OBJECT-ORIENTED MODELING OF ELECTROMECHANICAL SYSTEMS WITH SWITCH CIRCUITS BY USING TRANSMISSION ELEMENTS

Yongbo Chen[1], Oliver Lenord[1], Robert Michel[2], Alexander Schmitt[1], Dieter Schramm[3]
[1]Bosch Rexroth AG, Germany, [2]Breuell & Hilgenfeldt GmbH, Germany, [3]Uni. Duisburg Essen, Germany

Corresponding author: Yongbo Chen, Bosch Rexroth AG, Department of Advanced Engineering
97816 Lohr am Main, Rexrothstraße 3, Germany, yongbo.chen@boschrexroth.de

Abstract. The object-oriented modeling via *transmission elements*, which was originally introduced to model the kinematics and dynamics of multi-body systems, is applied for modeling electric drive systems with switch circuits in this paper. This approach is implemented with C++ in the simulation software *Drive&Control Engine*, abbr. *D&C Engine*, which is designed to model and simulate mechatronic systems in the drive engineering by the department of advanced engineering of the Bosch Rexroth AG.

The switch elements, used in the power converters of electric drive systems, are treated as ideal switches and are described in an efficient, stable way. Discontinuities caused by the ideal switch models are dealt with an event-handling algorithm. The application example is a complete permanent-magnet-synchronous-motor (PMSM) drive system. A benchmark simulation is carried out to demonstrate this approach's performance in terms of computational time and accuracy.

1 Introduction

Modern drive systems, e.g. the hydromechanical or electromechanical, are generally composed of components from different physical domains. Using the signal-oriented approach to model such systems usually results in a huge amount of effort to derive the explicit form of the equations. Beyond this drawback, models of the signal-oriented approach are lack of hierarchical structures, and hardly reusable. The object-oriented modeling of physical systems, first introduced by [8], can be used to overcome such difficulties. In recent years, this paradigm has become the mainstream in the physical modeling. After a series of international efforts, the *Modelica* [13] language has been developed to support the physical modeling in 1999. Since then it becomes more and more significant in the industrial application. At the moment several commercial simulation programs of mechatronic systems support its syntax and semantics, e.g. Dymola [7].

The efficiency of the physical modeling paradigm, especially the acausal modeling with the *Modelica* language, depends strongly on its language compiler and preprocess optimization, where all of the declarative equations of the models are sorted and computed symbolically. Concerning the aspect of conservation laws at nodes, e.g. power continuity and mass balance, the object-oriented modeling via transmission elements does have the same root as that of the physical modeling paradigm. In addition, it also owns lots of the other merits of the physical modeling paradigm. The largest difference is that it offers an another approach to generalize the equations of multi-domain systems in a numerical manner. This approach was primarily used for modeling the kinematics and dynamics of multi-body systems in [11] and [12]. Its applications in hydraulic systems can be found in [14]. In this paper, the approach is applied to model electric systems, and implemented in the simulation software named *D&C Engine*.

Switching power converters are frequently applied in the modern electric drive systems. They are used to control the electric motors. Simulation of such converters is problematic for several reasons. They are operated by the frequent switching up to several hundred khz. In contrast, some of the time constants, at least for the mechanical parts, are more orders of magnitude larger than the switching period. This property makes the equation system potentially stiff. On the other hand, each switching event, in an ideal sense, is a discontinuity. Typical power converter simulations can run for hundreds or thousands switching cycles, which may lead to problems for the simulation software that does not have the explicit mechanism for handling these discontinuities. Problems are for example long simulation time, lack of convergence, and convergence to an erroneous solution.

As the default solution in many simulation tools, the switch is non-ideally modeled by a variable resistor, in which the resistance is assumed to be a small value at switch-on and a large value at switch-off. This could introduce an artificial stiffness to equation systems and may lead to inaccurate results [5]. In most cases of simulating complex electric drive systems, the dynamics of circuits during switching is of little interest. Hence, the switch operation can be regarded as instantaneous and the switch itself can be treated as ideal and controlled by discrete states, such as the variables of the enumeration type ON and OFF.

The ideal switch model, in the object-oriented fashion, involves however the change of the modeling causality and leads to variable structure systems in simulation. To deal with these problems, modeling ideal switches by using *parameterized curves* has been developed for the physical modeling paradigm, e.g. in [15]. But this doesn't seem to be supported by the current version of *Dymola* for the standard electrical library of the *Modelica* language, especially when the ideal switch is connected to an inductive storage element in [5]. In [6] the ideal switch model

was described by using constraint equations and implemented as transmission elements. This method, however, due to the implicit nature of differential algebraic equations (DAE), could be superior in terms of the computational performance only in case of extremely stiff systems. The ideal switch has been also handled by using a linear complementary formulation [9]. But it seems to be difficult to apply it in an object-oriented modeling environment and be applicable only in special cases (e.g. it seems not possible to describe the ideal thyristors). To eliminate these disadvantages above, the complete ideal power converter has been developed as a whole transmission element, and the energy storage elements that couple directly with power converters may be considered in the model.

Discontinuities, which are caused by the ideal switch models, are dealt by an event-handling algorithm integrated in the *D&C Engine*. They are described as events, which can be divided into two types: time and state events. Time events can be directly provided in form of a linearly-linked list of time points for the integration algorithm, which is able to hit the point accurately with its step-size control. The basic idea of handling the state events is to lock the system equations in the subintervals [3]. Event conditions are specified in the form of a set of root functions. An event occurs if a value of the root functions crosses through zero. The root-searching algorithm locates its exact time occurrence, and the integration process will be stopped. After the proper handling it will be restarted. The implementation in the *D&C Engine* was demonstrated in [6].

Due to the fact that the time constants of power converters are much smaller than those of mechanical parts, the equation systems of electric drive systems are high-potentially stiff. For this reason, the implicit integration algorithm RADAU5 [10] is applied for the simulation experiments. The application example is a PMSM drive system. Its model is set up in the *D&C Engine* and a commercial simulation tool respectively. The performance in terms of computational time and accuracy is demonstrated by means of the benchmark simulation.

2 Object-oriented Modeling via Transmission Elements

An object-oriented modeling environment for physical system modeling should offer at least the following features [4]: encapsulation of knowledge, hierarchical modeling, object instantiation, class inheritance, topological interconnection capability, and generalized networking capability. The first four features have the deep root in the object-oriented programming, thus can be performed by make use of the object-oriented programming language. The others, as will be seen, can be realized by the concept of transmission elements.

The basis for the following object-oriented modeling is the classification of its components into two categories: *node* (state) objects and transmission elements. The node objects hold the information about the *through* variable f and *across* variable e for the respective physical domain. It serves as the interface between the models and equation solvers. The transmission elements transmit this information from one set of node objects to another.

2.1 Nodes and Transmission Elements

Considering the behavior of physical components in reality, they all can be regarded as a kind of transmission function regardless of currents, forces, flow rates and so on. This common property can be defined as the virtual function of an abstract object, whose implementation can be easily performed by using the abstraction mechanisms of the object-programming language C++. The abstract object, in our case, is defined as a basic class called *EdgeBase*, see Figure 1. It owns the virtual member function named *Update* for the description of transmission behaviors. The transmission elements on the level of physical objects are the classes derived from the basic class *EdgeBase*. The virtual function *Update* of its base class can be overridden by those of transmission elements locally. In this way, the user can restrict himself just to encapsulating the required knowledge, for whatever the physical object contains, into the corresponding transmission element. The C++ compiler ensures that the right function in the right object is invoked in each case. This property also makes the application of the object-oriented modeling via transmission elements domain-independent.

In general there are two types of transmission behaviors to be found. In case of the *across-through* transmission, the *through* variable f can be determined by the *across* variables, while, for the *across-across* transmission the *across* variable e_i is calculated by the other *across* variable e_{i-1} or with the *through* variable f. These behaviors can be similarly imagined as the two kinds of electric resistors in Bond Graph: *voltage-drop-causers* and *current-flow-causers*. Whereas the physical transmission behaviors exhibit the bi-directional nature, the signal-oriented components used in control engineering can be considered as unidirectional transmission elements with the input-output behavior. It can be treated as either of the two transmission behaviors.

The physical transmission elements have interfaces or connectors for the respective domain. They are implemented as classes, and obtain all quantities needed to describe the interaction. Current and voltage are for example needed for electrical components. Position, velocity, mass, and force are needed for one dimensional translational mechanics. By using the connectors, the modeler is able to interconnect the component models in the same way as he does it in a laboratory.

The node object is used to interconnect the transmission elements. It offers an unique *across* variable e, which can be referenced by all interfaces of the connected transmission elements. The sum of all connected *through*

variables is calculated in the node object. In order to ensure the conservation law, the sum of *through* variables at the node should be equal to zero according to physical principles. Therefore, it is guaranteed that no transmission elements will ever generate or lose energy in an uncontrolled fashion. It is subsequently used to determine the right-hand sides of differential equations or the residuals of constraint equations respectively. An unlimited number of interfaces of transmission elements can be attached to the node object.

The interconnection of transmission elements is limited to the same physical domain. The interactions of different domains are executed by the transmission elements of actuators, e.g. electric motors, generators, pump, just as they are in the physical world.

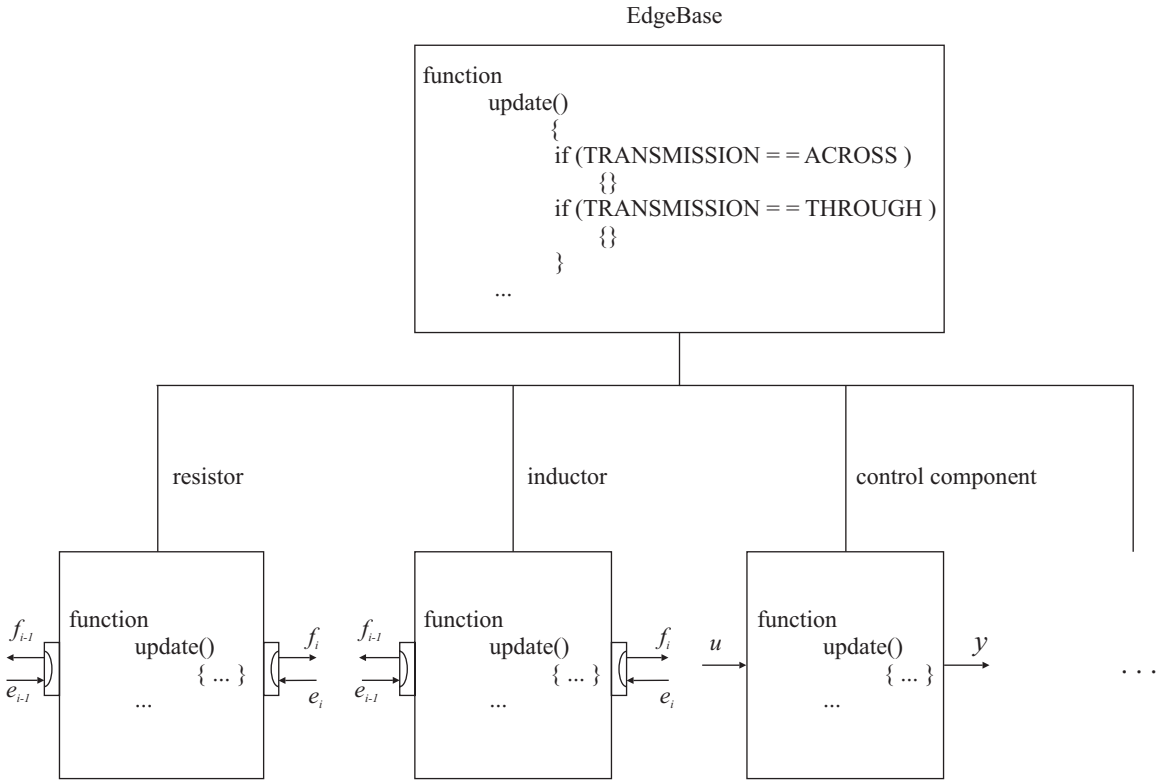


Figure 1: Inheritance tree for transmission elements

The physical modeling paradigm, particularly its recent development, is characterized by the acausal equation-oriented description of models of complex systems. Following this properties strictly, all equations of the models are first acausally stored in a flat model. Global reordering of equations (causalization of equations) and symbolic calculation are inevitable, if they are going to be simulated in an efficient way. This process makes the simulation performance particularly dependent on the stable and powerful *Modelica* language compiler and its optimizer, which are, at least right now, not as easy to be accessible as a C++ one.

In contrast, the object-oriented modeling via transmission elements may provide the same features of most importance that the physical modeling paradigm must offer. Its models of complex systems consist of a set of objects of nodes and transmission elements. They are assembled in a causal manner according to their topological structures, which can be obtained by using the well-established graphic theory. During simulation the objects are invoked in the same order as they are assembled. The equations of the invoked objects, most of which are assignments in the *Update* function, are calculated locally. Only equations, given by the dynamics of nodes and transfer behavior of transmission elements, are abstracted and formulated as a set of DAEs

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{w}, \mathbf{u}, t) = 0 \,, \tag{1}$$

in the form of residuals and right-hand sides of differential equations. $\mathbf{x}$ is a set of state variables, $\mathbf{w}$ is a set of algebraic variables, $\mathbf{u}$ is a set of inputs, and t is the time. They can be solved directly by using the numerical DAE-solver in the *D&C Engine*. In some cases of electrical systems, the DAEs can also be converted to ODEs by combining different types of transmission behaviors.

2.2 Example of an electrical circuit

The simple electrical circuit in Figure 2 is used to illustrate the procedure of equation generalizations via transmission elements. This procedure is usually performed by the graphical analysis, but it is illustrated here manually in the purpose of demonstration. Each basic electrical component consists of pins (connectors). They contain current

and potential, the physical modeling paradigm alike. The objects of pins are classes, therefore, the classical notation for the class members can be used to express the current or potential, e.g. *pin.i* or *pin.v*. For the purpose of simplicity, this expression is not employed to derive the equation system of the electrical circuit in Figure 2.

The object equations of basic electrical components, which are encoded in the *Update* functions of objects, are shown in Table 1. Notice that, the second equation of the capacitor and the first equation of the node are residual equations. The others, except for the differential equations, are all assignments. All of components in this circuit are first described by the *across-through* transmission. That means, the components get the potentials from pins and provide the currents to them.

The voltage source and ground provide simply the constant potential references, thus their connection is actually redundant. Their assignment equations will not be executed all the time in simulation due to the computational efficiency. The connections between the potential-providers and the other components do not yield the equations as the node object does, because the potential variables are already known. The components (R_1, L, R_2, and C) and their connections (the potentials v_1 and v_2) are instantiated in a consecutive way and pushed back into a vector: first voltage source, second R_1, following L, then node for v_2 between R_1 and L, R_2, C, and node for v_2 between R_2 and C. These objects as well as their *Update* functions are invoked in the same order as the instantiation during simulation. Considering this sequence, a set of the right-hand sides of differential equations and residuals of constraint equations, equivalent to the following DAEs

$$\frac{\mathrm{d}i_L}{\mathrm{d}t} = \frac{v_1}{L}, \tag{2}$$

$$\frac{u_0 - v_1}{R1} - i_L = 0, \tag{3}$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{i_C}{C}, \tag{4}$$

$$u - v_2 = 0, \tag{5}$$

$$\frac{u_0 - v_2}{R2} - i_C = 0 \tag{6}$$

are generalized, where v_1 and v_2 are the node potentials between R_1 and L, R_2 and C. They are five equations with five unknowns v_1, i_L, v_2, i_C, and u. So mathematically they are solvable.

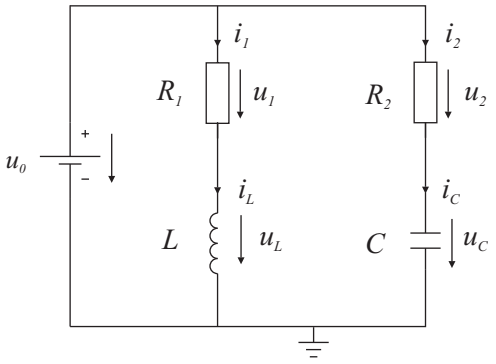


Figure 2: A simple electrical circuit

These equations are similar to the results from the nodal approach in the circuit analysis. The difference is that the object-oriented modeling via transmission elements makes the components easy to be accessed and edited by the users, which can be treated as building blocks rather an elaborate net list. Note that the procedure of the equation generalization described above, due to the additional algebraic equations, is computationally not very efficient, even the sparse-matrix techniques are applied. But its simplicity and easy adaption to the other domains make it particularly well suited for the object-oriented modeling of physical systems. More efficient procedure can be achieved by reducing the DAEs into ODEs. This, for some circuits, can be done by combining different types of transmission behaviors. The technique is demonstrated with the same circuit example as shown in Figure 2.

The causality of capacitors and inductors, or their types of transmission behavior, is usually fixed, because in simulation the numerical integration is more preferable than the numerical differentiation. So the *across-through* capacitor can be expressed as the *across-across* one, and its object equations are

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{i}{C}, \tag{7}$$

$$v_1 = u + v_2, \tag{8}$$

$$i_1 = -i_2 = i. \tag{9}$$

Replacing the *across-through* capacitor in Figure 2, the node equation for the potential v_2 between C and R_2, becomes redundant, because the potential v_2 can be calculated by the assignment with the state variable u.

Similarly the *across-across* resistor can be described as

$$v_2 = v_1 - i\,R\,, \tag{10}$$
$$i_1 = -i_2 = i \tag{11}$$

in the *Update* function. The potential v_1 of the node between R_1 and L, can be obtained by using the known source potential u_0 and the state variable i_L, see equation 10. Thus, this node object can also be removed. The sequence of the objects that are pushed back into the vector is changed into: first voltage source, second R_1, then L, R_2, and C. The solved equation system can be subsequently reduced to

$$\frac{\mathrm{d}i_1}{\mathrm{d}t} = \frac{u_0 - i_L\,R}{L}\,, \tag{12}$$
$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{u_0 - u}{RC}\,. \tag{13}$$

By using proper combinations of different transmission behaviors, the DAEs resulted from the electrical circuits may be adapted into the ODE form. But this could be difficult for overdetermined systems, such as circuits with parallel connected capacitors or series connected inductors, where the symbolic operations are usually necessary.

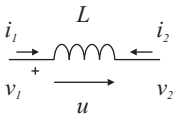
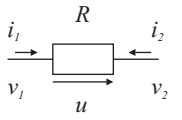
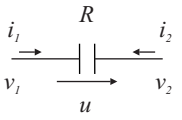
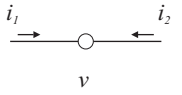
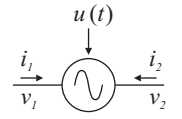
inductor		$u = v_2 - v_1$ $\dot{i} = u/L$ $i_1 = -i_2 = i$
resistor		$u = v_2 - v_1$ $i = u/R$ $i_1 = -i_2 = i$
capacitor		$\dot{u} = i/C$ $0 = u - v_2 + v_1$ $i_1 = -i_2 = i$
node		$0 = i_1 + i_2 + \cdots + i_n$ $v_1 = v_2 = \cdots = v_n$
voltage source		$u = u_0$ $v_1 = u + v_2$ $i_1 = -i_2 = i$
ground		$v = 0$

Table 1: Equations of basic electrical components

2.3 D&C Engine

D&C Engine is a simulation software developed by the advanced engineering department of the Bosch Rexroth AG. It is currently based on the object-oriented modeling via transmission elements introduced above, and designed for modeling and simulation of mechatronic systems of the drive engineering. Its applications cover the mechanical (1D and 3D), hydraulic, pneumatic, electric, magnetic, and control engineering, but not limited to them due to the extensibility of its modeling approach.

D&C Engine offers a comprehensive graphical user interface (GUI) which has e.g. the ability of the *drag-drop* modeling and the result plotting. The calculation in *D&C Engine* is based on SI units, and the unit conversions between parameters or variables are considered. The object diagrams are images of the scalable vector graphics (SVG) format. The component models are implemented as classes in the C++ language. However, they are instantiated by the extensible makeup language (XML) files. The declaration of parameters and connectors is included, see the following content of a XML file for the resistor model:

```
<Component Name="Resistor" Caption="Widerstand"
```

```
        <Inputs>
                <Electric Name="Pin1" Caption="Anschluss P" ></Electric>
                <Electric Name="Pin2" Caption="Anschluss N" ></Electric>
        </Inputs>
        <Parameters>
                <Analog Name="Resistance" Caption="Widerstand" Value="10" Unit="Ohm"/>
        </Parameters>
        <Implementation Library="DCE.Electric.dll" Function="Resistor"/>
</Component>
```

A typical system in *D&C Engine* is graphically made of the object diagrams of components and their connections. Behind them are a set of objects of nodes and transmission elements, whose topological structures are obtained by the graphical *analyst* inside of *D&C Engine*. The generalized equation systems, either a set of DAEs or ODEs, are solved by its different numerical solvers. The event-handling algorithm is implemented. Hence, it is able to simulate both of continuous and discontinuous systems.

3 Modeling a PMSM Drive System

The application example is a PMSM speed servo drive system, which is composed of the machine, speed and position feedback, speed and current controller, smoothing capacitors, AC net, inverter, and rectifier, see Figure 3. The rectifier converts the AC supply into the DC, where the inductances of the electric net are considered. The DC supply is smoothed by the capacitors. The speed feedback control is used to generate the reference d, q axis currents. Together with the position feedback, these go through the inverse Park transform to provide the a, b, c reference currents. The actual currents are measured by the current sensors. Their values are compared to the references and the error signals are generated. According to these error signals, the hysteresis or pulse-width modulated (PWM) controller fires commands to the six switches of the inverter. The actual motor currents are forced through the frequent switch-on and -off of the switches to equal the commanded values at all times.

The model of this complex electric drive system is completely built up by modeling the components as nodes and transmission elements. The simulation takes place in the *D&C Engine*. The solid lines illustrated in Figure 3 indicate the physical transmission behavior of electromechanical parts, while the dashed lines suggest the signal-oriented transmissions of control parts.

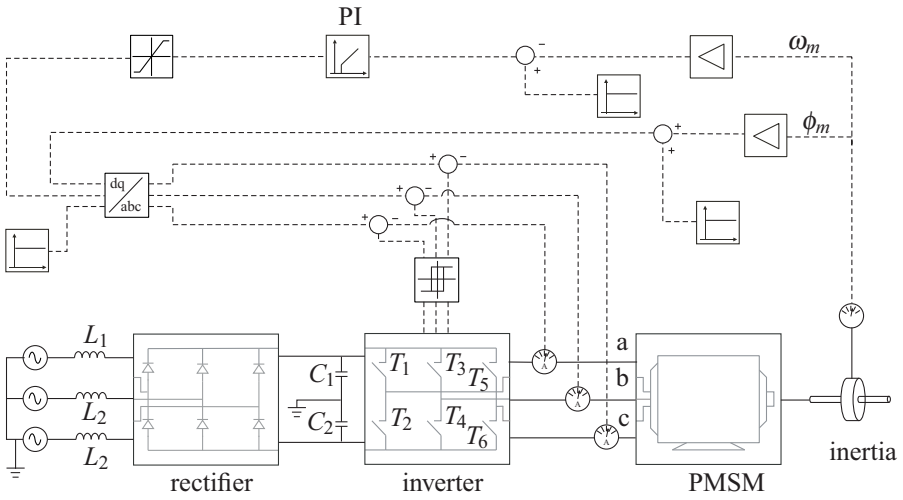


Figure 3: Schematic of a PMSM speed servo drive system

3.1 Machine Model

The PMSM has numerous advantages over other machines which are conventionally used as AC servo drives. It is characterized by the higher torque-to-inertia ratio and power density. These properties make it preferable for certain high-performance applications like robotics, aerospace actuators, and machine tools. The mathematical model of a PMSM is similar to the well-established d, q model of the wound-rotor synchronous machine. The following assumptions are made for the model applied here.

- Saturation is neglected.
- The induced EMF is sinusoidal.
- Eddy currents and hysteresis losses are negligible.

- There is no field current dynamics.
- There is no cage on the rotor.

The detailed mathematical derivation can be found in [16]. As shown in Figure 4, the transmission element of the PMSM contains three pins (a, b, c) of the three-phase current supply and a rotational mechanical connector for the mechanical shaft. The transmitted variables of the rotational mechanical connector include the angular velocity, angle, moment of inertia, and torque.

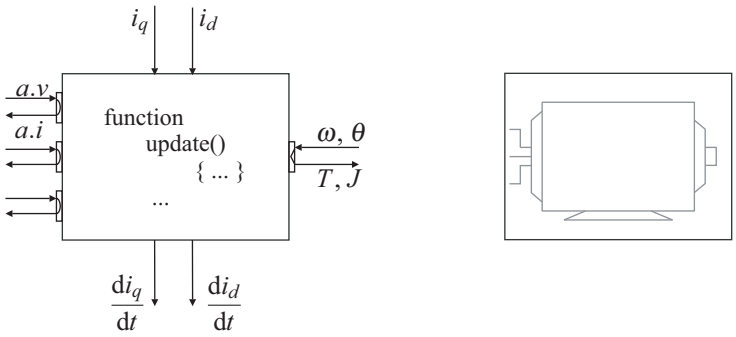


Figure 4: Transmission element of the PMSM and its object diagram

For dynamic simulation, the d, q axis equations of the PMSM are expressed in (14) and (15):

$$\frac{di_d}{dt} = \frac{1}{L_d}\left(u_d - R\,i_d + \omega_e L_q\, i_q\right) \tag{14}$$

$$\frac{di_q}{dt} = \frac{1}{L_q}\left(u_q - R\,i_q - \omega_e L_d\, i_d - \omega_e\, \lambda_{af}\right). \tag{15}$$

u_d and u_q are the d, q axis voltages, i_d and i_q are the d, q axis stator currents, L_d and L_q are the d, q axis inductances, while R and ω_e are the stator resistance and synchronous velocity, respectively. λ_{af} is the flux linkage due to the rotor magnets linking the stator.

The motor torque is

$$T_e = \frac{3}{2} P\left(\lambda_{af}\, i_q + (L_d - L_q)\, i_d\, iq\right). \tag{16}$$

P is the number of pole pairs.

The variables of d, q voltages are obtained from the a, b, c variables through the Clark-Park transform defined below

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \mathrm{a}.v \\ \mathrm{b}.v \\ \mathrm{c}.v \end{bmatrix}, \tag{17}$$

where θ_e is the synchronous position.

The variables of a, b, c currents are obtained from the d, q variables through the inverse of the Clark-Park transform defined below

$$\begin{bmatrix} \mathrm{a}.i \\ \mathrm{b}.i \\ \mathrm{c}.i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix}. \tag{18}$$

The PMSM model serves as a domain transformer, through which the electrical power is converted into the mechanical power.

The motor dynamics of the mechanical parts is separately described by the node object of 1D rotational mechanics with the following equations

$$\frac{d\omega_m}{dt} = \frac{1}{\sum_{i=1}^{n} J_i} \sum_{i=1}^{n} T_i\,, \tag{19}$$

$$\frac{d\theta_m}{dt} = \omega_m. \tag{20}$$

J_i and T_i is the concentrated parameters of the moment of inertia and torque, transmitted by the ith connected transmission element, e.g. that of the PMSM. Note that the difference of the synchronous and mechanical speed is $\omega_e = P\,\omega_m$, the same for the position $\theta_e = P\,\theta_m$. This node object can be connected to any number of transmission elements of the 1D rotational mechanics. Hence, the other mechanical models or subsystems beyond the drive systems, like the damping, friction, and gears, can also be considered.

3.2 Current Controllers and Inverter

For high-performance servo drives, hysteresis or PWM current controllers are used to ensure that the actual currents flowing into the motor are as close as possible to the sinusoidal references. In this application example the hysteresis current controller is used. It is implemented as a signal-oriented transmission element, see Figure 5. The control strategy is introduced as follows.

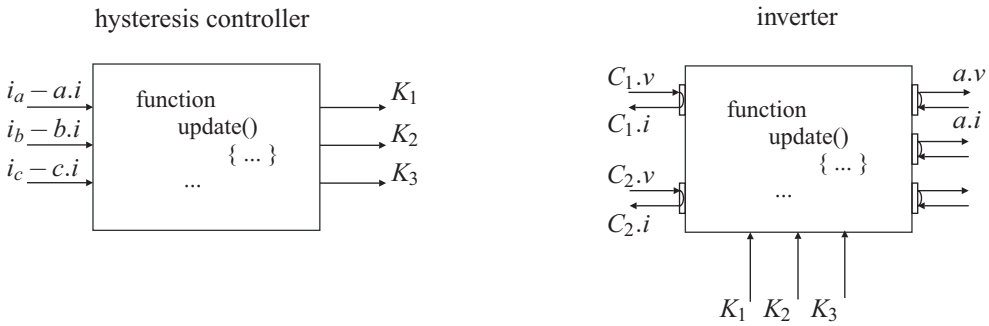


Figure 5: Transmission elements of the HCC and inverter

The actual values of a.i, b.i, and c.i that are flowing into the PMSM are measured. These values are compared with the sinusoidal reference values of currents i_a, i_b, and i_c that are generated by the speed control. Error current signals, e.g. $i_a - a.i$, are subsequently generated. The hysteresis parameter Δi defines the hysteresis bands between $i_{a,b,c} - \Delta i$ and $i_{a,b,c} + \Delta i$. The hysteresis property allows the actual values to exceed or be less than the reference values by Δi, but always within the band. The phase currents are, therefore, approximately sinusoidal as the reference values (i_a, i_b, i_c) are. The smaller the hysteresis bands, the more similar are the phase currents to the sine wave. Small hysteresis bands, however, imply high switching frequencies, which result usually the long simulation time. The smallest hysteresis window is limited by the switching capability of the inverter. The trial-and-error procedure for determining the Δi should be adopted to ensure that the inverter switching frequency is not exceeded.

Concerning only the stator phase a, the hysteresis property is formulated as root functions $f(a.i, i_a)$, whose sign is continuously observed during simulation. A sign change in this case means an event that the hysteresis band is exceeded. The exact occurrence of this event is located by a zero-search algorithm like bisection, after that the proper command will be sent to the commuting switch of the inverter. The switching of the commuting switch (T_1 and T_2) is assumed to occur simultaneously, this means, if the switch gets the control signal OFF and T_1 becomes open, T_2 is at the same time switched on (ON). The logic is shown in Table 2. Similar logic can be applied to the other two phase.

$f(i) =$	ON	OFF	voltage drop	current
$(a.i - i_a) - \Delta i \le 0$	T_1	T_2	$a.v = C_1.v$	$C_1.i = a.i,\ C_2.i = 0$
$(a.i - i_a) + \Delta i > 0$	T_2	T_1	$a.v = C_2.v$	$C_1.i = 0,\ C_2.i = a.i$
$(b.i - i_b) - \Delta i \le 0$	T_3	T_4	$b.v = C_1.v$	$C_1.i = b.i,\ C_2.i = 0$
$(b.i - i_b) + \Delta i > 0$	T_4	T_3	$b.v = C_2.v$	$C_1.i = 0,\ C_2.i = b.i$
$(c.i - i_c) - \Delta i \le 0$	T_5	T_6	$c.v = C_1.v$	$C_1.i = c.i,\ C_2.i = 0$
$(c.i - i_c) + \Delta i > 0$	T_6	T_5	$c.v = C_2.v$	$C_1.i = 0,\ C_2.i = c.i$

Table 2: Logic of the hysteresis controller and the ideal commuting switch

The inverter that converts the DC supply into the sinusoidal currents is shown in Figure 3. It consists of six switches or three commuting switches. Due to the complementary control signal of commuting switches, the freewheeling diodes of the inverter are neglected. The transmission element of the inverter in Figure 5 has five pin objects, three for the AC (a, b, c) currents and two (C_1, C_2) for the DC currents. Due to the difficulties introduced in the introduction section, the switches of the inverter are treated as ideal, which means simply a zero voltage drop over it at its switch-on, and zero current through it at its switch-off. With regard to the stator phase a of the PMSM

drive system, the zero voltage drop over the switch T_1 (ON), $a.v - C_1.v = 0$, is equivalent to the assignment of $a.v = C_1.v$, the current through the switch depends on the stator current; The voltage drop of the switch T_2 (OFF) is of little interest and the current is zero, see Table 2.

The required stator currents can also be generated by using the PWM current controller. The error current signals are compared to a sawtooth-shaped triangular wave. Depending on whether they are larger or small than the sawtooth, the proper commands are created and fired to the inverter to control the stator currents.

3.3 Rectifier

The Rectifier is used to convert the AC currents into the DC currents. It comprises generally a number of diodes or thyristors. The rectifier shown in Figure 3 is composed of six ideal diode models. The ideal diode model is mathematically similar to the ideal switch model. The only difference is that the switch conditions of diodes are controlled by the internal variables instead of the external. An ideal diode model is characterized by the facts that $i \geq 0$ and $u \leq 0$ (see the left part of Figure 6). The diode keeps open as long as the voltage u negative or zero, and i is not positive.

In order to define the switching condition, it is advantageous to split the diode model into two regions: ON and OFF. This can be described by a deterministic finite state machine (DFSM), see more properties of a DFSM e.g. in [1]. The state transition diagram of the DFSM for the ideal diode model is shown in the right part of Figure 6. The DFSM has two states corresponding to the two regions. Starting willfully from OFF, the state transition can be determined uniquely by the values of i and u. If the state is OFF and u just becomes larger than zero, it will turn out to be ON. The state transition is also tracked by the event-handling.

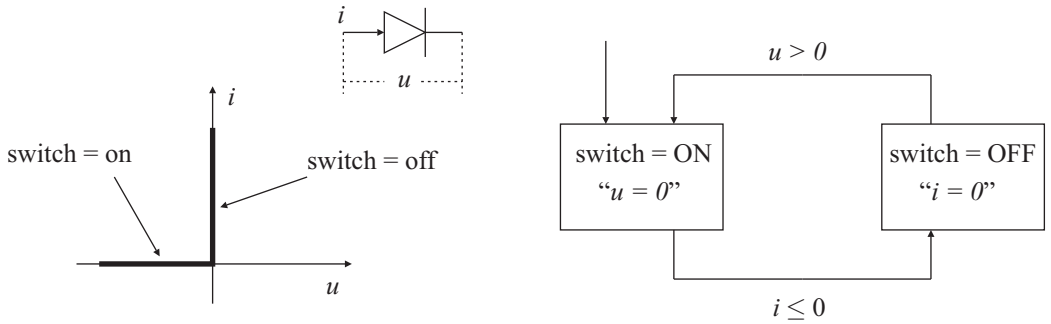


Figure 6: Characteristics of the ideal diode model and its state transition diagram

If the ideal switch model is connected to an energy storage element, e.g. an inductor, the initial value of the state variable of the inductor has to be renewed at the state transition. This can be done only if the state variable of the inductor object can be accessed from the diode object. This attempt, however, violates one of the original merits of the object-oriented modeling via transmission elements, the modularity. On the other hand, the structures of power converters are relatively simple, and they are for example always 4-pulse or 6-pulse. For these reasons, the ideal model of the complex power converter is developed just in one object of transmission elements, and the net inductances can be considered in it. They are still structure variable systems in terms of the individual diode, but variance of their structures, or better to say equations, is more easily controlled by using the DFSM in one object rather than assembling six single object of the ideal diode. Thus the model is numerically more stable.

4 Results

As described in the introduction, simulation of the PMSM drive system can not be performed by just using the standard electrical library of the *Modelica* language, e.g. in *Dymola*, if the net inductances are considered. So a parallel model for the benchmark simulation has been built in the software package *Plecs* [2].

The numerical simulation is performed with the following key parameter values: $R = 2.875\Omega$, $L_d = 8.5mH$, $L_q = 8.5mH$, $\lambda_{af} = 0.175V/rad/s$, $P = 4, C_1 = 10mF$, $C_2 = 10mF, f = 60Hz$, $A = 240V$, and $J = 0.0008kgm^2$. The applied numerical solvers for the *D&C Engine* are RADAR5, and *ode15s* [17] for the *Plecs*. Their configuration parameters were set to be same. The simulation time contributes one second.

The synchronous rotor speeds of the PMSM are shown in Figure 7, in both of the *D&C Engine* and *Plecs*. The left part of Figure 7 shows the key transient results when the PMSM is started up from standstill to a synchronous speed of $700rad/s$. A hysteresis current controller with a band from -0.25 to 0.25 is used. The right part of Figure 7 shows the speed for a $700rad/s$ stepwise increase in the speed of the machine after it has run up.

The simulation results between these two models are not visibly distinguishable. The relative deviation of the numerical results between these two simulators is smaller than 0.0025. For the constant reference value, the model in the *D&C Engine* runs 27 percent faster than that in the *Plecs*, and for the stepwise reference value 17 percent.

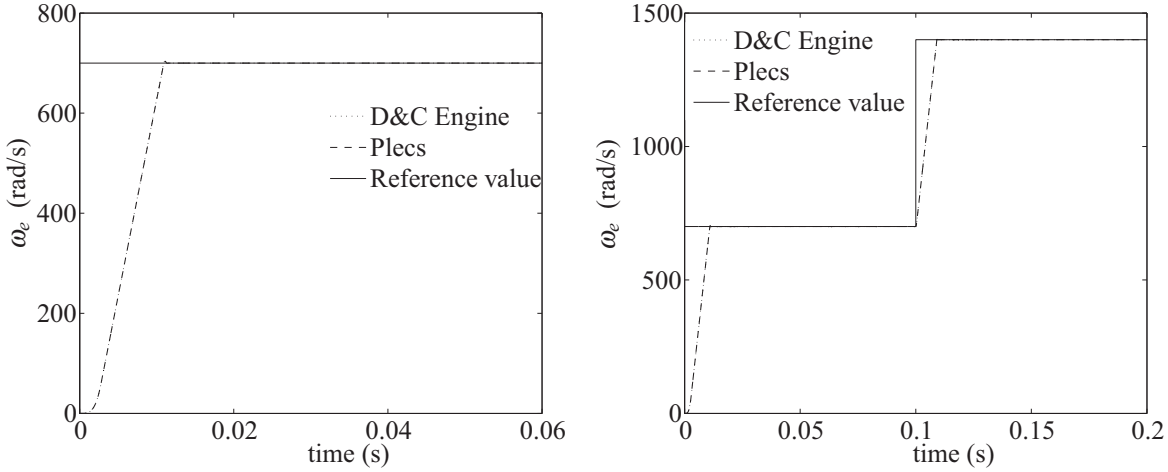


Figure 7: Speed control with the constant and stepwise reference values

5 Conclusions

The object-oriented modeling via transmission elements, as demonstrated above, shares lots of common features [4] that an object-oriented modeling environment for the physical system modeling should have. Compared to the acausal physical modeling by using the *Modelica* Language, it provides an another method to generalize the equations of complex physical systems in an iterative and numerical way. This approach was implemented in the simulation software *D&C Engine*, demonstrated with the example of the simple circuit in this paper, and applied to model the PMSM speed servo drive system.

Power converters, characterized as switch elements or their combination, are widely used in electric drive systems. To avoid the artificial stiffness and possible inaccuracy caused by the regularized switch models, ideal models are generally preferred for investigating the behavior of electric drive systems. Because of the particular difficulties in the object-oriented modeling of ideal switches, the complete converter was modeled modularly as one object of transmission elements. The event-handling algorithm was applied to deal with the discontinuity induced by the ideal switch models.

The PMSM speed servo drive system has been modeled by using a set of objects of nodes and transmission elements in *D&C Engine*. Its performancewise advantage over the signal-oriented approach, in terms of the computational time and accuracy, was presented in the attached benchmark simulation. Furthermore, the interdisciplinary nature of the object-oriented modeling via transmission elements proposes the opportunity that the components or subsystems beyond the drive systems can be easily added.

As a result, the object-oriented modeling via transmission elements provides a modular, easy-to-use scenario for the system development in the electric drive engineering, in which component developers may develop and test their component's models, system designers may then just use these as building blocks per drag and drop in *D&C Engine*'s GUI without detailed knowledge of the components. Along with the ideal models for switch elements and the event-handling algorithm, the stable, efficient and numerically accurate simulation for such systems can be achieved.

6 References

[1] Aho, A. V., Sethi, R., and Ullman, J.D., *Compilers. Principles, Techniques and tools*. Addison-Wesley, New York, 1987.

[2] Allmeling, J.H. and Hammer, W.P., *PLECS-piece-wise linear electrical circuit simulation for Simulink*. In: Proc. IEEE Int. Conf. Power Electronics and Drive Systems, pp. 355-360, 1999.

[3] Cellier, F.E., *Combined Continuous/Discrete System Simulation by Use of Digital Computer: Techniques and Tools*. PhD Thesis, Swiss Federal Institute of Technology, Zürich, Switzerland, 1979.

[4] Cellier, F.E., Elmqvist, H., Otter, M., *Modeling from Physical Principles*. In: The Control Handbook, 1996 by CRC Press, Inc., ed. William S. Levine, S. 99 - 108.

[5] Cellier, F.E., Krebs M., *Analysis and Simulation of Variable Structure Systems Using Bond Graphs and Inline Integrations*. Proc. ICBGM'2007, San Diego, 2007.

[6] Chen, Y., Lenord, O., Schramm, D., *Modeling Discontinuous Elements in Mechatronic Systems by Using Regularized and Idealized Approaches*. 4. ASIM Workshop, Wismar, 2008.

[7] Dymola, homepage: http://www.dynasim.se/

[8] Elmqvist, H., *A Structured Model Language for Large Continuous Systems*. PhD Thesis, Depart. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1978.
[9] Glocker, Ch., *Models of Non-smooth Switches in Electrical Systems*, International Journal of Circuit Theory and Application, 33: 205-234, 2005.
[10] Hairer E., Wanner G., *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer Verlag, Berlin, 1991.
[11] Kecskeméthy A., *Objektorientierte Modellierung der Dynamik von Mehrkörpersystemen mit Hilfe von Übertragungselementen*, Fortschritt-Berichte VDI, Reihe 20 Nr. 88, VDI Verlag, Düsseldorf, 1993b.
[12] Kecskeméthy, A., Hiller, M., *An object-oriented approach for an effective formulation of multibody dynamics*, Computer Methods in Applied Mechanics and Engineering, 115: 287-314, 1994.
[13] Modelica, http://www.modelica.org/
[14] Müller, J., Hiller M., *A mechatronic simulation model for the large-scale hydraulically driven ALDURO*. In: Procceedings of the International Workshop on Computer Software for Design, Analysis and Control of Fluid Power Systems, 1999, Trondheim, Norway.
[15] Otter, M.; Elmqvist, H.; Mattsson, S.E., *Hybrid modeling in Modelica based on the synchronous data flow principle,* Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on , vol., no., pp.151-157, 1999.
[16] Pillay, P.; Krishnan, R., *Modeling, simulation, and analysis of permanent-magnet motor drives. I. The permanent-magnet synchronous motor drive,* Industry Applications, IEEE Transactions on, vol.25, no.2, pp.265-273, 1989.
[17] Simulink, http://www.mathworks.com/