AUTOMATION
Machine Automation

VTT-AUT3 C-9501

FINAL REPORT
HUMAN CAPITAL AND MOBILITY FELLOWSHIP
ERBCHBICT930534

DESIGN AND SIMULATION OF MECHATRONICAL SYSTEMS

Johan Scholliers

VTT AUTOMATION
Machine Automation

January 1995

VTT AUTOMATION
Machine Automation

Kanslerinkatu 8 G
P.O.BOX 13021
FIN–33101 TAMPERE, Finland

Tel. +358 31 163 3111
Telefax + 358 31 163 3494
Telex 22313 ttktr sf

SUMMARY

This is the final report of the research project “Design and Simulation of Mechatronical Systems”, which has been financed by the Human Capital and Mobility Programme of the Commission of the European Communities (Contract No ERBCHBICT930534). The duration of the project was 18 months (01.08.1993 – 31.01.1995).

In mechatronic systems, which consist of smart motion controllers, actuators and mechanisms, different technologies, like analog and digital electronics, mechanics and hydraulics, interact with each other. Through the integration of the various technologies, a more optimal product can be obtained. By the use of Computer Aided Engineering (CAE) tools throughout the complete design process, better and more efficient products can be designed. At this moment, advanced commercial CAE-tools are available for each technology, but tools that can aid in the design of a multitechnical product are still rare.

The aim of the research project “Design and Simulation of Mechatronic Systems” is to develop a design and simulation environment for mechatronic systems. Two main methods have been developed and tested for simulating mechatronic systems: the integration of different CAE-tools by concurrent simulation, and the extended analog circuit simulation.

In the concurrent simulation method, the system is modelled in various CAE-tools, which interchange data during simulation. The advantage of this method is that each designer can use the most appropriate tool to model subsystems of the multitechnical system. Existing component models and libraries can be used, without requiring any model conversions.

In the second method, the complete system is converted, by applying analogies between the different technologies, to an electric circuit. This allows to use an affordable SPICE-based analog circuit simulator to model and simulate multitechnical systems. The model of the system can be graphically entered in a schematic editor, which is customised to the multitechnical needs.

The analog circuit simulator does however not have all the functions that are desired during the design process. This requires the possibility to exchange system models between the analog circuit simulator and other CAE-tools. During this research project, interfaces have been developed between the analog circuit simulator and the other design tools.

Publications

An article has been presented on the European Simulation Multiconference 1994 (1-3 June 1994, Barcelona) on the concurrent simulation approach: "**Development of an Integrated Environment for the Simulation of Multitechnical Systems**", (In *Proceedings of the Conference on Modelling and Simulation 1994,, Barcelona, 1-3 June 1993*, SCS, pp. 740-744). This article forms the basis of Chapter 2 from this report.

An article will be presented on the 1995 IEEE International Symposium on Circuits and Systems, Seattle, Washington, April 29-May 3,1995: "**A SPICE-based library for Mechatronic Systems**" (J. Scholliers, T. Yli-Pietilä), in the special session on "Circuit Theory Approach to Mechatronics".

An article will be presented on the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, May 21-27: "**Simulation of Mechatronic Systems Using Analog Circuit Simulation Tools**" (J. Scholliers, T. Yli-Pietilä).

An article will be presented on the 16th International Power Conversion and Intelligent Motion Conference (PCIM'95), Nürnberg, Germany, June 20-22, 1995: **"The Modelling of Motion Control Systems with an Analog Circuit Simulator"** (J. Scholliers, T. Yli-Pietilä).

An extended abstract has been submitted to the 9th European Simulation Multiconference 1995, Prague, Czech Republic, June 5-7, 1995: **"Integrating Conceptual and Physical Design Tools for Motion Control Systems"** (J. Scholliers, T. Yli-Pietilä, T. Kivento, P. Yli-Paunu, P. Viitanen).

The work has also been presented at the seminar on "Integroitujen Teho-Ohjaimien Käyttö Koneissa ja Laitteissa" (Use of Integrated Power Control in Machines and Devices), organised at VTT. An internal report was produced (VTT-KAU3 C-9407: "**Design and Simulation of Mechatronical Systems by Extended Analog Circuit Simulation"**), which has been incorporated in this report.

TABLE OF CONTENTS

CHAPTER 1.

INTRODUCTION

1.1. Mechatronic Systems

Mechatronics is the synergetic combination of precision mechanical engineering, electronic control and systems thinking in design of products and manufacturing processes [6].

Motion control systems are the backbone of mechatronic machines and instruments. Motion control systems consist generally of the following elements (Fig. 1) [41]:

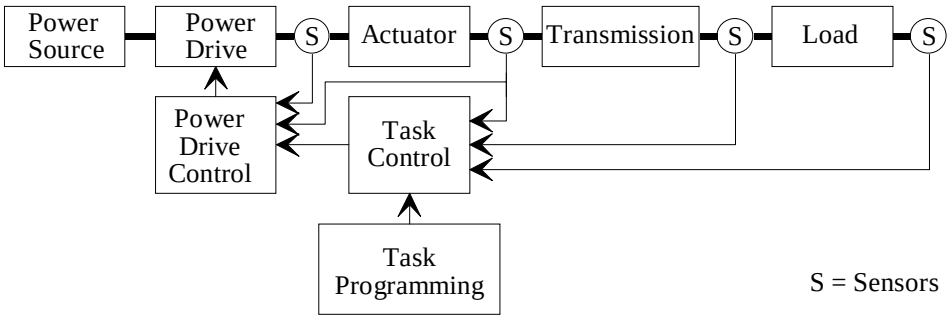


Figure 1: Elements of a generic motion control system.

Power source: Source that supplies the energy, e.g. electric net or battery, hydraulic tank.

Power drive: (or power amplifier) controls the amount of energy retrieved from the power source. Examples of electrical power drives are switch-mode controlled converters and thyristor controlled rectifiers. Valves are hydraulical power drives.

Actuator: converts the electrical or hydraulical energy to mechanical energy.

Transmission: conversion of the mechanical energy, e.g. from rotational to translational, and reduction of the velocity by means of gears, belts, screws, cams, planar or 3-D linkages,...

Load: the system that has to be moved.

Sensors: perform measurements of e.g. position, velocity, pressure, current, and convert it to a signal that can be input to the controller.

Task programming unit: defines the task to be performed by the mechanism.

Task control: calculates the setpoint for the power drive controller, with as input the desired task and the outputs of the sensors. The task control can be implemented in a continuous way (e.g. by the use of analog electronic circuits or hydraulic systems) or digital, by the use of a motion control IC or an external PC or DSP-board.

Power drive control: creates the signals for the switches from the power drive, based on the setpoint delivered by the task control. The power drive controller uses also sensor data, e.g. the output current of the power drive or the actuator velocity. Electrical power drive controllers and power drives are often provided in the same package, which is often delivered together with the actuator.

The different components of mechatronic systems belong to different energy domains (electronics, mechanics, hydraulics,..). The electronic part can consist of both analog and digital electronics.

Designers of mechatronic systems have mostly only a strong background in a few technologies. Due to the lack of knowledge or lack of experience with the other technologies, a wide spectrum of design concepts and design problems can be ignored. The application of Computer Aided Engineering (CAE)-tools, like simulation, throughout the complete design process, can aid the designer to design the product faster and more efficiently, and improve the information flow between different subsystem designers. By the use of simulation, the designer can verify the correct behaviour of the subsystems in the complete mechatronic system, and the influence of design parameters on the behaviour of the complete system.

1.2. Design Levels

Due to the different technologies and domains involved in the design of mechatronic systems, the design process is rather complex. To manage the design of the mechatronic product, the design process is decomposed in several steps: conceptual design, functional design and physical design [20]:

1.2.1. Conceptual Design

The conceptual design level deals with the design of the product architecture, and the selection of the main system components. At this level, rule-based algorithms are used.

An example of such a program is **ServoPlan** [14], a program for the selection of motor drives, developed at VTT Automation/Machine Automation: ServoPlan uses a set of rules for the selection of AC- and DC-motors and reducers, which are retrieved from a database.

1.2.2. Functional Design

The functional design level refines the product architecture, and considers the dynamic interactions among the components. The physical character of the interactions is neglected: interactions are treated as unidirectional dimensionless signals [22]. The control design is generally performed at this level. For the functional design, block oriented diagrams, like MATLAB (from Mathworks Inc.) and XMATH (from Integrated Systems Inc.), are mostly used.

Continuous time systems can be divided in two classes: *lumped parameter models* and *distributed parameter models*. Distributed parameter models are described by partial differential equations (PDE's) - which have to be solved with Finite Element Methods; lumped parameter models by *differential-algebraic equations* (DAE's). Most engineering systems can be treated as lumped parameter models. The general description for a lumped parameter model is:

$$\begin{aligned} \mathbf{0} &= \mathbf{f}\left(\dot{\mathbf{x}}(t),\mathbf{x}(t),\mathbf{u}(t),t\right) \\ \mathbf{y}(t) &= \mathbf{g}\left(\mathbf{x}(t),\mathbf{u}(t),t\right) \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned} \tag{1}$$

with **x** the state variable vector, **u** the input variables vector and **y** the output variables vector, which all are time dependent. The first equation can in most cases be converted to an explicit form, i.e. to a set of *ordinary differential equations* (ODE's):

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t),\mathbf{u}(t),t\right) \tag{2}$$

Differential equations can be solved numerically with integration rules, like Euler, Runge-Kutta and Gear [27]. If the system is *linear*, Eq. (1) can be written under the *state-space form*:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned} \tag{3}$$

If Eq. (1) does not depend on the time derivative $\dot{\mathbf{x}}$, a set of *non-linear algebraic equations* is obtained:

$$\mathbf{F}\left(\mathbf{x}(t),\mathbf{u}(t),t\right)=0 \tag{4}$$

Non-linear algebraic equations are often solved with Newton-Raphson techniques.

In computer-controlled systems, the signals do not vary continuously, but only at equidistant time intervals. Discrete-time models are characterised by sets of *difference equations*:

$$\mathbf{x}_{k+1}=\mathbf{f}\left(\mathbf{x}_k,\mathbf{u}_k,t_k\right) \tag{5}$$

During this research project **MATLAB** [51] has been used for functional design. Block diagrams can be entered graphically with SIMULINK, an extension to MATLAB. Built-in blocks are provided for continuous and discrete linear state space systems and transfer functions, and for linear and non-linear algebraic functions, including look-up tables and if-then-else expressions. Non-linear state space systems (in an explicit form) can be modelled by a set of user-written routines in a higher programming language (C). Some of the properties of MATLAB are given in Table 1.

1.2.3. Physical Design

The physical design level takes the energetic interactions between the various subsystems into account. Most engineering systems can be treated as lumped parameter systems, so that the energy flow between subsystems can be concentrated to a few paths. There are two main methods to model continuous systems at the physical level: as multiports or as bond graphs.

A **multiport** [21] describes a system in function of his energetic interactions with the environment. The multiport exchanges energy with the environment at a (finite number of) ports. To each port a *through* and an *across* variable is attached. A multiport diagram of a system resembles a block diagram. In a block diagram the signals are however unidirectional, whereas in a multiport diagram power can flow in both directions between the multiports. Analog electronic circuits are examples of multiport diagrams: resistors and transistors are multiports, the behaviour of which is completely determined by the voltage (across variable) and the current (through) at the element terminals. The multiport can be constructed from basic multiport elements, like (the equivalence of) resistors, inductors and capacitors, or be defined on a behavioural level, i.e. directly from some mathematical equations.

Bond graphs [3,12,44] are a unified graphical and topological description of the energy interaction, storage and dissipation within a dynamic system. In bond graphs, the energy flow is characterised by a *flow* and *effort* variable. A complex system can be described by a number of bond graph basic elements, which are connected by bonds. A bond is denoted by a stroke, representing the energy flow from one end of the bond to the other end. A bond connecting two bond graph elements has a causal relationship. The construction of bond graphs requires however a high abstraction of the system, often through abstraction of the multiport model [22]. Moreover, there are not many direct bond graph simulators.

Multiports and bond graphs refer to the mathematical representation, and could both be used to model multitechnical systems. In this research project the multiport approach is mainly used during modelling and simulation at the physical level.

Physical level simulators can be divided in two classes: procedural and non-procedural. Procedural simulators allow constructs such as assignment and explicit control statements, non-procedural simulators exclude these. In non-procedural simulators the system is described by a set of primitives, the connections between the primitives and parameter values. The description of the system in a non-procedural simulator is simpler, and does not require a profound programming knowledge as procedural simulators. Due to the user-friendliness of the system entry, non-procedural simulators have become very popular. Procedural simulators are however more flexible, and allow to extend the simulator better to new applications. Graphical pre-processors have been developed for both procedural and non-procedural simulators, and allow to enter the system graphically in a user-friendly way.

characteristic CAE-tool	MATLAB	ADAMS	PSpice
available in VTT-KAU on platform	Sun & PC	Sun[1] [2]	Sun (with ABM)[3] PC
version	4.1	6.1	5.1
graphical input	SIMULINK	AView	schematic editor (e.g. ViewDraw)
textual input	MATLAB script file	data set file	netlist
conversion from graphical input to simulator input	internal	internal	netlister (e.g. SpiceLink)
basic model components	blocks (continuous and discrete state space equations, transfer functions, non-linear elements (e.g. delay, relay, limiter, tables),...)	parts, joints, forces, motions, variables, arrays	resistors, inductors, capacitors, diodes, transistors, switches, transmission lines, voltage and current sources
attached component libraries	none	none	libraries of commercial electronic components
algebraic functions ?	yes	yes	ABM
look-up tables	yes	yes	ABM
transfer function input?	yes	yes	ABM
frequency response table?	*developed during the research project*	no	ABM
state space equations ?	linear	linear & general	no
differential equations ?	no	yes	no
if-then-else expressions?	yes	yes	no
parameters ?	yes	to be modelled as VARIABLE element	yes
frequency response analysis ? frequency response analysis output	yes state space matrix (also transfer function and freq. resp. table)	ADAMS/Linear state space matrix (MATLAB and MATRIXx format)	yes only tabular output
link to MATLAB ?	-	output ADAMS/Linear	*developed during the research project*
parameter sweep	no	no	yes
interactive?	yes	yes	no
start CAE-tool as subprocess ?	yes ("Engine" routines)	no	tool not interactive
neglect screen output ?	Engine output not written to screen	set-up file (mdi) has to be edited	option -d0
add user-written programs?	script files (interpreted) C-programs (MEX)	FORTRAN-routines with predetermined names, which have to be linked to ADAMS code	no (only with Device Equations option)
can external libraries be included?	yes	no	yes

Table 1: Comparison between the available CAE-tools

[1] ADAMS has only been available until 31.12.93

[2] ABM = Analog Behavioural Modelling. From version 6 onwards, ABM is included in the standard version of PSpice.

At this moment, there is a large variety of single technology dedicated simulators, but only a few, mostly expensive, multitechnical simulators, e.g. Saber [26]. Electronic circuits are commonly simulated with circuit analysis programs like SPICE [32] and it's derivates, e.g. PSpice (from Microsim Corp.). Mechanical systems can be simulated with general purpose mechanism analysis programs like ADAMS (from Mechanical Dynamics) and DADS (from Computer Aided Design Software Inc.). During this research project, PSpice and ADAMS have been used.

PSpice [51] is an analog circuit simulator, based on SPICE [32]. PSpice has currently a customer base that is larger than all the other SPICE vendors combined, and a better convergence and performance than most other SPICE-derivates [51]. PSpice is a non-interactive and non-procedural simulator. The electric circuit is input as a text file (netlist). Two versions, one for PC and one for Sun are available at VTT Automation. The Sun version has the analog behavioural option, which allows the use of algebraic functions, look-up tables and Laplace transforms for voltage controlled voltage and current sources. PSpice is delivered with a large library of electronic components.

As graphical pre-processor (or: schematic editor), ViewDraw from Viewlogic has been used. The schematic editor ViewDraw, the netlister SpiceLink (which converts the schematics to a netlist) and PSpice are encapsulated in the framework WORKVIEW PLUS from Viewlogic [55]. Table 1 gives some of the properties of PSpice.

ADAMS [47] is a general purpose mechanism analysis program. The system can be entered as a text-file, or graphically with the pre-processor ADAMS/View [48]. For the definition of the forces and motions, the user can enter algebraic functions, transfer functions, systems of differential equations, state space equations and if-then-else expressions. The user can also write C- or FORTRAN-routines (with fixed names) for the calculation of forces and variables, which have to be linked to ADAMS. From the geometry, the Lagrange equations are derived. ADAMS was only available during the first part of the project (until 31.12.1993). Table 1 gives some of the properties of ADAMS.

1.3. Libraries

For rapid prototyping of mechatronic systems, libraries with models of the various components are needed. In the ideal case, the designer only needs to pick a model from the library and add the model to the system; and after building the complete system model, just press the button to start simulation. Two types of libraries can be developed: parametric component model libraries and the commercial component libraries.

1.3.1. Parametric Component Model Libraries

Parametric component model libraries exist for most CAE-tools. Each CAE-tool has a few built-in primitives that can be called through the built-in graphical user-interface, e.g. capacitors, resistors, sources for analog circuit simulators; parts, joints, forces for mechanical analysis programs, transfer function and algebraic function blocks for block diagram simulators. These libraries contain only components for the technology for which the CAE-tool has been developed. Only the expensive simulator Saber [26] offers multitechnical libraries.

The libraries are defined in a CAE-tool dependent language. Almost no standard languages are currently available to define the models. Standards are currently only used in electronics: VHDL (VLSI Hardware Description Language) is the standard in digital electronics, SPICE the de facto standard in analog electronics. Basic SPICE is however very limited, and most commercial SPICE packages have add functions to SPICE to ease modelling. Some attempts have been provided to model multitechnical components in a system-independent way and hence stimulate model re-use.

A first attempt is **DSblock** (Dynamic System Block) [34], developed at DLR. A DSblock describes an input-output block of a general non-linear dynamic system in a neutral way. DSblocks are mathematically described by explicit ordinary differential equations (ODE), differential-algebraic

equations (DAE) or higher index differential-algebraic equations. The DSblock can be coded in FORTRAN or C. A DSblock can be used in every simulation environment, which supports the calling of user-defined FORTRAN or C routines. Unfortunately, no commercial interface between DSblocks and the CAE-tools used for this project, was available.

Another current development is **VHDL-A** [49]. VHDL-A is an extension to VHDL for analog and mixed-mode analog-digital modelling. Since multitechnical systems can be converted to electronic systems by the energy conservation principle, the standard can also be used to model multitechnical systems. The negotiations for the standardisation of the VHDL-A language are still going on. No HDL-A simulator was however available during this research project.

Another initiative is ICOSYM [23], sponsored by the European Commission in the Tempus program, will use the Internet (World Wide Web) to make models of mechatronic components available for public use.

The transfer of models is also difficult since different parameters are required to describe the same component at different design levels: for example, motor catalogue data like maximum torque and velocity are used for the conceptual design level of electric drives, whereas the physical design level requires e.g. the rotor and stator resistances.

1.3.2. Commercial Component Libraries

Commercial component libraries can be built from the CAE-tools primitives, or be based on parametric models. In the latter case the model contains only a reference to the parametric model with the corresponding set of parameters.

Commercial component libraries are currently only available for a limited number of technologies, e.g. for electronics. Libraries with commercial analog electronic components (transistors, operational amplifiers,...) are available for SPICE. ServoPlan, a program developed at VTT Automation for the conceptual design of servo drives, contains DBase III databases with parameters of DC and AC motors and reducers.

Commercial component libraries can be developed independent from the CAE-tool, e.g. in a commercial database system. These libraries contain for each component the parameters for a parametric model in a library of the CAE-tool. The CAE-tools do normally not have any possibility to access these general libraries during the modelling phase. For each CAE-tool, which would use the commercial component library, an interface program has to be written, which converts the commercial component library to a CAE-tool dependent model library.

1.4. Simulation of Mechatronic Systems

Due to the complexity of the design process of mechatronic systems, a single CAE-tool cannot have all the functions desired during the design process. Moreover, simulators are mostly limited to a single technology, and are not able to handle complete multitechnical systems. In order to integrate the design process, models or simulation data have to be passed between the different CAE-tools that are used during the design process. Three possibilities are possible for passing data between CAE-tools:

Model Transfer When passing from one design level to an other, the complete model has to be transferred between CAE-tools. Unfortunately, no standardised techniques exist currently to transfer non-linear models between CAE-systems.

Concurrent Simulation Physical design level simulators are mostly limited to a single technology. By performing the simulation of the complete mechatronic system concurrently in different CAE-tools, which exchange data during simulation, each subsystem could be modelled in the most appropriate tool - without the need for model conversions.

Tool Extension The energy conservation principle allows to convert a multitechnical system to a single technology. This allows to extend the use of a single technology dedicated tool to multitechnical systems. This method has the advantage that only one CAE-tool is needed to model the complete system, so that installation costs are reduced. .

1.4.1. Model Transfer

A single CAE-tool is not able to handle the complete design process of a mechatronic system. During the design process, models need thus to be passed between different CAE-tools. The requirements at the different design levels are completely different, so that model representation and parameters, as well as the structure of the system model, can be completely different. This makes transfer of models between systems difficult. Moreover, there are practically no standard techniques to transfer models between systems.

At the conceptual design level, catalogue data are used, in order to select the components for steady-state properties, life time and strength. Rules are used for the design. The model is very rudimentary, based on catalogue data.

At the functional design level, block diagrams are used. For the design of the control the models are mostly linearised. The components are described in state-space form or by transfer functions. State-space matrices can be transferred relatively easy between CAE-tools, which allow the definition of state-space equations and the generation of user-written programs.

At the physical design level, the non-linear effects are taken into account. Non-linear systems are more difficult to transfer. The only method is to transfer the topology of the system and the parameters for the component models. This requires that both CAE-tools have libraries with component models. An advantage is that, if the topology of the model is related to the topology of the actual system, the model is less abstract than the state-space form.

Chapter 4 will describe how data can be passed between the different design levels, more specific for the tools that were used during this research project.

1.4.2. Concurrent Simulation

The need to transfer models, in order to simulate the complete system could be avoided by simulating the subsystems concurrently in different CAE-tools. This allows to model each of the subsystems in the most appropriate CAE-tool. The user could use directly already available model libraries and models that have been generated by other persons. Models at different abstraction level, e.g. functional or physical, can easily be combined.

If the global multitechnical system can be subdivided in subsystems that are sequentially linked by infinite impedances, and no feedback exists, the simulation can be performed for each subsystem individually. The results of one subsystem simulation are then entered as input for the next subsystem simulation in another CAE-tool. This requires only that the input of the simulation can be entered as a table which is function of time. ADAMS, MATLAB and PSpice offer interpolating tools for these purposes.

This method cannot be used for feedback systems, or if the impedances between the subsystems are not infinite. If the models are available in different CAE-tools, the simulation has to be performed concurrently in the different tools. A disadvantage of this approach is that the computing time, needed for the simulation of the complete system, increases. This drawback is compensated by the decrease in the time needed to model the complete system.

The main requirement for the integrated simulation environment is that new CAE-tools can be easily integrated into the environment. The topology of the multitechnical system and the relations between the input and outputs of the subsystem models should be described in a user-friendly way. The environment should also impose only minimal requirements to the definition of the input and output ports of the

subsystem models. A computer-integrated environment, called MISE has been developed, and will be described in the Chapter 2.

1.4.3. Tool Extension

The energy conservation principle [3,12,21] allows to convert the complete multitechnical system to a single energy domain, and in this way to extend the use of a single technology dedicated physical level simulator to other technologies. One of the main possibilities is to convert mechanical and hydraulic systems to electronic circuits [46]. A large tradition exists already in this field, as it has been used previously to simulate mechanical systems on analog computers [15].

The user, which is often only expert for a limited set of technologies, does not have to be aware of the mathematical model description of the components. The simulator's design entry and the model description language should be enhanced to meet the multitechnical aspects.

During this research project, the circuit simulator PSpice has been extended towards multitechnical systems. A library has developed for multitechnical devices, which includes electronic, electro-mechanic, mechanic and hydraulic component models. In addition, the library contains a large number of mathematical blocks that allow to model subsystems on a functional level. Component models are available at different accuracy levels. The schematic editor ViewDraw has been customised to allow user-friendly entry of the multitechnical system. The extension of PSpice to multitechnical systems will be discussed more in detail in Chapters 3 to 5. Chapter 3 describes the global structure of the library, Chapter 4 the modelling of multitechnical components, and Chapter 5 discusses some examples.

CHAPTER 2.

INTEGRATED SIMULATION ENVIRONMENT MISE

This chapter describes the development of a computer integrated simulation environment for multitechnical systems. Different CAE-tools work together during simulation and exchange data. The advantage of this approach is that each user can use the most appropriate tool to model the subsystem. Models, which have been developed at different tools, can be combined without conversion

Concurrent simulation exists currently only for mixed-mode simulation of electronic systems, in which the simulation of the analog (continuous) system is performed in one simulator, and another CAE-tool simulates the digital (discrete) system [36]. These mixed-mode simulators have been developed for the two simulators explicitly, and it is not possible to add other simulators.

This chapter will describe how an open integrated simulation environment for mechatronic systems has been developed. The same approach has also been applied by other researchers in other engineering domains, as thermodynamics [8].

2.1. Mathematical Principles

The complete multitechnical system consists of several subsystems, which are modelled in different CAE-tools. The input data for the simulation of the subsystems consist of global input data and output data from other subsystems. The scheme that depicts the data flow between the subsystems is equivalent to a block diagram, in which the blocks correspond to subsystem models.

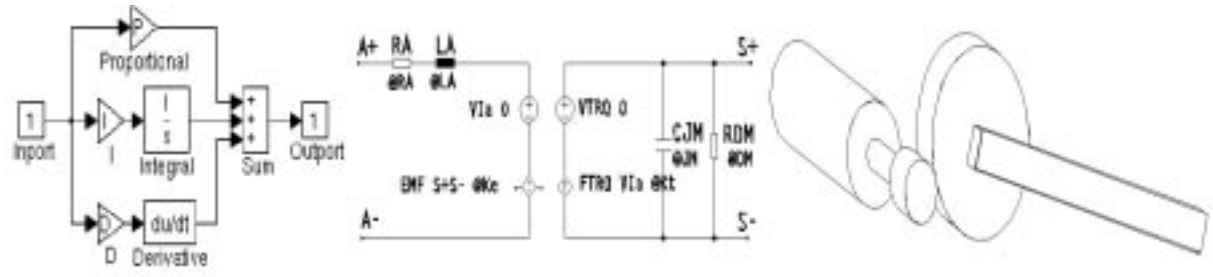


Figure 2: Motion Control System, consisting of a PID-controller (block-diagram model), actuator (modelled as electronic circuit), and a mechanism

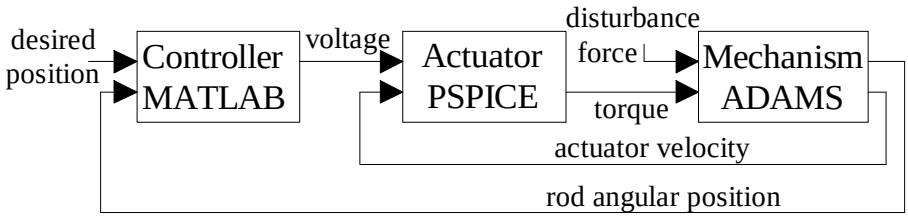


Figure 3: Block diagram of motion control system

Fig. 2 shows, for instance, a motion control system, which consists of a controller, an actuator and a mechanism. The controller is modelled on a functional level in the control design package MATLAB/SIMULINK, the actuator in the circuit simulator PSpice, and the mechanism in the mechanical analysis program ADAMS. The input and output data, which connect the different subsystem models, do not always correspond to physical connections. In physical connections, like the connection actuator-mechanism, two power variables (e.g. velocity and torque) are related to each other. In the block diagram, this relation has to be modelled by an external feedback loop. For the connection actuator-mechanism, one power variable (e.g. the torque) is entered as input for the mechanism

analysis; the other power variable (the velocity) is retrieved as output and is fed back to the input of the actuator simulation. In this way, the block diagram in Fig. 3 is obtained.

The dynamical behaviour of the subsystem *i* can be described by the equations:

$$\begin{aligned} \mathbf{0} &= \mathbf{f}_i\left(\dot{\mathbf{x}}_i, \mathbf{x}_i, \begin{bmatrix}\mathbf{u}_i & \mathbf{v}_i\end{bmatrix}, t\right) \\ \begin{bmatrix}\mathbf{w}_i & \mathbf{y}_i\end{bmatrix} &= \mathbf{g}_i\left(\mathbf{x}_i, \begin{bmatrix}\mathbf{u}_i & \mathbf{v}_i\end{bmatrix}, t\right) \\ \mathbf{x}_i(0) &= \mathbf{x}_{0,i} \end{aligned} \qquad (6)$$

The variables $\mathbf{x}_i$ are the state variables of the subsystem *i*, which equal the initial conditions $\mathbf{x}_{0,i}$ at the start of the simulation. The subsystem inputs consist of global system inputs $\mathbf{u}_i$ and local inputs $\mathbf{v}_i$ (Fig. 4). The subsystem outputs consist of global system outputs $\mathbf{y}_i$ and local outputs $\mathbf{w}_i$. The input data $\mathbf{v}_i$ are formed from the output data $\mathbf{w}_j$ from other subsystems.

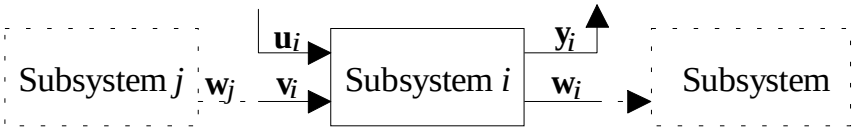


Figure 4: Inputs and outputs of subsystem *i*

The global simulation interval is divided in a set of small time intervals $[t_{k-1}\ t_k]$. The subsystem output data $\mathbf{w}_i$ and $\mathbf{y}_j$ are only retrieved at the end times t_k of these time intervals. Consequently, the input functions $\mathbf{v}_i(t)$ are only an approximation of the actual output functions $\mathbf{w}_j(t)$. Generally, the output function $\mathbf{w}_j(t)$ is approximated by a linear function. If the derivatives of $\mathbf{w}_j(t)$ can also be retrieved as output, a cubic function can also be used for the input function $\mathbf{v}_i(t)$.

If the block diagram contains feedback loops, a system of non linear equations of the form $\mathbf{v}_i(t_k)=\mathbf{h}_k(\mathbf{v}_i(t_k))$ is generated for each the time interval $[t_{k-1}\ t_k]$. The function $\mathbf{h}_k(\mathbf{v}_i)$ is unknown, so that the system of non-linear equations has to be solved iteratively. For the solution of the non-linear equations, a block containing non-linear equation algorithm is inserted into the block diagram at the input of subsystem *i*. The input of this block is $\mathbf{h}_k(\mathbf{v}_{i,n-1})$, with $\mathbf{v}_{i,n-1}$ the input data to subsystem *i* for the previous iteration; the output is the updated value of the input data $\mathbf{v}_{i,n}$ (Fig. 5).

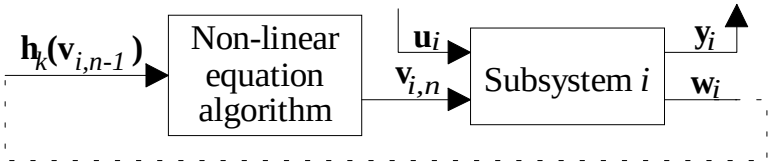


Figure 5: Inputs and outputs of subsystem *i*

2.2. Integrated Simulation Environment

During concurrent simulation, one program puts a signal on an external port, while another program monitors this port for input. During simulation, the various CAE-tools have to exchange data with each other. There are mainly two possibilities to realise this :

Master-slave approach: One of the CAE-tools (the master), e.g. the system with the largest time constant, performs the time management. The subsystem in the slave is modelled as an external function to the master model. A complete simulation run is started at the master. At each simulation step, the master sends a command to the slave to evaluate the slave's output data. At each request of the master, a small simulation run is performed in the slave, with the output of the master system as input, and after this simulation run the output data are sent to the master system.

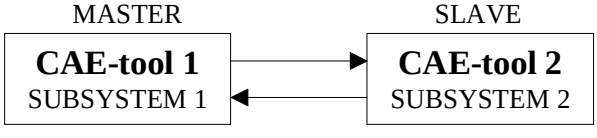


Figure 6: Master-slave configuration

This set-up is very simple, but can in practice only be used if the system is divided over two CAE-tools. The timing depends completely on the master, and can be influenced seriously by the occurrence of a slave.

General manager: A program determines the time step, and the data flow between the various processes. The program must also solve the set of non-linear equations. Since these equations are solved iteratively, the program must also synchronise the sequence of the iterations.

The environment consists of two parts: a general manager, called MISE, which controls the complete simulation, and simulation engines, which are generated for each CAE-tool individually (Fig. 7).

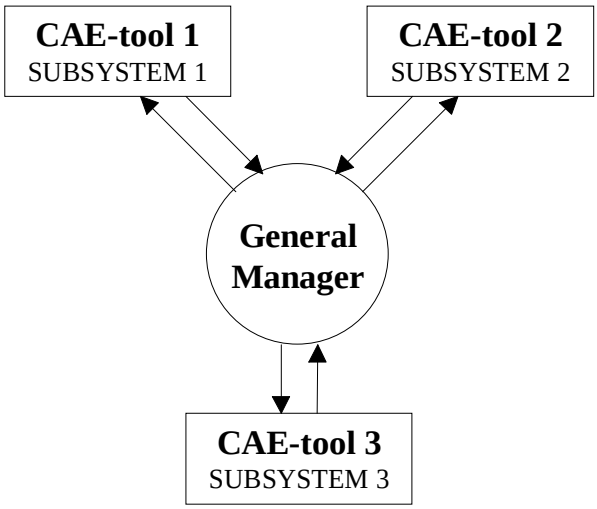


Figure 7: Configuration with general manager

The main tasks of the general manager are the time step management, the control of the information flow between the various CAE-tools, and the generation of output. The general manager divides the global simulation interval in small simulation intervals, generates the input data for the different simulation engines, and interprets the output data.

The simulation engines perform the pre- and post-processing of the simulation in the CAE-tool. Pre-processing includes the conversion of the data from the general manager to input functions for the simulation during the interval $[t_{k-1}\ t_k]$. Post-processing includes retrieval of the output values.

2.3. Communication between Processes in UNIX

The simulation environment has been developed on a SUN-workstation under UNIX. For the communication between processes the Inter Process Communication (IPC) protocol from UNIX is used.

Data between systems are sent over message channels. The message channels are generated by the receiving simulation engine. Per program one receiving channel is generated. The general manager MISE sends thus the messages to the different simulation engines on different message channels, but all the engines send their output data over the same message channel to the general manager.

To make the simulation environment as system-independent as possible, and to reduce the amount of post-processing by the simulation engines, the message sent by MISE (or by the master) contains, except a message header, only floating point data. The messages consist of a message header - which includes e.g. the number of the sending process and a message counter - and floating point data. The messages that are transmitted are :

- initial message, containing the number of input and output signals.
- normal message: floating point data in ASCII-format.
  - the general manager (or the master) sends the end time the end time t_k of the simulation interval and the input data $\mathbf{u}_i(t_k)$ and $\mathbf{v}_i(t_k)$ for the simulation in the CAE-tool..
  - the simulation engine (or the slave) return only the values of the output data $\mathbf{y}_i(t_k)$ and $\mathbf{w}_i(t_k)$.
- "exit": message to denote the end of the simulation. The general manager closes all the open message channels and produces output, to be able to examine the results of the simulation.

To speed up the simulation, the simulation output, which is normal sent to screen, is discarded. This can be hard to realise for some CAE-tools. For ADAMS e.g., an initialisation file had to be edited.

2.4. Master-Slave Configuration

The master-slave configuration is a simplified case of the configuration with general manager, with the master as general manager. The time stepping and the solution of the set of non-linear equations are performed by the master. An external routine must be added to the master, which puts a message on the message channel for the slave, and reads the reply message from the slave. The master CAE-tool must therefore have the possibility to access user-written routines, which are written in a higher level programming language, as FORTRAN or C.

The master-slave configuration has been tested for the configuration ADAMS-MATLAB with ADAMS as master and for MATLAB-PSpice with MATLAB as master. PSpice could not be configured as master, since the current version does not have the possibility to add external routines. The master slave configuration for ADAMS is performing well, but for MATLAB the process becomes extremely slow. This is mainly due to the time step management of MATLAB. Let us, as an example, take a simple PSpice system (a model of a mass), with as input the applied force and as output the position of the mass. For an open-loop system, in which MATLAB sends a sinusoidal signal to PSpice, which returns it output to MATLAB, the time steps are: t=0, 0.2, 0.4,...s. If, to check the accuracy, the transfer function of the PSpice system ($G(s)=1/s^2$) is included in a parallel open loop branch, the first 30 time steps are:

0	0.000125	0.0001875	0.0004615	0.0005	0.00025	0.0005	0.00075
0.000875	0.001423	0.0015	0.001	0.0015	0.002	0.00225	0.003346
0.0035	0.0025	0.0035	0.0045	0.005	0.007192	0.0075	0.0055
0.0075	0.0095	0.0105	0.01488	0.0155	0.0115	0.0155	0.0195

As can be seen from this table, the time step is not monotonically increasing. This imposes additional requirements on the simulation engine, since it does not only have to remember the starting conditions from the last simulation step, but also from the previous time steps.

2.5. Development of the General Manager

The general manager MISE (**M**ultitechnical **I**ntegrated **S**imulation **E**nvironment) is developed in such a way, that it is independent of the CAE-tools.

A syntax has been developed to describe the topology of the system and the data flow between the different subsystems. The description file for the motion control system in Fig. 2 is :

```
CONTROLLER matlab control.m in={des_pos,end_pos} out={voltage}
ACTUATOR PSpice motor.cir in={voltage,velo_motor} out={torque}
MECHANISM adams linkage.adm in={disturb,torque} out={end_pos,velo_motor}
.input des_pos const 0.00
.input disturb sqimp 0 0.1 1
```

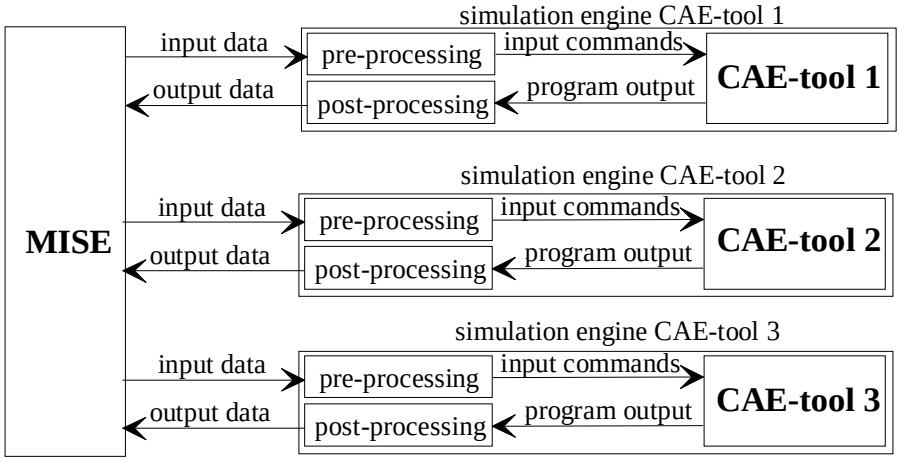


Figure 8: Integrated Simulation Environment

```
.tend 2
.tstep 0.01
```

The subsystems are described by a name (e.g. "CONTROLLER"), the CAE-tool in which the subsystem is modelled (e.g. "matlab"), the file containing the model (e.g. "control.m"), and the input and output data. The name of the simulation engine, which is called by MISE, is derived from the name of the CAE-tool (e.g. Mmatlab for MATLAB and Madams for ADAMS). The description file contains further the input functions to the system (e.g. "const" for a constant and "sqimp" for an impulse of a fixed duration), the global simulation time ("tend") and the constant time step ("tstep").

If the block diagram contains feedback loops, a system of non-linear equations is generated, which is solved iteratively. At start-up, MISE decides where the block, containing the non-linear equation algorithm (Fig. 5) has to be added.

After reading the system description file, MISE starts all the simulation engines as subprocesses, so that only 1 command has to be given to start the complete simulation. If one of the simulation engines is halted (e.g. due to a convergence error), also the complete simulation will end.

MISE divides the different signals in input signals, feed forward signals and feedback signals. Input signals are calculated by MISE through the evaluation of an input function (e.g. sine, step). Output signals do not occur at an input port of a subsystem. Feedforward signals are used as input during the same iteration run n as they have been generated as output. Feedback signals are only used during the next iteration run (n+1). During the first iteration run of each time step on a subsystem, the values of the feedback signals are extrapolated from the previous time steps.

MISE takes a constant time step. The first time steps are made small, to allow for an accurate

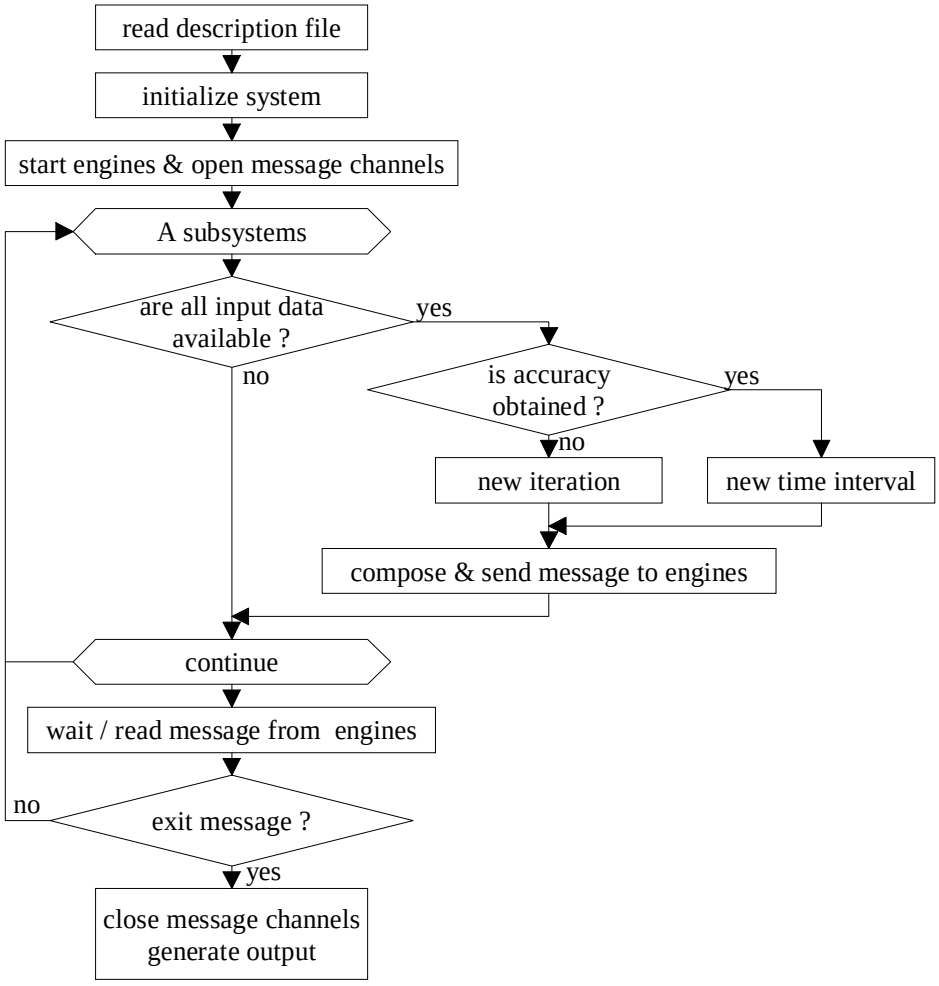


Figure 9: Algorithm MISE

calculation of the state variables at the starting point. MISE checks, for each subsystem, if all the input data, which are needed to perform a new simulation run are available, and if all the output signals have been used for other subsystem calculations (Fig. 9). This means for feedforward signals that their values during the same time step t_k and iteration run n must be available, and for feedback signals, that their values during the same time step t_k and the previous iteration run n-1 must be available. If all the data are available, MISE checks if the demanded relative accuracy is reached. The accuracy ε is defined as:

$$\varepsilon = \max_j \left[\frac{\left| v_{j,k,n} - v_{j,k,n-1} \right|}{\min\left(\left| v_{j,k,n-1} - v_{j,n-1} \right|, \left| v_{j,k,n} - x_{j,k-1} \right| \right)} \right] < \varepsilon_{\max} \qquad (7)$$

with $v_{j,k,n}$ the value of the input element v_j, at the k-th time step and at the n-th iteration, and $v_{j,k-1}$ the value of v_j, at the time step k-1. If the accuracy is not reached, a new iteration step is performed. After a maximum number of iterations, the current time step is halved (so, in the output there is an additional time step). If the time step reaches a minimum value, the data are accepted without obtaining the desired accuracy, and a new time step is taken.

The algorithm, which is used to solve the non-linear equations, depends on the number of equations. If the system contains only one equation, a modified regula falsi method is used [43]. The modified regula falsi can however not be extended to systems with more than one degree of freedom. For higher order systems, no methods can be found which assure convergence [43]. Broyden's method, an adapted secant method, is used. The secant method offers however only often slow convergence for some cases with one degree of freedom, like a pure backlash curve, which occurs in many engineering applications. In MISE, a single degree of freedom regula falsi method is used during the first iterations if the error for one variable is much larger than the errors for the other variables.

If no input data are available, MISE waits for a message on the common sending channel from the simulation engines. This method allows to perform calculations concurrently in the different simulation tools.

At the end of the simulation, MISE sends an "exit" message to all the subprocesses, and generates output. As output, an ASCII-file is constructed which contains all the signals between the different processes. A few MATLAB-commands are added at the start and the end of the files, so that the results can be post-processed in MATLAB.

2.6. Development of the Simulation Engines

For each CAE-tool, a simulation engine has to be generated. This engine takes care of, consecutively, the retrieval of the input data, pre-processing, transmission of the simulation command to the CAE-tool, post-processing of the simulation output and export of the output data (Fig. 8). The sequence of these calls is identical for all CAE-tools. A general method has been developed for writing the engine.

The simulation engine is the same for MISE as for the master-slave approach (except that for the master-slave approach the output is generated by the simulation engine instead of by the general manager.

Fig. 10 shows the algorithm for the simulation engine. The program is started up by MISE (or the master). The engine initialises first the model and generates the receiving message channel., and waits then for a message from MISE.

If the multitechnical system contains feedback loops, MISE works iteratively. The engine has to check if MISE has sent data for a new iteration on the old time interval, or for the first iteration on a new time interval. In the latter case, the time, sent by MISE, is larger than the end time of the previous simulation. The values of the state variables at the end of the previous simulation are then the new initial conditions. If a new iteration on the old time interval is performed, the initial conditions, which were used during the previous simulation run, have to be reloaded.

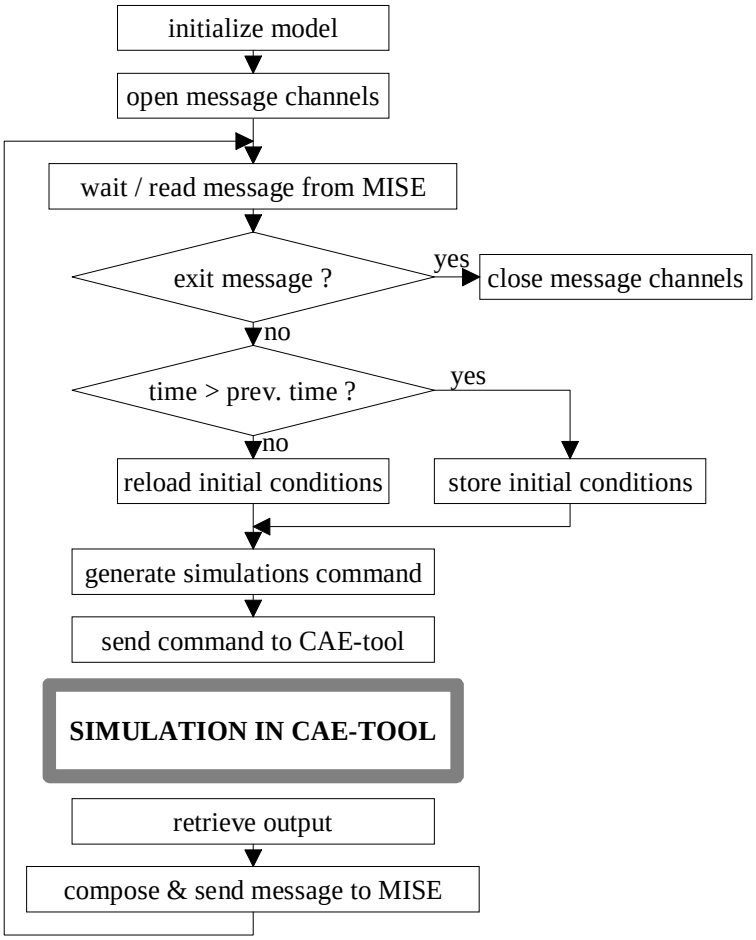


Figure 10: Algorithm simulation engine

The simulation engine has then to generate the input commands for the simulation tool, send them to the CAE-tool, retrieve the output from the CAE-tool, compose the message to be sent, and send the message to MISE (or the master).

In principle, a simulation engine can be generated for each CAE-tool, which can be executed under UNIX. The efficiency of the simulation depends on the two factors: the amount of pre- and post-processing required and the completeness of the system state information.

In the ideal case, i.e. if the CAE-tool is completely open, the simulation engine needs only to transmit the new input functions and (a pointer to) the initial conditions to the integration routine of the CAE-tool; and to send one request to retrieve the output values and the state variables. The CAE-tool is started as a subprocess during the initialisation of the simulation engine, and closed at the end of the complete simulation. Information between the simulation engine and the CAE-tool passes over sockets.

In the worst case, e.g. if the CAE-tool is non-interactive, pre-processing includes the updating of the system's input-file with the new input functions and the initial conditions of the state variables, and post-processing browsing to the output file for retrieval of the output data and the values of the state variables (Fig. 11). The CAE-tool is started as a subprocess at each new iteration step

MISE splits the global time interval up in small simulation intervals $[t_{k-1}\ t_k]$. The global simulation will therefore only be successful if the simulation is not affected by the division of the simulation interval. Therefore, the complete description of the system at the end of the simulation must be accessible. This means that the simulation engine must retrieve all the state variables $\mathbf{x}_i$ at t_k. However, if the subsystem model contains also time delays, the state variables do not determine the state of the system completely: most CAE-tools use (inaccessible) internal buffers to store the values of the delayed signal, and initialise this buffer at the start of each simulation. If the complete state of the system cannot

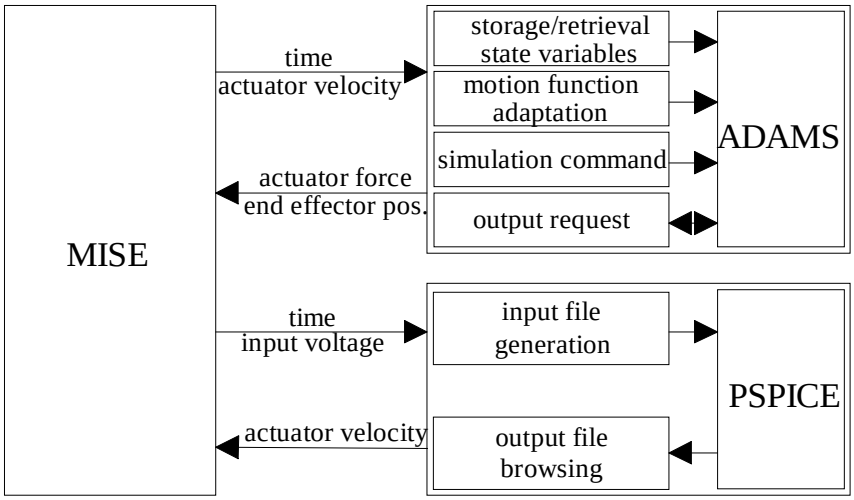


Figure 11: Communication of MISE with the CAE-tools ADAMS and PSpice for a motion control system

be retrieved, the subsystem simulation will always have to begin at the absolute starting time (t=0), resulting in an inefficient use of computing time.

The input and output ports of the subsystem model have to be compatible with MISE, which sends only floating point data to the simulation engines. The engine has to be coded in such a way, that the effort of the user to define and to change the input and output variables is as small as possible. There are mainly two possibilities for the definition of in- and output ports: the names of input- and output variables can be fixed by the simulation engine; or the names can be chosen freely, and written into a file, which is read at the initialisation of the simulation engine.

Engines have currently been developed for PSpice, ADAMS and MATLAB/SIMULINK (Fig. 11). PSpice is a non-interactive program, which demands an input file and produces an output file. For each simulation interval, the input file has to be updated, and the output file has to be scanned for the output variables and the values of the state variables (i.e. node voltages and inductance currents). Before the first use of the PSpice model, a start-up file is generated which contains the names of all the state variables, and the names of the subsystem's input and output variables, which can be freely named.

ADAMS and MATLAB/SIMULINK are more open. For ADAMS, the state of the system at the start of the previous simulation step is reloaded, or the state at the end of the previous step saved, dependent on the time transmitted by MISE; then the parameters of the input function (motion or force) are updated; consecutively a simulation execute command is given; and finally the value of the output variables are retrieved. ADAMS allows only numerals for the input and output ports. The simulation engine requires fixed numbers for the input and output ports. The simulation engine for MATLAB uses a similar procedure.

2.7. Examples

The simulation with MISE has been tested with several examples. Two examples will be discussed: a simple motion control system, with an eccentric rod, an a double pendulum.

2.7.1. Motion Control System

The motion control system consists of a permanent magnet DC-motor, which drives a gear box and an eccentric rod (Fig. 2). For the control, an analog PI position controller is used. The DC-motor is modelled in PSpice, the gear box and the rod in ADAMS. The controller is on a functional level modelled in MATLAB/SIMULINK.

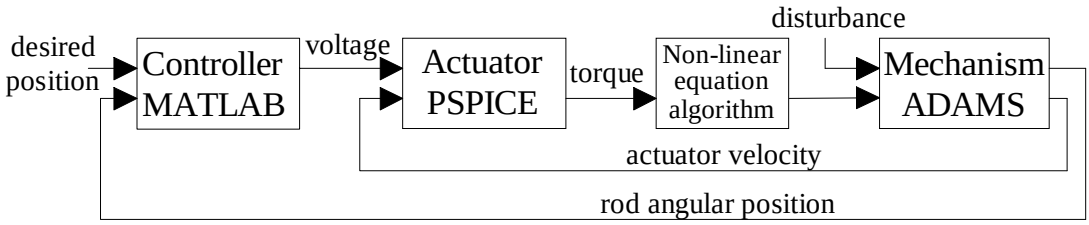


Figure 12: Adapted block diagram of motion control system

The block diagram of the system is shown in Fig. 2 and the system description file for MISE on p. 17. The input for the MATLAB model is the angular position of the rod, and the output the voltage applied to the motor. The inputs for the PSpice model are the input voltage to the motor and the velocity of the motor shaft; the output is the motor torque. The inputs for the ADAMS model are the torque, developed by the motor, and a disturbance (a square pulse of 1 Nm during the first 0.1 s); the outputs are the motor velocity and the angular position of the rod. MISE adds a block to solve of the non-linear equations between the SPICE block and the ADAMS block. At that point, there is only one non-linear equation. The adapted block diagram is shown in Fig. 12.

The accuracy of MISE is checked by simulating the linearised system (i.e. no eccentricity for the centre of gravity of the rod) both in MATLAB/SIMULINK and with MISE. Fig. 13 shows the torque at the output shaft of the motor, and Fig. 14 the angular position of the rod. Both figures show a good resemblance between the results of the program MISE and the results of MATLAB.

The complete non-linear system has then been simulated. The results have been compared with a simulation of a linearised system with MATLAB. The mechanism has been linearised using the built-in linearisation algorithm in ADAMS [38]. Fig. 15 shows the motor torque and Fig. 16 the position of the rod for the non-linear simulation. The differences between the results with MISE and with MATLAB are rather large, due to the friction in the gears, which has not been modelled in MATLAB, and the non-linearity of the rod angle.

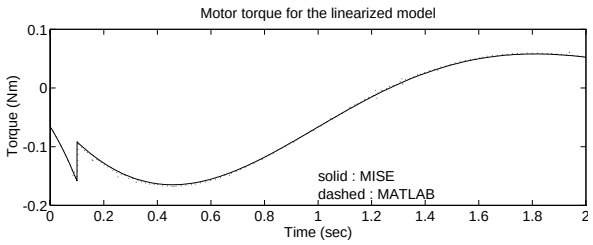


Figure 13: Motor torque for the linearised simulation

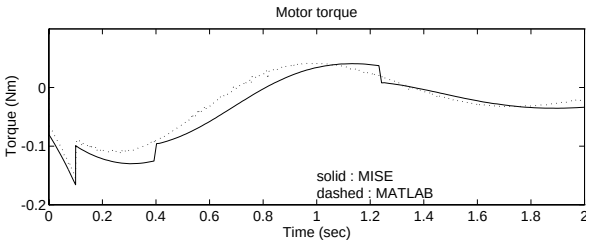


Figure 15: Motor torque for the non-linear simulation

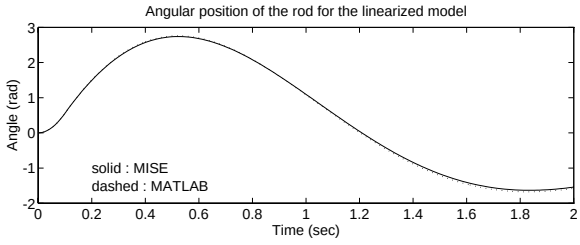


Figure 14: Position of the rod for the linearised simulation

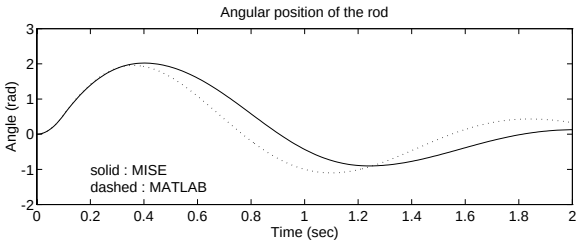


Figure 16: Position of the rod for the non-linear simulation

2.7.2. System with two degrees of freedom

The mechanism with two degrees of freedom is a double pendulum, which consists of two identical rods (Fig. 17). Both rods are driven by a permanent magnet DC-motor and two pairs of spur gears. The motor and the gears of the second rod are mounted on the end of the first rod. The position of both rods is measured with an encoder, and fed back to a PI-controller. The controllers are modelled on a functional level. A disturbance is added at the shaft of the first DC-motor. The desired position corresponds to the starting position, i.e. both rods vertically downwards.

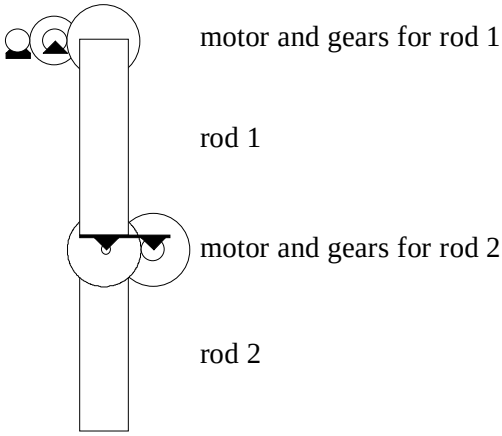


Figure 17: Mechanical system with two degrees of freedom

The system is subdivided in subsystems in the same way as in the previous example. The complete mechanism (i.e. gears and rods) is modelled in ADAMS. Both PI-controllers are modelled in one SIMULINK-model, and both motors in one PSpice netlist. Both systems are described in one file, since only one program licence is available, and to limit the time needed for system initialisation. The block diagram of the double pendulum is the same as the block diagram of the system with one degree of freedom (Fig. 5), except that two variables are passed over each arrow. The description file is now:

```
MOTORS PSpice twomotor in=Vin1,spm1,Vin2,spm2 out=torq1,torq2
PENDUL2 adams  tworod.txt in=torq1,torq2,disturb out=spm1,spm2,pos1,pos2
CONTROL matlab twoctrl in=des_pos1,des_pos2,pos1,pos2 out=Vin1,Vin2

.input des_pos1 const 0
.input des_pos2 const 0
.input disturb sqimp 0 .1 1

* simulation parameters
.tend 2
.tstep 0.0105
.toler 0.0001
.maxiter 20
.minstep 2e-4
```

The results are shown in Fig. 18.

The simulation with MISE has been compared with simulations of the complete system in ADAMS and MATLAB. The controller and the actuator were modelled as transfer functions in ADAMS. The results were comparable. Simulation of the complete (linearised) system in MATLAB was not so successful. MATLAB uses a Newton-Raphson method to solve the set of non-linear equations, and does not always find a solution.

2.7.3. Discussion

Some conclusions could be drawn from the simulation of the examples:

* The execution of MISE is slow compared with the complete simulation in one CAE-tool. This is a.o. due to the large amount of pre- and post-processing for every call to PSpice. The simulation is much more efficient for the CAE-tools that are more open, like ADAMS and MATLAB.

  If the system is modelled in one CAE-tool, the CAE-tool will try to optimise the number of equations. Division of the system over several tools will lead to a non-optimal set of equations, and thus to longer calculation times. This can be noticed by comparing the time needed for the simulation of a motor-inertia combination: if only the motor is modelled in PSpice (and the motor velocity entered as input parameter), a longer processing time is required than if the complete system is modelled in PSpice. Another example is the example that was used to test the time efficiency of MATLAB (p. 17).

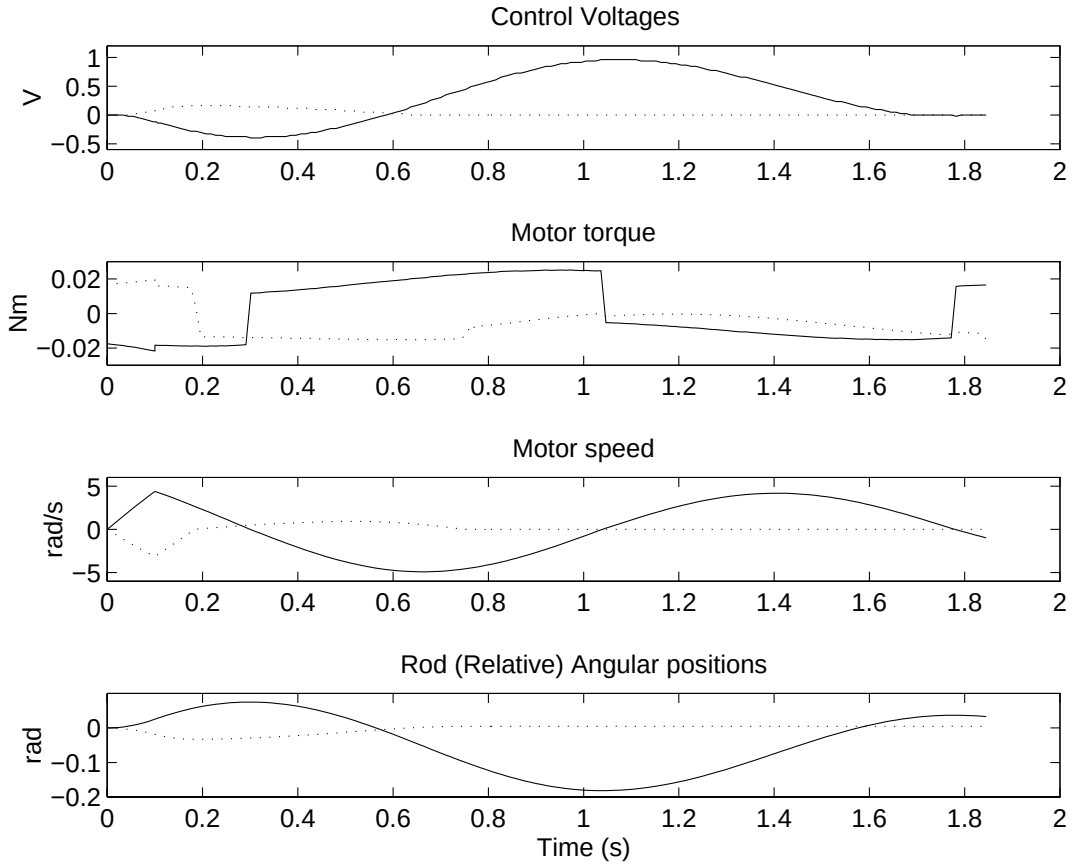


Figure 18: Results of the simulation of the double pendulum with MISE

* The simulation of the double pendulum in MISE was halted before the desired end time (2 s), due to a convergence error during one of the PSpice simulations. This is a classic problem in PSpice and is generally solved by changing the tolerance of the calculations and the maximum number of iterations. During other examples, the program halted due to convergence errors in ADAMS. Also these problems can be solved by trimming simulation parameters. During simulation with MISE the CAE-tools are called many times, and is it more difficult to find a set of simulation parameters for which always convergent solutions can be found. Since the CAE-tools are called very often, MISE is also sensitive to bugs in the various CAE-tools.

2.8. Conclusions

This chapter described the development of an integrated environment for the simulation of multitechnical systems, based on commercial CAE-tools. Different commercial CAE-tools work concurrently during simulation. The environment consists of two parts: the general manager MISE, which controls the information flow between the various CAE-tools, and the simulation engines, which are attached to the CAE-tools. Simulation engines can be generated for each simulation tool, which can work under UNIX.

The main advantage of this environment is that the subsystems of the multitechnical system can be modelled in the most appropriate system, so that cumbersome model conversions are avoided. It also allows to use already available models and model libraries.

The efficiency of the simulation engine depends on the ease of access to the CAE-tool's integration routine, and on the possibility to extract the complete system description. A disadvantage is however that the simulation time is very long, due to the communication between the various processes, and the required pre- and post-processing work of the messages in the various CAE-tools.

CHAPTER 3.

EXTENDED ANALOG CIRCUIT SIMULATION

Due to analogies between the different energy domains, the complete multitechnical system can be modelled and simulated in a single simulation tool. However, at this moment there are no commercial multitechnical CAE-tools that are affordable for SME's. The requirements for an (affordable) multitechnical simulator are:

- allow behavioural modelling, i.e. to describe the phenomena by mathematical expressions rather than by technology dependent primitives.
- allow parametric models, to ease model reuse.
- user-friendly, i.e. allow graphical input. Simulation will only be used in industry, if the effort to enter the model is very small. In the ideal case, the model can be constructed by simply picking objects from a library and defining the connections between the models.
- low price, so that the system is affordable for SME's
- if possible, based on industrial standards.

3.1. Energy Conservation Principle

The total energy rate flow between a component and the environment can, for most engineering systems, be approximated by a finite sum of some complementary physical quantities [21]. One of these quantities represents a **through** variable, the other an **across** variable. The criterion for selecting a physical quantity as either a through or across variable is based on the way it can be identified, e.g. by measurement (cf. Table 2): through variables are measured between adjacent points by first disconnecting them and including the measuring instrument between them, across variables are measured between distant points without disconnecting [21]. The reference of the across variable is related to the energy domain.

	Power variables		Power (P)
	through	across	[W]
Electrical	current [A] i	e.m.f. [V] v	$i*v$
Magnetical	flux rate [$Wb/s \equiv V$] $d\phi/dt$	m.m.f. [A] $\mathcal{F}$	$\mathcal{F}*d\phi/dt$
Mechanical translational	force [N] F	velocity [m/s] $v=dx/dt$	$F*v$
Mechanical rotational	torque [Nm] T	angular velocity [$1/s$] $\omega=d\theta/dt$	$\tau*\omega$
Fluidic	volume flow [m^3/s] Q	pressure [Pa] p	$Q*p$

ϕ = *magnetic flux* x = *position* θ = *angle*

Table 2: Power variables in the different energy domains

The energy conservation principle allows to convert multitechnical systems to a single energy domain. By translating the through and across variables in the different energy domains to the through and across variables of a single energy domain, e.g. voltage and current, multitechnical systems can be modelled as electric circuits, and be simulated with analog circuit simulators.

3.2. Analog Circuit Simulation

Due to the selection criterion for the across and through variables in the various energy domains, there is a complete analogy between mechanical and electrical nodes and connections. The structure of the electric circuit resembles hence the physical structure of the model, which eases the generation of multitechnical models. For example, if a motor is connected to a load, the connection provides an internal feedback. Opposed to block diagram modelling, there is no need in analog circuit modelling to build an external feedback loop to model the effects of the load on the driver. The modelling process can be simplified by providing macromodels for the various components, which can be added to the equivalent electric circuit in the same way as the physical components are added to the system.

3.2.1. SPICE and PSpice

In analog electronics, the de facto standard for simulation is **SPICE** (**S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis). All SPICE-simulators derive from the SPICE2 circuit simulator developed at the University of California, Berkeley, in the mid-1970s. Many derivates (e.g. HSpice, PSpice, ISSpice) are marketed commercially. All of these programs offer a wide variety of analyses, including DC-analysis, transient analysis, AC-analysis,...

In SPICE, the circuit is built from primitives, like resistors, capacitors, inductors, independent and controlled voltage and current sources, diodes, transistors,.... The circuit is entered in a text file, called a netlist. Circuit analysis is based on the application of Kirchhoff's current law, Kirchhoff's voltage law and branch constitutive equations, i.e. the relation between the voltage and current for the electric elements [27]. SPICE converts the circuit elements and their interconnections to system equations by using node equations, expressed in terms of node voltages and Kirchhoff's current law [27]. The system equations are a set of non-linear first-order ordinary differential equations. During transient analysis, a numerical integration method is used to convert the non-linear differential equations into a set of non-linear difference equations, which are solved simultaneously with the Newton-Raphson method. This involves the conversion of the non-linear equations to linear equations and their subsequent solution using a sparse LU decomposition technique [32].

Standard **SPICE2G.6** does not allow behavioural modelling: controlled sources can only have a fixed (non parametrical) linear gain or be a polynomial of a combination of node voltages. This makes the development of parametric and multitechnical models in standard SPICE very difficult. Many SPICE derivates have developed their own methods for behavioural modelling.

PSpice [51] is one of the derivates of SPICE, which has currently a customer base that is larger than all the other SPICE vendors combined, and a better convergence and performance than most other SPICE-derivates. Together with PSpice, an extensive library of electronic components is delivered. PSpice uses a combination of the trapezoidal and the gear integration method. The results of PSpice simulations can be viewed with the post-processor **Probe**.

The **analog behavioural option** of PSpice allows, in addition to the standard SPICE primitives, the use of mathematical expressions look-up tables, transfer functions and frequency response tables in controlled voltage and current sources. With these capabilities it is possible to model complex mechatronic systems in a relatively simple way.

The system can be entered graphically with the schematic editor ViewDraw or by directly writing the system's netlist. The schematic editor **ViewDraw**, and the netlister **SpiceLink**, which converts the schematics to a netlist, are included in the framework **WORKVIEW PLUS** from Viewlogic [55]. WORKVIEW PLUS provides a user-friendly environment for the design and analysis of the system. WORKVIEW PLUS is a CAE framework that conforms to CFI (CAD Framework Initiative) standards. PSpice and Probe can easily be encapsulated in the framework.

3.2.2. PSpice primitives

The basic components in PSpice, which are used to model multitechnical systems, are [51]:

- **resistor** (prefix **R**).
- **capacitor** (prefix **C**). The initial voltage can be set.
- **inductor** (prefix **L**). The initial current can be entered as a parameter in the definition of the inductor. Core material models (including hysteresis) are provided in a library. The magnetic flux density can be accessed in the post-processing program Probe, but can unfortunately not be used in controlled sources in PSpice.
- **voltage source** (prefix **V**). Generates a voltage source in function of time. Possible waveforms are: constant, exponential, pulse, piecewise linear, sinusoidal, frequency-modulated.
- **current source** (prefix **I**). cf. voltage source
- **voltage controlled voltage source** (VCVS) (prefix **E**). Generates a voltage in function of the voltage between other nodes. Basic PSpice allows only a non-parametric linear and polynomial gain. The analog behavioural option allows also algebraic functions, tables, transfer functions and frequency response tables. Parameters can be used, except in the table descriptions.

  The transfer function and frequency response controlled sources have to be treated with care: the precision of these sources depends not only on the accuracy settings, but also on the total simulation time. Simple transfer functions can however easily be replaced by equivalent electric schemes.
- **voltage controlled current source** (VCCS) (prefix **G**). cf. voltage controlled voltage source
- **current controlled voltage source** (CCVS) (prefix **F**). This source generates a voltage in function of the current through a voltage source. Only linear and polynomial gain is allowed.
- **current controlled current source** (CCVS) (prefix **H**). cf. current controlled voltage source
- **voltage controlled switch** (VCSW) (prefix **S**). Generates a switch, which is controlled by a voltage between two controlling nodes. PSpice treats the switch as a voltage-controlled resistance, the value of which changes from high to low over a voltage interval. The switch model easily causes convergence errors, and is therefore as much as possible replaced by tables.
- **current controlled switch** (CCSW) (prefix **W**).
- **transmission line** (prefix **T**). Generates a bi-directional ideal delay line. Distributed models are provided to simulate lossy lines, but these produce extremely long CPU-times.
- **diode** (prefix **D**). Together with the diode, a model must be entered. An ideal diode (infinite flow for positive voltage, no flow for negative voltage) is modelled as [5]:

```
D 1 2 DIDEAL
.MODEL DIDEAL D(N=0.001)
```

  Diodes lead however easily to convergence errors. In the multitechnical library, diodes are replaced as much as possible by tables.
- **transistors** (Bipolar, JFET, MOSFET,...). These electronic devices have no equivalent in other energy domains.

3.2.3. Subcircuits and Parameters in PSpice

In addition to these primitives, PSpice allows also to declare an ensemble of circuit elements as a subcircuit (prefix **X**). Subcircuits can be nested. Parameters can be passed without problems to the subcircuits and to the lower level subcircuits. In the definition of the subcircuit (in the netlist: **.SUBCKT**), the default parameter values are entered. If the parameter does not occur in the call to the

subcircuit, the default parameter value is used. The definition for the a simple amplifier subcircuit (input node 1, output node 2) is:

```
.SUBCKT AMP 1 2 PARAMS: K=1
```

The call in the netlist is:

```
XAMP 1 2 AMP PARAMS: K=100
```

Subcircuits can be stored in a library. In the netlist a reference is made to the library, and the subcircuits that are used in the netlist are retrieved from the library. Next to subcircuits, a library can also contain primitive models (e.g. for diodes, transistors,...).

Global parameters can be defined with the keyword ".PARAM". Parameters can be defined as an algebraic function of previously defined parameters. Parameters can be used in voltage controlled sources (only with analog behavioural option), for model values (e.g. switch control voltages, resistor values), and for initial conditions. Parameters can be swept: the simulation is performed for different values of the parameter, and all simulation results can be loaded and easily compared in the post-processor Probe.

3.3. Mechatronic Library

A hierarchical PSpice library has been generated to extend the use of PSpice to multitechnical devices. The library consists of mathematical elements and electrical, electro-mechanical, mechanical and hydraulic components. Table 3 shows the elements contained in the library.

3.3.1. Hierarchy, Parameters and Accuracy Levels

The library is constructed hierarchically: the component models are constructed from lower level models or phenomena. The basic phenomena for each energy domain are described by the built in PSpice primitives (resistance, inductance, capacitance), or by voltage controlled sources. The models are parametrically.

sources	electrical voltage and current sources (PSpice primitives, net supply) mathematical input function: (step, impulse, ramp, PWL,...)
control	task control : PID power drive control: PWM, hysteretic current control, thyristor firing angle
power drives	electrical : diode and thyristor rectifiers, switch-mode converters, behavioural models MOSFET, BJT, thyristor gate drivers hydraulical : valves
actuators	electro-mechanical : DC-motors (permanent magnet, separately excited, shunt, series, compound), brushless DC and AC motors, induction motors, synchronous motors, stepping motors, switched reluctance-motors, solenoids hydraulical : pump (fixed and variable displacement), hydraulic motor, cylinders
transmission	gears, rack and pinion, belt drives, chain, lead screw, friction drive, brake, clutch, freewheel, cam, cardan, levers, pulleys
sensors	encoder, tachometer, potentiometer, current sensing resistor, accelerometer, synchro, resolver, LVDT, switch
mathematical	Algebraic operations (+,-,*,/) and functions (abs, sqrt, sin,...), derivator, integrator, sample and hold, first and second order transfer functions, second order filters, limit, delay, dead zone, relay, hysteresis, saturation, cubic curve segments, power calculations, statistics, transformers, gyrators, variable impedances
electrical	PSpice primitives and libraries
mechanical	translational: mass, damper, spring, friction, gravity, coupling, contact, backlash rotational: inertia, damper, spring, friction, coupling, shaft, contact, backlash mixed translational/rotational: rot.↔trans., rolling (sliding), lever, unbalance planar (linkages) : mass, spring, damper, point on object, rotational and translational joint
hydraulical	pipes, tanks, accumulators, orifices, losses (leaks, bends), pressure relief valves

Table 3: Components in multitechnical library

During the different phases of the design of a mechatronic product, the different models do not have to be modelled at the same accuracy level. Modelling of the components at the highest accuracy level requires a lot of computing time, and also a large number of parameters, which are not always known. The model accuracy has to be adapted to the design level. The various (often not yet selected) components can be modelled roughly, e.g. as ideal element, during the initial phases of the design. To check the influence of non-linear elements on the system behaviour, all the components that may have an influence have to be modelled accurately. For instance, during the design of the power drive in a motion control system, the power switch behaviour is of interest, and the mechanical load can be modelled rather roughly (e.g. ignoring flexibility and backlash). During the design of e.g. the computer control loop for positioning the load, calculation of the different power switch signals is not desired – if they do not influence the motion of the load – since they increase the computing time tremendously. The power drive is then modelled on a functional level, e.g. as a first order system with output limits. In order to verify the starting transients and the influence of the motor torque ripple on the motion of the load, the complete motion control system has to modelled accurately.

The library provides therefore models at various accuracy levels. The accuracy level is indicated by a numeral that is appended after the model name (e.g. GEAR0, GEAR1, GEAR2) : level 0 is the most simple (ideal) level, whereas levels 1,2,... are more complex models. For instance, the most simple model of a gear pair (GEAR0) is an ideal transformer. The more complex models GEAR1 and GEAR2 include the inertia of the gears, damping, friction and backlash.

The different levels are compatible with each other: all the parameters that are needed for the most complex level return in the definition of the more ideal levels, although they are not used there. The default values of the parameters are chosen so, that they do not influence the model if they are not defined (e.g. friction force 0, gain 1, initial velocity 0, infinite limits,...).

While selecting a more accurate model, the number of input ports of the model increases, if the more accurate model describes the influence of a new time-dependent parameter.

Also the complete model structure can change, since a behavioural model can consist of more than one physical component. For instance, a power amplifier is modelled at the behavioural level as a first order system. At the physical level, the power amplifier model consists of one or more converters, the drivers for the gate signals, the control algorithm for the switch signals and the power supply.

3.3.2. Scaling

PSpice works optimally in the range $10^{-3}...10^{3}$, which is the most common rage in electrical engineering. In multitechnical systems, not all the variables are in this range. For instance, hydraulic systems are characterised by large pressure values (in the order of 10^{5} - 10^{6} Pa) and small volume flow values, in SI-units. To convert the actual values to the PSpice range a scaling factor σ is used. The relation between the PSpice voltage and current values and the original values (for across variable a and through variable q) is:

$$\begin{aligned} v &= \frac{a}{\sigma} \\ i &= \sigma q \end{aligned} \tag{8}$$

The through variable is determined with respect to a reference, which depends on the energy domain. For instance, temperature can be entered in degrees Celsius and in Kelvin; pressure can be entered as absolute pressure or related to the atmospheric pressure.

The reference at which (e.g. temperature dependent) parameters have been determined, does not have to correspond to the reference for the across variable. If the across variable temperature is entered in Celsius, and the measuring reference is T_{meas} =20°C, then the measuring temperature and the temperature dependence are passed as parameters to the subcircuit. For instance, in the model of a

temperature dependent resistor $R=R_{meas}+k_T(T-T_{meas})$, the temperature dependence k_T and the measuring temperature T_{meas} have to be entered as parameters.

The scaling influences the values of the primitives (the index * indicates the values, entered in PSpice):

$$\begin{aligned} R &= \frac{\Delta a}{q} = \frac{\sigma \Delta v}{i/\sigma} = \sigma^2 \frac{\Delta v}{i} && \Rightarrow \quad R^* = \frac{R}{\sigma^2} \\ C &= \frac{q}{da/dt} = \frac{i/\sigma}{\sigma\, dv/dt} = \frac{1}{\sigma^2}\frac{i}{dv/dt} && \Rightarrow \quad C^* = C\sigma^2 \\ L &= \frac{\Delta a}{dq/dt} = \frac{\sigma \Delta v}{(di/dt)/\sigma} = \sigma^2 \frac{\Delta v}{di/dt} && \Rightarrow \quad L^* = \frac{L}{\sigma^2} \end{aligned} \tag{9}$$

In the multitechnical library, scaling has only be included for hydraulics.

3.4. User Interface: The Schematic Editor

The multitechnical library consists of a large variety of models. This results in a large variety of subcircuit definitions (two-node subcircuits, four-node subcircuits, mechanical, electrical and hydraulical inputs,...) and of subcircuit parameters. For efficient and error-free model entry, the user interface should be such that it gives the user as much information as possible on the models and the model parameters. A graphical environment is therefore desired, in which the user can pick the component model from a menu, draw the connections between the various components, and enter the model parameter values.

Schematic editors have been developed for graphical input of electronic circuits. The user can create a schematic by retrieving the desired symbols from a library and connecting the symbol nodes with nets. Commercial programs are available to convert these schematic to SPICE-netlists.

A small drawback of this method is however that the user does not have direct control over the sequence of the commands in the netlist. The graphical method is also not so suited to enter algebraic expressions and tables. During the development of component models, which consist mostly of a few, often non-linear blocks, it is therefore more affordable to create the netlist directly.

This section will describe how a schematic editor has been customised to allow the design of multitechnical systems.

3.4.1. WORKVIEW

WORKVIEW PLUS is the name of the open design environment from Viewlogic [55]. WORKVIEW PLUS is a CAE framework that conforms to CFI (CAD Framework Initiative) standards. Tools can easily be encapsulated in the framework. WORKVIEW PLUS organises tools in toolboxes, which consist of toolbox drawers, which contain the different tools. At VTT Automation, the environment is both available on PC and on Sun. Both environments are compatible with each other. The name of the Sun environment is Powerview.

In the standard environment various tools are encapsulated, like the schematic editor ViewDraw, the program SpiceLink, which generates netlists from the ViewDraw schematic, and the simulators PSpice.

ViewDraw is a Window based schematic editor for electronic circuits. The schematic is generated by picking symbols from a library, and by connecting the ports of the symbol (or pins) with nets. The symbols can refer to primitives (voltage sources, resistors,...), to subcircuits (which are available in a separate file or organised in a library), or to other schematics.

3.4.2. Customisation

3.4.2.1. ViewDraw Symbol Library

For each model from the library, which has been discussed in the previous section, a symbol has been generated. The model parameters are added as attributes to the symbol. The attributes added to e.g. the symbol of a (mathematical) amplifier symbol are:

```
PREFIX          X
REFDES          A?                              AMP
PINORDER        IN OUT
ORDER           MOD$ " PARAMS: " K=
MOD             AMP
K               1                               100
```

The first column gives the name of the attribute, the second the definition values (or default values during the schematic entry), and the third column gives the attributes assigned by the user in the schematic. SpiceLink builds the PSpice netlist command as follows from the attributes:

```
PREFIX-REFDES PINORDER ORDER
```

The meaning of the attributes is:

PREFIX: The first character on the line in the PSpice netlist (cf. PSpice primitives) For the elements of the library, which are all subcircuits, the prefix is always **X**.

REFDES: The name of the object, which is attached to the prefix in the PSpice description file.

PINORDER: Indicates in which order the pins (connection points for nets at the symbol) occur in the PSpice netlist.

ORDER: Describes the format of the SPICE subcircuit, after the node names. For the library models this is always:

```
MOD$ " PARAMS: " PARAM_NAME=
```

The character **$** after MOD indicates that the word MOD is replaced by its attribute, the character **=** after PARAM_NAME indicates that PARAM_NAME= is copied into the netlist. The text in between **" "** is also copied to the netlist. The following text appears hence in the netlist:

```
MODEL_NAME PARAMS: PARAM_NAME=PARAM_VALUE
```

MOD: Corresponds to the name of the subcircuit in the multitechnical library. For components for which models have been provided at multiple accuracy levels, the level number is attached to the name of the subcircuit. Level 0 indicates always the ideal level, levels 1,2,.. more complex levels. The default value in the symbol is always the most complex level.

PARAM_NAME : Corresponds to the name of a model parameter. In the symbol definition, the default values occur. The default values are chosen so, that, if they are omitted, they do not influence the model (e.g. friction value 0, gain 1, initial velocity 0, infinite limits,...). In this way, the models are compatible with the lower (ideal) accuracy levels. The parameters that are included are the parameters for the highest accuracy level (most complex), and are not always used by the lower accuracy levels. They occur however in their subcircuit definition, to avoid syntax errors in PSpice.

3.4.2.2. ViewDraw Menu Customisation

To the standard menu structure of ViewDraw, a menu for including mechatronic components has been added. Fig. 21 shows a detail of the menu.

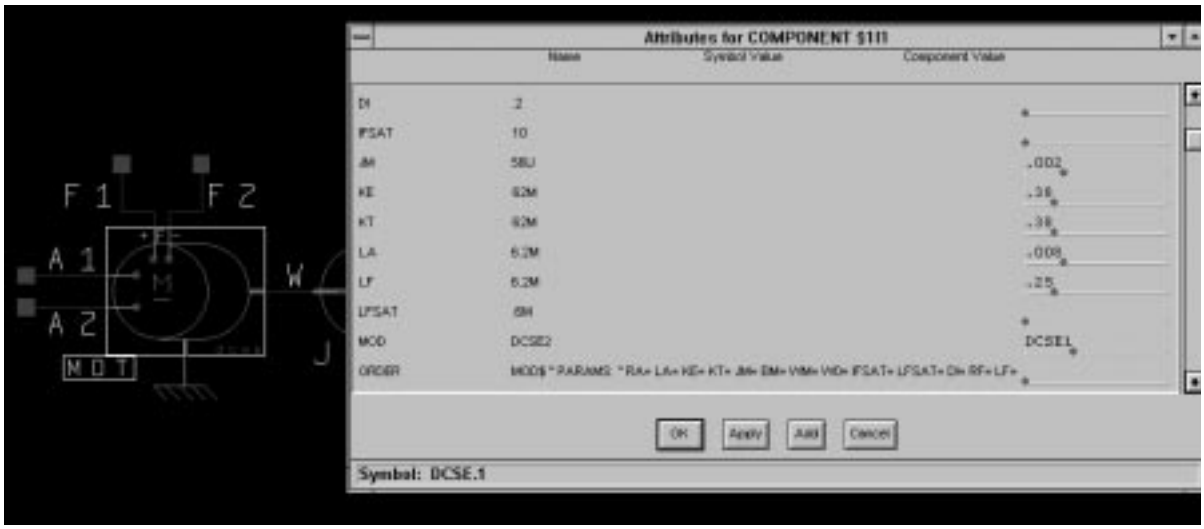
Add Comp

Command		Lib Mechatronics, PSpice commands (.tran, .ac, .param, .options, .probe, .step, .lib,...)
Probe		Voltage & Current Probe, Bus Element (X-velo.,...,X-pos.,...), Terminals
Ground		Electrical, Mechanical, Hydraulical
Source		Mathematical input functions (step, impulse, ramp, constant), Sources for Through and Across Variables (Constant, Sine, Pulse, PWL; dependent sources)
Control		
	Task Control	PID (PI, limited PI, PD, P)
	Power Drive Control:	PWM, Hysteretic Current Control, Thyristor Fire Angle
Power Drives		
	Electrical	Behavioural models (voltage controlled, current controlled, frequency controlled) Converters: line frequency converters (1-phase and 3-phase diode and thyristor rectifiers), switch-mode converters (1-quadrant, 2-quadrant, 4-quadrant DC/1-phase and 3-phase) Gate drivers: MOSFET, BJT, thyristor gate drivers
	Hydraulical	four-way valves
Actuators		
	Electro-Mech.	Behavioural models (motor +power controller) DC-motors (PM, series, shunt, compound, separately excited, brushless), induction motors (3ph., 1ph), synchronous motors (field wound, PM), stepping motors (PM, VR), SR-motors solenoids
	Hydraulical	Pumps & Hydraulic motors (fixed and variable displacement), Cylinders
Transmission		Gears, Belt drives (flat, V, synchronous), Chain, Lead Screw (ball screw), Friction Drive, Brake, Clutch, Freewheel, Cam, Cardan, Lever, Pulley
Sensor		Encoder, Tacho, Potentiometer, Current sensing resistor, Accelerometer, Vibrometer, Synchro, Resolver, LVDT, Switch
Mathematical		Gain, +, -, *, /, abs, sign, sqrt, Harmonics, Powers, exp, log, Integration, Derivation, Extrema, Sample & Hold, Transfer Functions (1st order, 2nd order, pole/zero, filters), Non-Linear (Delay, Limit, Dead zone, Relay, Table, Saturation, Cubics), Unit Conversions, Power Calculations, Statistics (max, min, mean, RMS). Variable Impedances, Transformers, Gyrators, Efficiency, Time dependent switches, initial values
Electrical		R, L, C, opamp, diode, thyristor, MOSFET, call ViewDraw analog menu
Mechanical		
	Translational	Load, Mass (gravity, feedstock), Damper, Spring, Friction, Coupling, Position, Contact (limits, backlash, lock)
	Rotational	idem as translational
	Mixed	Roll, Unbalance
	Planar	Mass, Spring, Damper, Point on Object, Rot. Joint, Transl. Joint, Rot.+Transl. Joint
	Planar bus	idem as Planar
Hydraulical		Pipe, Tank, Losses (leak, bend, connector), Orifice (fixed, variable, throttle), Accumulator (gas, spring), Pressure Relief Valve, Pos. Pressure element

Figure 19: Menu Structure

The highest levels of this menu structure are shown in Fig. 19. Dialogue boxes are used to enter the model parameters and the model accuracy level. Fig. 20 shows how a separately excited motor is entered in the schematic editor, in the netlist and in the multitechnical library.

Schematic



Netlist

```
...
XMOT A1 A2 F1 F2 W 0 DCSE1 PARAMS: RA=.0125 LA=.008 KE=.38 KT=.38
+ JM=.002 BM=1N WM=15M W0=0 IFSAT=10 LFSAT=.6M DI=.2 RF=3.7 LF=.25
XJ W MASS PARAMS: M=.02
...
.LIB "mecano.lib"
```

Definition of the subcircuit in the multitechnical library:

.SUBCKT	**DCSE1**	**1 2 3 4 5 6**	**PARAMS:**	**Ra=5.5 La=6.2M ...**
	model name (+ level) level 1 = linear level 2 = saturation	nodes		parameter names + default values

Figure 20: Separately Excited DC-motor in schematic editor, netlist and library

3.4.3. Nets and Buses

The different components of the multitechnical systems are connected by nets or buses. A net between two symbols corresponds to a common node in the SPICE-netlist. A bus is an array of nets, and is normally only used in digital electronics. They are however very useful for linking components that have large number of similar nodes.

Buses are used in the mechatronic library for planar mechanics, and for the gate signals in switch-mode and thyristor power converters.

Figure 21: ViewDraw mechatronic menu

CHAPTER 4.

MULTITECHNICAL MODELLING

Table 2 gives an overview of the relation between the electric power variables and the variables in the other technologies. Table 4 gives the basic electric primitives in the various energy domains.

Electrical	Mechanical		Fluidic
	translational	rotational	
voltage source (**V**)	velocity source (v)	velocity source (ω)	pressure source (p)
current source (**I**)	force source (F)	torque source (T)	flow source (Q)
resistor (**R**)	damping (b)	rot. damping (b)	fluidic resistance
capacitance(**C**)	mass (m)	inertia(J)	tank
inductance (**L**)	spring (k)	rotational spring(k)	fluidic inertia
ideal diode (**D**)	((cable))	freewheel	ideal relief valve
ideal switch (**S**)	((contact))	((contact))	ideal on-off valve

Table 4: Electric primitives in the different energy domains

4.1.Mathematical Model Building

Most mathematical models, which are included in the library, are blocks, like integrators, derivators, algebraic functions, transfer functions, limiters,... The mathematical library contains further coupling element models and variable impedances.

4.1.1.Block diagrams

The definition of blocks allows to use PSpice for the functional design level. The output of the block is only function of the input and a set of internal state variables. The input and output impedance are infinite. Infinite impedances can be created by copying voltages over voltage controlled sources. For instance, the subcircuit for an amplification is:

```
.subckt amp 1 2 params: K=1                  ; subcircuit definition (amplification)
rin 1 0 1T                                   ; input impedance (1e12)
ecopy 101 0 value={K*v(1)}                   ; copy and amplify input voltage
rout1  101 2 1f                              ; small resistor to avoid voltage loops
rout2 2 0 1T                                 ; output impedance (1e12)
.ends
```

For most of the blocks equivalent electric circuits or SPICE models can be found in literature [5].

4.1.2.Coupling elements

Pure coupling elements are elements which neither store nor dissipate energy, but simply transform it from one form to another [20]. In a pure two-port coupling element the power at both sides is the same, or: $i_1v_1=i_2v_2$. Coupling elements are essential for modelling power transmission components and actuators, where energy is transformed from one energy domain to another.

An **ideal transformer** is a two-port coupling element, in which the voltages at both sides are proportional to each other:

$$\begin{aligned} v_2 &= Kv_1 \\ i_2 &= \frac{1}{K} i_1 \end{aligned} \tag{10}$$

A transformer is modelled by two voltage controlled current sources, one in the primary loop and one in the secondary loop [5].

Lossy transformer. At the conceptual level, losses are characterised by entering the efficiency of the power transmission, and is expressed as a percentage of the global power. In most cases, one of the power variables is correctly transmitted (e.g. the velocity in a gear pair), and the losses occur in the other power variable (e.g. the torque in a gear pair). The efficiency can be represented by η_i for the through variable and η_v for the across variable:

$$\begin{aligned} v_2 &= \eta_v K v_1 \\ i_2 &= \frac{\eta_i}{K} i_1 \end{aligned} \tag{11}$$

If the gain K of the transformer is not constant, but depends on an external variable, the coupling element is called a **modulated transformer.** [12].

For an **ideal gyrator** the voltage at the secondary end is proportional with the current through the primary circuit:

$$\begin{aligned} v_2 &= K i_1 \\ i_2 &= \frac{1}{K} v_1 \end{aligned} \tag{12}$$

The gyrator is modelled by two current controlled voltage sources.

4.1.3. Variable Impedances

Non-linearities can often be modelled by the use of non-linear impedances. The resistors, inductors and capacitors used by PSpice are however always constant. In this section the modelling of non-linear impedances is described. The value of the impedance equals always a reference value multiplied with the voltage between two control nodes.

Variable Resistances A non-linear resistance R can be modelled by a controlled voltage source. The value of the voltage source equals the actual value of the resistance $R=R_{ref}*v_{ctrl}$ (with R_{ref} a reference resistor and v_{ctrl} the voltage over the control nodes), multiplied with the current i through the circuit:

```
.subckt rvar 101 102 201 202 params: rref=1 ;  101 102: power nodes
                                               ; 201 202: control nodes
rin 201 202 1T                                 ; input resistance for control nodes
eout 101 106 value={Rref*i(vsense]*v(201,202)}            ; multiply current with Vctrl
vsense 106 102 0                               ; sense output current
.ends
```

Variable Inductances can be defined in two ways: $v=Ldi/dt$ (LVAR) and $v=d(Li)/dt$) (LVARX). The schematic is the same for both definitions (Fig. 22), but the values for the controlled sources differ.

For LVAR, the current i is sent through a reference inductor L_{ref}; a voltage equal to the product of the voltage drop over the inductor and the control voltage v_{ctrl} is then placed between the output nodes. For LVARX, the product of the current i and the control voltage v_{ctrl} is sent through the reference inductor L_{ref}; the voltage drop over the reference inductor is copied over the output nodes.

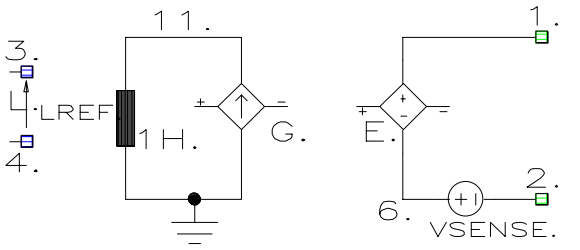


Figure 22: Schematics for the variable inductance

$$v = L\frac{di}{dt} = L_{ref} v_{ctrl} \frac{di}{dt}$$

```
.subckt lvar 1 2 3 4 params: Lref=1
rin   3 4 1G            ; input resistance
l 11 0 {Lref}
gcopy  0 11 value={i(vsense)}
eout 1 6 value={v(3,4)*v(11)}
vsense 6 2 0            ; sense iout
.ends
```

$$v = \frac{d(Li)}{dt} = \frac{d(L_{ref} v_{ctrl} i)}{dt} = L_{ref} \frac{d(v_{ctrl} i)}{dt}$$

```
.subckt lvarx 12 3 4  params: lref=1
rin   3 4 1G            ; input resistance
l 11 0 {lref}
gcopy 0 11 value={i(vsense)*V(201,202)}
eout 1 6 11 0 1         ; copy VZ to Vout
vsense 6 2 0            ; sense iout
.ends
```

Variable Capacitances A variable capacitance can also be defined in two ways: $i=Cdv/dt$ and $i=d(Cv)/dt$.

4.1.4. Equation Solution

Algebraic Equations The program PSpice can be used to solve algebraic equations, by application of a VCCS. The netlist for the equation $x-\cos(x)=0$ is:

```
g 0 x value={v(x)-cos(v(x))}
r 0 x 1T
```

The VCCS applies a current, which is function of the voltage over the output nodes of the VCCS. The large resistance pushes the current to zero. The result can be obtained from the bias point calculation. If the equation is written in explicit form ($x=f(x)$), an alternative model is:

```
g 0 x value={cos(v(x))}
r 0 x 1
```

The current through the resistor and the VCCS should then be equal to `v(x)`, since the value of the resistor is 1Ω.

Systems of Algebraic Equations can be modelled by defining a set of electric circuits, all of which consist of a VCCS, and large resistances, which push the current to zero. As an example, a system with two equations ($x+y=1$ and $xy=1$) can be modelled as:

```
gx 0 x value={v(x)+v(y]-1}
rx 0 x 1T
gy 0 y value={v(x)*v(y)-1}
ry 0 y 1T
```

Differential Equations (in explicit form) can be solved, by changing the resistor (in the explicit form of the algebraic equation) by a capacitor of 1 Farad. For instance, the equation $dx/dt=4x-2$ is modelled as:

```
g 0 x value={4*v(x)-2}
c 0 x 1 ic=1                    ; capacitor with initial conditions.
r 0 x 1T                        ; large resistor to avoid floating nodes
```

For **Integral Equations** (in explicit form) the capacitor has to be replaced by an inductance of 1 Henry.

4.2. Modelling of mechanical systems

4.2.1.Translational mechanics

In translational mechanics, the across variable is the translational velocity *v* and the through variable the force *F* (cf. Table 2).

Mass The law of Newton for translational motion is:

$$F = \frac{d(mv)}{dt} \tag{13}$$

In most cases, the mass *m* is constant, and the law of Newton is: $F=m*dv/dt$. The electric analog of a mass is a capacitance $C=m$. A variable mass, e.g. a fuel tank, is represented by a variable capacitance. In some models **Gravity** is added as a constant current source, proportional to the mass.

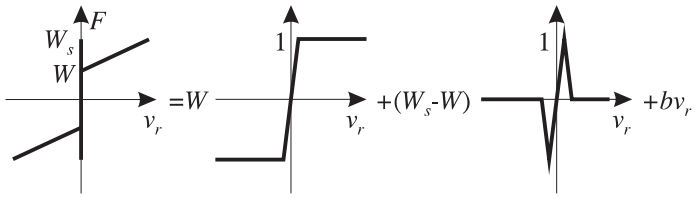
Spring. The force in a spring is given by: $F=kx=k\int v dt$. The electric analog of a spring is an inductance $L=1/k$. If initial conditions on the position are applied, it is however more appropriate to represent the spring by a position controlled force. This method allows also more easily to include non-linear springs.

Damping. The force in a damper is: $F=bv$. This corresponds to a resistance $R=1/b$.

Friction. Friction cannot be represented easily by electric primitives, and requires behavioural modelling. The friction force between two objects consists of three components, which depend on the relative velocity v_r between the objects (Fig. 23): the viscous friction $W_b=bv_r$, the kinetic friction force $W_k=W\text{sign}(v_r)$, and the static friction W_s, which only occurs at zero velocity.

The discontinuous functions for W_k and W_s are extremely difficult to handle by simulators, and are therefore approximated by piecewise linear functions, which have a small ramp near the discontinuity. Friction is modelled by two tables and one mathematical expression in PSpice (Fig. 23}.

The static friction force W_s is however often not taken into account. The library contains two friction models: FRICT1, which models only kinetic friction, and FRICT2 that models viscous, kinetic and static friction. The friction model FRICT1 returns in many mechanical models to determine friction losses and to check relative motion: the friction force is the maximum force that can be transmitted between two objects; if the force is smaller than the maximum force, no relative motion occurs between the objects. The kinetic friction is evaluated in function of $W^0 v_r$: the term W^0 is added to discard the evaluation of the tabular function if the friction force *W* equals 0, what happens often in hierarchical models where friction is included as a parameter in the model. The kinetic friction force can be entered as a constant force *W* or by a friction coefficient μ.




```
.SUBCKT FRICT2 1 2 PARAMS: W=0 Ws=0 b=0
E1 7 0 TABLE {V(1,2)}=(-.1M -1) (.1M 1)
E2 8 0 TABLE {V(1,2)}=(-.2M 0) (-.1M -1) (.1M 1) (.2M 0)
GW   1 2 VALUE = {W*V(7)+(Ws-W)*V(8)+b*v(1,2)}
r    1 2 1T
R1   7 0 1T
R2   8 0 1T
.ENDS
```

Figure 23: PSpice model of stick-slip friction

Contacts. Two models are provided for contacts: a rigid contact and a flexible contact.

The rigid model is modelled as a position controlled switch and a diode. A contact can only exert pressure and no tension, which can be modelled by a diode. This approach causes however

sudden velocity jumps, and thus infinite accelerations. A disadvantage is however that the combination of diode and switch easily causes convergence errors.

The flexibility of the switch is modelled by a spring with limited length, damping, friction and mass. The contact is only active if the spring is pressed.

Backlash. Many authors model backlash by opening or closing a switch in function of the relative position of the two objects. This model is not correct: the switches also allow to pass negative forces (i.e. a tensional force on the contact), which is not possible. A backlash is modelled by two contacts. Two models have been provided: one for rigid backlash and one for flexible backlash.

A **General Load** consists of a mass, damping and friction. This model is almost always called if a mechanical component is included (e.g. the rotor in a motor, the piston in a hydraulic cylinder).

4.2.2.Rotational mechanics

For rotational mechanics, the across variable is the rotational velocity ω and the through variable the torque T (cf. Table 2). Due to the analogy between rotational and translational systems, the same models as for translational mechanics can be used.

4.2.3.Planar mechanics

The analog circuit simulator can also be used to simulate the motion of mechanisms. General purpose mechanism analysis programs perform advanced manipulations on the system's geometrical description to obtain an optimal set of equations. SPICE uses only node equations to construct the system's equations, so that only a simple Cartesian model can be used to model planar mechanisms. This method is not so computer-efficient, but complete simulation of the system in PSpice allows to investigate e.g. the influence of actuator transients on the mechanism motion.

For the simulation of a planar mechanism, following basic models have been developed: the planar motion of the mass centre an object, a model for relating forces and velocities in a point to the forces and velocities of the mass centre, and models for rotational and translational joints.

Mass For a mass, moving in a plane, three motion laws (Eq. 13) are generated. The motion of the mass centre is modelled by three electrical circuits, one for each velocity component. The voltage in each of these nets corresponds to the absolute velocity component. Gravity is included.

Position The velocity components are integrated to obtain the position components. The nets for the position components do however not transport energy. The position components may therefore only be used as function values in controlled sources.

Motion of a point on an object A point P on the object is characterised by the coordinates x_p,y_p with respect to the coordinate system attached to the mass centre M of the object. The absolute position of this point P is given by:

$$\begin{bmatrix} x_P \\ y_P \end{bmatrix} = \begin{bmatrix} x_M \\ y_M \end{bmatrix} + \begin{bmatrix} x_{p,\theta} \\ y_{p,\theta} \end{bmatrix} = \begin{bmatrix} x_M \\ y_M \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x_p \\ y_p \end{bmatrix} \tag{14}$$

with x_M,y_M and x_P,y_P the absolute position components of resp. the mass centre M and the point P, x_p,y_p the position components of point P in a coordinate system attached to the mass centre M and θ the angle of the object's local coordinate system with respect to the global coordinate system. The translational velocity components of P equal:

$$v_{P,x} = v_{M,x} - \omega\, y_{p,\theta} \tag{15}$$

$$v_{P,y} = v_{M,y} + \omega\, x_{p,\theta} \tag{16}$$

with $v_{M,x}$,$v_{M,y}$ the absolute velocity of the mass centre M and ω the rotational velocity of the object. The contribution T_{MP} of the forces X_P and Y_P and the torque T_P in P to the torque in the mass centre M is:

$$T_{MP} = T_P - X_P y_{p,\theta} + Y_P x_{p,\theta} \tag{17}$$

Equations (15), (16) and (17) are equivalent to a combination of two transformers with variable ratio: one between the electric circuit for the rotation and the circuit for the *x*-direction with as ratio $-y_{p,\theta}$, and one between the circuit for the rotation and the circuit for the *y*-direction with as ratio $x_{p,\theta}$

Rotational Planar Joint In a rotational planar joint (or pin), the translational velocity components at both sides equal (what corresponds to a direct link between the two circuits), and the torque is zero (what corresponds to a very high impedance to the ground at both sides). Macromodels have been generated for the rotational connection between two parts, combining both the rotational joint and the transformation between the mass centres and the connection point. A more complex model of the pin joint includes joint damping and friction.

Translational Planar Joint In a translational joint, in which a fixed point of part 2 moves in part 1, the following relations yield between the velocity components:

$$v_{P1,x} = v_{P2,x} + v_r \cos(\theta_1 + \theta_s) \tag{18}$$

$$v_{P1,y} = v_{P2,y} + v_r \sin(\theta_1 + \theta_s) \tag{19}$$

with θ_1 the angle of part 1, θ_S the angle of the slider with respect to the local coordinate system in part 1, and v_r the relative velocity. In a frictionless slider, the force along the slider is zero, or:

$$0 = F_r = X_P \cos(\theta_1 + \theta_s) + Y_P \sin(\theta_1 + \theta_s) \tag{20}$$

Equations (18), (19) and (20) are equivalent to two transformers with variable gains: one between a circuit, which represents the relative motion, and the circuit for the *x*-motion with as ratio $\cos(\theta_1 + \theta_s)$, and one between the circuit for the relative motion and the circuit for the *y*-motion with as ratio $\sin(\theta_1 + \theta_s)$.

Initial Velocity If, for a mechanism with one degree of freedom, the initial velocity of the driver differs from zero, all the initial velocities of all mass centres (translational and rotational) should normally be entered. To avoid this, an initial velocity component has been generated, which applies the desired initial velocity during a small starting period (e.g. 1 μs or 1 ms). The initial velocity is applied by a constant voltage source over a switch, which is opened after the starting period, and does not influence the calculations afterwards.

The use of buses in the Schematic Editor The models for planar mechanics have many nodes, e.g. for a translational joint, the *x*- and *y*-translational velocity components and the rotational velocity and position of the mass centre of both parts are required. The resulting schematic for a simple mechanism consists of a large number of nets, so that connection errors can be easily made.

Through the use of buses, the schematic for planar mechanic systems becomes much simpler. The dimension of the bus is 6: *x*-position, *y*-position, rotational velocity, *x*-position, *y*-position and rotational position. For each subcircuit for planar mechanics, a subcircuit has been built, which converts the bus-format of the subcircuit to the original library format. Bus components, which are not used in the original subcircuit, are, connected over a very large impedance to ground or connected to equivalent bus components at other input ports. Position component nets do not transmit power. The position components are always considered as input, except for the mass model (which gives the position of the mass centre) and for the point on object model.

4.2.4. Mixed Rotational-Translational

Mixed Rotational-Translational systems have one translational and one rotational component, like rolls, and can be treated as special cases of planar mechanisms.

Rolls An ideal roll is a transformer from rotational to translational: v_{roll}-v_{basis}=-$R\omega_{roll}$, with v_{roll} and ω_{roll} the translational and rotational velocity of the roll centre, and v_{basis} the velocity of the reference. A higher level model includes slippage: slippage occurs if the force between the roll and

the reference at the contact point is larger than the maximum transmittable friction. At slippage, the velocity of the contact point will differ from the velocity of the reference, and only the maximum friction force is transmitted.

Unbalance Due to unbalances of rotors, vibrations are passed to the frame on which the system is mounted. The unbalance is defined as a mass, which is added eccentrically to the rotor. The force that is passed to the system frame, depends on the position of the rotor. An unbalance is a special case of the planar mechanical model "point on an object" (the centric rotating disk). The "point on the object" is the unbalance, where an additional mass is added. Only 1 velocity component is used (the system is supposed to be very stiff in the other direction, so that no power is transmitted). The unbalance transforms the rotating energy of the mass to vibration energy of the motor support, and is modelled as transformer (cf. Eq. (15) and Eq. (17)).

4.2.5.Transmission

The ideal power transmission element is a pure coupling element, since no power is dissipated. Models have been provided at three levels: level 0 corresponds to an ideal coupling element, level 1 includes inertia and joint friction, and level 2 other non-linear effects.

Gears. An ideal gear is a transformer, with as ratio $k=(\omega_2-\omega_{ref})/(\omega_1-\omega_{ref})=-n_1/n_2$, with n_1 the number of teeth of the first gear. This is valid for all kinds of gears: spur gears, epicyclical gears, helical gears, rack and pinion,... Models have been provided for gears with fixed axes and for planetary gears. More advanced models include gear inertia, friction and backlash.

The **Friction Drive** is equivalent to a gear. Slippage occurs if the driving torque is larger than the maximum friction force.

Lead Screws transmit a rotation to a translation, with as ratio $k=v_2/\omega_1=p/2\pi$, with p the pitch per round. A distinction is made between conventional screws and ball screws.

The higher level model for the normal screws includes the friction in the screw, which depends on the motion direction and the geometry of the screw. The tangential force F_2 equals, in function of the driving moment T_1 [16]:

$$F_2 = \frac{2\pi r + p\mu' \operatorname{sign}(v_2)}{p - 2\pi r\mu' \operatorname{sign}(v_2)} T_1 \tag{21}$$

with μ' the friction coefficient, r the screw radius, and v_2 the translational velocity.

For ball screws a constant friction force is taken. A preload is included, which aims to avoid backlash [54]. The total friction force becomes:

$$W = W_t + \mu\left(F_N + F_{pre}\right) \tag{22}$$

with W_t the translational friction force in the joint, F_N the normal force and F_{pre} the preload.

Belt drives. An ideal belt drive is a transformer with as ratio $k=\omega_2/\omega_1=r_1/r_2$, with r_1 the radius of the first pulley. At belt drives, slippage can occur if the driving torque is larger than the maximum torque transmitted by friction. The friction along the belt is derived from the equation of Eythelwein [16]:

$$\frac{F_1 - F_f}{F_2 - F_f} = e^{\mu\beta} \qquad \text{with} \qquad F_f = \rho A v^2 = q v^2 \tag{23}$$

with F_1 and F_2 the force acting along the belt sides, β the arc of contact, μ the friction coefficient, F_f the centrifugal force, with q the mass per unit length for the belt. Two possible pretension methods have been included: a constant force at the slack side, and pretensioning by adjustment of the centre distance. Models have been developed for flat belts, for V-belts and for synchronous belts.

The model of synchronous belts is based on the chain model, since no slippage can occur. Synchronous belts can also be used for transmission of rotational to translational energy. A translational node and load have been included in the model.

Chains are analogous to belts, but no slippage can occur. The model includes the flexibility of the chain. A model has also been included which models the polygon effect [16].

Clutches are modelled by two inertia, in between which a friction force is applied.

Brakes are a special case of clutches, where one inertia corresponds to the fixed frame.

Freewheels are the equivalent of diodes. In one direction, they transmit only a small friction torque, in the other direction a large torque can be transmitted.

Cams Ideal cams are modulated transformers. In most cases the pressure angle is not zero, and the total driving torque is not transmitted to the follower. The cam model calls two external tables (or algebraic functions): the first for the velocity of the follower in function of the position of the cam (at unit cam speed) (i.e. the derivative $dx/d\theta$ with x the position of the follower and θ the position of the master; the second table gives the relation between the torque delivered by the master and the force on the follower (which depends on the pressure angle and the moment arm). PSpice has however problems with rotating cams: these cams are described by tables in which θ varies between 0 and 2π. PSpice can however not jump from one end from a table to another end, without passing through all the other elements, what is needed at the crossing 2π-0.

Cardans allow to pass the rotation to non-aligned shafts. The rotation speed of the output shaft depends on the angular position θ_1 of the input shaft [16]:

$$\frac{\omega_2}{\omega_1} = \frac{\cos\alpha}{1-\sin^2\theta_1 \sin^2\alpha} \tag{24}$$

with α the angle between the two shafts.

Levers convert a rotation to a translation. If the rotations are small, the lever corresponds to a transformer with fixed ratio. Models with fixed and moving pivot have been provided. The models are only valid for small rotations; for larger rotations, the planar mechanics models must be used.

Pulley. The ideal pulley reverses the sign of the velocity (relative to the motion of the axis of the pulley). Models have been provided with fixed, moving and driven axis. If the force between the belt or cable and the pulley is larger than the allowable friction, slippage occurs. The maximum friction force is derived from the equation of Eythelwein (Eq. 23).

4.3. Electro-mechanical and magnetic systems

Electro-mechanical systems convert energy from the electric to the mechanic energy domain. In most cases, energy is transferred in two steps: first from electric to magnetic, and then from magnetic to mechanic. In the magnetic system the across variable is the magnetomotive force $\mathcal{F}$ and the through variable the flux rate $d\phi/dt$.

4.3.1. Magnetic elements

Reluctance The only element that occurs in magnetic systems is the magnetic material, which is characterised by its reluctance $\mathcal{R}=\mathcal{F}/\phi$, which is - if the material is not saturated - constant.

$$\mathcal{F} = \frac{l}{\mu A}\phi = \frac{l}{\mu_0\mu_r A}\phi = \mathcal{R}\phi \tag{25}$$

with $\mathcal{R}$ the reluctance, l the path length, A the section and μ the permeability (μ_0 the permeability of air, and μ_r the relative permeability). In the electric-magnetic equivalence, the reluctance corresponds to a capacitor $C=1/\mathcal{R}$, since it is the ratio between the across variable and the integral of the through variable.

Permanent Magnet Materials are, in a first approximation, treated as elements that generate a constant flux ϕ (the flux rate is thus zero, which is the same as a ground element for an electric circuit and a fixed point in translational mechanics).

Hysteresis To model the saturation of the core, and the losses due to hysteresis, the model of Chua and Stromsmoe [4] is used. In this model, the hysteresis is described by a differential equation:

$$\frac{d\phi(t)}{dt} = g\left[\mathcal{A}(t) - f(\phi(t))\right] \tag{26}$$

which can be rewritten as :

$$\mathcal{A}(t) = g^{-1}\left(\frac{d\phi}{dt}\right) + f(\phi) \tag{27}$$

This equation can be transformed into an electric circuit: the function g^{-1} corresponds to a (non-linear) resistance, and the function f^{-1} to a non-linear capacitance (in series with the resistance). The capacitance models the energy stored in the magnetic field, the resistance the hysteresis losses. Both the resistance and the inductance are modelled by a cubic curve in the PSpice model.

Electro-magnetical Conversion The element that converts energy from the electric circuit to the magnetic system is a gyrator. The electric circuit generates a magnetomotive force $\mathcal{F}=Ni$ in the magnetic system. The magnetic system generates in the electric circuit a back e.m.f. voltage, which equals:

$$V_g = \frac{d\psi}{dt} = N\frac{d\phi}{dt} \tag{28}$$

with $\psi=N\phi$ the flux linkage, and N the number of windings per stator pole.

Magneto-mechanical Conversion The energy in the gap is converted partly to mechanical energy, and partly by increasing the magnetic field energy in the gap. The total energy in the gap (magnetic coenergy) equals [9,13]:

$$W_{gap} = \int_0^{\mathcal{F}_{gap}} \phi(\mathcal{F}_{gap}, \theta)\, d\mathcal{F}_{gap} \tag{29}$$

with $\mathcal{F}_{gap}$ the magnetomotive force over the air gap, which depends on the position θ of the rotor. The torque, produced by the rotating magnetic field, equals to the partial derivative of the magnetic coenergy with respect to the angle [9]:

$$T_g = -\left.\frac{\partial W_{gap}}{\partial\theta}\right|_{\phi=const} \qquad \text{or} \qquad T_g = \left.\frac{\partial W_{gap}}{\partial\theta}\right|_{\mathcal{F}_{gap}=const} \tag{30}$$

Electro-mechanical Conversion The inductance L is defined as the ratio between the flux linkage ψ and the input current i, and depends generally on the rotor angle θ, and the current i (in the case of saturation). Eq. (28) becomes for a linear system (i.e. L independent of i):

$$V_g = \frac{d\psi(i,\theta)}{dt} = \frac{\partial\psi}{\partial i}\frac{di}{dt} + \frac{\partial\psi}{\partial\theta}\frac{d\theta}{dt} = L(\theta)\frac{di}{dt} + i\frac{dL(\theta)}{d\theta}\left(\omega_r - \omega_s\right) = L(\theta)\frac{di}{dt} + i\frac{dL}{d\theta}\left(\omega_r - \omega_s\right) \tag{31}$$

The first term in Eq. (31) is the self-inductance, the second term is the back e.m.f. Eq. (30) becomes [29]:

$$T = \frac{\partial W_{gap}}{\partial \theta} = \frac{\partial \int_0^{\phi} \phi\left(\mathcal{J}_{gap}, \theta\right) d\mathcal{J}_{gap}}{\partial \theta} = \frac{\partial \int_0^{i} \frac{1}{N} L(\theta) i d(Ni)}{\partial \theta} = \frac{\partial \left[\int_0^{i} L(\theta) i di\right]}{\partial \theta} = \frac{i^2}{2} \frac{dL(\theta)}{d\theta} \qquad (32)$$

The dependence of the inductance on the position increases the simulation time for motors operating at high velocities. In many cases, a position-independent motor model can then be chosen (e.g. a permanent magnet DC-motor model).

Behavioural model In many cases, not all the parameters to model the motor are available. The models that have been developed for e.g. stepper motors, model each electric phase separately, and contain position-dependent inductances. This makes simulation very slow. To speed up simulation, behavioural models are used, which use parameters, that can be retrieved from catalogues.

The motor is bought combined with the power amplifier, allowing to operate the control the torque between a minimal and a maximum torque. A model has been developed for the combination of motor and amplifier, with an input node for the desired velocity and two mechanical nodes. The model uses a PI controller to calculate the desired torque and limits the output torque, according to a maximum torque-velocity relation.

4.3.2. DC-motors

In DC motors, the orientation of the magnetic field is constant with respect to the stator. In this case, the torque can be derived from the force on a conductor, which carries a current *i*, in a magnetic field [11]: $F=Bil_{eff}$, with $B=\phi/A$ the flux density and l_{eff} the effective length of the conductor. The torque T_g, which is developed in the motor, equals hence:

$$T_g = C_t \psi\, i_a \qquad (33)$$

with K_t the torque constant, a factor that depends on the design of the motor, ψ the rotor flux linkage, and i_a the armature current.

The back-e.m.f. (electromotoric force) voltage in a conductor, which moves with a velocity *v* through a magnetic field, equals [11]: $V_{emf}=Bvl_{eff}$. For the armature windings in the DC-motor, this becomes:

$$V_{emf} = C_e \psi \left(\omega_r - \omega_s\right) \qquad (34)$$

with K_e the voltage constant, ω_r the rotational velocity of the rotor and ω_s the rotational velocity of the stator frame.

Separately Excited DC-motor. The equations for a separately excited DC-motor, including field saturation, are [19]:

$$V_f = R_f i_f + \frac{d\psi(i_f)}{dt} = R_f i_f + \frac{d\psi(i_f)}{di_f}\frac{di_f}{dt} \qquad (35)$$

$$V_a = R_a i_a + L_a \frac{di_a}{dt} + C_e\left(\omega_r - \omega_s\right)\psi \qquad (36)$$

$$T_l = C_t i_a \psi - \left[J_m \frac{d\omega_r}{dt} + b_m\left(\omega_r - \omega_s\right) + T_{fric} \right] \qquad (37)$$

with V_f the field voltage, V_a the armature voltage, T_l the torque at the motor shaft, ψ the magnetic flux linkage, i_a the armature current, ω_r and ω_s resp. the rotor and stator velocity, R_a and R_f resp. the armature and the field resistance, L_a the armature inductance, J_m the rotor inertia, b_m the damping coefficient, and $T_{fric}=W_m \text{sign}(\omega_r-\omega_s)$ the friction torque between the rotor and stator.

If the magnetic circuit is not saturated, the flux linkage ψ is proportional to the field current i_f: $\psi=L_f i_f$, with L_f the field inductance. Eq. (35), (36) and (37) are then simplified to:

$$V_f = R_f i_f + L_f \frac{di_f}{dt} \tag{38}$$

$$V_a = R_a i_a + L_a \frac{di_a}{dt} + K_e(\omega_r - \omega_s) i_f \tag{39}$$

$$T_l = K_t i_a i_f - \left[J_m \frac{d\omega_r}{dt} + b_m(\omega_r - \omega_s) + T_{fric} \right] \tag{40}$$

with K_e=$C_e L_f$ the back-emf constant and K_t=$C_t L_f$ the torque constant. If K_e and K_t equal, all the electrical power is converted to mechanical power. By making K_t smaller than K_e, losses in the

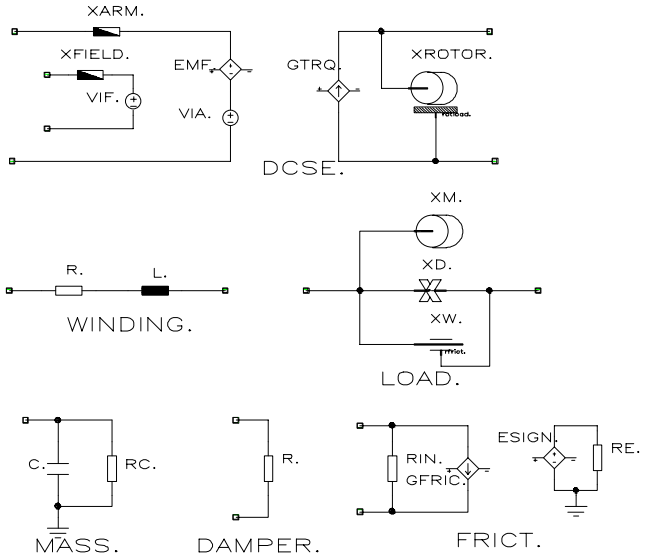


Fig. 24: Schematics for the separately excited DC-motor model and its subcircuits

```
* DCSE : separately excited shunt DC-motor
.subckt DCSE 1 2 3 4 5 6 PARAMS: Ra=1f La=1f RF=1f
+ LF=1f Ke=1 Kt=1 Jm=1f Bm=1f Wm=0 w0=0
Xarm 1 107 WINDING PARAMS: r={ra} l={la}                    ;arm. winding
EMF 107 105 value={ke*V(5,6)*I(VIF)}                        ; back-emf
VIA 105 2 dc=0                                              ; arm. current
XFIELD 3 201 WINDING PARAMS: R={rF} l={lF}                  ; field winding
VIF 201 4 dc=0                                              ; field current
GTRQ 6 5 value={Kt*i(vIA)*I(vif)}                           ; motor torque
XROTOR 5 6 LOAD  PARAMS: m={Jm} B={Bm} w={Wm} v0={w0}
.ends
*------
.subckt WINDING 1 2 PARAMS: R=1ohm L=1mH
R 1 101 {R}
L 101 2 {L}
.ends
*------
.subckt LOAD 1 2 PARAMS: m=1 b=1n w=0 v0=0
XM 1 MASS  PARAMS: m={m} v0={v0}
XD 1 2 DAMPER PARAMS: b={b}
XW 1 2 FRICT PARAMS: w={w}
.ends
*------
.SUBCKT MASS 1 PARAMS: m=1 v0=0
C 1 0 {m} ic={v0}
RC 101 0 1e12
.ENDS
*------
.SUBCKT DAMPER 1 2 PARAMS: B=1
R 1 2 {1/B}
.ENDS
*------
.SUBCKT FRICT 1 2 PARAMS: W=0
rin 1 2 1e12
ESIGN 111 0 TABLE {V(1,2)*pwr(w,0)}= (-.1m -1) (.1m 1)
RE 111 0 1
GFRIC 1 2 VALUE = {W * V(111)}
.ENDS
```

Fig. 25: Netlist for the separately excited DC-motor model

power conversion can be modelled.

Fig. 24 shows the schematic and Fig. 25 the netlist for the PSpice model of the separately excited DC-motor and the submodels. The models of the actuators always include a model of the electrical subsystem (the field and armature windings), a power conversion part, and a mechanical subsystem (the load). The electromotive force is modelled by a voltage controlled voltage source and the generated torque by a current controlled current source. For both the windings and the mechanical load lower level models are provided.

If the machine is operating near rated load, the linear model is not valid anymore, due to saturation of the magnetic field. The flux linkage ψ is then modelled as a (parabolic) blended piecewise linear function of the field current i_f (Fig. 26). The expressions for the controlled sources in the power conversion part are adapted. The derivative of the flux linkage in Eq. (35) is modelled as a variable inductance (which is a function of the field current i_f).

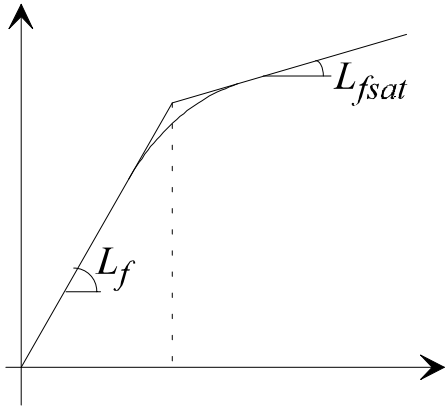


Figure 26: Saturation of magnetic field for a permanent magnet DC-motor

In a **Shunt DC-motor** the same voltage is applied over the armature and the field winding. The model is therefore the same as the model of the separately excited motor, except that it has only two electric terminals.

Series DC-motor In a series DC-motor, the field winding is in series with the armature winding, so the field current equals to the armature current. Eq. (33) and (34) become (for the non-saturated case) :

$$T_g = C_t i_a \psi(i_a) = K_t i_a^2 \tag{41}$$

$$V_{emf} = C_e \psi(\omega_r - \omega_s) = K_e i_a (\omega_r - \omega_s) \tag{42}$$

The PSpice model is analogous to the model of the shunt and the separately excited DC-motor.

Compound DC motors. The flux in a compound DC-motor is generated both in a shunt winding, and in winding in series with the armature winding. Eq. (33) and (34) become :

$$T_g = C_t i_a (\psi_{sh} + \psi_{se}) \tag{43}$$

$$V_{emf} = C_e (\psi_{sh} + \psi_{se})(\omega_r - \omega_s) \tag{44}$$

Permanent magnet DC-motors. For a permanent magnet DC-motor, the rotor flux ϕ is constant, and the torque hence only function of the current through the windings. The equations for a motor, including winding losses and mechanical losses, are [9]:

$$\begin{aligned} V &= R_a i_a + L_a \frac{di_a}{dt} + V_{emf} \\ T_l &= T_g - \left[J_m \frac{d\omega_r}{dt} + b_m(\omega_r - \omega_s) + T_{fric} \right] \end{aligned} \quad \text{with} \quad \begin{aligned} V_{emf} &= K_e(\omega_r - \omega_s) \\ T_g &= K_t i_a \\ T_{fric} &= W_m \text{sign}(\omega_r - \omega_s) \end{aligned} \tag{45}$$

Brushless DC-motors have a permanent magnet rotor and a salient stator [29]. The back-e.m.f. voltage and the generated torque are function of the position, and can be approximated by a sinusoidal function [40]. The back-e.m.f. in the *j*-th phase and the torque generated by the *j*-th phase are:

$$V_{emf,j} = K_e(\omega_r - \omega_s)\sin\left(n_s\theta - (j-1)\frac{2\pi}{p}\right)$$
$$T_{g,j} = K_t i_j \sin\left(n_s\theta - (j-1)\frac{2\pi}{p}\right) \quad (46)$$

with n_S the number of north poles on the stator, p the number of phases, and j=1...p the phase number. A PSpice model for a 3-phase brushless DC-motor has been developed, which includes armature losses and mechanical losses, like the other motor models. The mutual inductance of the phases has been included. This motor needs also a commutator model.

4.3.3. Stepping Motors and Switched Reluctance Motors

Models have been developed for Permanent Magnet (PM) and Variable Reluctance (VR) stepping motors, and for Switched Reluctance (SR) motors. SR-motors need feedback, stepping motors not.

Permanent Magnet Stepping motors have a round permanent magnet rotor and a salient stator (with concentrated windings). The back-e.m.f. in the *j*-th phase and the torque generated by the *j*-th phase are:

$$V_{emf,j} = K_e(\omega_r - \omega_s)\sin p(\theta - (j-1)\lambda)$$
$$T_{g,j} = -K_t i_j \sin p(\theta - (j-1)\lambda) \quad (47)$$

with p the number of pole pairs and $\lambda=\pi/2$ for a two-phase and $\lambda=2\pi/3$ for a three-phase motor.. A PSpice model has been generated for a 2-phase and a 3-phase motor.

Variable Reluctance Stepping motors have both a salient stator and rotor. The back-e.m.f. in the *j*-th phase and the torque generated by the *j*-th phase are:

$$V_{emf,j} = K_e i_j(\omega_r - \omega_s)\sin p(\theta - (j-1)\lambda)$$
$$T_{g,j} = -K_t i_j^2 \sin p(\theta - (j-1)\lambda) \quad (48)$$

The mutual inductance between the phases has been neglected in these equations [18]. A PSpice model has been generated for a 2-phase and a 3-phase motor.

Switched Reluctance (SR) Motors have a salient rotor and a salient stator. The equations are therefore similar to the equations for the VR stepping motor. The efficiency of the SR-motor can be improved by using saturation. Saturation can be modelled in two ways: by applying a physical model, based on the reluctances of the gap and the rotor and stator core, or by the use of a more behavioural model, which is based on current-dependent inductances.

Behavioural model The model is based on the model of Miller and McGilp [30], which is also used by the program PC-SRD for the design of SR motors. The flux linkage ψ varies between the unaligned aligned flux $\psi_u = L_u i$ and the aligned flux $\psi_a(i)$. For intermediate positions the function is - except for a transition zone - a linear function of the angle θ: $\psi(\theta, i) = \psi_u + k_a(\theta - \xi_0)$, with k_a and ξ_0 both dependent on the current *i*. The transition zones are modelled with a rational function (Fröhlich curve). For each zone (and each phase), the relations for the induced voltage and the generated torque are derived Eq. (31) and Eq. (32).

Physical model The core is modelled as a magnetic circuit. The inductance is replaced by the gap reluctance $\mathcal{R}_{gap}$, which remains constant, since the gap is not saturated. The gap reluctance is derived from the inductance: if the rotor is not saturated, the core reluctance is very small compared to the gap reluctance, so that the inductance L is inverse proportional to the gap reluctance $\mathcal{R}_{gap}$:

$$L = \frac{\psi}{i} = \frac{N\phi}{\mathcal{F}/N} = N^2\frac{\phi}{\mathcal{F}} = \frac{N^2}{\mathcal{R}_{core} + \mathcal{R}_{gap}} \approx \frac{N^2}{\mathcal{R}_{gap}} \quad (49)$$

The expression for the generated torque becomes:

$$T = \frac{\partial W_{gap}}{\partial \theta} = \frac{\partial \int_0^{\phi} \mathcal{F}_{gap}(\phi, \theta)\, d\phi}{\partial \theta} = \frac{\partial \int_0^{\phi} \mathcal{R}_{gap}(\theta)\, \phi\, d\phi}{\partial \theta} = \frac{\partial \left[\frac{\mathcal{R}_{gap}(\theta)\, \phi^2}{2} \right]}{\partial \theta} = \frac{\phi^2}{2} \frac{d\mathcal{R}_{gap}(\theta)}{d\theta} \qquad (50)$$

The SR-motor requires position feedback and commutation. A PSpice model has been developed for an SR motor with a 3-phase stator.

4.3.4. AC-motors

Three-phase induction motor The 3-phase induction motor is characterised by the equations:

$$\vec{u}_s = R_s \vec{i}_s + \frac{d\vec{\psi}_s}{dt} = R_s \vec{i}_s + L_s \frac{d\vec{i}_s}{dt} + M \frac{d\left(\vec{i}_r e^{j\varepsilon}\right)}{dt} \qquad (51)$$

$$\vec{u}_r = R_r \vec{i}_r + \frac{d\vec{\psi}_r}{dt} = R_r \vec{i}_r + M \frac{d\left(\vec{i}_s e^{-j\theta}\right)}{dt} + L_r \frac{d\vec{i}_r}{dt} \qquad (52)$$

The model is based on the generalised machine theory [33,37]. Two stationary coils are attached to the stator (direct axis *d* and quadrature *q*), and two coils are attached to the rotor (which can be seen as the real and the imaginary component of one complex coil). This leads to the following 4 equations [37]:

$$\begin{aligned}
u_{ds} &= R_s i_{ds} + L_s \frac{di_{ds}}{dt} + M \frac{di_{dr}}{dt} \\
u_{qs} &= R_s i_{qs} + L_s \frac{di_{qs}}{dt} + M \frac{di_{qr}}{dt} \\
u_{dr} &= R_r i_{dr} + M \frac{di_{ds}}{dt} + L_r \frac{di_{dr}}{dt} + \frac{d\varepsilon}{dt}\left(M i_{qs} + L_r i_{qr}\right) \\
u_{qr} &= R_r i_{qr} + M \frac{di_{qs}}{dt} + L_r \frac{di_{qr}}{dt} - \frac{d\varepsilon}{dt}\left(M i_{ds} + L_r i_{dr}\right)
\end{aligned} \qquad (53)$$

with u_{ds} the direct component of the stator voltage and i_{qr} the quadrature component of the rotor current. R_S is the stator phase resistance, L_r the rotor phase self inductance and M the mutual inductance between rotor and stator phases. The angle ε is the electrical angle, which is related to the mechanical angle $\theta = N_p \varepsilon / 2$, with N_p the number of poles. The electric circuit is shown in Fig. 27, which is analogous to the steady state circuit of an induction motor [19], but in the rotor branch a controlled source in entered dependent on the velocity and current components. The rotor circuit is short circuited, so that u_{dr}=0 and u_{qr}=0.

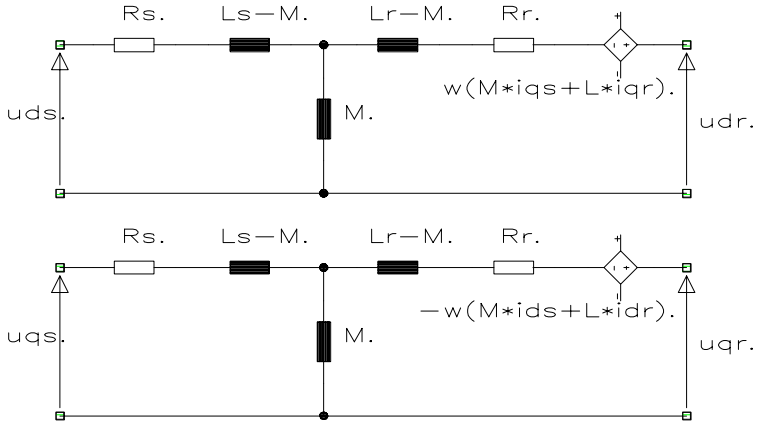


Figure 27: Equivalent circuit for the induction motor.

The generated torque is calculated from [19] :

$$T_g = \frac{N_p M}{3} \mathrm{Im}\left[\vec{i}_s \left(\vec{i}_r e^{j\varepsilon}\right)^*\right] = \frac{N_p M}{3}\left(\vec{i}_s \otimes \vec{i}_r\right) = \frac{N_p M}{3}\left(i_{sd} i_{rq} - i_{sq} i_{rd}\right) \tag{54}$$

The direct and quadrature components of the stator voltage are derived from the phase voltages:

$$\begin{aligned} u_{sd} &= u_{s1} - \frac{1}{2}\left(u_{s2} + u_{s3}\right) \\ u_{sq} &= \frac{\sqrt{3}}{2}\left(u_{s2} - u_{s3}\right) \end{aligned} \tag{55}$$

The stator phase currents and the direct and quadrature current components are related by:

$$\begin{aligned} i_{s1} &= \frac{2}{3}\mathrm{Re}(\vec{i}_s) = \frac{2}{3} i_{sd} \\ i_{s2} &= \frac{2}{3}\mathrm{Re}(\vec{i}_s e^{-j\frac{2\pi}{3}}) = \frac{2}{3}\left[-\frac{1}{2} i_{sd} - \frac{\sqrt{3}}{2} i_{sq}\right] \\ i_{s3} &= \frac{2}{3}\mathrm{Re}(\vec{i}_s e^{+j\frac{2\pi}{3}}) = \frac{2}{3}\left[-\frac{1}{2} i_{sd} + \frac{\sqrt{3}}{2} i_{sq}\right] \end{aligned} \tag{56}$$

Single phase induction motor The equivalent scheme for the three phase motor is used. The net is connected to two stator windings; the third stator winding is not connected.

Synchronous motor with field winding The model is also based on the generalised machine theory. The machine is modelled by three stator windings (the field windings and distributed winding), and two rotor windings [19,37]. If the field winding is placed on the rotor, as mostly is the case, rotor and stator are switched in the generalised machine model.

Synchronous motor with permanent magnet rotor This motor is more commonly called a brushless AC (or synchronous) motor. The motor differs from the brushless DC motor since the windings are not concentrated, but are equally spread over the stator. The flux in the rotor, and hence the rotor current, is constant. The equations are [19]:

$$\begin{aligned} u_{sd} &= R_s i_{sd} + L_s \frac{di_{sd}}{dt} - \frac{3}{2}\phi_M \frac{d\varepsilon}{dt}\sin\varepsilon \\ u_{sq} &= R_s i_{sq} + L_s \frac{di_{sq}}{dt} + \frac{3}{2}\phi_M \frac{d\varepsilon}{dt}\cos\varepsilon \end{aligned} \tag{57}$$

with ϕ_M the rotor flux. The torque equals [19] :

$$T_g = \frac{N_p \phi_M}{2}\left(u_{sq}\cos\varepsilon - u_{sd}\sin\varepsilon\right) \tag{58}$$

4.3.5. Solenoids

The solenoid is the only translating electro-mechanical motor which is included in the library [9]. The magnetic field is generated by a current in a winding. A cylindrical plunger moves in a cylindrical solenoid magnet. The flux is guided to two air gaps. The air gaph width depends on the position of the plunger, and equals to $l_0 \pm x$, with x the displacement (if x=0 both gaps are equal to l_0). The width of the air gaps is limited between 0 and a, the width of the poles. The plunger is assumed to be larger than the magnet, so that the width of one of the air gaps equals a. The inductance of the system equals:

$$L = \frac{N^2}{\mathcal{R}} = \frac{N^2}{k_R\left(\frac{1}{a} + \frac{1}{l_0 + |x|}\right)} = \frac{k_L a\left(l_0 + |x|\right)}{(a + l_0) + |x|} \tag{59}$$

4.4. Modelling of Electrical Power Drives and their Controls

The power drive regulates the power flow from the power supply to the actuator. Power drives can be hydraulical or electrical. This section will discuss the electrical power drives; hydraulical power drives (valves) will be discussed in the next section.

The basic module of the power amplifier is the power electronic converter, which is built from power semiconductor devices (or "power switches"), like transistors, MOSFET's, thyristor,... [31]. The energy flow from the power supply to the actuator is controlled by opening or closing the power switches, viz. by sending an on-off signal to the gate of the power switch. The behaviour of power drives is highly non-linear due to the switching, which can cause severe problems for the convergence of the simulation.

The complete power drive consists of one or two converters and the circuits for the control signals for the power semiconductors. The power drive consists of the following elements:

- One or two **converters**, which are built of power electronic devices. The converter architecture is limited to a few types, but there is a wide variety in the types of power electronic devices. Power electronic systems are subdivided according to how the devices are switched:

  Line Frequency Converters: The devices are switched on at the line frequency of 50 Hz. Line frequency converters can be uncontrolled (diode rectifiers), or controlled (thyristor bridges).

  Switch-mode converters: The controllable switches (e.g. MOSFET's, BJT's, IGBT's) in the converter are turned on and off at frequencies that are high compared to the line frequency.

  Power converters have often two converters: a controlled or uncontrolled rectifier to convert the net supply to a DC-voltage and a switch-mode converter to generate the pulsed signals for the actuator.

- A **switch control algorithm** to determine the state of the switch, e.g. a PWM or hysteretic current control algorithm. The algorithm is relatively independent of the type of the power switches.

- The **gate driver** for the power semiconductor devices. This circuit converts the output signal of e.g. the PWM-algorithm to a voltage or a current. The driver depends on the type of the switches.

4.4.1. Power Drive Models

The power drive has a discontinuous behaviour, so that an accurate model of a switch-mode converter requires a lot of computing time. An accurate model of the power drive is however not always needed: the mechanical time constant of the system is generally much larger than the switching interval of the power converter, so that a behavioural model of the power drive - which does not take switching into account - can be used for the simulation of the mechanical system. Behavioural power switch models have been developed at different abstraction levels:

Behavioural model of complete power amplifier The most simple and most abstract model of the power drive is a first order transfer function, i.e. a voltage or current amplifier with a time constant. The output voltage is limited. The model is almost linear, and does hence not take switching into account. The model represents actually the power converters (i.e. rectifier and switch-mode converter), the switch control algorithm, the gate drivers as well as the net supply. Models have been developed for current controlled power drives (i.e. input variable is current, output is voltage), voltage controlled power drives and frequency controlled power drives (i.e. the output signal is a sinusoidal signal with a frequency proportional to the input signal). Models have been developed for 1-phase and 3-phase output.

Behavioural model of the Power Converter This is a model of the power converter, the net supply, and the driver for the switch signals. The output signals of the switch control algorithm are the input to the model. The system is modelled as a first order system, a limiter and a voltage source.

Opposed to the first model, this model takes the switching into account, but still models the converter as an input-output block with infinite impedances.

Behavioural model of the Power Switches In this model, the power switches are assumed to be ideal: the power converters are modelled as variable gain (mathematical) transformers. The transformer gain (±1) is determined by the gate control algorithm. Models have been provided for VSI (Voltage Source Inverter), CSI (Current Source Inverter), VCR (Voltage Controlled Rectifier) and CSR (Current Source Inverter).

Electrical Converter Models The least abstract model corresponds to the actual electrical model. The topology of the converters is limited to a few types, which all use a number of identical power switches, of which there is a large variety. In order to avoid to write a new subcircuit for each possible type of switch, the PSpice library uses a user-defined subcircuit for the power switches and the power diodes.

In the schematic editor, the converter models are available as schematics (Fig. 28). The converter switch and diode are defined as a lower level schematic. In order to change the type of the switch, only the schematic for the converter switch has to be updated.

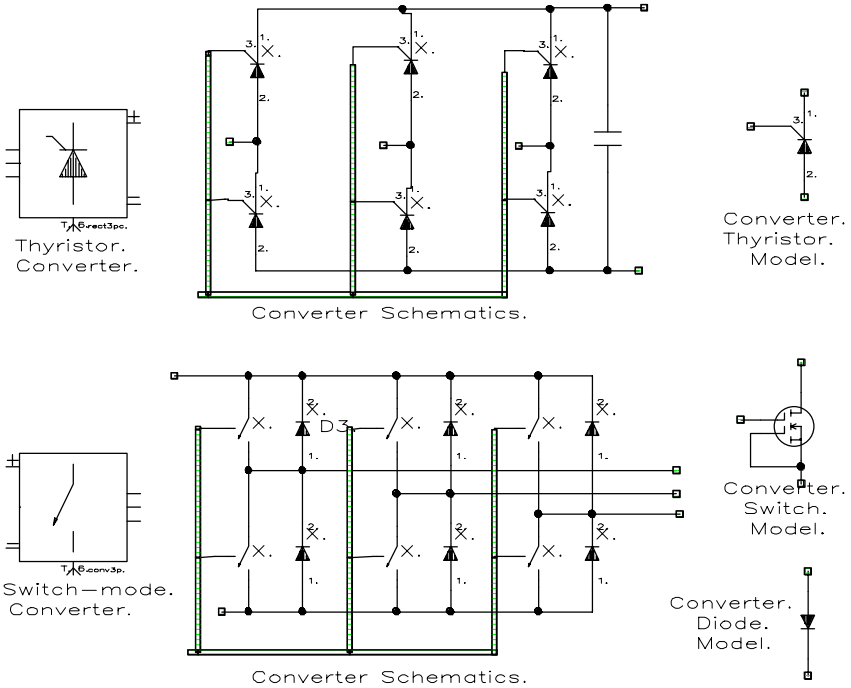


Figure 28: Lower level schematics for a thyristor and a switch-mode 3-phase converter.

Models have been developed for diode and thyristor rectifiers (1 phase and 3 phase) and for 1-quadrant, 2-quadrant DC-drives and full bridge switch-mode converters (DC, 1-phase and 3-phase).

4.4.2. Switch Control Algorithm

Two switch control algorithms have been modelled: PWM and hysteretic current control. Both algorithms have been modelled on a behavioural level.

PWM: The model for PWM consists of two parts: the PWM triggering signal (triangular or ramp), and the PWM-signal generation. Dependent on the sign of the difference between the control signal and the triggering signal, a positive or negative constant voltage is output. The control signal, which is the input to the PWM model, has to be generated by the task controller.

Hysteretic Current Control The model for hysteretic current control consists is a Schmitt-trigger: if the current $i > i_{ref}+\Delta i$, with i_{ref} a reference current and Δi a tolerance, a negative constant voltage is output, if $i < i_{ref}-\Delta i$, a positive constant voltage is output. The output is analogous to the output of the PWM.

The switch control algorithms are blocks with infinite input- and output impedance. The output of the algorithm is a single signal for a DC or 1-phase motor, or 3 signals for a 3-phase motor.

4.4.3. Power Switch Drivers

The power switch driver models generate the gate signal from the control algorithm (e.g. PWM). Behavioural driver models have been developed for MOSFETs, BJTs and thyristors.

MOSFET Drivers The gate signal is a voltage source. The value of the voltage is about 10 V higher than the voltage between the drain and the source of the MOSFET.

BJT Drivers The gate signal is a current source.

Thyristor Drivers The gate signal is a current pulse. The commutation of the thyristor is characterised by the firing angle, which determines the moment that the current pulse starts. The value of the firing angle α is determined from the desired output voltage V:

$$\alpha = \arccos\left(\frac{V}{kV_m}\right) = \frac{\pi}{2} - \arctan\left(\frac{V/kV_m}{\sqrt{1-\left(V/kV_m\right)^2}}\right). \tag{60}$$

with V_m the amplitude of the input signal and k the relation between the maximum output voltage and V_m ($2/\pi$ for a 1-phase rectifier and $3/\pi$ for a 3-phase full bridge rectifier [7]).

The pulse for the first thyristor starts when the phase of the input signal equals to the firing angle. The model assumes a constant input frequency, so that the phase of the input signal can be determined by a triangular pulse wave. The pulse is stopped after the next thyristor is ignited.

The input for the model is the output of the switch control algorithm. The output is the voltage or current signal that is sent to all the power switches. Buses are used in the schematic editor between the power switch driver and the converter model, in order to reduce the number of nets that has to be entered by the designer.

4.4.4. Discussion

With the models that have been provided, normal motor control methods can be modelled.

The abstraction level of the power drive model determines the computing time drastically: the difference in computing time between the least and the most abstract model of the power converter is of the order of magnitude 100...10000.

The switching of the power switches causes a discontinuity in the voltage over the motor, which will make it more difficult to find a convergent solution. Power controllers can be modelled without problems if the switching occurs according to a predetermined pattern. If the switching is controlled by a feedback signal, PSpice has often severe problems to find a convergent solution, and halts the simulation. In order to find a convergent solution, the convergence parameters (voltage and current tolerance, relative tolerance, number of iterations) can be changed, but this does not guarantee that a solution will be found, and that this solution will be correct. Another possibility is to add a delay, in order to cut the feedback loop in the system of equations. This can however cause a ripple in the output results, which can be reduced by reducing the delay – which will however increase the simulation time, since the simulation step is always smaller than half of the delay.

4.5. Modelling of Sensors

Sensors convert a signal from the electrical, mechanical or hydraulic energy domain to an analog or digital electric signal. The model of a sensor consists therefore of two parts:

- a model in the sensing energy domain.
- an electric model. The sensor consists normally of the actual sensor and a signal conditioning device, which converts the measurement to a usable form. An encoder e.g. has a pulse train as output, which has to be further processed to derive the position. In the multitechnical library, the electrical part is often modelled on a behavioural level, as e.g. a voltage source that is proportional to the measured position.

The following models have been developed with behavioural output:

Encoder: the model consists of a mechanical load. The output signal is proportional to the position.

LVDT: the model consists of a mechanical load. The output signal is proportional to the position.

Tacho: the model consists of a mechanical load. The output is proportional to the velocity.

Current Sensing Resistor: the model is a resistor. The output is proportional to the current through the resistor.

Vibrometer: the mechanical model is a mass, which is connected with a damper and spring to the system. The output is proportional with the displacement of the spring.

Accelerometer: the mechanical model is the same as the model of the vibrometer. Normally the spring (representing a piezoelectric element) is much stiffer. The output is proportional with the force between the two masses.

The following models have an electric output (hence at least two electric output nodes).

Potentiometer: the mechanical model is a load, the electric model a potentiometer circuit.

Resolver: is modelled as two transformers, the gains of which depend on the position [41].

Synchro: is modelled as three transformers, the gains of which depend on the position [41].

Position controlled switch: a switch is closed if the relative position between the measured object and the reference object is larger than a predetermined value

4.6. Modelling of hydraulical systems

In hydraulical systems, the across variable corresponds to the pressure p and the through variable to the volume flow rate Q. In the PSpice model, the reference pressure (v=0) corresponds to 0 Pa or 10^5 Pa, the pressure for which the reference parameters are determined.

4.6.1. Basic Hydraulic Laws

Compressibility: Bulk modulus Hydraulic fluids are compressible: the density of the fluid ρ depends on the pressure p and the temperature T. The compressibility is characterised by the bulk modulus E [28]:

$$\rho = \rho_0\left[1+\frac{1}{E}\left(p-p_0\right)-\alpha\left(T-T_0\right)\right] \tag{61}$$

with p_0 the measuring reference pressure (i.e. 10^5 Pa if v=0 corresponds to 0 Pa, 0 if v=0 corresponds to 10^5 Pa), and α the cubical expansion coefficient. The cubical expansion coefficient

Continuity Law For hydraulic systems the continuity equation for a flow through a chamber becomes [28]:

$$\sum Q_{in} - \sum Q_{out} = \frac{dV_0}{dt} + \frac{V_0}{E}\frac{dp}{dt} \tag{62}$$

with V_0 the volume of the chamber.

Flow Through Conduits Flow can be laminar or turbulent, dependent on the Reynolds number. In the PSpice modelling, the type of flow is predetermined.

Orifice Flow is characterised by the relationship :

$$Q = C_d A_0 \sqrt{\frac{2\Delta p}{\rho}} \tag{63}$$

with C_d the discharge coefficient, A_0 the orifice area and Δp the pressure drop over the orifice. The orifice flow is modelled as a controlled current source.

Scaling Hydraulic systems are characterised by large pressure values (in the order of $10^5 - 10^6$ Pa) and small volume flow values, in SI-units. To convert the hydraulic SI-values to this range, a scaling factor $\sigma=10^6$ Pa/V is used. The relation between the PSpice values and the hydraulic SI-values is:

$$\begin{aligned} v &= \frac{p - p_{ref}}{\sigma} \\ i &= \sigma Q \end{aligned} \tag{64}$$

4.6.2. Hydraulic Primitives

Tank For a constant volume tank with one input port, Eq. (62) becomes:

$$Q_{in} = \frac{V_0}{E}\frac{dp}{dt} \tag{65}$$

which corresponds to a capacitance $C=V_0/E$ connected to the ground.

Hydraulic inertia Due to the law of Newton, a force, or a pressure difference, will generate an the acceleration of a fluid in a pipe:

$$\Delta p = \frac{\rho l}{A}\frac{dQ}{dt} \tag{66}$$

with l the length and A the section of the pipe. This corresponds to an inductance $L=\rho l/A$.

Viscous flow During laminar flow in a pipe, a pressure drop is generated in a pipe, proportional to the velocity, e.g. for a circular pipe [28]:

$$\Delta p = \frac{128\mu l}{\pi d^4} Q$$

with l the length and d the diameter of the pipe. This corresponds to a resistance $R=128\mu l/d^4$.

Leaks are also modelled by a resistance (flow proportional to pressure difference)

4.6.3. Hydro-mechanical Actuators

In an ideal hydro-mechanical actuator, the flow is proportional to the rotational (or translational) velocity of the shaft (resp. plunger). An ideal hydro-mechanical actuator corresponds to a gyrator, due to the relation between force and pressure ($F=pA$) and between velocity and volume flow ($Q=vA$).

Models have been provided at two levels: Level 0 corresponds to the ideal gyrator, level 1 includes losses.

Pump A pump converts rotational energy to hydraulic energy. Models have been provided for fixed and variable displacement pumps. The models include a mechanical load, leaks (both internal between low and high pressure compartment and external leaks), and a cavitation loss [28]. The mechanical load consists of the rotating mass, and friction (consisting of a constant term, a term proportional to the velocity, and a term proportional to the pressure).

Hydraulic motor The models for the hydraulic motor are the same as for the pump. Models are provided for both fixed and variable displacement motors.

Cylinder A cylinder consists of two hydraulic chambers and a mechanical load. The model of the load is the same as for the pump. The influence of the changing volume of the chambers on the pressure (cf. Eq. 62) is modelled by a variable capacitance. The force created through the acceleration of the fluid on the piston is modelled by a variable capacitance (to the mechanical load). The model also includes internal and external leakage.

4.6.4. Hydraulic Power Drives : Valves

The physical model of a 4-way valve is a network of 4 orifices, with a mechanical load. The flow through the orifice is determined by a non-linear equation (cf. Eq. 63). The forces from the fluid on the plunger include an inertia force and the steady state force, which is derived from the approximated law: F=0.43$wx\Delta p$ [28], with w the area gradient of the orifice and x the plunger displacement. Problems occur during simulation, especially after closing of the valve. The model parameters are physical parameters, like the gap width, the length between the ports, and the contraction coefficients, which are difficult to determine.

Linear models have been developed for the valve. The flow to the load equals :

$$Q = K_q x - K_c \Delta p_L \tag{67}$$

with K_q the flow gain, K_C the flow-pressure coefficient and Δp the pressure difference between the input and output port of the load. The number of inputs and outputs to this model differs from the physical model, since this model does not include the supply and return pressure, and has no mechanical load.

4.6.5. Other Hydraulic Elements

Pipe A pipe is modelled by a resistance (for the pressure drop due to viscous friction), a capacitor at both pipe end (for the compressibility), and an inductance (for the inertia). The flexibility of the pipe is taken into account in the capacitance [2]

Accumulator Two models for accumulators have been developed : for gas loaded accumulators and for spring loaded accumulators. The accumulators are modelled as hydraulic chambers (thus variable capacitors). The pressure of the fluid is the same as the pressure of a gas volume for the gas loaded accumulator, and in equilibrium with the spring force for the spring loaded accumulator.

Hydraulic Losses A model has been added to model hydraulic losses at bends, fittings, and entries. At bends, the pressure loss is described by [28]:

$$\Delta p = K \frac{\rho}{2} \left(\frac{Q}{A} \right)^2 \tag{68}$$

with K the loss coefficient and A the section of the pipe.

Pressure Relief Valve An ideal pressure relief valve is a limiter: the valve is always closed until the pressure exceeds a preset maximum pressure. Above the maximum pressure, it acts as a resistor. Both a behavioural model and a physical model have been generated.

Positive Pressure Element Negative PSpice voltages indicate pressures smaller than the reference. The pressure cannot be smaller than a minimum value. A model has been provided to keep the pressure

above the minimum value. Eq. (61) is however a linearisation round the reference pressure, and is not valid for low pressures. For low pressures the results will be inaccurate.

4.7. Commercial Motor Libraries

All the model libraries, which have been discussed until now, contained parametric component models. For each commercial component, the parameters have to be retrieved and to be entered in the model.

For rapid prototyping of mechatronic systems, it would be advantageous if the designer could take the model of the commercial component direct from the library, without the need to look for the component parameters. This requires the availability of libraries with commercial components, not only of libraries with (parametric) component models.

4.7.1. Electronic PSpice libraries

PSpice is delivered with an extensive set of commercial electronic component libraries. These libraries are partly developed by MicroSim, partly by the producers of electronic components. The libraries contain either models (for diodes, bipolar transistors, MOSFETs) or subcircuits (e.g. Operational Amplifiers, Thyristors,...).

For some of the components, symbols are available in ViewDraw. The ViewDraw symbols are however meant for Printed Circuit Board (PCB) design, and do often not have the correct attributes for simulation with PSpice. For PSpice primitives, e.g. for commercial diodes and transistors, only the MODEL attribute has to be input. For subcircuits, the following attributes have to be provided: PINORDER for the order of the pins, ORDER=MOD$ with MOD the name of the subcircuit and PREFIX=X to indicate that the component is a subcircuit.

4.7.2. Motor and reducer libraries

ServoPlan is a program developed at VTT Automation for the conceptual design of electric drives. The program uses a set of rules to select motors and reducers from databases, which are all in a dBase III format.

A program CreaLib has been developed to convert these libraries to PSpice libraries. The program uses, for each different model, a conversion routine that derives the parameters of the PSpice model from the parameters in the database. The program CreaLib will be discussed in section 6.2.4..

Databases are available for DC-motors (permanent magnet), AC-motors (3-phase induction motor) and Reducers (gears).

A component occurs in this library as:

```
* GFRK 090-22/1,7       : Permanent magnet DC-motor
* Manufacturer : LENZE
* Supplier     : REFIMEX OY
.SUBCKT DCGFRK090_22/1*7 1 2 3 4
XMOT 1 2 3 4 DCPM1 PARAMS:
+ Ra=4.53 La=0.0141 Ke=0.69 Kt=0.69 Jm=0.004
.ENDS
```

The component is a subcircuit, which contains only one element, viz. a model of the multitechnical library. All the models in one library have the same number of input nodes and represent the same type of component. Symbols have been provided in ViewDraw for these library models. The attribute MOD of these symbols has to be set to the library model of the respective component.

4.8. Digital Behavioural Modelling

Modern motion control systems consist not only of analog components, but also of digital electronic components. The digital electronics can be a digital network on a printed circuit board, but also an advanced IC (e.g. motion control IC) or be coded as a computer program.

Digital systems can be modelled on several levels: the electrical (analog) level, the switch level, the gate (or logic) level, the register transfer level and the behavioural level [36].

- In the **switch level**, transistors are modelled as gate-controlled switches and modelled as “on” or “off”.
- At the **gate or logic level**, transistors are grouped into logic gates. Rather than dealing with voltages and currents at signal nodes, discrete logic states are defined, and Boolean operations are used to determined the new logic value at each node.
- The **Register Transfer Level** (RTL) is a higher level of abstraction, and is used for data path design. At the RTL related bits of information are grouped into ordered sets of words or buses. The set of statements for the RTL model involves a sequence of register transfers and arithmetic operations that are similar to data-flow descriptions.
- The **behavioural level** allows the definition of functional blocks. The statements are mostly written in a hardware description language (HDL).

The possibility to model multitechnical systems at the behavioural level is desired for the modelling of digital controllers. These are often software coded programs or e.g. motion control IC’s, from which the internal structure is not known, only their behaviour. Software coded programs must be modelled as sampled systems, in which the input and output signals change at equidistant time intervals.

4.8.1. Mixed-mode simulation

Mixed-mode simulation techniques can be classified in three groups [36]: manual, glued and fully integrated.

- In the **manual** approach, the results of a digital logic simulation are fed into an analog circuit. This technique cannot be used if feedback paths exist.
- In the **glued** approach, two existing simulators - one analog and one digital - work concurrently run concurrently under a multitasking operating system and exchange data back and forth [51]. This solution is used by companies that have invested severely in the development of separate simulators, and not willing to abandon them in favour of a completely new simulator [36]. Examples of glued mixed-mode simulators are e.g. ViewSim (with PSpice or HSpice) and Saber. The approach works mostly only between two specific simulators, and is not open to other simulators.
- The **fully integrated** approach is the most efficient and flexible approach, in which the algorithms for analog and digital (logic) simulation are tightly coupled. Examples of fully integrated mixed-mode simulators are PSpice and SPLICE [36]. This approach has also the advantage that the whole system can be modelled in one netlist, and the results verified in one post-processing run.

PSpice is a fully integrated mixed-mode simulator. Digital systems are always modelled at the logic level: digital signals are represented by logical levels, such as “1”, “0” and “unknown”. The digital simulation algorithm uses an event-driven logic processing technique, which calculates logic transitions and propagation delays. This approach allows to simulate systems containing analog electronics or behavioural analog models with digital electronics efficiently.

Digital systems can however not be modelled at the behavioural level, which is desired for systems including software coded programs, or IC’s with unknown structure. If the digital system cannot be

represented on the logic level, the system can be modelled on an analog behavioural level. The digital sampling has then to be modelled by a Sample and hold component, which consists a.o. of a constant pulse train and a capacitor [5]. This element causes however almost discontinuous transitions at the sampling moment, which causes problems in the PSpice simulation. Digital controllers require also delay elements, which also can increase the simulation time severely, if the delay is small. Modelling of digital control systems on a behavioural level is therefore rather inefficient in PSpice.

4.8.2. VHDL

VHDL (**V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage) is a standardised programming language for describing the structure and the behaviour of digital electronic systems [1]. VHDL allows the description of the structure of a design, i.e. how it is decomposed into subsystems, and how these subsystems are interconnected; and it allows the specification of the function of the design on a behavioural level. The structure of the design indicates how the subsystem is connected to the environment, through ports. The behaviour of a model can be described in a programming language, which offers the basic functions provided by each programming language. The signals can be bits, characters, integers, floating point, arrays of these types and user defined types.

4.8.2.1. Implementation of VHDL in Viewlogic

Viewlogic has as digital simulator ViewSim, which supports VHDL. There are a few differences between Standard VHDL and Viewlogic's VHDL, e.g.:

- The Standard VHDL type BIT has been replaced by VLBIT, which assumes values 0, 1, X and Z, instead of only 0 and 1.
- Viewlogic's VHDL currently does not allow signals of any type other than VLBIT, whereas Standard VHDL allows any type of signals.
- Viewlogic's VHDL does not support floating point.

VHDL files can be translated and debugged with ViewSim. A VHDL model can be included in the ViewDraw model by the generation of an appropriate symbol, which has the correct number of ports, and an attribute that refers to the VHDL file.

4.8.2.2. Mixed mode simulation in Viewlogic

Viewlogic allows mixed-mode simulation with the "glued" approach: the digital system is simulated with ViewSim, Viewlogic's digital simulator, the analog system with PSpice or HSpice. The digital system can either be a hardware model or a VHDL model.

The mixed-signal system is entered as a schematic in ViewDraw. In the schematic elements D2A (digital-to-analog) and A2D (analog-to-digital) are required to interface with the digital and the analog simulators. These models have to be included for each digital signal.

The program **madsnet** generates a separate digital and analog netlist from the schematic. The simulation in both the analog and the digital simulator is controlled by the program **madsim**. The results can be post-processed with the post-processor ViewTrace from Viewlogic.

4.8.3. Mixed-mode Behavioural Modelling with Viewlogic's VHDL

Viewlogic's VHDL has two limitations that make behavioural mixed-mode modelling difficult: the restriction of signals to (extended) bits, and the limitation of variables to integers.

Since Viewlogic's VHDL does not support floating point data, all calculations on the behavioural level have to be performed with integers. The integers in Viewlogic's VHDL are always 32 characters long, which allows to use scaling factors, and so to use fixed decimal point values. This complicates however the calculations. Mathematical expressions, like goniometric functions, are not supported,

which makes the modelling of e.g. an IC that performs a coordinate transformation for vector control of induction machines almost impossible.

Ports of subsystems can in Viewlogic's VHDL only be (extended) bits, whereas Standard VHDL supports all types of signals. In Standard VHDL it would therefore be possible to e.g. enter the position as a floating-point value to a VHDL-model of a motion control IC or algorithm. The procedure in Viewlogic is much more complicated: the position has to be translated to separate bits before input to the VHDL-model. This requires the model of an ADC (and on the other side of the VHDL-model possibly a DAC). These models have to be provided as analog models in PSpice. They can be modelled as physical models, using transistors, or by use of behavioural descriptions. The use of behavioural models is more flexible, but due to the discontinuous character of the DAC and the ADC it is very difficult to produce models that are efficient and which converge always.

If Viewlogic would support both floating point values and signals, it would be possible to model mixed-mode systems on a behavioural level. At this moment it is however extremely difficult.

4.8.4. The future: VHDL-A

VHDL-A (e.g. HDL-A[49]) is an extension to VHDL for analog modelling. HDL-A is basically an analog Hardware Description Language, but capable of describing mixed signal models and designed to work either with SPICE simulators or in conjunction with VHDL simulators [49]. The language allows also the modelling of digital logic, either stand alone or imbedded in analog models, using a subset of VHDL, but the digital modelling capabilities are not as extensive as the analog ones. The negotiations for the standardisation of the VHDL-A language are still going on.

HDL-A supports objects of the type analog, which consist of a through and across variable. The syntax of VHDL has been extended to describe analog systems, by using the terminology that is used in schematic editors and analog circuit simulators. Since multitechnical systems can be converted to electric circuits by the energy conservation principle, VHDL-A can also be used to model multitechnical systems, and become the standard in the simulation of multitechnical systems.

CHAPTER 5.

EXAMPLES

5.1. Motion Control System

The first example is the design of a cascade current-velocity-position control system, which is shown in Fig. 29 [24]. The desired position φ is compared with the (sampled) feedback signal φ_S from an encoder, and sent through a limiting PI-controller (LPIC). The output is the desired voltage, which is compared with the return signal ω_S from a tacho, and sent to a second LPIC, which calculates the desired motor current i_D. The current signal is sent through a limiter (LIM), compared with the signal i_S of a current sensor (CSEN), and sent through a third LPIC. The output of this i_A of this LPIC is the control signal for the current controlled power amplifier. The power amplifier consists a diode 1-phase full bridge rectifier, which converts the AC net voltage to a DC voltage, and of a full bridge switch-mode converter, built from 4 MOSFETs. The algorithm which is used to control the MOSFET gates is a hysteretic current control. The motor is a permanent magnet DC-motor. A flexible coupling with backlash connects the motor to the rotational load, which consists of an inertia, damping and friction.

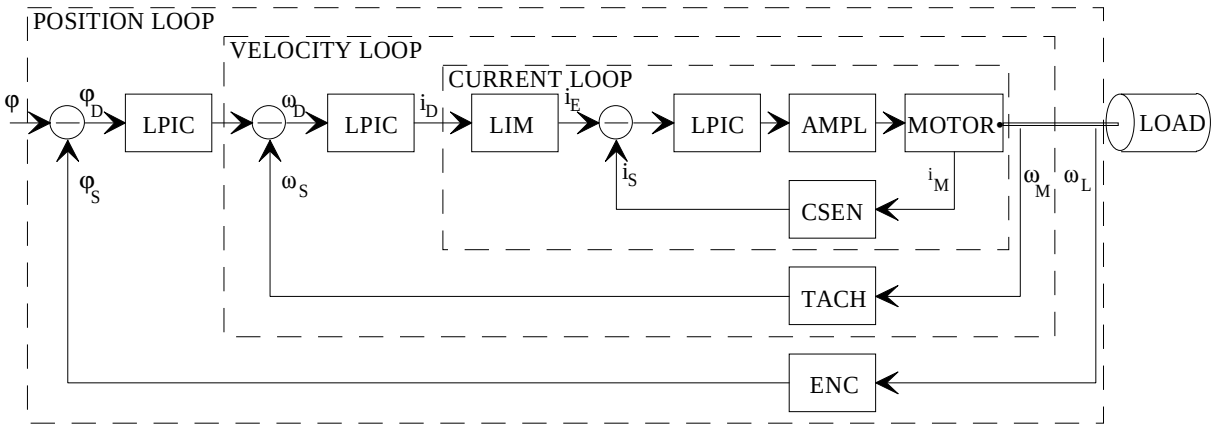


Figure 29: Cascade current-velocity-position controlled drive system.

For each of the components of this system, subcircuits have been provided in the library and symbols in ViewDraw. Fig. 30 shows the ViewDraw model and Fig. 31 the PSpice netlist (generated by SpiceLink). The (current controlled) power amplifier is modelled on a behavioural level: i.e. as a first order system, with a limiter on the output voltage. For the tacho and the encoder the ideal models (no mechanical part) have been used. The parameters, which are used, are stored in a file, which is included in the model: if the netlist is built directly from the schematic, the user has no control on the sequence of the different commands in the netlist, which can lead to syntax errors in PSpice. By writing the parameters to a separate file, the user can define dependencies between the parameters. The parameters of the PI-controllers have been optimised using classical control techniques [42]. To determine the system control parameters, the system has been linearised, by making the friction in the motor zero and by removing the backlash element.

Fig. 32 shows the PSpice simulation results for the load position and the load and motor velocity, for a step change input of the desired position. The oscillations in the load velocity are due to the flexibility between the motor and the load. Fig. 33 shows the motor current and the motor voltage.

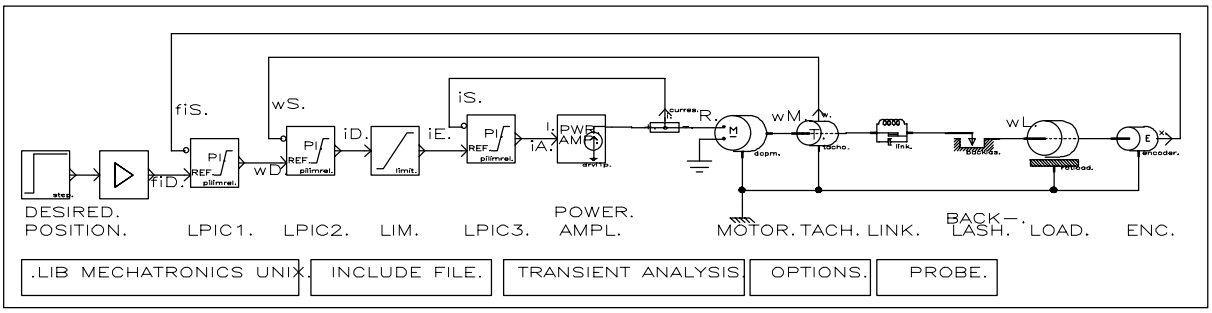


Figure 30: ViewDraw model of the motion control system

```
* Project DYNMOT1
* Powerview Wirelist Created with Version 5.1.2
* Inifile   : wspice.ini
* Options   : -p -f -n
* Levels    :

XSTEP   FIDES STEP  PARAMS: T0=0 DT=1MS
XAMP     FIDES FID AMP  PARAMS: K={PSEN}
XCP      FIS FID WD  PILIMREL  PARAMS: KP={KPP} KI={KIP} VLIM={VLIM} KLIM=1
+ V0=0 RREF=1MEG
XCV      WS  WD  ID  PILIMREL  PARAMS: KP={KPV} KI={KIV} VLIM={VLIM} KLIM=1
+ V0=0 RREF=1MEG
XLIMI    ID  IE  LIMIT  PARAMS: VMIN={-ILIM} VMAX={ILIM}
XCI      IS IE  IA  PILIMREL  PARAMS: KP={KA} KI={KA/TAPI} VLIM={VLIM}
+ KLIM=1 V0=0 RREF=1MEG
XPWR     IA  R1  DRVI1P  PARAMS: K={KA} TAU={TAPOW} VS=200
XCSEN    R1  R   IS CURRES  PARAMS: K={CSEN}
XPMDC    R   0   WM  0   DCPM1  PARAMS: RA={RA} LA={LA} KE={KEM} KT={KEM}
+ JM={JM} BM={BM} WM=0 W0=0
XTAC     WM  0   WS  TACHO0  PARAMS: K={VSEN} J=1N B=1N W=0 W0=0
XSH      WM  WBL LINK0  PARAMS: K={KS} B={BS} W=0 L0=0 X0=0 V0=0 M=1F
XBL      WBL WL  BACKLAS0  PARAMS: XBL={WBL} K={KBL} B={BBL} XTOL={WBL/5}
+ XBL0=0 RSW=1MEG
X137     WL  0   ROTLOAD  PARAMS: J={JL} B={BL} W={TD} W0=0 TCT=0
X139     WL  0   FIS ENCODER0  PARAMS: K={PSEN} J=1N B=1N W=0 TH0=0 W0=0
.PROBE
.OPTIONS LIMPTS=2000 ITL1=40 ITL2=20 ITL4=20 ITL5=0 ABSTOL=1NA CHGTOL=.01PC
+ CPTIME=1E6 GMIN=1E-12 RELTOL=.01 NUMDGT=6 VNTOL=1UV TNOM=27
.TRAN 1MS 70MS 0 .1MS UIC
.INC "dynmot.par"
.LIB mecano.lib
.END
*-------------------------------------------------------------------------------------------
* File dynmot.par
* parameters dynast example motion control
.param La=.62 Ra=2.05 Kem=45.2m Jm=7.48u Bm=4.72u Ilim=3.67 Csen={10/3}
.param Vsen=4.17m Ka=2.4 Tapi=1m Vlim=10
.param R=5MEG Tapow=.2u ks=0.885 Bs=79u Jl=2.96u Bl=12.5u
.param Td=4.72u Wbl=3.34m
.param kbl={1/4.72e-6} ;1meg
.param Bbl=1.25u       ;1meg
.param Psen=2.43m
.param Ts=1m
.param KpV=50 KiV=30
.param KpP=150  KiP=0
```

Figure 31: PSpice netlist of the motion control system, generated by SpiceLink

At the start of the system, the desired output motor current is very high. Due to the inductance of the motor, the current cannot reach immediately the desired value, and the output of the voltage is as high as possible. As the motor current i_s increases rapidly, the input current i_a for the amplifier decreases rapidly, and after about 6 ms, the (scaled) current i_a becomes smaller than the actual motor current i_s. Since i_a increases rather rapidly, the motor current cannot follow the desired current, and the motor voltage is as negative as possible. After 7 ms both currents equal again approximately to each other, and a (temporary) equilibrium is found.

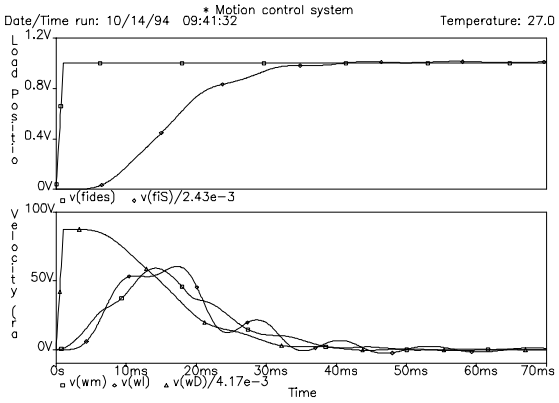


Figure 32: PSpice results: Desired (φ_{des}) and actual (φ_s) position of the load; desired (ω_d) and actual (ω_m) motor and load (ω_l) velocity.

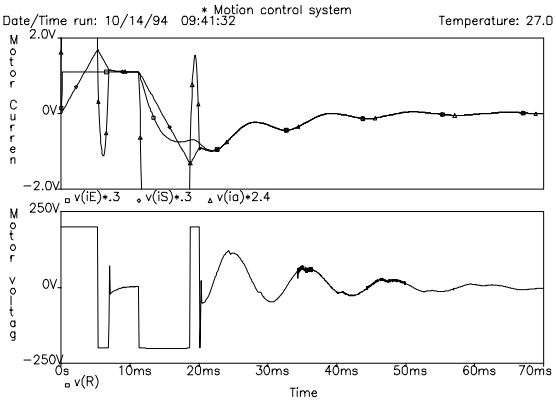


Figure 33: PSpice results: Desired (i_d) and actual (i_s) motor current and desired output current of the amplifier (i_a); Output voltage of the amplifier.

To analyse the motor control more exactly, the behavioural model of the power amplifier is replaced by an electric model of the amplifier. Fig 34 shows the new ViewDraw model: the behavioural amplifier model is replaced by the net supply, diode bridge rectifier, switch-mode converter, a hysteric current control algorithm and a model for the drivers of the MOSFET's of the converter. For the rectifier and the converter schematics have been defined. For the diodes and switches in these schematics lower level schematics have been defined. In order to change the type of all the switches, only the lowest level schematic has to be changed.

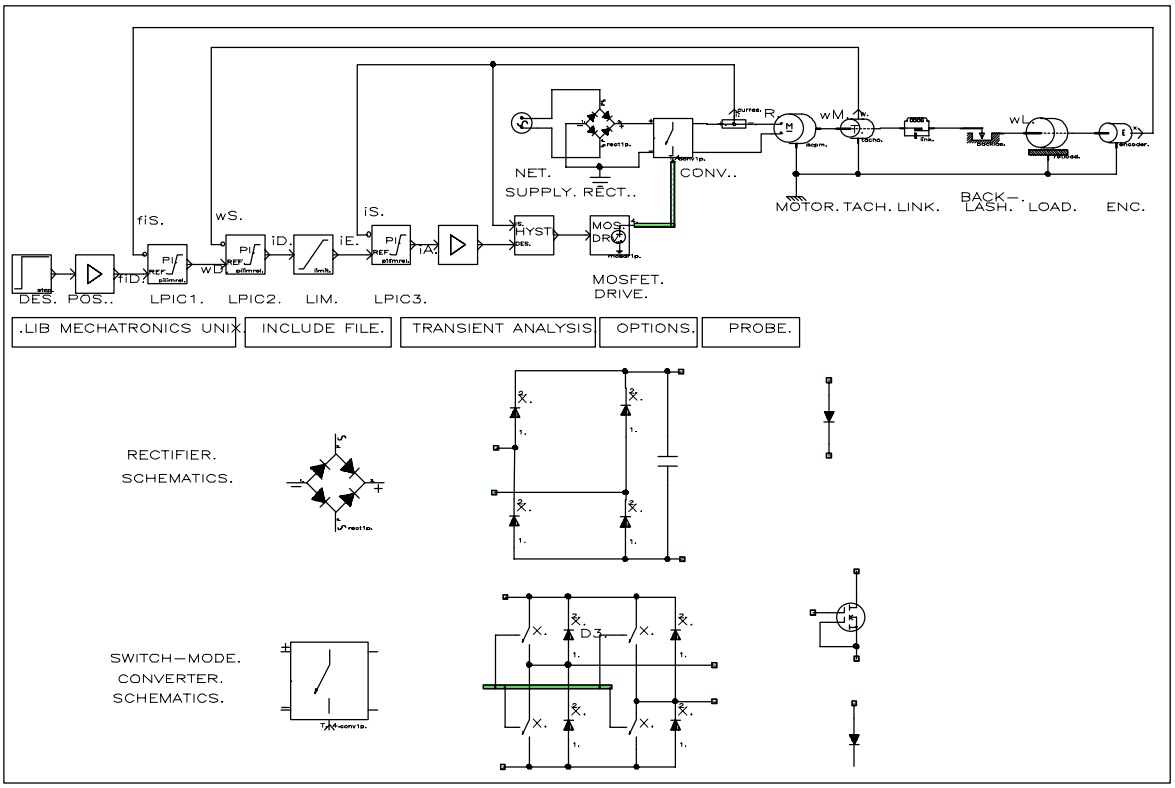


Figure 34: ViewDraw model of the motion control system with electric model of the power amplifier and schematics of the diode and the converter.

Problems occurred however in the simulation of the complete system. The simulation of the converter itself, with a predetermined desired input current (to the hysteretic control algorithm) worked well. The simulation of the complete system worked well during the first 7 ms, i.e. when the motor output voltage is maximal and the switch only has to change once (cf. Fig. 33). After 7 ms, the output

voltage is not saturated, meaning that the switches regularly have to change. At one of the switch changes, PSpice has convergence errors.

5.2. Crank-slider Mechanism

The crank-slider mechanism consists of a permanent magnet DC-motor, a crank, conrod (connection rod) and slider. Fig. 35 shows the ViewDraw model. For each rod (crank and conrod) a component is provided for the motion of the mass centre. Three nets are provided for each rod, one for each velocity component. The motor velocity is attached to the velocity component of the mass centre of the crank. A component is included for the pin connection between the ground and the ground, and another for the connection between crank and conrod. A last component is the slider, which is a joint with two degrees of freedom (rotational and translational). The slider model also includes friction between the slider and the ground. Terminals are added to symbol pins that are not used to avoid an error message in the converter program SpiceLink.

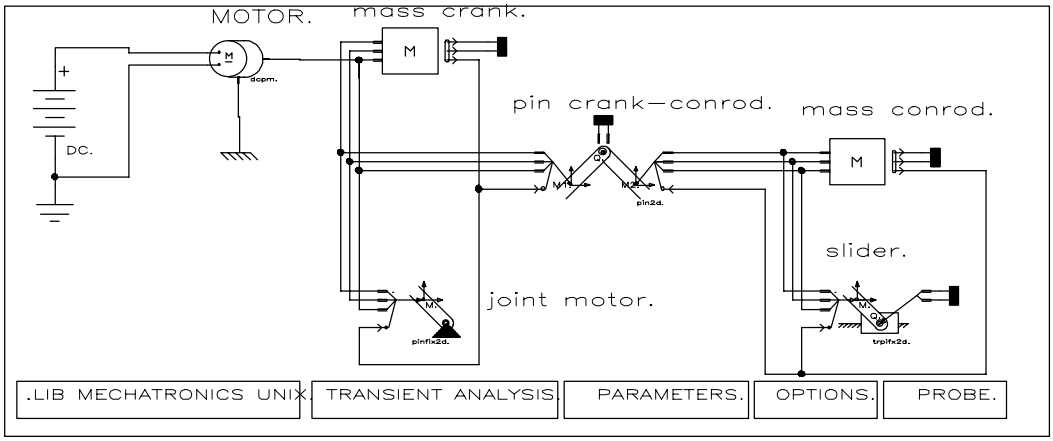


Figure 35: ViewDraw model of the crank-slider mechanism

The ViewDraw model in Fig. 35 is rather complicated. The model can be simplified by the use of buses. Fig. 36 shows the ViewDraw model using buses and Fig 37 the PSpice netlist. The model consists of the same components, as the previous model, but is simpler. An additional component has been added to imply an initial velocity to the motor shaft during the first millisecond.

Fig. 38 shows the simulation results. The system starts in fully aligned position, with a rotational velocity of 190 rad/s. In this example, a fixed voltage was applied to the DC-motor. As can be seen in Fig. 7, neither the motor torque nor the crank velocity is constant. In most mechanical system analysis programs, the system is driven by either a predetermined torque or motion. In this example, such an approach would lead to serious errors.

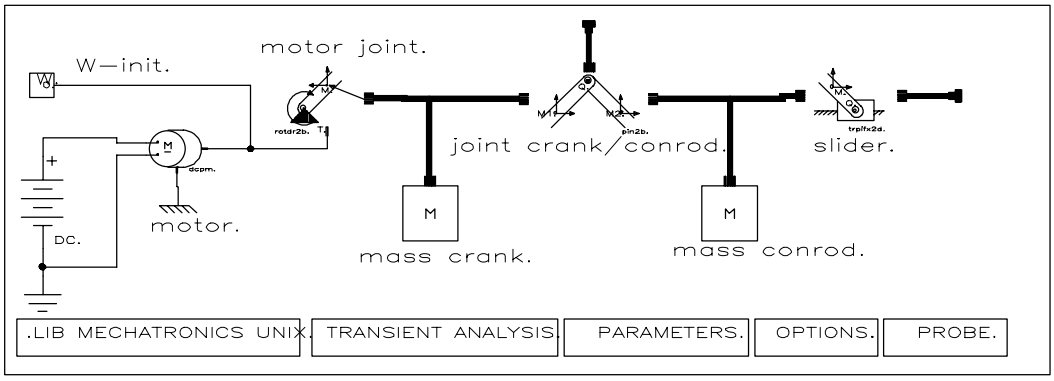


Figure 36: ViewDraw model of the crank-slider mechanism, using buses.

```
* Project CRANK3
* Powerview Wirelist Created with Version 5.1.2
* Inifile   : wspice.ini
* Options   : -p -f -n
* Levels    :

VS      VS  0   DC 12
XWINI    WM  INIVEL  PARAMS: V0= 190 DT=1MS RON=1U ROFF=1MEG
XPMDC    VS  0   WM  0   DCPM1  PARAMS: RA=5.5 LA=6.2M KE=62M KT=62M JM=56U
+ BM=.1N WM=15M W0=0
X01      Q11 Q12 Q13 Q14 Q15 Q16 WM  ROTDRF2B  PARAMS: X1={-LK/2} Y1=0
XMKRUK   Q11 Q12 Q13 Q14 Q15 Q16 MASS2B  PARAMS: M={MK} J=1U GX=0 GY=-10
+ X0={LK/2} Y0=0 TH0=0
X12      Q11 Q12 Q13 Q14 Q15 Q16 Q21 Q22 Q23 Q24 Q25 Q26 QC1 QC2 QC3 QC4 QC5
+ QC6 PIN2B0  PARAMS: X1={LK/2} Y1=0 X2={-LD/2} Y2=0 W=0 B=1F
XMCON    Q21 Q22 Q23 Q24 Q25 Q26 MASS2B  PARAMS: M={MD} J=1U GX=0 GY=-10
+ X0={LK+LD/2} Y0=0 TH0=0
XTRAN    Q21 Q22 Q23 Q24 Q25 Q26 QB1 QB2 QB3 QB4 QB5 QB6 TRPIFX2B1  PARAMS:
+ X1={LD/2} Y1=0 M={MS} GX=0 GY=-10 WT=.1 MUT=0 BT=1F WR=0
R1179    QC1 0   1T
R117912  QC2 0   1T
R117913  QC3 0   1T
R117914  QC4 0   1T
R117915  QC5 0   1T
R117916  QC6 0   1T
R1180    QB1 0   1T
R118012  QB2 0   1T
R118013  QB3 0   1T
R118014  QB4 0   1T
R118015  QB5 0   1T
R118016  QB6 0   1T
.PROBE
.OPTIONS LIMPTS=2000 ITL1=40 ITL2=20 ITL4=10 ITL5=0 ABSTOL=1UA CHGTOL=.01PC
+ CPTIME=1E6 GMIN=1E-12 RELTOL=.1M NUMDGT=6 VNTOL=1UV TNOM=27
.PARAM LK=.05 MK=.05 LD=.5 MD=.5 MS=1
.TRAN .1 2 0 UIC
.LIB mecano.lib
.END
```

Figure 37: PSpice netlist of the crank slider mechanism

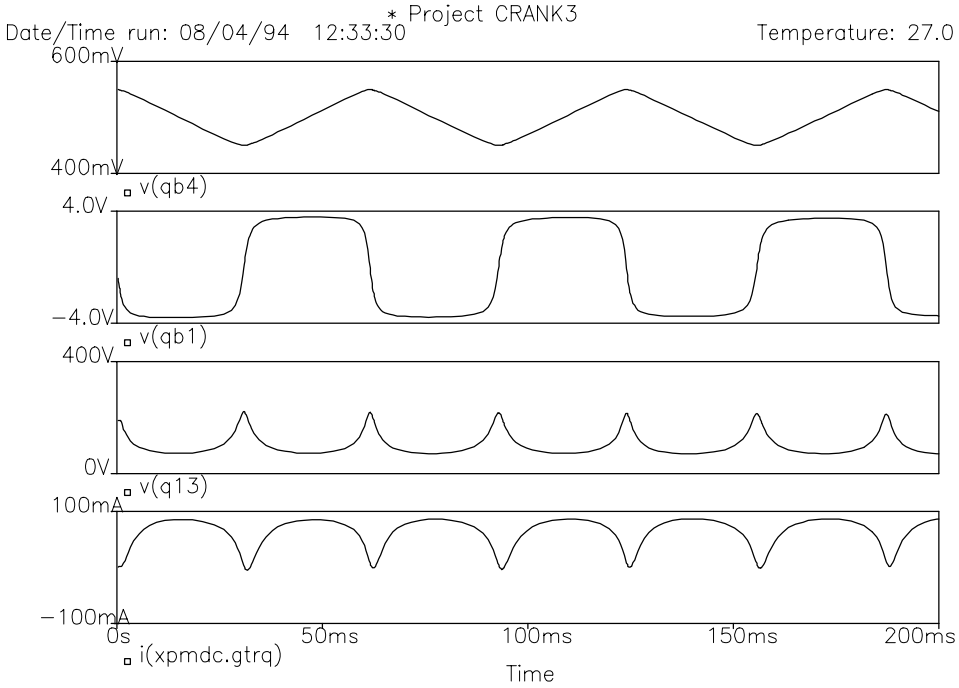


Figure 38: Simulation results of the crank-slider mechanism : Position of the slider; Velocity of the slider ; Rotational velocity of the crank ; Motor torque

5.3. Mechanism with two degrees of freedom

The example, which is presented here, is a double pendulum, which consists of two identical rods (Fig. 17), which is also discussed as an example in Chapter 2 (section 2.7.2.).

The models, which have been developed for the motor and the gears, are pure rotational. Since the gears and the motors are also translating, additional components (masses and pin connections) have to be added for the planar motion of the gears and the motor.

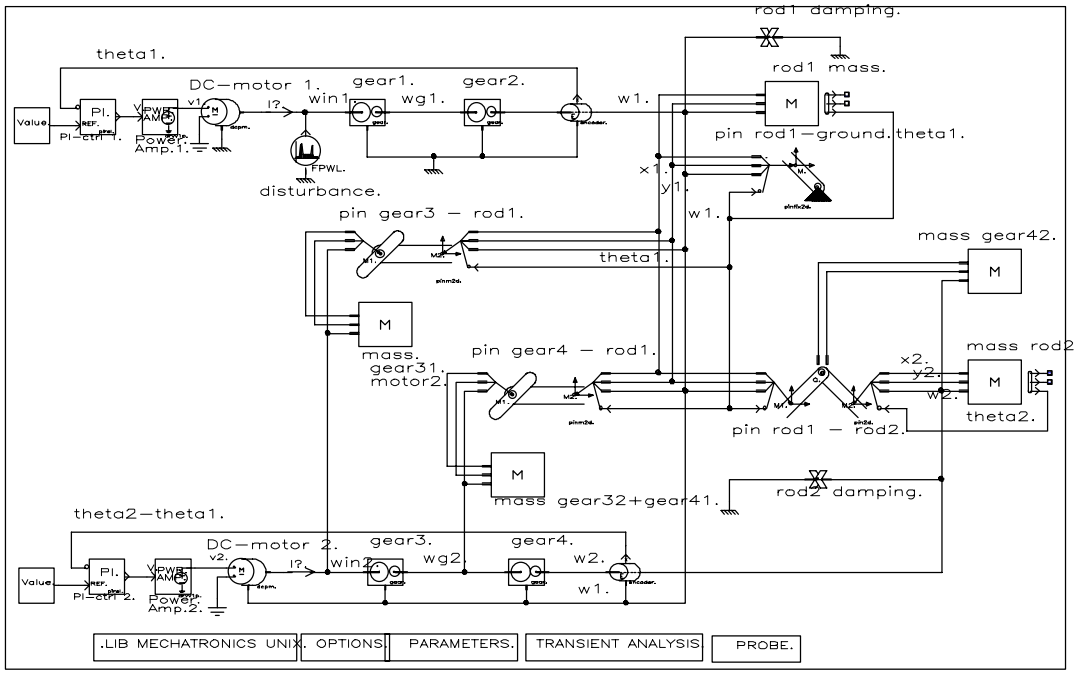


Figure 39: ViewDraw model of the system with two degrees of freedom

The ViewDraw model, as shown in Fig. 39, is rather complicated. Through the use of buses, the model can somehow be simplified. Fig. 40 shows the ViewDraw model using buses and Fig. 41 the PSpice netlist.

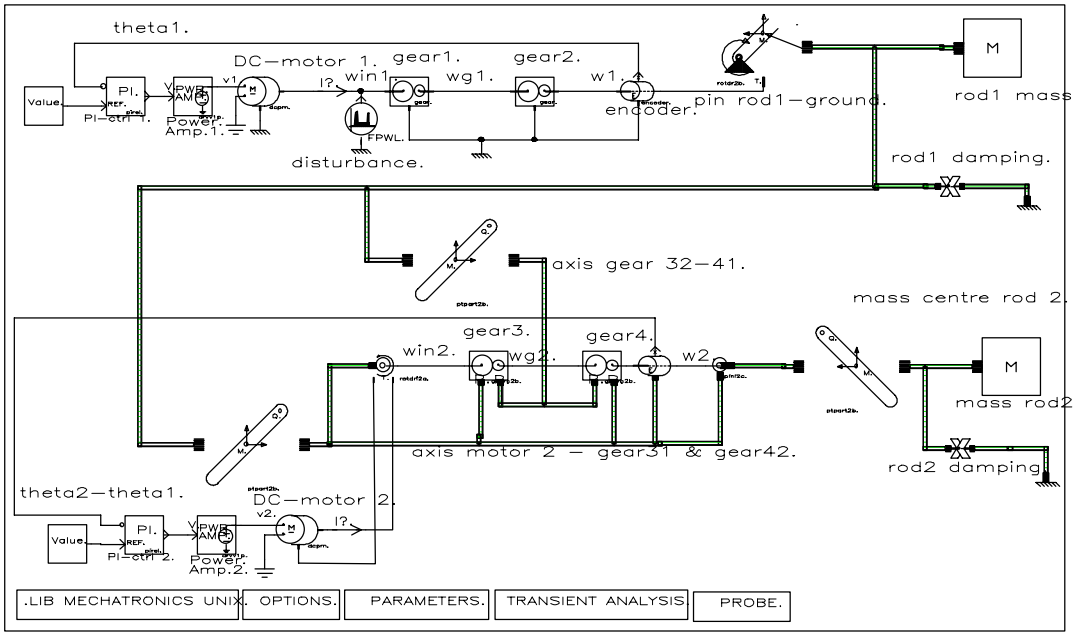


Figure 40: ViewDraw model of the system with two degrees of freedom with buses.

```
* Project TWOROD3
X151     VDES1 CONST  PARAMS: X=0                          ; desired position rod 1
X1411    PW1 VDES1 VPWR1 PIREL  PARAMS: KP=5 KI=2                    ; control rod 1
XPWR1    VPWR1 V1  DRVV1P  PARAMS: K=1 TAU=1MS VS=24                 ; power amplifier rod 1
XPMDC    V1  0   WM11 0   DCPM1  PARAMS: RA=5.5 LA=6.2M KE=62M KT=62M JM=56U
+ BM=.1N WM=15M W0=0                                        ; motor rod 1
V1406    WM11 WM1 DC 0 AC 0                                 ; sense torque rod 1
IIN      0   WM1 PWL ( 0 0 .001 1 .1 1 .101 0 )            ; disturbande
XGEAR1   WM1 0   WG1 GEAR1  PARAMS: N1=20 N2=60 J1=4E-5 J2=3.17E-3 B1=1N
+ B2=1N W1=0 W2=0 BL=1 BL0=0 KBL=1MEG BBL=1MEG RSW=1MEG XTOL=1M EFFIC=1. W0=0 ; gearpair1
XGEAR2   WG1 0   W1  GEAR1  PARAMS: N1=1 N2=3 J1=4E-5 J2=3.17E-3 B1=1N B2=1N
+ W1=0 W2=0 BL=1 BL0=0 KBL=1MEG BBL=1MEG RSW=1MEG XTOL=1M EFFIC=1. W0=0 ; gear pair 2
XENC1    W1  0   PW1 ENCODER1  PARAMS: K=1 J=1N B=1N W=0 TH0=0 W0=0    ; encoder 1
XPIN     M11 M12 M13 M14 M15 M16 W1  ROTDRF2B  PARAMS: X1=0 Y1={L1/2}  ; fixed pin gears-rod
XROD1    M11 M12 M13 M14 M15 M16 MASS2B  PARAMS: M=0.42411 J=8.836E-3 GX=0
+ GY=-9.81 X0=0 Y0={-L1/2} TH0=0                            ; mass rod 1
XD1      M11 M12 M13 M14 M15 M16 G11 G12 G13 G14 G15 G16 DAMPER2B  PARAMS:
+ BR=0.1                                                   ; damping rod 1
V1811    G11 0   DC 0                                      ; connect end of damper to ground
V18118   G13 0   DC 0
V1812    G12 0   DC 0
R18136   G14 0   1T
R18138   G15 0   1T
R18141   G16 0   1T
XA       M11 M12 M13 M14 M15 M16 A1  A2  A3  A4  A5  A6  PTPART2B  PARAMS:
+ XP=0 YP={-L1/2}                                          ; axis of motor and gear pair 3
XA2      M11 M12 M13 M14 M15 M16 A21 A22 A23 A24 A25 A26 PTPART2B  PARAMS:
+ XP=.08 YP={-L1/2}                                        ; axis of gear pair 4
XDES2    VDES2 CONST  PARAMS: X=0                          ; desired position rod 2
XPI2    PW21 VDES2 VPWR2 PIREL  PARAMS: KP=5 KI=2          ; control rod 2
XPWR2   VPWR2 V2  DRVV1P  PARAMS: K=1 TAU=1MS VS=24        ; amplifier rod 2
XDCPM2  V2 0   WM2R1 WM2S DCPM1  PARAMS: RA=5.5 LA=6.2M KE=62M KT=62M
+ JM=56U BM=.1N WM=15M W0=0                                ; motor rod 2
V1440    WM2R1 WM2R DC 0 AC 0                              ; sense torque rod 2
XBEAR1    A1  A2  A3  A4  A5  A6  WI2 WM2S WM2R ROTDRI2A  PARAMS: M1=1F J1=1F
+ M2=1F J2=56U                                             ; convert bus to net and add torque
XGEAR3   WI2 WG2 A1  A2  A3  A4  A5  A6  A21 A22 A23 A24 A25 A26 GEARN2B1
+ PARAMS: N1=20 N2=60 J1=4E-5 J2=3.17E-3 B1=1N B2=1N W1=0 W2=0 BL=1 BL0=0
+ KBL=1MEG BBL=1MEG RSW=1MEG XTOL=1M EFFIC=1. W0=0 M1=0.196 M2=1.923 GX=0
+ GY=-9.81                                                 ; gear pair 3
XGEAR4   WG2 W2  A21 A22 A23 A24 A25 A26 A1  A2  A3  A4  A5  A6  GEARN2B1
+ PARAMS: N1=1 N2=3 J1=4E-5 J2=3.17E-3 B1=1N B2=1N W1=0 W2=0 BL=1 BL0=0
+ KBL=1MEG BBL=1MEG RSW=1MEG XTOL=1M EFFIC=1. W0=0 M1=1F M2=1.764 GX=0 GY=-9.81; gearp. 4
XENC2    W2  A1  A2  A3  A4  A5  A6  PW21 ENCODER2B0  PARAMS: K=1 JR=1F JS=1F
+ MR=1F MS=1F B=1N W=0 TH0=0 W0=0                          ; encoder 2
* bearing : construct the velocity of the pin of the 2nd rod
* from the velo. of the gear & the gear axis
XBEAR2    W2  A1  A2  A3  A4  A5  A6  AO1 AO2 AO3 AO4 AO5 AO6 PINI2C
XM2      M21 M22 M23 M24 M25 M26 AO1 AO2 AO3 AO4 AO5 AO6 PTPART2B  PARAMS:
+ XP=0 YP={L2/2}                                           ; connection with rod 2
XROD2    M21 M22 M23 M24 M25 M26 MASS2B  PARAMS: M=0.42411 J=8.836E-3 GX=0
+ GY=-9.81 X0={SIN(TH02)*L2/2} Y0={-L1-L2/2*COS(TH02)} TH0={TH02}      ; mass rod 2
XD2     M21 M22 M23 M24 M25 M26 G21 G22 G23 G24 G25 G26 DAMPER2B  PARAMS:
+ BR=0.1                                                   ; damping 2nd rod
V134711  G21 0   DC 0                                      ; connect end of damper to ground
V1347118 G23 0   DC 0
V134712  G22 0   DC 0
R1347136 G24 0   1T
R1347138 G25 0   1T
R1347141 G26 0   1T
.PROBE
.TRAN 1M 2 0 1M UIC
.LIB mecano.lib
.OPTIONS LIMPTS=2000 ITL1=40 ITL2=20 ITL4=10 ITL5=0 ABSTOL=1NA CHGTOL=.01PC
+ CPTIME=1E6 GMIN=1E-12 RELTOL=.01M NUMDGT=6 VNTOL=1UV TNOM=27
.PARAM L1=.5 L2=.5 TH02=0
.END
```

Figure 41: PSpice netlist of the system with two degrees of freedom

For the first rod, the rotational motion of the motor is fed to a pin-connection (which includes actuator input) to the first rod. Two components have been provided for the position of the connection points for the gear axes. The motor of the second motor is fed at a bearing, which converts the bus to a single net for the rotational motion. In the bearing, inputs are provided for the DC-motor model (which includes only translational motion) is input. The bus models for the gears have two bus inputs for the positions of the gear carrier points (which are attached to the first rod), and two single net nodes for the rotational velocities of the gears. After the encoder, a second bearing combines the rotational velocity of

the encoder shaft with the translational velocity of the gear carrier axis, to build the velocity of the second rod.

The simulation results are given in Figure 42.

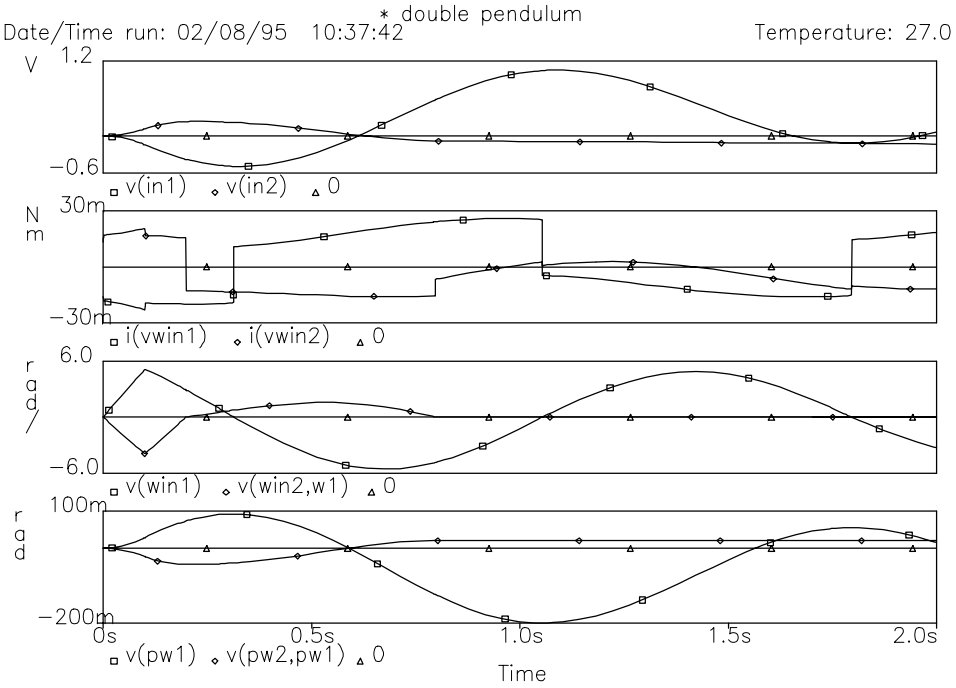


Figure 42: Results of the simulation of the system with two degrees of freedom: output voltages of the PI-controllers, torques at the DC-motor shaft, (relative) rotational velocities of the motor shafts, and relative angular positions of the rods.

5.4. Electro-hydraulic Positioning System

The electro-hydraulic system [25] (Fig. 43) has to position a mechanical load, on which a constant force and friction acts. The system consists of a servo-amplifier, which sends a current to a servo hydraulic four-way valve. The valve controls a symmetric hydraulic positioning cylinder, which actuates the load. The position and the acceleration of the load are fed back to the servo amplifier.

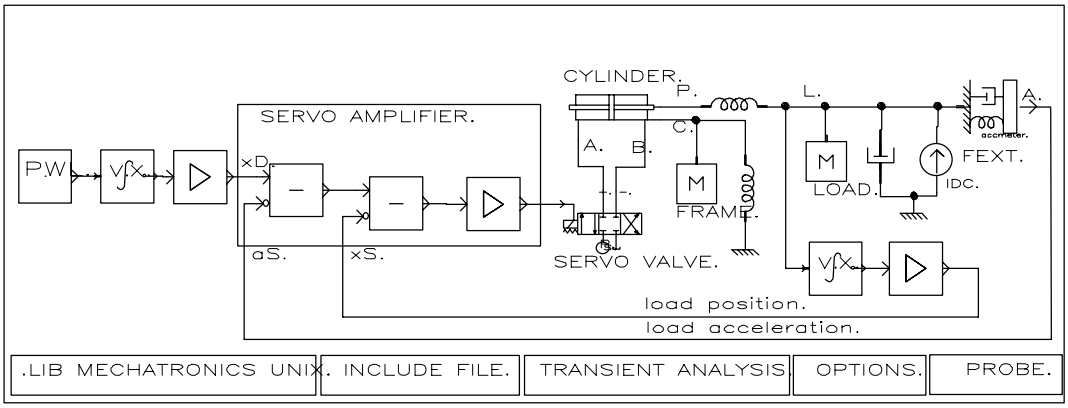


Figure 43: ViewDraw model of the electro-hydraulic system.

Fig. 43 shows the ViewDraw model and Fig. 44 the PSpice netlist of the system. A behavioural linear model is taken for the servo valve (the physical parameters are not known). An accelerometer is used for measuring the acceleration. A behavioural model (integrator) is used for position measurement.

A spring is added to model the flexibility between the load and the cylinder, and between the cylinder frame and the ground. The gains of the amplifier and of the position feedback are calculated using classical control techniques.

```
* Project DYNHYDR
* Powerview Wirelist Created with Version 5.1.2
* Inifile   : wspice.ini
* Options   : -p -f -n
* Levels    :

VDES     VDES  0 PWL ( 0 0 75MS .2462 .325 .2462 .4 0 )              ; desired velocity
X15      VDES XDES POS PARAMS: X0=0                          ; desired position
XAMP     XDES XD  AMP  PARAMS: K={KX}                        ; gain
X17      XD  AS  A1  SUB2                                    ; servo amplifier
X18      A1  XS  A2  SUB2
X0       A2  I    AMP  PARAMS: K={KAMP}
XVALV    A   B   I   SERVALV  PARAMS: KQ={KQ} PSCH=1MEG PS=10MEG PR=1E5 KP=0       ; valve
XCYL     A   B   P   C   CYLINSYM0  PARAMS: A={A} BETA={E} RHO=1N M={MP} W=0
+ B=1N VR1=1U VR2=1U X0=0 P01=1E5 P02=1E5 DIR=1 CEP1=1E-21 CEP2=1E-21
+ CIP={GAB} PSCH=1MEG PEXT=0 L={H} KS={FS} BS=1N V0=0 PREF=1E5         ; cylinder
XCON     C   MASS PARAMS: M={MC} V0=0                        ; mass frame
XCK      C   0  SPRING0 PARAMS: K=100G                       ; flexibility frame
XK1      P   L   SPRING0 PARAMS: K=100G                      ; flexible link cyl - load
XMASS    L        MASS PARAMS: M={M} V0=0                    ; mass load
XDAMP    L   0   DAMPER PARAMS: B={BM}                       ; damping
IFEXT     0   L   DC {-FEXT}                                 ; external force
XACC     L   AS  ACCMETER0  PARAMS: KAMP={KA} KM=1G BM=1K M=1M MH=1F ; acceleration meas.
XPOS     L   XL  POS PARAMS: X0=0
XS       XL  XS  AMP  PARAMS: K={KX}                         ; position meas.
.PROBE
.OPTIONS LIMPTS=2000 ITL1=40 ITL2=20 ITL4=10 ITL5=0 ABSTOL=1PA CHGTOL=.01PC
+ CPTIME=1E6 GMIN=1E-12 RELTOL=.001 NUMDGT=6 VNTOL=1UV TNOM=27
.TRAN 1M .05 0 .1M UIC
.INC "dynhydr.par"
.LIB mecano.lib
.END
```

Figure 44: PSpice netlist of the electro-hydraulic system

Fig. 45 shows the simulation results. Due to the compressibility of the fluid, high frequency pressure oscillations occur during opening and closing of the valve.

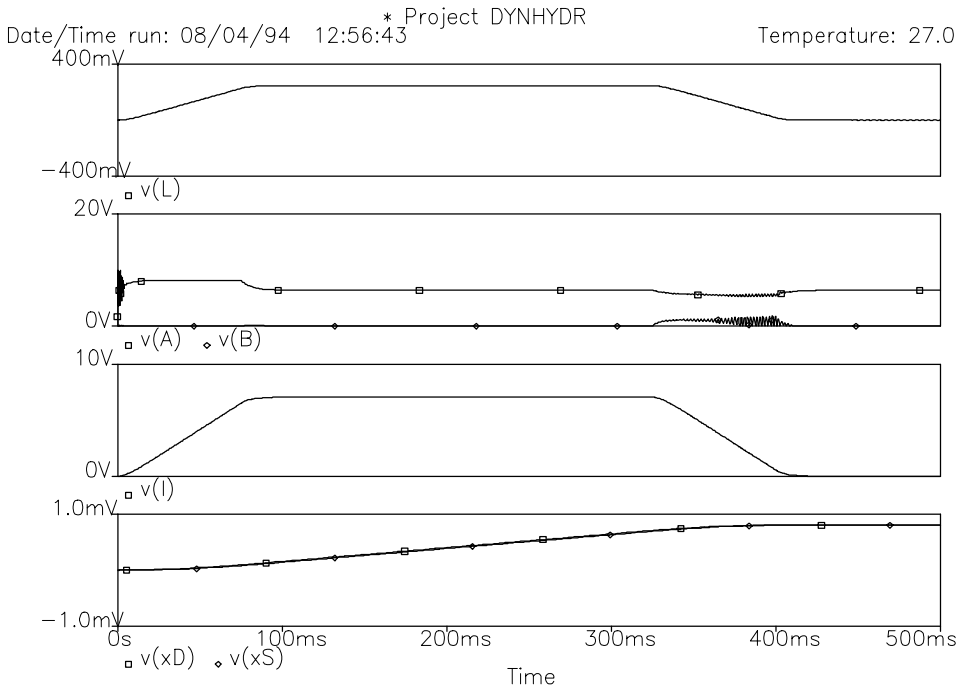


Figure 45: Simulation results of the electro-hydraulic system : Velocity of the load; Pressure on both cylinder sides; Input to the servo valve ; desired and actual position.

5.5. Field Controlled Induction motor

This example discusses a 3-phase induction motor with field control. The applied control is the indirect field control method, as described in [19]. The control is modelled on an analog behavioural level. The induction motor is driven by a current controlled converter, which is also modelled on a behavioural level.

5.5.1. Field Coordinate Transformation

The aim of field control of an induction motor is to decouple magnetic flux and torque, so that the motor can be controlled as a separately excited DC-motor [19]. In the field-orientated approach, both the stator and rotor variables are transformed to a coordinate system (the field coordinate system), which rotates with the magnetising rotor current i_{mR}, which is defined as:

$$\vec{i}_{mR} = i_{mR} e^{j\rho} = \vec{i}_s + \frac{L_r}{M}\vec{i}_r e^{j\varepsilon} \tag{69}$$

with ρ the angle between the magnetizing current and the fixed coordinate system. The torque relation Eq. (54) becomes then:

$$T_g = \frac{N_p M}{3}\mathrm{Im}\left[\vec{i}_s\left(\vec{i}_r e^{j\varepsilon}\right)^*\right] = \frac{N_p M}{3L_r} i_{sq} i_{mR} \tag{70}$$

with i_{sq} the quadrature component of the stator current in the field coordinate system. The relation for the rotor current Eq. (52) becomes [19]:

$$T_r \frac{d\vec{i}_{mR}}{dt} + \left(1 - jT_r \frac{d\varepsilon}{dt}\right)\vec{i}_{mR} = \vec{i}_s \tag{71}$$

with $T_r = L_r/R_r$ the rotor constant. From this equation, two equations can be derived, which give the magnetizing current and the angle ρ:

$$\frac{di_{mR}}{dt} = \frac{1}{T_r}\left(i_{sd} - i_{mR}\right) \tag{72}$$

$$\frac{d\rho}{dt} = \frac{d\varepsilon}{dt} + \frac{i_{sq}}{T_r i_{mR}} \tag{73}$$

The angle ρ is defined from Eq. (73), and then used to derive the currents i_{sd} and i_{sq} from the currents i_{sa} and i_{sb}, which are defined in a local coordinate system, and derived (with Eq. 55) from the measured currents i_{s1}, i_{s2} and i_{s3}:

$$\begin{aligned} i_{sd} &= i_{sa}\cos\rho + i_{sb}\sin\rho \\ i_{sq} &= -i_{sa}\sin\rho + i_{sb}\cos\rho \end{aligned} \tag{74}$$

In Eq. (73) the magnetizing current occurs in the denominator, which can lead to problems. In PSpice, the magnetising current is derived from a system of differential equations, which is derived from Eq. (69):

$$\begin{aligned} T_r \frac{di_{mRa}}{dt} &= i_{sa} - i_{mRa} - T_r \frac{d\varepsilon}{dt} i_{mRb} \\ T_r \frac{di_{mRb}}{dt} &= i_{sb} - i_{mRb} + T_r \frac{d\varepsilon}{dt} \omega i_{mRa} \end{aligned} \tag{75}$$

with indices *a* and *b* indicating coordinates in a fixed coordinate system. In the field coordinate system yields: i_{mRq}=0, and:

$$\begin{aligned} i_{mR} &= \sqrt{i_{mRa}^2 + i_{mRb}^2} = i_{mRa}\cos\rho + i_{mRb}\sin\rho \\ 0 = i_{mRq} &= -i_{mRa}\sin\rho + i_{mRb}\cos\rho \end{aligned} \tag{76}$$

5.5.2. Field Control

The indirect field control method [19], which is described here, uses stator current and velocity measurements. The control consists of a torque and a flux control. The torque control loop defines the desired current i_{sqRef}, the flux control i_{sdRef}.

The torque control loop consists of a PI-controller that compares the velocity with the actual velocity. The output (desired torque) is limited to the maximum possible torque at the measured velocity (using field-weakening). The desired torque is compared with the motor torque, as calculated with Eq. (75) and sent through a PI-controller to generate the desired current i_{sqRef}.

The flux control loop compares the calculated magnetising current i_{mR} with the desired magnetising current i_{mR}, which is proportional to the maximum torque. The difference is sent through a PI-controller, and gives the desired current i_{sdRef}.

These desired currents are transformed to a fixed coordinate system, by using the inverse transformation of Eq. (74). The desired 3-phase currents are then derived by using Eq. (56).

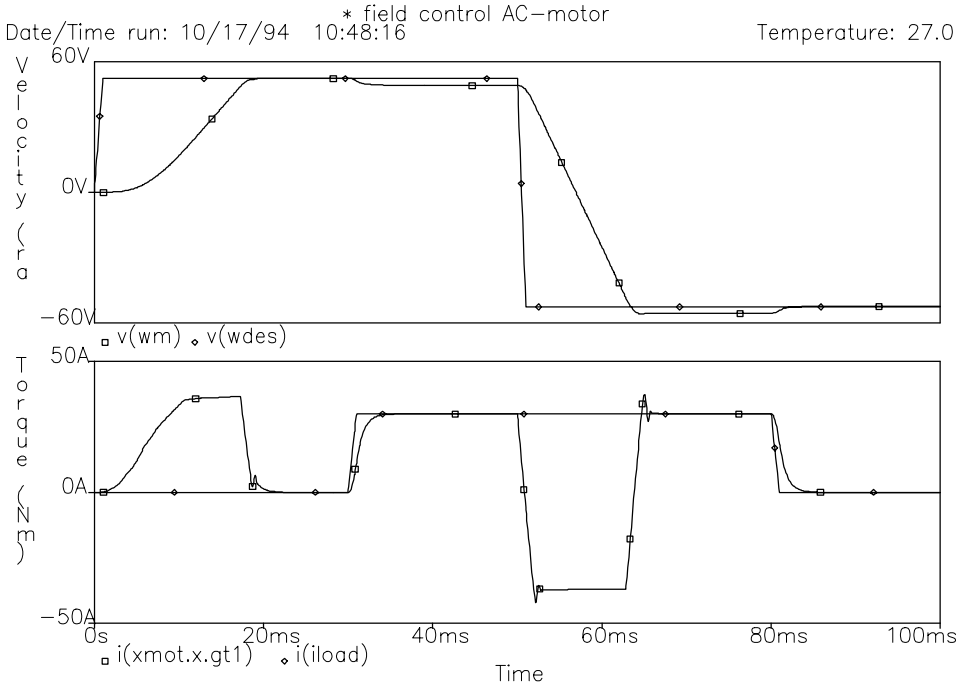


Figure 46: PSpice results of the field control simulation: actual (ω_m) and desired (ω_{des}) motor velocity; Generated torque and load torque.

Fig. 46 shows the results. The netlist is:

```
* field control AC-motor
* book Leonhard p.215 ev. Fig.12.13
.lib mecano.lib
.options noecho

* calculation of the magnetizing current vector in a fixed coordinate system,
* starting from the stator currents in a fixed coordinate system
.subckt magcur isa isb w imra imrb params: Tr=1  Np=2
c1 z1 0 {Tr} ; ic={x10}                          ; z1 : temp. variable
g1 0 z1 value={v(isa)-v(z1)-(Np/2)*Tr*v(w)*v(z2)}
r1 z1 0 1T

c2 z2 0 {Tr} ; ic={x20}
g2 0 z2 value={v(isb)-v(z2)+Tr*(Np/2)*v(w)*v(z1)}
r2 z2 0 1T

rin1 isa 0 1T                                    ; output resistances
rin2 isb 0 1T
rin3  w  0 1T
rout1 imra 0 1T
```

```
rout2 imrb 0 1T
rout3 imra imra1 1n
rout4 imrb imrb1 1n
ex1 imra1 0 z1 0 1                          ; copy voltages
ex2 imrb1 0 z2 0 1
.ends
*--------------------------------------------------------------------------------------------
* Calculation of cos(rho) and sin(rho) of the magnetizing current vector
.subckt rhocalc imra imrb imr cosq sinq
xhyp imra imrb imr hypot

g1 0 x1 value={v(imra)*(1-v(x1))+v(imrb)*v(x2)-v(imr)}
r1 0 x1 1T
g2 0 x2 value={-v(imra)*v(x2)+v(imrb)*(1-v(x1))}
r2 0 x2 1T

e1 cosq1 0 value={1-v(x1)}
e2 sinq1 0 x2 0 1
rout1 cosq1 cosq 1n
rout2 sinq1 sinq 1n
rout3 cosq 0 1T
rout4 sinq 0 1T
.ends
*--------------------------------------------------------------------------------------------
* field control=velocity control
* nodes : wdes : desired velocity
*  w    : measured velocity
*  is1,is2 : measured current (as velocity signals)
*  is1ref,is2ref,is3ref : output currents (as velocity signals)
* params: Rr,Lr,Rs,M, Np : motor parameters
*  rho0 : initial magnetization angle
*  Trat : rated torque (also : m0 in leonhard control scheme )
*  wrat : rated velocity
*  Imrat: rated magnetization current
*  Kpvel, Kivel : velocity control Kp and Ki
*  Kptrq, Kitrq : torque control Kp and Ki
*  Kpflux, Kiflux : flux control Kp and Ki
*  Ilim : current limit.
.subckt field_ctrl wdes w is1 is2 is3 is1ref is2ref is3ref
+ params: Lr=1f Rr=1f rho0=0 M=1f  Np=2
+  Kpvel=1 Kivel=0 Kptrq=1 Kitrq=0 Kpflux=1 Kiflux=0 Kcurr=1
+  Trat=1 wrat=300 Imrat=10  Ilim=50A
+         ph2dq=0.816496580928      ; conversion factor from 2ph to 3ph

x3ph2ab  is1 is2 is3 isa isb ph3toph2 params: ph2dq={ph2dq} ; transfo. from 3-phase to 2-phase
ximr    isa isb w imra imrb magcur params: Tr={Lr/Rr} Np={Np}; magnet. current in fixed co.sys
xrho    imra imrb imr cosq sinq rhocalc        ; angle and amplitude of magnetizing current
xab2dq  isa isb isd isq cosq sinqn cotra2d     ; coordinate transform to field co.sys.
xsinneg sinq sinqn inv                         ; invert sine

xweak   w tmax fieldwea params: wrat={wrat} yrat={Trat}    ; maximum torque (field weakening)

xvelo w wdes mdref1 pirel params: Kp={Kpvel} Ki={Kivel} ; pi-control   ; velocity control
xtlim   mdref1 mdref tmax limitvas             ; limit to maximum torque
xprod imr isq md mul2 params: k={(2/(3*ph2dq*ph2dq))*(Np/2)*M/(Lr/M)}  ; torque calculation
xtorq md mdref isqref1 pirel params: kp={Kptrq} ki={Kitrq} ; torque controller

ximref tmax imref amp params: K={Imrat/Trat}    ; desired magnetizing current
xflux  imr  imref isdref1 pirel params: Kp={Kpflux} Ki={Kiflux}        ; flux controller

xlimq isqref1 isqref limit params: vmin={-Ilim} vmax={Ilim}            : limits on currents
xlimd isdref1 isdref limit params: vmin={-Ilim} vmax={Ilim}

xdq2ab isdref isqref isaref isbref cosq sinq cotra2d       ; coord. transform to fixed co.sys.
xab2ph3 isaref isbref is1ref is2ref is3ref ph2toph3 params: dq2ph={2/(3*ph2dq)} ; 3-ph. cur.
.ends
*--------------------------------------------------------------------------------------------
* actual model
*
xconv is1ref is2ref is3ref v1 v2 v3 drvi3p params: Vs=380  ; converter

xseni1 v1 p1 is1 curres0                       ; current sensors
xseni2 v2 p2 is2 curres0
xseni3 v3 p3 is3 curres0

* motor model
xmot p1 p2 p3 wm 0 acind3p params: Rs={Rs} Rr={Rr} M={M} Ls={Ls} Lr={Lr}
+    Jm={Jm} Np={Np} w0={w0}
xtach wm 0 wt tacho0                           ; velocity sensor
```

```
x wdes wt is1 is2 is3 is1ref is2ref is3ref field_ctrl      ; field control
+ params: Rr={Rr} Lr={Lr} M={M} Np={Np}
+  Trat=40 wrat=150 Imrat=10 rho0=0
+  Kpvel={Kpvel} Kivel=0  Kptrq={Kptrq} Kitrq=0  Kpflux=1 Kiflux=0

vdes wdrpm 0 pwl (0 0 1m 500 50m 500 51m -500) ;  desired velocity
xdrpm wdrpm wdes rpm2rads

iload wm 0 pwl (0 0 30ms 0 31ms 30 80ms 30 81ms 0)         ; load

.param Rs=0.490 Rr=2.364 Ls=0.1468 Lr=0.1468 M=0.1312 Np=4 Jm=8.2e-3 Vs=380
.param w0=0 ; {.9 *2* 50*3.14159265/Np}
.param Kpvel=10 Kivel=10
.param Kptrq=10 Kitrq=10

.tran 10us 100ms 0 .1ms uic
.options vntol=1m abstol=1u reltol=1m itl5=0
.probe
.end
```

Figure 47: PSpice netlist for the field controlled motor

CHAPTER 6.

DESIGN INTEGRATION

Design of mechatronic systems is a complex process. The previous chapters described how a single CAE-tool can be used for modelling and simulation of the complete multitechnical system at the physical level. The CAE-tool does however not have all the functions, which are desired during the conceptual and the physical design. At the conceptual design level, different design alternatives are compared and the optimal components are selected, mostly by the use of some design rules. At the functional design level, the aim is to design the controller for optimum dynamical behaviour. The tools that are used at the conceptual design level have mostly no advanced possibilities for simulation, whereas the physical and functional design tools have no possibilities to select commercial components efficiently. By integrating conceptual and physical (or functional) design, different design alternatives can be verified rapidly. For integration of the design process, models and design data can be passed from one design tool to another.

This chapter will describe how models can be transferred from one design level to another, or from one CAE-tool to another. Three tools have been used during this design project: ServoPlan for the conceptual design of electric drives, MATLAB/SIMULINK for the functional design of the controller, and PSpice for the physical design.

6.1. ServoPlan: Conceptual Design of Servo Drives

The aim of the conceptual design level is to define the architecture of the system, by comparing different design alternatives, and by selecting the basic system components. At the conceptual design level, rules are used to select the various components.

6.1.1. Description of ServoPlan

ServoPlan [14] is a program for the conceptual design of electric drives. ServoPlan has been developed at VTT Automation in co-operation with component suppliers (ABB Industry, SKS-tekniikka Oy and Refimex Oy) and end-users (Cimcorp Oy and Lillbackan konepaja).

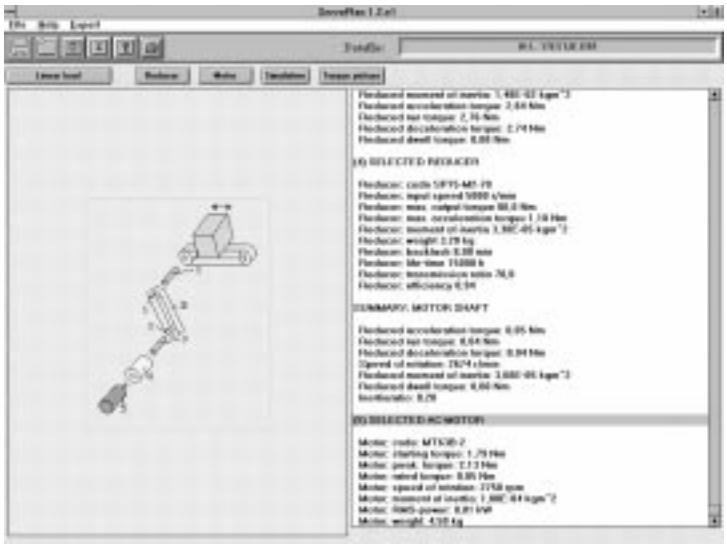


Figure 48: User Interface of ServoPlan1.2 with drive system, and the input and calculated values.

ServoPlan is a Windows based program, and is programmed in Visual Basic. Figure 48 shows the user interface. The load, the velocity profile and the parameters of the power transmission components are interactively entered. The following power transmission components are provided: gear, rack-pinion, belt drive and ball screw. ServoPlan uses a set of rules to select AC-, DC-motors and reducers from a database. All databases are in dBase III format. The parameters in the motor and reducer databases are catalogue data, e.g. maximum and rated torque, rated velocity, moment of inertia,...

ServoPlan has also some simple simulation routines to check the starting of the motor. During simulation, the complete mechanical transmission system is treated as a single inertia, and no external forces are applied.

Since ServoPlan is mainly aimed at the selection of the motor and reducer, the controller is not modelled. Only for the simulation of DC-motors a PID velocity control loop is added.

6.1.2. Export Module

The simulation algorithm, that is used by ServoPlan is very simple: the integration rule is a Euler method with a fixed time step. In order to check the behaviour more accurately, a more elaborate CAE-tool like MATLAB or PSpice has to be used. The basic version of ServoPlan can however only print the global system configuration, and has no possibilities to export the designed system to other CAE-tools. Since the source code of ServoPlan was available, a module, called EXPORT has been added to the original code.

Unfortunately, there are no standard languages for the exchange of simulation models. Each CAE-tool uses its own language to defined models. A different conversion routine has therefore to be written for each CAE-tool. In EXPORT, the system can be transferred to the other tools, which were used during the project, i.e. PSpice and MATLAB/SIMULINK.

The model, which is generated by EXPORT, consists of macromodels, their connections and the model parameters. This requires that in each CAE-tool a library of these models is available. The library for PSpice has been described in the previous chapter. The library that has been developed for MATLAB will be discussed in section 6.3.3.

The module can also add a controller to the model, in order to have directly a complete model of the system.

6.1.2.1. Program Flow

The module has the following steps:

- Selection of the Export format. MATLAB and PSpice are at this moment included.

  MATLAB/SIMULINK: the output file is a script file with the SIMULINK model.

  PSpice: the output file is a PSpice netlist

- Input of the output file name
- Selection of the transient analysis type: step function or velocity profile. If the step function is selected, the starting of the motor and the control loop is tested. If the velocity profile is selected, the velocity profile and the forces, which have been input to ServoPlan, are used.
- Semi-automatic or interactive input. In semi-automatic mode, the program reduces requests to the user for input to a minimum. In interactive input mode, the user can change most of the parameters and enter the parameters that do not occur in the standard conversion routines.
- Writing the initialisation commands to the output file, as e.g. the name of the model library, transient simulation parameters,...
- Export of the models for motor, mechanical transmission and load.

- Selection of the Controller. Position control and velocity control have been included. Feedback sensors can be selected at the motor or at the load. If net input is selected, the motor is driven by a DC- or AC-supply. If no controller is selected, only motor, transmission and load are written to the output file.
- Export of the models for the controller.
- Closing of the output file.

6.1.2.2. Model Window

For each model, a similar model form (or window) is used. Fig. 49 shows the model window for a DC-motor and PSpice output.

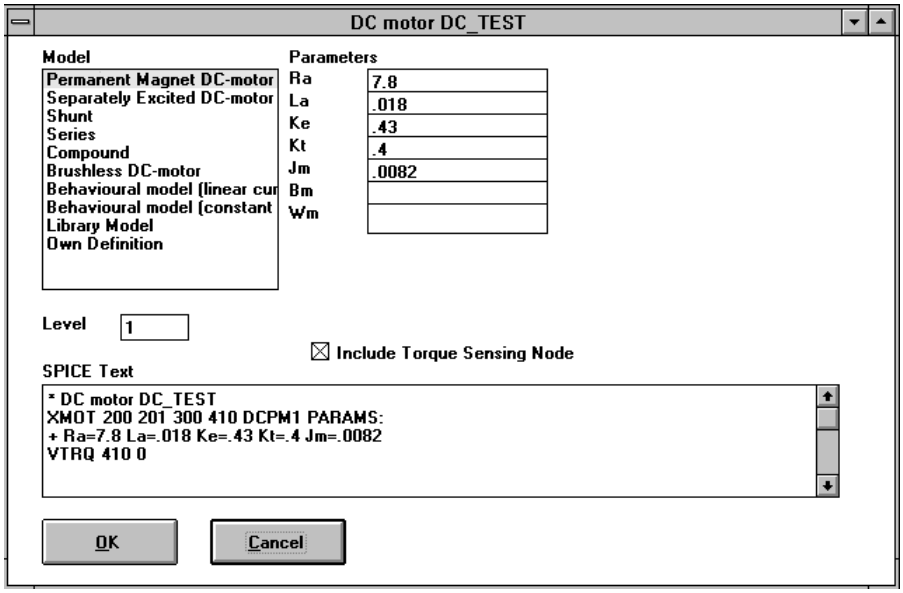


Figure 49: Model Window for DC-motor with PSpice output

The controls on the window are :

- A list with the possible models:

  Parametric Component Model: the list contains the various models that are available in resp. the PSpice or MATLAB library.

  Library Model : is available for PSpice output of DC- and AC-motors and gears. With the program CreaLib, which will be discussed in section 6.2.4., the dBase databases in ServoPlan can be converted in directly to PSpice libraries. At this selection, EXPORT asks for the name of the library, and lists the names of the subcircuits in the library in the window, from which the user can choose. No parameters can be selected.

  Own Definition : the user can enter his own model. The program indicates the names of the nodes, which have to be used.

- The level. The level is an integer number that is appended after the model name. If the PSpice library has models for several accuracy levels, the level can be changed between 0 (ideal) and the maximum level number (most complex).
- The parameter names and values. The module converts the model, which is used in ServoPlan, to a physical model, and lists the parameters of the physical model on screen. The parameters, which are used by ServoPlan, correspond mostly directly to parameters of the physical model; in the other cases a conversion routine is used to calculate the physical model parameters from the parameters in ServoPlan. For e.g. DC-permanent magnet motors, the parameters in ServoPlan correspond

directly to the parameters of the PSpice model, except for damping b_m and the friction W_m, which are not used in ServoPlan. The user can change the parameter values.

- The output text in MATLAB or PSpice format. The text is updated automatically if the model or a parameter value is changed. The user can also change the text. When the "OK" button is pressed, the content of the output text box is written to the output file.
- Additional controls, dependent on the model type and the output format. For AC- and DC-motors, a torque probe can be added to the PSpice file. For the complex belt drive models, the type of pretensioning can be entered. For MATLAB output, text boxes are added to name the output variables.; if these boxes are filled, the program generates additional blocks to output the variables.

6.1.2.3. Coding of the model window

The model window is programmed so, that it can also be used by other programs (e.g. VarComp, which will be discussed in section 6.7.2.). Due to the limited possibilities to handle user-defined types in Visual Basic, one global variable `Xmodel` is used to represent all the components. `Xmodel` is a user-defined type `model`, and consists of the following fields:

`name`	name of the component (the prefix X is added for PSpice subcircuits)
`type`	built-in model or subcircuit. The output format for both MATLAB and PSpice differs if the model is built-in (PSpice primitive, SIMULINK built-in blocks) or defined in a user-written library.
`model`	the name of the model
`max_level`	the highest accuracy level model
`level`	the actual accuracy level
`n_in`	the number of input nodes
`node_in`	names of the input nodes
`n_out`	number of output nodes
`node_out`	the names of the output nodes
`n_par`	the number of parameters
`par_name`	the names of the parameters
`par_val`	the values of the parameters
`comments`	comments, which are written to the output file.

Three files are used for programming of the model window: the Visual basic file for the form, the Basic file MODEL.BAS and the application dependent Basic file APPL_yyy.BAS. MODEL.BAS defines the global variables used by the model window routines and generates a “blank” model for `Xmodel`. The “blank” model defines the variables that are fixed for the model (i.e. the number of input and output nodes, model name, parameter names). At selection of the model `xxx` in the model window, an application specific routine `appl_model_xxx` is called (in the file APPL_yyy.BAS), which performs a call to the routine `model_xxx` in MODEL.BAS. This routine generates a “blank” `Xmodel`; the routine `appl_model_xxx` then fills in the node names and converts the parameters used in ServoPlan to component model parameters.

6.1.2.4. Extension to other CAE-tools

The module EXPORT can be extended to other tools than PSpice and MATLAB. For each other tool, a library for the components, which are used by ServoPlan (i.e. reducers, AC- and DC-motors) has to be developed.

In the module MODEL.BAS, the routines for the “blank” model have to be adapted to match the model of the component in the CAE-tool. If the model uses other parameters than the models in MATLAB or PSpice, the conversion routines have to be changed. An additional form has to be generated for the initialisation of the model description file in the CAE-tool. Routines have to be

developed for the output of the model and the connections between the models in the model description file.

The physical design tool could be further integrated in the conceptual design tool by using the other CAE-tool directly as simulator: after pressing the button “Simulation”, the model is automatically converted, the simulation started, and the post-processor called. A user interface has to be provided between ServoPlan and the post-processor, to allow to view the desired results on screen (without accurate knowledge of the model in the simulator).

The technique, which is used by the module EXPORT could be applied to other conceptual design tools than ServoPlan. The main routine generates the sequence in which the different components in the conceptual design tool are written to the description file. For each component, a model window is generated, in which the ports to the models (i.e. the connections with previously generated models) are the parameters.

6.2. From Conceptual to Physical

6.2.1. Physical Model of the Electric Drive

Due to the selection criterion for across and through variables for the different energy domains, the topology of the model in PSpice resembles the physical system. The topology of the physical model of the electric drive corresponds to the topology of the graphical model in ServoPlan. Fig. 50 compares the ServoPlan input with a ViewDraw model of the same system. Due to this analogy, the physical model can be constructed by converting component by component. The nodes in the physical model correspond to mechanical connection points.

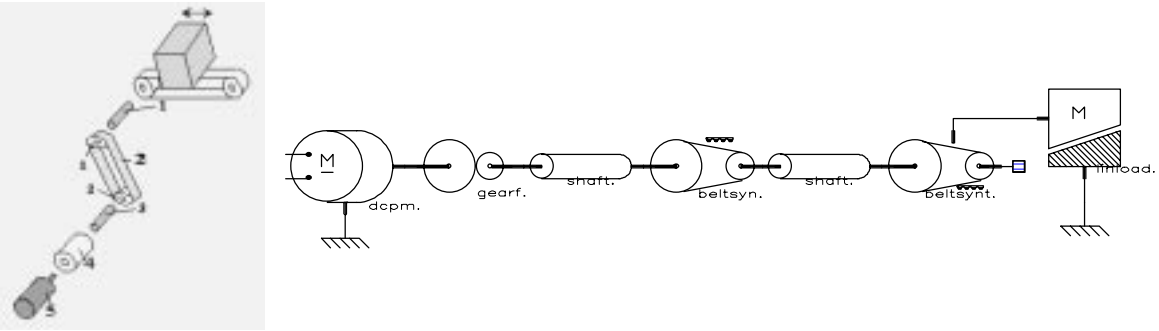


Figure 50: The graphical model in ServoPlan and the equivalent ViewDraw model.

If we want to generate a model of the complete motion control system, we also need to include a model of the control system, which is not included in ServoPlan

6.2.2. Physical Model Parameters from Catalogue Data

Conceptual models are based on catalogue data, whereas physical models are based on physical properties. For instance, the conceptual model of the AC induction motor uses e.g. the rated and the maximum torque, whereas the physical torque uses rotor and stator resistances and inductances.

If the physical model of a product is available at an early stage of the design, different design alternatives can be verified rapidly and decisions on the selection of the other components can be made early. If the physical parameters cannot be measured, they have to be derived from the catalogue data. Conversion routines have to be developed for each different model, in order to calculate the parameters for the physical model from the catalogue data. These conversion routines make often rough approximations. The physical model that is derived in such way may therefore not be accurate, and has to be handled with care.

This section will describe how the models in ServoPlan (DC-motors, AC-motors and reducers) are converted to physical models.

6.2.2.1. DC-motors

The databases for DC-motors include the torque and the gain constant, the armature resistance and inductance, and the rotor inertia. These parameters return all in the physical model of the DC-permanent magnet motor. All DC-motors are modelled as permanent-magnet DC-motors. Major motor manufacturers use these models in their guidelines for motor selection [50].

6.2.2.2. AC-motors

ServoPlan treats all AC motors as 3-phase induction motors. The rotor and stator resistances and inductances are derived from the catalogue data with a rather simple conversion routine. The conversion is based on the following equations:

$$T_{rat} = \frac{3N_p}{4\omega_e} \cdot \frac{V_e^2}{\left(R_s + \frac{R_r}{s_{rat}}\right)^2 + X_\sigma^2} \cdot \frac{R_r}{s_{rat}} \qquad T_{peak} = \frac{3N_p}{8\omega_e} \cdot \frac{V_e^2}{R_s + \sqrt{R_s^2 + X_\sigma^2}}$$

$$\text{with} \quad \begin{aligned} \omega_e &= 2\pi f \\ V_e &= V_s\left(1 - \frac{X_1}{X_m}\right) \\ s_{rat} &= n_{syn}\left(1 - \frac{n_{rat}}{n_{syn}}\right) \\ n_{syn} &= \frac{2}{N_p}\frac{30\omega_e}{\pi} = \frac{120f}{N_p} \\ X_\sigma &= X_1 + X_2 \\ X_1 &= \omega_e\left(L_s - M\right) \\ X_2 &= \omega_e\left(L_r - M\right) \\ X_m &= \omega_e M \end{aligned} \tag{77}$$

The first 2 equations give resp. the rated torque T_{rat} at rated speed n_{rat} rpm, and the peak torque T_{peak}. The following assumptions are made:

$$\begin{aligned} L_s = L_r = 1.05M \quad &\Rightarrow \quad X_1 = 0.05X_m \\ R_s = 0.05X_\sigma \end{aligned} \tag{78}$$

With these assumptions the rated torque can be written as:

$$T_{peak} = \frac{3}{8\frac{\pi n_{syn}}{60}} \cdot \frac{\left(0.95V_s\right)^2}{\left(0.05 + \sqrt{1.0025}\right)X_\sigma} \quad \Rightarrow \quad X_\sigma = \frac{3}{8\frac{\pi n_{syn}}{60}} \cdot \frac{\left(0.95V_s\right)^2}{\left(0.05 + \sqrt{1.0025}\right)T_{peak}} \tag{79}$$

From the reactance X_σ the stator resistance R_S and the inductances M, L_S and L_r are calculated:

$$X_\sigma = \omega_e\left[\left(L_s - M\right) + \left(L_r - M\right)\right] = \omega_e\left[2 \cdot 0.05M\right] \quad \Rightarrow \quad M = 10\frac{X_\sigma}{\omega_e} \tag{80}$$

The rotor resistance is approximated as: $R_r = s_{peak}\sqrt{R_s^2 + X_\sigma^2}$, with s_{peak} the slip, corresponding to the maximum torque. At the rated speed, the torque is approximately proportional to the slip, or:

$$s_{peak} = s_{rat}\frac{T_{peak}}{T_{rat}}$$

This leads to: $R_r = s_{rat}\frac{T_{peak}}{T_{rat}}\sqrt{R_s^2 + X_\sigma^2} = s_{rat}\frac{T_{peak}}{T_{rat}}\sqrt{1.0025}X_\sigma$

These assumptions are rather rough. The calculated theoretical starting torque does not correspond with the catalogue data. Another (iterative) approach has been used, which uses also the value of the starting torque and the power factor (cosφ). It is however not always possible to find values for rotor and stator resistances and inductances, which match all these parameters.

6.2.2.3. Behavioural motor models

The behavioural motor models in the PSpice library use the parameters stored in the dBase database as parameters. These motor models represent the combination of motor, motor control and power amplifier. The parameters for the control (PI-velocity control) are entered as parameters to the model.

6.2.2.4. Reducers

The reducer is modelled as a single gear pair. The two gears are supposed to have the same width *b* and material density ρ. The radius of the output gear equals to the product of the radius of the input gear and the transmission ratio *i*. The inertia of each of the gears is calculated of the combined inertia at the entry shaft. The ratio of the inertia is:

$$\frac{J_2}{J_1} = \frac{m_2 r_2^2}{m_1 r_1^2} = \frac{\left(\pi \rho b r_2^2\right) r_2^2}{\left(\pi \rho b r_1^2\right) r_1^2} = \frac{r_2^4}{r_1^4} = i^4 \tag{81}$$

The total inertia at the input shaft J_{tot} equals: $J_{tot} = J_1 + \left(\frac{\omega_2}{\omega_1}\right)^2 J_2 = \left(1 + i^2\right) J_1$. For higher transmission ratios this assumption is however not valid. If there is no backlash, it is not necessary to divide the inertia over the two gears.

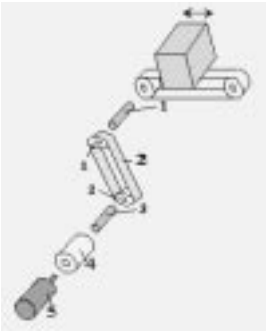
The efficiency is a parameter that also occurs in the physical model. In the catalogue data, the efficiency is entered at four different velocities. A model has been added to the PSpice library for a gear pair with variable efficiency.

6.2.2.5. Mechanical transmission and load

The input parameters of ServoPlan, correspond mostly to parameters that occur in the physical PSpice model. Properties, which are calculated by ServoPlan are sometimes parameters in the PSpice model, e.g. the inertia of gears and belt pulleys, which are calculated from the diameter and mass, which are entered interactively. In some cases, models have been added to the PSpice library to convert a ServoPlan input element directly to a PSpice model (e.g. the linear load model, and time dependent forces).

6.2.3. Conversion from ServoPlan to PSpice

The module EXPORT, which has been added to ServoPlan, converts the input data and the calculations from ServoPlan to a PSpice netlist. Due to the analogy between the physical model and the graphical model in ServoPlan (cf. Fig. 50), the conversion is rather straight-forward : component by component is converted, and the nodes are numbered accordingly.



```
* Example for conversion from ServoPlan to PSpice
.LIB mecano.lib
.TRAN .050001 5.0001 0 .050001 UIC
.PROBE
* DC motor 443.1.20
XMOT 200 201 300 410 DCPM1 PARAMS:
+ Ra=4.1 La=.023 Ke=.67 Kt=.67 Jm=2.00E-03
VTRQ 410 0
VGND 201 0
* Reducer SP60-M1-10
XRED 300 301 GEARF2 PARAMS:
+ N1=1 N2=-10 J1=1.287129E-07 J2=1.287129E-03 bl=1.745333E-
03
+ effic=0.97
```

Figure 51: ServoPlan model and start of corresponding PSpice netlist

EXPORT can add a velocity or position controller to the system. The feedback signal can correspond to the motor or load velocity (resp. position).

The input data of ServoPlan can however not be converted to a ViewDraw model. ViewDraw schematics are stored as ASCII-files, but no functions are available to generate these files from external programs. Moreover, the nets, which link the different symbol pins in these files, are determined by their initial and final position coordinates, not by the pin labels. If the size of a symbol or the position of the pins change, the pins have to be connected again manually. This makes it almost impossible to generate ViewDraw files from an external program.

6.2.4. Generation of PSpice Libraries from Catalogue Data

The conversion routines that are used to generate physical models from the data in ServoPlan can also be used to directly generate PSpice libraries from the dBase libraries, used by ServoPlan.

The program CreaLib has been generated for the conversion of the dBase libraries to PSpice libraries. Currently, the following models can be output: DC permanent magnet, AC 3-phase induction motor, behavioural motor models and gear models. Figure 52 shows the user interface for CreaLib.

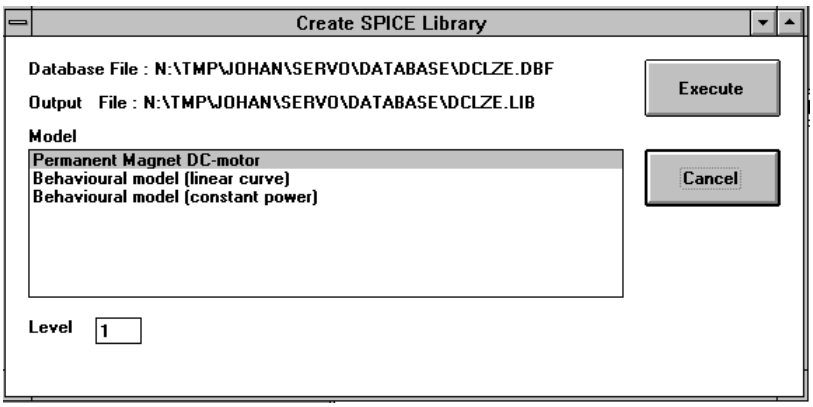


Figure 52: User interface of CreaLib

Figure 53 shows a part of the library in ServoPlan and the resulting PSpice library. The component name is changed so, that the subcircuit name contains only characters that are allowed by PSpice, e.g. blanks are removed, "-" becomes "_" and "," becomes "*".

	Code	Supplier	Manufacturer	Continuous torque	Peak torque	Torque constant	Armature resistance	Speed	Max. speed	Moment of inertia
0	GFRK 090-22/1,7	REFIMEX OY	LENZE	5.7	9.12	0.69	4.53	2850	4500	0.004
1	GFRK 090-22/1,8	REFIMEX OY	LENZE	5.7	9.12	0.39	1.48	3050	4500	0.004
2	GFRK 100-22/2,7	REFIMEX OY	LENZE	8.6	13.76	0.7	1.04	3000	4500	0.0061
3	GFRK 100-22/2,5	REFIMEX OY	LENZE	8.6	13.76	0.46	0.755	2800	4500	0.0061
4	GFRK 112-22/4,7	REFIMEX OY	LENZE	14.9	21.84	1.2	2.55	3050	4500	0.0124

(a) dBase data in ServoPlan

```
* PSpice Library created by CreaLib v.1.0
* VTT/KAU 18.11.1994
* Database file : J:\COMM\DCLZE.DBF
* Models converted to PSpice model for : Permanent Magnet DC-motor
* (model dcpm level 1 )
*****************************
* GFRK 090-22/1,7       : Permanent magnet DC-motor
* Manufacturer : LENZE
* Supplier     : REFIMEX OY
.SUBCKT DCGFRK090_22/1*7 1 2 3 4
XMOT 1 2 3 4 DCPM1 PARAMS:
+ Ra=4.53 La=0.0141 Ke=0.69 Kt=0.69 Jm=0.004
.ENDS

* GFRK 090-22/1,8       : Permanent magnet DC-motor
* Manufacturer : LENZE
* Supplier     : REFIMEX OY
.SUBCKT DCGFRK090_22/1*8 1 2 3 4
XMOT 1 2 3 4 DCPM1 PARAMS:
+ Ra=1.48 La=0.0046 Ke=0.39 Kt=0.39 Jm=0.004
.ENDS
```

(b) PSpice library

Figure 53: Conversion from dBase to PSpice library

These libraries can then be used directly when writing or editing the netlist, to verify the behaviour of other motors, or by other schematic editors. All the models in one library have the same number of input nodes and represent the same type of component. Symbols have been provided in the schematic editor ViewDraw for these library models. The attribute MOD of these symbols has to be set to the library model of the respective component.

6.3. From Conceptual to Functional

The translation of the graphical input in ServoPlan to a functional form is not so straight-forward as the generation of the physical model. This section will first discuss how the motor - transmission system can be modelled as a block diagram, consecutively how block diagram libraries have been developed, and then how ServoPlan data are converted to a MATLAB/SIMULINK block diagram.

6.3.1. Modelling of motor-transmission

Block diagrams are unidirectional. Power flow between two components, described by two blocks, can be represented by separating the through and the across variable over two separate signal lines. One of the power variables is modelled as output of the first block and is input to the second block. The second power variable is modelled as a feedback signal: it is output of the second block and input to the first block.

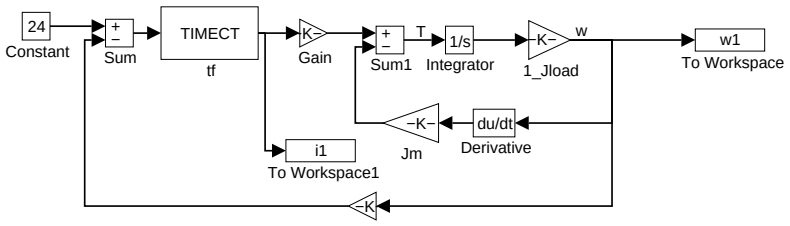
The mechanical transmission is characterised by the impedance Z_L: $\omega_m=Z_L T_m$, with ω_m the motor shaft velocity and T_m the torque available at the motor shaft. This can be modelled in 3 ways in MATLAB

1. The load is modelled as $\omega_m=Z_L T_m$. The inputs to the motor are the voltage V and the velocity ω_m, output the torque T_m.
2. The load is modelled as $T_m=Y_L \omega_m$, with Y_L the load admittance. The inputs to the motor are the voltage V and the torque T_m, output the velocity ω_m.
3. The transmission impedance is included in the model of the motor. The torque equation of the motor is always of the type: $T_m=T_g-Y_m\omega_m$, or $T_g=(Y_m+Y_L)\omega_m$, with $Y_m=J_m s+b_m$ the motor

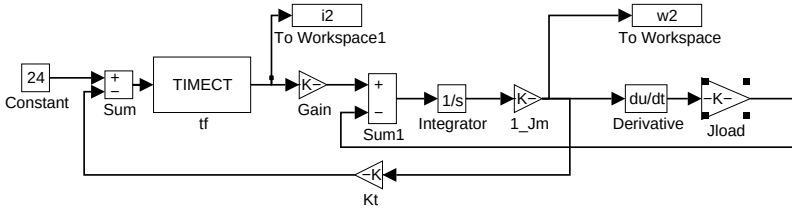
admittance. This last model is the most simple one, but has as disadvantage that the model includes the model of the transmission, so that the conversion is not so straight-forward.

In models 1 and 2 the structure of the block diagram reflects the structure of the system. In the block diagram, one block is provided for the motor, and one for the transmission. The output of the transmission block is input to the motor block. This would allow to translate component per component of the ServoPlan model to the block diagram model. The linking of the blocks is a little bit more difficult than in the ServoPlan-PSpice conversion.

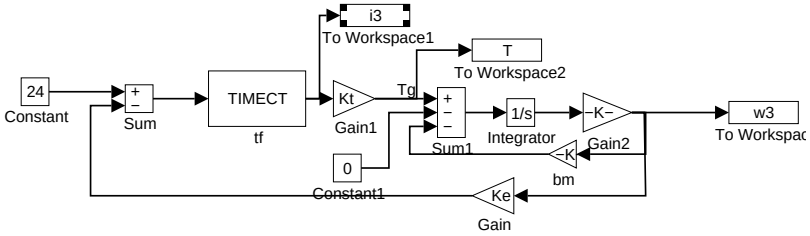
Model 3 does not keep the topology, but the global block diagram is simpler, since the complete transmission and load are included in the motor model.



(a) Torque is output of motor model. $\omega=(1/J_{load}s)T$



(b) Rotational velocity is output of motor model. $T=J_{load}d\omega/dt$



(c) Load inertia is included in motor model

Figure 54: Alternative models for permanent magnet DC-motor with inertial load J_{load}

As an example, the different models are compared for a voltage-fed DC permanent magnet motor with a pure inertial transmission (Z_L=1/ J_{load} s) (Fig. 54). Table 5 gives the number of time steps for a simulation (parameters: R_a=5.5Ω, L_a=6.2mH, K_t=6.2e-3Nm/A, K_e=6.2e-3Vs/rad, J_m=56e-6kgm^2).

The first model worked well for large inertia, but the simulation became unstable for smaller inertia. The second model worked well for small inertia, but became unstable for larger inertia. This instability might be due to the presence of the derivation, which easily produces accuracy problems. This model type, which maintains the topology of the system, is therefore not so interesting. They are however used by Yan and Sharpe [44] for the modelling of mechatronical systems. In their article however, only the motor has an inertia, and their power transmission system is without inertia, so that no derivation is needed.

J_{load}(kgm^2)	Model 1 $\omega=(1/J_{load}\,s)T$	Model 2 $T=J_{load}\,d\omega/dt$	Model 3 inertia in motor model
1e-6	Erroneous	244	244
1e-5	Unstable	241	230
1e-4	251	506	245
1e-3	247	Unstable	250
0.01	248	Unstable	248
0.1	248	Unstable	248

Table 5: Number of time steps for a simulation of DC permanent magnet motor with inertial load (simulation time 1s, tolerance 0.001, minimum time step 1e-5).

The third model provided always good results, and has therefore been selected.

Currently, only inertial transmission loads are included. The transmission inertia is added to the inertia of the rotor. The model of the rotor with and without load is therefore the same.

6.3.2. Textual input in SIMULINK

SIMULINK is an extension of MATLAB, which allows to enter block diagrams in a graphical way. The SIMULINK models are stored as a MATLAB script file. MATLAB provides commands to generate SIMULINK blocks, to set parameters in blocks, and to link block gates. For linking blocks, the names of the output block/port and the input block/port have to be given. This allows to generate SIMULINK models automatically. The EXPORT module of ServoPlan therefore converts the input data to a SIMULINK file.

6.3.3. SIMULINK Libraries

In SIMULINK, high level blocks can be constructed from component models. These blocks are masked, i.e. an interface of a number of inputs, output and a dialogue window is attached to it. The masked blocks can be collected into a library, which can be used to develop the system model.

A small library "**servolib**" has been developed, which contains the models of the motors and drives used in ServoPlan. The model parameters are entered in dialogue windows. SIMULINK dialogue

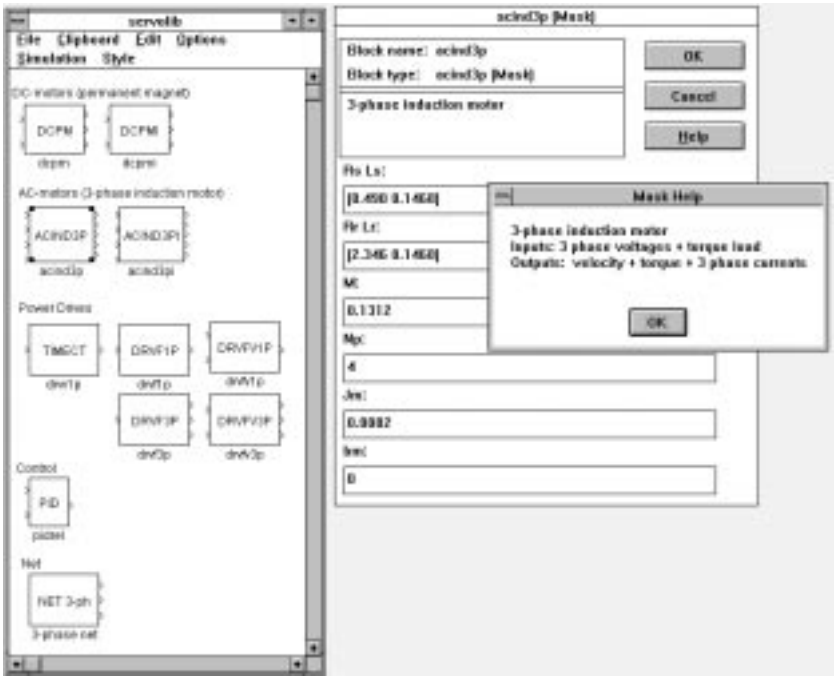


Figure 55: SIMULINK library and dialogue box for 3-phase induction motor.

windows are however limited to 6 input lines, which can however contain arrays. The model parameters are equally spread over the 6 input lines, in which the first input lines contain one model more than the last input lines. Figure 55 shows the SIMULINK library and the dialogue and help window for the 3-phase (voltage controlled) induction motor.

The motor models are based on models that have previously been developed at VTT Automation [17]. For each motor two models have been provided: a model for a voltage controlled motor and a model for a current controlled motor. The motor models have as input the output signal of the power amplifier (one for each phase), and the torque applied at the load. The first output is the motor shaft velocity, the second output is the generated torque. The following output ports correspond to the electric variable that is no input variable (i.e. current output for voltage controlled motors, voltage output for current controlled motors).

The inertia of the transmission is added to the inertia of the rotor. This has as advantage that the user interface in ServoPlan and in SIMULINK is kept simple. If the transmission has flexibility, the motor model must be unmasked, and the block representing the rotor and transmission inertia has to be adapted.

6.3.4. Conversion from ServoPlan to SIMULINK

The EXPORT module of ServoPlan generates a SIMULINK script file from the input data and the calculated data of ServoPlan.

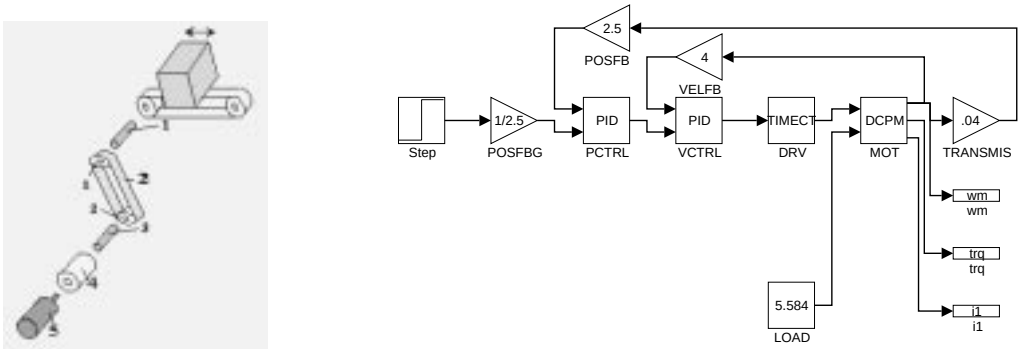


Figure 56: Conversion from ServoPlan to MATLAB/SIMULINK (including controller)

The program flow is the same as for the conversion to SPICE. The transmission model is reduced to a single parameter in the motor model, instead of the various components in the PSpice model. A gain block has been provided which represents the reduction of the motor velocity. Figure 56 shows the SIMULINK model for a motor with position controller.

6.4. From Physical to Functional

Physical simulators do normally not have the advanced facilities for control design as block oriented programs like MATLAB and XMATH. Direct model conversion is however difficult, due to the different model representations used by physical simulators and functional simulators.

For the design the controller, the system is linearised, the frequency response calculated, and transferred to the block oriented simulator. Model linearisation requires two steps: first, removing all the non-linearities, e.g. friction, backlash,... as well as constant forces; secondly, calculate the transfer function or the frequency response.

Currently, there are no standard methods to transfer data between systems. Transfer functions, state space systems and frequency responses can be represented as matrices, which can be relatively easy written to file. The representation of matrices in MATLAB script files is relatively easy, and requires only a header line to name the matrix, and an ending line to give names to the different rows. Procedures

can be easily written for the different CAE-tools to convert their output data to MATLAB script files or to directly write data to MATLAB script files. This section will describe the procedure for the conversion from PSpice and ADAMS to MATLAB.

6.4.0.1. PSpice → MATLAB

PSpice allows a frequency response analysis for small-signals on analog systems. The frequency response is output in a table format.

For the conversion from PSpice to MATLAB, the program "**out2mat**" has been developed. "out2mat" reads the ASCII-output file from PSpice and writes the simulation results to a MATLAB script file. The simulation results are written into a matrix, in which each column correspond to an output variable. The columns are given a name in MATLAB, which corresponds approximately to the name of the output variable (i.e. vm(1)→VM1 ; time→TIME; freq→FREQ; vm([a])→VMa, i(R)→ iR).

For frequency responses, three tables (FREQ, VMx (magnitude) and VPx (phase)) are created. A program, developed at Katholieke Universiteit Leuven [39], is used to retrieve the parameters of the transfer function from the frequency response. The user enters the order of the numerator and the denominator of the transfer function, based on the Bode-plot of the system. The program ("curfitc") calculates then the coefficients of the transfer function, which fits best the frequency response. A shell ("fr2tf") has been written round this program to make it more interactive and user-friendly: after calculation of the transfer function coefficients, the program generates a picture with the original Bode-plot and the new Bode-plot, and allows the user to perform further iterations.

6.4.0.2. ADAMS → MATLAB

The module ADAMS/Linear linearises the mechanism round the equilibrium point [38], and outputs the state space equations in a format that can be read by MATLAB or MATRIXx. For the MATLAB output format, 7 files are generated, 4 of them containing the state-space matrices, and the others the input and output state variable definitions [47].

A user-written function (ADstates) has been generated, which imports all these files into MATLAB. A SIMULINK block has been developed, which generates the MATLAB state space equations directly from the name of the ADAMS/Linear output file.

6.5. From Functional to Physical

At the functional design level the systems are linearised. Linearisation is always an approximation. If the system contains many non-linear elements, the behaviour of the non-linear system can differ severely from the behaviour of the linear system.

The controller, which has been designed in the block oriented simulator has to be transferred to the physical model. The controller can be implemented as analog electronics or in a digital way, as a digital electronic circuit or as a program coded on a DSP or on a computer. The interface between analog and digital is however a problem in most analog circuit simulators. PSpice is a native mixed mode simulator, and can therefore simulate circuits consisting of both analog and digital electronic components, but has no possibilities for behavioural modelling of digital systems. Digital systems can be modelled as analog behavioural models, but this technique is very time consuming, and can easily cause convergence errors.

Due to the problems of PSpice with digital control systems, the transition from functional to physical has not been studied profoundly during this research project. For simple controllers, the block diagram can be converted manually to a netlist in PSpice. For most built-in blocks in MATLAB there is an equivalent in the PSpice library. General transfer functions can be modelled as a voltage source with the Laplace function. State space matrices have to be implemented as set of matrix equations. The previous

chapter described some controlled systems that have been simulated in PSpice, as DC-motors, hydraulic systems and a field oriented control for an induction motor.

6.6. Design Environment Customisation

The schematic editor ViewDraw is encapsulated in the open design environment WORKVIEW PLUS from Viewlogic [55]. WORKVIEW PLUS is a CAE framework that conforms to CFI (CAD Framework Initiative) standards. WORKVIEW PLUS organises tools in toolbox, and in toolbox drawers, which contain the different tools.

A toolbox "Mechatronics" with one toolbox drawer "Mechatronics" has been generated to ease the design of mechatronic systems. The toolbox drawer consists of the design tools that have been used and developed during this research project, viz.:

- ViewDraw schematic editor
- SpiceLink netlister
- PSpice analog circuit simulator
- Probe post-processor for PSpice
- ServoPlan conceptual design of electric drives
- MATLAB/SIMULINK control design system
- out2mat conversion of PSpice output to MATLAB script file
- CreaLib library conversion from ServoPlan to PSpice and MATLAB

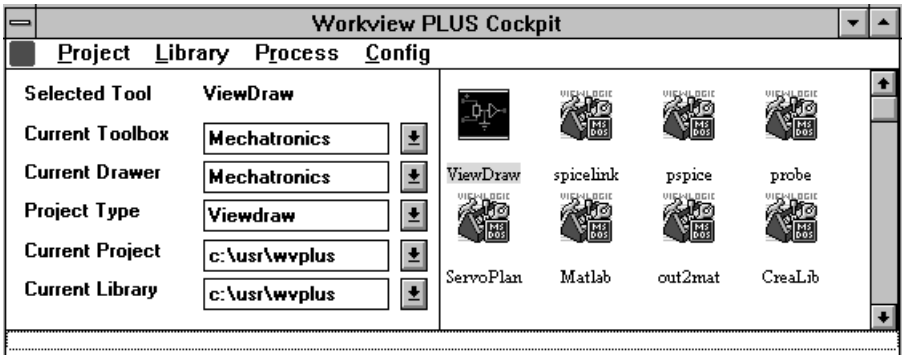


Figure 57: WORKVIEW PLUS Mechatronics Environment

The first three programs were already included in the framework. The generation of the encapsulation files is not so difficult. The programs ServoPlan and CreaLib, which are written in Visual Basic, are only available for the PC version and not for the Sun version.

6.7. Design Optimisation with Physical Design Tools

As described in the previous chapter, design tools at the physical level can be extended to other technologies. Physical design tools can also be used for conceptual design, but have mostly only limited possibilities to select optimally components. Many design tools allow to vary a single parameter, and in this way to optimise the system for one variable. It would however be interesting to compare different design alternatives in the same iteration run. This requires that two analyses can be loaded in the same post-processing run (this is e.g. possible with MATLAB), or that two systems are analysed in the same simulation run. In this section we will show how a component can be changed semi-automatically in PSpice, so that in a single post-processing run the simulation results can be verified.

6.7.1. Varying Parameters at the Functional and the Physical Level

The procedure that has to be used in the different CAE-tools to perform a sensitivity analysis on a single parameter can be very simple or very complicated:

- PSpice: A variable can be swept easily by the command ".STEP". The results can be analysed in the same session of the program PROBE.

  Paragon, a product of Microsim, is an analog circuit optimise, which allows to optimise parameters in a PSpice netlist [53]. After creation of the netlist, design parameters, performance specifications (goal functions), and optimisation constraints can be defined via simple dialogues. Paragon performs multiple iterations on the systems and compares the performance to the specifications. This program would therefore allow to optimise the parameters of the multitechnical component. The product Paragon was unfortunately not available during the research project.

- MATLAB: A function "sensitivity" has been developed for parameter sweeping. The output for all simulations is stored in one matrix, e.g. "Y". If the model has n output ports, then columns 1,n+1,2n+1,... contain the output of the first port, columns 2,n+2,2n+2,... the output of the second port,...

- ADAMS: The simulation process is controlled by the subroutine CONSUB, which gives the command MODIFY to vary a parameter, which is defined as an array. For each parameter value, a simulation is performed and the results are stored in separate files sens_001.res, sens_002.res,.... The result files can then be loaded in ADAMS/Aview and analysed. This method is however not so user-friendly, and requires especially a lot of post-processing work.

6.7.2. Varying Components at the Physical Level: VarComp

The conceptual design level contains only little information on the dynamics of the system, and it is often difficult to imagine how the system will interact with the other components. Therefore it is often desired to compare the dynamical behaviour of two components.

This can be performed in PSpice by generating a separate netlist for each component, and then verifying the results. In Probe, the post-processor of PSpice, it is however not possible to load two simulation outputs in the same time. It is therefore not possible to verify the results of a simulation of two design alternatives at the same time.

A Visual Basic program VarComp has been developed to change components in PSpice, so that they can be simulated in the same simulation run, and the outputs can be verified in the same Probe session. VarComp performs the following steps:

- Asking for the input and output file.
- Reading the input file. The data of the input file are stored in three ways: PSpice commands, PSpice primitives and PSpice subcircuits. Only the PSpice subcircuits can be changed.
- Selection of the subcircuit that has to be changed. The names of the models of the various subcircuits appear in the window in Fig 58. VarComp tries to recognise the component by verifying the name of the model and the type and the class are shown in the window. Type is a group to which the model belongs (e.g. DC-motor, AC-motor, gear), class is a group to which the type belongs (e.g. motor, rotational transmission, power amplifier,...).
- Definition of the alternative subcircuits. For each alternative, the model window, which is used by the program EXPORT is used. The subcircuit can only be changed by a model from the same class and with the same number of nodes. Fig. 59 shows the user interface.
- Creation of the file, which contains all the alternatives. To each node and component name the suffix "$*n*" with *n* the alternative number is added (*n*=0 : original)

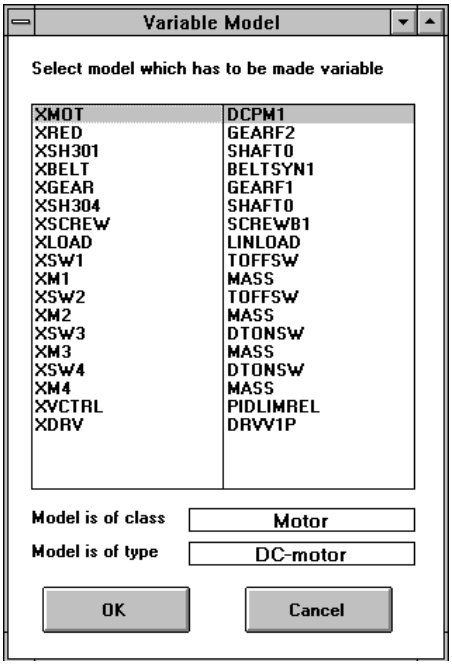


Figure 58: User interface for VarComp

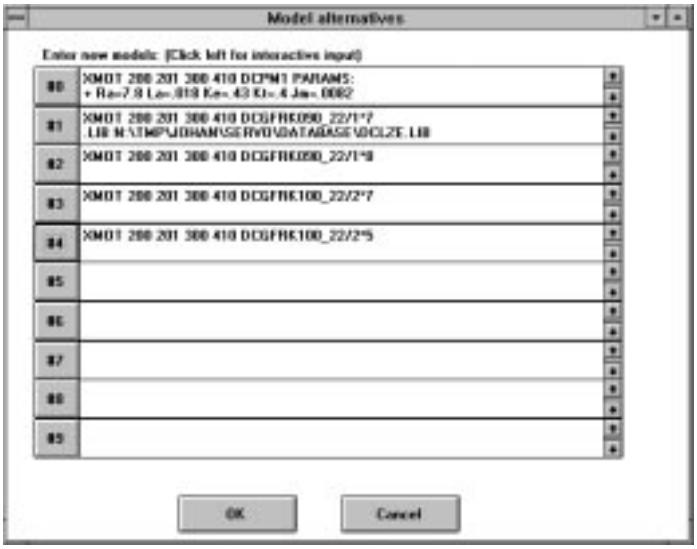


Figure 59: Alternative components.

The resulting file can be simulated with PSpice and all the different alternatives can be post-processed in one Probe run. Fig. 60. shows an example listing, in which four motors for the libraries are tested for the same DC-voltage input and inertial load, and the simulation results (motor shaft velocity).

```
* test VarComp
*$*$*
* mot$0 : DCGFRK090_22/1*71
* mot$1 : DCGFRK090_22/1*81
* mot$2 : DCGFRK100_22/2*71
* mot$3 : DCGFRK100_22/2*51

.lib mecano.lib
.lib j:\comm\dclze.lib
.tran 1m .4 0 0 uic
.probe
*$*$*
Xmot$0 in$0 0 w$0 0 DCGFRK090_22/1*7
xload$0 w$0 mass PARAMS: m=.001
vin$0 in$0 0 dc 24
*$*$*
Xmot$1 in$1 0 w$1 0 DCGFRK090_22/1*8
xload$1 w$1 mass PARAMS: m=.001
vin$1 in$1 0 dc 24
*$*$*
```

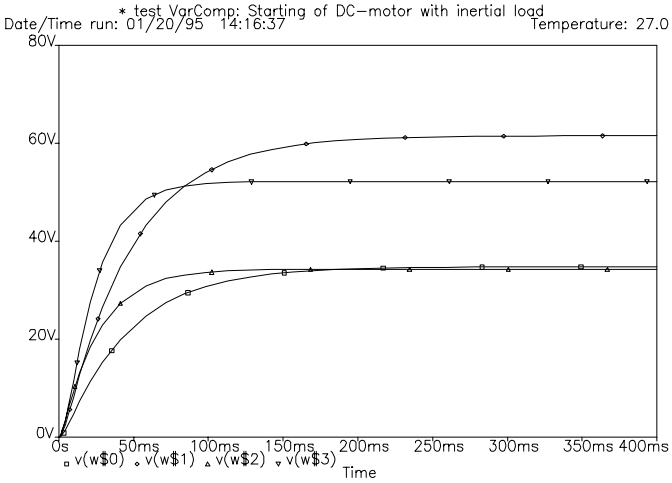


Figure 60: Variable Component netlist (start) and results

It has however been noticed that the results of the simulation may be influenced by the number of alternatives in the same netlist: the number of equations, which has to be solved by PSpice increases, which can affect the time stepping and accuracy checking algorithms, and lead to other results as when the system is modelled alone, or even to convergence problems.

6.8. Conclusions

This chapter described how the different design levels can be integrated. Conceptual design tools have mostly only limited simulation possibilities, whereas physical level design tools have mostly only limited component optimisation possibilities. A major problem is however the lack of standardisation: at this moment, there are no standards for model transfer that are supported by the commercial CAE-tools. Another problem is that the model components are modelled at different abstraction levels in the different tools, and hence use different parameters.

The first sections showed how a physical or functional model can be generated from a system description in a conceptual design program. This provides the conceptual design program with a simulation tool, and allows to verify the dynamical behaviour of the various design alternatives in the conceptual design tool. Transfer of models requires however that both tools have libraries of the same components, as well as conversion routines for the model parameters. The export module, which has been added to ServoPlan, can be extended to other tools, by entering for each model the model representation in the other tool. If the new tool uses other parameters than the parameters that are used by the PSpice model, conversion routines have to be added.

On the other hand, a routine has been written to allow the comparison of different design alternatives in the physical design tool in a single post-processing run. A problem is however that the simulation results can be affected by the other design alternatives.

CONCLUSIONS

This report discussed the results of the research project “Design and Simulation of Mechatronical Systems.”

Two methods have been developed and tested for the simulation of mechatronic systems: the concurrent simulation of subsystems in different CAE-tools, and the extended analog circuit simulation.

A computer integrated environment, called MISE, has been developed for the concurrent simulation of a system in different CAE-tools. The simulation environment consists of two parts: the general manager MISE and simulation engines. The general manager controls the timing, the synchronisation of the simulations and the communication between the different CAE-tools. The simulation engines, which have to be developed for each CAE-tool separately, convert the messages from MISE to CAE-tool commands. The environment is tool-independent: simulation engines can be developed for each tool that can be executed under UNIX. Simulation engines have been developed for ADAMS, PSpice and MATLAB. The advantage of this approach is that it allows to model each subsystem in the most appropriate tool, and thus to use already existing models and available model libraries. A drawback is the long execution time, due to the large amount of pre- and post-processing that is required for tools, which do not allow easy access to the integration routine.

The second method, extended circuit simulation, is based on the energy conservation principle, which allows to model complete multitechnical systems in one single technology dedicated physical simulator, e.g. an analog circuit simulator. PSpice, a SPICE-based analog circuit simulator, is used to model multitechnical systems. A large multitechnical library, containing mathematical, electric, electro-mechanic, mechanic (translational, rotational and planar mechanics) and hydraulic component models, has been developed for rapid prototyping of mechatronic systems. Component models are available at different accuracy levels, which allows to change the system's accuracy depending on the design needs. The schematic editor ViewDraw has been customised to allow the graphical input of multitechnical system models: a graphical symbol library has been developed and a menu for adding mechatronic components has been appended to the standard menu. This method allows to model and simulate complete multitechnical systems in a single, affordable CAE-tool.

The simulator PSpice is however not able to perform all the tasks, which are desired during the design process, as e.g. the selection of commercial components. Links have therefore been developed between ServoPlan, a program developed at VTT Automation for the conceptual design of electric drives, and the multitechnical PSpice library.

PSpice is a native mixed-mode simulator, but has no possibilities to model digital systems on a behavioural level. The standardised language VHDL has been developed for this purpose, but the functions that were needed for mixed-mode behavioural modelling were unfortunately not implemented in Viewlogic’s VHDL, which was available during this project. Full mixed-mode behavioural modelling will be possible with the VHDL-A language, which will be standardised in the following years, but which was unfortunately not yet available during this project.

REFERENCES

[1] P.J. Ashenden, *The VHDL Cookbook*, University of Adelaide, 1990

[2] J.F. Blackburn, G. Reethof, J.L. Shearer, *Fluid Power Control*, Technology Press of M.I.T. and John Wiley & Sons, New York, 1960

[3] F.E. Cellier, *Continuous System Modeling*, Springer Verlag, New York, 1991

[4] L.O. Chua, K.A. Stromsmoe, "Lumped-Circuit Models for Nonlinear Inductors Exhibiting Hysteresis Loops", *IEEE Trans on Circuit Theory*, Vol. CT-17, No. 4, November 1970, pp. 564-574

[5] J.A. Connelly, P. Choi, *Macromodelling with SPICE*, Prentice Hall, Englewood Cliffs, NJ, 1992

[6] R. Comerford, "Mecha...what?", *IEEE Spectrum*, August 1994, pp. 44-49

[7] G.P. Dubey, *Power Semiconductor Controlled Drives*, Prentice Hall, Englewood Cliffs, NJ, 1989

[8] J. Eickhoff, R. Bisanz, "Thermophysical Simulation in Aerospace Supported by Object Oriented Software Technologies", In *Proceedings of the Conference on Modelling and Simulation 1994*, Barcelona, Spain, June 1-3, 1994, Society for Computer Simulation Int., pp. 304-308

[9] A.E. Fitzgerald, C. Kingsley, A. Kusko, *Electric Machinery*, McGraw-Hill, New York, 1971

[10] D.C. Hamill, "Lumped Equivalent Circuits of Magnetic Components: The Gyrator-Capacitor approach", *IEEE Trans on Power Electronics*, Vol. 8, No 2, April 1993, pp. 97-103

[11] C.I. Hubert, *Electric Machines - Theory, Operation, Applications, Adjustment and Control*, Maxwell Macmillan, New York, 1991

[12] D. Karnopp, R. Rosenberg, *System Dynamics: A Unified Approach*, John Wiley & Sons, New York. 1975

[13] T. Kenjo, *Stepping Motors and their Microprocessor Control*, Clarendon Press, Oxford, 1984

[14] T. Kivento, P. Yli-Paunu, "Conceptual Design Tool for Electric Servo Drives" in *Proceedings of the Tampere International Conference on Machine Automation, "Mechatronics Spells Profitability"*, Tampere, Finland, February 15-18, 1994, pp. 857-866

[15] R.J. Kochenburger, *Computer Simulation of Dynamic Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1972

[16] W. Krause, *Konstruktionselemente der Feinmechanik*, Carl Hanser Verlag, München, 1989

[17] A. Kühnle, *Report on the Training of Andreas Kühnle a the Machine Automation Laboratory of the Technical Research Centre of Finland from July 20th to November 10th 1993.*

[18] B.C. Kuo, *Theory and Applications of Step Motors*, West Publishing Co., St. Paul, 1974

[19] W. Leonhard, *Control of Electrical Drives*, Springer Verlag, Berlin, 1990

[20] H. Mann, "State of the Art in Structural Modeling and Simulation of Controlled Systems" In *Lecture Notes of the Short Course on Computer Controlled Motion*, Heverlee, Belgium, June 1992, Katholieke Universiteit Leuven, 1992, pp.195-224

[21] H. Mann, H. Van Brussel, T. Yli-Pietilä,. "Physical Level Modeling and Simulation of Multidisciplinary Systems" In *Proceedings of the 35th Simulation Conference, "Applied Simulation in Industry"*, Kongsberg, Norway, June 9-11, 1993, Scandinavian Simulation Society, pp. 51-60

[22] H. Mann, H. Van Brussel, "Metamodel and Design Methodologies for Mechatronics" in *Proceedings of the Tampere International Conference on Machine Automation, "Mechatronics Spells Profitability"*, Tampere, Finland, February 15-18, 1994, pp. 671-685

[23] H. Mann, *ICOSYM - Information and Conference System for the Czech TEMPUS and Mechatronics.* Tempus Complementary Measures Proposal, 1994

[24] H. Mann, *Computer-assisted design of drive control system*, Katholieke Universiteit Leuven, 1992

[25] H. Mann, *Electrohydraulic positioning system*, Katholieke Universiteit Leuven, 1992

[26] H.A. Mantooth, M. Vlach, "Beyond SPICE with Saber and MAST", In *Proceedings from the IEEE International Symposium on Circuits and Systems*, San Diego, Ca., May 10-13, 1992, pp. 77-80

[27] W.J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation*, Kluwer, Boston, 1988

[28] H.E. Merritt, *Hydraulic Control Systems*, John Wiley & Sons, New York, 1967

[29] T.J.E. Miller, *Brushless Permanent-Magnet and Reluctance Motor Drives*, Clarendon Press, Oxford, 1989

[30] T.J.E. Miller, M. McGilp, "Nonlinear theory of the switched reluctance motor for rapid computer-aided design", *IEE Proceedings*, Vol. 137, Pt. B, No. 6, Nov. 1990, pp.337-347

[31] N. Mohan, T.M. Undeland, W.P. Robbins, *Power Electronics: Converters, Applications, and Design*, John Wiley & Sons, 1989

[32] L.W. Nagel, *SPICE2: A computer program to simulate semiconductor circuits*, University of California, Berkeley, ERL-M520, 1975.

[33] D. O'Kelly, S. Simmons, *Introduction to Generalized Electrical Machine Theory*, McGraw-Hill, London, 1968

[34] M. Otter, *DSblock: A neutral description of dynamic systems - Version 3.2*, DLR, Oberpfaffenhofen, Germany, TR R81-92, 1992

[35] K. Rintanen, P. Viitanen, P. Yli-Paunu, T. Yli-Pietilä, *Mekatronisen laitteen hierarkkinen suunnittelu ja toiminnallinen simulointi - Loppuraportti*, VTT, Tampere, VTT-KAU C-9204, 1992

[36] R.A. Saleh, A.R. Newton, *Mixed-Mode Simulation*, Kluwer Academic Publishers, Boston, 1990

[37] D.P. Sen Gupta, J.W. Lynn, *Electrical Machine Dynamics*, The Macmillan Press Ltd, London, 1980

[38] V.N. Sohoni, J. Whitesell., "Automatic Linearization of Constrained Dynamical Models". *Journal of Mechanisms, Transmissions, and Automation in Design.* Vol 108 , Sept. 1986, pp. 300-304

[39] J. Swevers, *Linear Identification and Control of Flexible Robots*, Ph.D. Dissertation, Katholieke Universiteit Leuven, 1992

[40] C.K. Taft, R.G. Gauthier, S.R. Huard, T.J. Harned, *Brushless Motor Design and Analysis*, University of New Hampshire, Durham

[41] H. Thielemans, H. Van Brussel, "State of the Art in Motion Control Hardware", In *Lecture Notes of the Short Course on Computer Controlled Motion*, Heverlee, Belgium, June 1992, Katholieke Universiteit Leuven, 1992, pp. 143-194

[42] J.G. Truxal, *Control Engineers' Handbook*, McGraw-Hill, New York, 1958

[43] C. Woodford, *Solving Linear and Non-linear Equations*, Ellis Horwood, New York, 1991

[44 X.T. Yan, J.E.E. Sharpe, "A System Simulation Platform for Mechatronic Product Design", In *Proceedings of the Conference on Modelling and Simulation 1994*, Barcelona, Spain, June 1-3, 1994, Society for Computer Simulation Int., pp. 304-308

[45] P. Yli-Paunu, T. Kivento, *MOTSUSI - Loppuraporti*, VTT Automation, Tampere, 1993

[46] T. Yli-Pietilä, H. Huovila, P. Yli-Paunu, L. Wildenburg. "Using Analog Circuit Simulation for Rapid Prototyping of Multitechnical Devices." In *Proceedings of the International AMSE Conference "Modelling & Simulation"* New Orleans, Oct. 28-30, 1991, AMSE Press, 1991, Vol. 2, pp. 67-77

[47] N., *ADAMS Reference Manual Version 6.1.*,Mechanical Dynamics, Ann Arbor, Mi., 1992

[48] N., *ADAMS/View User's Guide Version 6.1*, Mechanical Dynamics, Ann Arbor, Mi., 1992

[49] N., *HDL-A User's Manual*,Revision 2.0, Anacad, Ulm, 1994

[50] N., *DC Motors - Speed Controls - Servo Systems*, Electro-Craft Corporation, Hopkins, 1980

[51] N., *PSpice - Circuit Analysis - Version 5.1.*, Microsim Corp., Irvine, Ca., 1992

[52] N., *SIMULINK - User's Manual*. Mathworks Inc., South Natick, Mass., 1992

[53] N., *The Design Center,* Microsim Corp., Irvine, Ca., 1994

[54] N.,*THK LM System Ball Screws, Catalog No. 75-1 BE,* s.d.

[55] N., *Workview PLUS on Windows 5.1,* Viewlogic Systems Inc., Marlboro, Mass., 1994