B. Johansson, P. Krus:
Modelica in a Distributed Environment Using Transmission Line Modelling.
Modelica Workshop 2000 Proceedings, pp. 193-198.

Paper presented at the Modelica Workshop 2000, Oct. 23.-24., 2000, Lund, Sweden.

All papers of this workshop can be downloaded from
http://www.Modelica.org/modelica2000/proceedings.html

Modelica in a Distributed Environment Using Transmission Line Modelling

Björn Johansson and Petter Krus
Department of Mechanical Engineering
Linköping University, SE-581 83 Linköping, Sweden
{bjojo, petkr}@ikp.liu.se

Abstract

Products are becoming increasingly complex with functionality and components from different engineering domains. The object-oriented modelling language Modelica facilitates the modelling of such systems. However, by making it possible to create complex models, there will be an increased focus on the actual simulation.

The distributed modelling concept is a way to deal with system complexity also during the simulation. The concept implies that the system can be partitioned into sub-systems and executed in parallel. Each component is equipped with a solver for the component specific equations. To achieve the full system behaviour, the components exchange data at specific time instants. To increase the model robustness, the components can be numerically isolated from each other using the transmission line modelling technique. The delay when signals are propagating through a system is then used to numerically separate the components.

In this paper, the feasibility of combining Modelica with transmission lines for distributed simulation is demonstrated. Also discussed are the advantages compared to the traditional techniques of simulation.

Introduction

As simulation is used as a tool for designing and analysing complex systems, a number of aspects must be considered regarding model complexity. An object-oriented and non-causal modelling language like Modelica provides a base for designing well-structured simulation models [2,10]. However, the possibilities of creating large and complex models sets the focus on other aspects that needs to be considered.

The simulation model and its executable interpretation must be numerically stable, efficient and flexible. Multi-domain models consist of sub systems with different numerical properties and time constants. The possibility to connect sub systems with different solvers and time steps is an advantage when it comes to simulation of complex multi-domain systems.

Distributed modelling

The distributed modelling concept can be viewed as an opposite solution to the centralised technique regarding the execution of the simulation model. A brief description of the centralised approach is that all equations describing the system are collected to one large set of equations. An integration algorithm is then applied and all state variables are solved for in each time step during the execution. The centralised concept is very common and used in several simulation packages. Advanced symbolic manipulation of the equations is possible to increase the numerical efficiency. The drawback of the centralised simulation technique is though that the whole system must be stated in a uniform way. Since the solver often use variable time step, fast dynamics in one part of the system affect the step-size and simulation time of the whole system.

Using the distributed modelling technique, the components building the system model are considered as objects, not only in the modelling phase but also during the execution. A local solver is attached to every component. The components are then executed in parallel exchanging the same information as the physical components. The distributed environment has a number of advantages. Most important is that a distributed environment can be very flexible. As the components are independent objects, it is rather straightforward to connect external components to the simulation model. Examples of such external components can be a piece of hardware (HardWare In the Loop [6]) or a connection to an external simulation package. These connections can be either locally on one computer or over a network (LAN, Internet etc.) [4]. Another advantage is that the distributed concept offers linear scaling properties. When the size of the systems, i.e. the number of components is increased, the simulation time is increased with the same magnitude. Although it is possible to use a

lot of information from the system structure to increase the computation speed using a centralised method, linear scaling is very hard to achieve. The scaling properties for centralised solvers are very much depending on the system properties.

In Figure 1, a possible simulation environment using a distributed structure is shown.

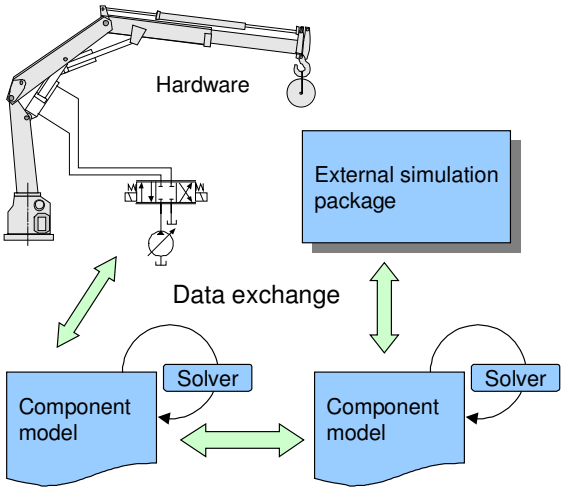

Figure 1: Distributed simulation environment

A distributed environment with or without connections to hardware and external simulation packages requires control by a top-level process. The main task for this process is to control the simulation speed for the system. A global step size can be set for the system while components with fast dynamics can be looped faster and executed with a shorter time step.

To achieve numerical robustness for a distributed approach, it is possible to numerically isolate the components from each other. This separation can be achieved using the transmission line modelling concept, TLM [7,1]. The main idea is to include the propagation of information through the system in the model. In most physical systems, there is a delay in time when a signal propagates from one component to another. For example a pressure pulse that is caused in a hydraulic valve does not immediately affect the whole system. The delay is normally depending of the speed of sound in the transmission media. This delay when information propagates is a valuable piece of extra information that can be used to separate the components numerically.

Distributed modelling using transmission lines

The core component in Transmission Line Modelling, is the Unit Transmission Line. This component can be viewed as a carrier of energy with a time delay. Depending of the system that is to be modelled, the medium passing through the transmission line is different, but the mathematical variables are flow and effort variables, the same way as in the Bond-graph modelling technique.

Figure 2: Unit Transmission Line

The equations describing the transmission line are

$$e_1(t+T) = c_1(t) + Z_c f_1(t+T) \quad (1)$$

$$e_2(t+T) = c_2(t) + Z_c f_2(t+T) \quad (2)$$

where Z_c is the characteristic impedance, c_1 and c_2 are the characteristics describing the wave from the connected component. The characteristics are of the same magnitude as effort and are defined as

$$c_2(t) = e_1(t) + Z_c f_1(t) \quad (3)$$

$$c_1(t) = e_2(t) + Z_c f_2(t) \quad (4)$$

At each component between the lines, the equations that needs to be solved are the following set of equations

$$\mathbf{f} = \mathbf{f(e)} \quad (5)$$

$$\mathbf{e} = \mathbf{c} + \mathbf{Z_c f} \quad (6)$$

where $\mathbf{f}$ is a vector containing all the flow variables, $\mathbf{e}$ is the corresponding effort variables, $\mathbf{c}$ is the characteristics and $\mathbf{Z_c}$ is a diagonal matrix with the characteristic impedances in the diagonal.

Transmission line components in a system

In a system, the components modelled as TLM-components are divided into two groups of components.

- Q-components, calculating flow
- C-components, calculating characteristic impedance and the characteristic (the wave, effort variable)

During the simulation, the different types of components are executed alternately. First, the C-components are called, updating the characteristics and characteristic impedances. Then the Q-components are called calculating the state variables as flow and effort. Examples of Q-components are resistive components as electrical resistors and hydraulic orifices. Examples of C-type components are capacitive components as volumes, springs and electrical capacitors. It is not always possible to separate components from a

component library. On illustrative example is a rectifier bridge, where the transmission line technique would require separating the diodes with transmission line elements. This separation would then affect the behaviour of the system unless the time step is very short, causing long simulation time. To avoid this, the rectifier bridge can be treated as one component and solved as an ordinary centralised model.

Using Modelica in a transmission line environment

Modelica is a powerful language for designing simulation models with a structure similar to the physical structure. The TLM-concept keep the objects separated through the simulation with benefits as numerical efficiency, linear scaling, parallel computation etc. A solution where Modelica components are implemented in a TLM-environment would have the potential of being a powerful alternative to the traditional implementation using the centralised technique.

In this work, a tool for automatic translation of single Modelica components has been created using the Mathematica package [11]. Mathematica is a symbolic math package, enabling high-level programming. Besides the mathematical processing, advanced text operations can be performed as well. The tool translates a Modelica component to a component written in Fortran that can be inserted in a HOPSAN [5] simulation model.

HOPSAN is a simulation package using the transmission line modelling technique. The models that are simulated in HOPSAN are normally written in the Fortran language, and components are stored in a library. System models are created using a graphical modelling tool [9]. The following sections will demonstrate how a Modelica component is adapted to fit in a TLM environment.

Translating Modelica components

Distributed modelling using the TLM-method requires models described in a specific way. The components should have extra variables for the wave propagation and a solver for independent solving of the equations in the component. The Modelica component could be stated directly in that way, still following the Modelica syntax. It is desired however that the model designer should not have to care about issues that are specific to the implementation. The main advantage with Modelica is that the model code should be separated from the implementation. This means that the Modelica component must be manipulated to fit into the TLM-environment.

Extending a previously created tool in the Mathematica package, called *Compgen* [8], the component file written in Modelica can be processed according to the following sequence:

Step 1: Read the Modelica component file
Step 2: Find connectors, variables and parameters
Step 3: Find the equations describing the component
Step 4: Transform the equations to Mathematica notification
Step 5: Add the boundary equations according to the type of component
Step 6: Calculate partial derivatives of the equations
Step 7: Define the Jacobian from the partial derivatives
Step 8: Add a Newton-Raphson solver for the equations and write the component as Fortran code to a file

Through step 1-3, the text file containing the component is scanned by the Mathematica program. Using the Modelica syntax definition, the information is then sorted and grouped. The mathematical notification in Modelica and Mathematica is not the same, which is why a translation must be performed to be able to analytically manipulate the equations in Mathematica. As several components can be designed as either Q or C-components, it is for the user to select accordingly. Depending on the type, different variables are updated during the execution.

The tool is only able to translate a subset of the Modelica language. The effort has been focused on creating a tool that works for single components and the main purpose is to facilitate exchange of components between tools and people. The major limitation is that inheritance is not handled, which means that a Modelica component must be fully described with the complete set of equations.

The topology description

The topology of a system of components can in Modelica be described using the `connect` statement. This is not yet implemented in the tool presented here. It is though interesting to notice that the system description for the program generator DYNMOC, developed for the HOPSAN environment by Arne Jansson in 1997, has a very similar syntax compared to Modelica. In Figure 3 and Figure 4, the topology of the same system is described using Modelica and HOPSAN syntax. The minor syntactical differences imply that it would

be straightforward to create a translator also for the topological description, making it possible to simulate complete Modelica systems in HOPSAN.

```
model SimpleHydraulicSystem
   HydraulicPump P1;
   HydraulicMotor M1;
   MechanicLoad L1;
   Tank T1,T2;
equation
   connect(T1.n1, P2.n1);
   connect(P1.n2, M1.n1);
   connect(M1.n2, T2.n1)
   connect(M1.n3, L1.n1)
end SimpleHydraulicSystem;
```

Figure 3: Topological description in Modelica

```
component HydraulicPump P1
component HydraulicMotor M1
component MechanicLoad L1
component Tank T1
connect T1 n1 to P2 n1
connect P1 n2 to M1 n1
connect M1 n2 to T2 n1
connect M1 n3 to L1 n1
```

Figure 4: Topological description in HOPSAN

One problem is that in Modelica, the system description is necessarily not separated from the descriptions of the components. Connections are transformed to normal equations when the model is transformed to low-level code. HOPSAN uses a separate file to describe the connections on system level with the syntax shown in Figure 4. This means that to be able to transform a Modelica system to a distributed environment like HOPSAN, the code must be restricted. It must be stated what defines the topology of the system, expressed in a separate file, or separated between identifiers.

Example

Consider a simple system describing a hydraulic transmission. The system is modelled in the HOPSAN package using components from the component library. Using the model translator presented in this paper, a model of an electric motor written in the Modelica language can be automatically translated and connected to the hydraulic transmission.

This example can also illustrate a possible industrial scenario where engineers that are specialised in one specific engineering domain would need to import a model designed by engineers specialised in another domain.

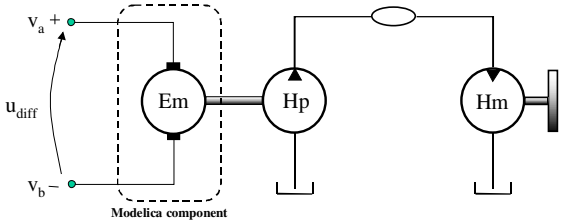

Figure 5: Hydraulic transmission in the HOPSAN environment

The hydraulic system consists of a pump and a motor with an intermediate volume. A mechanical load is attached to the hydraulic motor. To the hydraulic pump, a prime mover can be connected through a mechanical shaft, modelled as a stiff spring. This is where the Modelica model of an electric DC-motor is to be attached, see Figure 5.

System of equations

The simplest model of an electric DC-motor can be viewed as an inductance, a resistance and a converter describing relations between electrical and mechanical quantities, see Figure 6. The following equations describe the component:

$$\frac{di_a}{dt} = \frac{(v_a - v_b) - R_a \cdot i_a - \omega_x \cdot \psi_m}{L_a} \tag{7}$$

$$J \cdot \frac{d\omega_x}{dt} = i_a \cdot \psi_m - b \cdot \omega_x - M_x \tag{8}$$

$$i_a = -i_b \tag{9}$$

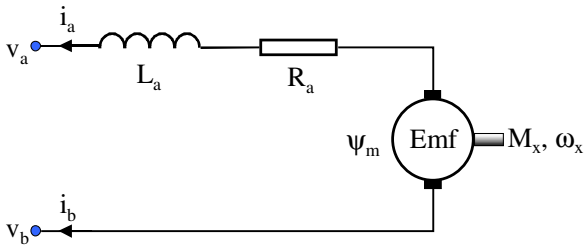

Figure 6: DC-motor circuit

If the electric motor is viewed as one component, there are three connections to surrounding components: two electrical nodes and one mechanical. For this component to fit in a TLM-environment it is designed as a Q-component. To model the exchange of energy and the wave propagation, four variables in each connector must be present. These variables are flow, effort, the characteristic and the characteristic impedance. In the component, this is handled by adding one extra equation for each

node describing the boundary conditions. The three boundary equations are:

$$v_a = c_a + Z_c \cdot i_a \tag{10}$$

$$v_b = c_b + Z_c \cdot i_b \tag{11}$$

$$M_x = c_x + Z_{cx} \cdot \omega_x \tag{12}$$

where c_a, c_b, and c_c are the waves that are calculated from the connected nodes. Z_c and Z_{cx} are the characteristic impedance also delivered from the connected components.

Solving the system

The model must be equipped with a numerical solver to become an independent component in the TLM system. This can be done using any suitable solver. In this work, the Newton-Raphson iteration algorithm is used based on that it has shown good robustness for stiff systems and works well in combination with the TLM-method. A short description of the steps performed inserting the solver to the component will be given below. For a detailed description of the process, see [8].

The equations (7)-(9) describe the internal relations in the component and are those that we found in the Modelica base component. Call this set of equations F_a and the boundary equations (10)-(12) is called F_b. The total set of equations can then be expressed as:

$$F(F_a, F_b, t) = F(y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, \ldots, \frac{d^ny}{dt^n}, t) = 0 \tag{13}$$

The following steps are:

- $F(F_a, F_b, t)$ is transformed into time discrete representation using bilinear transform. The result is called G.
- The Jacobian J is evaluated from partial derivatives of G.
- The equations are solved numerically using the Newton-Raphson iteration.

$$y_{k+1}(t) = y_k(t) - J_k(t)^{-1} G(y_k(t)) \tag{14}$$

After performing these iterations, the component can be written to a Fortran file with the correct node interface and the solver. The procedure translating the DC-motor written in the Modelica language has now resulted in a component that can be inserted into the HOPSAN environment and simulated. The system inserted in the graphical modelling tool *GDynmoc* [9] can be viewed in Figure 7.

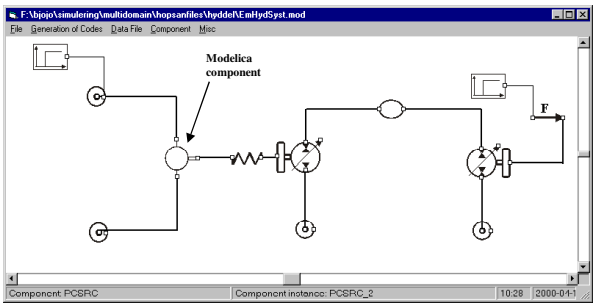

Figure The simulated system

Simulation results

From the complete system model in the HOPSAN environment, some simulation results can be obtained.

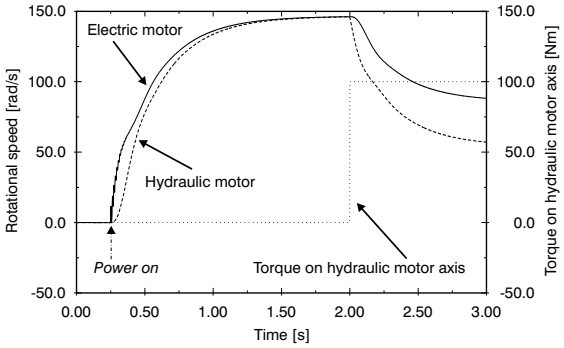

Figure : esults from the simulation

The simulation shows the speed of the hydraulic motor and the electric motor, see Figure 8. When the power to the electrical motor is turned on, the speed of the electric motor increases followed by a similar behaviour of the hydraulic motor. The delay is due to the volume in the hydraulic system, which also affect the behaviour when a step in load is applied. The speed of the hydraulic motor decreases faster and more than the electric motor.

The same model executed in HOPSAN has also been simulated in the Dymola environment [3]. The Dymola simulation package uses centralised solving and has full implemented support for Modelica. The results show good accordance between the different simulation packages.

Discussion

Object oriented modelling shows great advantages in the process of modelling physical systems. With characteristics as encapsulation, object instantiation and node connections, the simulation models show a structure that is flexible, logical and with

strong similarities to the physical system. Non-causal modelling removes the need of state-space models, reducing error-prone transformations. Instead, differential and algebraic equations from handbooks can be used directly.

The Modelica approach is a promising effort to a standard modelling language for dynamic systems. If a standard becomes widely accepted, it would mean a large step forward in simulation of multi-domain dynamic systems. As simulation packages are becoming more general and capable of performing simulation in different domains, the possibilities with exchanging models between different participants in a design group are significant.

The approach presented in this paper is an implementation of Modelica where component models are simulated in a distributed environment using the transmission line modelling technique. The demonstrated translator is capable of translating a subset of Modelica with the major limitation that inheritance is not implemented. The limitation is only of significance for the import function to HOPSAN and has been considered of secondary importance in this first stage. The most important thing is that standard Modelica code can be used without modifications.

The outlook using an object-oriented modelling language in a distributed environment is to handle large simulation models by partitioning the system not only through the modelling process but also through the simulation. A flexible, numerically robust and efficient environment can be achieved suited for large and complex systems. The distributed approach also facilitates connections with hardware and external simulation packages either locally or over a network. Using a standardised language as Modelica would also add advantages as increased integration between project members and supported exchange of knowledge between engineering disciplines.

Conclusions

By using an object-oriented modelling language together with a distributed modelling technique, the objects in a system model can be kept partitioned as objects through both modelling and simulation. The main advantages concern flexibility, scaling properties and possibilities to have simulation models distributed over a network. In this paper, it has been demonstrated how to use Modelica in a distributed environment. Components written in standard Modelica code are automatically translated to fit in the distributed HOPSAN environment.

The authors consider the presented approach with Modelica in combination with transmission line modelling to be a powerful alternative to the centralised approach. The advantages presented will probably be important issues using simulation as a tool in product development.

eferences

[1] Auslander D. M., Distributed System Simulation with Bilateral Delay-Line Models, *Jo na of a ic nginee ing T an M* , pp. 195-200, June 1968.

[2] Cellier F. E., Object-Oriented Modeling: Means for Dealing With System Complexity, in *oceeding of t ene x Meeting on y tem and Cont o* , The Netherlands, 1996.

[3] Elmquist E., Br ck D., and Otter M., *Dymo a e Man a* , Dynasim AB, 1999.

[4] Jansson A. and rus P., Real-time simulation using parallel processing, in *oceeding of T e nd Tampe e nte nationa Confe ence on F id o e* , Tampere, Finland, 1991.

[5] Jansson A. and rus P., HOPSAN a Simulation Package User s uide , http: hydra.ikp.liu.se hopsan.html, 1998.

[6] Jansson A. and Palmberg J.-O., Load Simulation, a flexible tool for assessing the performance of hydraulic valves., in *oceeding of FL C M T e Fo t T iennia nte na tiona ympo i m on F id Cont o F id Mea ement and i a i ation*, Toulouse, France, 1994.

[7] Johns P. B. and O Brien M., Use of transmission line modelling (t.l.m) method to solve nonlinear lumped networks, *T e Radio ec t on and nginee* , vol. 50, pp. 59-70, 1980.

[8] rus P., An Automated Approach for Creating Components and Subsystems for Simulation of Distributed Systems, in *oceeding of t at nte nationa F id o e o k op*, Bath, U , 1996.

[9] Larsson J., Dynmoc User s guide , I P-R-1088, Dept. of Mech. Eng., Div. of Fluid and Mech. Eng. Systems, Linköping University, Linköping, Sweden, 1999.

[10] Modelica Design roup, Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification , http: www.Modelica.org, 1999.

[11] Wolfram S., *T e M T M T C ook*, 4th revised ed, Cambridge University Press, 1999.