are correspondingly supported with industrial devices or are easily implemented with basic signal processing (e.g. in MATLAB).

Harmonic excitation

Signal generator:	$u(t) = U_0 \sin \omega t, \omega \in [\omega_{\min}, \omega_{\max}]$
Signal evaluation:	after transients have dissipated, the amplitude ratio $ Y(j\omega) / U(j\omega) $ and the phase offset $\arg Y(j\omega) - \arg U(j\omega)$ are determined
Advantage:	simple signal processing
Disadvantage:	wait time for transients; large amplitudes when MBS eigenmodes are excited → danger of mechanical failure!

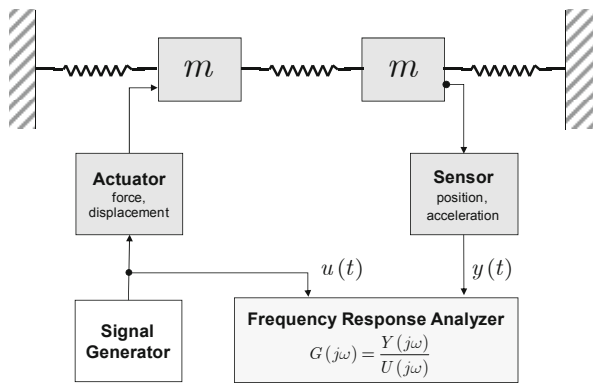


Fig. 2.71. Configuration for measurement of frequency response

Impulse excitation

Signal generator:	$u(t) = U_0 \cdot \delta(t)$
Signal evaluation:	The impulse response $g(t)$ is measured directly, i.e. $Y(s) = L\{g(t)\} \cdot L\{u(t)\} = G(s) \cdot U_0$, $G(j\omega)$ can then be easily computed via FFT
Advantage:	small excitation energy at MBS resonant frequencies
Disadvantage:	complicated signal processing (FFT); only direct force excitation of the MBS is possible

Noise excitation

Signal generator:	$u(t) = \text{random signal}$
Signal evaluation:	The impulse response $g(t)$ is measured directly, i.e. $Y(s) = L\{g(t)\} \cdot L\{u(t)\} = G(s) \cdot U_0$; $G(j\omega)$ can be easily computed from the series $(g(kT_A))$ via FFT
Advantage:	small excitation energy at MBS resonant frequencies; very robust to signal disturbances; direct force and dis- placement excitation of the MBS possible (given suit- able actuators)
Disadvantage:	complicated signal processing (FFT)

2.7.3 Frequency responses measurement via noise excitation

Introductory remarks Frequency response measurement using noise excitation has proven itself particularly suitable in practical situations. In particular, the use of the correlation function allows the effects of random signal disturbances to be very efficiently canceled out. This section sketches significant theoretical concepts which are easily implemented in a computational tool (e.g. MATLAB), a more rigorous discussion is given in Ch. 11. For a high-grade implementation, e.g. including windowing functions, refer to the applicable literature (Rabiner and Gold 1975).

Computational procedure Consider an LTI system according to Fig. 2.70 with

- $u(t)$... Realization of a (zero-mean) ergodic random process,
- $g(t)$... Impulse response of the LTI system.

The output $y(t)$ is then computed via the convolution integral

$$y(t) = g(t) * u(t) = \int_{-\infty}^{\infty} g(\tau) u(t - \tau) d\tau. \quad (2.81)$$

The FOURIER transform of a signal $x(t)$, $x \in \{u, y, g\}$ is, by definition,

$$X(j\omega) := \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt.$$

For the LTI system in Fig. 2.70, it holds that

$$Y(j\omega) = G(j\omega) U(j\omega).$$

The *cross-correlation* $r_{uy}(\tau)$ of the signals $y(t)$, $u(t)$ is defined as

$$r_{uy}(\tau) := \lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} u(t) y(t + \tau) dt. \quad (2.82)$$

Replacing $y(t)$ in Eq. (2.82) by Eq. (2.81),

$$r_{uy}(\tau) := \lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} u(t) \left[\int_{-\infty}^{\infty} g(\lambda) u(t + \tau - \lambda) d\lambda \right] dt,$$

and changing the order of integration

$$r_{uy}(\tau) = \int_{-\infty}^{\infty} g(\lambda) \left[\lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} u(t) u(t + \tau - \lambda) dt \right] d\lambda. \quad (2.83)$$

For the autocorrelation $r_{uu}(\tau)$ of the input $u(t)$ (see also Eq. (2.82)), a similar procedure gives

$$r_{uu}(\tau - \lambda) = \lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} u(t) u(t + \tau - \lambda) dt. \quad (2.84)$$

From Eqs. (2.83) and (2.84), the convolution integral for the correlations is then

$$r_{uy}(\tau) = \int_{-\infty}^{\infty} g(\lambda) r_{uu}(\tau - \lambda) d\lambda = g(\tau) * r_{uu}(\tau). \quad (2.85)$$

The FOURIER transform of Eq. (2.85) gives

$$S_{uy}(j\omega) = G(j\omega) S_{uu}(j\omega), \quad (2.86)$$

where

$$S_{uy}(j\omega) = \int_{-\infty}^{\infty} r_{uy}(\tau) e^{-j\omega\tau} d\tau \quad \text{cross spectral density}, \quad (2.87)$$

$$S_{uu}(j\omega) = \int_{-\infty}^{\infty} r_{uu}(\tau) e^{-j\omega\tau} d\tau \quad \text{power spectral density}. \quad (2.88)$$

The frequency response $G(j\omega)$ can thus be determined via the power spectral densities (2.87) and (2.88), along with Eq. (2.86), as

$$G(j\omega) = \frac{S_{uy}(j\omega)}{S_{uu}(j\omega)}. \quad (2.89)$$

Measurement of the power spectral densities (2.87), (2.88) is easily achieved. For example, MATLAB makes available the following pre-made functions for processing signal sequences $(u(kT_a))$, $(y(kT_a))$:

- `psd` *power spectral density estimate*,
- `csd` *cross spectral density estimate*.

Measurement noise Under small signal disturbances (measurement noise), averaging even a few individual measurement sequences enables a representative estimate of the frequency responses (see Example 2.15). In particular, MBS eigenmodes and complex zeroes (the collocation problem), as well as phase lag due to delays and low-pass elements, can be elegantly ascertained with this method. Basically, the measured frequency response (following possible smoothing of “outliers”) can be employed directly in robust controller design using a computational tool (e.g. MATLAB), as presented in Ch. 10.

Example 2.15 *Experimental frequency response determination for two-mass oscillator with force excitation.*

System configuration For the multibody system shown in Fig. 2.68, noise excitation is to be used to experimentally determine the frequency response $G(j\omega) = Y_1(j\omega) / F(j\omega)$ and the mechanical parameters (m, k) .

Experiment frame To model the excitation with a wideband noise source, a second-order shaping filter with $\omega_n = 10$ rad/s, $d_n = 1$ is assumed (Fig. 2.72). Following the dissipation of transients, the measurements of u and y are sampled with a sampling time of $T_a = 0.1$ s and are stored in blocks of 1024 values. Ten frequency series calculated according to Eq. (2.89) are then averaged and output as the *estimated* frequency response.

Discussion The results of frequency response estimation are shown in Fig. 2.73 for noise-free and noisy measurements. Even assuming measurement disturbances, the resonant and anti-resonant frequencies are clearly visible. However, in the noisy case, due to the reduction in the signal-to-noise ratio, the measured response is no longer useful at high frequencies (there is only a small output signal due to the amplitude falloff of -40 dB/decade).

Using the measured resonant and antiresonant frequencies and the mathematical model in Eq. (2.79), the masses and spring constants of the system can be estimated (as a check for the interested reader: $m \approx 10$ kg, $k \approx 400$ N/m). Due to measurement uncertainty, the damping can be only relatively imprecisely estimated (overshoot at resonance, undershoot in the anti-resonance), as is usual in practice. In any case, an experimentally determined frequency response model as shown in Fig. 2.73b gives direct access to *robust controller design*, as discussed in more detail in Ch. 10.

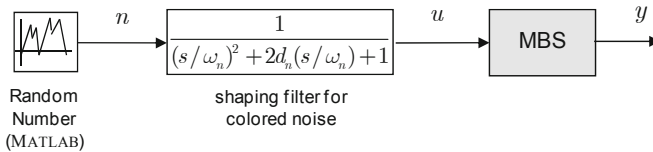


Fig. 2.72. Signal model for frequency response measurement

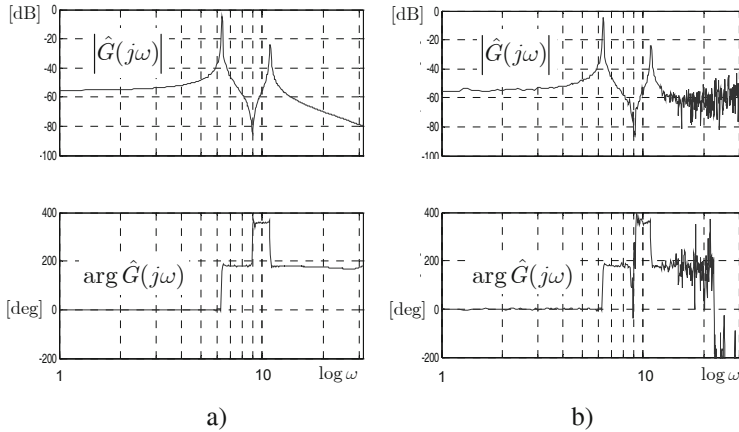


Fig. 2.73. Estimated frequency responses (BODE diagrams) for two-mass oscillator, averaged values from 10 measured frequency series:
a) without measurement noise, b) with measurement noise

Bibliography for Chapter 2

- Angermann, A., M. Beuschel, M. Rau and U. Wohlfarth (2005). *Matlab-Simulink-Stateflow. Grundlagen, Toolboxen, Beispiele*. München. Oldenbourg Wissenschaftsverlag.
- Brenan, K. E., S. L. Campbell and L. R. Petzold (1996). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM.
- Buss, M. (2002). *Methoden zur Regelung Hybrider Dynamischer Systeme*. Fortschritt-Berichte, VDI Reihe 8, Nr. 970.
- Cellier, F. E. (1991). *Continuous System Modeling*. Springer.
- Cellier, F. E. and H. Elmqvist (1993). "Automated formula manipulation supports object-oriented continuous-system modelling." *IEEE Control System Magazine* **13**(2): 28-38.
- Cellier, F. E. and E. Kofman (2006). *Continuous System Simulation*. Berlin. Springer.
- Cervera, J., A. J. van der Schaft and A. Banos (2007). "Interconnection of port-Hamiltonian systems and composition of Dirac structures." *Automatica* **43**(2): 212-225.
- Conrad, M., I. Fey and S. Sadeghipour (2005). "Systematic Model-Based Testing of Embedded Automotive Software " *Electronic Notes in Theoretical Computer Science* **111**: 13-26

- Damic, V. and J. Montgomery (2003). *Mechatronics by Bond Graphs*. Springer.
- Duindam, V., A. Macchelli, S. Stramigioli and H. Bruyninckx, Eds. (2009). *Modeling and Control of Complex Physical Systems - The Port-Hamiltonian Approach*, Springer.
- Engell, S., G. Frehse and E. Schnieder, Eds. (2002). *Modelling, analysis, and design of hybrid systems*. Lecture notes in control and information sciences, Springer.
- Fritzson, P. (2011). *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. John Wiley & Sons Inc.
- Fuchshumer, S., G. Grabmair, K. Schlacher and G. Keintzel (2003). "Automatisierungstechnik in der Mechatronik — zwei Beispiele aus der Stahlindustrie " *e&I* **120**(5): 164-171.
- Geitner, G. H. (2006). Power Flow Diagrams Using a Bond Graph Library under Simulink. *Proc. of 32nd Annual Conference on IEEE Industrial Electronics, IECON 2006-*. pp.5282-5288.
- Geitner, G. H. (2008). Bondgraphen-Modelle für ausgewählte mechatronische Anschauungsbeispiele. Persönliche Kommunikation, Elektrotechnisches Institut, Technische Universität Dresden.
- Goldstein, H., C. P. Poole and J. L. Safko (2001). *Classical Mechanics*. Addison Wesley.
- Harel, D. (1987). "Statecharts - A Visual Formalism for Complex Systems." *Science of Computer Programming* **8**: 231-274.
- Hatley, D. J. and I. A. Pirbhai (1987). *Strategies for Real-Time System Specification*. New York, NY. Dorset House.
- Hatley, D. J. and I. A. Pirbhai (1993). *Strategien für die Echtzeitprogrammierung*. München, Wien. Hanser.
- IEEE (1997). IEEE Trial-Use Recommended Practice for Distributed Interactive Simulation -Verification, Validation, and Accreditation. IEEE Std 1278.4-1997. I. C. Society.
- Isidori, A. (2006). *Nonlinear Control Systems*. Springer.
- Karnopp, D. C., D. L. Margolis and R. C. Rosenberg (2006). *System dynamics: modeling and simulation of mechatronic systems*. John Wiley & Sons, Inc.
- Koycheva, E. and K. Janschek (2007). Performance analysis of system models with UML and Generalized Nets. *EUROSIM 2007, 6th EUROSIM Congress on Modelling and Simulation*, Ljubljana, Slovenia
- Kugi, A. and K. Schlacher (2001). "Dissipativitäts- und passivitätsbasierte Regelung nichtlinearer mechatronischer Systeme." *e&i* **120**(1): 40-48.

- Kugi, A. and K. Schlacher (2002). "Analyse und Synthese nichtlinearer dissipativer Systeme: Ein Überblick (Teil 2)." *at - Automatisierungstechnik* **50**(3): 103-111.
- Lenk, A., R. G. Ballas, R. Werthschützky and G. Pfeifer (2011). *Electromechanical Systems in Microtechnology and Mechatronics*. Springer.
- Litz, L. (2005). *Grundlagen der Automatisierungstechnik*. Oldenbourg Verlag München Wien.
- Lunze, J. (2002). What Is a Hybrid System? *Modelling, Analysis, and Design of Hybrid Systems*. S. Engell, G. Frehse and E. Schnieder. Springer: 3-14.
- Maschke, B. M. and A. J. van der Schaft (1992). Port-controlled Hamiltonian systems: Modelling origins and system theoretic properties. *IFAC Symposium on Nonlinear Control Systems Design (NOLCOS) 1992*, Bordeaux, France. pp.359-365.
- Mattsson, S. E. and G. Söderlind (1993). "Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives." *SIAM Journal on Scientific Computing* **14**(677-692).
- Nenninger, G., M. Schnabel and V. Krebs (1999). "Modellierung, Simulation und Analyse hybrider dynamischer Systeme mit Netz-Zustands-Modellen." *at-Automatisierungstechnik* **47**(3): 118-126.
- Oestereich, B. (2006). *Analyse und Design mit der UML 2.1 - Objektorientierte Softwareentwicklung*. Oldenbourg Wissenschaftsverlag.
- Ogata, K. (1992). *System Dynamics*. Prentice Hall.
- Ogata, K. (2010). *Modern Control Engineering*. Prentice Hall.
- Ortega, R., A. J. van der Schaft, B. M. Maschke and G. Escobar (2002). "Interconnection and damping assignment passivity-based control of port-controlled Hamiltonian systems." *automatica* **38**: 585-596.
- Otter, M. (1999). "Objektorientierte Modellierung Physikalischer Systeme, Teil 4." *at-Automatisierungstechnik* **47**(4): A13-A16.
- Otter, M. and B. Bachmann (1999). "Objektorientierte Modellierung Physikalischer Systeme, Teil 5." *at-Automatisierungstechnik* **47**(5): A17-A20.
- Otter, M. and B. Bachmann (1999). "Objektorientierte Modellierung Physikalischer Systeme, Teil 6." *at-Automatisierungstechnik* **47**(6): A21-A24.
- Pantelides, C. C. (1988). "The consistent initialization of differential-algebraic systems." *SIAM Journal of Scientific and Statistical Computing* **9**: 213-231.
- Paynter, H. M. (1961). *Analysis and Design of Engineering Systems*. MIT Press, Cambridge, Mass.

- Rabiner, L. R. and B. Gold (1975). *Theory and Application of Digital Signal Processing*. Englewood Cliffs, New Jersey. Prentice Hall.
- Rau, A. (2002). *Model-Based Development of Embedded Automotive Control Systems*. Dissertation, Universität Tübingen.
- Reinschke, K. (2006). *Lineare Regelungs- und Steuerungstheorie*. Springer.
- Reinschke, K. and P. Schwarz (1976). *Verfahren zur rechnergestützten Analyse linearer Netzwerke*. Akademie Verlag Berlin.
- Schnabel, M., G. Nenninger and V. Krebs (1999). "Konvertierung sicherer Petri-Netze in Statecharts." *at - Automatisierungstechnik* **47**(12): 571-580.
- Schnieder, E. (1999). *Methoden der Automatisierung. Beschreibungsmittel, Modellkonzepte und Werkzeuge für Automatisierungssysteme*. Braunschweig, Wiesbaden. Vieweg.
- Schultz, D. G. and J. L. Melsa (1967). *State functions and linear control systems*. McGraw-Hill Book Company.
- Schwarz, P., C. Clauß, J. Haase and A. Schneider (2001). *VHDL-AMS und Modelica - ein Vergleich zweier Modellierungssprachen*. 15. *Symposium Simulationstechnik ASIM 2001*, Paderborn. pp.85-94.
- Schwarz, P. and T. Zaiczek (2008). *Torbasierte Rechnermodelle für ausgewählte mechatronische Anschauungsbeispiele*. Persönliche Kommunikation, Fraunhofer Institut Integrierte Schaltungen, Institutsteil Entwurfsautomatisierung, Dresden.
- Short, M. and M. J. Pont (2008). "Assessment of high-integrity embedded automotive control systems using hardware in the loop simulation." *Journal of Systems and Software* **81**(7): 1163-1183.
- Siciliano, B., L. Sciavicco, L. Villani and G. Oriolo (2009). *Robotics: Modelling, Planning and Control*. Springer.
- Thomas, R. E., A. J. Rosa and G. J. Toussaint (2009). *The Analysis and Design of Linear Circuits*. John Wiley and Sons, Inc.
- Tiller, M. M. (2001). *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers.
- van der Schaft, A. J. and B. M. Maschke (1995). "The Hamiltonian formulation of energy conserving physical systems with external ports." *Archiv für Elektronik und Übertragungstechnik* **49**: 362-371.
- van der Schaft, A. J. and B. M. Maschke (2002). "Hamiltonian representation of distributed parameter systems with boundary energy flow." *Journal of Geometry and Physics* **42**: 166-194.
- Vogel-Heuser, B. (2003). *Systems Software Engineering*. München. Oldenbourg.
- Wellstead, P. E. (1979). *Introduction to Physical System Modelling*. London. Academic Press Ltd.
- Yourdon, E. (1989). *Modern Structured Analysis*. Yourdon Press.



<http://www.springer.com/978-3-642-17530-5>

Mechatronic Systems Design

Methods, Models, Concepts

Janschek, K.

2012, XXI, 805 p., Hardcover

ISBN: 978-3-642-17530-5