Control Systems, Robotics and Automation

Modeling and System Identification I **Volume 4**

Edited by Heinz Unbehauen

ENCYCLOPEDIA OF LIFE SUPPORT SYSTEMS

In partnership with

UNESCO

United Nations Educational, Scientific and Cultural Organization

EOLSS

Encyclopedia of Life Support Systems

CONTROL SYSTEMS, ROBOTICS AND AUTOMATION

Modeling and System Identification-I

Editor: Heinz Unbehauen

Eolss Publishers Co. Ltd.,
Oxford, United Kingdom

Rev 1.1, October 2009

Information on this title: www.eolss.net/eBooks

ISBN 978-1-84826-143-3 (e-Book Adobe Reader)
ISBN 978-1-84826-593-6 (Print (Full Color Edition))

British Library Cataloguing-in-Publication Data
A catalogue record of this publication is available from the British Library.

Library of Congress Cataloging-in-Publication Data
A catalog record of this publication is available from the library of Congress

Singapore

CONTENTS

VOLUME IV

MODELING AND SIMULATION OF DYNAMIC SYSTEMS

Inge Troch and Felix Breitenecker
Vienna University of Technology, Austria

Keywords: System, Process, Model, Model of problem, Model of system, Modeling goals, Model structure, Model complexity, Equivalent models, Behavioral equivalence, Structural equivalence, Model simplification, Bottom-up modeling, Top-down modeling, Verification, Validation, Identification, Continuous-time simulation, Discrete-event simulation, Simulation experiment, Simulation run, Dynamic system, Static system, Deterministic system, Stochastic system, Continuous-time system, Discrete-time system, Discrete-event system, Hybrid system, State-events, Linear system, Nonlinear system, Parametric model, Non-parametric model, Time-domain, Frequency domain, Modeling concepts, Classification of systems, Classification of models, White-box, Black-box, Behavioral model, Structural modeling, Structured Analysis and Design Technique, Internal structure, Transition function, Semi-group property, Dynamical system, Simulation tools, Differential analyzer, Analogue computation, Hybrid computation, Feedback principle, Analogy of behavior, Principle of analogy, State-space concept, Input-output model, Input-state-output model, Cause-effect relationship, Manifest variables, Latent variables, Causal loop, Influence diagram, Credibility of results, Pseudo-random numbers, History of simulation

Contents

Summary

Models, especially mathematical models, are a powerful tool in automation and in analysis and design of control(led) systems. There are strong interconnections not only between the system, the modeling goal and the modeler but also between the model and the resulting solution for the industrial problem. The modeling process itself is examined from a rather general point of view highlighting especially those aspects which are important for automation and control. Further, simulation and simulation tools are discussed from the same point of view not only including topics such as model complexity, validation and verification but also numerical aspects. Important properties of systems, models and simulation tools such as e.g. *'continuous'* and *'discrete'* are not used in a unified way. Therefore, the most important pairs of properties are compared and reviewed briefly. Tools for modeling and simulation have become indispensable in automatic control. Their history is reviewed briefly together with the state-of-the-art with

reference to subsequent chapters of this chapter which provide more details.

1. Introduction

Today, creation, improvement and running of automated systems are based on the use of computers and consequently, on models. Such models can be verbal, lists of actions to be taken, mathematical expressions or computer programs to name but a few. Computer aided analysis and design and, consequently, modeling and simulation are fundamental tools in automation and control engineering. Models are used during the various stages of the design process of a new system, be it a model for the procedure of creating a new automation system, be it a model of a system for which a controller has to be designed or be it a model for the investigation of a newly designed system or of an already existing control system which has to be investigated. In many situations models are used to demonstrate that a new system or a new controller will indeed do the job it was designed for.

However, both terms, *'model'* and *'simulation'* respectively, are used not only within the control community but by engineers of various disciplines and also in everyday speech due to the universality of the concepts behind the terms. Both notions are closely related to what is called today a *'system'* or *'process'*. Unfortunately, the triple *'system - model - simulation'* is not well defined and hence, scientists or engineers using one or the other of these terms in a discussion may use them with quite different meaning - a situation which is told to be rather frequent in philosophical discussions but normally unknown among scientists or engineers. Therefore, the various meanings and uses will be exploited in the sequel from a control engineering viewpoint together with - some - of the answers to *'WHY'* has modeling and simulation become so important in control engineering.

There exists a variety of types of systems and consequently many possibilities to model a system. The type of model to be used will depend not only on the system under investigation or to be designed, but also on the modeler's background and preferences concerning approach to be taken and tools to be used in order to achieve successfully the given goal. This can be seen easily when scanning the table of contents of the chapter on *Control Systems, Robotics and Automation of this Encyclopedia*. Modeling in general and especially modeling of systems for control or automation purposes is based essentially on the knowledge of those system properties that are important for the specific task. Proper identification of important system properties indicates which model classes will be useful and, which simulation tools will be appropriate. It is well known that a system with certain properties can be modeled in several ways and that its mathematical model is not uniquely defined but can be of various nature such as e.g. differential equations or describing functions. However, knowledge of the various classification concepts for systems and models respectively, is helpful for appropriate choice of model and tools for analysis and simulation. Therefore, the most important classification aspects will be discussed shortly. This provides at the same time some insight into use and abuse of certain terms which are used not only by control engineers but are rather common also in a more general setting.

Further, important aspects model building are discussed not only from the point of view of control but also from a more general point of view, providing thus also some insights

into the reflections of scientists which were dealing with models and simulation either from a theoretic point of view or who worked in a different area of application. However, only those reflections, guidelines and results are presented in the following which are of interest for control engineering applications. Among them are general guidelines together with a discussion of appropriate model complexity, model structure and last but not least of relations between several possible models for one and the same system. Relations between models and the question of model equivalence is especially important in control engineering because there, many different description of one and the same system are often used in parallel - think e.g. on linear state space models and transfer function models or, on linear models derived by a special type of linearization or, on (linear or nonlinear) state space models of different order or different complexity etc. A detailed discussion of some of these aspects can be found in various places of the chapter on *Control Systems, Robotics and Automation*. Here, only the general aspects and the most important results will be presented which can - hopefully - lead to new theoretical investigations for various simplification techniques used in control applications.

It is to be noted that control engineers were among the first who used mathematical models for both for the analysis of plants and controlled systems and, for controller design. The same holds true for the more general setting of automation (of) systems. Analysis of a system, especially of a controlled one is based to a great extent on the evaluation of its future behavior under various disturbances. A further question concerns the dependency of the system's behavior on changes of parameters i.e. questions of sensitivity and robustness. An analytic answer to these questions is often difficult to derive especially, if nothing about the system's behavior is known. Therefore, often a first investigation of these questions is given on the basis of extended simulation experiments which yield the time-histories of important system variables for various choices of parameters and disturbances respectively. Such simulations became a general accepted tool shortly after World War II - however by using neither the term *'simulation'* nor the equipment control engineers are accustomed to today. This leads to a short historical view on simulation and simulation tools which demonstrates at the same time what has been achieved and which developments are to be expected.

2. Systems, Processes and Models

As early as in 1954 Soroka wrote that *'engineering problems have increased continually in complexity as the fund of scientific knowledge and the store of engineering experience have accumulated. The rough approximations used in the past for design purpose are often no longer satisfactory as modern practice makes more stringent demands on materials, processes, and performance. Analytical solutions are often impossible without further advances in mathematics, while numerical calculations on desk calculators may become impractical because of the length of time required or the cost involved. If such problems are to be solved, one must have recourse to high-speed, high-capacity automatic digital computing machines, or else one must resort to some experimental method. Engineers always have tended to rely heavily on direct experimentation to obtain numerical results. It may be that such experimentation is at times impractical or impossible. Under such conditions an indirect experimental approach by means of an analog computer or simulator may well prove to be feasible and economical'*. Meanwhile half a century has passed and these statements have — with the only exception of the

mentioned analog computer — not lost actuality.

Soroka states further, that *'the term 'analogy' means similarity of properties or relations without identity'*. And, this statement is one very appealing explanation of the characteristics of a *'model'*. A further — and also rather old — definition is due to Minsky: *'An object A is a model of the object B if an observer can use A to answer questions that interest him about B'*. Hence, a model is a representation of a system.

From the above follows that every model is related to an original i.e. to something what usually is called system or process and which is something *'real'* consisting of components and with boundaries which define what is to be included within a system and what belongs to the system's environment. Normally, it is assumed that the environment influences the system's behavior through inputs but, that there is no (or only negligible small) influence of the system on the environment. From this follows that - depending on the particular task and view on it — a controller may belong to what is called *'environment'* or, may be part of the system. The first view appears in controller design when plant and controller are viewed as two interacting but independent subsystems, the latter holds e.g. for stability or robustness analysis of the controlled system.

Reflections such as these indicate that the term *'system'* is not easily defined or explained. Ergo, there exist many explanations of it (some prefer process to indicate more clearly that there are changes as time passes). Rather common are descriptions like *'A system is a structured total of elements with well defined properties and with well established relations between these elements and with the environment'*. From the point of view of modern (computer-oriented) control engineering Cellier's favorite formulation, *'A system is a potential source of data'* is rather attractive. A goal-oriented exposition is given by Chorafas where a system is defined as a *'group of interdependent elements acting together to accomplish a predetermined task'*. Hence, a model is defined as the body of information about a system gathered for the purpose of studying, influencing or modifying etc. the system. Obviously, the purpose of the study will determine the nature of the information that is gathered. Hence, there is no unique model of a system.

Further explications of the two notions *'system'* and *'model'* can be found not only in many books on modeling and/or simulation but also in almost every encyclopedia. Among the more recent definitions of the term 'model' the one of Sage refers to the fundamental importance of mathematics for control engineers of today: *'A model is an image or abstraction of reality; a mental, physical or mathematical representation or description of an actual system.'* Engineers (no matter whether or not they were called by this name) are concerned with finding a realizable solution for a certain problem. Through centuries they used their imagination (i.e. mental models), more or less accurate sketches (i.e. visualization) or a physical model - often of reduced scale. Calculations were restricted mainly to static relations between a few quantities or, to rough estimates which often resulted from experience (rule of the thumb). It is important to note that the above concept of a model being the image of reality covers two aspects - the *'descriptive model'* of a really existing system and, the *'prescriptive model'* of a system which hopefully will exist and work satisfactory in the future.

However, demands have increased considerably and improvement of existing - industrial,

engineering etc. - control systems or design of new ones is in most cases only possible, when appropriate mathematical models are used. A mathematical model is an abstract, simplified, mathematical construct related to a part of reality and created for a particular purpose. It is adequate if it is adequate for the goal in the mind of the modeler. It is important to note that *'a mathematical model is a symbolic representation composed of mathematical symbols. These symbols have precise mathematical meaning and the manipulation of symbols is dictated by the rules of logic and mathematics. A mathematical formulation becomes a model by relating its symbols to a system characterization'.*

Manipulation of variables contained in model need not be analytical. Often, a model is formulated as computer model and a solution is sought by experimenting with it, for instance by varying parameters. This was for long the case especially for discrete-event systems and led to a goal-oriented explanation *'A model is a description of some system intended to predict what happens if certain actions are taken'.*

So far, the arguments centered on a *'model'* which describes certain phenomena or - using control terminology - the plant and maybe some or all of the restrictions to be obeyed. If analysis of an existing - controlled or uncontrolled - plant is of concern then a model of the plant will be sufficient and appropriate for insight. But, whenever design is concerned, then a model for decisions is needed. In such situations not only the plant but also the task in question needs to be modeled. Such a 'complete' model will make the designer's task better defined and hence, somewhat easier. Moreover, application of design tools will become feasible. Such tools are certain mathematical methods such as e.g., optimal control design via maximum principle, dynamic programming or parameterization for which often computer support is available as e.g. in the various toolboxes of Matlab (Some information about this software tool is provided in Section 6, for more detailed information see *Software Development and Trends*) such as pole placement, LQG design, model predictive control, nonlinear control, optimization etc. Consequently, relations describing the various restrictions on plant variables should be included in the model as well as suitable formulations of all constraints on the controller(s), no matter whether these constraints result from physical, chemical etc. properties or from economic, safety etc. considerations. Last but not least, also the various and often conflicting goals have to be modeled.

Today, it is common practice to view a system as a process that converts inputs over which the system has no direct control, to outputs i.e., one type of behavioral quantities to another. As long as interaction of a system or a subsystem with its environment (which may be one or several further subsystems) is concerned, a description of the relations between these two types of behavioral quantities would be sufficient. Such a description is called a *behavioral model* or *I/O-model* or *black-box model.* However, a system may react to one and the same input in quite different manners depending on its history. In order to model also these effects, the notion of state was introduced. The state is regarded as the most concise description of the system's past history. The current state and subsequent inputs determine the future states of the system. In the corresponding model, the state (if properly defined) contains all information needed to calculate future responses without reference to the history of inputs and responses. There are many ways that it could have gotten to the current value, but this history is irrelevant for the future

responses. A mathematical model of such a system needs only the present state and future inputs to calculate future responses.

However, this input-output or input-state-output view of a system has also shortcomings and is no longer accepted unanimously. First, it is important to note that the choice of state variables for a particular system is not unique - most physical systems can be described with many different sets of state variables. Moreover - and maybe even more important - also the choice of inputs and outputs needs not be stringent. Electrical circuits are good examples in which the state-space concept has severe shortcomings and - as suggested by Jan Willems - the use of manifest and latent variables may be more appropriate for modeling dynamical systems.

3. Simulation

There are few terms that have changed so much their meaning and are used within so different settings as 'simulation'. Obviously, its roots are the Latin 'similis' and in every day life *'to simulate'* means (e.g. Webster's Collegiate Dictionary) 'to feign, to attain the essence of, without the reality'. Since long, simulation was connected closely with two areas, control engineering and the investigation of server/queuing systems. The models behind are quite different - for a long time only two types were used—differential equations in one area and models based on the use of probability distributions in the other. Meanwhile, systems and hence, also models are of great variety - events occur in many continuous-time systems either as state-events or as (random) stochastic events. Moreover, discrete-event systems have continuous-time subsystems. Therefore, boundaries have become weak and most simulations performed today are in one or the other sense 'hybrid'.

Nevertheless, it is useful to see how the term *'simulation'* can be defined and used. One rather commonly accepted definition of simulation dates back to Granino Korn and states: *'A simulation is an experiment performed on a model.'* However, this raises two further questions: "What is an experiment?" and "Why is it performed?"

Moreover, simulation is here closely related to the notion of a model. This holds true also for two further statements. The first was given by McLeod on the occasion of the founding of the journal 'Simulation': *'... as Editor of the Journal, I proclaim ..... simulation to mean the act of representing some aspects of the real world by numbers or symbols which may be easily manipulated to facilitate their study.'* The second statement stems from VDI-Richtlinie 3633, and reads as *'Simulation is the imitation of a dynamical process in a model in order to derive knowledge which can be transferred to reality.'* (The original German text reads*: 'Simulation ist die Nachbildung eines dynamischen Prozesses in einem Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind.'*) Again, we have a purpose behind simulation which (as stated by Mezencev) can be *'to draw conclusions about process properties'* achieved by *'driving a model of a system with suitable inputs and observing the corresponding outputs'*. This is rather close to Cellier's definition: *'An experiment is the process of extracting data from a system by exerting it through its inputs'* as background. This applies to many applications of simulation, be it analysis of an existing one (such as the plant) or a proposed system (e.g. a newly designed controller) or, design of a new system by a guess-and-test method

or, prediction, what will happen under certain - normal or extreme - conditions or, information on what to do to allow for a smooth running during the system's whole life cycle. The last question is not as easily answered because it requires inclusion of the design process. This has led to a more general definition of the term experiment: *'An experiment is the application of a method to a model'*. As a consequence, not only time histories of inputs and corresponding outputs are experiments but also methods such as (optimal) controller design, stability analysis, linearization, statistical analysis of certain events etc.

Obviously, modeling and simulation are closely related. Perhaps, a statement by Ingels gives a suitable conclusion for the above discussions: *'Modeling is the development of equations, constraints, and logic rules, while simulation is the exercising of the model'*.

Control engineers were among the first to use simulation as an important tool for model development, for controller design and analysis of controller performance. However, at these early days, one did scarcely talk about continuous-time simulation but about analog computation and later on about hybrid computation. In these early days, the term *'simulation'* was more extensively used by people interested in the design of server systems (e.g. bank counter, fast food restaurant, taxicab flow) and other queuing systems such as renewal or repair processes. From this resulted the situation in which the term *'simulation'* was often used synonymously for *'discrete-event simulation'*. Gradually, terminology became more precise. Today, similar to modeling and as given in more detail in Section 4, simulation is usually (although not always) classified into four types

(1) continuous-time (differential equations)
(2) discrete-time (difference equations)
(3) discrete-event
(4) hybrid.

Last but not least, it should be mentioned that control engineers often confronted with the challenge to perform not only a simulation but to do it in real-time because hardware such as a newly designed controller, or human operator is to be part of the simulation system. Well-known examples for the latter are flight simulators.

4. Classification of Systems and Models

As stated earlier, a system is a collection of one or more related objects which normally are physical entities with specific characteristics or attributes and, which can interact but need not to do so. Further, entities and the system may change with time. Aspects such as these together with the various types of changes are characteristics which lead to classifications of systems and models. Unfortunately, again these characteristics are used in the various scientific and engineering areas differently. Especially the property 'discrete' is often used without specification to which variable it applies. Depending on area and author it can denote discrete-time, discrete-event, discrete state or output variables respectively. Therefore, an overview of some of the most important characteristic features or, more precisely, characteristic pairs used for classification is given and their various uses are discussed briefly.

4.1. Properties of Systems and Models

The first group of pairs of properties relates to both, system and/or model (see General Models of Dynamic Systems):

dynamic - static: A system or model is called static, whenever the relations between all relevant elements of the system do not depend on time. There are only few types of possible and interesting models for static systems as indicated below. Dynamic formulation involves two types of variables, independent and dependent ones. Usually, at least one independent variable concerns time, either a sequence of instants (discrete-time, discrete-event) or values in an interval (continuous-time). A variety of models comes up depending on the other properties of the system. A short discussion follows at the end of this (incomplete) list of properties used for classification.

deterministic - stochastic: In many real-world problems demands (events) occur whose occurrence and lengths can, in general, be specified only probabilistically. Examples of such systems are computer systems, communication networks, inspection, maintenance and repair operations, industrial production processes, and inventory systems. The system is of random nature and called a stochastic system. Among them especially Markov processes are of interest and rather well understood. They can be discrete or continuous with respect to state or time, respectively. Exact prognosis (i.e. computation if all data is given) of future states is not possible. Moreover, a process can be stationary which means that its distribution function is invariant under time shift. Deterministic systems show in principle predictable behavior, if modeled exactly; the future behavior can be computed from its present state and the known influences on it. However, it must be noted that being deterministic does not necessarily result in what in every-day-life is called *'predictable'* behavior. Also deterministic systems and consequently, models (even very simple ones) may show a behavior where very small changes in one parameter lead to large and unexpected changes in the system's future behavior. The various behaviors are in principle computable (if all computations are indeed exact) but ad hoc prediction of long-term results of small parameter changes is impossible. The technical term for this is known as chaotic behavior of variables. Examples are beat of airplane wings, or the growth of certain insect populations. It should be noted that there are systems which - for the same goal - can be modeled in both ways e.g. weather forecast (for a detailed discussion of models for stochastic systems see *Models of Stochastic Systems*).

lumped parameter - distributed parameter: Both properties are related to systems of dynamic nature. The elements of it may change with time only or, may depend on time and space as the oscillations of a beam or BOD- and DO-values (BOD and DO stand for biological oxygen demand and dissolved oxygen respectively which describe water quality and hence, are important for its control) along a river. In the continuous-time case lumped parameter systems are described by ordinary differential equations whereas distributed parameter systems are modeled by partial differential equations. In the time-discrete case the simplest models for the latter are so-called 2D- or 3D-systems. Sometimes, distributed parameter systems

(see *Some Basics in Modeling of Mechatronic Systems* and *Modeling and Simulation of Distributed Systems*) are called infinite-dimensional systems. However, the latter mathematical term includes also systems with (finite or distributed) dead lag i.e. differential-difference equations and integro-differential equations.

stationary (time-invariant) - time varying: A system is called stationary when it is invariant under arbitrary time shift. In the deterministic case such systems are modeled for instance by differential (difference) equations which do not depend explicitly on time (t, t_k resp.), in the stochastic case their distribution function is

invariant under time shift.

continuous-time and discrete-time: This is not a precise contrast, because systems may change at discrete instants of time in a deterministic manner or, such changes may occur stochastically. Therefore, it is better to distinguish between three properties as follows:

continuous-time and discrete-time - discrete-event - hybrid: Systems which change continuously with time include very often a plant to be controlled. Among the most commonly used models are differential equations and bond graphs or, for linear systems only, transfer functions and frequency domain descriptions (operator models). Other systems change only at certain instants of time or during rather short intervals. Hence, one uses models (e.g. difference equations) which account for this fact. Moreover, certain operations (e.g. maintenance and repair, grinding, path welding) in industrial production processes appear randomly and/or have a duration which is of random nature. These systems are subjected to a sequence of countable events where it can be assumed that nothing of interest takes place between them which leads to the name discrete-event system or model, respectively. It must be mentioned that discrete-events of deterministic nature are also known. More precisely, one talks about state-events which normally appear in continuous-time systems as a switching between behaviors such as the switching between rolling and sliding of a car when aquaplaning takes place. This is one rather simple example of a hybrid system where both continuous time and (stochastic or deterministic) discrete-events are present. Continuous-time and discrete-time models are discussed in various contributions to the chapter on Control Systems, Robotics and Automation (see e.g. *Models for Discrete Event System* , *Modeling and Simulation of Large-Scale Hybrid Systems*, *Modeling of Hybrid Systems*).

continuous - discrete: This distinction dates back to the controversy between analog (= continuous) and digital (= discrete) simulation. Meanwhile, use of these two attributes without further precision should be avoided because this causes confusion. Both, dynamic systems and models have dependent and independent variables and one, several or all may change continuously or in a discrete way. It shall be mentioned that these qualities can be used also with a quite different meaning as in problems arising in continuum physics where discrete models may refer to models consisting of ordinary differential equations, whereas continuous

models refer to partial differential equations.

linear - nonlinear: Linear systems have the nice property that the principle of superposition is valid. Therefore, they are - relatively - easy to understand and to handle (the latter only in the time-invariant case where several easy-to-handle descriptions are available). However, it is well-known, that real-world systems have essentially nonlinear behavior, and linear models are therefore always approximations which can be used only in a certain neighborhood of a working point etc. This dilemma between easy-to-handle and realistic has been formulated very nicely by Pindyck already in 1972: *'Our preoccupation with linear time-invariant systems is not a reflection of a belief in a linear time-invariant real world, but instead a reflection of the present state of the art of describing the real world'.*

4.2. Properties of Models only

The following features relate only to models (including so-called *'simulation models'* which are used only for the computation of time histories):

parametric - nonparametric: A parametric model represents a generic system in a certain class of systems (e.g. linear ordinary differential equation or equivalently transfer function $y(s) = \frac{b_m s^m + \ldots + b_1 s + b_0}{a_n s^n + \ldots + a_1 s + a_0} u(s)$ with known or unknown n and m); the concrete model for a given application is determined by assigning numerical values to constant parameters contained in the model. It is characterized by a finite number of parameters (e.g. ARMAX what stands for Auto-Regressive Moving Average with eXogenous input, $A(q)y(k) = B(q)u(k) + C(q)e(k)$ where q is the backward shift operator, $q[z(k)] = z(k-1)$ and A, B, C denote parametric polynomials. The definition of a non-parametric model is varying but concerns essentially the same type of models. Matko, Karba, and Zupancic say that a nonparametric model contains no parameters and thus represents only one specific system, its response is obtained directly from the system or indirectly through experimental analysis. On the other hand, Bosch and Van der Klauw state, that nonparametric models contain an infinite or large number of parameters as in the case of impulse response of discrete-time systems $y(k) = \sum_{j=0}^{\infty} g(j)u(k-j)$ or the Infinite Inpulse Response (IIR) transfer function $G(q) = \sum_{j=0}^{\infty} g(j)q^{-j}$ where q again denotes backward shift. Parametric models can thus be obtained from nonparametric models with the aid of numerous and various identification methods, while conversion in the opposite direction is achieved by simulation.

time-domain - frequency-domain : Time-domain models handle variables of interest as functions of (continuous or discrete) time, models consist of differential or difference equations with or without discrete-time events), DAEs etc.

Frequency-domain stands for a description achieved by a mapping between the space of time-functions and some other functions space. Normally, Laplace transform is used as mapping leading to transfer functions in the linear time-invariant case and to describing functions in the nonlinear case (for details see *Description of Continuous Linear Time-Invariant Systems in Time-Domain* and *Description of Continuous Linear Time-Invariant Systems in Frequency-Domain* and *Describing Function Method*).

qualitative - quantitative: These terms concern the type of scales used for measuring i.e. whether intervals or nominal values are used. While the latter was usually done for rather long time and is still done in many cases successfully - although one knows quite well that every nominal value stands in reality for a certain 'small' interval, the length of which depends on measuring and/or computing accuracy. The last decades saw a growing interest in *'fuzzyness'* and hence, in qualitative models. The fuzzy-concept is a qualitative one: it is based on intervals with boundaries which are not exact but again some sort of intervals. Peschel indicated already in 1976 the importance of Zadeh's fuzzyness concept for modeling (for a survey on data-based fuzzy modeling see *Data-Based Fuzzy Modeling*).

insight - decisions: From the control engineering point of view this is more or less equivalent to the distinction between models for system analysis and models for design. However, one has to be careful not to expect that every model used for system analysis be at the same time an analytical model — insight into a system's properties and behavior can be gained also from computer models.

analytical - simulation: The term *'analytical'* refers in most cases to a model which is formulated as a set of equations and inequalities or, more generally, as a set of graphs, nets etc. which hopefully can be investigated further using mathematical techniques. A simulation model originally consists of computer code of a certain level and type used for the computation of time histories. However, development of programs such as Maple or Mathematica (some information about this software tool is provided in Section 6; for more detailed information see *Software Development and Trends*) with their capability to manipulate formulas has made this distinction to a historical one.

analytical - algorithmic: This is an out-of-date distinction cited here only for the sake of completeness. Originally, a mathematical model was called analytical if the relation between variables, parameters etc. are given in form of functional terms i.e. as equations and inequalities. A model is algorithmic if changes in the process variables are determined by an ordered sequence of elementary operations and conditions where, after each step what is to do during the subsequent step is uniquely determined . This distinction has also lost actuality - due to increasing complexity of systems, models now in use may have parts which are analytical and parts which are algorithmic.

off-line - on-line: Off-line simulations can be in real-time e.g. for operator training, or, can be slower or - hopefully - quicker than real time. Online simulation can be real-time e.g. if the simulator is used as essential part of a controller (observer,

Kalman filter) or, for fault detection and fault diagnosis or it can be quicker than real time. The latter is necessary when prediction of critical states, dynamic optimization, predictive control, etc. are of concern.

micro - macro: Concerns the type of data used for modeling and simulation respectively, i.e. whether individual data or aggregated data is used. This distinction appears primarily within economic considerations. In principle, it is also valid for the investigation of production processes or multi-level control of complex systems although normally these terms are normally not used in the latter areas.

4.3. Some Additional Remarks on the Properties *'Static'* and *'Dynamic'*

Normally, a static formulation is a mathematical one involving either algebraic equations (here *'algebraic'* is used in a generalized sense where other functions such as trigonometric functions are allowed) such as e.g. $T \cong 2\pi\sqrt{L/g}$ for the period of a so-called simple pendulum of length L or, function optimization with one or more variables - an example is profit maximizing product distribution within a firm during a certain production period - which results in a mathematical model of the form $\max_x f(x)$ subject to $g(x) \le 0$ where f is scalar and x and g can be vectors. As mentioned, dynamic formulations involve usually time as at least one independent variable. For deterministic systems the resulting models are in the discrete-time case difference equations (in one or more independent variables), and in the continuous-time case ordinary or partial differential equations, integral equations, differential equations with (one or several) discrete time lag(s) (also called differential difference equations or differential equations with retarded argument) and integro-differential equations (e.g. for systems with continuous time lag).*

Among the models for dynamic stochastic systems, regression formulations such as $Y(t_i) = g(X(t_i), t_i) + V(t_i)$ are quite frequent, where $X(t_i)$ represents the input(s), $Y(t_i)$ the output(s) and $V(\bullet)$ is an unknown stochastic variable. Examples for such models are queuing systems or renewal processes (e.g. in an industrial production plant). In control engineering ARMA systems (Auto-Regressive Moving Average) and their variants have become rather popular. There, $X(t_i)$ is the sum of a deterministic $Y(t_i)$ and a stochastic one, $V(t_i)$. The changes in $V(t_i)$ are given by a linear difference equation formulation of the form $V(t_i) = \sum_{j=1}^{n} \alpha_i V(t_{i-j}) + \sum_{j=1}^{m} \beta_i W(t_{i-j})$ where $W(t_i)$ is a sequence of independent and identically distributed Gaussian random variables with zero mean and variance σ^2. The parameters α_i and β_i are constant. However, time can also change continuously. This leads to a formulation using stochastic differential equations (see *Models of Stochastic Systems*). However, if given their general form mathematical theory does not provide yet results which allow for an easy application for control design. Consequently, only special types and especially linear state space models $\dot{x} = Ax + Bu + Cv$ with additive (white or colored) noise v have become quite popular.

5. Modeling

Models need not be abstract constructs; they can be of physical nature e.g. electrical circuits can serve as a physical model based on analogy of behavior or, when investigating e.g. water flow in a river for improving its banks often physical models of reduced scale are used. Further, development of a so-called *'mathematical model'* is often based on preliminary verbal models, on rough sketches, causal-loop diagrams or systems dynamics influence diagrams indicating the system structure and components etc. However, modern control engineering is based on formal models for the dynamic behavior of a system and of the underlying constraints and requirements. These formal models can be equations and inequalities of various types and using various spaces (e.g. time-domain, frequency domain) but also graphs (see *Bond Graphs*), nets (e.g. Petri nets) but also algorithmic descriptions or computer code. In the following only mathematical models will be considered, however, some of the aspects discussed there hold true also for other types of models. A good starting point is a statement of S. I. Hayakawa: *'The symbol is NOT the thing symbolized, the word is NOT the thing, the map is NOT the territory it stands for.'*

In principle, modeling is simply *'establishing the model structure and supplying the data'*. However, this is oversimplification; there are some rules yet, good *'modeling'* is often considered to be an art, as expressed e.g. in the title *'Systems and simulation - the art and science'* of a book by Shannon. Normally, there is no straightforward procedure of creating a mathematical description of e.g. physical, chemical, electro-technical, biological or environmental phenomena such that this representation is relatively simple, yet accurate enough to serve the purpose of the modeler i.e., can be used to solve the given problem at acceptable cost and within reasonable time. Normally - and especially in control engineering, not only the process (plant) has to be modeled but also the constraints and the various objectives.

5.1. Some General Considerations

The subsequent statements and reflections are despite their rather general formulation important for the modeling of control problems. They are concerned primarily with the mathematical modeling of a (dynamic) system and hence, in general with the modeling of the - controlled or uncontrolled - plant.

5.1.1. Modeling and Modeler

Creating a mathematical model means to move from the real world into the abstract world of mathematical concepts. Then the model is manipulated using mathematical techniques or computer-aided numerical computation, respectively. Finally one re-enters the real world and there, the solution to the mathematical problem is then translated into a useful solution to the real world problem. Any model must have a definite purpose which is clearly stated at the start. This statement may itself vary according to the point of view of the model user. In some cases and there can be a clash between opposing groups of model users regarding the particular objectives involved. For example, the effect of a new road bypass on a town center traffic jam ..... Therefore, a model is always a relation between three components *'original system - model system - modeler'*, Figure 1. The model is

created by the modeler accounting not only for the modeler's interests but also for his/her knowledge and leading ideas.

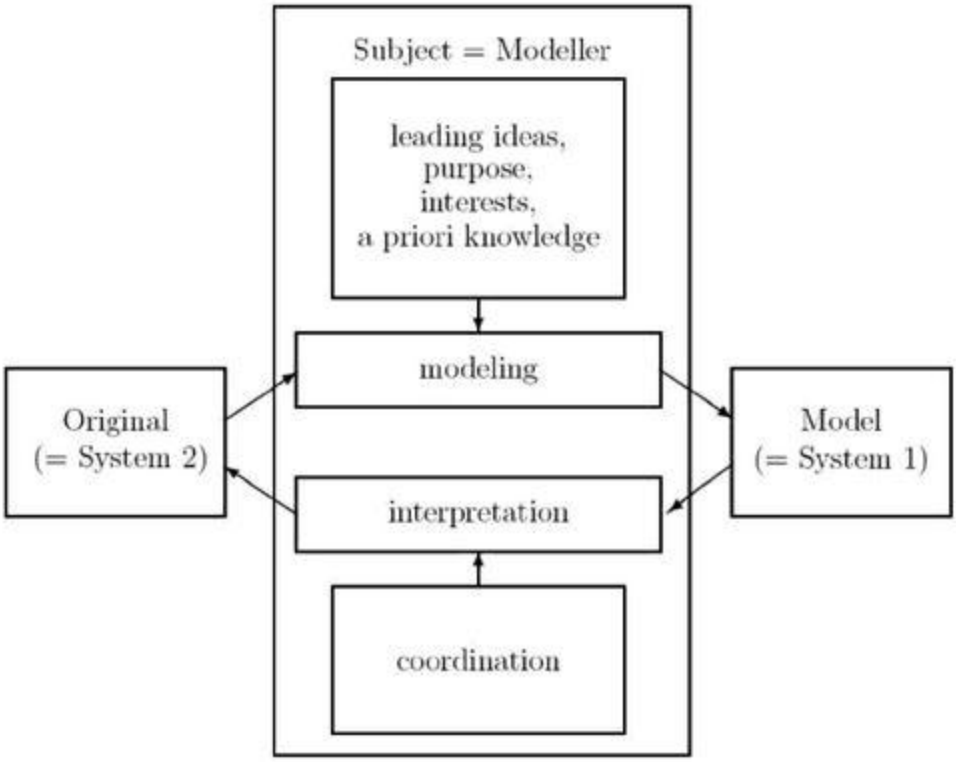

Figure 1: Relations between system, model and modeler

This demonstrates also, that there is not one right and proper model for a particular problem. Many different models can be developed for tackling the same problem. Some models may be better than others in the sense that they are more useful or more accurate, but this is not always the case. Any model will have limited range of validity and should not be applied outside this range.

One of the various goals of modeling is to gain a better understanding of the system's behavior or to test its behavior when a newly designed controller is added. For such an analysis of a system's behavior it is known for long that one has to be careful not to misuse the model such that it supports preconceived ideas on the system's behavior or on what a good control has to look like. Such abuse can result from the used mathematical formulations or from imposing artificial constraints. For analysis of system's behavior a simple and well-known example can be taken from population analysis. If the modeler assumes that the rate (with respect to time) of change of the size N of a population is proportional to its size then a model

$$\frac{N(t_{k+1})-N(t_k)}{t_{k+1}-t_k}=\alpha N(t_k) \qquad \text{or} \qquad \frac{dN(t)}{dt}=\beta N(t) \tag{1}$$

results depending on whether a discrete-time or continuous-time model is created. In any case, exponential growth of N results for mathematical reasons and has not been proved! Similar reflections hold true for the choice of method for controller design. When based for instance on optimization concepts, the concrete formulation of goal and constraints determine the type of resulting control. When designing fuzzy controls then the concrete choice of how e.g. fuzzyfication and defuzzyfication are performed or, in learning control which learning concept is used. In conclusion, it is always the control engineer and modeler who decides what is good and what is not.

It is also important to remember, that the same abstract model can often be used for quite different physical situations. This results from analogies between systems of different nature. A well-known and very simple example is the mechanical one-mass-oscillator (consisting of mass, damper and spring) and the electric RLC-circuit. Other examples are sand models of oil reservoirs or, soap-film models for the investigation torsional-stress distribution in beams. Such analogies are often discovered only after a model of a system is established and, can be successfully used if there are also analogies between the tasks to be solved. This indicates one benefit of a modeling effort - sometimes obtained unintentionally: modeling forces an improved understanding of what is really happening in the system being modeled. If a model does not reflect some real understanding and insight about the system, it is probably useless.

5.1.2. Modeling and Modeling Goals

Design of a controller or of any other system, often requires repeated simulation runs to identify designs or policies which are good. However, specification of what is termed 'good' is vital, objectives and constraints must be made explicit and hence, formulated mathematically. Often that is one of the hardest but most rewarding aspects of modeling. Consequently, the first step in a (simulation) study is the identification of its objectives or, as Einstein once stated *'the proper formulation of a problem is even more essential than its solution'*.

Basic modeling goals are analysis or design, respectively. These goals can be achieved either analytically by using suitable mathematical methods or by simulation. However, design of controllers, of a plant etc. should always be followed by a careful analysis of the newly designed system. Therefore, in both cases a model is used to predict what happens if certain inputs, stimuli, controls, disturbances, certain actions excite the system under investigation. However, the model used for design and the one used for analysis need not be the same. Any useful model simplifies and idealizes; normally a model used for design simplifies to a greater extent than a model used for the subsequent analysis. A whole spectrum of models can be considered, increasing both in realism and complexity, though increasing the complexity need not necessarily increases realism. The main question is always whether the model used adequately reflects the features of the real system that are important to the application in mind.

When building a model of a process, two important propositions should be remembered:

(1) Any useful model simplifies and idealizes.
(2) Definition of the boundaries between system and environment is rather arbitrary.

Many forces that influence the system's behavior must be neglected. This is in part due to insufficient knowledge of the system and results on the other hand from the requirement to keep the model tractable. Normally, there is no rigorous proof that such neglect is justified. This is a dilemma and a request. The model should contain only those features or characteristics of the system which the model builder feels are relevant and significant for the goal in mind or, as already Aristotle wrote *'It is the mark of an educated man to look for precision in each class of things just so far as the nature of the subject admits.'* And it is a good rule not to go into more details than indeed necessary or, (following Ockham) *'What can be done with fewer assumptions is done in vain with more'*. So, in general parsimony in assumptions and parameters is considered to be good.

Often problems have no single answer because different models can illuminate different facets of a problem. This aspect was investigated to some extent from a model-theoretic point of view by Zeigler already in 1984. Multifaceted modeling leads to hierarchy of models. Models high on the hierarchy pay little attention to system details, whereas models lower on the hierarchy are detailed models whose objectives are to obtain information about a part of the overall system.

5.1.3. Model Structure

In control engineering, today's relatively large systems are divided into connected and interacting subsystems and consequently, also the resulting model consists of models for the various subsystems and of models for the interactions. Control design for the complete system in many cases can be based on a relatively simple model for the total system. On the other hand, design of a controller for an individual subsystem requires a good model of that subsystem in which the other subsystems belong to the environment and need not always be modeled in every detail. But, one has to be careful because modeling and its details depend also on which method for controller design the modeler wants to apply. In this respect a statement by Levins, *'It is not possible to maximize simultaneously generality, realism and precision'* is notable. Therefore, the significant features have to be identified and then translated into mathematical entities, and in relations between these entities.

Model structure and details of modeling are strongly influenced by both, by modeling goals and by the analysis or design methods to be used with the model. Especially when design is concerned, there will often be a strong preference for that design method(s) which one has used in the past successfully and/or with which the engineer is familiar. But one should be aware that not only the model developed but also the resulting control and its qualities depend strongly on that choice. For instance, preference for LQG-design necessarily requires a linear process model and neglect of all nonlinear effects no matter how strong they are. On the other hand, use of more recent tools based e.g. on the concept of a flatness allow for rather detailed modeling of nonlinear effects. A further aspect is the description of the desired behavior of a system by using an appropriate model. Construction of such models is needed especially in control design be it in the context of

model-reference-adaptive control or, in control design for linear systems by pole placement or stabilization techniques or, when designing controls for nonlinear systems by performing as a first step linearization and/or decoupling by static or dynamic feedback.

Increasing demands and a more general point of view of controlled systems resulted in the need to model also systems of complex structure. Aspects such as overall performance of a system with many subsystems, productivity, quality of product, economy, environmental demands etc. do not allow any longer use of extremely simplified and decoupled models. Couplings need to be considered and hence, relatively large and complex models result which have continuous-time and discrete-event structure. Moreover, control often requires also to consider not only the classical control design or control analysis problem but at the same time also flow of materials, communication between various production units etc. For this purpose often simulation models are used where the individual subsystems must be appropriately modeled.

In any case, relations between system, model and the questions to be answered are important and need careful investigation. It is known since long that nearly always there is a tendency to simulate too much detail rather than too little. Thus, one should always first state the questions to be answered and then design the model around these questions rather than try to imitate the real system exactly. Pareto's law says that in every group or collection exists a vital few and a trivial many. Nothing really significant happens unless it happens to the vital few. Moreover, the truly significant aspects and relationships may get lost in all the trivial details. Therefore, the model must include only those aspects of the system relevant to the study objectives.

The SADT (Structured Analysis and Design Techniques) is one of the first theoretical efforts to give a method for "asking the right questions to find the right answers". The basic idea is given in Figure 2. The bottom of a box is reserved to indicate a mechanism, which may be the person or device carrying out a function. A box represents the three basic questions:

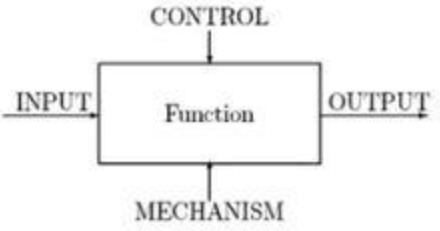

Figure 2: Basic idea of SADT

The object-oriented approach has as basic principle "abstraction" by using models organized around real-world concepts. The fundamental structure is the object, which combines both data structure and behavior in a single identity. The greater emphasis on the essential properties of an object forces the modeler to think more carefully and more deeply about WHAT an object IS and DOES rather than HOW it is used. The fundamental characteristics required by an object-oriented approach are: identity

classification, polymorphism and inheritance. This way of thinking is based on ABSTRACTION that exists in real-world. So abstraction is the selective examination of certain aspects of problems, the goal being to isolate those aspects important for some PURPOSE. Abstraction mechanisms cover the following concepts:

- classification/exemplification
- aggregation/decomposition
- generalization/specialization.

Abstraction is a fundamental human capability that permits us to deal with complexity. To build complex systems, the developer must abstract different views of the systems. The proper use of abstraction allows the same model to be used for analysis, high-level design, program structure, database structure and documentation.

The object-oriented and structured approaches are different for the world's view of the system; they share common features, because the reality is between structure and function. An ideal methodology will be one which associates the advantages of both.

5.1.4. Model Complexity

Reflections such as the above ones lead to guidelines such as these: *'factor if possible the system problem into simpler problems - establish a clear statement of the objectives - seek analogies - write down the obvious - if a tractable model is obtained, enrich it, otherwise simplify.'* However, reality is not as simple. Not every simple model needs enrichment; not every complex model is easily simplified. Whether more or less details are to be included in an existing model is - at least in part - due to the two main modeling approaches. Normally, pure structural modeling leads to a relatively large, detailed and complex model with many parameters which is difficult (if at all) to handle. On the other hand, if there is no or a little *a priori* knowledge of the system to be controlled, the modeler will have to start with a simple model derived e.g. via the systems analysis approach or as an appropriate guess of a transfer function etc.

This gave raise to two basic philosophies called *'bottom-up'* and *'top-down'*. Bottom-up means enrichment e.g. by including further variables, including terms that allow for nonlinear behavior, accounting for higher-order derivatives etc. Especially in the behavioral modeling approach, it is usually recommendable to learn from experiences by starting with relatively simple models.

Top-down approaches start with a relatively detailed model which is then simplified as far as appropriate. They are quite common in structural modeling. This concerns especially reducing the number of variables by changing some into constants, linearization, adding stronger assumptions and restrictions, restricting the boundaries of the system, and, last but not the least, order reduction. The latter is equivalent to eliminating or combining some variables. Control engineers have developed a number of powerful tools for this purpose (see *Order Reduction*). Linearization is a well-established mathematical concept for which powerful software tools also on the symbolic level are available. A special case is the describing function method (see *Describing Function Method*). It is an open question whether or not the various linearization concepts based on

static or dynamic feedback belong to model development or not. In view of e.g. control constraints, this feedback has to be regarded as part of the control law. But, at the same time a new (controlled) system is created for which a control has to be designed in order to achieve the original goal. This demonstrates again the strong interconnections between model and control.

In principle, there are two main approaches:

(1) **The behavioral approach** is based on induction and views the system as *'black-box'* where nothing is known about what is going on inside. It is characterized by experiments i.e. by the collection of data. For the purpose of control this leads especially to the observation of inputs and corresponding outputs. Hence, the behavior of the system is modeled. Then, one tries to construct theories or hypotheses that account for the observed behavior. Meanwhile, there are systematic methods known as *'system (process) identification'* which allow us to develop a model from experiments, provided suitable assumptions (e.g. linearity) on the system's structure and/or properties are made (see *Frequency Domain System Identification, Identification of Linear Systems in Time Domain, Identification of Nonlinear Systems, Bound-based Identification and Practical Issues of System Identification*).

(2) **The structural approach** uses deduction and is based on prior knowledge of the process, as collected in fundamental laws of i.e. physics such as preservation of energy or impulse, mesh and node equations, d'Alembert's principle, Lagrange equations etc. Variables of the model represent the signals of the system, its components relate directly to (and interact as) the corresponding system components. From the basic assumption that everything is known about the system, the resulting model received its name of a *'white box'* model. Once the model structure is established, the various parameters must be assigned concrete values either from direct measurements or in an indirect way known as (parameter) identification (see *Identification of linear Systems in Frequency Domain, Identification of Linear Systems in Time Domain, Identification of Nonlinear Systems, Bound-based Identification and Practical Issues of System Identification*).

It must be noted, that reality is not as simple. Normally, every modeling approach is somewhere in the broad grey-box zone between black-boxes and white-boxes. There are several black-white-scales published in the literature, but as every system is an individual one, all these scales are at the same time true and false. To give an idea of such scales two shall be given here, each from white to gray. It is rather commonly accepted that electric circuits, mechanical multi-body systems and many chemical reactions are rather well understood, whereas hydraulic, pneumatic systems or multi-layer systems are more difficult to model. Environmental, biological, physiological, economic and social systems are even more difficult and - depending on the goal of the investigations - belong to the gray or dark gray area.

System dynamics is one method to establish input/output models for systems for which no basic laws are known. Basic idea is the representation of the (rate of) change as difference of (the respective rates of) increase and decrease such as e.g. the rate of change of a population as difference of birth rate and death rate. The various parameters of

proportionality can be constant but can also depend on system variables such as e.g. in the model for logistic growth. In most cases, the resulting models are linear ones. However, the basic idea can result also in nonlinear models such as models for (optimal) feeding and/or harvesting based on logistic growth or, in models for controlled or uncontrolled prey/predator systems used e.g. in (biological) pest control, to name some very simple ones. Often, such models are set up using so-called influence diagrams. Using differential equations, these models are derived from equations of the type (1) while taking into account interactions between individuals and are of the type

$$\dot{x} = x(\varepsilon - \mu x) + u - v \tag{2}$$

with feeding u and harvesting v of a population x whereas differential equation models for uncontrolled prey/predator systems take also interactions between the populations into account and are typically of the form

$$\begin{aligned} \dot{x} &= x(\varepsilon - \mu x - \alpha y) \\ \dot{y} &= y(-\delta - \nu x + \beta x) \end{aligned} \tag{3}$$

This approach requires thinking in terms of cause-effect relationships, focusing on what often are - somewhat misleading - called 'feedback' linkages and should better be explained as a search for circular chains (or causal loops) among components of a system within the influence diagram. Consideration of *'cause-effect'* interactions is a rather general and important modeling principle. They are best indicated by means of either graph-theoretic or matrix tabular displays.

Many systems which need to be controlled are only partially known or, their parameters are difficult to identify. Therefore, models which describe the system's behavior qualitatively have gained interest. Especially approaches have become popular which are based on the concept of a 'fuzzy system'. This notion was introduced by Zadeh and then disregarded for some time. Gradually, it became attractive especially, the successful use of the first fuzzy controllers lead to a real boom of this concept. Meanwhile, fuzzy controls have become (among others) a valuable design tool, especially in combination with what is known as learning (adaptive) modeling and together with neural networks used for the latter.

Port-based modeling is a rather old concept for structural modeling. Bond graphs are especially well suited for modeling of multidisciplinary dynamic systems. Originally restricted to linear systems their applicability has been meanwhile enlarged to nonlinear systems. Moreover, their use for sensitivity/robustness investigations or, for the construction of reduced order observers and mathematical foundations are meanwhile established. Using bond graphs as tool for representation of systems allows not only for a systematic modeling approach - which is of great help for engineers with little practice in modeling , but provides also a means for computer-supported modeling (see *Bond Graphs*).

5.2. Verification and Validation

One of the fundamental issues when a new system has been created or when essential

changes in an existing one have been made is credibility of the results i.e. a 'proof' that everything will work as requested and promised during the system's whole life cycle. One instrument to demonstrate usefulness and quality of a design is simulation. Therefore, also the question of model credibility (not only for the designer but also for the customer or user) has central importance. The ultimate test of a model is how well it performs when it is applied to the problems it was designed to handle (Bender accentuates this with the remark *'You cannot reasonably criticize a geological map if a major highway is not marked on it; however, this would be a serious deficiency in a road map.'*). Nevertheless, it would be desirable not to wait until this very last moment. Therefore, at least some rules and tests for verification and validation of models should be applied right at an earlier stage of the investigations.

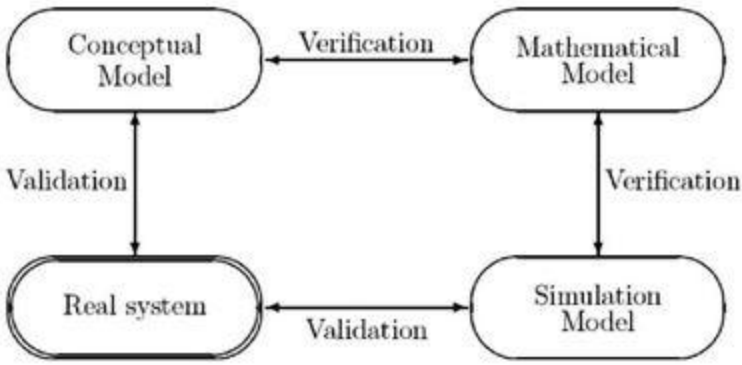

Figure 3: Validation and verification

These two technical terms are often used like synonyms or, even more confusing, with a meaning opposite to the one which is now common use. *'Validation'* is a test for correctness with respect to the real system, whereas *'verification'* tests whether one model is a correct image of another model no matter of which nature (physical, verbal, mathematical, algorithmic, computer code etc.) and type (causal-loop diagram, Petri net, systems dynamics influence diagram, differential equations and inequalities, describing function, frequency domain description, computer flow diagram, etc.) the two models are. An alternative characterization reads as follows: Verification tests whether model development was correctly done, validation tests whether the correct (right) model was developed. Validation is based on appropriately designed experiments; verification is a topic being extensively discussed in the framework of software engineering.

A first step in establishing validity of a model is a check of physical dimensions. Moreover, mathematics itself must be self-consistent and obey the laws of mathematical logic. Further steps concern

(1) *Replicative validity:* model can reproduce known system input/output behavior
(2) *Predictive validity:* model can predict system behavior, i.e. behavior not previously observed or, not used during the whole modeling process
(3) *Structural validity:* the components of the model are directly related to corresponding system components and interact in the same manner.

Extrapolation is the other side of the coin: Given the behavior of the model, what can be said about the real system? Can reliable inferences be made? This question is impossible to answer precisely. If feasible, one has to compare the outputs of the model with the ones of the real system driven by corresponding inputs. However, to avoid self-fulfilling prophecy, input data different from the data used to develop and calibrate the model has to be used!

Clearly one must properly specify, at the very beginning of a study, the study's objectives and hence, the conditions (experimental frame) under which the model should be valid. This will not only assure that the right model is developed and that a clear definition of the study's objectives is available, but - hopefully - also that the model's result will be properly interpreted, i.e., within the boundaries for which the model is valid. Further, validation should include validation of the data used to develop the conceptual model, and the data used for parameter identification and state estimation. One must honestly admit that validation is a difficult task which normally can be carried out only to a certain extent. Due to approximations made during the modeling process, we know in advance that the model and the real system will not have identical input/output behavior. The real question is the practical significance of any disparities.

5.3. Numerical Aspects

There is one danger when using a powerful software packages as tool for problem solving be it at analysis problem such as stability or robustness of a controlled system or, be it a design problem such as the computation of an observer or a controller by a suitably chosen mathematical method. Extensive computations are performed and - even if the mathematical method used is an exact one - errors are inevitable due to the necessary rounding operations. There are certain rules on how to write a program in order to keep these effects small. However, one seldom really knows what happens during calculations or, one did not read carefully enough the description of the algorithm. An example for the latter is the straightforward use of optimization routines by providing only the formulas or algorithms for the calculation of the cost function but without providing corresponding formulas or algorithms for the computation of its gradient or Hessian or, in the case of multi-criteria optimization of the Jacobian. Then, in many cases numerical differentiation will be performed by the optimization tool. The latter is based on difference quotients and hence, extremely sensitive to small inaccuracies. As a consequence, the optimization may lead to a failure of - even worse because unrecognized - may result in a solution far from the optimal one. The same reflections hold true for the calculation of zeros of nonlinear functions, especially of vector-valued ones because internally these zeros often are calculated as minima of an appropriate cost functional.

Modern control engineering requires often involved and accurate computations. In most cases, problems with insufficient numerical accuracy are due to the influence of rounding errors. Therefore, modeling and programming should be such that no ill-conditioned numerical calculations appear during computations for design or analysis, respectively. It is generally accepted that in most situation accuracy provided by simulation programs (e.g. about 16-17 Digits for Matlab) will be sufficient for most practical applications. However, one never can exclude that there may occur a certain specific combination of numbers which leads to totally wrong results. Take as an example the 'simplest' case of a

linear time-invariant system. Design by computation of the stabilizing solution of the algebraic Riccati equation or, by pole placement (requiring the computation of eigenvalues of a matrix) or, by using singular value decomposition are well-known examples. Ludyk has investigated cases such as these in more detail and gives among others an example for controller design via pole placement by state feedback. In this example (with an ill-conditioned matrix) the resulting feedback system (computed by Matlab using standard procedures) has instead of the two real eigenvalues -1 and -2 the conjugate complex pair $-1.5 \pm 670.5i$. The designers and programmers of Matlab are aware of problems due to e.g. ill-conditioned matrices (as documented in the help section on 'reliable computations'). It is the user who is responsible for appropriate and safe use of the various software tools! This is not only a problem of computation (which in principle but with far less comfort can be overcome for many algorithms by using interval arithmetic but, is a problem closely related to modeling especially, to modeling of the task. Needless to say that similar effects occur in time-dependent and/or nonlinear systems. However, in the latter case it is far more difficult (if at all possible) to determine the numerical condition of certain computations to be performed or, their sensitivity with respect to certain problem parameters. Consequently, the properties of the plant - and their influence on certain numerical calculations - should be taken into consideration when deciding which design tool is to be used and ergo, how constraints and/or goals are to be formulated.

But not only design may depend on appropriate modeling and computation. When the investigations concern analysis such as stability properties, existence of chaos, sensitivity of the system's behavior with respect to certain parameters, performance of a newly designed controller etc., then simulation based on numerical calculations can indicate how the real system will behave. However, simulation cannot replace a proof based on mathematical tools. There can always exist one or several situations which - for what so ever reason - either did not occur during the simulation or, where the user did not recognize that the computations were not correct and hence, misleading.

The above problems result at least to a great extent from rounding errors. Accumulation of rounding errors is to a great extent unpredictable and depends on concrete numbers and on the concrete sequence in which elementary operations are carried out. In order to make the above consideration more easily to understand, a simple example (which is a modification of one Ludyk presented in a lecture) shall be discussed shortly.

Example: Consider the polynomial in two variables

$$f = f_0 = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 \tag{4}$$

which can be written also as

$$f = f_1 = \{[(5.5y^2 + 333.75 - x^2)y^2 - 121x^2]y^2 + 11x^4\}y^2 - 2x^2 \tag{5}$$

or

$$f = f_2 = [11y^2x^2 - (121y^4 + y^6 + 2)]x^2 + (5.5y^2 + 333.75)y^6 \tag{6}$$

or - using no floating point numbers but integers only -

$$f = f_3 = (1335/4)y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + (11/2)y^8 \qquad (7)$$

It is to be evaluated at the point $x_0 = 77617.0$, $y_0 = 33096.0$. Floating point computation with Matlab (see Section 6 for more details see *Software Development and Trends*) with about 16 digits cannot compute the correct (integer!) value $f = -2$, but yields $f_0 = -1.180591620717411 * 10^{21}$, $f_1 = -1.204879737800000 * 10^{10}$ and $f_2 = -1.180591620717411 * 10^{21}$, respectively. Maple (see Section 6) allows for a user-defined accuracy by specifying the number of decimal digits which are to be used in all numerical computations. This allows for repeated computation as often recommended for an experiment-based a posteriori evaluation of the influence of rounding errors. Using accuracies between 16 and 37 Digits yields the results outlined in Table 1. This demonstrates that - due to the ill-conditioning of that polynomial near the point (x_0, y_0) - especially the representations f_0 and f_2 can be calculated with correct sign and correct order of magnitude only when using at least 37 digits for the calculations (without rounding to less digits between the various individual elementary operations). The Horner representation f_1 is better conditioned but still requires at least 21 Digits for correct computation of $f = f_1$

Digits	16	17	20	21	28	30	35	36	37
f_0	$1*10^{21}$	$-5*10^{20}$	$-1*10^{17}$	$-1*10^{16}$	$-1*10^{9}$	$1*10^{9}$	-300	20	-2
f_1	$-1*10^{10}$	$-1*10^{10}$	$-1*10^{9}$	-2	-2	-2	-2	-2	-2
f_2	0	$-1*10^{20}$	0	$2*10^{16}$	0	0	0	0	-2

Table 1: Floating point computation results by Maple with various numbers of digits.

It is important to note that the representations f_0, f_1 and f_2 of the above polynomial can be programmed such that only integers appear what results e.g. for the form f_0 in f_3. Then, Maple's ability to do exact arithmetic can be used and all possible integer representations result in the correct value $f = -2$.

As a conclusion, accuracy and result depend on arithmetic used (floating point or integer) and in the usual case of floating point arithmetic, essentially on the sequence in which necessary elementary operations are carried out. However, the effects reported here may appear always in one or the other way. Moreover, one has to be aware that when carrying out long and involved computations it is normally not possible to program every operation in the optimal way. In view of the involved computations appearing in real world problems where theoretic investigations of the achievable accuracies are to be involved if not impossible, the following remedies are often recommended: Repeated computation with different sequences of computational steps and/or with different word lengths should be performed. If the results coincide within the desired accuracy, they are judged to be correct. However, the latter need not be true and hence, is dangerous as

demonstrated by the above simple example where repeated computations with $28 \leq Digits \leq 36$ always yield $f_2 = 0$. Consequently, such repeated calculations with changing results are helpful as indicator for incorrect computations. But on the other hand, one has always to remember that agreement of results of several computations is no guarantee for correctness!

A proof of a certain quality of the result (e.g. stability, robustness) must always be based on theoretic investigations. Their result should not consists only of a statement such as 'the controlled system has an asymptotically stable equilibrium' but, contain at the same time a statement characterizing types and sizes of disturbances the system will be able to handle automatically. Many of the available theoretic results require involved computations when applied to a realistic problem. However, rather comfortable packages (Maple, Mathematica, see *Software Development and Trends*) for manipulation of formulas are available. Maple for instance offers in addition the possibility to switch directly to Matlab/Simulink and hence, to have comfortable support for both, symbolic and numerical investigations.

5.4. System Structure and Model Structure

It is interesting to note how control theory and, especially the traditional input-state-output view, has influenced the definition of the term *'system'* also in the more general context of systems theory e.g. Spriet and Vansteenkiste who distinguish three main model levels:

(1) Behavioral level: the system is regarded as a black-box with inputs and outputs (I/O-model)
(2) State structure level: the internal behavior of the system is modeled too and states are used to describe the system's history (I/S/O-model)
(3) Composite level (network description): provides a hierarchical or modular concept for large or complex systems. The system consists of several subsystems each of which has a state structure. Hence, every subsystem can be represented by an I/S/O-model which is coupled by the respective outputs and inputs. The system itself has inputs and outputs and its model consists of the subsystem's models together with the description of the couplings.

From this it is clear that any system may be a component of a larger system and consequently must be provided with an interface, i.e., a means to interact with other systems. The interface should represent the potential events which can occur at the system boundary. What goes on at this boundary is determined from inside and from outside i.e. by the system and by its environment. The system itself imposes constraints on the interface by what is called its internal structure. It is this internal structure which must be allowed either to remain abstractly described or to become more concrete by further decomposition.

It is important to remember that decomposition of a system into subsystems need not lead in any case to very detailed models of the various subsystems. It depends on the problem to be solved - which details must be included and which are better omitted. If for instance general guidelines for the design of a large system are to be developed then only those

properties of subsystems are to be included in the model which are of global importance. Detailed modeling of a certain subsystem will be of importance only for decisions concerning this particular subsystem and which will have only little influence on the performance of other subsystems. An easily understood example is the design of a welding line in car industries. There a large number of robots are working. Each of these robots has many degrees of freedom. Their movement must be coordinated and must also be controlled individually. Further hierarchical control levels are coordinated distribution of the tasks among robots of a working cell and between the various cells. Coordination can be based on models providing information on how much time is needed to perform a specific work, but no information is needed on how the individual drives of a certain robot are controlled in order to carry out this work. Nevertheless, when analyzing subsystems or designing controls for them, one has to be careful not to forget the whole — the combination of solutions to (simplified) subproblems need not be a good solution for the overall complex problem.

Structured models of system are of growing interest in control engineering. Recently, graph theoretic models have been presented for linear systems which allow for an easy investigation of system-specific properties (e.g. controllability) and solvability issues (e.g. disturbance rejection, I/O-decoupling) even in case of poor information about the system.

5.5. System Descriptions and Relations between Models

As far as dynamical systems are concerned, a system concept can be based on a set structure which has as elements a time base T, an input set U consisting of functions belonging to a certain function set F_u an internal state set X with state transition function $\varphi : X \times F_u \to X$ having the semi-group property and, an output set Y with output function $\psi : X \to Y$ denoted for short as

$$S = \{T, U, F_u, X, Y, \varphi, \psi\} \tag{8}$$

Again, the internal state set represents the memory of the system, i.e. the residue of its past history which will affect its present and future response. This is the heart of the modeling of internal structure. It should be clear that the state set is mainly a modeling concept. Nothing in the real system needs to correspond directly to it. A concrete example is the well-known state-space description for continuous-time lumped-parameter control systems:

$$S = \{[t_0, t_1], \Re^m, F_u, \Re^n, \Re^s, \varphi, \psi\} \tag{9}$$

with $F_u = \{u : [t_0, t_1] \to \Re^m$, bounded and piecewise continuous$\}$ and with φ solution of $\dot{x} = f(x,u)$ with initial condition $x(t_0) = x_0$ where $u \in F_u$ is given and $f(.,.)$ satisfies a Lipschitz-condition, and ψ defines the output $y = \psi(x)$.

Even for one and the same modeling goal there are any models of a system serving the purpose. Differences between models may be due to different choices e.g. of state

variables or, to simplifying assumptions during the modeling process or, to different choices of system descriptions. Consequently, an interesting and important question concerns relations between models or, more precisely, between system representations. Since systems have both internal structure and external behavior, these are two fundamental levels at which models (and systems) can be related.

At the behavioral level, the basic relation is that of equivalence. Two system descriptions, S_1 and S_2 are *'behaviorally equivalent'* if they have exactly the same input/output relation (and consequently, also the same numbers of inputs and outputs respectively). Consequently, any observer (or the environment) can not distinguish between them.

At the *'structural level'* one has two cases that must be distinguished depending on whether or not the two models have the same number of states. The appropriate connections are *'homomorphism'* and *'isomorphism'*. The first is of importance whenever simplification (and especially order reduction) of models is concerned. The second notion concerns models of the same state dimension but with different choices of states. Without going in to much detail, these important concepts can be stated as follows:

Two system representations S_1 and S_2 of the above form are given which have the same T, U, F_u and Y sets, but possibly different internal structures X_1, φ_1, ψ_1 and X_2, φ_2, ψ_2 respectively. Then, a homomorphism from S_1 to S_2 is a map h from the state set X_1 of S_1 onto the state set X_2 of S_2, such that the following hold:

(1) Preservation of transition function: $h(\varphi_1(x_0, u)) = \varphi_2(h(x_0), u)$ for all $x_0 \in X$, $u \in F_u$

(2) Preservation of output function: $\psi_1(x_0) = \psi_2(h(x_0))$ for all $x_0 \in X_1$

An isomorphism is a homomorphism in which the map h is a one-one correspondence of the states. It can be shown that if there is a homomorphism from S_1 to S_2 then the systems are behaviorally equivalent. Thus the homomorphic image S_2 can be much simpler (have fewer states) than the system S_1 and yet be indistinguishable from it concerning input/output behavior.

The well-known state-space description of linear continuous-time systems provides an illustrative example: The systems

$$S_1 : \begin{cases} \dot{\mathbf{x}}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_1\mathbf{u} \\ \mathbf{y} = \mathbf{C}_1\mathbf{x}_1 + \mathbf{D}_1\mathbf{u} \end{cases} \quad \text{completely controllable and observable} \tag{10}$$

$$S_2 : \begin{cases} \dot{\mathbf{x}}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_2\mathbf{u} \\ \mathbf{y} = \mathbf{C}_2\mathbf{x}_2 + \mathbf{D}_2\mathbf{u} \end{cases} \quad \dim \mathbf{x}_2 \geq \dim \mathbf{x}_1 \tag{11}$$

with

$$\mathbf{A}_2\mathbf{H} = \mathbf{H}\mathbf{A}_1, \quad \mathbf{B}_2 = \mathbf{H}\mathbf{B}_1, \quad \mathbf{C}_2\mathbf{H} = \mathbf{C}_1, \quad \mathbf{D}_1 = \mathbf{D}_2 = \mathbf{D} \tag{12}$$

for a suitable full-rank matrix $\mathbf{H}$ have the same I/O description (Laplace transform)

$$\begin{aligned}&\mathbf{y}(s) = \mathbf{F}(s)\mathbf{u}(s) = [\mathbf{C}_1(s\mathbf{I} - \mathbf{A}_1)^{-1}\mathbf{B}_1 * \mathbf{D}]\mathbf{u}(s) = \\ &[\mathbf{C}_2(s\mathbf{I} - \mathbf{A}_2)^{-1}\mathbf{B}_2 * \mathbf{D}]\mathbf{u}(s)\end{aligned} \tag{13}$$

and hence, are behaviorally equivalent. Moreover, for a suitable $\mathbf{H}$ of full rank, there exists a homomorphism between S_1 and S_2, because the relation between states is onto. If moreover, $\mathbf{H}$ has full rank and the dimensions of state spaces are equal, $\dim \mathbf{x}_2 = \dim \mathbf{x}_1$ then an isomorphism between S_1 and S_2 exists and hence, also the internal structures are equivalent.

As mentioned already, in control engineering many different models of one and the same system are often used in parallel. One example is given above, others are linear models derived by a special type of linearization or, the relation between a continuous-time model and a discrete-time model derived by sampling - a question which is answered theoretically for linear systems but, the answer can be used only in the time-invariant case whereas the theoretical result for time-varying linear systems is hard, if at all to apply. The question of how to derive a discrete-time model from a continuous-time nonlinear model is without answer. Use of discretization primarily Euler-discretization is rather common but may lead to a sampled-data system with completely different behavior as is demonstrated for the - at a first glance - simple and famous example of logistic growth described mathematically by the ordinary differential equation

$$\dot{x}(t) = a[1 - bx(t)]x(t) \tag{14}$$

and the corresponding Euler-discretization with $x_k = x(t)\big|_{t=kT} = x(kT)$

$$x_{k+1} = [\alpha - \beta x_k]x_k \tag{15}$$

which yields a difference equation with chaotic behavior. Further examples are (linear or nonlinear) models of different order or, different complexity etc. A detailed discussion of some of these aspects is to be found in various places of *Control Systems, Robotics and Automation*. Here, only some general aspects were briefly reviewed in order to indicate which type of investigations may yield results useful for various simplification techniques used in control applications.

A second major theme is that of distributed simulation and its potential to support the coexistence of multiple formalisms in multiple model components. Systems morphisms play a fundamental role - any claim relating systems, models and simulators to each other ultimately must be phrased with an equivalence or morphism of such kinds. Both perfect and approximate morphisms are discussed and applied to model abstraction and system

representation. Especially in the latter vein, we focus on the ability of the DEVS (Discrete Event System Specification) formalism to represent arbitrary systems including those specified in other discrete event and continuous formalisms. The importance of these discussions derives from two sources - the burgeoning use of discrete event approaches in high technology design (e.g. manufacturing control systems, communication, computers, traffic systems) and the HLA (High Level Architecture)-stimulated growth of distributed simulation, for which discrete events match the discreteness of message exchange. Three system specification formalisms are distinguished:

(1) DESS (differential equations)
(2) DTSS (difference equations)
(3) DEVS (discrete event).

For (1) and (2) mathematical representation had preceded their computerized incarnation whereas the reverse is true for (3). Discrete event models were largely prisoners of their simulation language implementations or algorithmic code expressions. Indeed there was a prevalent belief that discrete event "world views" constituted new mutant forms of simulation, unrelated to the traditional mainstream paradigms. Fortunately, that situation has begun to change as the benefits of abstractions in control and design became clear. A variety of discrete event dynamic system formalisms has emerged, examples are Petri nets, min-max algebra, and GSMP (generalized semi-Markov processes); each has its application area, none was developed deliberately as subclass of the systems theory formalism. Thus to include such a formalism into an organized system-theory-based framework requires "embedding" it into DEVS. One formalism can be embedded in another if any system in the first can be simulated by some system in the second. Actually, more than one such relationship, or morphism, may be useful, since there are various levels of structure and behavior at which equivalence of systems could be required.

Considerations such as these lead Zeigler to the following **hierarchy of systems specifications**:

(1) I/O frame: collection of all observed I/O-pairs
(2) I/O relation: description of the I/O-behavior (e.g. by a differential equation) without auxiliary variables, output need not be unique because it may depend on the interior state
(3) I/O function: initial state is added to (2) - when the initial state is known there is a functional relationship between input and output, i.e. the initial state determines the unique response to any input, an example is a differential equation with initial conditions
(4) I/O system: contains in addition to (2) a state transition function; has semigroup property
(5) Structured system: we can specify how the system is composed of interacting cells
(6) Multi-component system: the system is structured into a set of interacting components, each with its own set of states and state transition function. Each component may be influenced by some other components and may influence other components by its state transition function. Furthermore, each

component's state also might be influenced by the input of the system and it may contribute the output of the system. However, in contrast to coupled systems components at this level are not stand-alone system specifications and do not employ their own input and output interface

(7) Network of systems: Individual component systems are coupled by connecting their input and output interfaces in a modular way

In the above hierarchy hierarchical systems and multi-level systems will be found at level (6) only, whereas large-scale systems can appear already at levels (4) and (5), depending on the number and complexity of interacting cells and components.

6. A Short History of Simulation

6.1. Continuous-time Simulation

As mentioned above, continuous-time simulation started as analog computation. Analog computers represent the variables of a given problem by corresponding physical quantities, such as continuously variable voltages or, at the very beginning, as shaft rotations, thus allowing for true continuous variation. These machine variables are made to obey mathematical relations analogous to those of the original problem. Conceptually, analog computation is based on feedback. This made their use easy for engineers and especially for control engineers. Perhaps this is also the explanation for why so many control engineers hesitated to use digital computers for their simulations. In the early 1970's block-oriented programs for digital computations became available which were rather comfortable to use, required no scaling of variables (as analog computers) and had short computation times. But, they were analog-computer-oriented and hence, algebraic loops could appear which could not always be handled automatically by the digital simulation program, a fact to which analog-oriented (control) engineers were not accustomed.

Although analog computers are primarily fitted for the integration of differential equations, they were sometimes used also for other mathematical problems such as solving algebraic equations, linear and nonlinear optimization. As a mathematical instrument, analog computers have, during the 1960 and 1970's, not only been a most powerful tool for engineers, but they have, in return, strongly influenced the engineer's way of thinking. Now, continuous-time simulation is implemented on digital computers as comfortable analysis and design tool. For control engineers Matlab-Simulink with its various control toolboxes and Maple with its link to it, provide easy-to-handle environments for control system design and analysis and, this possibility of symbolic and numerical computation has again changed the control engineer's way of thinking. Switching to purely digital simulation provides certain facility and easiness but, also involves various constraints. Therefore, those who had experienced these early (and very uncomfortable) days of analog computation still are not satisfied with the need to transform every continuous variable - hidden or visible - into a discrete one. The study of the consequences of sampling (important especially when digital control is used for the first time) is not easily performed during an all-digital simulation. From a didactic viewpoint, visualization of parameter influence was somewhat more impressive with analog equipment. However, it is to be assumed that these and other shortcomings will

quite soon have disappeared and, that new demands will come up.

For a long time, computation of the length of a curve or the area within a closed curve in a plane was an extremely difficult task. For purposes such as these, mechanical devices in the form of disk-and-wheel integrators were developed which date back to the early 19th century. They yielded the result with a limited but often sufficient accuracy. Among these devices was Sang's planimeter with an integrating wheel sliding axially, which was presented at an exhibition in 1851. Maxwell was impressed and developed this device further. Making use of two equal spheres he obtained pure rolling in all directions. Also James Thomson sought for a simpler mechanism than Maxwell's to overcome the disadvantage entailed in sliding an integrating wheel axially. He developed a tilted-disk and rolling-ball integrator, where a loose ball is held by gravity between a disk or cone, and a cylinder to which it transmits the rotations of the disk or cone, Figure 4.

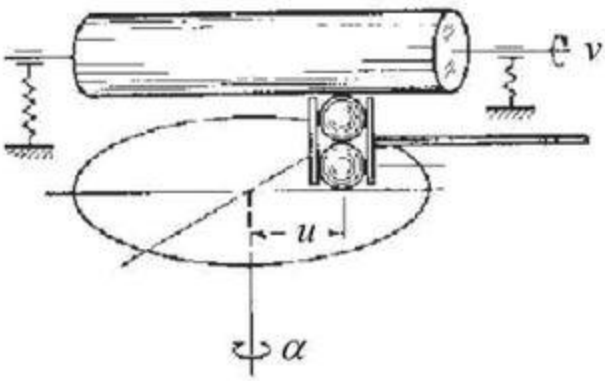

Figure 4: James Thomson's integrator

The possibility of machine solution of ordinary differential equations by successive mechanical integrations was probably first by suggested by W. Thomson (Lord Kelvin). Already in 1876, immediately after the development of a mechanical integrating unit by his brother, William Thomson (Lord Kelvin) suggested a machine solution of ordinary differential equations. He also wished to obtain mechanically the integral of the product of two functions in connection with Fourier analysis. Therefore, the differential equations were allowed to have variable coefficients. He starts with a differential equation of second order written in the form

$$\frac{d}{dx}\left(\frac{1}{P(x)}\frac{du}{dx}\right) = u \tag{16}$$

which can be rewritten as

$$u'(x) = P(x)v(x) \qquad \text{with} \qquad v(x) = C + \int_0^x u(x)dx \tag{17}$$

$$u(x) = \int_0^x P(x)v(x)dx = \int_0^x P(x)\left[C + \int_0^x u(x)dx\right]dx \qquad (18)$$

Hence, $u(x)$ is solution of (16) with initial conditions $u(0) = 0$ and $u'(0) = C$. Therefore, generation of the functions

$$u_{k+1}(x) = \int_0^x P(x)\left[C + \int_0^x u_k(x)dx\right]dx \qquad k = 1, 2, \ldots\ldots \qquad (19)$$

is equivalent to applying the method of successive approximations starting with an arbitrary function $u_1(x)$ (Thomson suggests $u_1(x) = x$). The relations given by (19) can be carried out using two of James Thomson's 'disk-, globe, and cylinder-integrators'. Having done so — i.e. assuming that an iteration would take place which yields a sequence $u_k(x)$ converging to that solution of (16) with $u(0) = 0^-$, he reports *'But then came a pleasing surprise. Compel agreement between the function fed into the double machine and that given out by it. This is to be done by establishing a connection which shall cause the motion of the center of the globe of the first integrator of the double machine to be the same as that of the surface of the second integrator's cylinder. The motion of each will thus be necessarily a solution of (16). Thus I was led to a conclusion which was quite unexpected; and it seems to me very remarkable that the general differential equation of the second order with variable coefficients may be rigorously, continuously and in a single process solved by a machine.'* What Lord Kelvin here describes honestly as a discovery (!), is in modern terminology the feedback principle being the basis of differential analyzers i.e., of programmable analogue computers. A modern block diagram which can be used as basis for the respective program has (omitting inversions of signs in integrators) the form of Figure 5.

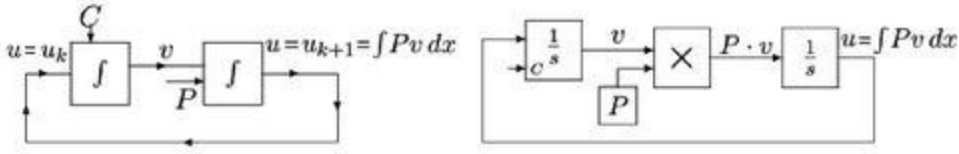

Figure 5: W. Thomson's idea for solving (16) using (19) and the corresponding Matlab-Simulink input

No successful computer of this type was built until the development of torque amplifiers in the middle 1920's. After 1925 a group around Vannevar Bush at MIT, without knowledge of Thomson's original suggestion, constructed the first general purpose mechanical analog differential analyzer, others followed in the US and in Europe, allowing solution of differential equations of orders between 4 and 10. The core of these machines was a number of wheel-and-disk type integrators used for integration, multiplication, and function generation. These mechanical computers were relatively slow, very expensive and - for this time - of high accuracy (better than 0.1 %). In 1942 V.Bush and S.H.Caldwell replaced the awkward mechanical linkage of the integrators by

'easy-to-program' electric connections between its 18 integrators. During World War II differential analyzers were used as fire control computers, as aircraft autopilots, and as navigational computers designed for all kinds of vehicles. Development of electronic computing devices started also at this time; in 1938 Phillbrick und Lovell used the first electronic amplifier. About 1948, electronic differential analyzers - as they were called now - became commercially available. Requirements of the aircraft industries and competition between several computer manufacturers gave a sudden impetus to a systematic investigation of the new equipment to a wide range of problems. Finally, in 1953 also multiplication of functions could be performed electrically. From that time on, differential analyzers working on a mechanical basis disappeared; the programmable electronic analog computer became the more comfortable device - although in general less accurate than the mechanical ones. Parallel to this development of *'general-purpose'* analog computers, special electric analogs (primarily RC-networks), were systematically designed for the solution of field problems described by partial differential equations. These special devices were also termed 'direct analog computer'.

In comparison with digital computers of that time, electronic analog computers offered the only possibility for quick solution of differential equations and hence, of control engineering problems. However, programming was time-consuming because the various computing elements had to be connected with patching cords by hand and consequently, programs could not be stored. Moreover, in order to achieve a minimum accuracy scaling of all variables (including time) was indispensable in order to use the full range without creating overflow due to saturation of electronic devices. A further drawback was the limited possibilities for function generation and especially, the lack of a possibility for decision making. These difficulties were in part overcome by programming aids which were to be run on a digital computer - the first was (proposed 1962, ready 1965) APACHE (Analog Programming And CHEcking) developed for setting a popular particular analog computer. Other aids were removable patch-boards (from 1963 on available), and addition of primitive digital elements (flip-flops etc.), which changed the name to hybrid computer. The 1960's saw also the first hybrid computing systems consisting of an analog and a digital computer and an interface for exchange and transformations of numbers (bit sequences had to be transformed to electric voltages and vice versa) and coordinating control. This development in combination with automatic patch-boards allowed for automatic programming and scaling and for new applications as e.g. the automatic solution of boundary value problems, partial differential equations, integral and dynamic optimization problems. Hybrid computation was a - very expensive - attempt to combine the advantages of both types of computers - the precision and the ability to store data and programs of the digital computer and the high-speed performance of the analog computer.

During all these years, analog and later hybrid computers were widely used by control engineers to solve problems, such as assignment of controller parameters, sensitivity and stability analysis etc. However, comfort and accuracy were low and only small problems could be handled due to limited hardware. On the other hand, control demands increased, plants became larger and more complex. In addition, already in 1955 a digital simulation language that would provide an analog type of programming was proposed by Selfridge. His idea of such a block-oriented language for digital simulation of continuous systems was not used again before 1959. From then on, about 50 different languages of that type

have been developed. Sometimes, they were termed *'digital analog simulation'* in order to indicate their usefulness for solving problems involving differential equations and/or the possibility of block-oriented programming. Simulation languages started as interconnection and control language for a collection of *'analog blocks'* in the form of subroutines. Input could be equation-oriented or block-oriented (often with a graphical surface) and the respective simulation language had to handle sorting of statements and to recognize algebraic loops. The best known became MIDAS (Modified Integration Digital Analog Simulation, 1964), DSL (Digital Simulation Language, 1965), Pactolus (1964), CSMP (Continuous System Modeling Program developed by Brennan and Silberberg, 1966) and DARE (1972). Most compilers translate first into FORTRAN subroutines. This gives the user access to all FORTRAN features (but also to all its limitations). Hence, intermixing with FORTRAN statements and use of FORTRAN libraries are possible. It should be remembered also, that at these early days of simulation, also FORTRAN was considered as a useful tool for simulation. In order to assist control engineers, subroutines were developed and published, which assisted for example for state variable models in the calculation of various transfer functions, in the graphical representation of time responses, in the calculation of sensitivity of closed-loop poles with respect to various parameters, in observability index computation, in the design of Luenberger observer, in the solution of matrix Riccati differential equations, decoupling, etc., and for transfer function representations calculation of frequency response, root locus, partial fraction expansion etc. More details about the early days of simulation languages are to be found in the standard literature.

These efforts eventually led in 1967 to a standard for simulation languages, CSSL (Continuous System Simulation Language, 1965). Among the requirements were

(1) possibility of all-digital simulation and generation of *'check solutions for modern analog/hybrid computers'* (!)
(2) easy use for both engineers unfamiliar with digital computer techniques and for skilled digital application programmers
(3) applicability to all scientific digital computers

At the time of publication, CSSL was primarily a communication language for dynamic system simulation with little concern for problems of implementation. It was developed further to allow e.g. for vector integration and choice between various integration algorithms. Many languages came up which follow the CSSL standard. Well-known examples are DYNAMO (developed by Pugh, 1973), GASP IV (developed by Pritsker, 1974), ACSL (developed by Mitchel and Gauthier, 1976), Dare-P (developed by Wait and Clark, 1976). The model description must be entered as (FORTRAN-like) assignment statements; simulation itself is mostly performed in batch mode. All these programs offer suitable numerical methods and a macro capability i.e. a possibility to use submodels but only little support for convenient graphical output.

Most of the languages of the 1960's and early 1970's were relatively slow. But, at this time Korn started on-line simulation using standard minicomputers for parallel solution of differential equations. What followed was a digital simulation versus analog computation controversy which ended when in the late 1980's commercial production of analogue and hybrid computers was stopped. In principle, a simulation processor had the following

functions:

(1) translation to an intermediate language,
(2) sorting of statements into a proper order
(3) simulation with
(4) integration as the heart
(5) handling of inputs and outputs
(6) a library of special routines such as e.g. time-delay, function generation, random-number generation etc.

What followed was a new controversy. This time between two groups of simulation programs, compilers and interpreters. In any case it was considered essential that simulators be associated with an interactively running post-processor (Mezencev still in 1992). New languages were developed, as e.g. Dymola (Elmquist, 1978 ff), which were interactive and had more modern user interfaces. In addition, also the theoretical foundations and basic concepts of advanced simulation and modeling methodologies were developed further. A process which is going on, is now concerned with the integration of discrete-event and continuous-time (i.e. of hybrid) system's modeling and simulation. In the late 1970's and early 1980's people involved in the creation of simulation languages became more and more aware that model description and specification of simulation runs are to be separated. Most languages have now this property. A further request was the availability of tools for systems analysis within a simulation language, which is realized e.g. in Simnon (developed by Elmquist, 1977), Matlab (developed by Moler, 1980), Matrix-X (1984) (for more detail see *Software Development and Trends*).

There are also problem-specific programs, each suitable for a restricted class of problems, e.g. for electric circuitry (SPICE, 1975), for mechanical multi body systems (e.g. MESA VERDE, 1985), for multi body systems in descriptor form (MBSSIM, 1999) or for thermal problems (MANIP, 1984). Also software has been developed that supports bond graph modeling, e.g. TUTSIM or CAMP (1985) the latter being a preprocessor for ACSL or more recently.

The end of the 20th century saw great progress in the development of packages for symbolic computation. Fowkes states in this respect that *'For the apprentice and professional alike symbol manipulation packages are a Godsend. At the press of a button Fourier coefficients to any order are calculated, differential equations solved, plots drawn etc. This enables one to direct one's mind to major issues such as the appropriateness of the solution procedure or the relevance of the calculations in the modeling context.'* Parallel to this development, many tools for CACSD (Computer Aided Control System Design) were developed, most of them faded. The late 1990's saw, as stated by Cellier & Rimvall, Matlab-Simulink as THE tool for CACSD. Meanwhile, this has developed further; use of Maple for necessary symbolic manipulations and comfortable switching to Matlab-Simulink from it, offer a powerful tool not only for numerical calculations but also for the necessary theoretical foundations needed for actually proving the performance of a controller.

Research and development are going on in several directions. One area concerns the

integration of more modeling, analysis and design tools in various areas of application and especially in control engineering and automation. Object-oriented features are added. Further directions concern continuing progress in hybridization and especially new modeling support (see *Modeling Languages for Continuous and Discrete Systems*). Especially, re-use of models and submodels is an important topic in a time of growing system complexity. Such developments resulted in concepts such as co-simulation and are carried out not only by software firms but especially at research institutions as e.g. at ETH Zürich (programming language Oberon which calls Matlab), at Politecnico di Milano (MOSES allowing modular modeling in an object-oriented database), Technische Universität München (system which overcomes numerical problems when coupling systems such that each subsystem corresponds a physical unit), or at Universidad NED in Madrid (reusable model libraries). Some of these concepts account also for the need of real-time simulation and rapid prototyping (see *Rapid Prototyping for Model and Controller Implementation*).

6.2. Discrete-event Simulation

Unlike continuous-time simulation, discrete-event simulation was always performed digitally. In principle, such a simulation project has three crucial components:

(1) data gathering, model building, and validation
(2) statistical design and estimation
(3) programming and implementation

However, from the late 1970's on, model development was done frequently in view of the concrete simulation language one wanted to use. Underlying are three concepts, which reflect the three main types of discrete-event systems:

Event scheduling gives the complete description of the steps that occur when an individual event takes place. SIMSCRIPT and GASP are well-known languages based on this concept.

Activity scanning emphasizes a review of all activities in a simulation, to determine which can be begun or terminated, each time when an event occurs. ECSL is an activity based language, but also GPSS has the features of this concept.

Process interaction is based on the progress of an entity through a system, from its arrival event to its departure. GPSS, SIMULA use this approach.

The main difficulties in discrete-event simulation are not the modeling or programming, but the design of experiments in order to get valid simulation results. The difficulties concern tactics such as how to start the simulation, when to begin collecting data, how long to run the simulation, and how to deal with highly correlated output. Further, in simulating systems of a random nature, it is important that a convincing statistical analysis be applied to the output of the simulation. In particular, estimation techniques are needed so that the simulator may address the important tradeoffs between simulation run length and the level of precision in the estimates.

A further difficulty concerns the generation of pseudo-random numbers. Meanwhile, several good generators exist. First, pseudo-random numbers which are equally distributed over the interval [0,1] are generated and later transformed to follow the desired probability distribution. One popular idea is based on using the remainder of divisions: With three natural numbers a, b and m and a y_1, the next 'random' number is created as the remainder as follows

$$y_{i+1} = ay_i + b \quad \text{modulo}(m) \quad \Leftrightarrow \quad ay_i + b = nm + y_{i+1} \quad n \in \mathbb{N}$$

Obviously, a starting value y_0 is needed and determines which sequence results. As only a finite number of computer numbers exists, such a sequence is always a periodic one. Their length and quality depends essentially on the choice of the three constants and of the starting value. For discrete-event simulations, not only one such sequence is needed but several ones which must be 'independent' pseudo-random sequences. And this is indeed a problem, the quality of a discrete-event simulation systems depends essentially on how many approximately independent pseudo-random number sequences can be generated (see *Software Development and Trends*).

Glossary

Analytical Model: A model consisting of a set of equations; for example, a system of equations that represents the movement of a helicopter.

Black Box Model: A model whose inputs, outputs, and performance are known, but whose internal implementation is unknown or irrelevant; for example, a model of a computerized change-return mechanism in a vending machine, in the form of a table that indicates the amount of change to be returned for each amount deposited. Syn: input/output model.

Block diagram: A block diagram is a graphic representation of a dynamic system showing individual basic elements and their interconnections.

Boundary Condition: The values assumed by the variables in a system, model, or simulation when one or more of them is at a limiting value at the edge of the domain of interest. Contrast with: final condition; initial condition.

Closed-Form Solution: A closed-form solution for representing time in dynamic models is a method in which the states or status of resources are described as explicit and computationally tractable functions of time. Thus, the status of a resource at time "t" can be found by evaluating the appropriate function at "t", without having to simulate combat from the start of that combat through time "t".

Computational Model: A model consisting of well-defined procedures that can be executed on a computer; for example, a model of the stock market, in the form of a set of equations and logic rules.

Conceptual Model: A statement of the content and internal representations which are the user's and developer's combined concept of the model. It includes logic and algorithms and explicitly recognizes

assumptions and limitations.

Condition: The values assumed at a given instant by the variables in a system, model, or simulation. See also: boundary condition; final condition; initial condition; state.

Continuous Model: A mathematical or computational model whose output variables change in a continuous manner. It is described mainly by ordinary differential equations (ODEs) or partial differential equations (PDEs). Contrast with: Discrete Model.

Continuous simulation: is simulation with dynamic continuous processes:

Continuous System: A system for which the state variables change continuously with respect to time.

CSSL - Standard: The **C**ontinuous **S**ystem **S**imulation **S**tandard from 1968 standardizes the structure of simulation languages (model frame with features for describing ODEs, experimental frame for analyzing the model).

Dependent Variable: A variable whose value is dependent on the values of one or more independent variables. Contrast with: independent variable.

Deterministic: Pertaining to a process, model, simulation or variable whose outcome, result, or value does not depend upon chance. Contrast with: stochastic.

Deterministic Model: A model in which the results are determined through known relationships among the states and events, and in which a given input will always produce the same output; for example, a model depicting a known chemical reaction. Contrast with: stochastic model.

Differential - algebraic equations: (DAEs) are differential equations extended by algebraic equations for the states of the differential equations. They occur e.g. in mechanical modeling, if constraints have to be observed.

Differential equations: are the classical description for dynamic continuous models. The dependent variables set up the state (vector).

Discrete Model: A mathematical or computational model whose output variables take on only discrete values; that is, in changing from one value to another, they do not take on the intermediate values organization's inventory levels based on varying shipments and receipts. Contrast with: continuous model.

Discrete simulation: is simulation with dynamic discrete processes mainly of processes with discrete events occurring in a stochastic manner.

Discrete System: A system for which the state variables change instantaneously at separated points in time. The systems or processes may be described by event graphs, process notations, state charts, etc.

Dynamic Model: A model of a system in which there is change, such as the occurrence of events over time or the movement of objects through space; for example, a model of a bridge that is subjected to wind.

Endogenous variable: A variable whose value is determined by conditions and events within a given model. Syn: internal variable. Contrast with:

	exogenous variable.
Environment:	The texture or detail of the natural domain, that is terrain relief, weather, day, night, terrain cultural features (such as cities or farmland), sea states, etc.; and the external objects, conditions, and processes that influence the behavior of a system (such as terrain relief, weather, day/night, terrain cultural features, etc.).
Environmental Effect Model:	A numerical model, parametric model, or database for simulating a natural environmental effect on an entity of a simulation exercise, such as a sensor or platform.
Environmental Model:	A numerical model, parametric model, or database designed to produce an accurate and consistent data set for one or more parameters that characterize the state of the natural environment.
Equilibrium:	See: steady state.
Event:	A change of object attribute value, an interaction between objects, an instantiation of a new object, or a deletion of an existing object that is associated with a particular point on the federation time axis. Each event contains a time stamp indicating when it is said to occur.
Exogenous Variable:	A variable whose value is determined by conditions and events external to a given model. Syn: external variable. Contrast with: endogenous variable
Final Condition:	The values assumed by the variables in a system, model, or simulation at the completion of some specified duration of time. Contrast with: boundary condition; initial condition.
Final State:	The values assumed by the state variables of a system, component, or simulation at the completion of some specified duration of time.
Graphical Model:	A symbolic model whose properties are expressed in diagrams; for example, a decision tree used to express a complex procedure. Contrast with: mathematical model; narrative model; software model; tabular model.
Hybrid simulation:	Simulation, dealing with an as well continuous as well discrete process, is called hybrid simulation.
Identification:	The identification procedure tries to determine unknown parameters in the model description. Usually the model results are compared with measured data of the process, so that the parameters can be tuned.
Implicit model:	We call a model implicit, if the description is an implicit ODE (the derivatives cannot be separated) or a DAE.
Initial Condition:	The values assumed by the variables in a system, model, or simulation at the beginning of some specified duration of time. Contrast with: boundary condition; final condition.
Initial State:	The values assumed by the state variables of a system, component, or simulation at the beginning of some specified duration of time. Contrast with: final state.
Integration algorithms:	In context with simulation, integration algorithms mean ODE solvers.

Model: A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process.

Modeling: is the procedure deriving (mathematical) models for dynamic behavior of the process under investigation.

Parametric Model: A model using parametric equations that may be based on numerical model outputs or fits to semi-empirical data to succinctly describe a particular process, feature, or effect.

Petri Net: An abstract, formal model of information flow, showing static and dynamic properties of a system; i.e., the Petri net is defined by its places, transitions, input function, and output function.

Real-Time: In modeling and simulation, simulated time advances at the same rate as actual time; for example, running the simulation for one second results in the model advancing time by one second. Contrast with: fast time; slow time.

Simulation: Simulation is the third pillar of science (beneath theory and experiment), performing experiments with a model of the process under investigation.

State-event: An event that will occur only if some specific condition has taken place, e.g. when a car is subjected to aquaplaning.

Slow Time: The duration of activities within a simulation in which simulated time advances slower than actual time.

State space: The higher dimensional space in which the dynamics of a system is studied in terms of the trajectory of the state vector

State Transition: A change from one state to another in a system, component, or simulation.

State Variable: A variable that defines one of the characteristics of a system, component, or simulation. The values of all such variables define the state of the system, component, or simulation.

State vector, State: Vector whose elements are the state variables of a dynamic system.

Static Model: A model of a system in which there is no change; for example, a scale model of a bridge, studied for its appearance rather than for its performance under varying loads.

Steady State: A situation in which a model, process, or device exhibits stable behavior independent of time.

Stochastic: Pertaining to a process, model, or variable whose outcome, result, or value depends on chance. Contrast with: deterministic.

Stochastic Model: A model in which the results are determined by using one or more random variables to represent uncertainty about a process or in which a given input will produce an output according to some statistical distribution; for example, a model that estimates the total dollars spent at each of the checkout stations in a supermarket, based on probable number of customers and probable purchase amount of each customer. Syn: probabilistic model. See also: Markov chain model. Contrast with: deterministic model.

Structural Model: A representation of the physical or logical structure of a system;

	for example, a representation of a computer network as a set of boxes connected by communication lines. Contrast with: process model.
Structural Validation:	The process of determining that the modeling and/or simulation assumptions, algorithms, and architecture provide an accurate representation of the composition of the real world as relevant to the intended use of the model and/or simulation.
System:	A collection of components organized to accomplish a specific function or set of functions.
Time-Dependent Event:	An event that occurs at a predetermined point in time or after a predetermined period of time has elapsed. See also: state-event.
Time Variable:	A variable whose value represents simulated time or the state of the simulation clock.
Validation:	After setting up a model, the model's correctness has to be checked. This process of determining the degree to which a model or simulation is an accurate representation of the real-world from the perspective of the intended uses of the model or simulation. It may be performed by comparing with measured data.
Variable:	A quantity or data item whose value can change. See also: dependent variable; independent variable; state variable. Contrast with: constant, parameter.
Verification:	After implementation of the model at a computer, the program has to be checked for correctness. The process of determining that a model or simulation implementation accurately represents the developer's conceptual description and specification. Verification also evaluates the extent to which the model or simulation has been developed using sound and established software engineering techniques.

Bibliography

Allgöwer F. and Zheng A. (Eds) (1991). *Nonlinear Model Predictive Control*, Basel: Birkhäuser. [This book address important theoretic topics, modeling aspects and applications from a control engineers point of view]

Atherton D. P. and Borne P. (Eds) (1992). *Concise Encyclopedia of Modelling and Simulation*, Oxford: Pergamon. [Modeling and simulation are addressed from a somewhat general point of view; however, as not only the editors but also many contributors are control engineers the focus is on aspects relevant for control systems]

Bender E. A. (1978). *An Introduction to Mathematical Modelling*, New York: Wiley. [An early undergraduate textbook with many simple examples]

van den Bosch P. P. J. and van der Klauw A. C. (1994). *Modeling, Identification and Simulation of Dynamical Systems*, Boca Raton: CRC Press. [This textbook on graduate level provides an overview for continuous-time and discrete-time (including stochastic) systems with emphasis topics relevant in control engineering]

Bossel H. (1992). *Modellbildung und Simulation*, Braunschweig: Vieweg. [Widely used textbook on undergraduate level, in German]

Breitenecker F. and Solar D. (1986). *Models, Methods, Experiments - Modern aspects of simulation languages,}* In: Proc. 2nd European Simulation Conference, Antwerpen, pages 195--199, San Diego: SCS.

[Early reflection on what simulation means]

Brennan R.D. and Linebarger R.N. (1964). *A survey of digital simulation: digital analog simulator programs,* In: Simulation, 3(6): 22--36 [Early survey on digital simulation languages, of historical interest]

Cellier F. E. (1991). *Continuous System Modeling,* New York: Springer. [Widely read textbook/monograph on modeling which presents numerous modeling approaches and applications]

Chapman W. L., Bahill A. T. and Wymore A. W. (1992). *Engineering Modeling and Design,* Boca Raton: CRC Press. [Textbook written from a systems engineer's point of view]

Fowkes N. D. and Mahony J. J. (1994). *An Introduction to Mathematical Modelling,* Chichester: Wiley. [This textbook presents a general introduction in modeling, model analysis and problem solving with applications from many different areas]

Garzia R.F. and Garzia M.R. (1990). *Network modeling, simulation, and analysis,* New York: Marcel Dekker. [Easy-to-read textbook which covers also some more general modeling aspects]

Karnopp, D.C. and Margolis D.L. and Rosenberg R. C. (2000) *System Dynamics - Modeling and Simulation of Mechatronic Systems,* New York: Wiley, third edition. [Well-known textbook on modeling of multi-port systems with bond graphs]

Kheir N. A. (Ed.)(1996) *Systems Modeling and Computer Simulation,* New York: Marcel Dekker, Inc., 2nd edition. [Provides a survey on modeling and various types of models and of simulation including simulation languages, primarily from the point of view of (control) engineers]

Korn G. A. and Wait J. V. (1978). *Digital Continuous-System Simulation,* Englewood Cliffs, NJ: Prentice-Hall. [Description of the first digital simulation language, of historical interest only]

McLeod J. (1963). *Simulation is What?* In: Simulation, 1(1): 5--6. [Definition of simulation, of historical interest only]

Monsef Y. (1997). *Frontiers in simulation,* Erlangen: SCS - Society for Computer Simulation. [Textbook on modeling and simulation written primarily for computer scientists]

Murthy D. N. P., Page N. W. and Rodin E. Y. (1990). *Mathematical Modelling -- A Tool for Problem Solving in Engineering, Physical, Biological and Social Sciences,* Oxford: Pergamon Press. [A well readable introduction with many hints to the literature]

Polderman J. W. and Willems Jan C. (1998). *Introduction to mathematical systems theory,* New York: Springer. [Textbook which presents models for control systems which are not based on input/output considerations]

Schwerin R. von (1999). *MultiBody System SIMulation - Numerical methods, algorithms, and software,* Berlin: Springer. [An example for special purpose modeling and simulation software]

Shannon R.E. (1975). *Systems and simulation - the art and science,* Englewood Cliffs: Prentice-Hall, [Covers general modeling and simulation aspects which are still valid]

Soroka W.W. (1954). *Analog methods in computation and simulation,* New York: McGraw-Hill. [At its time a well-known textbook on continuous-time system simulation i.e. on analog computers and analog computation which provides also rather detailed information about the history of analog computation]

Spriet J. A. and Vansteenkiste G. C. (1982). *Computer-aided modelling and simulation,* London: Academic Press. [Presents theoretic and application oriented aspects of modeling and simulation together with a careful discussion of model equivalence, provides further a link between the engineer's point of view and the one of computer science.]

Thomson W. (1875-1876). *Mechanical integration of the general linear differential equation of any order with variable coefficients,* In: Proc. Roy. Soc. (London)} 24: 269-271. [Describes the discovery of the analog computer, of historical interest]

West B. J. (1985). *An Essay on the Importance of Being Nonlinear,* Berlin: Springer. [This book provides some reflections in a general setting]

Zeigler B. P., Praehofer H. and Kim T.G. (2000). *Theory of Modeling and Simulation - Integrating discrete event and continuous complex dynamic systems.* San Diego: Academic Press, second edition. [This book

Zeigler B. P., Praehofer H. and Kim T.G. (2000). *Theory of Modeling and Simulation - Integrating discrete event and continuous complex dynamic systems*. San Diego: Academic Press, second edition. [This book addresses these topics from computer science point of view]

Biographical Sketches

Inge Troch graduated (Dr.techn.) with a thesis in mathematics of control at the Vienna University of Technology. She is University Professor since 1974 and initiated courses and scientific work in Mathematics of Control, Modeling and Simulation at TU-Vienna. Her teaching comprises courses in these areas as well as on basic mathematics for engineering students and on differential equations at TU-Vienna, courses in robotics at the Universities of Linz and Bologna and for the Scientific Academy of Lower Austria at Krems. At present she is head of the Institute for Analysis and Scientific Computing at TU-Vienna.

She is Austrian delegate in IFAC Technical Committees (TCs) 'Optimal Control' and 'Linear Systems' and was chairperson for a four year term (and is still active member) of the VDI/VDE-GMA Committee on 'Modeling and Simulation in Automation' in Germany. She is chairperson of the IMACS-TC on 'Mathematical Modeling' and organizes successfully a series of triennial conferences on Mathematical Modeling (MATHMOD Vienna). She is senior member of IEEE.

Inge Troch is Editor-in-Chief of the *journal Mathematical and Computer Modeling of Dynamical Systems* and a member of the international editorial board of *Mathematics and Computers in Simulation* (MATCOM), *Systems Analysis, Modelling and Simulation*, *J. Intelligent and Robotic Systems* (JIRS) and *Surveys on Mathematics in Industry*). She was member of the editorial board of C-TAT (*Control -- Theory and Advanced Technology*), and was/is also a member of the IPC and/or session organizer of some 60 international symposia and congresses.

Inge Troch is a co-author of two books, co-editor of nine Proceedings, editor of several special issues of scientific journals, author or co-author of about 120 articles in scientific journals and books in the fields of mathematics of control (stability, systems theory, optimization), modeling, simulation and robotics.

Felix Breitenecker studied "Technical Mathematics" at Vienna University of Technology (VUT), finishing with Master of Technical Sciences (Dipl.-Ing.) in 1976 and graduated (Dr.techn.) with a thesis in Mathematics of Control in 1979. From 1976 on he worked as research assistant at the VUT Mathematics of Control and Simulation Group and from 1984 on, as associate professor at VUT for "Simulation and Mathematics of Control" engaged in teaching and RD in modeling and simulation. He was guest professor at University Glasgow, at Univ. Clausthal-Zellerfeld, Univ. Ljubljana, and Univ. Linz.

He is active in various modeling and simulation societies: president and past president of EUROSIM since 1992, board member and president of the German Simulation Society ASIM, member of INFORMS, SCS, UKSIM, etc. In 2004 he has been elected into the executive board of GI, the German Gesellschaft für Informatik. In 2001 he received as first European, the Distinguished Service Award of INFORMS, from the OR Society of USA.

He has organized and co-organized the European Simulation Congress Vienna (1995), ASIM conferences in Vienna, the conference series MATHMOD in Vienna, and Simulation Workshops in UK, Germany and Austria.

Felix Breitenecker covers a relatively broad research area, from mathematical modeling to simulator development, from discrete event simulation to symbolic computation, from numerical mathematics to object-oriented simulation implementation, from biomedical and mechanical simulation to workflow and process simulation.

He is involved in various national and international research projects and he is active in industry projects, e.g. with Daimler-Chrysler and EADS. In co-operation with ARCs, the Austrian Research Center Seibersdorf, he takes part in research and industry projects in biomedical engineering and in process engineering.

He has published about 210 scientific publications, and he is author of 2 books and editor of 16 books (Proceedings and Series). Since 1992 he is editing the journal Simulation News Europe, Editor-in-Chief since 1995, and he is co-editor of the SCS Series "Frontiers in Simulation" and "Advances in Simulation".