



Norwegian University of
Science and Technology

Master's degree thesis

IP501909 MSc thesis, discipline oriented master

Component-based Modelling and Simulation for Marine
Crane System Design

140158/Yu Li

Number of pages including this page: 84

Aalesund, 03/06/2016

Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. **Failure to complete the statement does not excuse students from their responsibility.**

Please complete the mandatory statement by placing a mark <u>in each box</u> for statements 1-6 below.		
1.	I/we hereby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than is mentioned in the paper/assignment.	<input checked="" type="checkbox"/>
2.	I/we hereby declare that this paper <ol style="list-style-type: none"> 1. Has not been used in any other exam at another department/university/university college 2. Is not referring to the work of others without acknowledgement 3. Is not referring to my/our previous work without acknowledgement 4. Has acknowledged all sources of literature in the text and in the list of references 5. Is not a copy, duplicate or transcript of other work 	Mark each box: <ol style="list-style-type: none"> 1. <input checked="" type="checkbox"/> 2. <input checked="" type="checkbox"/> 3. <input checked="" type="checkbox"/> 4. <input checked="" type="checkbox"/> 5. <input checked="" type="checkbox"/>
3.	I am/we are aware that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8 and Examination regulations .	<input checked="" type="checkbox"/>
4.	I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check	<input checked="" type="checkbox"/>
5.	I am/we are aware that NTNU will handle all cases of suspected cheating according to prevailing guidelines.	<input checked="" type="checkbox"/>

6.	I/we are aware of the NTNU's rules and regulation for using sources.	<input checked="" type="checkbox"/>
----	--	-------------------------------------

Publication agreement

ECTS credits: 30

Supervisor: Houxiang Zhang, Vilmar Æsøy

Agreement on electronic publication of master thesis

Author(s) have copyright to the thesis, including the exclusive right to publish the document (The Copyright Act §2).

All theses fulfilling the requirements will be registered and published in Brage, with the approval of the author(s).

Theses with a confidentiality agreement will not be published.

I/we hereby give NTNU the right to, free of charge, make the thesis available for electronic publication: yes no

Is there an [agreement of confidentiality](#)? yes no

(A supplementary confidentiality agreement must be filled in and included in this document)

- If yes: Can the thesis be online published when the period of confidentiality is expired? yes no

This master's thesis has been completed and approved as part of a master's degree programme at NTNU Ålesund. The thesis is the student's own independent work according to section 6 of Regulations concerning requirements for master's degrees of December 1st, 2005.

Date: 03/06/2016

Preface

This master thesis is going to develop a hydraulic components library for modelling and simulation for marine crane system design. The topic of this thesis is “Component-based modelling and simulation for marine crane system design”. I got some helps from doctor candidate Yingguang Chu who is main technique assistant for this project research. The project is carried out only in software 20sim based on both Modelica-based methods and BG method without lab or field research, while the behaviour classification method is summarised from both OOM approach and Statechart description. The paper shall be edited as a research report in English including methodology survey, description and specification of applied methodology, description of component models, description of clear behaviour hierarchy for component models and how crane system performance will be, discussion and conclusion with a proposal for further work. All the detail language for modelling will be exported from 20sim posted in Appendix.

Summary

This master project is proposed as a part of the project Next Generation Simulator for Marine Crane Design and Operation-Virtual Crane Prototyping System which is supported by a cooperation of four local agencies in Ålesund, Norway. Since the spring 2016 semester started, I paid all my efforts on this master project. During the five months researching and writing of this mater thesis, the problems were coming one by one, I really want to thank the doctor candidate my partner Yingguang Chu who gave me the most technical support and supervisors Houxiang Zhang and Vilmar Æsøy who helped me to complete this master project. Meanwhile, I want to thank my girlfriend Rebecca Cooper who helped me to review the writing of this master thesis. Last, I wish all the work I have done that could contribute to the future science and technology world.

Table of content

Preface.....	5
Summary	6
List of figures	1
List of tables	4
1. Introduction	5
1.1. Background	6
1.2. Objectives.....	6
1.3. Scope and Architecture of Component Model Library.....	7
1.4. Modelling Tool	7
1.5. Literature	8
2. Method	9
2.1. Bond Graph	9
2.1.1. Knowledge from the OOM point of view	9
2.1.2. Bond Graph in Hydraulic System	10
2.1.3. Advantages	10
2.2. Object Oriented Modelling Approach.....	11
2.3. Statecharts	14
2.3.1. Orthogonality: Independence and Concurrency.....	15
2.3.2. Additional statecharts features	15
2.3.3. Actions and activities	16
2.4. Modelica.....	16
2.5. Modelling Method.....	17
3. Component Model Development	19
3.1. Hydraulic fluid	19
3.1.1 Hydraulic fluid selection	20
3.1.2 Compressibility	21
3.2. Restriction	22
3.2.1. Calculation for Laminar Flow	23
3.2.2. Calculation of Discharge Coefficient for Turbulent Flow through Orifices.....	23
3.2.3. Calculation for Orifices modelling	24
3.2.4. Cavitation	24
3.3. Hydraulic Cylinder.....	25
3.3.1 Behaviour description in statechart.....	26
3.3.2 Model Classification	28
3.4. Directional Control Valve	30
3.4.1. The first model layer	32
3.4.2. The second model layer	33
3.4.3. Bond Graph model	35
3.4.4. Behaviour description in Statechart	35
3.5. Counterbalance Valve	39
3.5.1. Description in layers	40
3.5.2. Bond Graph Model.....	42
3.5.3. Behaviour description in statechart.....	43
3.5.4. Model classification	44
3.6. Pump	46
3.6.1. The first model layer	47
3.6.2. The second model layer	48
3.6.3. The third model layer	49

3.6.4.	The fourth layer model.....	51
3.6.5.	Model classification	52
3.7.	Hydraulic motor	54
3.7.1.	Model classification	55
3.8.	Pipe Flow	56
3.8.1.	Bond Graph model	57
3.8.2.	Behaviour description in Statechart	58
3.8.3.	Model Classification	62
3.9.	Control Volume.....	62
4.	Hydraulic power system for Marine crane.....	63
5.	Simulation Result.....	66
5.1.	Simulation result of cylinder model	66
5.2.	Simulation result of “Ideal model”	68
5.3.	Simulation result of “Standard model”	69
5.4.	Simulation result of “Advanced model”	70
6.	Conclusion	72
7.	Further Work.....	73
	Reference.....	74
	Appendix.....	75

List of figures

Fig. 1.1 Hydraulic lifting system in 20sim.....	8
Fig. 2.1 The architectural view of ULM	11
Fig. 2.2 The composite structure diagram of Cylinder model	12
Fig. 2.3 The use case diagram of Cylinder model.....	13
Fig. 2.4 The sequence diagram of cylinder model	13
Fig. 2.5 The activity diagram of cylinder model (Second layer)	14
Fig. 2.6 Example of statecharts	15
Fig. 2.7 The conceptual design of the object-oriented component model	18
Fig. 3.1 The properties of Hydraulic Fluid	19
Fig. 3.2 The property of different type of hydraulic oil	20
Fig. 3.3 The symbol of cooling.....	21
Fig. 3.4 Hydraulic cylinder physical model	25
Fig. 3.5 Deflection of the cylinder seal	26
Fig. 3.6 The Statechart map of hydraulic cylinder.....	27
Fig. 3.7 The BG model of the hydraulic cylinder “Ideal model”	28
Fig. 3.8 The BG model of the hydraulic cylinder “Standard model”	29
Fig. 3.9 The BG model of the hydraulic cylinder “Advanced model”	30
Fig. 3.10 ‘Y’ type 4/3 way DCV.....	31
Fig. 3.11 Use case diagram of hydraulic DCV	32
Fig. 3.12 Hydraulic DCV composition model	32
Fig. 3.13 DCV – Composition model of first layer	33
Fig. 3.14 DCV – Sequence diagram of first layer.....	33
Fig. 3.15 DCV – composition model of second layer.....	33
Fig. 3.16 DCV – sequence diagrams of second layer	34
Fig. 3.17 The base BG model of DCV	35
Fig. 3.18 The Statechart analysis of DCV	38

Fig. 3.19 The un-clustering of spool motion.....	39
Fig. 3.20 The symbol of CBV	40
Fig. 3.21 Use case diagram of CBV.....	41
Fig. 3.22 Composition diagram of CBV	41
Fig. 3.23 Sequence diagram of CBV	42
Fig. 3.24 Bond graph model of CBV	43
Fig. 3.25 Statecharts analysis of CBV	43
Fig. 3.26 The low level valve piston statechart for Open and Close	44
Fig. 3.27 The BG model of the “Advanced model”.....	46
Fig. 3.28 The pressure compensated pump symbol	46
Fig. 3.29 Use case diagram of hydraulic pump.....	47
Fig. 3.30 Pump composition model	47
Fig. 3.31 The composition model of pump in first layer	48
Fig. 3.32 Sequence diagram of pump in first layer	48
Fig. 3.33 Composition model of pump in second layer	49
Fig. 3.34 Activity diagram of pump in second layer	49
Fig. 3.35 Composition diagram of pump in third layer.....	50
Fig. 3.36 Activity diagram of pump in third layer	51
Fig. 3.37 Activity diagram of pump in fourth layer (Part).....	52
Fig. 3.38 The Ideal pump model	53
Fig. 3.39 The Standard pump model.....	53
Fig. 3.40 The advanced pump model	54
Fig. 3.41 The symbol of the variable displacement hydraulic motor	54
Fig. 3.42 The Standard BG model of the hydraulic motor.....	56
Fig. 3.43 The composition diagram of pipe flow	56
Fig. 3.44 The activity diagram of the pipe	57
Fig. 3.45 The single-lumped Bond Graph model.....	57

Fig. 3.46	Moody's diagram.....	60
Fig. 3.47	The analysis of statecharts for the pipe flow	61
Fig. 3.48	The Ideal Bond Graph model of the pipe flow	62
Fig. 4.1	The hydraulic system of Knuckle Boom Crane.....	63
Fig. 4.2	The simple hydraulic cylinder test.....	64
Fig. 4.3	The BG model of one actuator hydraulic system for Knuckle Boom Crane.....	65
Fig. 5.1	The displacement of the cylinder piston of three levels model	67
Fig. 5.2	The inlet&outlet flow of the cylinder models.....	68
Fig. 5.3	The pressure in the chambers of the cylinder models.....	68
Fig. 5.4	The simulation result of the "Ideal model" system.....	69
Fig. 5.5	The simulation result of the "Ideal model" system.....	70
Fig. 5.6	The simulation result of the "Advanced model" system	71
Fig. 7.1	The family tree of hydraulic pumps.....	73

List of tables

Table 3-1 The spool type of DCV.....	31
Table 3-2 Models of different valve types	46
Table 3-3 Volume and torque losses of axial piston pumps	50
Table 3-4 The volume and torque losses of axial piston motors.....	55
Table 3-5 Typical hose construction, dimensions, and operating pressure.....	58
Table 3-6 Local loss coefficient of typical local loss features	59
Table 3-7 Determination of the pipe line friction coefficient	59
Table 5-1 the parameters of three classified models	66

1. Introduction

In the domain of maritime industry, working efficiency and operating safety are always considered as the main challenges for maritime productions design. Since the current maritime crane design is far from perfect, this situation urgently demands innovation to develop a virtual prototyping framework for overall crane design process. The challenge of this virtual prototyping framework is the complexity of interdisciplinary system, which requires the model to have difference levels, to be effective (accuracy) and efficient (real time capability), to be used for different users. The overall crane design mainly includes the mechanical system and the hydraulic system design. During the design process, the modelling and simulation of the physics and dynamics is always a vital process before the hardware testing or manufacturing which is time consuming and costly. The important generic models are going to be built to realise a virtual prototype and then transfer the models to a real prototype, which should respond to all possible functions for all users. The primary object of this thesis is to develop an object oriented hydraulic component model library for simulation. Component model library is an organization that is a collection of component models with characteristics of modularised, reusable, declarative, capsulized and inheritance. Meanwhile, it is offering designers an efficient minor error and low-risk method to model and build a hydraulic system, and eventually upgrading. As most important objective and advances for this project in appropriate model development, the flexible component models of different size and complexity and which may be scaled in size, structure and complexity that are the most difficult and a challenge to realise. Each component model in our proposal will be implemented in three different behaviour models which are 'Ideal', 'Standard' and 'Advanced' that are used for system modelling and simulation for evaluating different design concepts in different phases of the design process, which is oriented to co-simulations using the virtual prototyping simulator based on the application of the Functional Mock-up Interface standard. From the design point of view, at the beginning of a system design, the "Ideal model" which is with simplest and most basic behaviour, is enough to compose a system. Then system simulation result will show the detailed dynamic performance by using the "Standard models". After finishing system design, modification process is needed to check some aspects in deeper research which needs to adopt the "Advanced model". On the other hand for fitting the co-simulation, the models could also be selected for the adaption of varying environment. As increasing high level requirements are need to simulate the dynamic behaviour of varying complex engineering system, physical systems modelling method Bond Graph (BG) and Object-Oriented Modelling language (OOM) Modelica has become to a more and more popular modelling approach. Fortunately, 20sim is such a software that is based on the BG supported by Modelica. Meanwhile, in order to better understanding of complex system properties of behaviours, an object-oriented approach to dynamic behaviour modelling is popular day by day, which is used to develop software that centres on objects combined data and functionality is becoming more favoured. For a modeller, the way to model a component is varying from one to another and it is rarely possible to model as simple as for the beginning of modelling. By adopting the approach of iterative way of model development from Dragan H, Pršić and Novak N. Nedić, it will guide designers to start with a good physical understanding of how a component works and thereby set up models that represent the most important physical effects. One notice that iterative approach is used for software development, we change it into a layer approach for the hydraulic engineering field, which is a process of modelling from first the simplest model layer until the last to completed model. Additionally, modeller could focus on the certain layer to refine the model or adjust the complexity of model. The benefit of OOM

methodology is to provide better understanding of working principles of components and to give more flexible modelling process and capability of model reuse.

On another hand, although OO approach provides the component model which is described by the structure through which different functions can be performed, it is difficult to define behaviour models “*Standard*” and “*Advanced*”. Statecharts as a visual formalism for specification and modelling of complex reactive systems proposed by Harel is a hierarchical approach to express behaviour, which help modeller to execute an in-depth analysis for defining the model class “*Standard*” and “*Advanced*”. It contributes to evaluating the dynamic performance or weighting new features for a component. It enables an evaluation to decrease the complexity of models as “Standard model” achieved by using semi-empirical models which capture all the most important characteristics of the component’s behaviours that is sufficient for simulation purposes. “Advanced model” will be pure physical models having all behaviour characteristics involved for research or investigate deeper details. However, pure dynamic models may cause simulation problems including time costing or computing errors because of fully complicated differential algebraic equations (DAEs).

This master thesis project aims at building an object-oriented hydraulic component model library by using BG method based on the combination of OOM approach and Statecharts. The following objectives will illustrate the investigation and execution process.

1.1. Background

Hydraulic systems’ modelling and simulation has often been applied in industrial manufacturing and heavy machinery for decades since the testing of system performance by using applications on real hydraulic systems had proven to be a difficult task due to the cost or size of the hardware and its working conditions. Depending on the results of system performance, it is required to reproduce the hardware, reconstruct the system or reset operating conditions, which is sometimes impossible or has high costs. Hence, a component model development and management oriented hydraulic library is necessary for modern modelling and simulation world for some purposes of (1) New users start modelling quickly; (2) Better productivity for experienced modellers; (3) No need to drive equations; (4) Easier to use based on the modellers’ view of both system performance and operational behaviour. There are already some available libraries existing in software like Maplesim, EASY5, SABER, here all the components and relative systems are modelled and tested in 20sim which is a toll for modelling and simulation of dynamic behaviour of engineering systems. The function of software 20sim is to help the engineers to model and simulate the systems in the process of design, analysis and diagnosis.

1.2. Objectives

- **To introduce a modelling approach for engineering system virtual prototyping**
Requirements and characteristics: Modularised, Reusable, Declarative, Capsulized, Inheritance
- **To develop the component models for hydraulic systems with different complexities**

Main components: HPU, Pipe Flow, DCV, CBV, Cylinder, Knuckle boom crane

- **To test the performance of the developed component models**
Cylinder performance and a simple sub-system of a Knuckle Boom Crane
- **To develop and manage components model library**
 - * Interface and parameters management in order to serve Virtual Crane Prototyping System

1.3.Scope and Architecture of Component Model Library

The aim of this paper is going to set up a hydraulic components library for the proposed virtual crane prototyping system, the main components include Hydraulic pump, pipe flow, 4/3 directional control valve, counter balanced valve, cylinder, hydraulic motor. Additionally, the behaviour classes' hierarchy for each component will not be limited, it will be based on the complexity of the component. Furthermore, the crane system for simulation test will adopt a sub-system of the hydraulic system in Knuckle Boom Crane. The hydraulic component model library is developed in a modular fashion, each model has three level behaviour models. First, the *Ideal* model is only with the main physical function without power losses, i.e. Hydraulic cylinder is only modelled with the function of energy exchange. Second, the *Standard* model, which contains most dynamic behaviours. But some behaviours are negligible, i.e. the air volume mixed in the oil will affects the effective bulk modulus that can be ignored in standard models and effective bulk modulus also can be a constant factor to regardless pressure effect. Last, there is no doubt that the *Advanced* model will cover all states of behaviours for deeper research, however, there is a risk exist that the model will be too complex to run the simulation.

1.4.Modelling Tool

20-sim is a dynamic behaviour modelling and simulation program for engineering systems in many domains based on the bond graph theory supported by OO language Modelica. With 20-sim modellers can enter model graphically, similar to drawing an engineering scheme. With these models you can simulate and analyse the behaviour of multi-domain dynamic systems and create control systems. Furthermore, it can be even generated of C-code and run this code on hardware for rapid prototyping and hardware-in-the-loop (HIL) simulation.

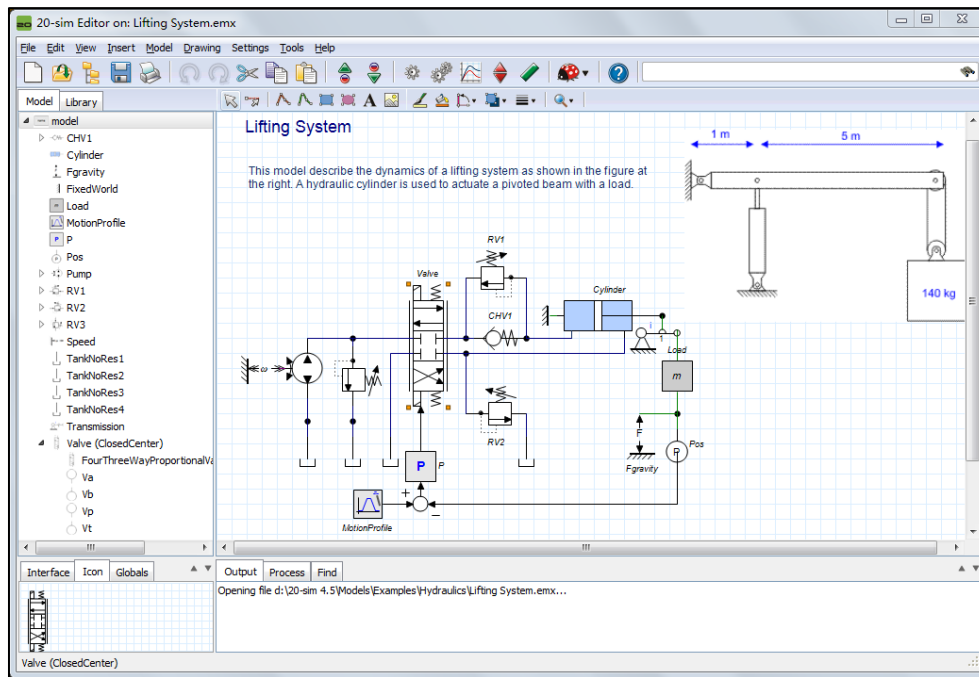


Fig. 1.1 Hydraulic lifting system in 20sim

20-sim is loaded with model libraries and toolboxes that will help modellers to create models more efficiently and analyse the results:

- 3D Mechanics
- Controller Toolbox
- C-code Generation
- Frequency Domain Toolbox
- and lots more

1.5.Literature

Bond Graph knowledge is mainly extracted from **Mathematical Modelling and Simulation of Physical Systems – Eilif Pedersen / Hallvard Engja;**
 Object Oriented Modelling approach is learnt from **Syllabus M.C.A. – Object Oriented Modelling and Design using UML;**
 Statecharts has been taken from **STATECHARTS: A VISUAL FORMALISM FOR COMPLEX SYSTEMS – David HAREL** and additional examples are from **Hybrid models for hardware-in-the-loop simulation of hydraulic systems – J. A. Ferreira, F. Gomes Almeida, M. R. Quintas and J. P. Estima de Oliveira.**
 Most formulas and theoretical knowledge of component models are from **Modeling of Hydraulic Systems-Tutorial for the Hydraulics Library**

2. Method

In this chapter, the applied theory and methods are presented. All the methods are combined for one purpose to help classifying and modelling hydraulic components.

2.1. *Bond Graph*

Bond Graph (BG) is a graphical approach as a network formalism for forming or decomposing a physical system into elemental physical properties through power exchange bond. It was invented by Professor H. M. Paynter at MIT in 1959. BG is a powerful modelling tool which has been applied in most dynamic domain and hybrid system domain.

2.1.1. Knowledge from the OOM point of view

Although these two modelling approaches are normally used in different domains, which BG modelling approach is for a system design in multiple energy domains involved, the OOM approach is more often used for a software development, the two modelling approaches have much in common. The essential features sharing by both the modelling approaches are discussed as following.

- *Hierarchy*: A model of a system may be composed of sub-models which in turn may contain sub-models as well. That is, models are hierarchical in nature.

The development of hierarchical models is supported by the concept of word bond graphs, viz. vertices representing the bond graph of a submodel are denoted by a word (in an ellipsis)

- *Inheritance*: If a submodel class is instantiated into a model its properties are inherited by the model.

Instantiation from generic BG models also mean that properties of the super class are inherited by the instantiated model. The generic model of a store for instance captures the properties of passivity, of storing a physical quantity expressed by a state variable, and of being energy conservative. With regard to the constitutive relations it is only determined which variables are related. These are properties inherited by any instantiation of the model class called store. By annotating the BG vertex, additional information specifying the constitutive relations is given. If the actual constitutive equations of a storage element do not comply with Maxwell's reciprocity condition, then the model has been instantiated inconsistently from the corresponding storage class.

- *Encapsulation*: Knowledge contained in a model is encapsulated. Only a well-defined part of it may be accessed in a well-defined manner via interfaces to outside world of an object.

The principle of encapsulation of knowledge is also employed. Submodels in a BG representing either components or elementary physical processes can be accessed only via their interfaces. On the component level, these are called plugs. On the level of physical processes, the interfaces are power ports and signal ports respectively. A storage element for instance does not pass the information carried by the state variable via the ports to other incident vertices. That is, the state variable is an encapsulated information.

Although the notions of knowledge encapsulation and inheritance are not common in the BG method older than the OOM paradigm, as the features introduced above these modern

notions denote modelling principles that are also well known in BG based physical systems modelling.

2.1.2. Bond Graph in Hydraulic System

In the hydraulic system, the fluid is treated as bond flow transmitting the energy from component to component. The variables are defined for hydraulic system as shown below:

	Effort (e)	Flow (f)	Momentum (p)	Displacement (q)
Hydraulic	Pressure [Pa]	Volume flow rate [m ³ /s]	Pressure momentum [N/m ² s]	Volume [m ³]

Furthermore, the procedure for hydraulic systems is a systematic construction procedure which is

- 1) Establish a *0*-junction for each distinctive pressure
- 2) Insert the component models via a *1*-junction between the two appropriate *0*-junctions
- 3) Add pressure and flow sources to appropriate *0*-junctions
- 4) Assign power directions
- 5) Define all pressures relative to atmospheric reference pressure and remove the *0*-junction representing reference pressure and their connected bonds
- 6) Simplify the bond graph

2.1.3. Advantages

As a popular dynamic modelling language, BG has many advantages in engineering activity summarised by Eilif Pedersen and Hallvard Engja

- The same symbolism is used to represent the power interaction in a large selection of physical systems.
- On a graphical form, bond graphs display the energetic structure of complex systems with several energy domains in a way which is close to what is known as the physics of the system.
- A physical based sign convention can be shown directly on the graph, which is important when interpreting numerical results from simulations.
- The method is equally applicable for linear and non-linear systems.
- A unique feature of bond graphs is the display of causality on the graph. That is, it indicates which variable for an element is the independent variable or input and which variable is dependent or output variable.
- The bond graph method gives an algorithmic procedure for converting the graph into mathematical equations

- It can be directly entered and processed by a computer.

2.2. Object Oriented Modelling Approach

Originally, OOM is a system development approach encouraging and facilitating reuse of software components. As long as it developed, it is not only applied in software but most domains. OOM is an approach based on functions and procedures to develop a system by building self-contained modules or objects that can be easily replaced, modified, and reused. OOM approach to physical systems modelling may be characterised essential features that

1. Objects: Objects can be models of technical components as well as models of physical processes.
2. Model hierarchy: A model of a system may be composed of sub-models which in turn may contain sub-models as well. That is, models are hierarchical in nature.
3. Instantiation: Models and sub-models are instantiated from generic models or classes.
4. Inheritance: if a sub-model class is instantiated into a model its properties are inherited by the model.
5. Encapsulation: Knowledge contained in a model is encapsulated. Only a well-defined part of it may be accessed in a well-defined manner via interfaces to outside world of an object.
6. Connection of sub-models: Sub-models are connected according to the physical structure of the model.

The Unified Modelling Language (UML) is a standardized specification language for object modelling. UML in this project is used for model architecture development, it visually describes the logical and physical structure from the architectural view.

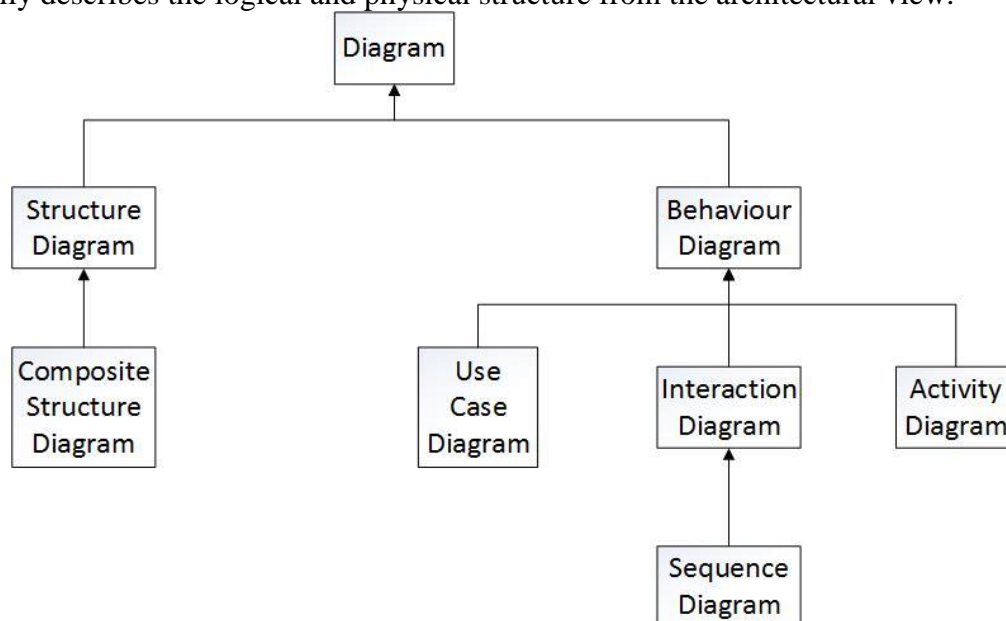


Fig. 2.1 The architectural view of UML

Composite Structure Diagram (CSD)

It shows the configuration and relationship of parts, which together, perform the behaviour of the containing classifier.

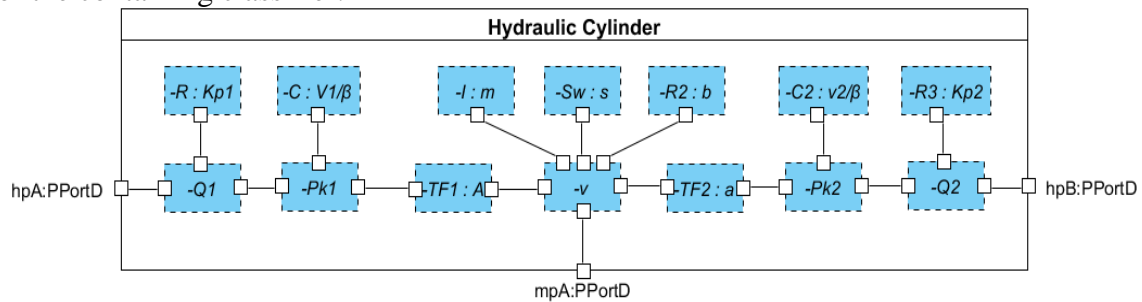


Fig. 2.2 The composite structure diagram of Cylinder model

In Fig. 2.2, it has all of the major objects to be found inside a typical hydraulic cylinder, together with BG transporting power and interacting each other. Each rectangular inside diagram is an *object*, the small square on the boundary of object is a *port* as interface for interaction between objects or environment. The objects can be labelled with both their BG symbols and mathematical symbols. The interfaces of ports are shown in diagram to interact with environment. *R-element* (K_{p1} and K_{p2}) stands for dissipation mechanisms of orifice on cylinder input or output, *C-element* as a storage represents the liquid compressibility, *TF* is modelling the transformer between mechanical energy and hydraulic energy, *I-element* and *R-element* (b) are cylinder rod dynamic inertia and viscous friction to cylinder house respectively. *Sw* is piston position constraint to limit the displacement of piston.

Use Case Diagram (UCD)

UCD is used to capture the dynamic nature of a system, to gather the requirements which are mostly design requirements of a system including internal and external influences. It consists of use cases, actors and their relationships. UCD is different from other behaviour diagrams from four aspects which are main purposes.

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

UCD is considered for high level requirement analysis of a system. Therefore, when the requirements of a system are analysed, the functionalities are captured in use cases.

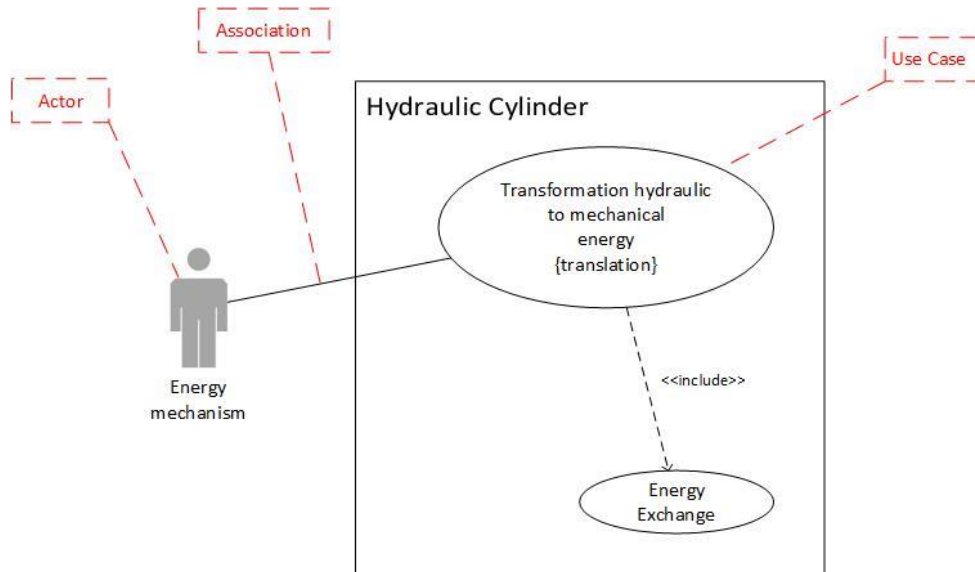


Fig. 2.3 The use case diagram of Cylinder model

In a hydraulic cylinder case, energy mechanism is being an actor that interacts with hydraulic cylinder. In the UML, an actor is shown as a "stylistic man figure" icon, or as a class marked with the actor keyword and labelled with the name of the actor class. The use case is shown graphically by the ellipse in which its function is described. Furthermore, the solid line means interaction and dash line represents include relationship.

Sequence Diagram (SD)

SD is used to capture time ordering of message flow that shows elements as they interact over time, showing an interaction or interaction instance. It is focused to represent interactions between objects for dynamic modelling.

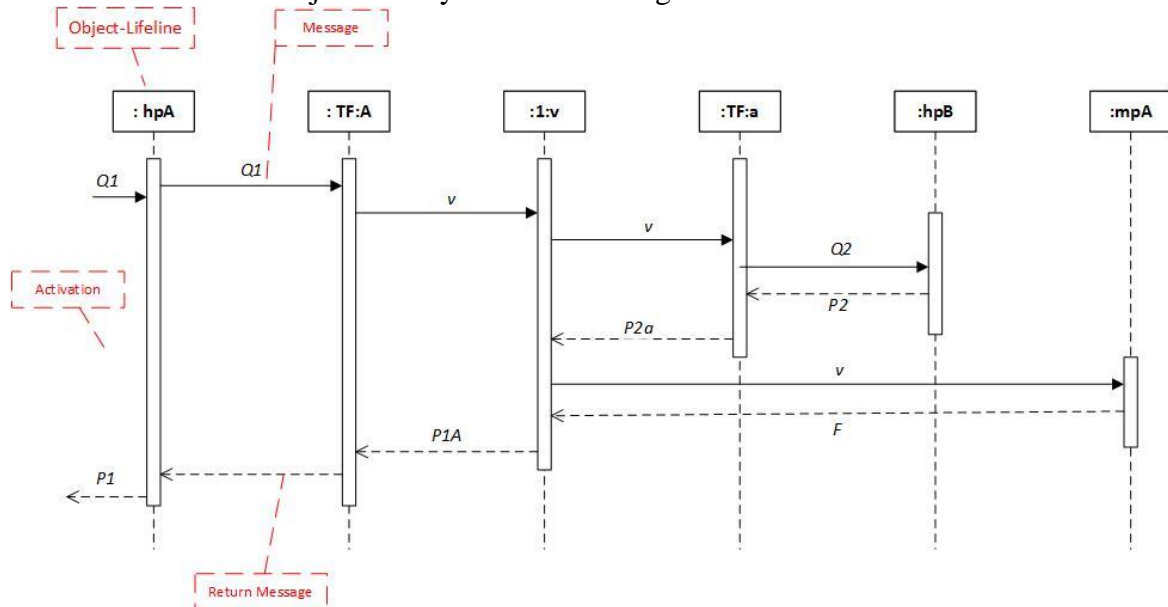


Fig. 2.4 The sequence diagram of cylinder model

As shown in Fig. 2.4, SD are made up of a number of elements, including class roles, specific objects, lifelines, and activations, which indicates the dynamic model through message processing context.

Activity Diagram (AD)

AD is basically a flow chart to represent the flow from one activity to another activity to describe the dynamic aspect of a system. The detail of cylinder model example is shown in Fig. 2.5.

Different from SD, an AD shows the actions for various objects to depict the operational workflows for functional modelling.

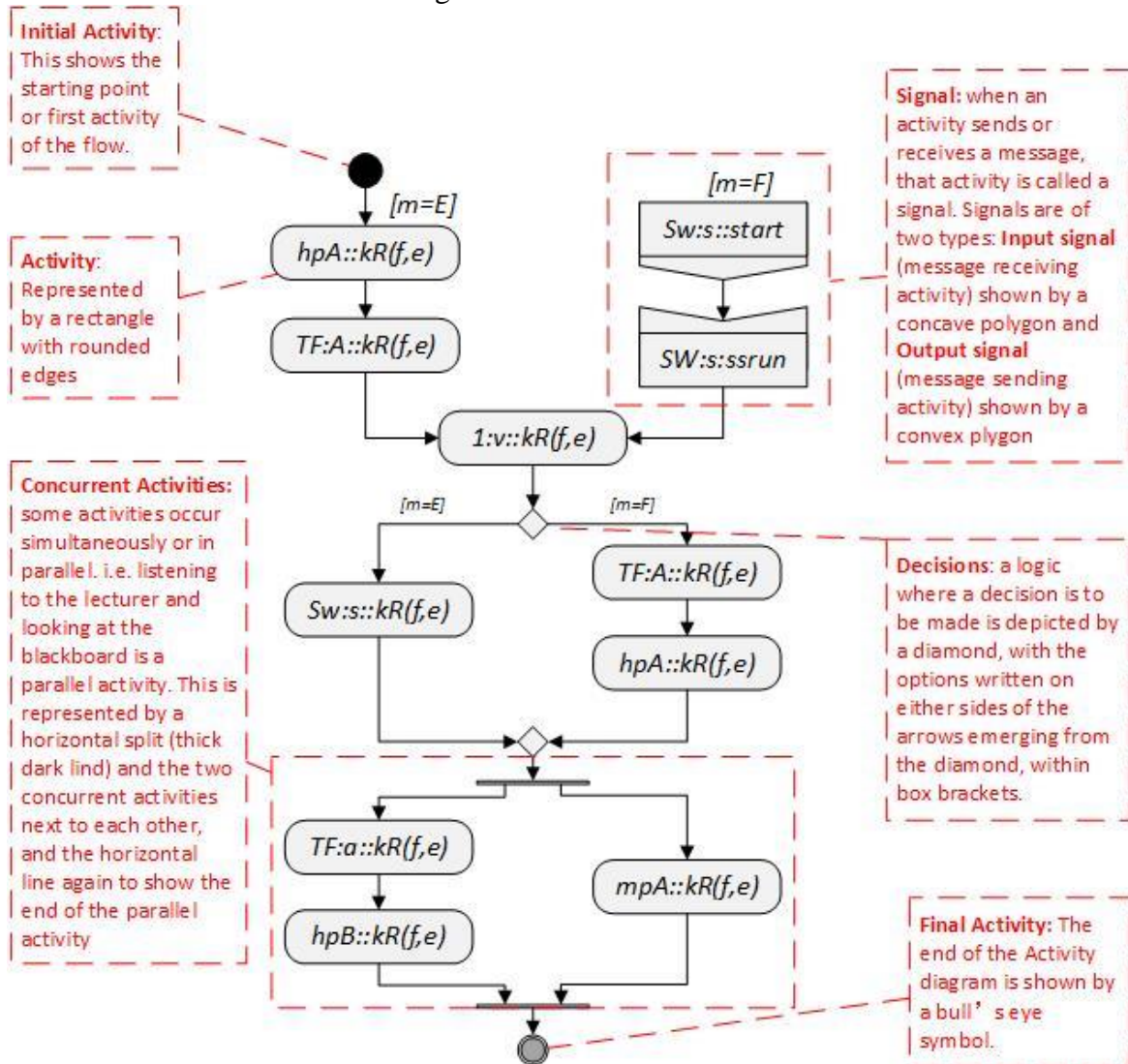


Fig. 2.5 The activity diagram of cylinder model (Second layer)

Advantage of Object Oriented methodology

OOM closely represents the problem domain. Because of this, it is easier to produce and understand designs.

The objects in the system are immune to requirement changes. Therefore, allows changes more easily.

OOM designs encourage more re-use. New applications can use the existing modules, thereby reduces the development cost and cycle time.

OOM approach is more natural. It provides nice structures for thinking and abstracting and leads to modular design.

2.3.Statecharts

Statecharts as a visual formalism proposed by David HAREL for the specification and modelling of complex reactive systems, extend conventional state-transition diagrams with essentially elements, dealing, respectively, with the notions of hierarchy, concurrency and

communication. Meanwhile, Statecharts, a graphical support, is suitable to describe the dynamic behaviour of complex component and system in Modelica and very useful for the hierarchical description of the model's dynamics. While it expands Finite State Machine (FSM) formalism with hierarchy, parallelism and broadcast communication. Technically speaking, the kernel of the approach is the extension of conventional state diagrams by AND/OR decomposition of states together with inter-level transitions, and a broadcast mechanism for communication between concurrent components. The two essential ideas enabling this extension are the provision for 'deep' descriptions and the notion of orthogonality. Since we strongly believe in the virtues of visual descriptions, the approach is described solely in its diagrammatic terms, although the reader should be able to provide a textual, language-theoretic or algebraic equivalent if so desired. In a nutshell, one can say:

Statecharts = state-diagrams + depth + orthogonality + broadcast-communication.

2.3.1. Orthogonality: Independence and Concurrency

The capabilities described in the previous section represent only one part of the story, namely, the XOR (exclusive or) decomposition of states, and some related concepts and notions. Meanwhile, AND decomposition captures the property that, being in a state, the system must be in all of its AND components. As shown in Fig. 2.6, Y is the orthogonal product of A and D , which state Y consisting AND components A and D , with the property that being in Y entails being in some combination of B or C with E , F or G . If event α then occurs, it transfers B to C and F to G *simultaneously*, resulting in the new combined state (C, G) . This illustrates a certain kind of *synchronization*: a single event causing two simultaneous happenings. If, on the other hand, μ occurs at (B, F) it affects only the D component, resulting in (B, E) . This, in turn, illustrates a certain kind of *independence*, since the transition is the same whether the system is in B or in C in its A component. Both behaviours are part of the *orthogonality* of A and D , which is the term we use to describe the AND decomposition. The "in G " condition causes A to depend somewhat on D , and indeed to 'know' something about the inner states of D .

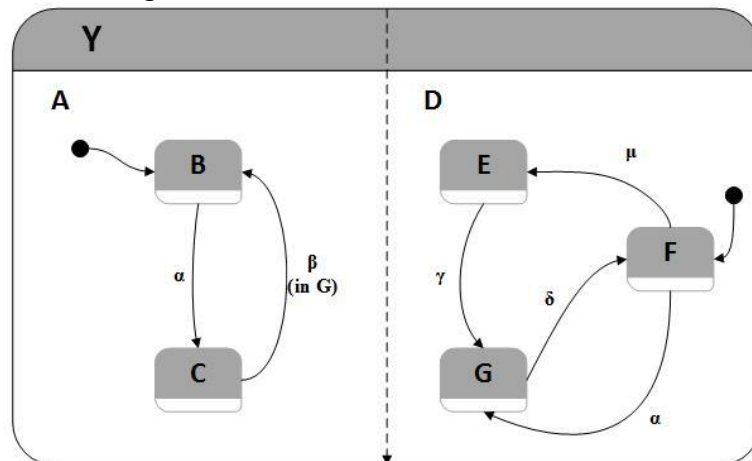


Fig. 2.6 Example of statecharts

2.3.2. Additional statecharts features

Condition and selection entrances

C-condition: If the actual conditions *and/or* the topology of the arrows are too complex, one can omit the details from the chart and use the simple incomplete form. The user will have to supply the full details separately and a computerised support system.

S-selection: It occurs when the state to be entered is determined in a simple one-one fashion by the 'value' of generic event, so that the event is actually the selection of one of a number of clearly defined options and the specifier has chosen to model those options as states

Delays and timeout

Timeout which represents the event that occurs precisely when the specified number of time units have elapsed from date. However, the need to limit the system's lingering in a state is something that occurs repeatedly in the specification of real systems.

Unclustering

It is the way of possibility to lay out parts of statecharts not within but outside of their neighbourhood. This conventional notation for hierarchical description has the advantages of keeping the neighbourhood small yet the parts of interest large.

2.3.3. Actions and activities

Action, which is an ability of statecharts to generate events and to change the value of conditions, which is carried out by the system for split-second happenings, instantaneous occurrences that take ideally zero time.

Activities, which are to actions what conditions are to events. An activity always takes a nonzero amount of time, thus, activities are durable – they take some time – whereas actions are instantaneous.

Obviously, these are all basic definitions for statecharts. In order to apply statecharts for describing the dynamic behaviour of complex systems, the more detail and examples need to be study deeper in Harel's paper.

2.4. Modelica

As a new object-oriented language for hierarchical physical modelling, Modelica has been used for many years since been announced in September 1997 through an international effort. It is a modern language built on non-causal modelling with algebraic and differential equations, and uses object-oriented constructs to facilitate model reuse, through hierarchical modelling, encapsulation, and inheritance.

Models and sub- models are declared as classes with connection interfaces called *connectors*. This connection capability allows the use of model libraries to compose complex models with the drag and drop, and connection drawing facilities of modern graphical editors. By using Modelica, the modelling of hydraulic component and system is supported via mixed continuous/discrete systems of equations. For the model with discontinuous case inside, it can be computerised with *if-then-else* expressions, allowing the modelling of behaviour with different expressions in varying operating regions. Models with different complexity levels can be supported by the adoption of conditional equations, in such a way that changes on behaviour are obtained by adding or setting a parameter of state. Discrete event and discrete time models are supported by *when* statements. The equations in a *when* clause are conditionally activated at event instants *where* the *when* condition becomes true.

Compared with the widespread simulation languages available today Modelica language offers three important advances: 1) non-causal modelling based on differential and algebraic equations; 2) multi-domain modelling capability, i.e. it is possible to combine electrical, mechanical, thermodynamic, hydraulic etc. model components within the same application model; 3) a general type system that unifies object-orientation, multiple inheritance, and templates within a single class construct.

In this project, the Modelica language as a support of BG to model hydraulic components in software 20sim.

2.5. Modelling Method

The applied method holds in the combination of an OOM approach for model both static and dynamic description which give the conceptual model, equation based modelling (*BG* and *Modelica*), with a virtual formalism (*Statecharts*) suitable for the dynamic behaviour description and modelling of hydraulic components. The leading principles to implement the dynamic behaviour models of hydraulic components are that object-oriented libraries of hydraulic component models; three levels of behaviour model complexity achieve different simulation experiments; ability to refine or redefine behaviour; graphical description of dynamic behaviour to enhance model understanding; Last, model interconnection would be investigated for building even more complex models and combining to other domains like electrical, mechanical and thermal, etc. as an *interface* to co-simulation for the virtual prototyping framework for overall next generation crane system design.

The concept behind the modelling method for building a component model library is from the both static and dynamic aspects to analyse, to model a hydraulic component, as a composition of two perspectives that structure and behaviour. The composite structure diagram will show internal structure of a component and interfaces to the outside environment. Meanwhile, behaviour diagrams, such as use case diagram, sequence diagram, etc., will pay an effort on the dynamic part to describe the dynamic performance of components. As a result, the basic model can be modelled in BG.

By adopting the OOM concepts, component or system can be decomposed into structure elements that clearly show different functions that can be performed. It is always difficult to decide where to model a component or system as a first step, OOM layer description is beneficial to acknowledging designer a process to know system better, then to model from the basic to complex until complete. Meanwhile, model architecture own benefits that are easy-refine and easy-maintain, which can be easily adapted to new needs. Meaningfully, architecture of a model also give the characters of usable, organisable and manageable. It is based on the layer description to give a model architecture, enables designer to start from the simple model including only the basic state but important concepts. Therefore, each layer results in an increment which gives improved functionality. And it has its typical details which means that other layers are not necessary to consider risks or emerged problems from certain layer, they are relatively isolated. In the method, the *physical graph* shows the component's construction, *use case diagram* tells the functions and requirements of components of how they work, *composite structure diagram* describes internal structure classifier and interfaces that interacts with environment and objects connection inside of models, furthermore, *sequence diagram* and *activity diagram* demonstrates how model works or how objects work together.

On the other hand, states and events are considered a rather natural way to describing the dynamic behaviour of complex components. Hierarchy, as a well-accepted approach in *Statecharts*, is used to group sets of states together, allowing high level description and step-wise development. In model organisation process, the model's behaviour is hierarchically detailed by nesting statecharts. By an advantage of virtual formalism approach, it is always possible to inherit and refine model's behaviour in a very understandable manner, and with well-defined rules; this allows the organization, for the same component, of models whose behaviours can have various complexity stages suitable for different simulation experiments. The user can always inhibit the behaviour of models and define a new behaviour, for a part or for the whole model in a very elegant way by adding or delating states, which is an obvious advantage of statecharts behaviour classification. However, a model can be never covered with all behaviour states, therefore, designer could choose some of states to check the simulation result during modification

process by using “Advanced model”. In this project, the statecharts description may not cover all behaviour states, the states could be added by users for unique modelling purpose. Moreover, by combining the OOM approach of layer way and statecharts, when designer is going to specify a behaviour, they do not build the new behaviour model structure from the very beginning, but upgrade one of existing layer.

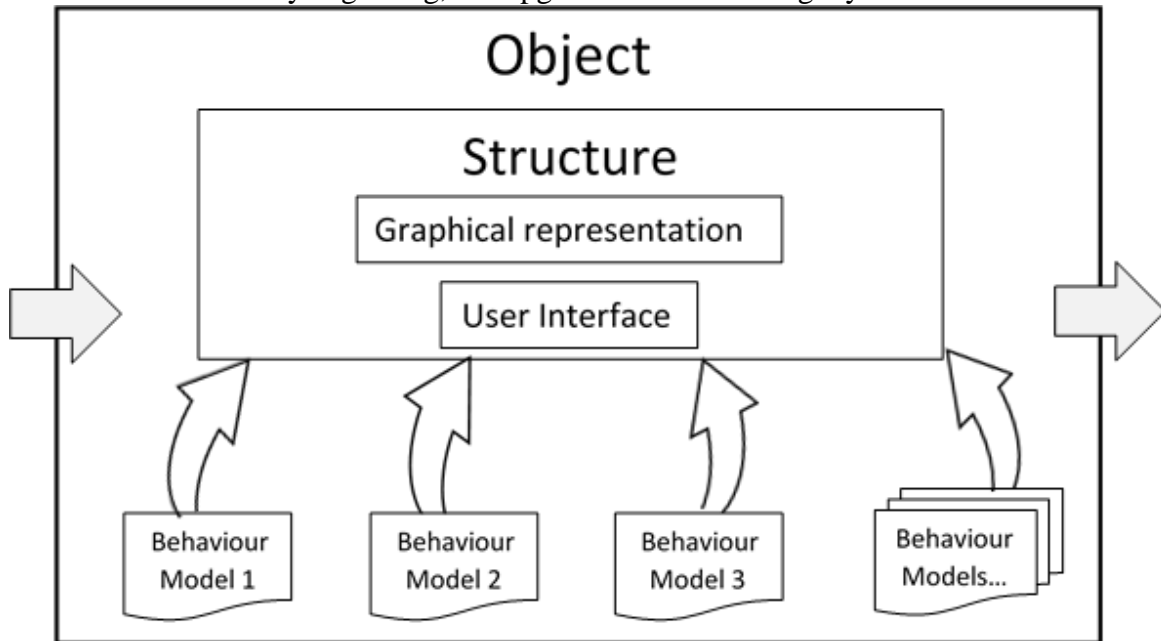


Fig. 2.7 The conceptual design of the object-oriented component model

The developed component models based on the OOM paradigm and Statechart description are depicted as shown in Fig. 2.7. The dynamic behaviour of the model is separated from its structure to make it easier for model encapsulation and inheritance. The structure model defines the representation and interfaces to the behaviour model. Model classification is dependant of the complexity of the modelling. There are three behaviour models for each component model in this project, the fourth shown in the picture is a potential part for more behaviour models in future. All behaviour models are sharing the same interfaces to the outside world.

3. Component Model Development

A model is a representation that mimic another object under study and should contain only those features of the system which the model builder feels are relevant and significant for the goal in mind. The hydraulic components are modelled in this chapter, they are analysed by using OOM approach firstly, and the behaviours of model are classified with adaptation of Statecharts. The modelling detail is exported profile from 20 sim that is attached in appendix.

3.1. Hydraulic fluid

As the energy transfer medium in hydraulic system, the hydraulic fluid is a key element to affect the behaviour of component and system performance. Meanwhile, every property of fluid is a critical feature to change the performance of system, the properties of fluid is shown in Fig. 3.1.

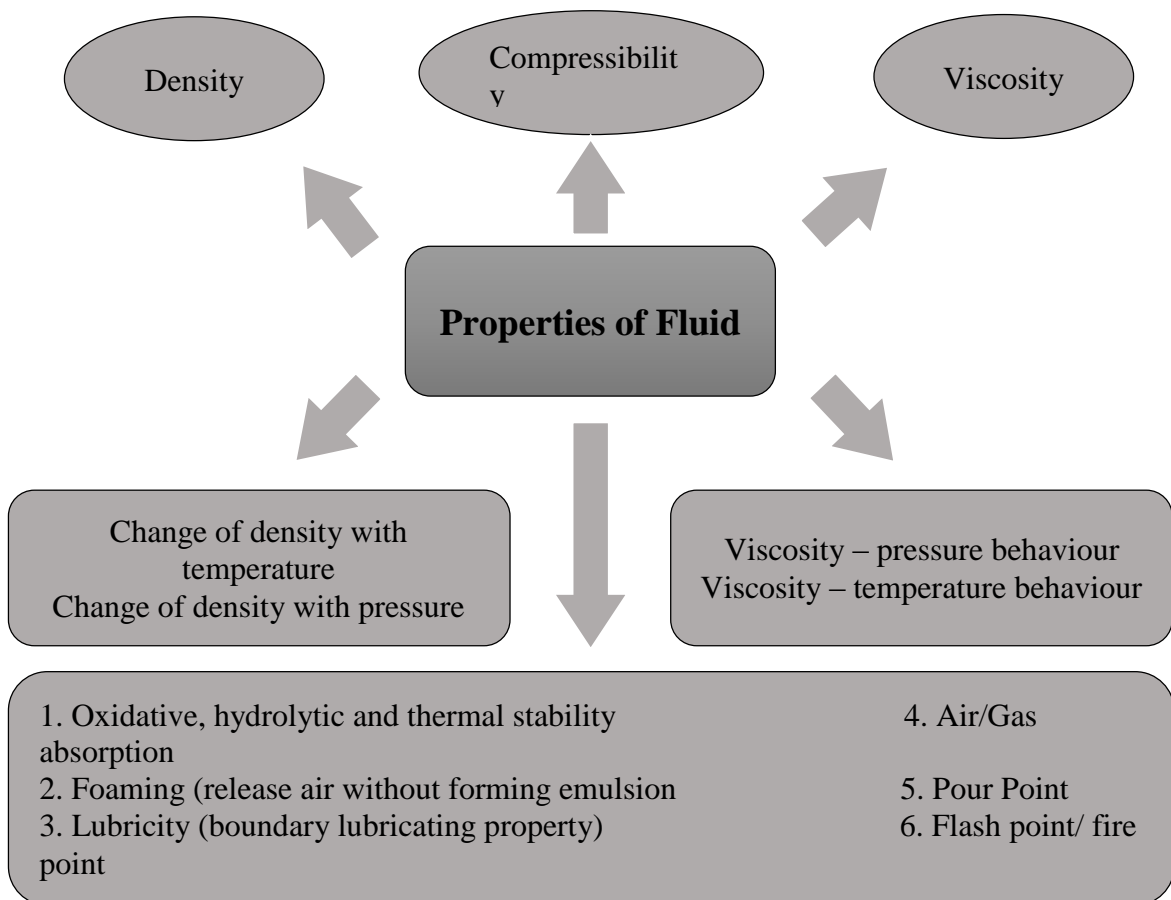


Fig. 3.1 The properties of Hydraulic Fluid

Apparently, there are such many properties that need to be considered for a hydraulic system, but it will be too complicated to model with the results of many errors or time waiting. Therefore, one type of hydraulic fluid is going to be chosen and assumed that the temperature in hydraulic system is constant at the common working condition.

3.1.1 Hydraulic fluid selection

There are different types of hydraulic fluids based on fluid properties that can be selected as energy transportation media in hydraulic system depending on the requirement of system. In this project, the effects of hydraulic fluid to system are not going to be analysed too much since we are only focusing on the OO behaviour modelling of hydraulic component and crane system performance. Therefore, the hydraulic fluid in analysis is assumed to use ISO VG 32 hydraulic oil. The property are shown as following.

	<u>ASTM</u>	<u>ISO VG 32</u>	<u>ISO VG 46</u>	<u>ISO VG 68</u>
Viscosity @ 40°C	D445	32.84 cSt	43.81 cSt	67.89 cSt
Viscosity @ 100°C	D445	5.62 cSt	6.74 cSt	8.90 cSt
ISO VG	-----	32	46	68
VI	-----	110	108	104
Color	D1500	L-0.5	L-0.5	L-0.5
API Gravity	D287	32.7	31.8	30.8
Specific Gravity	D1298	0.8618	0.8665	0.8718
Factor	-----	7.18#	7.22#	7.26#
Flash	D92	440°F	470°F	475°F
Fire	D92	480°F	510°F	510°F
Pour	D97	Minus 40°F	Minus 35°F	Minus 20°F
TAN	D974	0.3	0.3	0.3
Cu Corrosion	D130A	1b	1b	1b

Fig. 3.2 The property of different type of hydraulic oil

Temperature extremely have a pronounced effect on component behaviours as well as system performance. Temperature and viscosity have a converse performance as temperature is lowering, fluid viscosity is going to be higher. As a result, the hydraulic fluid often reaches the point where it actually congeals and will be even worse with no longer flow (pour point). High temperature also accelerates wear, destroys hydrodynamic lubrication regimes, increases the oxidation rate, fosters additive depletion and affects other critical aspects of the machine.

In order to avoid excessive oil temperature effect on component and system, we would like to keep the temperature at 45-55 Celsius degree which is common range for real working hydraulic system. Furthermore, most losses in a hydraulic system ends up as heat in the oil. To keep the temperature at a fixed range by cooling in terms of self-natural cooling and cooler adopting. Firstly, heat is dissipated from pipes, valves and the tank by natural and forced convection which is based on the equation

$$\text{Power} = A * K * \Delta T \text{ [W]}$$

A = the dissipation area

K = Heat transfer coefficient [w/(m²*K)]

ΔT = differential temperature [°C]

Secondly, a cooler will be placed where the pressure is low return line and/or offline in system. The symbol of a cooler in schematic of hydraulic system is shown in imagine.

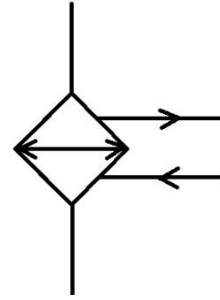


Fig. 3.3 The symbol of cooling

3.1.2 Compressibility

Transient flowrate in hydraulic system is always associated with important property of compressibility which defined as the change in volume per unit volume for unit change in pressure. The stiffness of the fluid is characterised by the bulk modulus β which is reciprocal of compressibility. The flowrate in system is proportional to rates of change of pressure expressed as

$$q = \frac{V}{\beta} \frac{dP}{dt}$$

Where V is the volume of oil involved, β is the effective bulk modulus and dP/dt is the rate of change of pressure of the oil.

Effective Bulk modulus

The value of the bulk modulus depends on the fluid, the pressure, the entrained air, the container and the temperature. In models' simulation, the effective bulk modulus is made constant in most time. However, when hydraulic system have to run over a wide pressure range, the pressure dependent effective bulk modulus is needed to achieve satisfactory simulation result. Especially for a hydraulic cylinder, the effective bulk modulus can be modelled for both cylinder chambers:

$$\beta_{e1} = \frac{10^5 + P_1}{BP_1 + C}, \quad \beta_{e2} = \frac{10^5 + P_2}{BP_2 + C}$$

where $B [Pa^{-1}]$ and C are constants related to the oil characteristics.

As the effective bulk modulus depends on a number of parameters, it should be measured with the actual component. There are many cases that describe models in different methods.

Dilation of container

Due to mechanical compliance, a dilation will be happened in an increasing pressure that effectively reduces the bulk modulus of the fluid. For a cylindrical container of inside diameter d_1 , outside diameter d_2 , made of material with Young's modulus E and Poisson's ratio ν , the effective bulk modulus is given by

$$\frac{1}{\beta} = \frac{1}{\beta_0} + \frac{d_1 + d_2}{E(d_1 - d_2)}$$

for thin-walled cylinder (thickness e less than $d_1/10$) or

$$\frac{1}{\beta} = \frac{1}{\beta_0} + \frac{2}{E} \left(\frac{d_2^2 + d_1^2}{d_2^2 - d_1^2} + \nu \right)$$

for thick-walled cylinders.

Where β_0 is fluid bulk modulus, d_1 is the inside diameter of container, d_2 is outside diameter of container, E is Young's modulus ($180 \cdot 10^9$ N/m² for steel), and ν is Poisson's ratio which 0.3 for steel.

Air content

The air in a hydraulic system will cause a drastic reduction of effective bulk modulus. The effect can be illustrated by assigning a 'bulk modulus' to the air itself. For pressure varying about some mean value P , this value of 'bulk modulus' is in fact equal to P .

Knowing the volume of the air presents per unit volume of the oil and assuming a perfectly rigid container, the effective modulus is estimated from

$$\frac{1}{\beta} = \frac{1}{\beta_o} + \frac{V_a}{V_o} \frac{1}{P}$$

Where V_a is the volume of the dispersed air in a volume V_o of the oil.

In real case, the air volume mixed in the oil is a very small amount, which will not affect the model behaviour too much. Therefore, the models in the library assumes that the air volume mixed in the oil is negligible which is omitted. If the designer is specialised to investigate the effect of the air content in a hydraulic system, the model can be refined based on the model in the library.

3.2. Restriction

In the “Ideal” model package, it will be considered about the resistance and other characters that will result energy lost and performance oscillations. However, when modelling the dynamic response of “Standard” and “Advanced” models, the resistance of these components somehow has to be considered. The resistance R-element is the ratio of pressure drop ΔP , across variable, to volume flow rate q , through variable:

$$R = \frac{\Delta P}{q} \left| \begin{array}{l} \Delta P = \text{constant} \\ q = \text{constant} \end{array} \right.$$

The inverse of the resistance is the conductance G:

$$G = \frac{1}{R}$$

The hydraulic flow running in a system that has two forms which are *Laminar* flow and *Turbulent* flow defined by Reynolds number. The Reynolds number – Re characterise the flow velocity and the kinematic viscosity of the fluid, Re is defined as

$$Re = \frac{\rho u D}{\nu}$$

u is flow velocity

ρ is fluid density

D is diameter of container (i.e. tube)

ν is kinematic viscosity

(Merritt 1967)

If the restriction has no circular cross section the diameter D can be approximated by the hydraulic diameter,

$$D_h = \frac{4A}{S}$$

A is flow section area

S is flow section perimeter

The critical value Re_{crit} of Reynolds number helps to define two form of hydraulic flow, if Reynolds number is less than Re_{crit} , the flow model is laminar flow, and if it is higher, the flow model turns in turbulent flow.

$R = \text{constant}$, then it has,

$$q = \frac{1}{R} \Delta P \sim \Delta P .$$

The linearity between pressure drop and flow characterises *laminar* flow.

$R = R(\Delta P, q)$, then it has,

$$q \sim \sqrt{\Delta P}$$

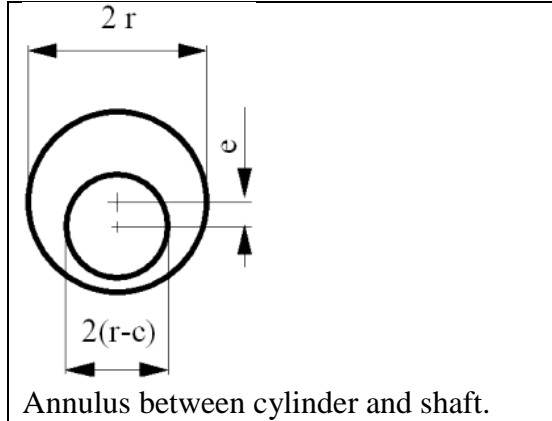
The “square root” dependency characterises *turbulent* flow.

The transition length from laminar to turbulent takes only a length of $10 \dots 20D$, while the transition from turbulent flow to laminar is about $0.03 * D * Re$.

3.2.1. Calculation for Laminar Flow

The typical example of laminar flow is the leakage in hydraulic system since the velocity of flow is tiny, i.e. leakage between the ports of a spool valve and leakage between two cylinder chambers.

It is assumed that the container section area is round, then it is writing as

 <p>Annulus between cylinder and shaft.</p>	$q = \frac{\pi r c^3}{6\mu L} \left[1 + \frac{3}{2} \left(\frac{e}{c} \right)^2 \right] (P_1 - P_2)$ <p>$c \ll r$ (Merritt 1967) valid for $Re < 1000$ (Ellman et al. 1995)</p> $q = \pi (r^2 - (r - c)^2) \sqrt{\frac{2(P_1 - P_2)}{\rho} \left(\frac{2c}{\xi L} + \frac{1}{1.5^2} \right)}$ $\xi = \frac{0.316}{Re^{0.21}} \frac{1}{1 + 0.2 \left(\frac{e}{c} \right)}$ <p>turbulent flow across annulus (Ellman et al. 1995), Bell and Bergelin (1957)</p>
--	---

3.2.2. Calculation of Discharge Coefficient for Turbulent Flow through Orifices

A typical example of turbulent flow is that the flow through a sharp edged orifice. When it is a thin sharp edged orifice, the turbulent flow can be modelled by the equation as

$$q = AC_d \sqrt{\frac{2}{\rho} \Delta P}$$

$A = \pi D_0^2 / 4$ which is section area

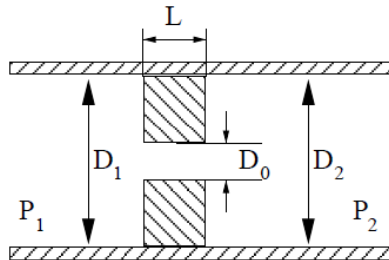
C_d is discharge coefficient

$\Delta P = P_1 - P_2$ which is pressure differential

The value of C_d depends on many parameters, mostly geometry and the condition of the orifice inlet. Even minor deviations from a sharp-edged inlet, such as roughness or a slight local radius, can produce a significant increase in the discharge coefficient (Ohrn et al. 1991). Meanwhile, the type of oil can also affect the discharge coefficient, but the oil type is already assumed to use ISO VG 32.

To make models simplest in most cases, value for C_d is treated as a constant varying between 0.6 and 1.0 that depends on whether the orifice is sharp edged or rounded.

Most orifice for real cases is actually consist of a short tube, shown as following



And in this situation, Merritt (1967) gave the equations for modelling this type of orifice.

$$C_d = \begin{cases} \left[2.163 + \frac{64L}{D_0 Re} \right]^{-0.5} & \text{if } \frac{D_0 Re}{L} < 50 \\ \left[1.5 + 13.74 \left(\frac{L}{D_0 Re} \right)^{0.5} \right]^{-0.5} & \text{if } \frac{D_0 Re}{L} > 50 \end{cases} \quad \text{with } Re = \frac{4q}{v\pi D_0}$$

3.2.3. Calculation for Orifices modelling

The ways to model orifices for hydraulic components are varying depending on the features and constructions of components, therefore, there are different models obtained from real case experiments. The goal of experiments are no different to gain the coefficient of resistance characteristic. Here equation is adopted which is also used in the German notional standard TGL to describe the resistance characteristic of fixed hydraulic resistors. In the library the model orifice uses the following equations to describe laminar/turbulent flow through fixed restrictions:

$$K = \frac{K_1}{Re} + K_2 \quad \text{and} \quad \Delta P = K \frac{\rho q^2}{2A^2}$$

The coefficient K_1 determines the response at low Reynolds numbers when laminar flow exists:

$$\lim_{Re \rightarrow 0} K = \frac{K_1}{Re} \Rightarrow \lim_{Re \rightarrow 0} q = \frac{\pi D^3}{2K_1 \nu \rho} \Delta P$$

for an annular restriction. K_1 depends on the type of restriction. For tubes with a circular cross section, K_1 is computerised as,

$$K_1 = \frac{64L}{D}$$

Turning to turbulent flow, K_1/Re goes to zero and K_2 determines K :

$$\lim_{Re \rightarrow \infty} K = K_2 \Rightarrow \lim_{Re \rightarrow \infty} q = \frac{\pi D^2}{4} \sqrt{\frac{2\Delta P}{\rho K_2}}$$

and $K_2 = \frac{1}{C_d^2}$.

Most cases are normally treating C_d as a constant coefficient for turbulent flow, *van mises(1917)* determined C_d that equals to 0.611.

3.2.4. Cavitation

Cavitation always happens at narrowest part of the flow path of a sharp edged orifice that when the velocity of the liquid is at its greatest and the pressure least, and the pressure is lower than the vapour pressure of the liquid which causes the appearance of bubbles. At the same time, the bubbles are going to collapse to result in cavities of equipment, which should be avoided in equipment design.

In order to judge that if the cavitation is happening or not, the equations is provided. At the condition that if the ratio length to diameter of an orifice is higher than 0.5, there is a critical pressure differential ΔP_k , if the actual pressure differential becomes higher than the critical pressure differential ΔP_k , cavitation will happen in system. This critical pressure differential ΔP_k is given by (Riedel 1973):

$$\Delta P_k = C_k^2 \left[\sqrt{\frac{P_1}{C_{D_{max}}}} + \frac{20\nu \left(1 + \frac{2.25L}{D_H}\right)}{C_k D_H \sqrt{\frac{2}{\rho}}} \right]^2$$

with: $C_k \approx 0.65$ and $C_{D_{max}} = 0.827 - \frac{0.0085L}{D_H}$.

Another model was given by Nurick (1976) and Schmidt and Corradini (1997), when cavitation is happening, C_d is taken as

$$C_d = \min\left(0.84, 0.61 \sqrt{\frac{P_1 - P_d}{P_1 - P_2}}\right)$$

3.3. Hydraulic Cylinder

Hydraulic cylinder (HC) is acting as an actuator in hydraulic system used to convert the fluid energy into mechanical energy. A HC consists of a piston and rod located within ported cylinder in a linear motion as shown in Fig. 3.4.

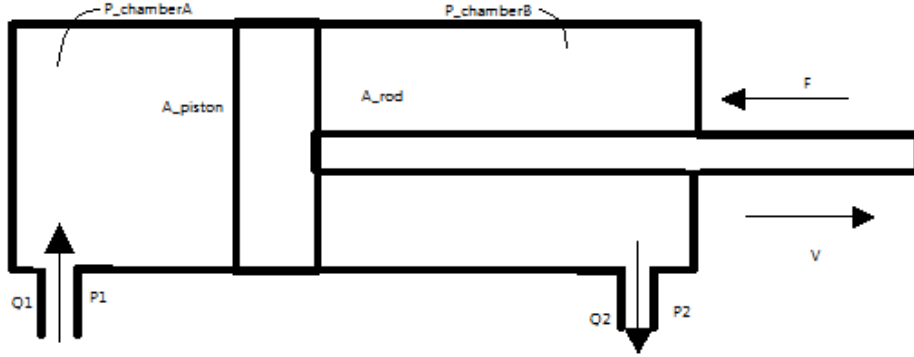


Fig. 3.4 Hydraulic cylinder physical model

For the real cylinder in a steady-state operation, the internal leakage Q_L and friction forces F_f should be taken into consideration. Therefore, the cylinder is described by relations,

$$F = P_1 A_{piston} - P_2 A_{rod} - F_f \quad \text{and} \quad v_{piston} = \frac{Q_1 - Q_L}{A_{piston}} = \frac{Q_2 - Q_L}{A_{rod}}$$

A_{piston} = Piston area [m^2]

A_{rod} = Rod side area [m^2]

F = Poston driving force [N]

F_f = Friction force [N]

P = Pressure in chambers [Pa]

Q = Flow rate runs in and out chambers [m^3/s]

Q_L = Internal leakage flow rate [m^3/s]

v_{piston} = Poston speed [m/s].

The inlet and outlet chamber flows are related to the piston velocity and compressibility effects. The oil compressibility is expressed in terms of the bulk modulus β . The continuity equation is normally taken for a constant oil bulk modulus,

$$Q_1 = G_{Leakage}(P_1 - P_2) + A_{piston}v_{piston} + \frac{A_{piston}x_p}{\beta} \frac{dP_1}{dt}$$

$$Q_2 = -G_{Leakage}(P_1 - P_2) + A_{rod}v_{piston} - \frac{A_{rod}(L - x_p)}{\beta} \frac{dP_2}{dt}$$

Where x_p is the piston position, $G_{Leakage}$ is the leakage conductance and L is the cylinder stroke.

In the real cylinder, there is always a stroke limitation. The piston movement constraint is modelled by a bumper. The bumper force can be programmed as a constant of infinite force to stop the piston or formula,

$$F_{Bumper} = ks + cv$$

where k =spring constant, s =spring deflection, c =damping coefficient and v =bumping velocity. When the piston reaches the constraint bumper, the force of bumper is active to stop the piston.

Friction

The seal frictions normally can be modelled by a constant leakage conductance. Refer to the seal material and deformation, it can be modelled more complex by the LuGre dynamic

model proposed in reference [19] that describes complex friction behaviour as the pre-sliding displacement, the stick-slip motion, the Stribeck effects and frictional lag.

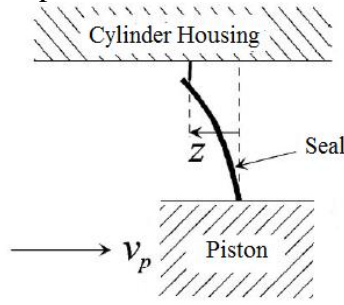


Fig. 3.5 Deflection of the cylinder seal

The seal deformation (Fig.3.5) is described by a state variable z computed by

$$\frac{dz}{dt} = v_p - \frac{|v_p|}{g(v_p)} z$$

where v_p is the piston velocity and $g(v_p)$ is given by

$$g(v_p) = \frac{1}{u_0} (F_C + (F_S - F_C) e^{-\left(\frac{v_p}{v_s}\right)^2})$$

F_C is the Coulomb friction force, F_S is the Stribeck force, v_s is the Stribeck velocity and u_0 is the seal stiffness.

The friction force generated from the bending of the seal that is given by

$$F_f = u_0 z + u_1 \frac{dz}{dt} + u_2 v_p$$

where u_1 is the damping factor to the seal motion and u_2 is the viscous friction coefficient. In modelling, the relation between velocity and friction force for steady-state motion is given by

$$F_f = u_0 g(v_p) \text{sign}(v_p) + u_2 v_p$$

The OOM approach for analysing the cylinder model promoted in project is based on the best practices that have proven successful in system development and, more specifically, the work done by Dragan H. Pršić. Therefore, the work will start from the behaviour description in Statechart.

3.3.1 Behaviour description in statechart

According to the behaviours of the HC, the HC model can be described in four orthogonal states. First of all, the state of the basic piston dynamics which is named the *Piston* state. The inlet pressure is converted into the force acting on the piston, the ideal cylinder model can be represented by

$$F = P_1 A_{piston} - P_2 A_{rod} \quad \text{and} \quad v_{piston} = \frac{Q_1}{A_{piston}} = \frac{Q_2}{A_{rod}}$$

Furthermore, the piston constraint is in regard to the piston motion which needs a limit value for the piston maximum displacement. This state can be a compound OR state with sub-states *StopLeft* and *StopRight* of the *Piston* state, while the basic piston dynamics is as sub-state *Normal* in the *Piston* state, when the piston working in the range of the stroke L , the *During* activities in the state *Normal* are always active. The main functionality to handle the hard cylinder stops is the restarting of the state variable v_{piston} when the piston reaches one of the limits (Piston position (x_p) < 0 or $x_p > \text{working stroke } (L)$). When such an event occurs, the cylinder ‘enters’ a stop state, and its force of the bumper can be set to an infinite to stop the movement of piston. To ‘leave’ the stop state the applied force

(Hydraulic force, load and friction) must invert its signal. *During* activities define the continuous state behaviour. In both constraint states, there are two sub-states inside of them, the simplest one is the constant or infinite force (*FBumperCons*), and the bumper force based on the maximum pressure in cylinder chambers (*FBumper*). This is the main purpose for adopting Statechart to handle models with different complexities. In BG, if the parameter *Fb* is changed to *0.1*, then the action *T5* is taken, more complex bumper force model is active. The other three orthogonal states are treated as additional states for cylinder models, which means that they could be modelled in a cylinder model or according to the purpose of designers. For example, if designers is focusing on the simulation result of the effect of seal friction, then they could only add the *Sealfriction* state into the model without considering the other two states model. In this way, the model can be in different levels appropriated to different type of simulation. In the project, “Ideal model” is modelled only with the description of state *Piston*. “Standard model” behaviours are all modelled in terms of constant factor except the end stopper since all three behaviour models are modelled as state *FBumper*, and there is no leakage assumed in “Standard model” which is negligible. In “Advanced model”, all the actions or events to activate the more complex behaviour will be active, and the leakage behaviour is also added in the model.

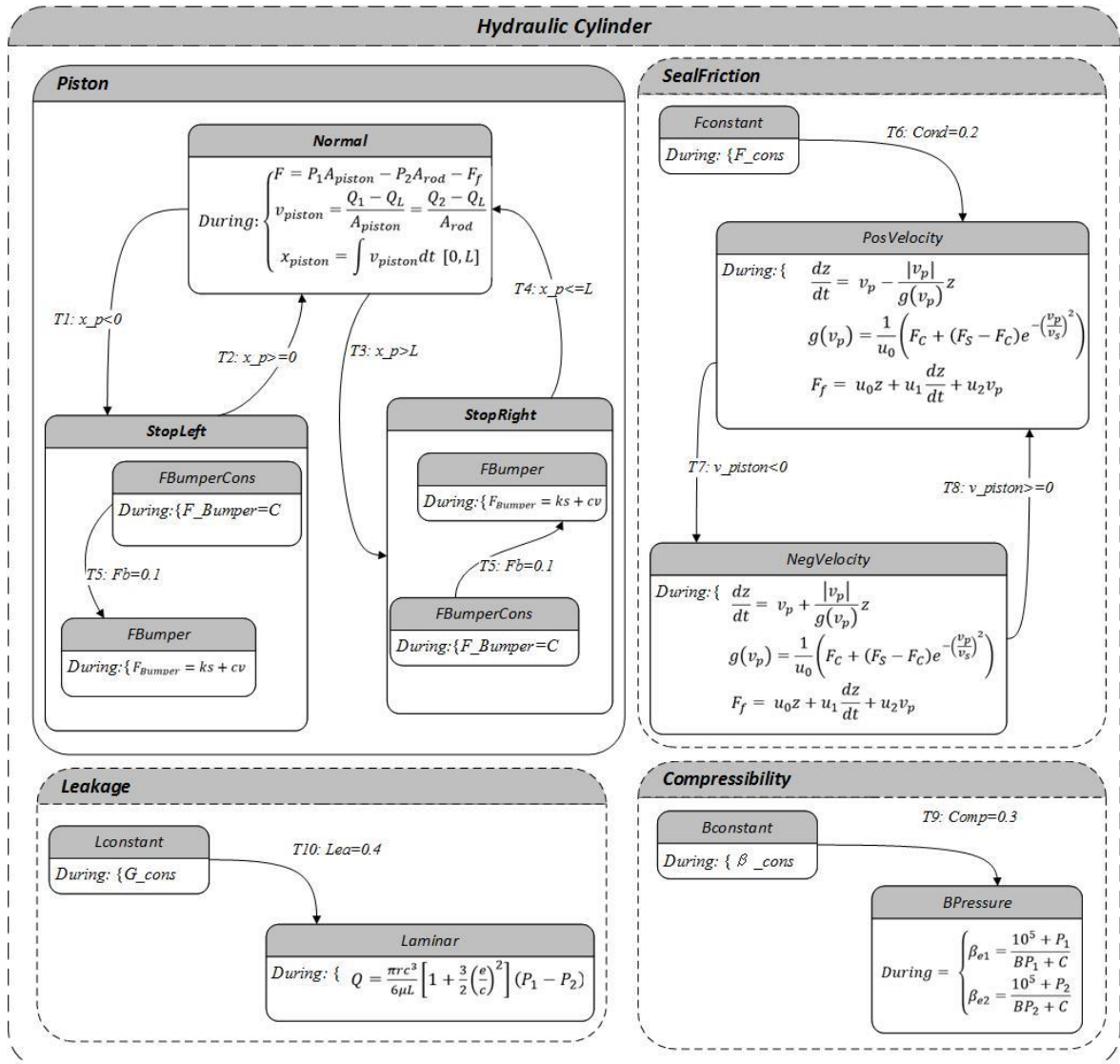


Fig. 3.6 The Statechart map of hydraulic cylinder

3.3.2 Model Classification

According to the description in Statechart, the classified behaviour models are modelled in this part.

Ideal Model

First of all, the “Ideal model” is developed in an energy lossless environment, it is even not considering the compressibility of fluid oil in cylinder chambers and the inlet and outlet of the cylinder. Literally, the fluid oil will interact directly on the piston of cylinder with a stroke constrain. The position motion is limited at the real cylinder.

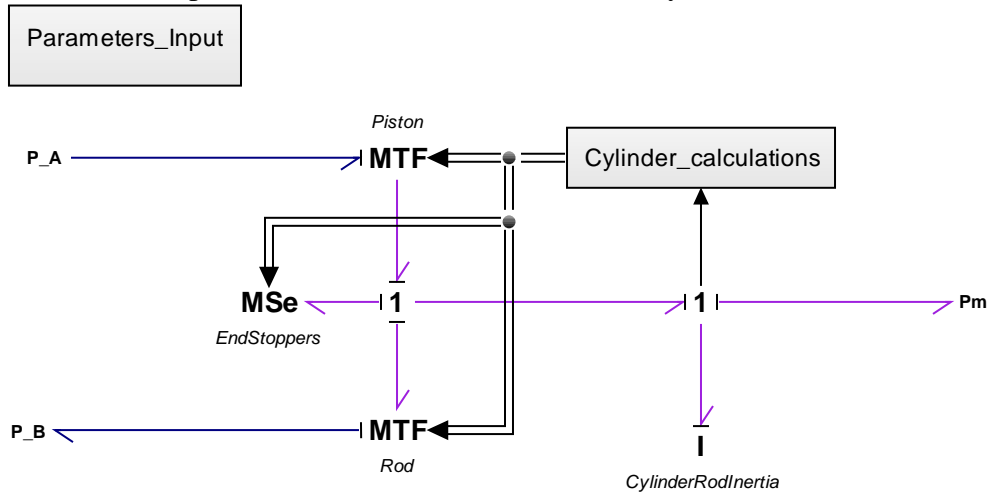


Fig. 3.7 The BG model of the hydraulic cylinder “Ideal model”

In Fig. 3.7 is shown the “Ideal model” as one behaviour model of the cylinder model in the library. The BG model is developed starting from establishing three interfaces to environment, they are the inlet P_A and the outlet P_B of the cylinder which interact with the hydraulic system through, the interface P_m is an energy port to connect the mechanical part. Next, a I -junction is added between P_A and P_B , since the pressure at the inlet and the outlet of the cylinder is different. Assuming positive flow direction from the inlet to the outlet, the MTF -element is inserted between P_A and I -junction as the chamber A of cylinder to convert the hydraulic energy into the mechanical energy, which another MTF -element is inserted between I -junction and P_B to describe that the movement of piston pushes the fluid oil out of the chamber B of the cylinder. Hence, the I -junction is connected to the mechanical part, and the stroke constrain of the piston is modelled as a force source MSe connecting to the I -junction. The inertia effect of the piston is modelled as an I -element between two MTF -elements. Note that the I -junction connecting to the I -element can be omitted for comfortably managing the BG connection. Last, the submodel *Cylinder_calculations* is applied to compute the formulas for two MTF -elements and the MSe -element that transports through signal source. And the submodel *Parameter_input* is for user to change the parameters according to real components.

Standard Model

In “Standard model”, the cylinder chambers are modelled with two ports. And the seal frictions are taken into consideration. However, the factors for computing the chamber model and friction effect are treated in a constant experienced number to make the model simple. On the other hand, in most cases the leakage does not affect the system behaviour considerably.

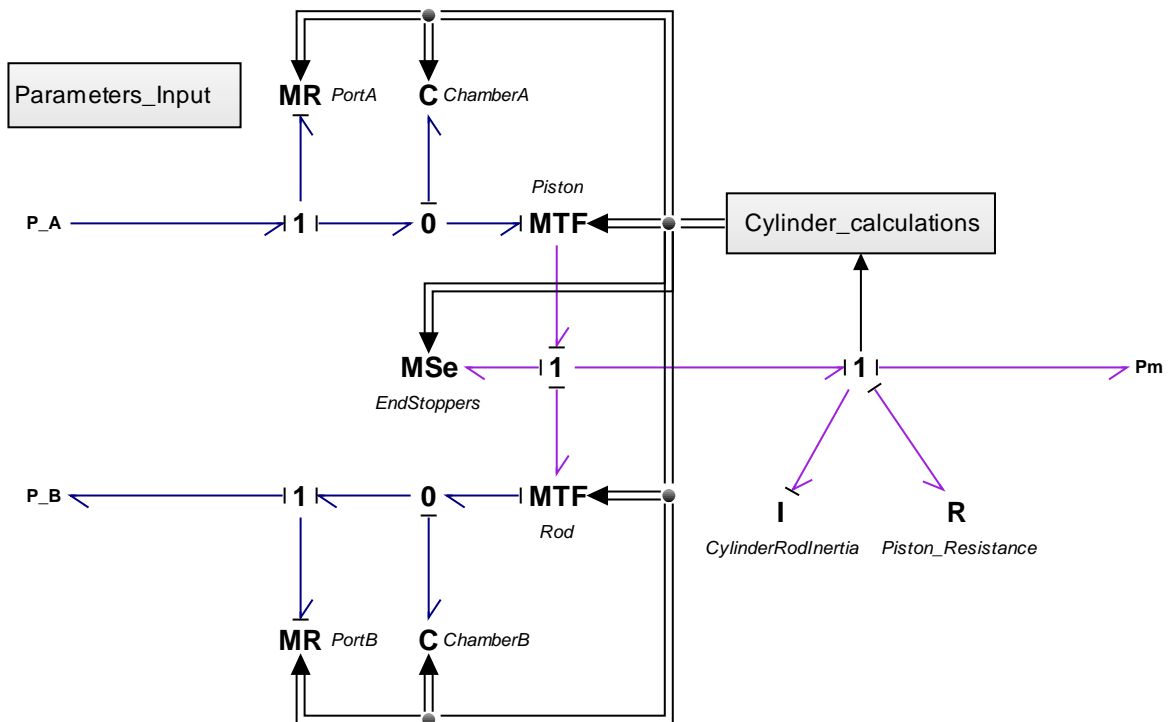


Fig. 3.8 The BG model of the hydraulic cylinder “Standard model”

Based on the “Ideal model”, two 0-junctions are added at the chamber A and the chamber B with distinctive pressures. And two 1-junctions are inserted between two ports of the cylinder and two 0-junctions. Then the compressibility effect of the liquid in the chambers assuming rigid walls is modelled by the C-element connecting to 1-junctions. The orifices of the inlet and the outlet of the cylinder is modelled by the R-element to restrict the flow areas.

Advanced model

The “Advanced model” explores deeper details of model behaviours, especially for the compressibility effect and the friction, according to the Statechart description, the effect bulk modulus is pressure dependent and the friction is material dependent. The leakages are modelled as a linear flow (R-element) based on the pressure differences.

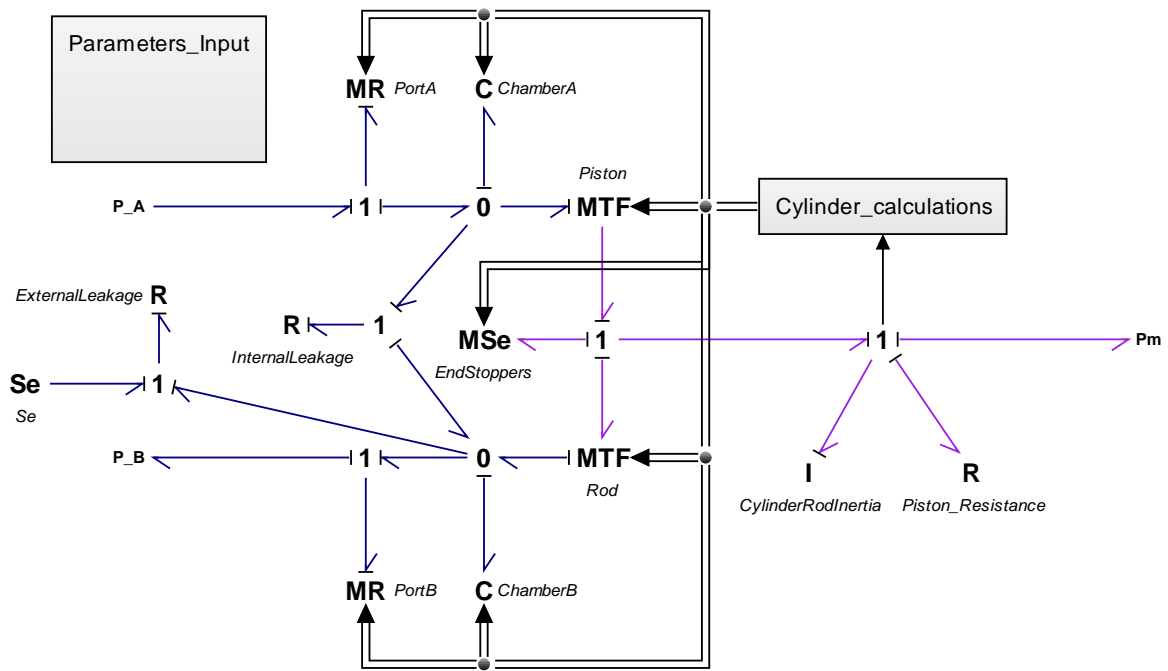


Fig. 3.9 The BG model of the hydraulic cylinder “Advanced model”

3.4. Directional Control Valve

The directional control valve (DCV) is used to control the start, stop and direct the flow of a pressure medium. There are many types of DCV according to the number of working ports and the number of spool positions, such as, 2/2 way DCV, 4/3 way DCV, etc. Afterwards, 4/3 way DCV will be analysed, then modelled and simulated based on the BG methodology.

4/3 way DCV has 4 working ports and 3 spool positions, A, B are the working ports, P is the pressure port which is also called pump port and T is the tank port or drain port, two working spool positions (E-extend and W-withdraw) and one neutral position. From a construction point of view, there are three types of DCV which are directional spool valve, directional poppet valve and rotary slide valve. There are advantages of a directional spool valve, such as,

- Simple construction
- Good pressure compensation, hence low actuating forces
- High switching power
- Low losses
- Variety of control functions

Which is why it is proven popular in many hydraulic systems. As a result, the analysis and modelling work is mainly based on the knowledge of directional spool valve.

According to the function of neutral position, there are seven types of 4/3 way DCV. It is selected for applying into a hydraulic system based on the function purpose when the spool is staying at neutral position.

Spool Type	Symbol	Characteristic
------------	--------	----------------

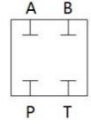
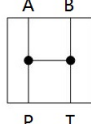
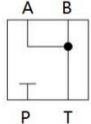
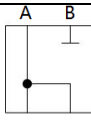
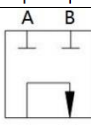
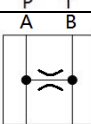
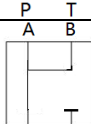
'O' type		0-0-0-0; Pump in load; Cylinder locked; It is used for parallel combination of several DCVs.
'B' type		1-1-1-1; Pump unloading; Cylinder piston can be moved by the force of payload.
'Y' type		0-1-1-1; Pump in load; Cylinder piston can be moved by the force of payload.
'K' type		1-1-0-1; Pump unloading; Cylinder locked.
'M' type		1-0-0-1; Pump unloading; Cylinder locked. It is used for series of several DCV.
'X' type		1-1-1-1; ports are opened in half way, pump is almost unloading with keeping slightly pressure.
'P' type		1-1-1-0; It is a differential circuit.

Table 3-1 The spool type of DCV.

In some situations, the DCV is not applied individually but corporately for the purpose of more accurate control. The valve orifices behave 'Open' and 'close' by modelling in terms of 'I' and 'O', which is shown in the table above as well as a serial of ports is 'P-A-B-T'. The DCV will be chosen and modelled by a combination of 'I's and 'O's to achieve the flow direction control.

Since 'Y' type 4/3 way DCV is most common applied, in this project it is used as an example for analyst.

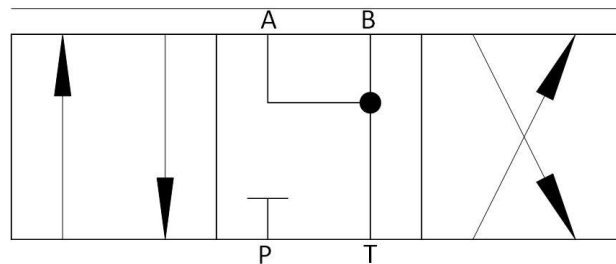


Fig. 3.10 'Y' type 4/3 way DCV

Directional spool valve controls oil flow by moving the spool in the valve housing. It is operated in various ways which are controlled electronically, mechanical & manual operation and fluid operation (hydraulic or pneumatic). The electrical control is the most commonly used due to the automatic processes required in industry.

The base function of DCV is to control the flow direction with the motion of spool in the valve housing. Its function is symbolically shown in the use case diagram (UCD).

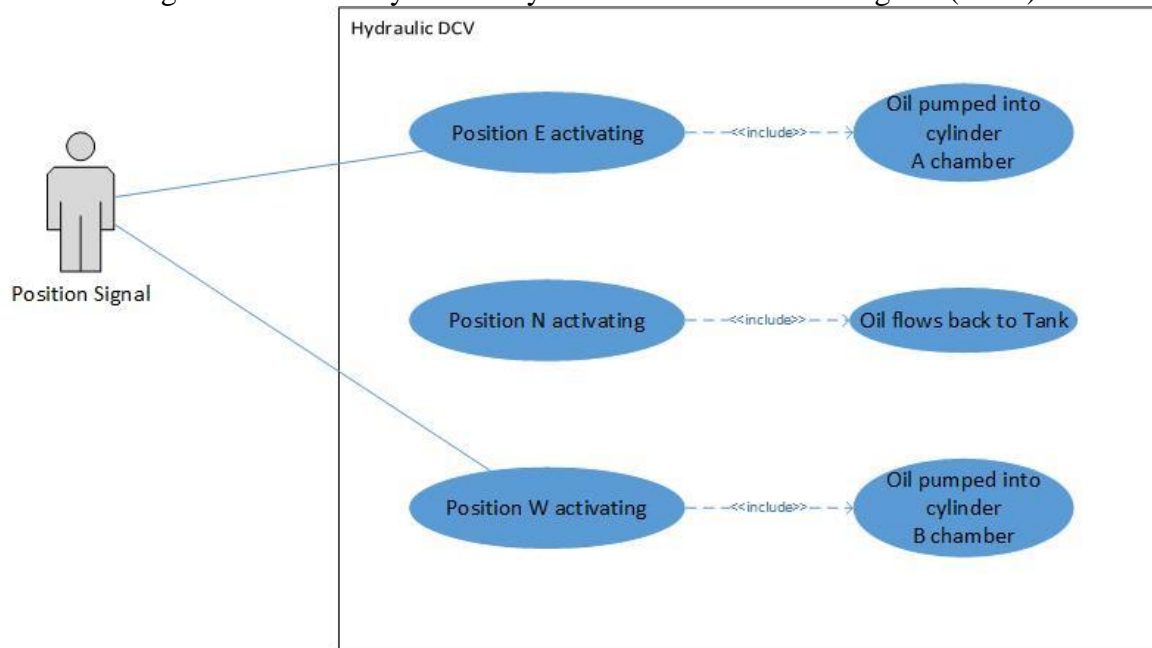


Fig. 3.11 Use case diagram of hydraulic DCV

In regards to the DCV, the environment plays a role interacting with the system by a directional change of the fluid flow and single input. As shown in Fig. 3.11 that base functionality includes positions activating controlled by position signal of actor which can be a position signal used to move the spool to the position E (Hoist function for an actuator) or position W (Slack function for an actuator), the position N (Stop function for an actuator) will be activated automatically depending on the active time of position E and position W.

By adopting a composition structure model, the interfaces can be constructed of by four metering orifices and two position signals (Fig. 3.12) interacting with environment. From BG aspect, four metering orifices are power ports and position signals are signal ports.

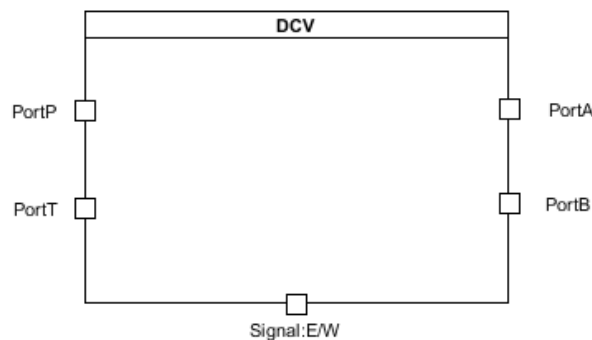


Fig. 3.12 Hydraulic DCV composition model

3.4.1. The first model layer

According to the spool position as the main functionality of DCV, the first model layer is starts from the neutral position (position N) which represents that the oil flow is blocked at *Port P* while the oil flow from working system goes through *Port A* and *Port B* then to *Port T* to the drain tank. In BG, the metering orifice is modelled as an *R*-element with energy loss due to viscous friction. Meanwhile, the metering orifice is a state affecting the behaviours of DCV. The modelling detail includes algebraic equations will be introduced

and discussed in Statecharts analysis. The figures following show how Position N function is performed.

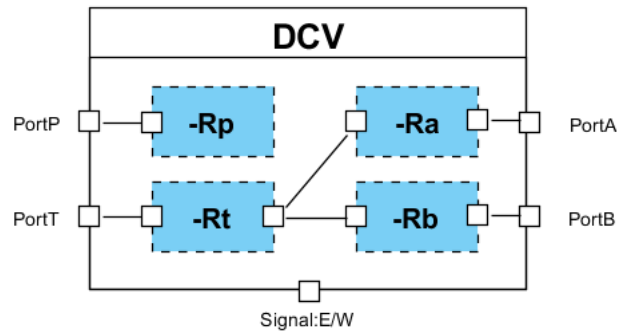


Fig. 3.13 DCV – Composition model of first layer

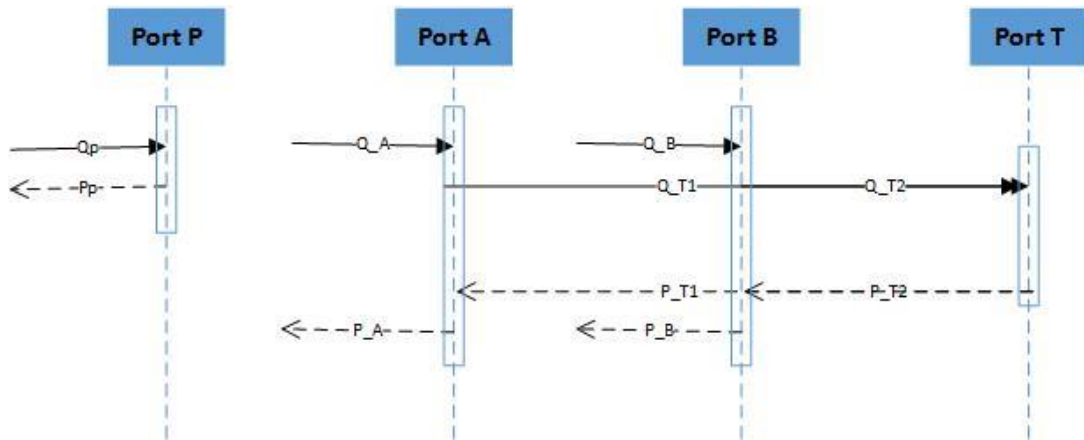


Fig. 3.14 DCV – Sequence diagram of first layer

In signal processing context, the order of Q_p , Q_A and Q_B messages are at the same level since the oil flow goes from system to DCV synchronously. However, the metering orifice of *Port P* is closed which results in the oil flow stopping at *Port P* with oil pressure return. On the other hand, messages from *Port A* and *Port B* are all at the same order since *Port A*, *Port B* and *Port T* are opened at the same time.

3.4.2. The second model layer

The working position E and W are activated by position signal input to move the spool for each ports opening. When position signal input with position E, the oil flow will go through from *port P* to *port A*. When it is the signal of position W, the oil flow from *port A* will be change to *port B*. How the oil flow respectively runs through DCV in two working positions is showing in Fig. 3.15

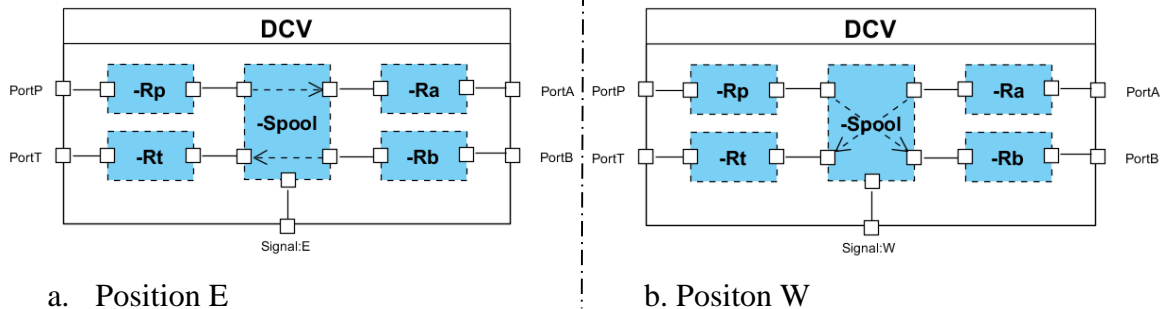
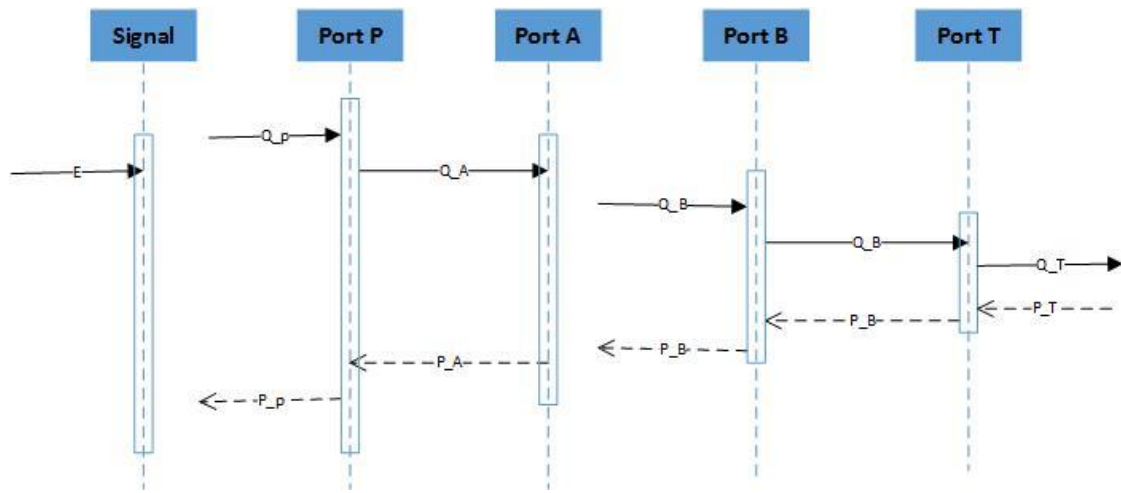
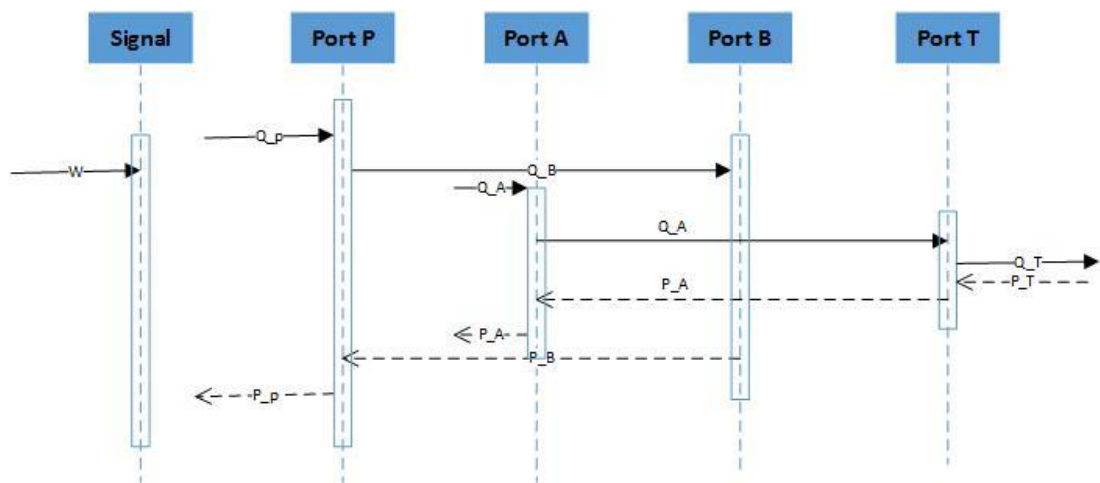


Fig. 3.15 DCV – composition model of second layer



a. Position E



b. Position W

Fig. 3.16 DCV – sequence diagrams of second layer

The second layer structure of DCV presents performance for the realization of different use cases in two working positions respectively. Fig. 3.16 the sequence diagrams describe that how fluid oil flow through DCV when the signal sends to DCV.

There should be a third layer for activities happening description, such as frictions and fluid compressibility during the process between open and close. Nevertheless, the frictions is negligible which is normally omitted. The fluid oil compressibility could be modelled as a control volume since the flow is not a large volume in front of valves. As a result, it is not necessary to develop one more layer for DCV, the detail behaviours will be specified in the Statecharts analyst.

From the result of OOM approach analysis, the main working principle of DCV is well understood. Furthermore, the model is able to be refined in each layer. For example, at the first layer, the model can be refined by choosing different function of spool position as shown in table 3-1. In second layer, it can not only add the friction and compressibility activities, but the internal leakage could be added as a modification model for an “advanced model”. Consequently, the OOM approach is used for the static and dynamic analysis to give a visual modelling process in this project. The behaviour classification is happening in Statechart research.

3.4.3. Bond Graph model

At this point, all objects and activities of DCV are presented and the base DCV BG model can be modelled as shown in Fig. 3.17. Furthermore, it is different from the composition diagram, BG model only has one *R*-element modelled as the port for two ports interlinked instead of two *R*-elements standing for two ports respectively in composition diagram since the volume in DCV chamber is small enough to regardless the volume property of compressibility and other factors. In front of ports, the control volume is modelled by *C*-element to act as the feature of the fluid compressibility. The position signal *Uspool* could be a pulse, linear or non-linear function to control the spool movement.

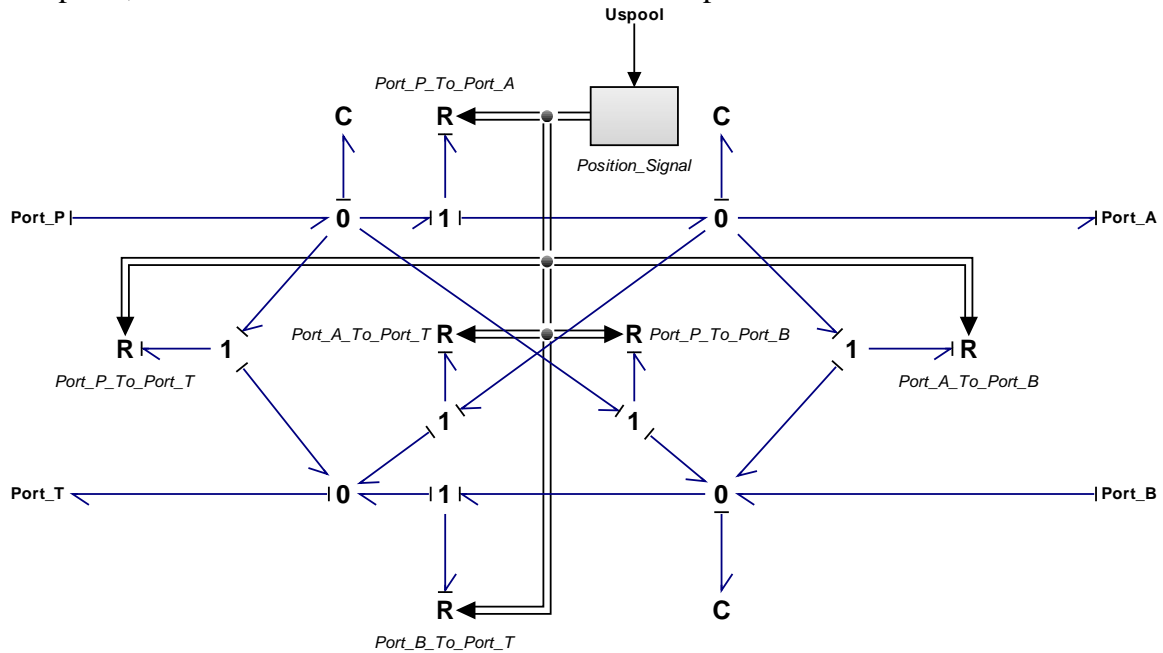


Fig. 3.17 The base BG model of DCV

3.4.4. Behaviour description in Statechart

The behaviour and quality of DCV is determined by the following aspects:

- Dynamic power limit
- Static power limit
- Resistance to flow
- Leakage
- Switching time

According to these aspects, the DCV model can be split into several blocks as states with serial connections: blocks of dynamic part, static part, control volume or flow model and pressure compensation.

Spool motion

Dynamic performance limit includes the product of flow and operating pressure as main factors. Power limits could be modelled as a function of the control spring, the solenoid or control pressure. For all of these functions, the actuating force for moving spool must be able to overcome the power limit, meanwhile, the power limit must be capable of returning the spool against the axial force to its initial position. The axial forces are created by some features as following,

- Mass force F_m
- Viscosity force
- Flow force
- Resistance force

Depending on these characters, spool motion is modelled mainly by the basis of the amplitude and phase frequency response. The most important part of dynamic is phase frequency response which is a function of spool position in regard to the input signal. The function of the spool position has a linear relationship with flow area for servo valves, while proportional valves have a considerable overlap and nonlinear characteristics between position and flow area and a considerable overlap caused by a limit of the spool velocity, hysteresis and friction.

Advanced control strategies can be used to obtain good spool dynamic characteristics over the whole valve operating conditions in order to overcome its highly non-linear operation. This is achieved with the use of a closed loop spool position control. The controller minimises the non-linear perturbation effects in the spool movement. In this project, the control system will not be discussed, hence the signal for control the movement of the spool is programmed in a simple proportional function. In a real case, the control system depends on the actual valve design.

Volumetric flowrate model

The volumetric flowrate which goes through DCV is controlled with a non-linear function of the normalised spool position u_{spool} and the pressure difference ΔP . The oil flow through DCV is combined with leakage and compressible flows, which controls the chamber pressure. The volumetric flowrate q through a restriction section from port to port is always modelled as described in Chapter 3.2 Restriction.

The flowrate is a function of pressure drop and orifice area, which the orifice area is function of spool position u_{spool} . It can be modelled only from its function of *Open* and *Close*, first order of linear function, or complex second order of nonlinear function. Depending on the model purpose, it will be modelled varyingly. In addition, the pressure drop is caused by the viscous friction which is described as a function of Reynolds number to define if it is laminar flow or turbulent flow.

Leakage

A leakage can occur when the spool is moving. However it does not affect the system behaviour considerably and to some extents, it helps seal lubrication and reduces the friction loss. As the leakage flow is always small, it can be modelled as laminar flow demonstrated in Chapter 3.2 Restriction.

$$q = \frac{\pi r c^3}{6\mu L} \left[1 + \frac{3}{2} \left(\frac{e}{c} \right)^2 \right] (P_1 - P_2)$$

Pressure Compensator

In most common hydraulic system, the DCV model always includes the load sensing circuit and the pressure compensator. It maintains a constant pressure drop across metering orifice of the DCV at all times, which develops a controlled flow independent to the load pressure and proportional to the spool position.

However, the problem for modelling is that how to define or establish a model of the compensator when the pressure saturation will occur. In this situation, the pressure drop across the metering orifice P_0 is known as compensated pressure P_{comp} can be described as

$$P_{comp} = \begin{cases} P_{LS} + P_0 & \text{for } P_0 \leq P_S - P_{LS} \\ P_S - P_{LS} & \text{for } 0 < P_S - P_{LS} < P_0 \\ P_{LS} & \text{for } P_S - P_{LS} \leq 0 \end{cases}$$

The first case demonstrates that the general operating condition to maintain the nominal pressure drops. The second situation is that when the load pressure is too close to supply pressure to maintain the nominal pressure drops. The third case is to prevent negative flow when the load pressure exceeds the supply pressure.

The load sensing circuit is a pilot function used for directing the load pressure to the pressure compensator which is modelled as a piecewise function

$$P_{LS} = \begin{cases} P_A & \text{for } u_{spool} < 0 \\ P_B & \text{for } 0 < u_{spool} \\ 0 & \text{otherwise} \end{cases}$$

Behaviour classification

According to the OOM approach analysis, the basic structure and dynamic function of DCV is visually shown that there are five interfaces, spool and valve house as basic structure of DCV and three spool positions give the dynamic function to control the direction of fluid flow.

By behaviour description of DCV in Statechart, DCV is the orthogonal product of spool dynamic, leakage, and pressure compensator, which correspond to the high-level description of the AND states. These states are able to be refined through state decomposition, meanwhile, more states like cavitation can be also added into statechart for a behaviour model to do detail simulation.

First, the state *spool dynamic* is modelled proficiently in a way with *Open* and *close* of orifice area (0 or 1), which can be compound OR state with substates *Position E*, *Neutral position* and *Position W*. As shown in Fig.3.18, when task *T1* takes place ($u_{spool} < 0$), the state *Position E* is decomposed to compute the behaviour that spool moves to *position E* while fluid flows from *port P* to *port A* and from *port B* to *port T* with fully opened orifice area, when task *T2* occurs ($u_{spool} > 0$), the state *Position W* is decomposed to compute the behaviour while spool moves to *position W* which there is a flow from *port P* to *port B* and *port A* to *port T* with fully opened orifice area. For the entry state of *Neutral position*, it is also defined as task *T3* occurs. The behaviour is described by defining the functions shown in table 3-1. Additionally, states *Position E* and *Position W* can be un-clustered from the state *Spool Dynamic* for hierarchical description which has the advantage of keeping the neighbourhood small yet the parts of interest large. Meanwhile, it is beneficial for model classification. However, this is not very exact for un-clustering since in un-clustering state which still has “0” activity meaning that the spool position is back to neutral position. For advantages, un-clustering is the way to classify the orifice opening behaviour as shown in Fig. 3.19, state *[0,1]* describes the behaviour of “Ideal model”. When the tasks *T10* or *T11* take place, the orifice section area will be a linear or non-linear opening condition, which defines the complexity for “Standard model” and “Advanced model”. Here it should be mentioned that the spool position is a function of control method, therefore, the function will be computed freely by designers and the function used in this project, for the sake of simplicity, is a linear function for both “Standard model” and “Advanced model”. The solid point in statechart is a function of default entrance, which will be initial value in BG models.

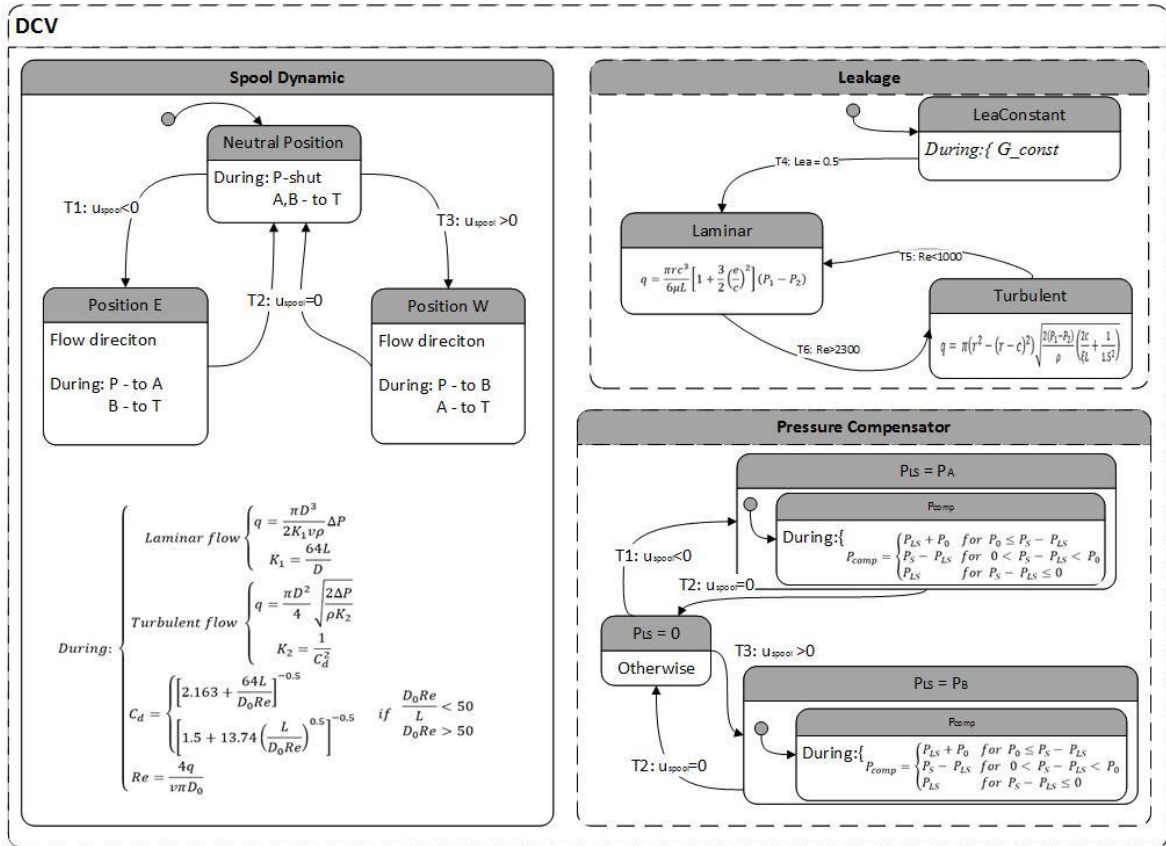


Fig. 3.18 The Statechart analysis of DCV

The purpose of the “Standard model” is for a simulation that needs to show more specific detail, such as the orifice metering area, internal leakage and a pressure compensated function which is always attached to DCV. Apparently, it is more complex than the “Ideal model”, however, some coefficient in stats are normally constant experienced factors from similar experiments since the “standard model” is only for checking the rationality of physical performance in the simulation. When adding stats into model, state *Leakage* enters a constant leakage coefficient. Finally, state *pressure compensator* is a dependent state of *spool dynamic*, when *T1* occurs, state $P_{LS} = P_A$ and state *Position E* will be active simultaneously which means *port A* will be a power port to move the cylinder. Task *T3* has the same theoretical performance as well as brings spool to neutral position and the compensator stops working.

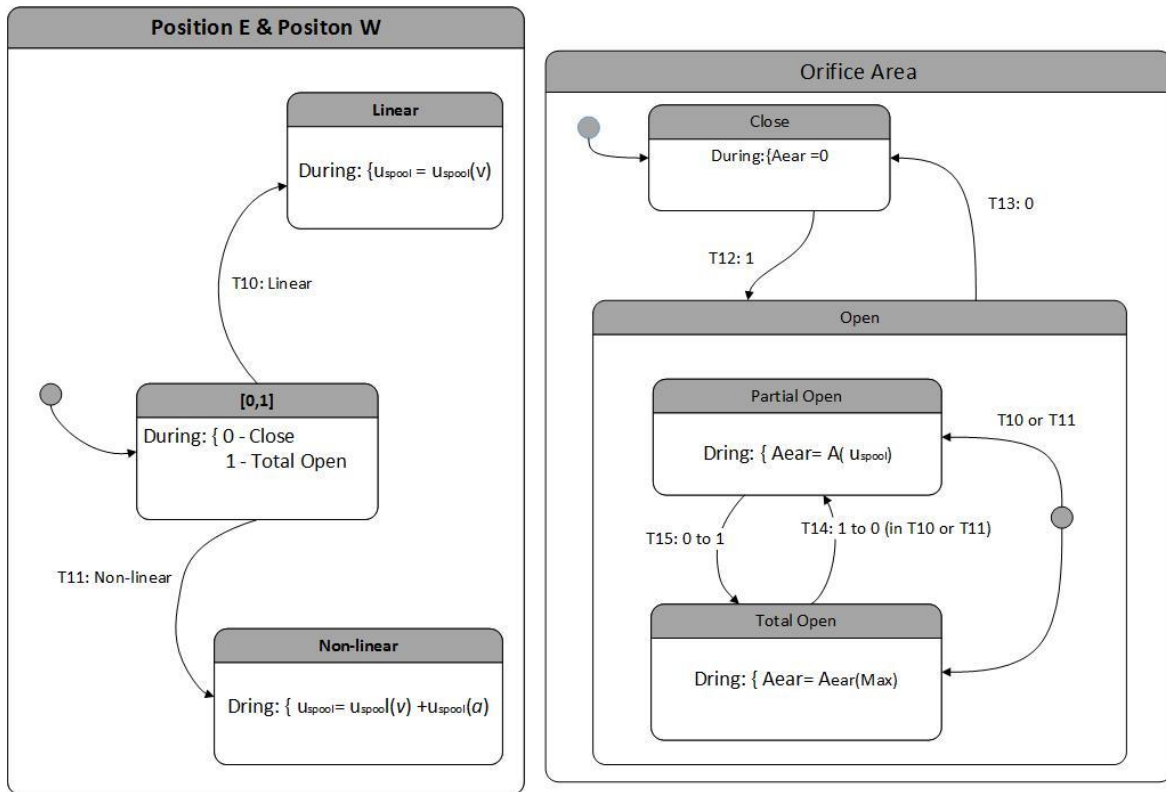


Fig. 3.19 The un-clustering of spool motion

Based on the “Standard model”, other states have to be considered in an “Advanced model” for the use of modification of in a system design. Since each state can give the suggestion for modification. From the fluid aspect, although “Ideal model” and “Standard model” are defaulted that the fluid oil is turbulent flow running through DCV since most time the fluid oil is turbulent flow running in the hydraulic system, “Advanced model” is modelled both laminar flow and turbulent flow determined by Reynold’s number. On the other hand, the event $T4$ is happening, the leakage will be modelled in a detail by a discussion of flow type.

The BG model structure for three behaviour models are presented same in 20sim as show in Fig. 3.17, the formulas for program is varying according to the behaviour hierarchy in BG elements.

3.5. Counterbalance Valve

The counterbalance valve (CBV) is modulating a valve which allows free flow into the actuator and then block the reverse flow until it feels a pilot pressure inversely proportional to the load in the pilot line. The model of the CBV is not an independent model which is comprising two components including a check valve for the free flow direction and a pilot assisted relief valve to control the flow in the reverse direction (Fig. 3.20).

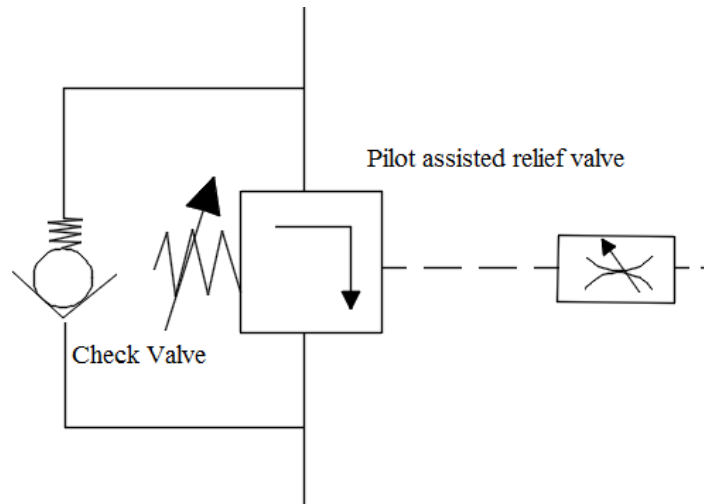


Fig. 3.20 The symbol of CBV

3.5.1. Description in layers

CBV is able to lock loads in a leak free mode and it is well suited for many clamping applications or to prevent negative loads from falling down in case of hoses failure. For the load lowering process, CBV improves control motion in most systems since it compels the DCV to always meter positive pressure, also under overrunning load conditions. CBV is beneficial to the application of the paired actuator: pilot pressure will open first the valve of the most heavily loaded actuator; this will cause load that transfers to another actuator and related valves, still closed, will require less pilot pressure for opening.

Since the complexity of the CBV's behaviour is not high with only "Open" and "Close" behaviours, there is no need to describe in many layers, therefore, the basic function of CBV is analysed by diagrams of UML to show its structure and dynamic model at the beginning, then statechart is used to give the description of behaviours classification.

In Fig. 3.21, it is visually showing that the main functions of CBV itself is "Open" and "Close". The actors are "Energy Mechanism" and "Energy Hydraulics" which interact with objects. Mechanical energy (*spring*) has a "close" association with CBV for closing the valves ($F_{mech} > P_{Hydr}$). Hydraulic energy includes "Energy Hydraulics Upstream flow" and "Energy Hydraulics Lowering flow". The former one is only for opening the check valve to let the fluid oil flow forward into the actuator ($P_{upstream} > F_{mech}$), the "Energy Hydraulics Lowering flow" is for lowering flow delivered upstream by DCV, which the pressure is a pilot for CBV to open relief valve ($P_{lowering} > F_{mech}$) to let the fluid oil flow out of the actuator.

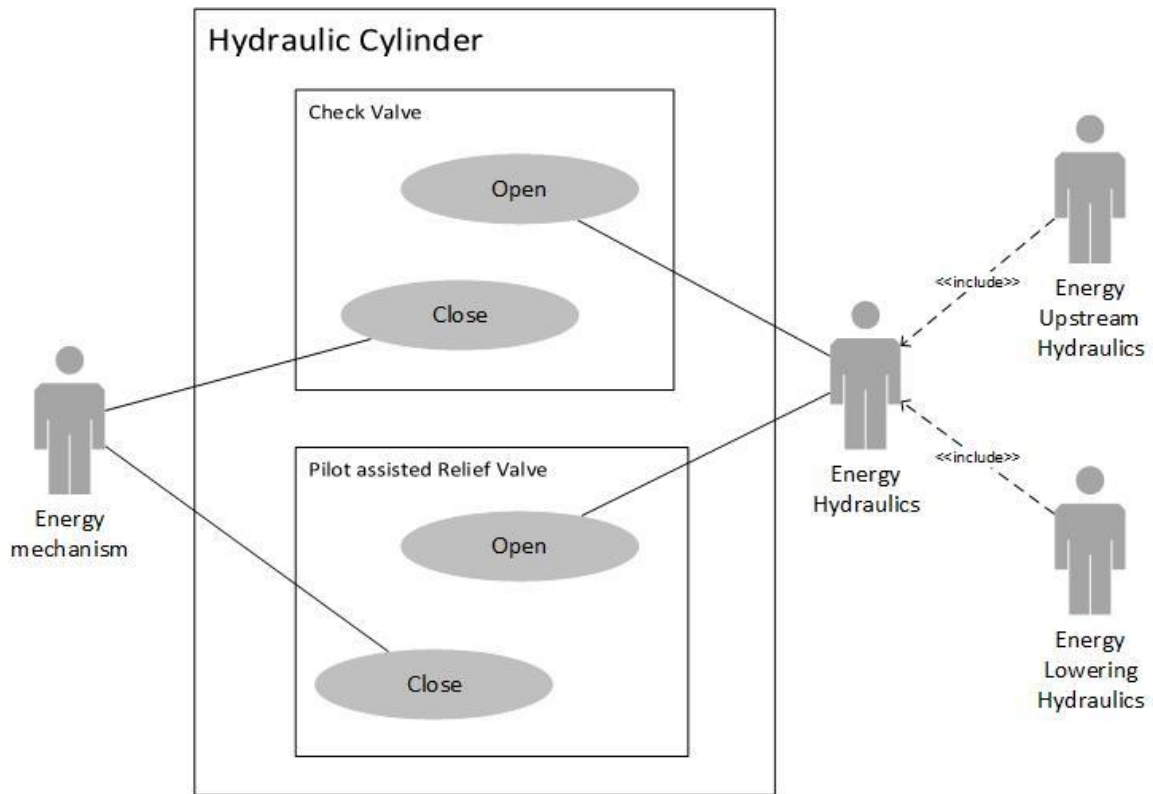


Fig. 3.21 Use case diagram of CBV

As a summary for the functions of CBV, they are,

- Free upstream flow through the check valve for load lifting.
- Locking of reverse downstream flow when the directional valve is not operated, or the pump is stopped.
- Load lowering metered by the piston of the relief valve opened and controlled by the pilot pressure of the “lowering” flow delivered upstream by the directional control valve, also when the load tends to overrun.

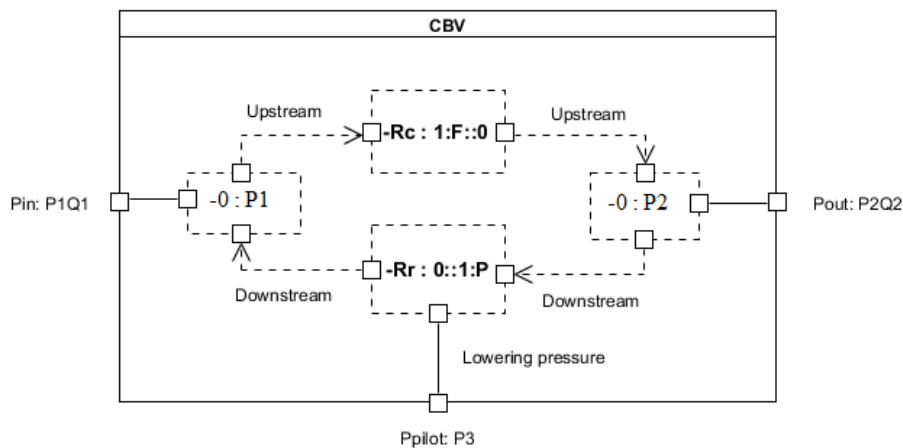


Fig. 3.22 Composition diagram of CBV

From the static aspect understanding, composition diagram shows the structure of CBV consistent with a check valve (-Rc) and a pilot assisted relief valve (-Rr) with three

interfaces to interact with the environment, which two ports are for the hydraulic energy transportation, one is pressure pilot assisted signal port (Fig.3.22).

Transporting energy from port Pin to port $Pout$ can only achieve through the check valve – Rc in the condition of $P1 > P2$ to open, conversely, from port $Pout$ to Pin is through relief valve – Rr depending on the pilot pressure $P3$ to open. The dynamic diagram (Fig. 3.23) shows clearer communications between objects with message flow (fluid flow rate).

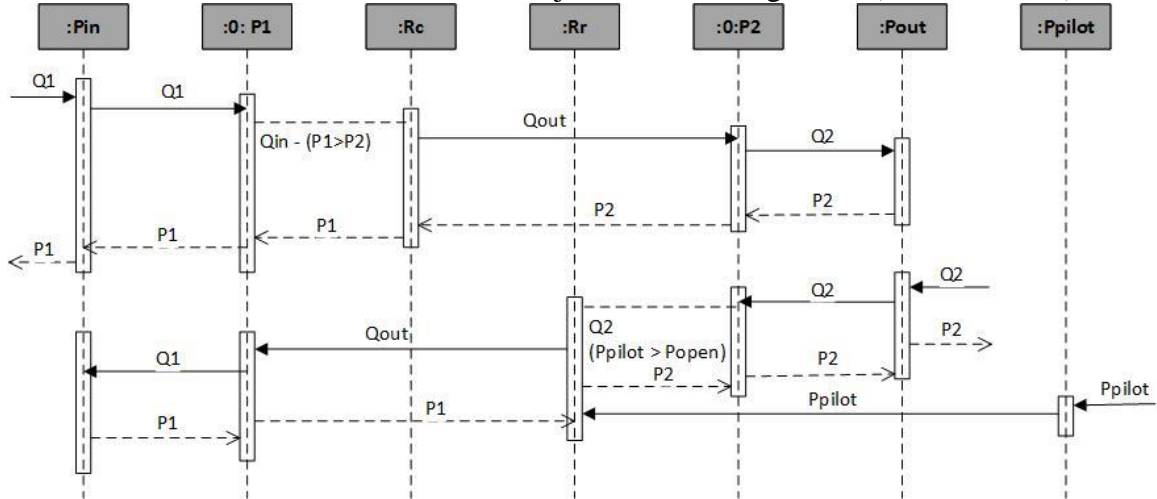


Fig. 3.23 Sequence diagram of CBV

In sequence diagram, it is actually two dynamic models which are upstream flow through the check valve and downstream flow through the relief valve respectively. There is a conditional flow in each sequence model: 1) in upstream flow, if $P1 > P2$, then $Q1$ flows through check valve; 2) in downstream flow, if $Ppilot > Popen$, then $Q2$ flows through relief valve. $Popen$ is a value which could be a set point or a mechanical formula of relief valve. 0-junction is to represent each distinctive pressure ($P1$, $P2$) in system.

One notation has to be mentioned that the mechanical energy as a preload pressure for closing the valve should be iterately modelled as a constraint to stop the valve piston when the valve is fully opened, and to close the valve in a leak free model when the pressure is lower than load pressure. Ideally, the position valve piston had no physical effect on hydraulic flow with only a function to the opening section area of valve. Therefore, mechanical constrain can be modelled as a function of opening area according to the pressure within the “Ideal model” and the “Standard model”. The “Advanced model” will be modelled with mechanical constrain as a spring or a linear function according to the pressure.

3.5.2. Bond Graph Model

According to the both static and dynamic analysis, the basic BG model of CBV can be set up. The structure of BG model is different from composition diagram, the check valve and the pilot relief valve are modelled in one MR -element for sake of simple by using *if...then...else* function.

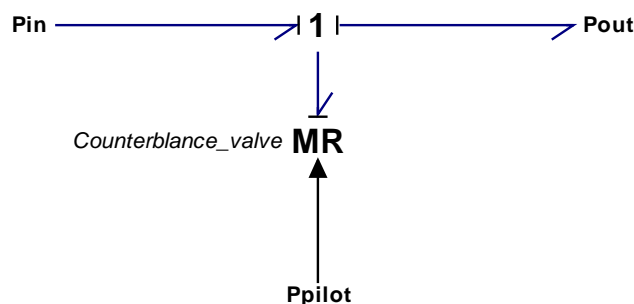


Fig. 3.24 Bond graph model of CBV

3.5.3. Behaviour description in statechart

Both check valve and relief valve are flow control valves with the function to measure flow rate and a restriction to shut flow from the opposite direction. Since the CBV is composed of a check valve and a relief valve, the statechart can start by two sub-objects of CBV with “Open” and “Close” behaviours. The orifice area is modelled varyingly according to the valve types. A leakage is taken into consideration when the valve is closed. Besides, sometime it is necessary to build a detailed model including the inertia of the piston, the oil volumes under pressure and the restrictions, especially for modification process of a valve design.

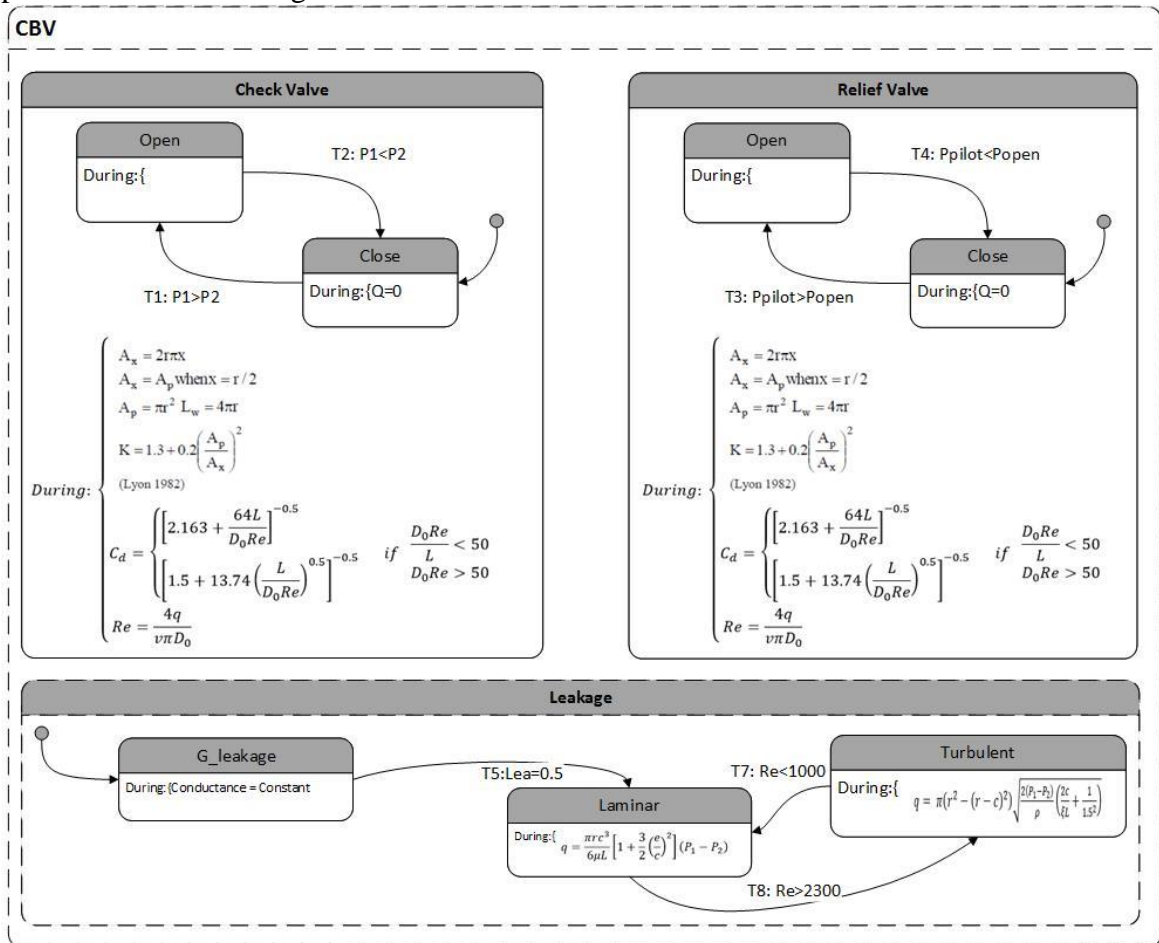


Fig. 3.25 Statecharts analysis of CBV

The dynamic formulas in Fig.3.25 are introduced in Chapter 3.1 Hydraulic fluid. The high level description of the AND state CBV with four orthogonal states *Check Valve*, *Relief Valve*, and *Leakage*.

In the state *Check Valve* and *Relief Valve*, it is a compound OR state with sub-states *Open* and *Close*. In *During* activity, the continuous state behaviour of *Open* state allows fluid flow through the valve and no flow in *close* state. Moreover, the state *Open* and *Close* can be decomposed to even lower level description in dynamic aspect of valve piston as shown in Fig. 3.26

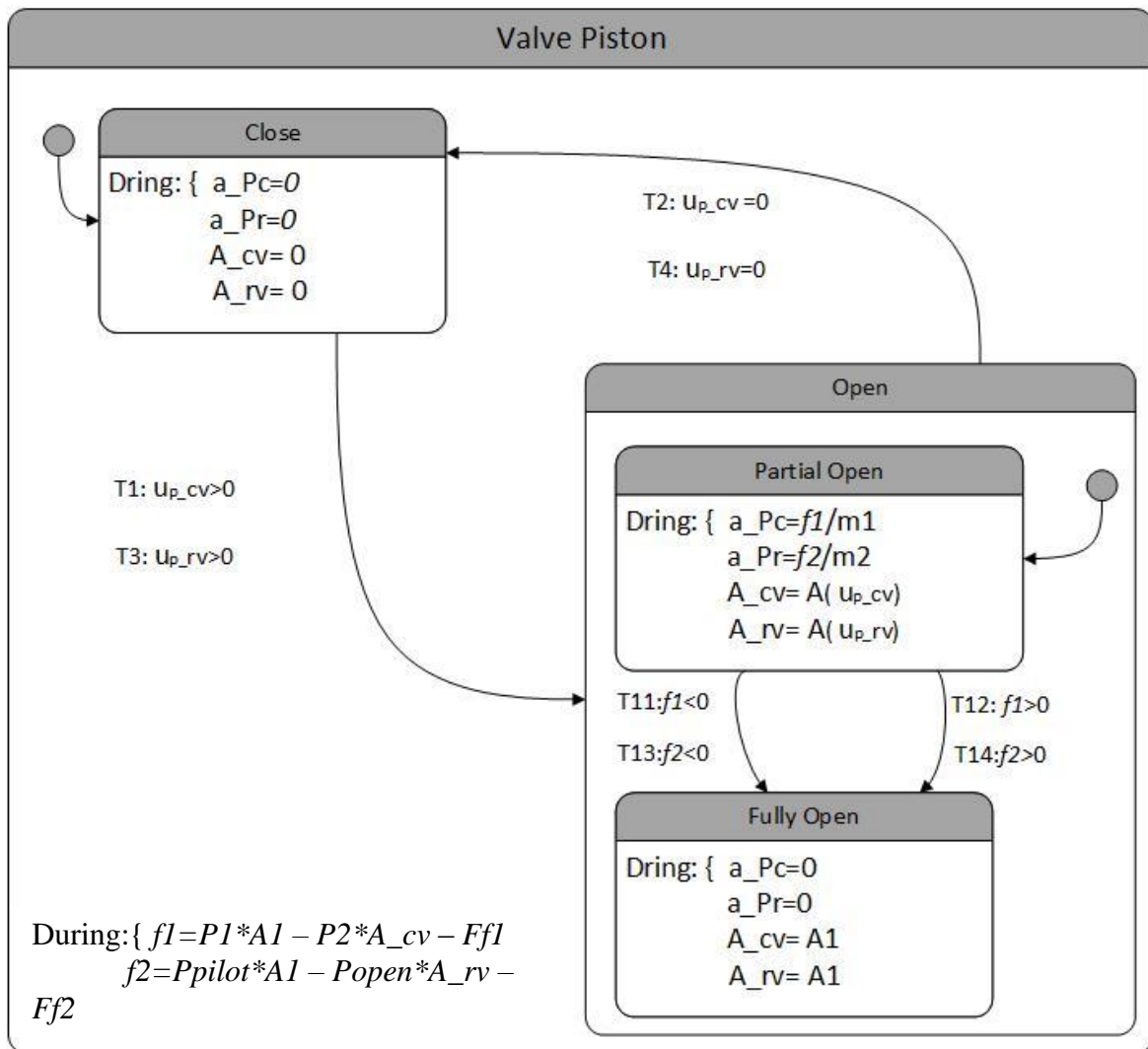


Fig. 3.26 The low level valve piston statechart for Open and Close

By introducing the mechanical dynamic into valve, the *Open* behaviour can be unclustered to describe as *Partial Open* and *Fully Open* in *Valve Piston*. The mechanical force $f1$ and $f2$ are the preload of the check valve and relief valve, for example, if the pressure $P2$ is zero, the hydraulic force $P1A1$ has to be larger than $f1$ to open the valve. During the valve opening process, the mechanical force is negative force to damp the piston until it is stopped as a restriction (fully open). When the valve is in closing process, the mechanical force is positive to move the piston to close the valve. While the section area is a function of piston position (u_p).

The leakage behaviour model can be treated in different complexities. Most commonly, the leakage can be dealt with constant conductance. While if the task $T5$ happens, then it turns to much detailed by the judgement of critical ΔP .

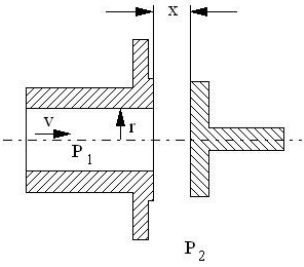
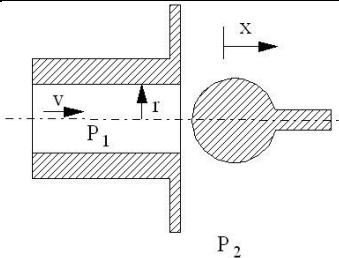
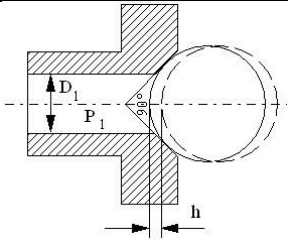
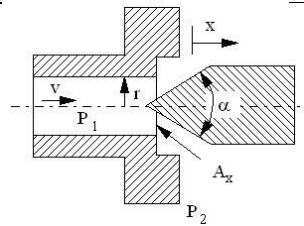
3.5.4. Model classification

The basic function model is well enough for an "Ideal model" with the design requirement that only "Open" and "Close" behaviours of check valve, while relief valve is depending on the relationship between pilot pressure and open pressure (set pressure by designer) to open and close. The BG model is same as basic model in Fig. 3.24.

"Standard model" adds the behaviour of leakage with constant conductance and the fluid compressibility (*control volume*) with constant bulk modulus, moreover, the pilot of Counterbalance valve is treated by *LowPassFilter* to limit the high frequencies passing to

give the relief valve a smooth opening process which is a linear function according to the position of valve piston. The “Standard model” is also no need to judge the flow type if it is laminar or turbulent. The structure of the BG model is same as “Ideal model” in Fig. 3.24.

Most detailed “Advance model” needs to be considered all behaviour states for modification process. The volume flow is defined by the judgement of Reynold’s number if it is laminar flow or turbulent flow. The section area also needs to program according to the detailed design, some examples are given in Table 3.2. The disk valve is modelled in library. While the mechanical model is computed by a linear function instead of a spring. Consequently, if some designers want to check the effect on system by typical factor, they could add them into the “Advanced model”.

 <p>Disk Valve</p>	$A_x = 2rx$ $A_x = A_p \text{ when } x = r/2$ $A_p = \pi r^2 \quad L_w = 4\pi r$ $K = 1.3 + 0.2 \left(\frac{A_p}{A_x} \right)^2$ <p>(Lyon 1982)</p>
 <p>$x = 0 \Rightarrow$ valve closed Simple Ball Valve</p>	$A_x \approx 1.5\pi r x$ $A_x \text{ is valid when } r \approx 1.3r$ $A_p = \pi r^2$ $L_w \approx 4\pi r$ $K = 0.5 + 0.15 \left(\frac{A_p}{A_x} \right)^2$ <p>(Lyon 1982)</p>
	$K = 2.65 - 0.8 \left(\frac{h}{D_1} \right) + 0.14 \left(\frac{h}{D_1} \right)^2$ $0.1 < h/D_1 < 0.25$ <p>(Chaimowitsch 1961)</p>
 <p>$x = 0 \Rightarrow$ valve closed Needle Valve</p>	$A_x = \pi \left(2rx \tan \frac{\alpha}{2} - x^2 \tan^2 \frac{\alpha}{2} \right), \quad A_x = 0 \text{ when } x = 0$ $L_w = 2\pi \left(2r - x \tan \frac{\alpha}{2} \right), \quad K = 0.5 + 0.15 \left(\frac{A_p}{A_x} \right)^2$ <p>K is valid from $A_x = 10$ to $A_x = 100$ % of entry area, A_p. (Lyon 1982)</p>

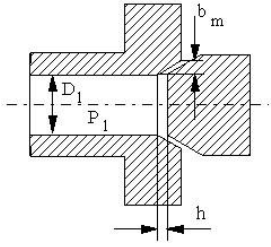
 <p>Poppet valve</p>	$K = 2.65 - 0.8 \left(\frac{h}{D_1} \right) + 0.14 \left(\frac{h}{D_1} \right)^2$ $0.1 < h/D_1 < 0.25$ $b_m / D_1 = 0.1$ <p>(Chaimowitsch 1961) see also (Bergada and Watton 2004)</p>
---	--

Table 3-2 Models of different valve types

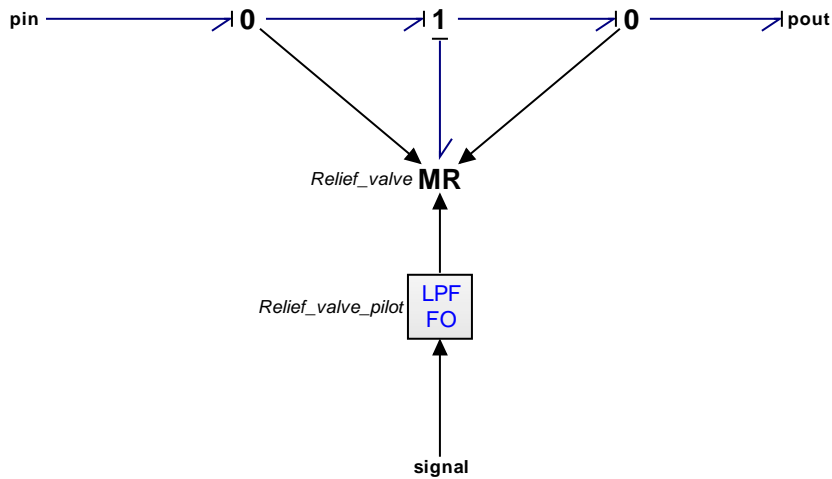


Fig. 3.27 The BG model of the “Advanced model”

In the BG model, the function for valve opening area is according to the pressure in front of the valve which are pressure signals from 0-junctions.

3.6.Pump

The pump is used to convert mechanical energy into hydraulic energy in a hydraulic power unit. Most hydraulic systems use a positive displacement pump as a power unit. It is important to understand that the pump does not generate pressure but only produce fluid flow. The pressure is determined by the resistance of the oil passing on its way through system. Therefore, pumps should be modelled as flow sources.

Axial piston pump is analysed and modelled for the library in this project, the rotating displacement pistons are supported by a swash plate. The angle of the swash plate determines the piston stroke. By defining the angle of swash plate, the pump can be modelled as a pressure compensated pump or fixed axial pump. The symbol of a pressure compensated pump is shown in Fig. 3.28.

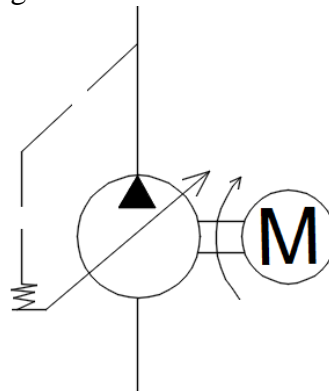


Fig. 3.28 The pressure compensated pump symbol

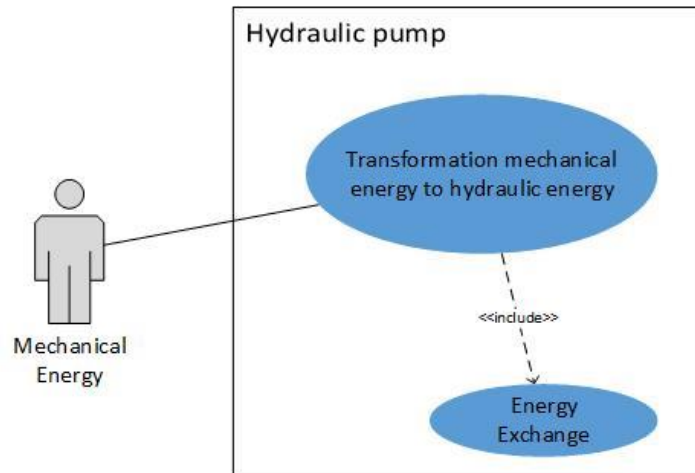


Fig. 3.29 Use case diagram of hydraulic pump

The use case diagram of the pump shows that its function is to transform the mechanical energy to hydraulic energy. It is reflected in the change of pressure, since the flow at the inlet of pump is equal to the flow at outlet of pump ideally. The composite model shows the interactions with environment through three interfaces with two hydraulic power port and one mechanical power port (Fig. 3.30).

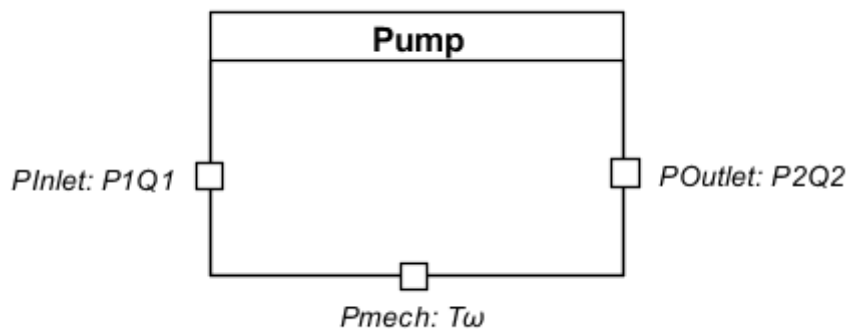


Fig. 3.30 Pump composition model

3.6.1. The first model layer

The basic function of a hydraulic pump is represented as a transformation in this layer. The mechanical energy entering from the mechanical port ($P_{mech}: \tau\omega$) is transformed by transformer element (TF) into hydraulic energy ($P_{outlet}: P_2Q_2$) at the outlet of pump. For an ideal axial piston pump, the capacity can be determined from the formula,

$$Q = A * n * \frac{D}{2 * \pi} * \omega * \tan(\alpha) = V_p * \omega * \tan(\alpha)$$

And the torque needed to drive the pump is computed as,

$$T = V_p * \tan(\alpha) * (P_2 - P_1)$$

Where Q is the pump capacity (m^3/s), T is the driving torque (Nm), A is the piston cross sectional area (m^2), n is the number of pistons, D is the pitch diameter (m), α is the inclination angle, ω is the pump rotational velocity (rad/s) and $V_p = A * n * \frac{D}{2 * \pi}$ is the pump displacement per radian (m^3/rad).

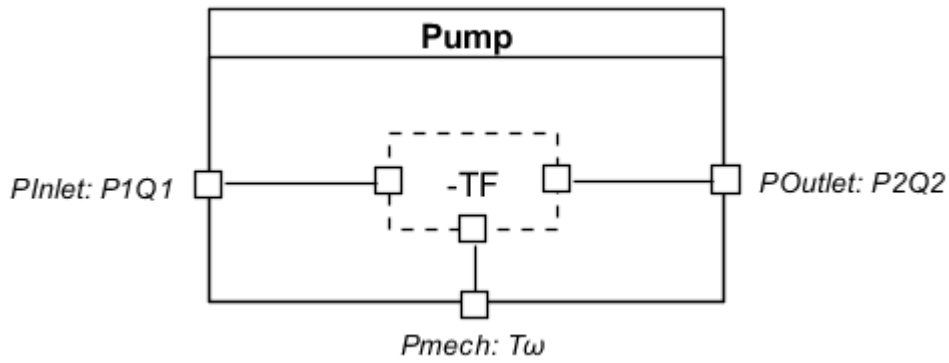


Fig. 3.31 The composition model of pump in first layer

From the dynamic aspect, the inlet flow variable Q_1 could be either a positive or negative flow in the system. The mechanical rotate velocity ω acts as the main power to drive the pump.

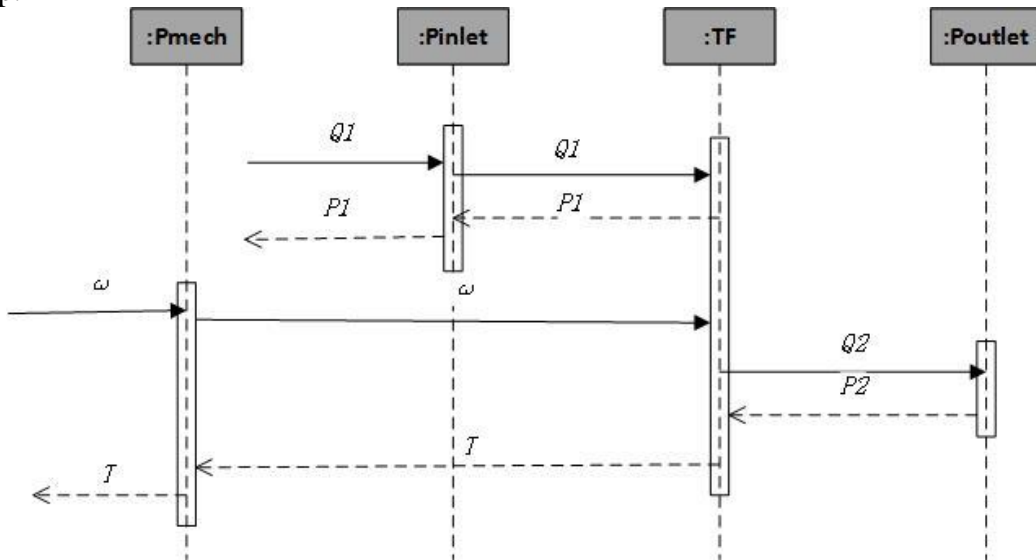


Fig. 3.32 Sequence diagram of pump in first layer

3.6.2. The second model layer

As a variable displacement pump, it has the function of pressure compensation by adjusting the angle of the swash plate according to the set pressure (P_{set}). The pressure compensated function is modelled by the pressure compensator (PC) adjusting the capacity of pump.

Physically, PC deals with the signal of pressure to adjust the angle of swash plate without energy transportation. When the outlet pressure (P_2) is lower than or equal to set pressure (P_{set}), the pump is generating flow in a proportion between set pressure and outlet pressure. Firstly the set pressure and maximum angle of swash plate are defined, then the angle of swash plate is computed as

$$\alpha = \alpha_{max} * \min([1.0, \frac{P_{set} - P_{outlet}}{K}])$$

otherwise, the pump flow is zero ($\alpha = 0$). K is a constant set pressure.

The structure form Fig. 3.33 presents the infrastructure for the realization of the use cases. The dynamic performance of second layer is shown in Fig. 3.34 activity diagram.

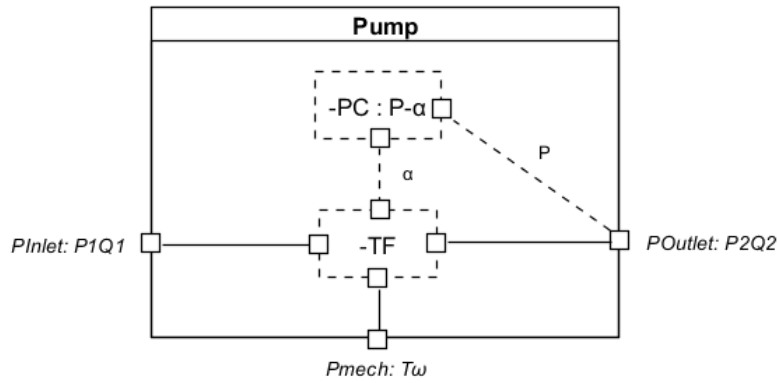


Fig. 3.33 Composition model of pump in second layer

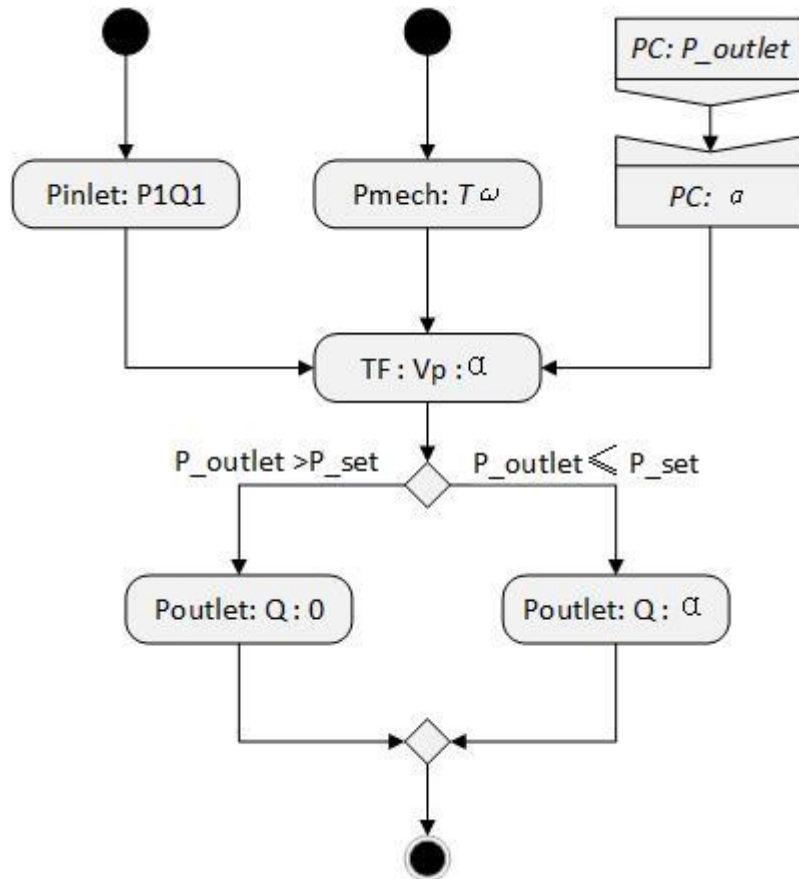


Fig. 3.34 Activity diagram of pump in second layer

Activity diagram visually shows that the activity pressure compensation is activated by the control signal of α . When α is zero ($P_{outlet} > P_{set}$), transformer TF stops working with a result of zero flow, otherwise, the activity running with a flow in proportion to signal α ($P_{outlet} \leq P_{set}$). It should be mentioned that when the transformer stops working, the mechanical energy still transfers the rotation velocity (flow variable) to the transformer without outcome.

3.6.3. The third model layer

Last two layers are all in the condition of an ideal world. The third model layer takes into consideration the leakage and mechanical friction. All activities happening inside of the pump model, hence, any changes relating to the environment does not change the performance of the pump as there are still three interfaces to outside world (Fig. 3.35).

There are two leakages in the pump model, one of which is an internal leakage from the high pressure port to the low pressure port. The other is an outlet leakage from each pump chamber pasting through the pistons to the case drain. It is described by internal and external conductance which is estimated from simple measurements or manufacturer data sheets. The internal conductance G_{in} is often much higher than the external conductance G_{ex} ($G_{in} = 2 \dots 6 G_{ex}$). The rough estimate of the total conductance is given by:

$$\text{Axial piston pumps with } 25 * 10^{-6} \text{ m}^3 < 2\pi * V_p < 200 * 10^{-6} \text{ m}^3$$

$$G = 8 * 10^{-8} * 2\pi * V_p - 1.2 * 10^{-12}$$

With: G conductance in $\text{m}^3/(\text{s} * \text{Pa})$

V_p volumetric displacement in m^3/rad

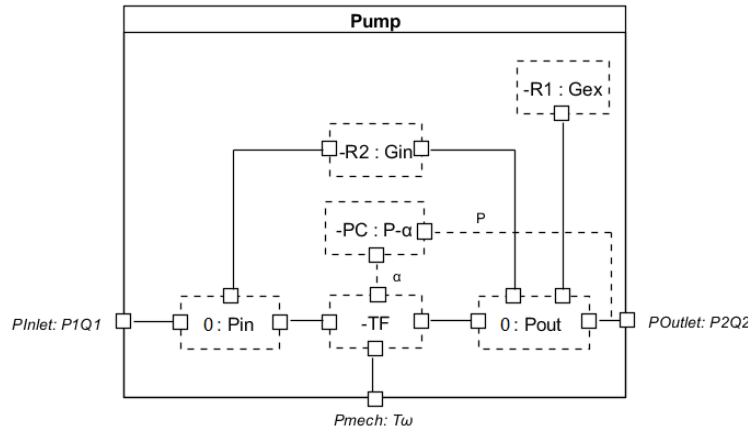


Fig. 3.35 Composition diagram of pump in third layer

Due to friction, there are mechanical energy losses in a pump which caused by bearings, seals and friction at the rotating parts. The torque loss is described by the torque efficiency, η_{tor} :

$$T = \frac{V_p * \tan(\alpha) * (P_2 - P_1)}{\eta_{tor}}$$

An appropriate model for the overall torque losses is given by:

$$T_{loss} = c_c + c_p * \Delta P + c_n * \omega$$

And Table 3-3 gives typical values for conductance and torque losses of axial piston pumps.

Table 3-3 Volume and torque losses of axial piston pumps

Volumetric displacement D in m^3	Conductance G [$\text{m}^3 / (\text{s} \cdot \text{Pa})$]	Constant Friction c_c [Nm]	Pressure dependent friction c_p [Nm / Pa]	Speed dependent friction c_n [$\text{Nm} / (\text{rad/s})$]
$30 \cdot 10^{-6}$	$1.492 \text{e-}012$	$8.4088 \text{e+}000$	$6.1968 \text{e-}007$	$5.5188 \text{e-}003$
$43 \cdot 10^{-6}$	$3.466 \text{e-}012$	$4.7536 \text{e+}000$	$5.5081 \text{e-}007$	$4.5970 \text{e-}003$
$50 \cdot 10^{-6}$ c	$2.833 \text{e-}012$	$1.5121 \text{e+}000$	$1.5611 \text{e-}007$	$3.4818 \text{e-}002$
$75 \cdot 10^{-6}$	$3.112 \text{e-}012$	$5.4443 \text{e+}000$	$1.4415 \text{e-}007$	$3.3614 \text{e-}002$
$100 \cdot 10^{-6}$ c	$5.201 \text{e-}012$	$3.7904 \text{e+}000$	$7.7440 \text{e-}007$	$9.2407 \text{e-}002$
$100 \cdot 10^{-6}$	$8.017 \text{e-}012$	$5.0966 \text{e+}000$	$5.4058 \text{e-}007$	$-1.3992 \text{e-}002$

105.10 ⁻⁶	<i>c</i>	5.03e-012	5.1283e+000	3.6003e-007	7.2576e-002
130·10 ⁻⁶		7.786e-012	-2.5540e-001	1.5723e-007	8.9084e-002
160·10 ⁻⁶		1.426e-011	7.8071e+000	6.9808e-007	3.9299e-002

(*c*: pump for closed circuit hydrostatic transmission)

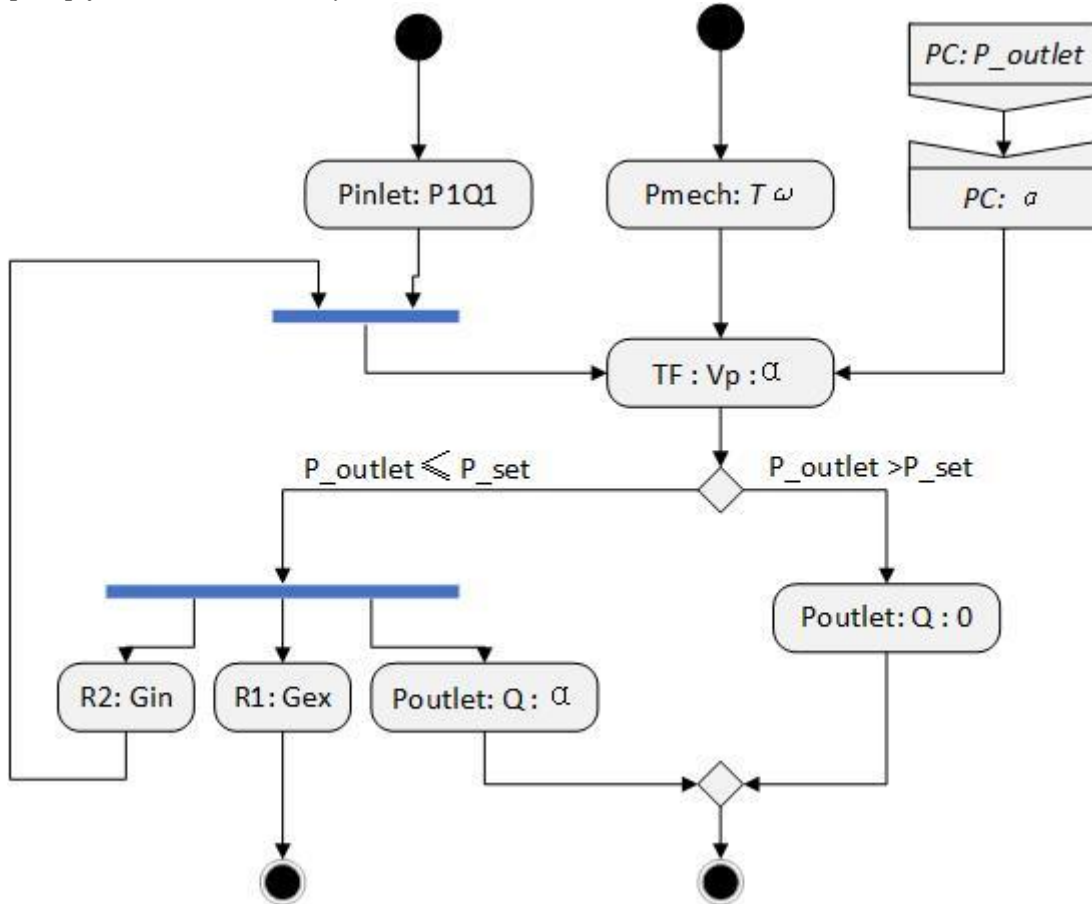


Fig. 3.36 Activity diagram of pump in third layer

Based on the second layer model, there are two more activities at the outlet of pump in the work regime $P_{outlet} \leq P_{set}$ for the leakages (Fig. 3.36).

3.6.4. The fourth layer model

The input pressure of a pump (P_1) can fall below atmospheric pressure because of hydraulic power losses, occurring in suction pipe, pip bends and filter. Additionally, the cavitation is always a crush problem exist in hydraulic system if the input pressure is too low. In order to avoid its occurrence, it is necessary to model the decrease of delivered flow when the input pressure decreases. Even more so, when the input pressure towards the vapour pressure of the fluid to detect the cavitation. The simulation result helps designer to modify the products before manufacture.

From the structure view of the model, there is no difference from the third layer model, therefore, the composition diagram is omitted. While the reduction of flow rate at the inlet of pump is modelled as a *R*-element with reference from Schulz (1979). The formula for piston pumps is given,

$$Q_{eff} = \begin{cases} Q_{normal} & P_{input} \geq P_{atmosphere} \\ Q_{normal} \left(1 - \left(1 - \frac{P_{input}}{P_{atmosphere}} \right)^2 \right) & \text{if } P_{input} < P_{atmosphere} \end{cases}$$

The cavitation is modelled as a simple judgement of that if the pump inlet pressure is lower than the vapour pressure of fluid oil, it will lead to the evaporation or boiling of oil, which is modelled as the function of “shut down” to the pump ($\omega=0$).

$$\begin{cases} \omega & \text{if } P_{inlet} > P_{vapour} \\ 0 & \text{if } P_{inlet} \leq P_{vapour} \end{cases}$$

The flow reduction and cavitation, as activities, are happening in the model visually showing in activity diagram (Fig. 3.37).

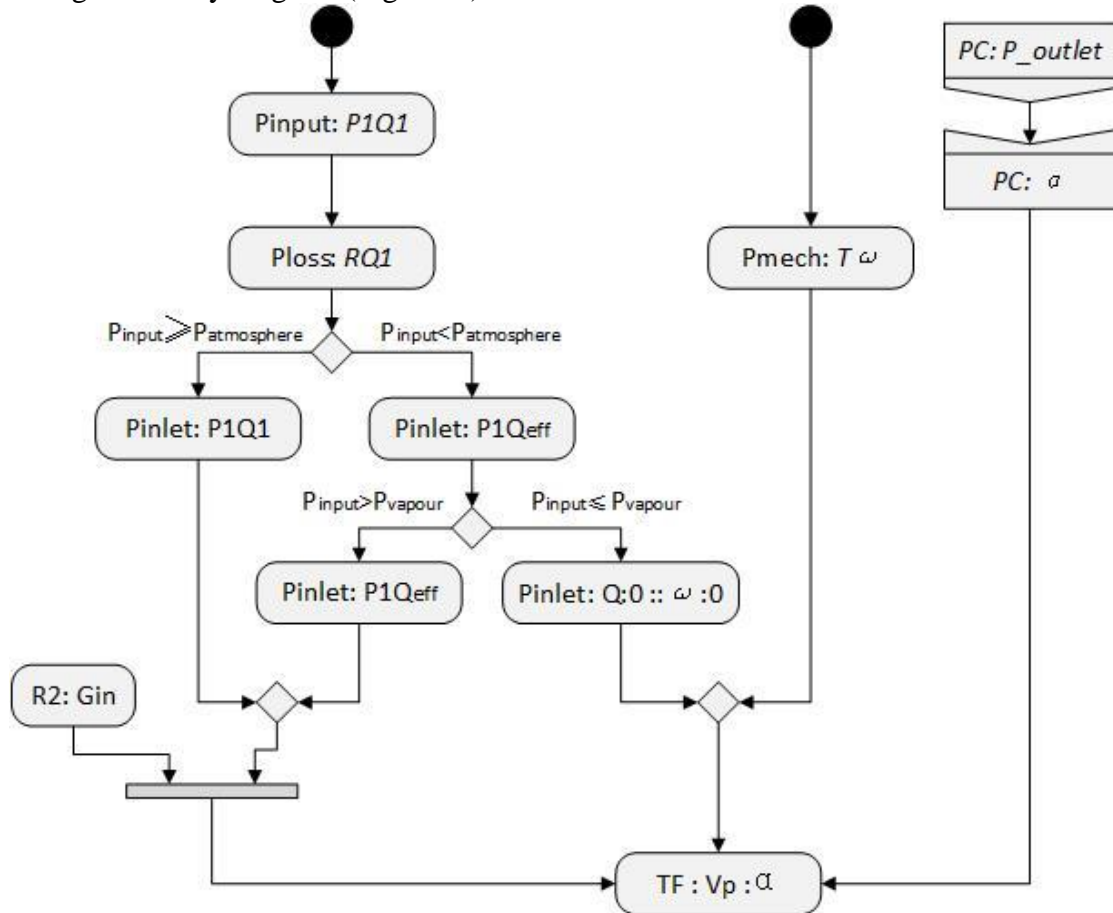


Fig. 3.37 Activity diagram of pump in fourth layer (Part)

It is not necessary to make changes in the last layer model, therefore, all changes are localised on a defined part of the model (Fig. 3.37). Before the fluid enters the pump (TF), the system first makes judgements on the inlet pressure. If the inlet pressure is not lower than the atmospheric pressure, the fluid will flow directly into the pump without any changes. Otherwise, the inlet flow rate will turn into a reduction as a function of reduced inlet pressure. Furthermore, if the inlet pressure is even not higher than vapour pressure of fluid oil, then a cavitation will occur with a signal output of $\omega=0$ to stop the pump.

3.6.5. Model classification

After the complete analysis in both structure and behaviour through the OOM layer approach, the component model can be modelled by BG in 20sim. Importantly, the OOM approach is also able to give a behaviour classification for models, especially for pumps, Statechart behaviour description is not necessary to show since the behaviours of a pump

to some extent mainly regarding experience factors, such as, mechanical efficiency, flow conductance, are usually treated as a constant number. In addition the OOM approach is already sufficient enough for this project, if the designer is going to explore deeper knowledge of behaviour, then Statecharts is helpful to conduct further research. Firstly, the “Ideal model” of the axis piston pump is simply modelled with only a transformer *MTF*-element to transform the mechanical energy into hydraulic energy.

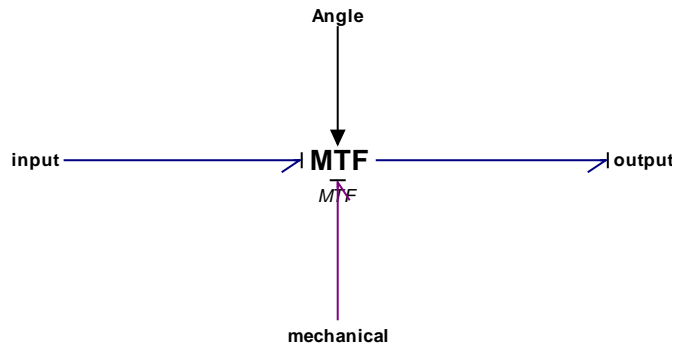


Fig. 3.38 The Ideal pump model

The signal “*Angle*” is a constant value to give a fixed capacity of volume. In the initial design, the pump is treated only as a flow source to supply the fluid oil into system. In “Standard model”, the leakages and pressure compensated function are added to model. The coefficient for hydraulic and mechanical efficiency is taken all from the experienced factor. The standard pump model can sufficiently express all behaviours for a hydraulic system design. Based on the “Ideal model”, two *O*-junctions are inserted at input and output respectively for different pressures. Two *R*-elements are for modelling the internal and external leakage, and the conductance is according to the volume displacement of the pump which is a signal in BG model from *MTF*-element to *R*-element. Sub-model *PC* is the function of pressure compensation.

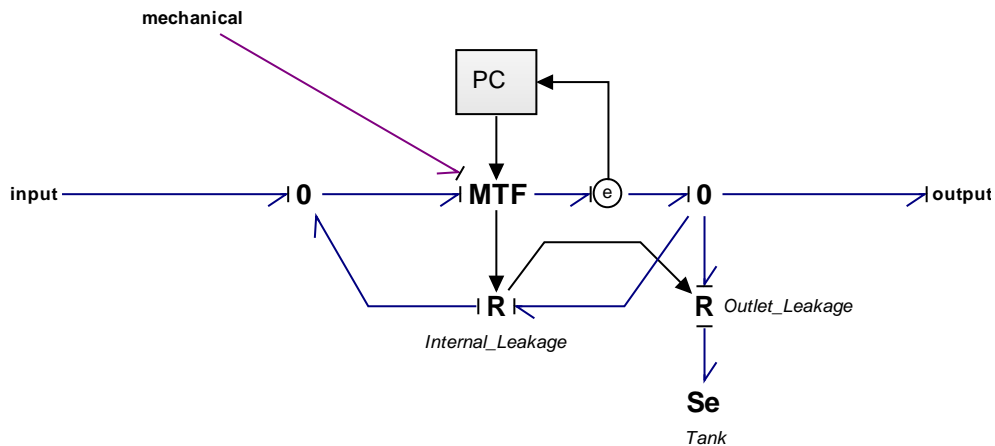


Fig. 3.39 The Standard pump model

As the description in OOM layer approach, the behaviour of cavitation needs to be modelled in “Advanced model” to check whether the design of a pump that has a potential cavitation problem, as well as how much flowrate deduction is at the inlet port.

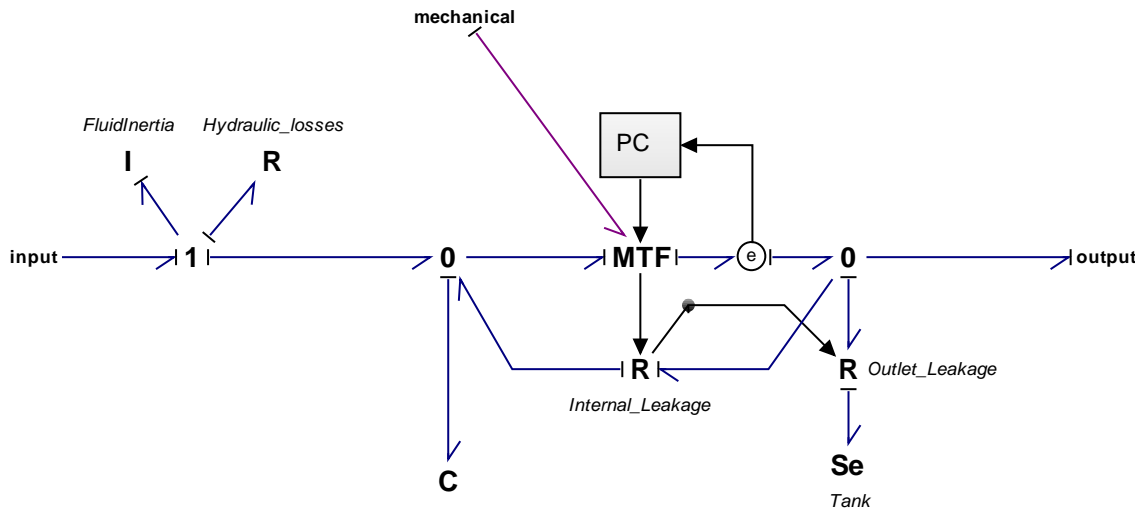


Fig. 3.40 The advanced pump model

Based on the “Standard model”, an *R*-element is added for the purpose of performing hydraulic losses before the inlet of the pump. If cavitation occurs, the outlet flow will be zero programmed in *MTF*-element.

3.7. Hydraulic motor

The hydraulic motor has a reversed function of the pump, which is used to convert the hydraulic power into mechanical power to provide the rotary motion. The symbol is shown in Fig. 3.41.

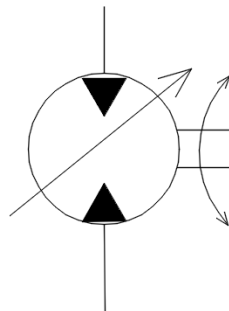


Fig. 3.41 The symbol of the variable displacement hydraulic motor

The motor speed depends on the flow rate, while the supply pressure depends mainly on the motor loading torque. Therefore, an ideal motor with no leakage and no friction is similar to an ideal pump. The formulas to describe motor speed and torque are

$$n_m = \frac{Q}{V_m}$$

$$T_m = V_m * (P_{in} - P_{out})$$

n_m = Motor speed [rad/s]

Q = Flow rate [m^3/s]

V_m = Geometric volume of motor [m^3/rad]

T_m = Loading torque [Nm]

P_{in}, P_{out} = Motor inlet pressure and outlet pressure [Pa]

Motors are able to run in both directions and all speeds between 0 and the maximum speed. In real motor case, due to the internal leakage and the mechanical friction, the energy losses need to be considered. Considerable pulsation of torque or speed may occur because of the low frequency change of pistons, which may increase the energy loss.

The torque losses due to the mechanical friction is given as

$$T_{loss} = C_c + C_p * \Delta P + C_n * n_m$$

The volume losses due to the speed dependency of the internal leakage is modelled by formula

$$Q_{loss} = G_p * \Delta P + G_n * n_m$$

The factors in equations are given in Table 3-4.

Volumetric Displacement	Pressure Dependent Conductance	Speed Dependent Conductance	Constant Friction	Pressure Dependent Friction	Speed Dependent Friction
V_m [m ³]	G_p [m ³ /(s · Pa)]	G_n [m ³ /rad]	C_c [Nm]	C_p [Nm/Pa]	C_n [Nm/(rad/s)]
$35 * 10^{-6}$	1.0565e-12	3.0393e-8	-2.8465e-1	3.7989e-7	2.1217e-2
$50 * 10^{-6}$	1.2798e-12	5.9098e-8	4.0394	5.5825e-7	5.5243e-2
$75 * 10^{-6}$	1.5489e-12	3.6158e-8	5.7974	4.7399e-7	4.1173e-2

Table 3-4 The volume and torque losses of axial piston motors

In the aspect of modelling, there is no big difference between a motor and a pump. Since the function of a motor is the reverse of a pump, the modelling process of the motor can be based on the pump modelling process. The pump will be modelled and classified directly according to the pump model. While there is still some important differences between a real pump and a real motor that needs to notice. First, the pump is usually optimised for one direction of rotation and a certain speed range. The motor is used for both directions and all speeds range from 0 to the maximum speed. Second, the mechanical energy is an input energy for a pump, but it is an output energy for a hydraulic motor. Third, the cavitation rarely happens in a hydraulic motor that is not necessary to program in a model.

3.7.1. Model classification

According to the OOM layer approach description of the pump and the classification of behaviour models. The “Ideal model” of the motor is completely same as the “Ideal model” of the pump. Meanwhile, for a motor the most complex field is located at the energy losses, torque loss and leakage, but there is no an exact formula for the model, therefore, the motor model is only modelled by the “Standard model”. If designers would investigate deeper for energy losses, which needs to do an experiment to attain the loss coefficient for models.

Standard model

Based on the “Ideal model”, the internal leakage and torque loss are modelled in the “Standard model” (Fig. 3.42).

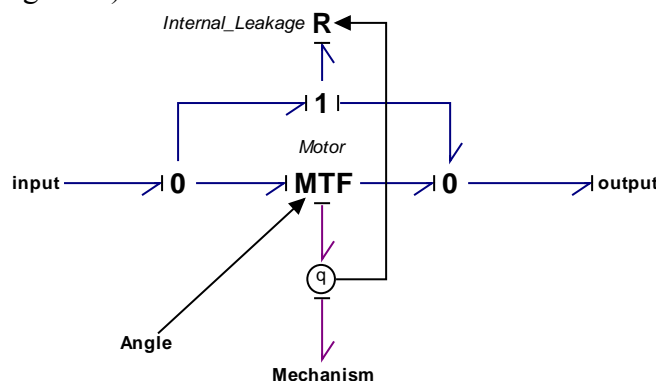


Fig. 3.42 The Standard BG model of the hydraulic motor

Firstly, four interfaces are defined to interact with the environment, one hydraulic energy input, one hydraulic output, one mechanical energy output and an *Angle* signal (varying displacement). Two *O*-junctions are added as pressure difference at the hydraulic inlet and outlet. Then the motor is modelled as a *MTF*-element between two *O*-junctions, and one *I*-junction is inserted to connect an *R*-element for internal leakage. The internal leakage is the motor rotation speed dependent, therefore, the *q*-sensor is amounted at the mechanical output to sense the rotation speed transporting to *R*-element as a signal. Last, all elements and interfaces are connected by bonds and single arrows.

3.8.Pipe Flow

In the hydraulic system, the components of system are interconnected by hydraulic transmission pipe which is a container for hydraulic liquid flows. The transmission pipe affects system performance in some ways:

- Hydraulic friction losses; hydraulic resistance of pipes
- Hydraulic local, or secondary, pressure losses
- Oil compressibility and elasticity of pipe material; hydraulic capacitance of lines
- Oil inertia; the hydraulic inertia of pipes

Mostly, the short pipes with large diameter are reasonable to ignore the resistance, capacitance and fluid inertia, since they are a very minor dynamic effect relative to the system. While if the pipes are long, are of small bore, or the system is being driven in a highly dynamic model, the dynamic models should arise a situation with some or all of a pipes resistance, capacitance and fluid inertia. Furthermore, the material of pipe, rigid tubing or flexible hoses, is also an important factor for the performance of a system. The pipe as a component in the hydraulic system is simplest only for transmitting the fluid oil with hydraulic energy from one component to other components. Therefore, the description by a composition diagram and an activity diagram is proficient to give both static and dynamic analysis. Then the statechart gives the description of the different complexity of models.

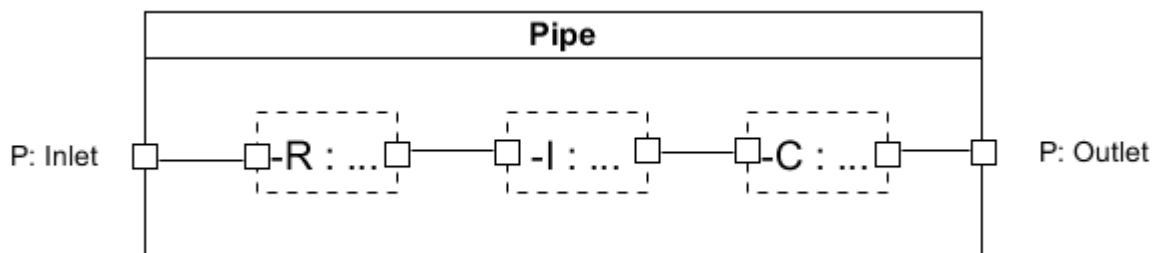


Fig. 3.43 The composition diagram of pipe flow

Literately, the pipes have two ports, inlet and outlet interacting with environment (Fig. 3.43). Inside, the motion of the fluid oil in the transient conditions takes place under the action of the fluid inertia, friction, and compressibility, as well as the driving pressure forces. The oil velocity, pressure, and the temperature vary from point to point along the pipe along the pipe length and pipe radius.

During the fluid flowing in a pipe from inlet (*Pinlet*) to outlet (*Poutlet*), there are three actions taking place along the process. In order to obtain a fairly precise model, it is assumed that the effect of line resistance, inertia, and capacitance are separate and each of them is localised in one of three separate actions in the process.

Since the fluid oil (*PIQ1*) flows into the port *Pinlet*, the fluid oil moves as one lump under the action (*Ploss*) of the friction forces. Therefore, the motion of the fluid oil can be described by

$$P_1 - P_r = R * Q_1$$

The following action (*Fluid_inertia*) describes the motion of the fluid oil lump of its inertia *I* according to

$$P_r - P_2 = I \frac{dQ_1}{dt}$$

By considering the effect of oil compressibility, the action *Compressibility* is happening with the motion deduced by

$$Q_2 - Q_1 = C \frac{dP_2}{dt}$$

At last the fluid oil (*P2Q2*) flows out through port *Poutlet*.

3.8.1. Bond Graph model

According to the analysis of OOM layer approach, the simplified BG model can be computed.

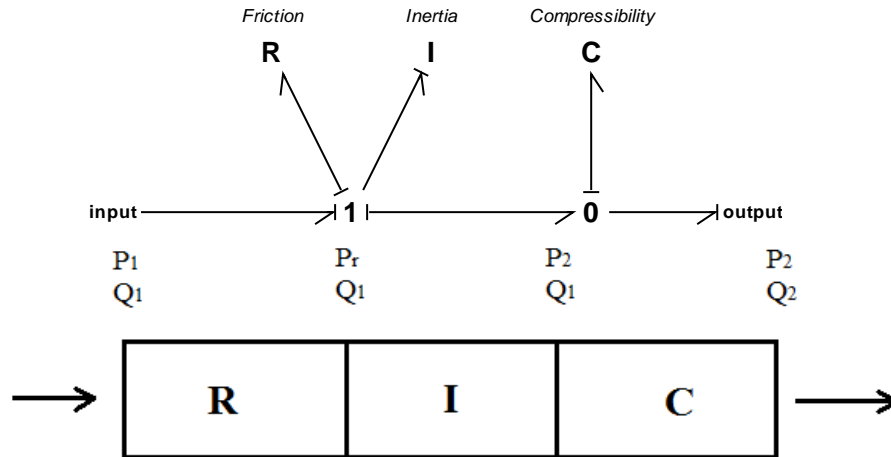


Fig. 3.45 The single-lumped Bond Graph model

The developed model is single dimension. The fluid speed and pressure are thought of as averaged quantities over the cross section of the pipe. The effect of pipe resistance, inertia, and capacitance are assumed to be localised in one of three blocks in the pipe (Fig. 3.45). The effect of the resistance of the whole pipe is localised in the first block, the effect of the inertia of the whole pipe is localised in the second block, while the effect of the pipe capacitance takes place in the third block.

In the BG model, the effect of pipe resistance (*R*-element) and the effect of pipe inertia (*I*-element) are connected to *I*-junction to describe the pressure change (*P2-PI*) with

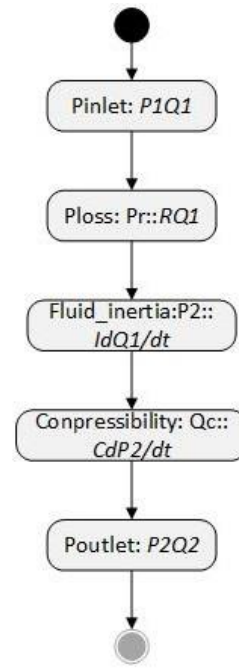


Fig. 3.44 The activity diagram of the pipe

unchanged fluid flow rate ($Q1$), the effect of fluid compressibility is connected to 0-junction to generate a volume change ($Q2-Q1$) with unchanged pressure ($P2$).

3.8.2. Behaviour description in Statechart

It is not clear to see the flexibility or behaviour hierarchy of the model from OOM approach analysis. The statechart analysis will give a clear behaviour hierarchy. First of all, the states are introduced.

Hoses

Hoses are used to interconnect elements that vibrate or move relative to each other. Different from the rigid tubing, hoses insure the required flexibility and can operate at high pressures, the proper hose diameter is determined according to the maximum flow rate and the selected fluid speed.

$$D = \sqrt{\frac{4Q_{max}}{\pi v}}$$

D = Hose inner diameter [m];

Q_{max} = Maximum flow rate [m^3/s];

v = Fluid mean velocity [m/s].

There are some typical constructional and operational parameters of high-pressure hydraulic hoses shown in Table 3-5.

Inner Diameter	External Diameter	Pressure [Bar]		Minimum Radius
		Operating	Rupture	
mm	mm			mm
9.52	21.4	350	1400	125
12.7	24.6	280	1100	180
19.1	31.7	210	850	240
25.4	39.7	210	850	305

Table 3-5 Typical hose construction, dimensions, and operating pressure

Pressure and power losses in hydraulic conduits

Minor Losses

The minor losses in the hydraulic systems result from the rapid variation of magnitude or direction of the oil velocity. The local pressure losses are computed as

$$\Delta P = \xi \frac{\rho v^2}{2}$$

ΔP = Pressure losses [Pa];

ξ = Local loss coefficient

ρ = Fluid density [kg/m^3]

In laminar flow, the effects of local disturbances are usually insignificant compared with the friction losses. In turbulent flow, the local loss coefficient is determined almost exclusively by the geometry of the local feature. Table 3-6 gives the values of the local loss coefficient for typical local loss elements.

Local Feature	ξ
Flexible pipe connection	0.3
Standard 90° elbow	1.2-1.3
Tee junction	3.5
Pipe inlet	0.5-1
Pipe outlet	1

Table 3-6 Local loss coefficient of typical local loss features

Friction Losses

The pressure losses in the pipes depend mainly on the geometry, surface roughness, fluid properties, and Reynolds number.

The three main types of flow are laminar flow, turbulent flow, and transition flow.

Laminar flow is a streamlined flow of viscous fluid, where all particles of the fluid move in distinct and separate lines. The type of flow is determined by calculating the Reynolds number.

$$Re = \frac{vD}{\nu} = \frac{\rho v D}{\mu}$$

D = Inner pipe diameter [m];

v = Mean fluid velocity = $4Q/\pi D^2$ [m/s];

μ = Dynamic viscosity [Pa.s];

ν = kinematic viscosity [m^2/s].

The friction losses in the pipe are calculated by the following expression:

$$\Delta P = \lambda \frac{L}{D} \frac{\rho v^2}{2}$$

The friction coefficient depends mainly of the Reynolds number (Table 3-7).

Laminar flow	$\lambda = \frac{64}{Re}$	Re < 2300	Hagen Poiseuille's law, 1856
Turbulent flow – Smooth pipe	$\lambda = \frac{0.3164}{\sqrt[4]{Re}}$	2300 < Re < 10 ⁵	Blasiu's law, 1915
	$\lambda = 0.0054 + 0.396(Re)^{-0.3}$	10 ⁵ < Re < 0.2 * 10 ⁶	Herman's law, 1930
Turbulent flow – Rough pipe	$\frac{1}{\sqrt{\lambda}} = -2 \log \left(\frac{\epsilon}{3.7D} + \frac{2.51}{Re\sqrt{\lambda}} \right)$	For the whole range of turbulent flow	Colebrook and White, 1939
	Moody's diagram shown in Fig. 3.46		

Table 3-7 Determination of the pipe line friction coefficient

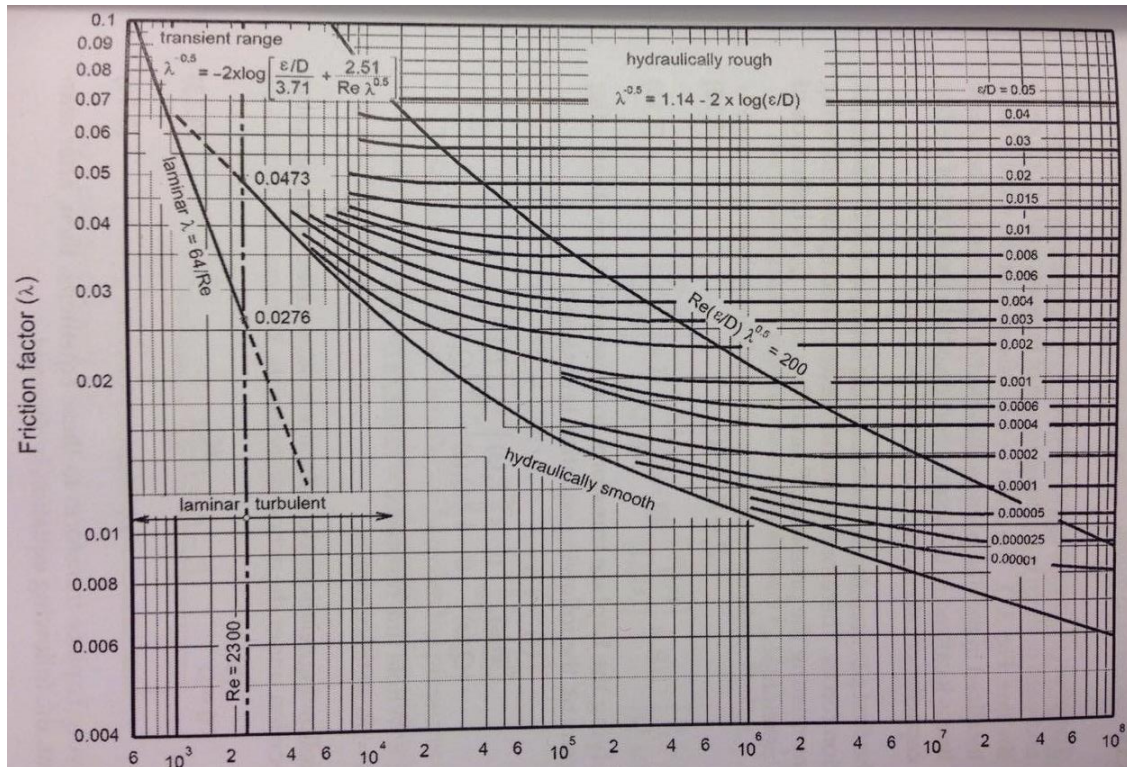


Fig. 3.46 Moody's diagram

The transition from the laminar flow to the turbulent one takes place at a critical value of the Reynolds number. As a rough guide, it is possible to say that flow will be turbulent for $Re > 2300$. The transition process is a consequence of the instability of laminar flow. The uncertainty of the critical value is due to the fact that the processes, near their stability limits, are easily destabilised even by minute disturbance effects (such as noise of the pump).

Fluid inertia

The fluid inertia computes the pressure differential, due to change in fluid velocity, across a pipe of constant cross-sectional area. The pressure differential is determined according to the following equation

$$I = \frac{4\rho L}{\pi D^2}$$

Compressibility

The effect of the pipe capacitance is introduced in Chapter 3.1, Hydraulic fluid.

Statechart map

By using statechart formalism, the pipe flow behaviours can be organised in three parallel state activities. They are orthogonal states *Power losses*, *Fluid Inertia* and *Compressibility*.

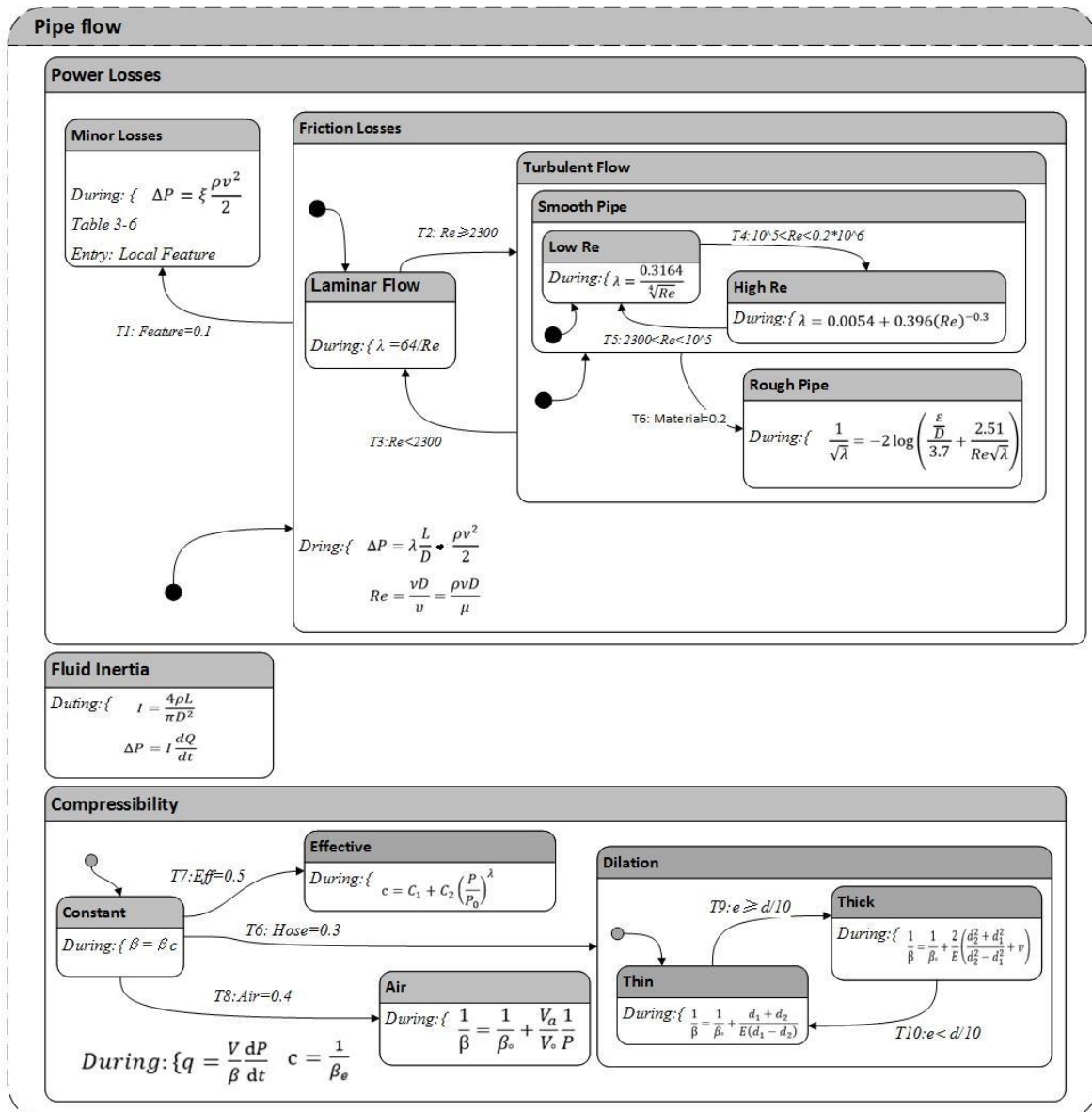


Fig. 3.47 The analysis of statecharts for the pipe flow

The state *Power losses* is a compound OR state with *Friction Losses* and *Minor Losses*, the *Friction Losses* state is defaulted as an entry state, depending on the Local feature, the state *Minor Losses* will happen by changing the *Feature* variable into *0.1*. The local loss coefficient is showing in Table 3-4. Furthermore, the state *Friction Losses* is decomposed into the *Laminar Flow* ($Re < 2300$) and *Turbulent Flow* ($Re \geq 2300$) according to the Reynold's number. The even lower-level description is attained in the *Turbulent Flow*, since the different material of the pipe will have a varying friction coefficient. The state smooth pipe is defaulted as an entry with two decomposed sub-states *Low Re* (*T5*) and *High Re* (*T4*). While the material can be changed into the rough pipe by changing the *Material* variable into *0.2* then the state *Rough Pipe* is activated.

The high-level state *Fluid inertia* is also a low-level state without a state decomposition process.

In state *Compressibility*, the state *Constant* is an entry state representing that the constant bulk modulus is defaulted for a model. If the users want to take the pressure and air factor into consideration, they can change the variable *Eff* and *Air* to *0.5* and *0.4* to activate the state *Effective* and *Air*. On the other hand, the state *Dilation* is defined for hoses, therefore,

if the hose models are modelled in hydraulic system, by changing the *Hose* variable to 0.3, then the pipe flow is changed from rigid tubing to flexible hose while the *Dilation* state becomes active.

3.8.3. Model Classification

The basic function for the pipe is to transport fluid oil from one port to others without any behaviour caused by the pipe actions. The ideal pipe is no energy losses ($P_1Q_1 = P_2Q_2$), hence, the “Ideal model” of the pipe flow will be developed as a *I*-junction.

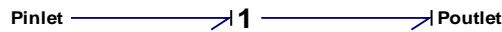


Fig. 3.48 The Ideal Bond Graph model of the pipe flow

The “Standard model” is modelled by a single-lumped model including all three effects of states. The pipe line is treated as a smooth, normal feature and rigid tubing with assumed that no air exists in system. From the statechart aspect, the tasks *T1* and *T6* are not considered to be activated. Meanwhile, the state *Compressibility* is also only activated by its constant bulk modulus regardless the effect of pressure and air content. The BG model is same showing in Fig. 3.45.

The “Advanced model” needs to be more specific incorporating all aspects, while some actions need to be activated by changing the value of variables. Another modelling area needed to specify is that the single-lumped is not precise enough for a pipe especially when the pipe is long. In reference [7] *Fluid Power engineering*, it shows that the simulation result of models are much closer to the experimental results when the pipe flow (18 meters long) is modelled by the four-lump model. Consequently, in the process of modification for a hydraulic system design, the more precise “Advanced model” is the multi-lumped model that gives the simulation result by including all effects of variables. The BG model will be connected in a serial way for a multi-lumped model.

3.9. Control Volume

During the modelling process, one of most important factor of causality has to be considered carefully for the simulation. Somewhere in system that needs to add control volume (*C*-element) for the compressibility of fluid, especially in front and behind valves. And a relief valve works as a pressure control valve to release the pressure when pump is fixed axil pump without the pressure compensated function. The formula for compressibility is introduced in Chapter 3.1 Hydraulic fluid.

4. Hydraulic power system for Marine crane

The typical hydraulic power system is the system of a knuckle boom cranes (KNC), which is special designed for the wide range use of an offshore environment. The structure of KNC mainly consist of three parts, a crane house which is bolted via the slew bearing to the pedestal, the main boom is hinged at the middle creating a knuckle boom and the outer boom for getting the outreach. There are three joints which the crane house rotates controlled with hydraulic motor and both booms are controlled with hydraulic cylinders. The hydraulic system for KNC is designed by a composition of component models in the library developed in this project. The simplified hydraulic system diagram depicts the main components as shown in Fig. 4.1.

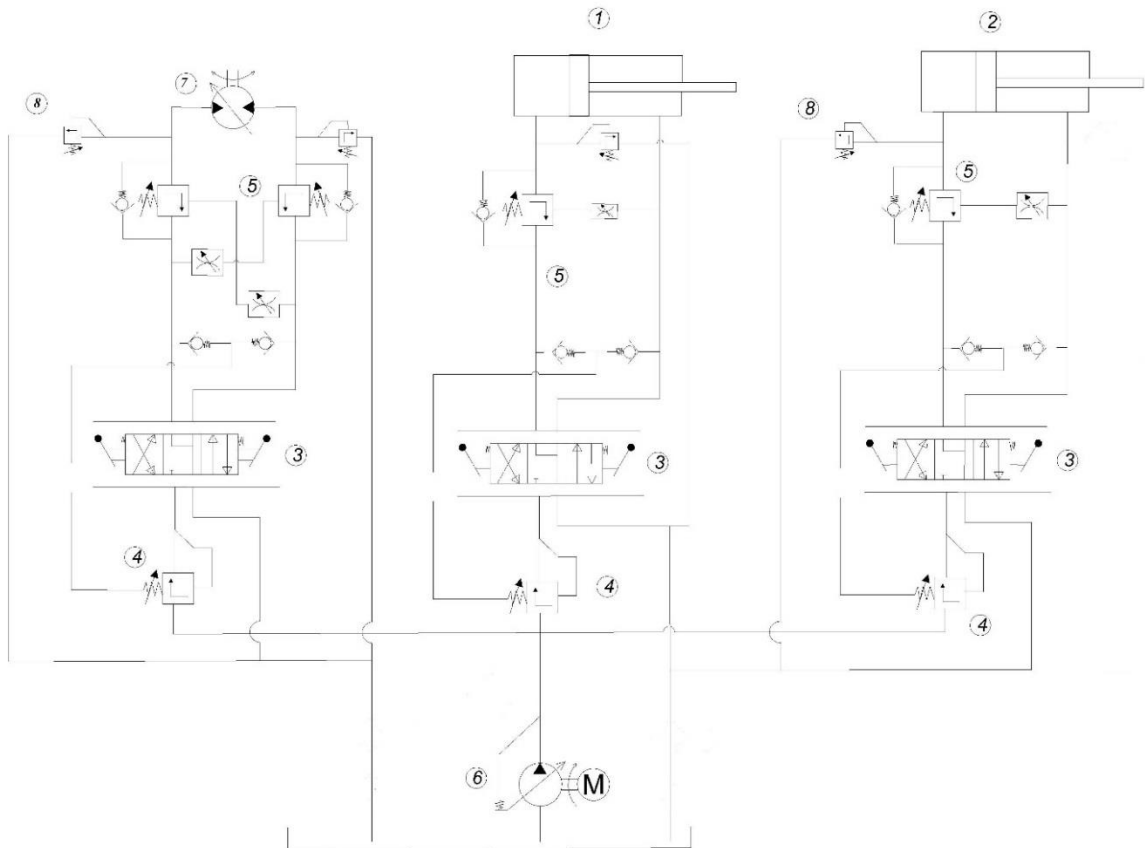


Fig. 4.1 The hydraulic system of Knuckle Boom Crane

In a real hydraulic system, there should be more detailed design, like couple of DCVs working corporately, to have a more stable movement. In addition at least two pumps should be installed for a purpose of redundancy, and more components, like coolers, breaks for the motor and sensors for the controller, are needed for a hydraulic system. While this hydraulic system is proficient for the component model library to specify and refine models according to the simulation result.

①	CYLINDER OF ELBOW DERRICK
②	CYLINDER OF MAIN DERRICK
③	4/3 DIRECTIONAL CONTROL VALVE
④	PRESSURE COMPENSATOR
⑤	LOAD CONTROL VALVE WITH COUNTER BALANCED VALVE
⑥	PRESSURE COMPENSATED PUMP
⑦	HYDRAULIC MOTOR
⑧	PRESSURE RELIEF VALVE

In order to make a simple and clear map of design process for a virtual prototyping framework for overall crane system design, a simple hydraulic lifting system that the cylinder connects directly to a rigid body of load as shown in Fig. 4.2, The behaviour models will be changed from the “Ideal model” to the “Standard model” to the “Advanced model” to validate the effectiveness and efficiency of the simulation by using the classified models. The load will be lifted up from 10 to 30 seconds then lowered down to 50 seconds.

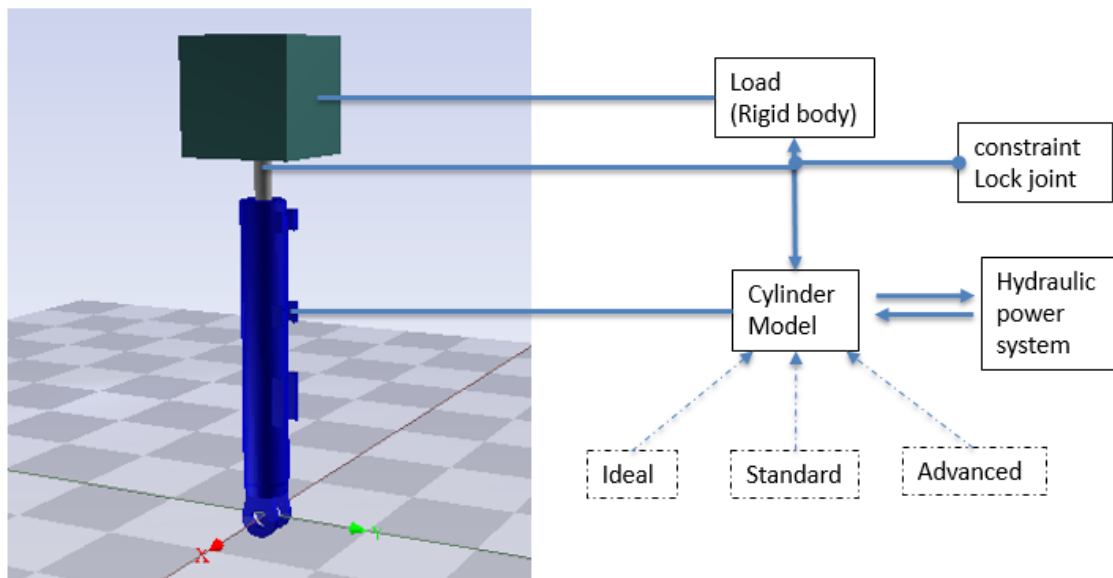


Fig. 4.2 The simple hydraulic cylinder test

Next, a simple BG sub-hydraulic of KBC is built in a very fast way by constructing the component models. Because of the causality, the control volume is added somewhere in front or behind of orifices. And one relief valve is add for release the pressure if there is no flow running through the DCV. The fixed axis piston pump is given a constant signal angle for “Ideal model” to give a constant flow. In order to apart from the influence factors from human control, the DCV is controlled via a sine function as the control signal. The *Load* is amounted directly on the cylinder.

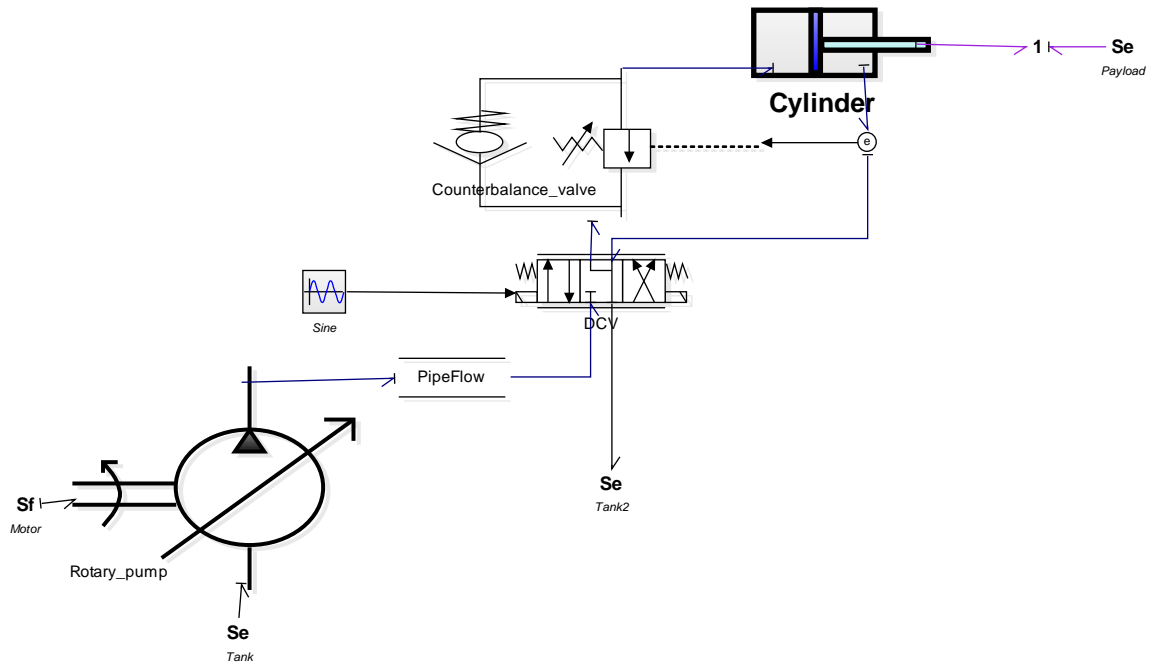


Fig. 4.3 The BG model of one actuator hydraulic system for Knuckle Boom Crane

The behaviours are affected by the parameters, which are defined by adopting the case from the assignment of “Design of Parallel linked crane for offshore lifting”. The working load is defined as 10 tons and the cylinder working stroke is 2 meters. The control signal sending to DCV is a simple sine function as

$$Input = \sin(time)$$

The main dimension parameters for components are calculated according to the payload and the cylinder working stroke which are shown in Appendix.

5. Simulation Result

Simulation is an important and useful tool available to those responsible for the design and operation of complex processes or systems. It allows designer to experiment with systems which would be impossible or impractical otherwise. With the developed component model library, the assembly of a hydraulic sub-system is straight forward. The simulation result will show how different from the classified models in the hydraulic system of KBC, and to prove that the different level models are sufficient for different time being of a design process of a hydraulic sub-system.

5.1. *Simulation result of cylinder model*

At the beginning, from the static point of view, the different implemented cylinder behaviour models have varying parameters as shown in Table 5.1.

	“Ideal Model”	“Standard Model”	“Advanced Model”
User Interface	Piston diameter Rod diameter Stroke End Stopper characters Rod mass	Piston diameter Rod diameter Stroke End Stopper characters Rod mass Inlet diameter Outlet diameter Dead volume Bulk modulus Friction ratio	Piston diameter Rod diameter Stroke End Stopper characters Rod mass Inlet diameter Outlet diameter Dead volume Bulk modulus Cylinder outside Diameter Piston gap to cylinder house Viscosity Friction characters (5)
Simulation running time (50s)	15 s	18 s	21 s
Behaviour Interface	Port A Port B Port M	Port A Port B Port M	Port A Port B Port M

Table 5-1 the parameters of three classified models

The simplest “Ideal model” has the least parameters to consider for the beginning process of a hydraulic system design. As the complexity increases, the more parameters need to be considered, especially for the “Advanced model”, such as, the material parameters of the seal of the cylinder.

Then based on the different implementations of the cylinder behaviour model, the displacement of the cylinder piston is shown as in Fig. 5.1. The piston with a stroke of 2m started moving from its position at 0.1m. The “Ideal model” does not include the flow restrictions as the physical ports of the cylinder oil chambers and the seal friction. As a result, the behaviour of the piston is only determined by the pressures and the force from the load. The piston displacement stopped increasing at 2m defined the arbitrary velocity stopper. With the extended implementation, the maximum deformation of the piston bumpers is set at 0.01m, hence the piston displacement stopped at 2.1m approximately. The “Standard model” which includes the seal friction modelled by a constant of the load independent frictional loss of 5%. The movement is slower than the “Ideal model”. In the real hydraulic system, the internal piston leakage and the rod external leakage need to be considered based on the clearance between the seals and the piston and the rod, which is represented by a slightly lowering when the cylinder is not working. Furthermore, the seal

friction is modelled based on the detail seal material during the modification period of the design process.

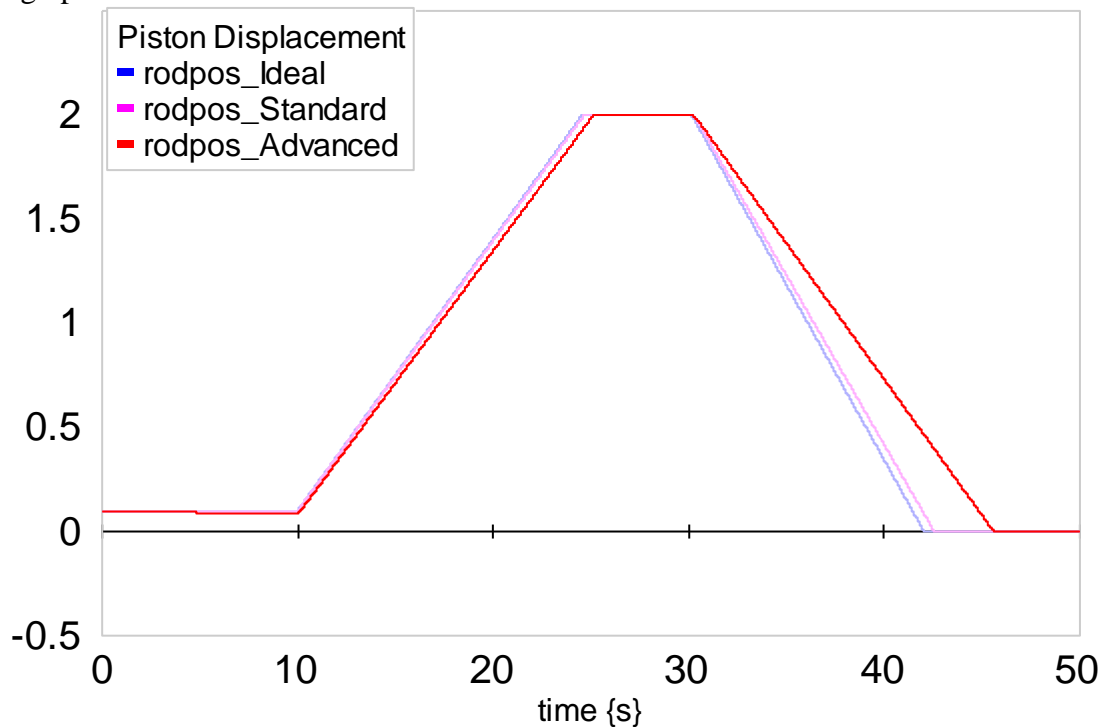


Fig. 5.1 The displacement of the cylinder piston of three levels model

In “Standard model” and “Advanced model”, the inlet and outlet orifices of the cylinder are defined, which represents the flow with a slight slop when it is going up or down (Fig. 5.2). While the flow plot shows that the leakage behaviour in “Advanced model”.

In pressure plots picture Fig. 5.3, the pressure is always going up and down with an oscillation since the fluid compressibility is modelled in “Standard model” and “Advanced model”. Furthermore, the pressure in “Advanced model” is lower than other two behaviour models when the cylinder stop working but higher in working process.

The differences of the plotted results between three behaviour models are minimal, since the hydraulic power system is simple and the control signal sends to DCV is a constant full open signal without and rough or surge performance. The oscillations of flow and pressure at the beginning of the simulation due to the initial conditions are not impossible to be ensured properly, and when switching the moving direction due to the momentum of the load. The plotted results proves that the feature of efficiency, effectiveness and complexity of behaviour models, which is valid for the co-simulation approach in a much more flexibility for modelling in terms of modifications, standardisation and reuse of knowledge.

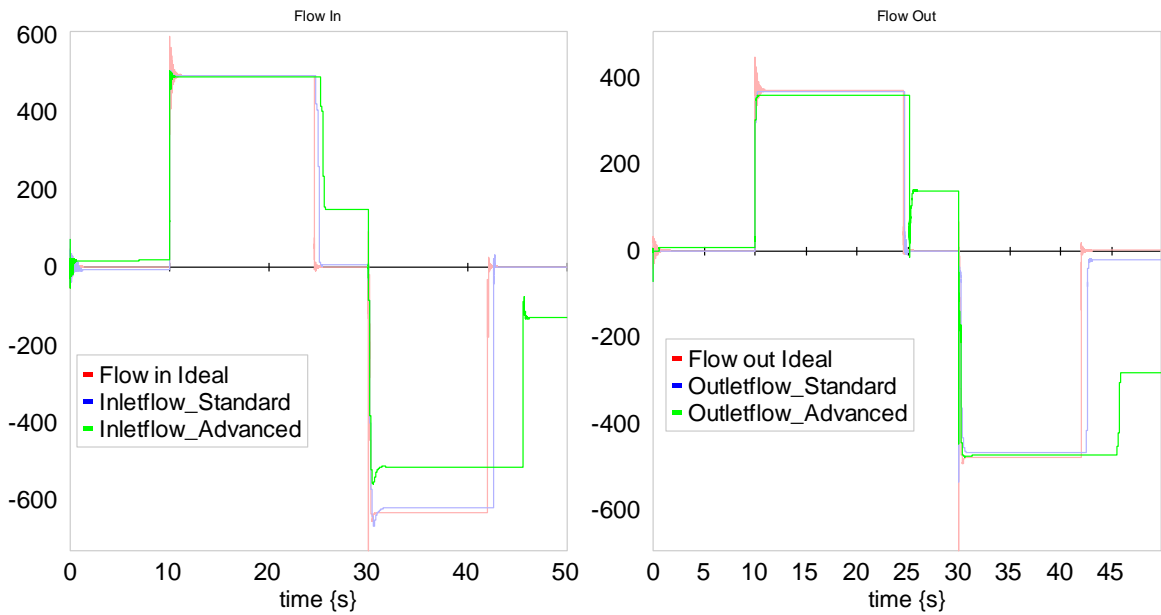


Fig. 5.2 The inlet&outlet flow of the cylinder models

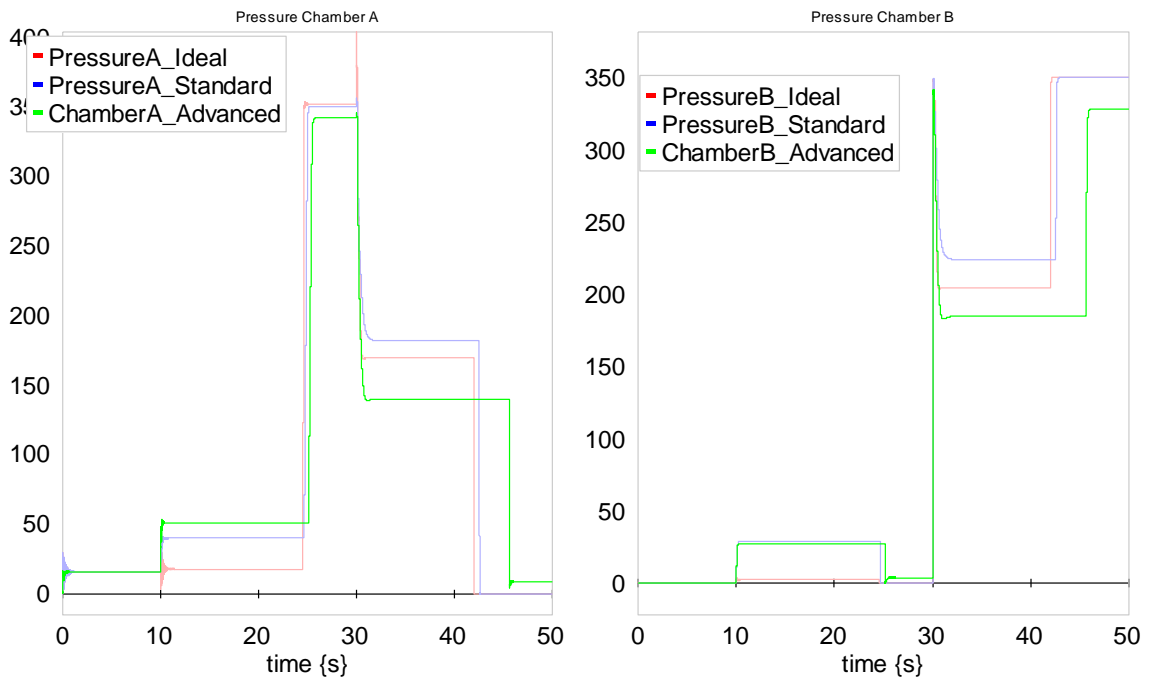


Fig. 5.3 The pressure in the chambers of the cylinder models

5.2.Simulation result of “Ideal model”

The result is showing that whether the “Ideal model” is working sufficiently or not for the conceptual design of a hydraulic system. In the simulation result, the pressures, flows and the movement of *Load* are measured.

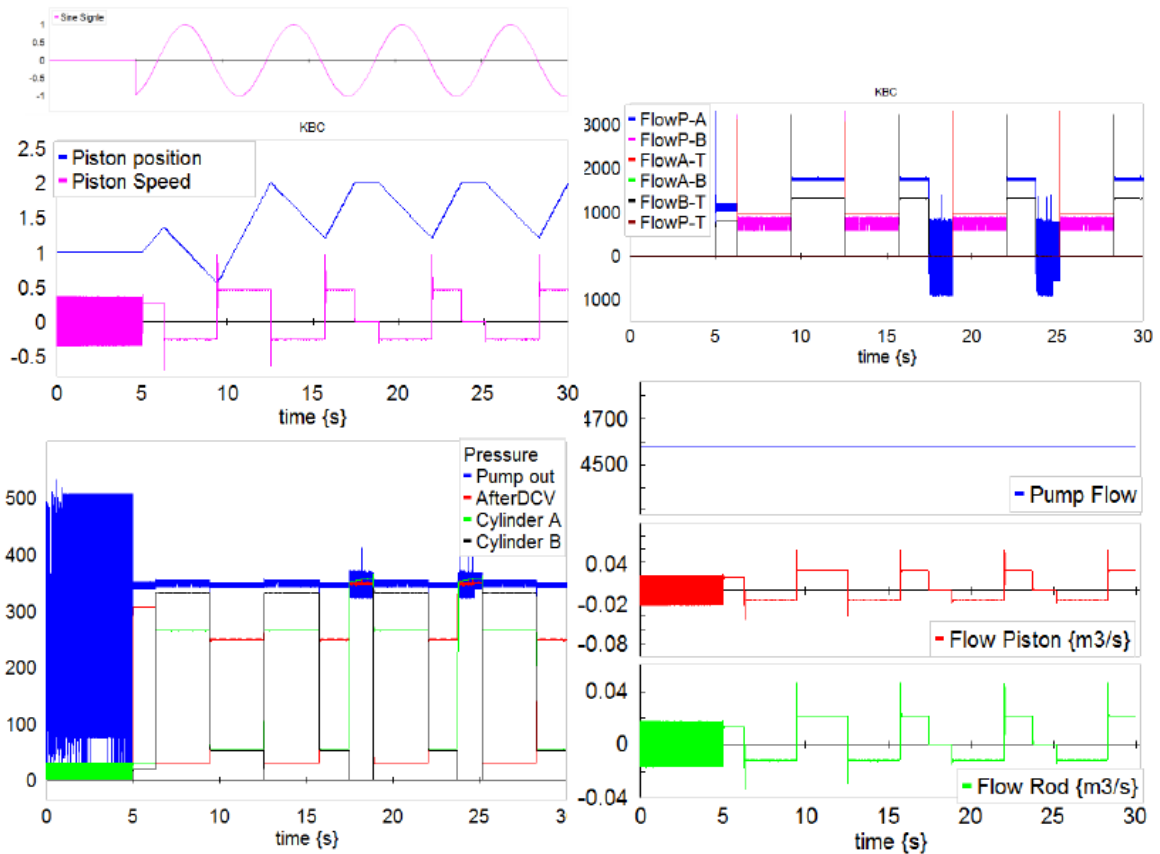


Fig. 5.4 The simulation result of the “Ideal model” system

During the entire running period, the flows and pressures are changing as a “pulse” from a value to another directly. The *Load* movement is showing as a linear function. The result is sort of perfect for the “Ideal model”, even there are some imperfect oscillations. However the performance of the “Ideal model” for a composition of a concept hydraulic system with no good result expectation is good enough.

5.3.Simulation result of “Standard model”

The “Standard model” is expected for much more detailed simulation result. At least, the result is able to show the main components’ behaviours, such as, the pressure compensation in pump, the opening presses of valves, etc.

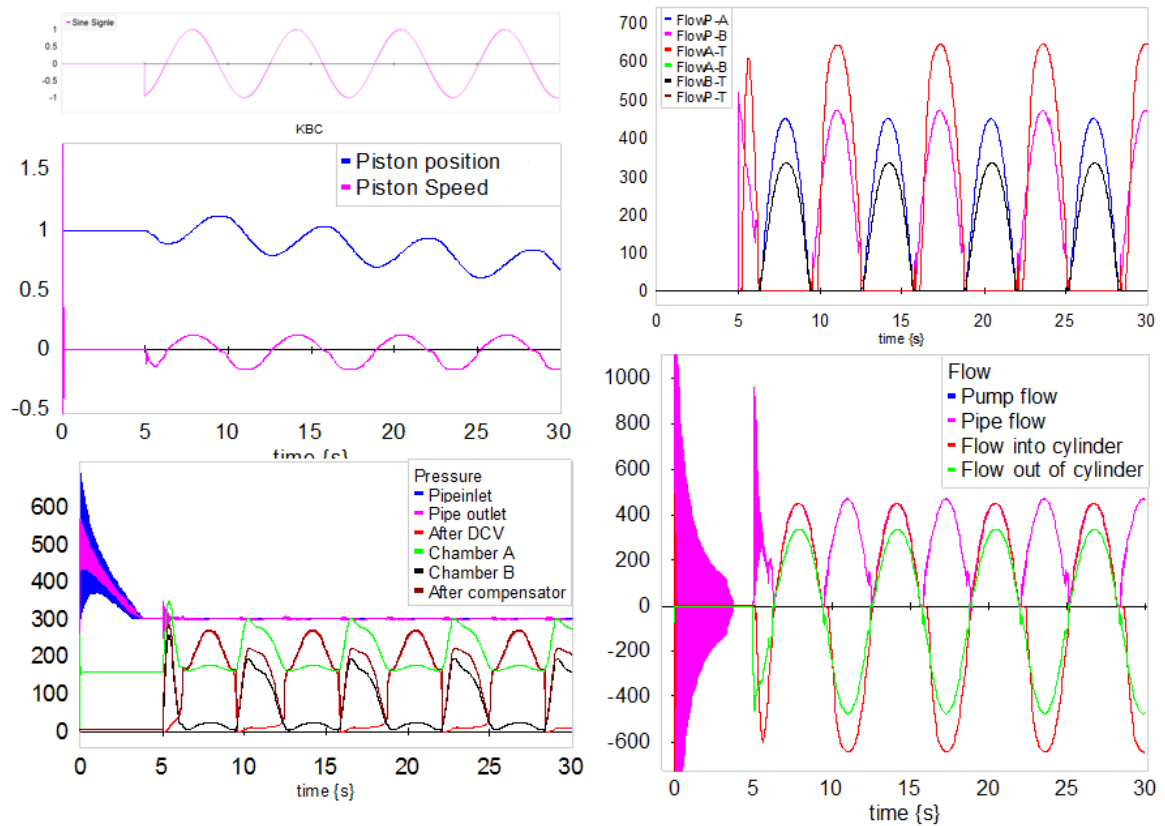
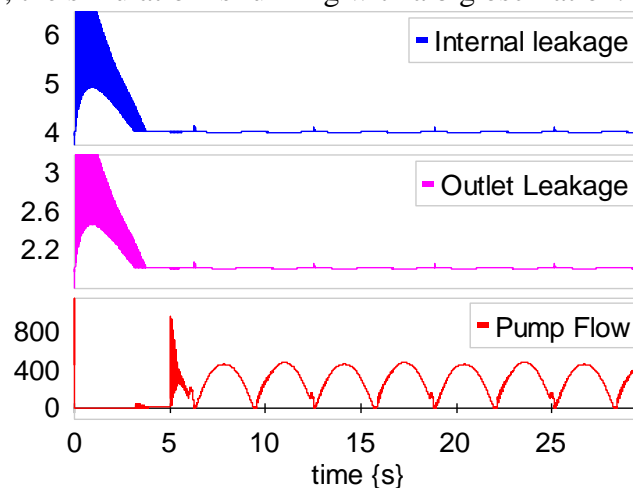


Fig. 5.5 The simulation result of the “Ideal model” system

Significantly, the simulation result of “Standard model” is much smoother than the result of the “Ideal model”. At the beginning, as the initial values are not fitted for the other parameters of system, the simulation is running with a big oscillation.



Furthermore, other important behaviours are modelled and can be measured such as the leakage of the pump.

5.4. Simulation result of “Advanced model”

The simulation results of the “Advanced model” are much dependent of the detail of real components, such as material and structure. The purpose of the “Advanced model” is for the modification during the design process and also for the analysis of some typical behaviours.

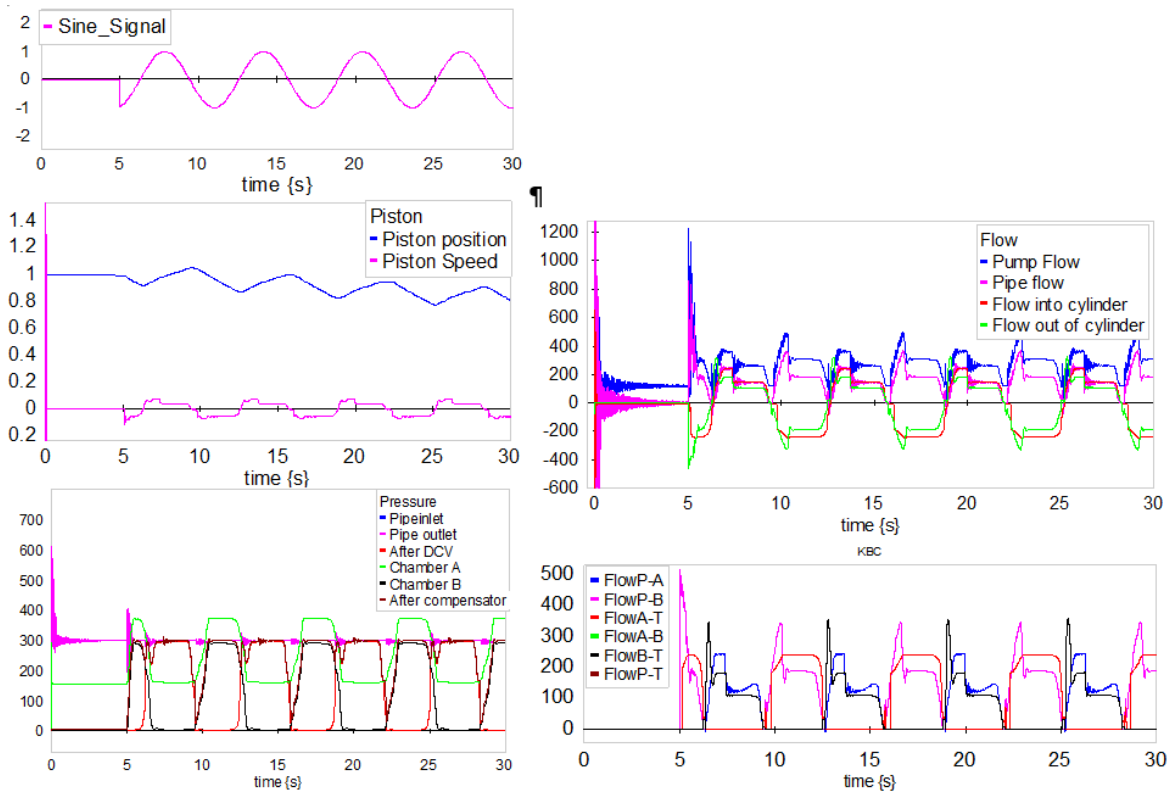


Fig. 5.6 The simulation result of the “Advanced model” system

From the plotted results, the “Advanced model” seems showing the more real simulation result close to the sensed result from the real system. However, the main objective in this project is focusing on the different complex level behaviour models set up, hence the real hydraulic system is needed to prove the accuracy of the simulation results of the “Advanced model”. There is one aspect in “Advance model” that is not complete in this project, which is the flow through orifices. In most orifice are not extremely thin but consist of a short tube as discussed in Chapter 3.2.2. Furthermore, the performance of cavitation is also a potential behaviour model can be setup for the modification phase. Anyway, “Advanced model” is successful to give a most complex behaviour model for the library.

6. Conclusion

The object-oriented hydraulic component models for component based modelling and simulation have been developed by using BG method in this project. The component model library includes the parts that been used for modelling of an offshore crane hydraulic system, such as hydraulic pump, pipe flow, direction control valve, counter-balance valve, cylinder, and hydraulic motor. Each component model is implemented in three different detail levels of modelling tailored to varying simulation purposes. At the beginning, the UML and Statechart method had been researched and adopted for the complex behaviours description, which succeeded to give a visual map for developing the different behaviour models. This library potentially provides designers an efficient, effective and flexible way to model and simulate for a hydraulic system. On the other hand, this library is flexible for the proposed virtual crane prototyping system which seeks to provide a heterogeneous simulation environment that allows for the overall product and system from as early as the concept design stage to detail design and analysis stage. In the case study, it is only a simple sub-system of a crane hydraulic system that shows small differences between each behaviour models. However, these component models have the potential to meet more complex hydraulic system which would show the efficiency, effectiveness and the flexibility of models. The models aim to provide a simulation result with effectively accuracy and efficient in reflecting the interested behaviours of the physical system. The development of the hydraulic model library is finally oriented to co-simulations using the virtual prototyping simulator based on the application of the Functional Mock-up Interface standard. Consequently, the main hydraulic component models are developed in library to support an efficient, effective and flexible simulation for a marine crane design. The behaviour classification has been done based on the OOM approach and Statechart description. However, there is still some further work that needs to be done to perfect the library, the last objective proposed at the beginning is willing to be done in further work. Meanwhile, the interfaces for users and co-simulation are very important for this library to be managed.

7. Further Work

The objectives in this project have been met and the main hydraulic components are modelled. However, there are still some fields of study which would be beneficial for researches and further works. Firstly, there is only one type of hydraulic component modelled in this project, every hydraulic component has a ‘family’ with many types not only from the function aspect but also the structure aspect. For instance, in the pump family (Fig. 6.1), only the axis piston pump is modelled in this project. In the future work, the hydraulic components library would include all types of the hydraulic components model for designers.

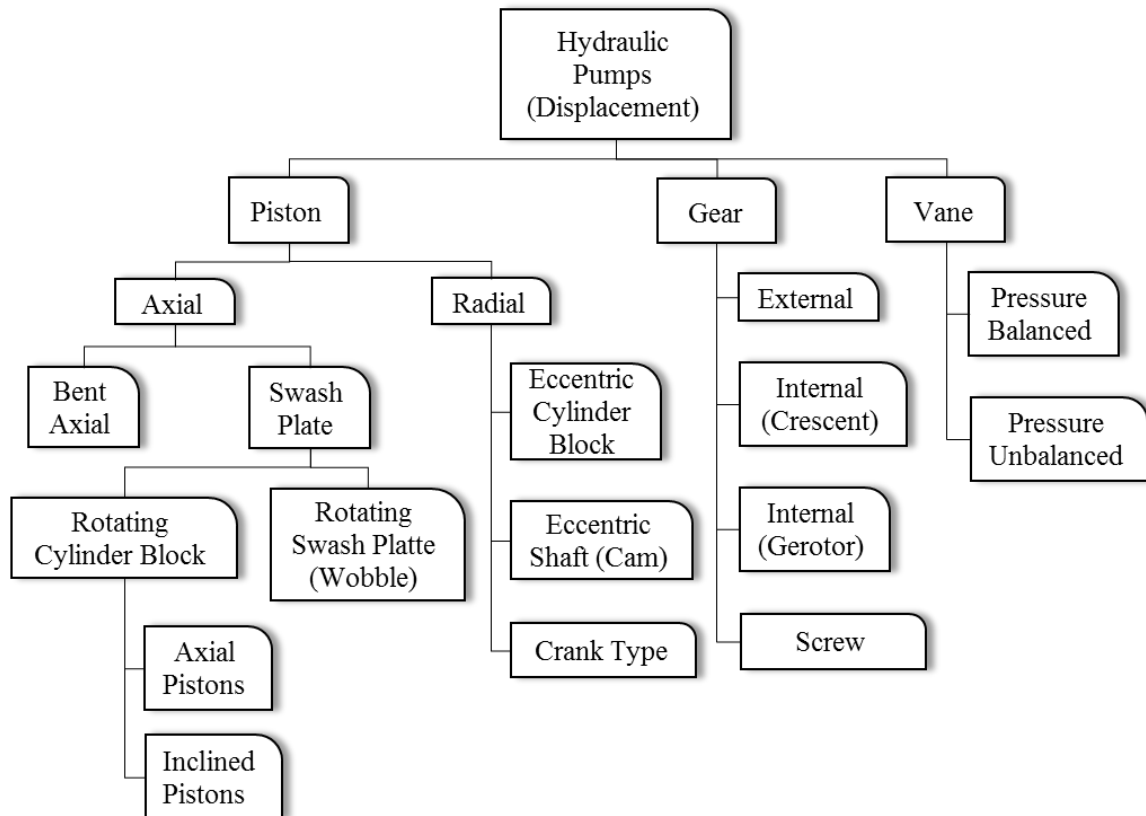


Fig. 7.1 The family tree of hydraulic pumps

Secondly, the hydraulic component library needs to be managed portable for designer to use. The hydraulic component models are already built in this project, however, the main parameters for the graphical structure of the model need to be defined by designers according to the real case. Therefore, the parameter interface needs to be designed. Specifically, part of the “Advanced model” which is modelled according to the hardware structures, while there are many computerised and mathematical formulas for a behaviour in one component model, hence, designers can chose any of the formulas in the user face for simulation during the design process of modification.

Finally, the methods for the behaviour complexity description is potential to use for other engineering fields, which could be discussed for further work. Meanwhile, the object-oriented component models with varying behaviour models is an advanced design not only for hydraulic field but also for other engineering fields serving for a virtual prototyping framework, which is becoming increasingly popular.

Reference

- [1] Eilif Pedersen / Hallvard Engja, Mathematical Modelling and Simulation of Physical Systems, 2014
- [2] J. A. Ferreira, F. Gomes Almeida, M. R. Quintas and J. P. Estima de Oliveira, Hybrid models for hardware-in-the-loop simulation of hydraulic systems, 2004.
- [3] Jorge A. Ferreira, João E. de Oliveira, Vítor A. Costa, Modelling of hydraulic systems for hardware-in-the-loop simulation: a methodology proposal,
- [4] Dragan H. Pusic, Novak N. Nedic, Object-Oriented Behavior Modeling and Simulation of Hydraulic Cylinder, 2006
- [5] David HAREL, STATECHARTS: A VISUAL FORMALISM FOR COMPLEX SYSTEMS, 1987
- [6] Deborah Hix and H. Rex Hatson, Developing user interfaces chapter 9 – Rapid prototyping of interaction design pages 249-281. Wiley professional computing, New York, USA. And others. 1993.
- [7] M. Galal Rabie, Fluid Power Engineering, 2009.
- [8] Syllabus M.C.A. (Semester IV), Object Oriented Modelling and Design using UML.
- [9] Peter Fritzson, Vadim Engelson, Modelica - A Unified Object-Oriented Language for System Modelling and Simulation, 1997.
- [10] LUBRIPLATE Marine-Safe Hydraulic Oils 32, 46 & 68.
- [11] W. Borutzky, Bond graph modeling from an object oriented modeling point of view, 1999.
- [12] Hubertus Tummescheit, Design and Implementation of Object-Oriented Model Libraries using Modelica, 2002.
- [13] Tutorial for the Hydraulics Library, Modeling of Hydraulic Systems, 2013
- [14] Alfred Theorin, Charlotta Johnsson, On Extending JGrafchart with Support for FMI for Co-Simulation, 2014.
- [15] Prof. Dr.-Ing. Peter Beater, Object-oriented Modeling and Simulation of Hydraulic Drives, 1998.
- [16] J. A. Ferreira*, F. Gomes de Almeida** and M. R. Quintas**, Semi-empirical model for a hydraulic servo-solenoid valve.
- [17] Ana Luísa Ramos, Member, IEEE, José Vasconcelos Ferreira, and Jaume Barceló, Model-Based Systems Engineering: An Emerging Approach for Modern Systems, 2012.
- [18] Yingguang Chu, Virtual Prototyping for Maritime Crane Design and Operations.
- [19] C. Canudas de Wit, H. Olsson, K. J. Åström, P. Lischinsky, A new model for control of systems with friction, 1995.
- [20] Bosch Rexroth Oil Control S.p.A. Counter Balance Valve
- [21] Morten Kollerup Bak, Michael Rygaard Hansen, Analysis of Offshore Knuckle Boom Crane-Part One: Modelling and Parameter Identification
- [22] Yu Li, IP501709-Product-and system design-Design of Parallel linked crane for offshore lifting, 2015
- [23] Rexroth Bosch Group, Hydraulics. Basic Principles and Components, 2004

Appendix

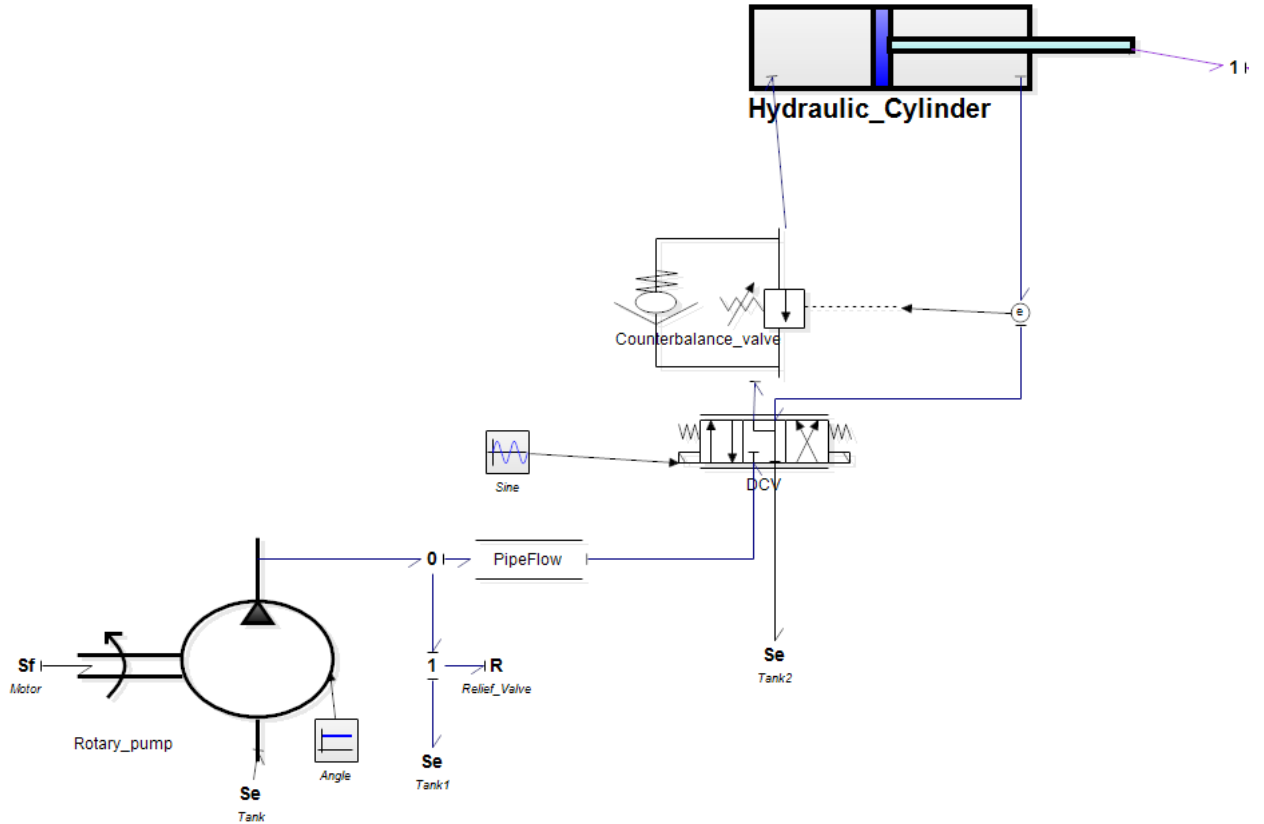
Appendix A: 20sim code of “Ideal model”

Appendix B: 20sim code of “Standard model”

Appendix C: 20sim code of “Advanced model”

model

implementation:



interface:

type Mainmodel

information:

name	model
Version	4.6
IsMainModel	0
KeepParameterValues	False
LibraryPath	Hydraulic components\KBC Ideal.emx
TimeStamp	2016-6-1 16:49:52

Angle

implementation:

equations
`output = pi/4;`

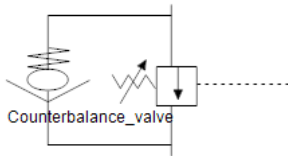
interface:

type	Constant																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>output</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	output	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description										
output	signal	real	1	out	1												

information:

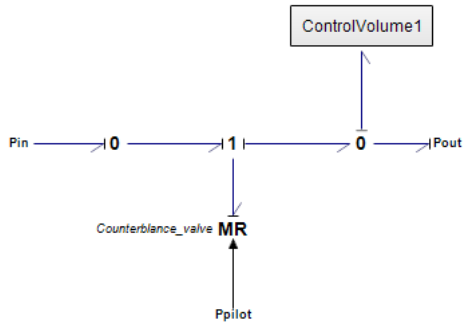
name	Angle
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Signal\Sources\Constant.emx
TimeStamp	2007-10-19 14:48:44
AllowLibraryUpdate	True

name DocumentationMask



Counterbalance_valve (Ideal)

implementation:



interface:

type		Submodel						
ports	name	domain	type	terminals	orientation	size	unit	description
	Pin	hydraulic	real	1	in	1		
	Pout	power	real	1	out	1		
	Ppilot	signal	real	1	in	1		

information:

name	Counterbalance_valve
implementation	Ideal
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

ControlVolume1

ControlVolume1 (contrl_volume1)

implementation:

```
// Control volume
parameters
  real bulkmodulus= 1600000000;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(cv.phi,Qinit);

  if Q>0.0 then
    cv.p = bulkmodulus/ControlVolume*Q;
  else
    cv.p = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = cv.p/1e5;
```

interface:

type		Submodel						
ports	name	domain	type	terminals	orientation	size	unit	description
	cv	hydraulic	real	1	in	1		

information:

name	Counterbalance_valve.ControlVolume1
implementation	contrl_volume1
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

MR Counterbalance_valve

implementation:

```

parameters
  real density = 861.8;
//Counterbalance valve settings
  real Qcbv=140.0, dPcbv=5.0e5;
  real Popening = 25e5;

variables
  real hidden FlowArea;
  real flow;
  real hidden hoistflow;
  real hidden slackflow;

initialequations
//Maximum flow area for checkvalve
  FlowArea = Qcbv / (60e3) / (2*dPcbv/density)^0.5;

equations
  if p.p>0.0 then
//Flow through checkvalve in hoist direction
    hoistflow = FlowArea*(2*abs(p.p)/density)^0.5*sign(p.p);
  else
    hoistflow = 0;
  end;

  if Ppilot<Popening then
    slackflow = 0;
  else
    slackflow = FlowArea*(2*abs(p.p)/density)^0.5*sign(p.p);
  end;

  p.f = hoistflow + slackflow;

//Calculating flow in l/min for plotting purposes
  flow = p.f*60.0e3;

```

interface:

type	MR																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> <tr><td>Ppilot</td><td>signal</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	1	in	1			Ppilot	signal	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	1	in	1																				
Ppilot	signal	real	1	in	1																				

information:

name	Counterbalance_valve.Counterbalance_valve
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\MR.emx
TimeStamp	2011-11-29 16:09:29
AllowLibraryUpdate	True
name	DocumentationMask

1 OneJunction14

implementation:

```

equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);

```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>any</td><td>none</td><td>1</td><td></td><td></td></tr> <tr><td>flow</td><td>signal</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
flow	signal	real	1	out	1																				

restrictions	kind	priority	type	ports
	causality	constraint	one_out	p

information:

name	Counterbalance_valve.OneJunction14
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

0 ZeroJunction

implementation:

```
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	Counterbalance_valve.ZeroJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction1

implementation:

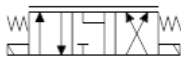
```
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

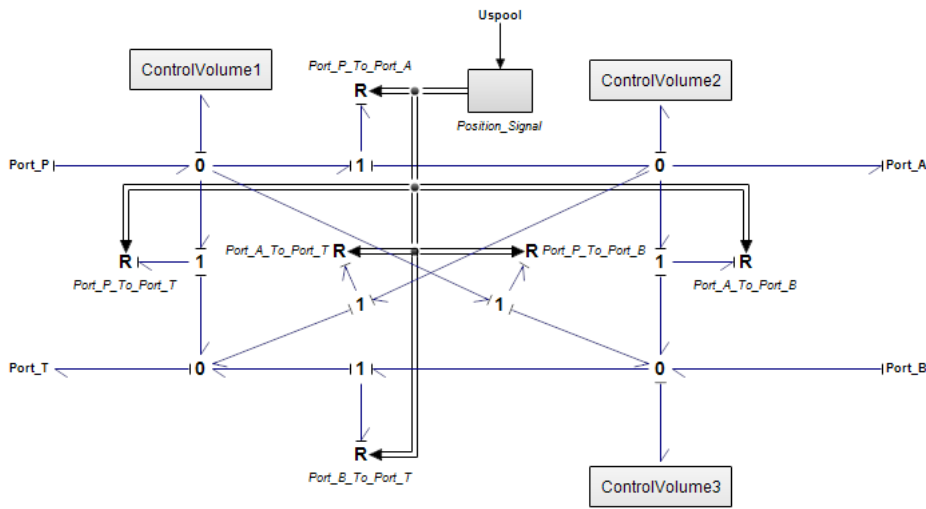
name	Counterbalance_valve.ZeroJunction1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16



DCV

DCV (ideal)

implementation:



interface:

type		Submodel						
ports	name	domain	type	terminals	orientation	size	unit	description
	Port_P	power	real	1	in	1		
	Port_T	power	real	1	out	1		
	Port_A	power	real	1	out	1		
	Port_B	power	real	1	in	1		
	Uspool	signal	real	1	in	1		

information:

name	DCV
implementation	ideal
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask
GlobalRelations	parameters real global density= 861.8; real global bulkmodulus=1.6e9; real global Pinit= 0; real global deadband = 0.01; real global Qmax flow in l/min real global dPQmax = 500000; //Pressure differential at maximum flow//Accoding to the function of Nuetral position to choos global C1= 0, C2= 0, C3= 1, C4= 1, C5= 0,C6=0.0; real global Rsn= 0.01; //the restriction of orifice area dring drain oil to tank

ControlVolume1 (contrl_volume1)

implementation:

```
// Control volume
parameters
  real bulkmodulus= 1600000000;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(cv.phi,Qinit);

  if Q>0.0 then
    cv.p = bulkmodulus/ControlVolume*Q;
  else
    cv.p = 0.0; // No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = cv.p/1e5;
```

interface:

type		Submodel						
ports	name	domain	type	terminals	orientation	size	unit	description
	cv	hydraulic	real	1	in	1		

information:

name	DCV.ControlVolume1
implementation	contrl_volume1
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

ControlVolume2 (contrl_volume1)

implementation:

```
// Control volume
parameters
  real bulkmodulus= 1600000000;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(cv.phi,Qinit);

  if Q>0.0 then
    cv.p = bulkmodulus/ControlVolume*Q;
  else
    cv.p = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = cv.p/1e5;
```

interface:

type	Submodel																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>cv</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	cv	hydraulic	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
cv	hydraulic	real	1	in	1												

information:

name	DCV.ControlVolume2
implementation	contrl_volume1
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

ControlVolume3 (contrl_volume1)

implementation:

```
// Control volume
parameters
  real bulkmodulus= 1600000000;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(cv.phi,Qinit);

  if Q>0.0 then
    cv.p = bulkmodulus/ControlVolume*Q;
  else
    cv.p = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
```

```
Pressure = cv.p/1e5;
```

interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	cv	hydraulic	real	1	in	1		

information:

name	DCV.ControlVolume3
implementation	contrl_volume1
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

1 OneJunction**implementation:**

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction1**implementation:**

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction5

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction5
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction6

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction6
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction7

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction7
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction8

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction																														
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>any</td><td>none</td><td>1</td><td></td><td></td></tr> <tr><td>flow</td><td>signal</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																								
p	hydraulic	real	any	none	1																										
flow	signal	real	1	out	1																										
restrictions	kind	priority	type	ports																											
	causality	constraint	one_out	p																											

information:

name	DCV.OneJunction8
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

R Port_A_To_Port_B

implementation:

```
parameters
  real global density;

variables
  real flowAB;

equations
  // Flow from A to B
  p.f = FlowArea[6,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

  //Calculating flow in l/min for plotting purposes
  flowAB = p.f*60.0e3;
```

interface:

type	R																														
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>power</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> <tr><td>FlowArea</td><td>signal</td><td>real</td><td>1</td><td>in</td><td>[6,1]</td><td></td><td></td></tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1			FlowArea	signal	real	1	in	[6,1]		
name	domain	type	terminals	orientation	size	unit	description																								
p	power	real	1	in	1																										
FlowArea	signal	real	1	in	[6,1]																										

information:

name	DCV.Port_A_To_Port_B
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_A_To_Port_T

implementation:

```
parameters
  real global density;

variables
```

```

real flowAT;

equations
// Flow from A to T
p.f = FlowArea[3,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
flowAT = p.f*60.0e3;

```

interface:

type R							
ports	name	domain	type	terminals	orientation	size	unit description
	p	power	real	1	in	1	
	FlowArea	signal	real	1	in	[6,1]	

information:

name	DCV.Port_A_To_Port_T
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_B_To_Port_T**implementation:**

```

parameters
real global density;

variables
real flowBT;

equations
// Flow from B to T
p.f = FlowArea[4,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
flowBT = p.f*60.0e3;

```

interface:

type R							
ports	name	domain	type	terminals	orientation	size	unit description
	p	power	real	1	in	1	
	FlowArea	signal	real	1	in	[6,1]	

information:

name	DCV.Port_B_To_Port_T
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_P_To_Port_A**implementation:**

```

parameters
real global density;

variables
real flowPA;

equations
// Flow from P to A
p.f = FlowArea[1,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
flowPA = p.f*60.0e3;

```

interface:

type R							
--------	--	--	--	--	--	--	--

ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		
	FlowArea	signal	real	1	in	[6,1]		

information:

name	DCV.Port_P_To_Port_A
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_P_To_Port_B**implementation:**

```

parameters
  real global density;

variables
  real flowPB;

equations
// Flow from P to B
  p.f = FlowArea[2,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
  flowPB = p.f*60.0e3;

```

interface:

ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		
	FlowArea	signal	real	1	in	[6,1]		

information:

name	DCV.Port_P_To_Port_B
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_P_To_Port_T**implementation:**

```

parameters
  real global density;

variables
  real flowPT;

equations
// Flow from P to T
  p.f = FlowArea[5,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
  flowPT = p.f*60.0e3;

```

interface:

ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		
	FlowArea	signal	real	1	in	[6,1]		

information:

name	DCV.Port_P_To_Port_T
Version	4.2

IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\I.R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

Position_Signal

implementation:

```

parameters
  real global density;
  real global Qmax ; //Maximum flow in l/min
  real global dPQmax ; //Pressure differential at maximum flow
  real global Rsn; //the restriction of orifice area dring drain oil to tank
  real global C1, C2, C3, C4, C5, C6;

variables
  real hidden Area; //Valve flow area
  real hidden AreaPA, AreaPB, AreaAT, AreaBT, AreaPT, AreaAB;

initialequations
  // Calculating maximum flow area
  Area = Qmax/(60e3)/(2*dPQmax/density)^0.5;

equations

  // Flow to tank within deadband for open center valve NOTE: DISABLED!

  if Uspool<0 then
  //Hoist direction
    AreaPA = Area;
    AreaPB = 0.0;
    AreaAT = 0.0;
    AreaBT = AreaPA;
    AreaPT = 0.0;
    AreaAB = 0.0;
  else
  //Slack direction
  if Uspool>0 then
    AreaPA = 0.0;
    AreaPB = Area;
    AreaAT = AreaPB;
    AreaBT = 0.0;
    AreaPT = 0.0;
    AreaAB = 0.0;
  else
    AreaPA = C1*Rsn*Area;
    AreaPB = C2*Rsn*Area;
    AreaAT = C3*Rsn*Area;
    AreaBT = C4*Rsn*Area;
    AreaPT = C5*Rsn*Area;
    AreaAB = C6*Rsn*Area;
  end;
end;

FlowArea = [AreaPA;AreaPB;AreaAT;AreaBT;AreaPT;AreaAB];

```

interface:

type FlowArea								
ports	name	domain	type	terminals	orientation	size	unit	description
	Uspool	signal	real	1	in	1		
	FlowArea	signal	real	1	out	[6,1]		

information:

name	DCV.Position_Signal
Version	4.0
LibraryPath	C:\Program Files\20-sim 4.0\System\Submodel.emx
TimeStamp	2007-10-31 11:32:54
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask

• Splitter1

implementation:

```
equations
```

```
collect (output) = input;
```

interface:

type		Splitter						
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[6,1]		
	input	signal	real	1	in	[6,1]		

information:

name	DCV.Splitter1
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter2

implementation:

```
equations
collect (output) = input;
```

interface:

type		Splitter						
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[6,1]		
	input	signal	real	1	in	[6,1]		

information:

name	DCV.Splitter2
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter3

implementation:

```
equations
collect (output) = input;
```

interface:

type		Splitter						
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[6,1]		
	input	signal	real	1	in	[6,1]		

information:

name	DCV.Splitter3
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

0 ZeroJunction3

implementation:

```
equations
sum (direct (p.f)) = 0;
equal (collect (p.e));
effort = first (p.e);
```

interface:

type		ZeroJunction						

ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		

restrictions	kind	priority	type	ports
	causality	constraint	one_in	p

information:

name	DCV.ZeroJunction3
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction4**implementation:**

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);

```

interface:

ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		

restrictions	kind	priority	type	ports
	causality	constraint	one_in	p

information:

name	DCV.ZeroJunction4
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction5**implementation:**

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);

```

interface:

ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		

restrictions	kind	priority	type	ports
	causality	constraint	one_in	p

information:

name	DCV.ZeroJunction5
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction6**implementation:**

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
  
```

interface:

type	ZeroJunction						
ports	name	domain	type	terminals	orientation	size	unit description
	p	hydraulic	real	any	none	1	
	effort	signal	real	1	out	1	
restrictions	kind	priority	type	ports			
	causality	constraint	one_in	p			

information:

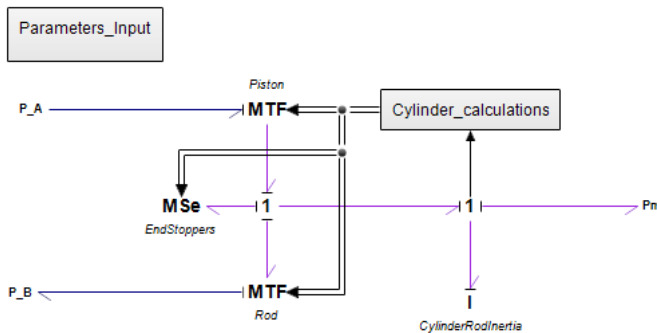
name	DCV.ZeroJunction6
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16



Hydraulic_Cylinder

Hydraulic_Cylinder (Ideal)

implementation:



interface:

type	Submodel						
ports	name	domain	type	terminals	orientation	size	unit description
	P_A	hydraulic	real	1	in	1	
	P_m	mechanical	real	1	out	1	
	P_B	hydraulic	real	1	out	1	

information:

name	Hydraulic_Cylinder
implementation	Ideal
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask
name	ScriptFilename
string	D:\Master Thesis\picture\cylinder.png
name	MaskChoice
int	1
name	BitmapFilename
string	D:\Master Thesis\picture\cylinder.png

I CylinderRodInertia

implementation:

```

parameters
  real global Rodmass; //Mass of cylinder rod and other translating (motion) inertias
variables
  real Rodmomentum, Rodspeed, Rodposition;
  real rodpos_Ideal;
equations
  //Integrating force to find momentum
  Rodmomentum = int(p.F);
  //Finding speed based on momentum
  p.v = Rodmomentum/Rodmass;
  Rodspeed = p.v;
  //Integrating speed to find position of rod
  Rodposition = int(Rodspeed,1);
  rodpos_Ideal=Rodposition;

```

interface:

type	I																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>mechanical</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	mechanical	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	mechanical	real	1	in	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>preferred</td> <td>in</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	preferred	in	p								
kind	priority	type	ports														
causality	preferred	in	p														

information:

name	Hydraulic_Cylinder.CylinderRodInertia
LibraryPath	Bond Graph\I.emx
TimeStamp	2011-11-29 15:55:55
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask

Cylinder_calculations

implementation:

```

// This is a submodel making calculations for a hydraulic cylinder.
// The model use rod position as input, and calculates output values
// such as flow area for inlet and outlet ports, initial fluid volume
// and current physical volume in chamber A and B,piston and rod area
// and force on cylinder piston/rod when reaching either of the end positions.
// The results are sent to other parts of the model in an signal array.
// Cylcalc = [Apist;Arod;Fbumper];

parameters
  // Defining initial conditions, rod position and pressures
  real global rodposinit, PAinit, PBinit;
  // Defining port dimensions, diameters, stroke and deadvolume for the cylinder
  real global dinlet, doutlet, dctl, drod, Stroke; // vdead;
  // Defining maximum pressure, deflection at maximum pressure and damping ratio at end strokes
  real global Pmax, deflection, dampingratio; //sequeece the end stopper to 0.0001m

variables
  // real Ainlet, Aoutlet;
  real hidden Apist, Arod;
  real rodpos_Ideal;
  // real VAinit, VBinit, VAoilinit, VBoilinit, VA, VB;
  real hidden kbumper, cbumper, Fbumper;

initialequations
  // Calculating area on piston and rod side
  Apist = 0.25*pi*dctl^2;
  Arod = Apist-0.25*pi*drod^2;
  // Calculating stiffness and damping for bumper
  kbumper = Pmax*Apist/deflection;
  cbumper = dampingratio*2*sqrt(Pmax*Apist/9.81*kbumper);

equations
  // Calculating rod position
  rodpos_Ideal = int(rodspeed,rodposinit);
  // Checking if minimum or maximum stroke has been reached
  if rodpos_Ideal> Stroke then
    Fbumper = kbumper*(rodpos_Ideal-Stroke) + cbumper*max([0,rodspeed]); //Maximum stroke reached
  else
    if rodpos_Ideal< 0 then
      Fbumper = kbumper*(rodpos_Ideal) + cbumper*min([0,rodspeed]); // Minimum stroke reached
    else
      Fbumper = 0.0; // Normal stroke - no bumper forces
    end;
  end;
end;

```

```
// Calculation results in a matrix
Cylcalc = [Apist;Arod;Fbumper];
```

interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	rodspeed	signal	real	1	in	1		
	Cylcalc	signal	real	1	out	[3,1]		

information:

name	Hydraulic_Cylinder.Cylinder_calculations
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

MSe EndStoppers**implementation:**

```
// Force when cylinder reaches maximum or minimum stroke (Damper and spring)
equations
  p.F = Cylcalc[3,1];
```

interface:

type	MSe							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	1	in	1		
	Cylcalc	signal	real	1	in	[3,1]		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Hydraulic_Cylinder.EndStoppers
LibraryPath	Bond Graph\MSe.emx
TimeStamp	2011-11-29 16:12:33
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

1 OneJunction13**implementation:**

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description

	p	mechanical	real	any	none	1
	flow	signal	real	1	out	1
restrictions	kind	priority	type	ports		
	causality	constraint	one_out	p		

information:

name	Hydraulic_Cylinder.OneJunction13
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction3**implementation:**

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Hydraulic_Cylinder.OneJunction3
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2007-9-27 9:51:18

Parameters_Input

Parameters_Input**implementation:**

```
parameters
// real global bulkmodulus = 1600000000;
// Defining initial conditions, rod position and pressures
real global rodposinit = 1;
real global PAinit= 5e5;
real global PBinit= 1e5;
// Defining port dimensions, diameters, stroke and deadvolume for the cylinder
real global dcyll = 283e-3;
real global drod = 141e-3;
real global Stroke = 2;
// Defining maximum pressure, deflection at maximum pressure and damping ratio at end strokes
real global Pmax = 35000000;
real global deflection = 0.001;
real global dampingratio= 0.1; //sequeece the end stopper to 0.0001m
// Defining the mass of Cylinder Rod
real global Rodmass= 500;
```

interface:

type Submodel

information:

name	Hydraulic_Cylinder.Parameters_Input
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1

AllowLibraryUpdate	False
name	DocumentationMask

MTF Piston

implementation:

```
// Piston side: Relation between pressure/flow and force/speed, i.e. piston area
variables
  real flow,Pressure_Ideal;
equations
  p1.p = p2.F/Cylcalc[1,1];
  p2.v = p1.phi/Cylcalc[1,1];
  flow =p1.phi*60e3;
  Pressure_Ideal=p1.p*1e-5;
```

interface:

type	MTF							
ports	name	domain	type	terminals	orientation	size	unit	description
	p1	hydraulic	real	1	in	1		
	p2	mechanical	real	1	out	1		
	Cylcalc	signal	real	1	in	[3,1]		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	p1, p2				

information:

name	Hydraulic_Cylinder.Piston
LibraryPath	Bond Graph\MTF.emx
TimeStamp	2011-11-29 16:15:31
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

MTF Rod

implementation:

```
// Piston side: Relation between pressure/flow and force/speed, i.e. piston area minus rod area
variables
  real flow,Pressure_Ideal;
equations
  p1.F = p2.p*Cylcalc[2,1];
  p2.phi = p1.v*Cylcalc[2,1];
  flow=p2.phi*60e3;
  Pressure_Ideal=p2.p*1e-5;
```

interface:

type	MTF							
ports	name	domain	type	terminals	orientation	size	unit	description
	p1	mechanical	real	1	in	1		
	p2	hydraulic	real	1	out	1		
	Cylcalc	signal	real	1	in	[3,1]		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	p1, p2				

information:

name	Hydraulic_Cylinder.Rod
LibraryPath	Bond Graph\MTF.emx
TimeStamp	2011-11-29 16:15:31
Version	4.2
IsMainModel	1
KeepParameterValues	False

AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

• Splitter1

implementation:

```
equations
collect (output) = input;
```

interface:

type Splitter								
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[3,1]		
	input	signal	real	1	in	[3,1]		

information:

name	Hydraulic_Cylinder.Splitter1
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter2

implementation:

```
equations
collect (output) = input;
```

interface:

type Splitter								
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[3,1]		
	input	signal	real	1	in	[3,1]		

information:

name	Hydraulic_Cylinder.Splitter2
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

Sf Motor

implementation:

```
parameters
  real flow = 1776;
equations
  p.f = flow;
```

interface:

type Sf								
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	fixed	in	p				

information:

name	Motor
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Sf.emx
TimeStamp	2011-11-29 16:38:03
AllowLibraryUpdate	True
name	DocumentationMask

1 OneJunction

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>any</td><td>none</td><td>1</td><td></td><td></td></tr> <tr><td>flow</td><td>signal</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
flow	signal	real	1	out	1																				
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>constraint</td><td>one_out</td><td>p</td></tr> </tbody> </table>	kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																						
causality	constraint	one_out	p																						

information:

name	OneJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction1

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>mechanical</td><td>real</td><td>any</td><td>none</td><td>1</td><td></td><td></td></tr> <tr><td>flow</td><td>signal</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	mechanical	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	mechanical	real	any	none	1																				
flow	signal	real	1	out	1																				
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>constraint</td><td>one_out</td><td>p</td></tr> </tbody> </table>	kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																						
causality	constraint	one_out	p																						

information:

name	OneJunction1
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2007-9-27 9:51:18

Se Payload

implementation:

```
parameters
  real effort = -98100;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;
```

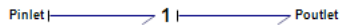

interface:

type	Se							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Payload
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

PipeFlow

PipeFlow (Ideal)**implementation:****interface:**

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	Pinlet	hydraulic	real	1	in	1		
	Poutlet	hydraulic	real	1	out	1		

information:

name	PipeFlow
implementation	Ideal
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

1 OneJunction**implementation:**

```

equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
  
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	PipeFlow.OneJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

ⓔ Pressure_sensor

implementation:

```

variables
  real Pressure;
equations
  p2.e = p1.e;
  p1.f = p2.f;
  effort = p1.e;
  Pressure=effort*1e-5;

```

interface:

type	EffortSensor							
ports	name	domain	type	terminals	orientation	size	unit	description
	p1	hydraulic	real	1	in	1		
	p2	hydraulic	real	1	out	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	p1, p2				

information:

name	Pressure_sensor
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\EffortSensor.emx
TimeStamp	2011-11-29 15:44:34
AllowLibraryUpdate	True
name	DocumentationMask

Ⓡ Relief_Valve

implementation:

```

parameters
  real D=30e-3;
  real Pset=350e5;
  real density=861.8;

variables
  real hidden FlowArea;
  real Flow;
initialequations
  FlowArea=pi*D^2/4;
equations

if p.e> Pset then
  p.f=FlowArea*(2*abs(p.e)/density)^0.5*sign(p.e);
else
  p.f=0;
end;
Flow=p.f*60e3;

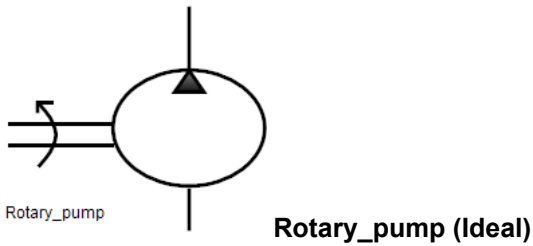
```

interface:

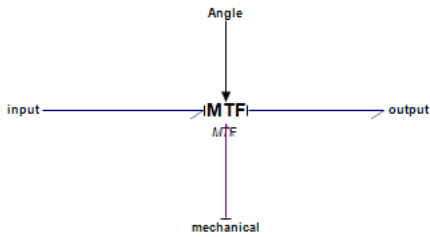
type	R							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		

information:

name	Relief_Valve
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask



implementation:



interface:

type Submodel							
ports	name	domain	type	terminals	orientation	size	unit description
	input	power	real	1	in	1	
	output	power	real	1	out	1	
	Angle	signal	real	1	in	1	
	mechanical	power	real	1	in	1	

information:

name	Rotary_pump
implementation	Ideal
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

MTF MTF

implementation:

```

parameters
  real A= 0.000675; //piston crosssectional area m^2
  real n= 8; //number of pistons
  real D= 0.05; //pitch diameter m
variables
  real Pressure, Flow;
equations
  P_motor.T = A*n*D/(2*pi)*(P_out.p-P_in.p)*tan(Angle);
  //flow rate of pump which A*n*D/(2*pi) is the pump displacement per radian(m^3/rad)
  P_out.phi = A*n*D/(2*pi)*P_motor.omega*tan(Angle);
  P_in.phi = P_out.phi;
  Pressure =P_out.p*1e-5;
  Flow = P_out.phi*60e3;
    
```

interface:

type MTF							
ports	name	domain	type	terminals	orientation	size	unit description
	P_motor	rotation	real	1	in	1	
	P_out	hydraulic	real	1	out	1	
	Angle	signal	real	1	in	1	
	P_in	hydraulic	real	1	in	1	
restrictions	kind	priority	type	ports			
	causality	constraint	not_equal	P_motor, P_out			

information:

name	Rotary_pump.MTF
Version	4.2
IsMainModel	1

KeepParameterValues	False
LibraryPath	Bond Graph\MTF.emx
TimeStamp	2011-11-29 16:15:31



Sine

implementation:

```

parameters
  real amplitude = 1.0;           // amplitude of the wave
  real omega = 1.0 {rad/s};      // angular frequency of the wave
  real motionstart=5;
variables
  boolean hidden change;
  real hidden half;
equations
  "calculate at least 2 points per cycle to get a triangle"
  half = pi / omega;
  change = frequencyevent (half, half / 2);
  if time < motionstart then
  output=0;
  else

  "calculate the sine wave"
  output = amplitude * sin ( omega * time);
  end;

```

interface:

type	WaveGenerator							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	1	out	1		

information:

name	Sine
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Signal\Sources\WaveGenerator-Sine.emx
TimeStamp	2007-9-27 16:12:6

Se Tank

implementation:

```

parameters
  real effort = 0;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

interface:

type	Se							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Tank
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

Se Tank1

implementation:

```

parameters
  real effort = 0;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

interface:

type	Se							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Tank1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

Se Tank2**implementation:**

```

parameters
  real effort = 0;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

interface:

type	Se							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Tank2
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

0 ZeroJunction**implementation:**

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);

```

interface:

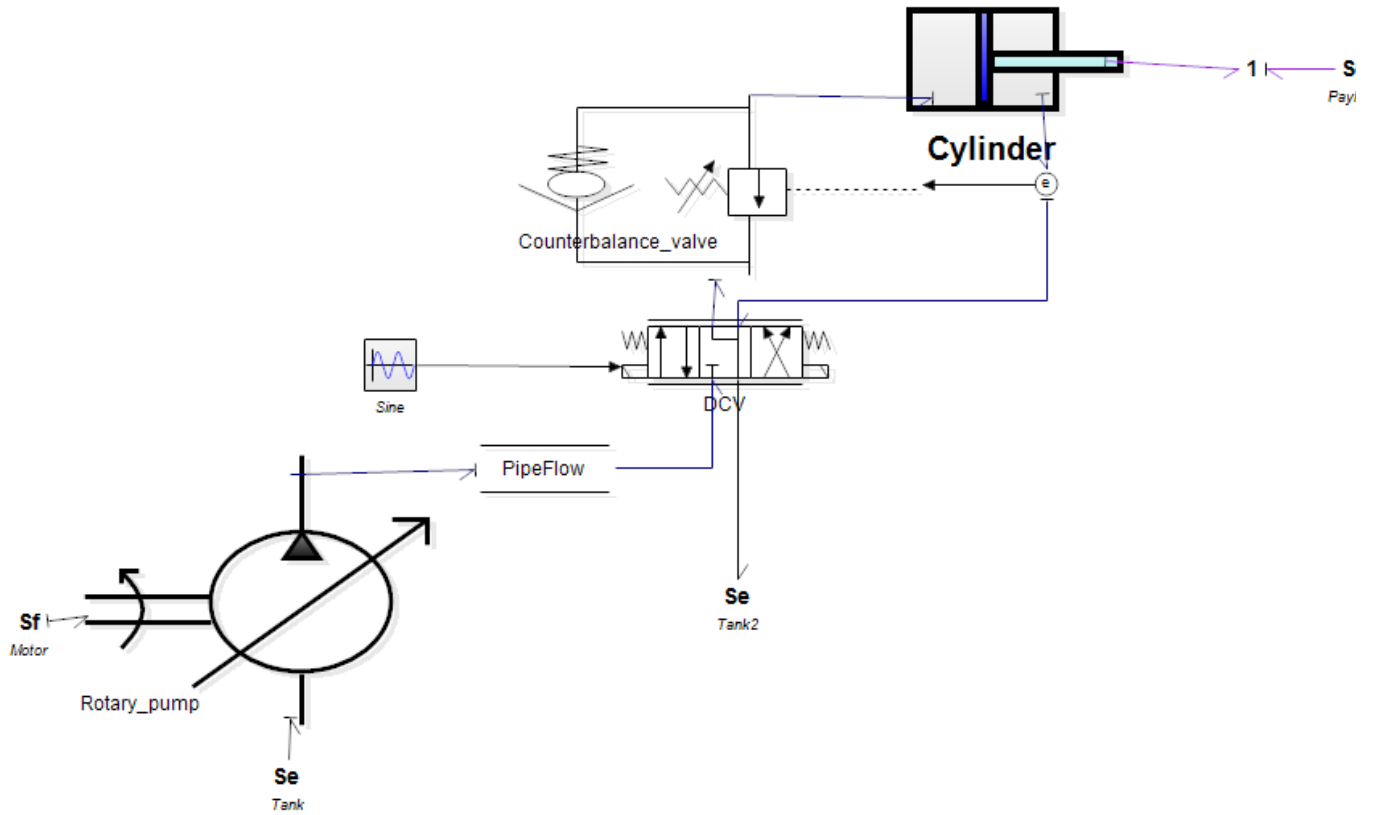
type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	ZeroJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

model **model**

implementation:

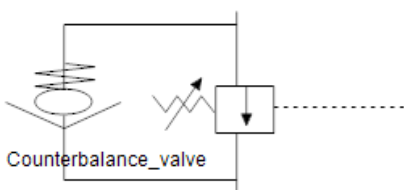


interface:

type Mainmodel

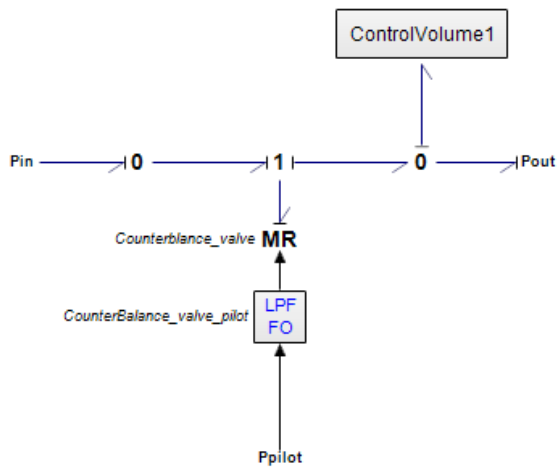
information:

name	model
Version	4.6
IsMainModel	0
KeepParameterValues	False
LibraryPath	Hydraulic components\KBC Standard.emx
TimeStamp	2016-5-30 13:09:30



Counterbalance_valve (Standard)

implementation:

**interface:**

type Submodel							
ports	name	domain	type	terminals	orientation	size	unit description
	Pin	hydraulic	real	1	in	1	
	Pout	power	real	1	out	1	
	Ppilot	signal	real	1	in	1	

information:

name	Counterbalance_valve
implementation	Standard
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

ControlVolume1

ControlVolume1 (contrl_volume1)**implementation:**

```
// Control volume
parameters
  real bulkmodulus= 1.6e9;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(cv.phi,Qinit);

  if Q>0.0 then
    cv.p = bulkmodulus/ControlVolume*Q;
  else
    cv.p = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = cv.p/1e5;
```

interface:

type Submodel							
---------------	--	--	--	--	--	--	--

ports	name	domain	type	terminals	orientation	size	unit	description
	cv	hydraulic	real	1	in	1		

information:

name	Counterbalance_valve.ControlVolume1
implementation	contrl_volume1
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

LPF
FO

CounterBalance_valve_pilot

implementation:

```
parameters
  real BW = 0.4 {Hz};          // Bandwidth
variables
  real rate, state, w;
equations
  w = 2*3.1415926536*BW;
  rate = (input - state) * w;
  state = int (rate);
  output = state;
```

interface:

type	LowPassFilter																								
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>input</td> <td>signal</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>output</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	input	signal	real	1	in	1			output	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
input	signal	real	1	in	1																				
output	signal	real	1	out	1																				

information:

name	Counterbalance_valve.CounterBalance_valve_pilot
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Signal\Filters\LowPassFilter-FO.emx
TimeStamp	2007-9-27 15:24:41
AllowLibraryUpdate	True
name	DocumentationMask
name	svgFilename

MR Counterblance_valve**implementation:**

```
parameters
  real density = 861.8;
  //Counterbalance valve settings
  real Qcbv=140.0, dPcbv=5.0e5;
  real Popening = 25e5, Pfullyopen = 250e5;
  real GLeak = 1.0e-16;
variables
  real hidden FlowArea;
  real flow;
  real hidden hoistflow;
  real hidden slackflow;

  initialequations
```

```

//Maximum flow area for checkvalve
FlowArea = Qcbv/(60e3)/(2*dPcbv/density)^0.5;

equations
  if p.p>0.0 then
//Flow through checkvalve in hoist direction
    hoistflow = FlowArea*(2*abs(p.p)/density)^0.5*sign(p.p);
  else
    hoistflow = abs(p.p)*GLeak*sign(p.p);
  end;

  if Ppilot<Popening then
    slackflow = abs(p.p)*GLeak*sign(p.p);
  else
    slackflow = min([1.0, (Ppilot-Popening)/(Pfullyopen - Popening)])*FlowArea*
(2*abs(p.p)/density)^0.5*sign(p.p);
  end;

  p.phi = hoistflow + slackflow;

//Calculating flow in l/min for plotting purposes
  flow = p.phi*60.0e3;

```

interface:

type	MR							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	1	in	1		
	Ppilot	signal	real	1	in	1		

information:

name	Counterbalance_valve.Counterbalance_valve
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\MR.emx
TimeStamp	2011-11-29 16:09:29
AllowLibraryUpdate	True
name	DocumentationMask

1 OneJunction14

implementation:

```

equations
  sum(direct(p.e)) = 0;
  equal(collect(p.f));
  flow = first(p.f);

```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Counterbalance_valve.OneJunction14
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx

TimeStamp	2011-11-29 16:17:51
-----------	---------------------

0 ZeroJunction

implementation:

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);

```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	Counterbalance_valve.ZeroJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction2

implementation:

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);

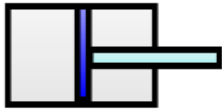
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

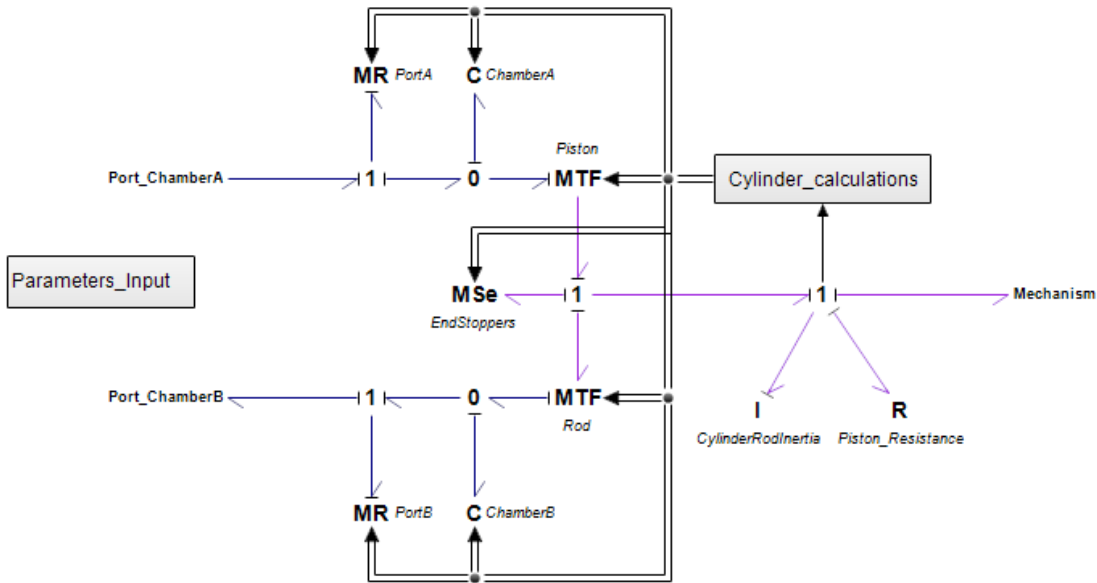
name	Counterbalance_valve.ZeroJunction2
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16



Cylinder

Cylinder (Standard)

implementation:



interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	Port_ChamberA	hydraulic	real	1	in	1		
	Port_ChamberB	hydraulic	real	1	out	1		
	Mechanism	mechanical	real	1	out	1		

information:

name	Cylinder
implementation	Standard
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask
name	ScriptFilename
string	D:\Master Thesis\picture\cylinder.png
name	MaskChoice
int	1
name	BitmapFilename
string	D:\Master Thesis\picture\cylinder.png

ChamberA

implementation:

```
// Calculating pressure in chamber A
parameters
  real global bulkmodulus;

variables
  real Voil, Pressure;
```

```

equations
// Integrating flow to find oil quantity
Voil = int(p.phi,Cylcalc[3,1]);
if Voil>0.0 then
  p.p = bulkmodulus/Cylcalc[5,1]*Voil;
else
  p.p = 0.0;// No pressure when no oil present in volume
end;

// Calculating pressure in bar for plotting
Pressure = p.p/1e5;

```

interface:

type	C							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	1	in	1		
	Cylcalc	signal	real	1	in	[9,1]		
restrictions	kind	priority	type	ports				
	causality	preferred	out	p				

information:

name	Cylinder.ChamberA
LibraryPath	Bond Graph\C.emx
TimeStamp	2011-11-29 15:58:35
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

C ChamberB**implementation:**

```

// Calculating pressure in chamber B
parameters
  real global bulkmodulus;

variables
  real Voil, Pressure;

equations
// Integrating flow to find oil volume
Voil = int(p.phi,Cylcalc[4,1]);
if Voil>0.0 then
  p.p = bulkmodulus/Cylcalc[6,1]*Voil;
else
  p.p = 0.0;// No pressure when no oil present in volume
end;

// Calculating pressure in bar for plotting
Pressure = p.p/1e5;

```

interface:

type	C							
ports	name	domain	type	terminals	orientation	size	unit	description

	p	hydraulic	real	1	in	1
	Cylcalc	signal	real	1	in	[9,1]
restrictions	kind	priority	type	ports		
	causality	preferred	out	p		

information:

name	Cylinder.ChamberB
LibraryPath	Bond Graph\C.emx
TimeStamp	2011-11-29 15:58:35
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

CylinderRodInertia

implementation:

```

parameters
  real Rodmass=500; //Mass of cylinder rod and other translating (motion) inertias
variables
  real Rodmomentum, Rodspeed, Rodposition;
equations
  //Integrating force to find momentum
  Rodmomentum = int(p.e);
  //Finding speed based on momentum
  p.f = Rodmomentum/Rodmass;
  Rodspeed = p.f;
  //Integrating speed to find position of rod
  Rodposition = int(Rodspeed,1);

```

interface:

type	I
ports	name domain type terminals orientation size unit description
	p mechanical real 1 in 1
restrictions	kind priority type ports
	causality preferred in p

information:

name	Cylinder.CylinderRodInertia
LibraryPath	Bond Graph\I.emx
TimeStamp	2011-11-29 15:55:55
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask

Cylinder_calculations

Cylinder_calculations

implementation:

```

// This is a submodel making calculations for a hydraulic cylinder.
// The model use rod position as input, and calculates output values
// such as flow area for inlet and outlet ports, initial fluid volume
// and current physical volume in chamber A and B, piston and rod area
// and force on cylinder piston/rod when reaching either of the end positions.
// The results are sent to other parts of the model in an signal array.
// Cylcalc = [Ainlet;Aoutlet;VAoilinit;VBoilinit;VA;VB;Apist;Arod;Fbumper];

parameters
  real global bulkmodulus;
// Defining initial conditions, rod position and pressures
  real global rodposinit, PAinit, PBinit;
// Defining port dimensions, diameters, stroke and deadvolume for the cylinder
  real global dinlet, doutlet, dcyl, drod, Stroke, vdead;
// Defining maximum pressure, deflection at maximum pressure and damping ratio at end strokes
  real global Pmax, deflection, dampingratio;//sequeece the end stopper to 0.0001m

variables
  real hidden Ainlet, Aoutlet;
  real hidden Apist, Arod;
  real rodpos;
  real hidden VAinit, VBinit, VAoilinit, VBoilinit, VA, VB;
  real hidden kbumper, cbumper, Fbumper;

initialequations
// Calculating area of inlet and outlet ports
  Ainlet = 0.25*pi*dinlet^2;
  Aoutlet = 0.25*pi*doutlet^2;

// Calculating area on piston and rod side
  Apist = 0.25*pi*dcyl^2;
  Arod = Apist-0.25*pi*drod^2;

// Calculating initial physical oil volumes in A and B chamber
  VAinit = vdead + Apist*rodposinit;
  VBinit = vdead + Arod*(Stroke-rodposinit);
// Calculating initial oil quantity in A and B chamber
  VAoilinit = PAinit*VAinit/bulkmodulus;
  VBoilinit = PBinit*VBinit/bulkmodulus;

// Calculating stiffness and damping for bumper
  kbumper = Pmax*Apist/deflection;
  cbumper = dampingratio*2*sqrt(Pmax*Apist/9.81*kbumper);

equations
// Calculating rod position
  rodpos = int(rodspeed,rodposinit);

//Calculating physical volume in A and B chamber
  VA = vdead + Apist*rodpos;
  VB = vdead + Arod*(Stroke-rodpos);

// Checking if minimum or maximum stroke has been reached
  if rodpos> Stroke then
    Fbumper = kbumper*(rodpos-Stroke) + cbumper*max([0,rodspeed]); //Maximum stroke reached
  else
    if rodpos< 0 then
      Fbumper = kbumper*(rodpos) + cbumper*min([0,rodspeed]); // Minimum stroke reached
    else
      Fbumper = 0.0; // Normal stroke - no bumper forces
    end;
  end;

// Calculation results in a matrix
  Cylcalc = [Ainlet;Aoutlet;VAoilinit;VBoilinit;VA;VB;Apist;Arod;Fbumper];

```

interface:

type Submodel							
ports	name	domain	type	terminals	orientation	size	unit description
	rodspeed	signal	real	1	in	1	
	Cylcalc	signal	real	1	out	[9,1]	

information:

name	Cylinder.Cylinder_calculations
------	--------------------------------

Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

MSe EndStoppers

implementation:

```
// Force when cylinder reaches maximum or minimum stroke (Damper and spring)
equations
  p.e = Cylcalc[9,1];
```

interface:

type	MSe																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>power</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> <tr><td>Cylcalc</td><td>signal</td><td>real</td><td>1</td><td>in</td><td>[9,1]</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1			Cylcalc	signal	real	1	in	[9,1]		
name	domain	type	terminals	orientation	size	unit	description																		
p	power	real	1	in	1																				
Cylcalc	signal	real	1	in	[9,1]																				
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>fixed</td><td>out</td><td>p</td></tr> </tbody> </table>	kind	priority	type	ports	causality	fixed	out	p																
kind	priority	type	ports																						
causality	fixed	out	p																						

information:

name	Cylinder.EndStoppers
LibraryPath	Bond Graph\MSe.emx
TimeStamp	2011-11-29 16:12:33
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

1 OneJunction10

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> </table>	name	domain	type	terminals	orientation	size	unit	description
name	domain	type	terminals	orientation	size	unit	description		

	p	hydraulic	real	any	none	1	
	flow	signal	real	1	out	1	
restrictions	kind	priority	type	ports			
	causality	constraint	one_out	p			

information:

name	Cylinder.OneJunction10
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction13

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction13
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction15

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction15
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction3

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	any	none	1		
restrictions	flow	signal	real	1	out	1		
	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction3
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2007-9-27 9:51:18

Parameters_Input

Parameters_Input

implementation:

```
parameters
  real global bulkmodulus = 1600000000;
// Defining initial conditions, rod position and pressures
  real global rodposinit = 1;
  real global PAinit= 5e5;
  real global PBinit= 1e5;
// Defining port dimensions, diameters, stroke and deadvolume for the cylinder
  real global dinlet = 0.01;
  real global doutlet = 0.01;
  real global dcy1 = 283e-3;
  real global drod = 141e-3;
  real global Stroke = 2;
  real global vdead = 0.001;
// Defining maximum pressure, deflection at maximum pressure and damping ratio at end strokes
  real global Pmax = 35000000;
  real global deflection = 0.001;
  real global dampingratio= 0.1;//sequeece the end stopper to 0.0001m
```

interface:

```
type Submodel
```

information:

name	Cylinder.Parameters_Input
------	---------------------------

Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

MTF Piston

implementation:

```
// Piston side: Relation between pressure/flow and force/speed, i.e. piston area
equations
  p1.p = p2.F/Cylcalc[7,1];
  p2.v = p1.phi/Cylcalc[7,1];
```

interface:

type	MTF																																						
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p1</td><td>hydraulic</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> <tr><td>p2</td><td>mechanical</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> <tr><td>Cylcalc</td><td>signal</td><td>real</td><td>1</td><td>in</td><td>[9,1]</td><td></td><td></td></tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p1	hydraulic	real	1	in	1			p2	mechanical	real	1	out	1			Cylcalc	signal	real	1	in	[9,1]		
name	domain	type	terminals	orientation	size	unit	description																																
p1	hydraulic	real	1	in	1																																		
p2	mechanical	real	1	out	1																																		
Cylcalc	signal	real	1	in	[9,1]																																		
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>constraint</td><td>not_equal</td><td>p1, p2</td></tr> </tbody> </table>							kind	priority	type	ports	causality	constraint	not_equal	p1, p2																								
kind	priority	type	ports																																				
causality	constraint	not_equal	p1, p2																																				

information:

name	Cylinder.Piston
LibraryPath	Bond Graph\MTF.emx
TimeStamp	2011-11-29 16:15:31
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

R Piston_Resistance

implementation:

```
parameters
  real u2=1e3;
equations
  p.F = u2 *9.81*5* p.v;
//The current cylinder is capable of lifting almost 100 Te at 280 bar.
//Assuming a linear, load independent frictional loss of 5%
```

interface:

type	R																						
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>mechanical</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p	mechanical	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description																
p	mechanical	real	1	in	1																		

information:

name	Cylinder.Piston_Resistance
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

MR PortA**implementation:**

```
// Calculating the flow through the cylinders inlet port
parameters
  real global density;

variables
  real inletflow;

equations
// Flow through inlet port
  p.phi = Cylcalc[1,1]*sqrt(2*abs(p.p)/density)*sign(p.p);

// Converting to l/min for plotting purposes
  inletflow = p.phi*60e3;
```

interface:

type	MR						
ports							
name	domain	type	terminals	orientation	size	unit	description
p	hydraulic	real	1	in	1		
Cylcalc	signal	real	1	in	[9,1]		

information:

name	Cylinder.PortA
LibraryPath	Bond Graph\MR.emx
TimeStamp	2011-11-29 16:09:29
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

MR PortB**implementation:**

```
// Calculating the flow through the cylinders outlet port
parameters
  real global density;

variables
  real outletflow;

equations
```

```
// Flow through outlet port
p.phi = Cylcalc[2,1]*sqrt(2*abs(p.p)/density)*sign(p.p);

// Converting to l/min for plotting purposes
outletflow = p.phi*60e3;
```

interface:

type	MR							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	1	in	1		
	Cylcalc	signal	real	1	in	[9,1]		

information:

name	Cylinder.PortB
LibraryPath	Bond Graph\MR.emx
TimeStamp	2011-11-29 16:09:29
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

MTF Rod**implementation:**

```
// Piston side: Relation between pressure/flow and force/speed, i.e. piston area minus rod area
equations
p1.F = p2.p*Cylcalc[8,1];
p2.phi = p1.v*Cylcalc[8,1];
```

interface:

type	MTF							
ports	name	domain	type	terminals	orientation	size	unit	description
	p1	mechanical	real	1	in	1		
	p2	hydraulic	real	1	out	1		
	Cylcalc	signal	real	1	in	[9,1]		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	p1, p2				

information:

name	Cylinder.Rod
LibraryPath	Bond Graph\MTF.emx
TimeStamp	2011-11-29 16:15:31
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1

name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

• Splitter3

implementation:

```
equations
collect (output) = input;
```

interface:

type	Splitter							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[9,1]		
	input	signal	real	1	in	[9,1]		

information:

name	Cylinder.Splitter3
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter5

implementation:

```
equations
collect (output) = input;
```

interface:

type	Splitter							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[9,1]		
	input	signal	real	1	in	[9,1]		

information:

name	Cylinder.Splitter5
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter6

implementation:

```
equations
collect (output) = input;
```

interface:

type	Splitter							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[9,1]		

input	signal	real	1	in	[9,1]
-------	--------	------	---	----	-------

information:

name	Cylinder.Splitter6
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter8

implementation:

equations

```
collect (output) = input;
```

interface:

type	Splitter							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[9,1]		
	input	signal	real	1	in	[9,1]		

information:

name	Cylinder.Splitter8
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

0 ZeroJunction1

implementation:

equations

```
sum (direct (p.f)) = 0;
equal (collect (p.e));
effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	Cylinder.ZeroJunction1
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2007-9-27 9:51:43

0 ZeroJunction2

implementation:

```

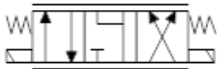
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
  
```

interface:

type		ZeroJunction						
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

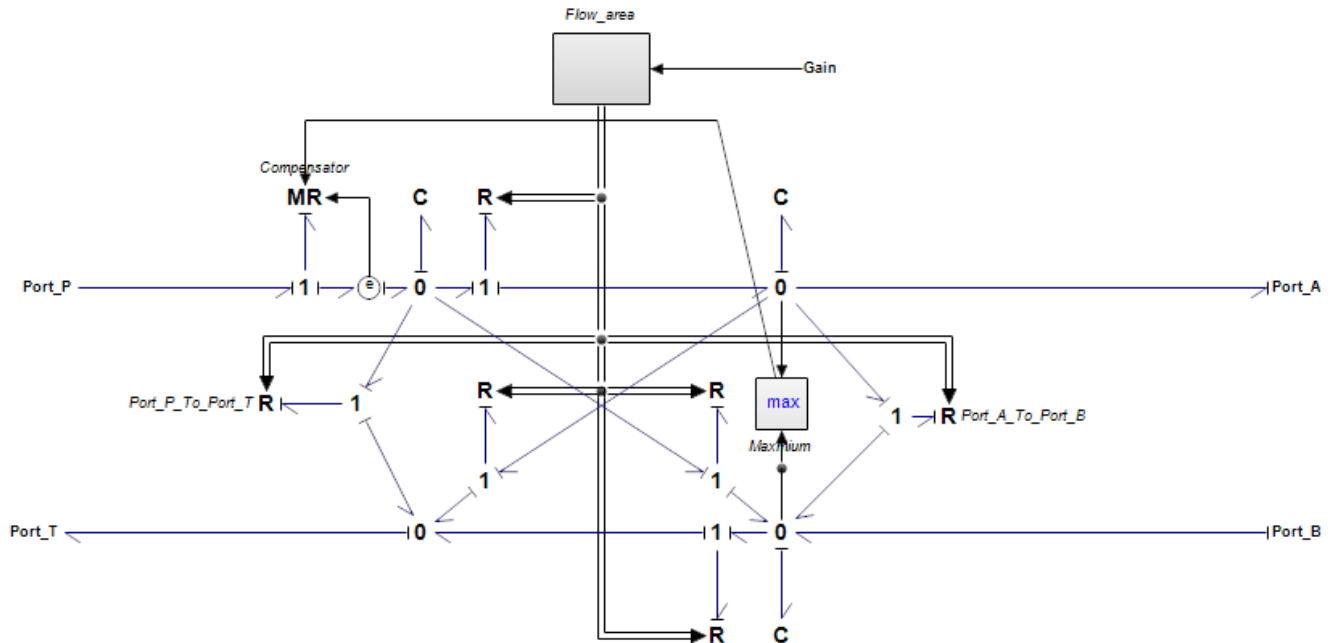
information:

name	Cylinder.ZeroJunction2
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2007-9-27 9:51:43



DCV (Standard)

implementation:



interface:

type		Submodel						
ports	name	domain	type	terminals	orientation	size	unit	description
	Port_P	power	real	1	in	1		
	Port_T	power	real	1	out	1		
	Port_A	power	real	1	out	1		

Port_B	power	real	1	in	1
Gain	signal	real	1	in	1

information:

name	DCV
implementation	Standard
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

MR Compensator**implementation:**

```

parameters
  real density=861.8;
  real compensatorpressure = 5e5; //Desired compensator pressure
  real Compensator_dp5 = 1000; //Flow through fully open compensator at 5 bar

variables
  real Area, flow, flow_dp5, dP;

initialequations
  // Valve flow area calculated from flow at 5 bar pressure drop
  flow_dp5 = Compensator_dp5;
  Area = (flow_dp5/60e3)/(sqrt(2.0*5e5/density));

equations
  // Avoiding reverse flow
  dP = max([(PHPU - Pload), 0]);
  // Flow only when load pressure is less than 5 bar lower than HPU pressure
  if dP>compensatorpressure then
    p.f = 0.0;
  else
    p.f = (compensatorpressure-dP)/compensatorpressure*Area*sqrt(2.0*abs(p.e)/density)*sign(p.e);
  end;

  // Calculating flow in l/min for plotting
  flow = p.f*60e3;

```

interface:

type	MR																																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>Pload</td> <td>signal</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>PHPU</td> <td>signal</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1			Pload	signal	real	1	in	1			PHPU	signal	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description																										
p	power	real	1	in	1																												
Pload	signal	real	1	in	1																												
PHPU	signal	real	1	in	1																												

information:

name	DCV.Compensator
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\MR.emx
TimeStamp	2007-9-25 12:2:35
AllowLibraryUpdate	True

Control_volume1**implementation:**

```

// Control volume
parameters
  real global Pinit, bulkmodulus;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume
  Qinit = Pinit*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(p.f,Qinit);

  if Q>0.0 then
    p.e = bulkmodulus/ControlVolume*Q;
  else
    p.e = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = p.e/1e5;

```

interface:

type	C																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	power	real	1	in	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>preferred</td> <td>out</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	preferred	out	p								
kind	priority	type	ports														
causality	preferred	out	p														

information:

name	DCV.Control_volume1
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\C.emx
TimeStamp	2007-10-31 10:43:59
AllowLibraryUpdate	True

C Control_volume2**implementation:**

```

// Control volume
parameters
  real global bulkmodulus;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(p.f,Qinit);

  if Q>0.0 then
    p.e = bulkmodulus/ControlVolume*Q;
  else
    p.e = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = p.e/1e5;

```

interface:

type	C																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	power	real	1	in	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>preferred</td> <td>out</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	preferred	out	p								
kind	priority	type	ports														
causality	preferred	out	p														

information:

name	DCV.Control_volume2
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\C.emx
TimeStamp	2007-10-31 10:43:59
AllowLibraryUpdate	True

C Control_volume3**implementation:**

```
// Control volume
parameters
  real global Pinit, bulkmodulus;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume
  Qinit = Pinit*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(p.f,Qinit);

  if Q>0.0 then
    p.e = bulkmodulus/ControlVolume*Q;
  else
    p.e = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = p.e/1e5;
```

interface:

type	C																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	power	real	1	in	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>preferred</td> <td>out</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	preferred	out	p								
kind	priority	type	ports														
causality	preferred	out	p														

information:

name	DCV.Control_volume3
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\C.emx
TimeStamp	2007-10-31 10:43:59
AllowLibraryUpdate	True

EffortSensor1

implementation:

```

equations
  p2.e = p1.e;
  p1.f = p2.f;
  effort = p1.e;

```

interface:

type	EffortSensor							
ports	name	domain	type	terminals	orientation	size	unit	description
	p1	hydraulic	real	1	in	1		
	p2	hydraulic	real	1	out	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	p1, p2				

information:

name	DCV.EffortSensor1
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\EffortSensor.emx
TimeStamp	2007-9-27 9:59:27

**Flow_area****implementation:**

```

parameters
  real density=861.8;
  real deadband = 0.05;
  real Qmax = 500; //Maximum flow in l/min
  real dPQmax = 5.0e5; //Pressure differential at maximum flow
  real Rsn=0.01; //the restriction of orifice area dring drain oil to tank
  real C1=0, C2=0, C3=1, C4=1, C5=0, C6=0;
variables
  real slidePos;
  real Area; //Valve flow area
  real AreaPA, AreaPB, AreaAT, AreaBT, AreaPT, AreaAB;

initialequations
// Calculating maximum flow area
  Area = Qmax/(60e3)/(2*dPQmax/density)^0.5;

equations
// Keeping slide position within +/- 10
  slidePos = min([1.0,max([-1.0,gain])]);

// Flow to tank within deadband for open center valve NOTE: DISABLED!
  if abs(slidePos)<deadband then
    AreaPA = C1*Rsn*Area;
    AreaPB = C2*Rsn*Area;
    AreaAT = C3*Rsn*Area;
    AreaBT = C4*Rsn*Area;
    AreaPT = C5*Rsn*Area;
    AreaAB = C6*Rsn*Area;
  else
    if slidePos>deadband then
//Hoist direction
      AreaPA = (slidePos-deadband)/(1.0 - deadband)*Area;
      AreaPB = 0.0;
      AreaAT = 0.0;
      AreaBT = AreaPA;
      AreaPT = 0.0;
      AreaAB = 0.0;
    else
//Slack direction

```

```

AreaPA = 0.0;
AreaPB = abs((slidePos+deadband)/(1.0 - deadband)*Area);
AreaAT = AreaPB;
AreaBT = 0.0;
AreaPT = 0.0;
AreaAB = 0.0;
end;
end;
FlowArea = [AreaPA;AreaPB;AreaAT;AreaBT;AreaPT;AreaAB];

```

interface:

type		FlowArea						
ports	name	domain	type	terminals	orientation	size	unit	description
	gain	signal	real	1	in	1		
	FlowArea	signal	real	1	out	[6,1]		

information:

name	DCV.Flow_area
Version	4.0
LibraryPath	C:\Program Files\20-sim 4.0\System\Submodel.emx
TimeStamp	2007-10-31 11:32:54
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask

max

Maxmium

implementation:

```

equations
  output = if input1 >= input2 then
    input1
  else
    input2
  end;

```

interface:

type		Switch						
ports	name	domain	type	terminals	orientation	size	unit	description
	input1	signal	real	1	in	1		
	output	signal	real	1	out	1		
	input2	signal	real	1	in	1		

information:

name	DCV.Maxmium
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Signal\Block Diagram Non-Linear\Switch-Maximum.emx
TimeStamp	2007-9-26 12:54:0

1 OneJunction

implementation:

```

equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);

```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction1

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	DCV.OneJunction1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction2

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		

restrictions	kind	priority	type	ports
	causality	constraint	one_out	p

information:

name	DCV.OneJunction2
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction5

implementation:

equations

```
sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);
```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>any</td> <td>none</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>flow</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
flow	signal	real	1	out	1																				
restrictions	<table border="1"> <tr> <td>kind</td> <td>priority</td> <td>type</td> <td>ports</td> </tr> <tr> <td>causality</td> <td>constraint</td> <td>one_out</td> <td>p</td> </tr> </table>	kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																						
causality	constraint	one_out	p																						

information:

name	DCV.OneJunction5
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction6

implementation:

equations

```
sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);
```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>any</td> <td>none</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>flow</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
flow	signal	real	1	out	1																				
restrictions	<table border="1"> <tr> <td>kind</td> <td>priority</td> <td>type</td> <td>ports</td> </tr> <tr> <td>causality</td> <td>constraint</td> <td>one_out</td> <td>p</td> </tr> </table>	kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																						
causality	constraint	one_out	p																						

information:

name	DCV.OneJunction6
Version	4.2

IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction7

implementation:

equations

```
sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);
```

interface:

type	OneJunction																														
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>any</td> <td>none</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>flow</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																								
p	hydraulic	real	any	none	1																										
flow	signal	real	1	out	1																										
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>constraint</td> <td>one_out</td> <td>p</td> </tr> </tbody> </table>							kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																												
causality	constraint	one_out	p																												

information:

name	DCV.OneJunction7
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction8

implementation:

equations

```
sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);
```

interface:

type	OneJunction																														
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>any</td> <td>none</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>flow</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																								
p	hydraulic	real	any	none	1																										
flow	signal	real	1	out	1																										
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>constraint</td> <td>one_out</td> <td>p</td> </tr> </tbody> </table>							kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																												
causality	constraint	one_out	p																												

information:

name	DCV.OneJunction8
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

R Port_A_To_Port_B

implementation:

```

parameters
  real global density;

variables
  real flowAB;

equations
// Flow from A to T
  p.f = FlowArea[6,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
  flowAB = p.f*60.0e3;

```

interface:

type R	
ports	name domain type terminals orientation size unit description
	p power real 1 in 1
	FlowArea signal real 1 in [6,1]

information:

name	DCV.Port_A_To_Port_B
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_A_To_Port_T**implementation:**

```

parameters
  real global density;

variables
  real flowAT;

equations
// Flow from A to T
  p.f = FlowArea[3,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
  flowAT = p.f*60.0e3;

```

interface:

type R	
ports	name domain type terminals orientation size unit description
	p power real 1 in 1
	FlowArea signal real 1 in [6,1]

information:

name	DCV.Port_A_To_Port_T
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_B_To_Port_T

implementation:

```

parameters
  real global density;

variables
  real flowBT;

equations
// Flow from B to T
  p.f = FlowArea[4,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
  flowBT = p.f*60.0e3;

```

interface:

type R	
ports	name domain type terminals orientation size unit description
	p power real 1 in 1
	FlowArea signal real 1 in [6,1]

information:

name	DCV.Port_B_To_Port_T
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_P_To_Port_A

implementation:

```

parameters
  real global density;

variables
  real flowPA;

equations
// Flow from P to A
  p.f = FlowArea[1,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

//Calculating flow in l/min for plotting purposes
  flowPA = p.f*60.0e3;

```

interface:

type R	
ports	name domain type terminals orientation size unit description
	p power real 1 in 1
	FlowArea signal real 1 in [6,1]

information:

name	DCV.Port_P_To_Port_A
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx

TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_P_To_Port_B

implementation:

```
parameters
  real global density;

variables
  real flowPB;

equations
  // Flow from P to B
  p.f = FlowArea[2,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

  //Calculating flow in l/min for plotting purposes
  flowPB = p.f*60.0e3;
```

interface:

type R							
ports	name	domain	type	terminals	orientation	size	unit description
	p	power	real	1	in	1	
	FlowArea	signal	real	1	in	[6,1]	

information:

name	DCV.Port_P_To_Port_B
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Port_P_To_Port_T

implementation:

```
parameters
  real global density;

variables
  real flowPT;

equations
  // Flow from A to T
  p.f = FlowArea[5,1]*(2*abs(p.e)/density)^0.5*sign(p.e);

  //Calculating flow in l/min for plotting purposes
  flowPT = p.f*60.0e3;
```

interface:

type R							
ports	name	domain	type	terminals	orientation	size	unit description
	p	power	real	1	in	1	
	FlowArea	signal	real	1	in	[6,1]	

information:

name	DCV.Port_P_To_Port_T
Version	4.2

IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

• Splitter1

implementation:

```
equations
collect (output) = input;
```

interface:

type	Splitter							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[6,1]		
	input	signal	real	1	in	[6,1]		

information:

name	DCV.Splitter1
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter10

implementation:

```
equations
collect (output) = input;
```

interface:

type	Splitter							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	1		
	input	signal	real	1	in	1		

information:

name	DCV.Splitter10
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter2

implementation:

```
equations
collect (output) = input;
```

interface:

type	Splitter							
------	----------	--	--	--	--	--	--	--

ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[6,1]		
	input	signal	real	1	in	[6,1]		

information:

name	DCV.Splitter2
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

• Splitter3

implementation:

```
equations
collect (output) = input;
```

interface:

ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	any	out	[6,1]		
	input	signal	real	1	in	[6,1]		

information:

name	DCV.Splitter3
Version	4.0
LibraryPath	Signal\Block Diagram\Splitter.emx
TimeStamp	2008-01-17 11:28:29
IsMainModel	1
KeepParameterValues	False

◦ ZeroJunction3

implementation:

```
equations
sum (direct (p.f)) = 0;
equal (collect (p.e));
effort = first (p.e);
```

interface:

ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		

restrictions	kind	priority	type	ports
	causality	constraint	one_in	p

information:

name	DCV.ZeroJunction3
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx

TimeStamp 2011-11-29 16:45:16

0 ZeroJunction4

implementation:

```
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	DCV.ZeroJunction4
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction5

implementation:

```
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	DCV.ZeroJunction5
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction6

implementation:

```
equations
```

```

sum (direct (p.f)) = 0;
equal (collect (p.e));
effort = first (p.e);

```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	DCV.ZeroJunction6
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

Sf Motor**implementation:**

```

parameters
  real flow = 1776;
equations
  p.f = flow;

```

interface:

type	Sf							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	fixed	in	p				

information:

name	Motor
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Sf.emx
TimeStamp	2011-11-29 16:38:03
AllowLibraryUpdate	True
name	DocumentationMask

1 OneJunction1**implementation:**

```

equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);

```

interface:

type	OneJunction							
------	-------------	--	--	--	--	--	--	--

ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	OneJunction1
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2007-9-27 9:51:18

Se Payload**implementation:**

```

parameters
  real effort = -981000;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

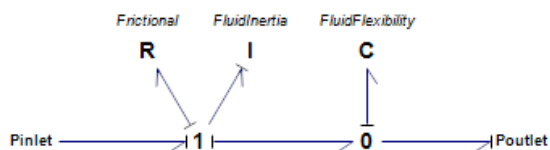
interface:

type	Se							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	out	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Payload
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

PipeFlow

PipeFlow (Advanced)**implementation:****interface:**

type	Submodel
------	----------

ports	name	domain	type	terminals	orientation	size	unit	description
	Pinlet	hydraulic	real	1	in	1		
	Poutlet	hydraulic	real	1	out	1		

information:

name	PipeFlow
implementation	Advanced
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

C FluidFlexibility**implementation:**

```

//Compressibility of a pipe segment
parameters
  real bulkmodulus=1.6e9, Pinit=1e5;
//Pipe dimensions
  real pipediameter=49e-3, pipelength=15.0;
  real pipethickness=0.001;
//number of lumps
real global n;
//Dilation factors
real Hose=0.0;
real E=180e9; //Young's modulus

variables
  real fluidvolume, mechanicalvolume, pressure, Qinit;
  real Bulkmodulus_eff;

initialequations
//The physical volume of the pipe
  mechanicalvolume=0.25*pi*pipediameter^2*pipelength/n;
//Initial oil volume in control volume
  Qinit = Pinit*mechanicalvolume/bulkmodulus;

equations
//Effective bulk modulus for hose
  Bulkmodulus_eff=1/(1/bulkmodulus+5*pipediameter/(4*E*pipethickness));
//Finding the volume of fluid within the pipe
  fluidvolume = int(p.phi,Qinit);

if Hose==0.3 then
  if fluidvolume>0 then
    p.p = Bulkmodulus_eff/mechanicalvolume*fluidvolume;
  else
    p.p = 0.0;
end;
else
//Calculating pressure
  if fluidvolume>0 then
    p.p = bulkmodulus/mechanicalvolume*fluidvolume;
  else
    p.p = 0.0; // Zero pressure when no fluid in volume
  end;
end;
//Calculating the pressure in bars
  pressure = p.e/1e5;

```

interface:

type	C																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	hydraulic	real	1	in	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> </tbody> </table>	kind	priority	type	ports												
kind	priority	type	ports														

causality preferred out p

information:

name	PipeFlow.FluidFlexibility
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\C.emx
TimeStamp	2011-11-29 15:58:35
AllowLibraryUpdate	True
name	DocumentationMask

I FluidInertia**implementation:**

```

parameters
  real density=861.8;
//Pipe dimensions
  real pipediameter=49e-3, pipelength=15.0;
//number of lumps
real global n;

variables
  real pipearea, flow, momentum, fluidinertia;

initialequations
//Calculating the pipe cross section area
  pipearea = 0.25*pi*pipediameter^2;
//Calculating the fluid inertia
  fluidinertia = density*pipelength/pipearea;

equations
//Finding fluid momentum
  momentum = int(p.p);
//Calculating flow
p.phi = momentum / fluidinertia;

// The flow in liters per minute
  flow = p.phi*60e3;

```

interface:

type	I																
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	hydraulic	real	1	in	1												
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality preferred</td><td>in</td><td>p</td><td></td></tr> </tbody> </table>	kind	priority	type	ports	causality preferred	in	p									
kind	priority	type	ports														
causality preferred	in	p															

information:

name	PipeFlow.FluidInertia
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\I.emx
TimeStamp	2011-11-29 15:55:55
AllowLibraryUpdate	True
name	DocumentationMask

R Frictional**implementation:**

```

parameters
//Pipe dimensions
  real pipediameter=49e-3, pipelength=15.0;
//Pipe parameters
  real piperoughness = 0.0002, viscosity=32e-6;
//Fluid density
  real density = 861.8;
//Task 6 signal
  real Material=0;
//If there are some losses because of Local features
real e=0; //local loss coefficient
real m=0; //the number of local features
//when the pipe is multi-lumped model
real global n=1; //number of lumps
variables
  real pipearea, f, Re, pressureloss;

initialequations
//Calculating pipe area
  pipearea = 0.25*pi*pipediameter^2;

equations
//Calculating the Reynoldsnumber
  Re = abs(p.phi)/pipearea*pipediameter/viscosity;

  if Re<2300.0 then
//Laminar flow
  if Re<1 then
//To avoid division by zero in case of zero flow
    f = 64.0/1.0;
  else
    f=64.0/Re;
  end;
  else
//Turbulent flow
if Material==0.2 then
f = 0.25/(log10(piperoughness/3.7+5.74/Re^0.9))^2;
else
  if Re>10^5 then
    f=0.0054+0.396*(Re)^-0.3;
  else
    f=0.3164/Re^0.25;
  end;
end;
end;
//Calculating pressure loss
  p.p = (sign(p.phi)*f*pipelength/n/(pipediameter)+e*m)*0.5*density*(p.phi/pipearea)^2;

//Calculating pressure loss in bars
  pressureloss = p.p/1e5;

```

interface:

type	R							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	1	in	1		

information:

name	PipeFlow.Frictional
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

1 OneJunction**implementation:**

```

equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);

```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	PipeFlow.OneJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

0 ZeroJunction1

implementation:

```

equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);

```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	PipeFlow.ZeroJunction1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

⊕ Pressure_sensor

implementation:

```

equations
  p2.e = p1.e;
  p1.f = p2.f;
  effort = p1.e;

```

interface:

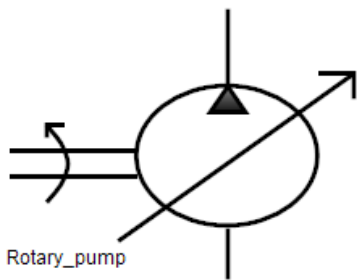
type	EffortSensor
------	--------------

ports	name	domain	type	terminals	orientation	size	unit	description
	p1	hydraulic	real	1	in	1		
	p2	hydraulic	real	1	out	1		
	effort	signal	real	1	out	1		

restrictions	kind	priority	type	ports
	causality	constraint	not_equal	p1, p2

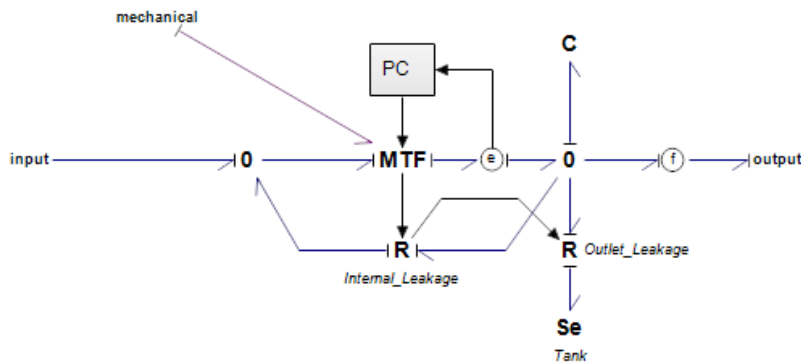
information:

name	Pressure_sensor
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\EffortSensor.emx
TimeStamp	2011-11-29 15:44:34
AllowLibraryUpdate	True
name	DocumentationMask



Rotary_pump (standard)

implementation:



interface:

type	Submodel
------	----------

ports	name	domain	type	terminals	orientation	size	unit	description
	input	power	real	1	in	1		
	output	power	real	1	out	1		
	mechanical	power	real	1	in	1		

information:

name	Rotary_pump
implementation	standard
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False

name	DocumentationMask
------	-------------------

⊖ Control_volume3

implementation:

```
// Control volume
parameters
  real global bulkmodulus;
  real ControlVolume = 0.01;
  real Pinit=1e5;
variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume
  Qinit = Pinit*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(p.f,Qinit);

  if Q>0.0 then
    p.e = bulkmodulus/ControlVolume*Q;
  else
    p.e = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = p.e/1e5;
```

interface:

type	C																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	power	real	1	in	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>preferred</td> <td>out</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	preferred	out	p								
kind	priority	type	ports														
causality	preferred	out	p														

information:

name	Rotary_pump.Control_volume3
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\C.emx
TimeStamp	2007-10-31 10:43:59
AllowLibraryUpdate	True

⊖ EffortSensor1

implementation:

```
variables
  real Pipeinlet;
equations
  p2.e = p1.e;
  p1.f = p2.f;
  effort = p1.e;
  Pipeinlet=effort*1e-5;
```

interface:

type	EffortSensor																																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p1</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>p2</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>effort</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p1	hydraulic	real	1	in	1			p2	hydraulic	real	1	out	1			effort	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																										
p1	hydraulic	real	1	in	1																												
p2	hydraulic	real	1	out	1																												
effort	signal	real	1	out	1																												

restrictions	kind	priority	type	ports
	causality	constraint	not_equal	p1, p2

information:

name	Rotary_pump.EffortSensor1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\EffortSensor.emx
TimeStamp	2011-11-29 15:44:34

FlowSensor1**implementation:**

equations

```
p2.f = p1.f;
p1.e = p2.e;
flow = p1.f*60e3;
```

interface:

type	FlowSensor							
ports	name	domain	type	terminals	orientation	size	unit	description
	p1	hydraulic	real	1	in	1		
	p2	hydraulic	real	1	out	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	p1, p2				

information:

name	Rotary_pump.FlowSensor1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\FlowSensor.emx
TimeStamp	2011-11-29 15:50:53

R Internal_Leakage**implementation:**

variables

```
real dp; //the differencial pressure between pump inlet and outlet
```

real Flow;

equations

```
Gin=(8e-8)*Dpump-(1.2e-12); //the internal conductance
dp = p_in.p-p_out.p;
p_out.phi = Gin * dp;
p_in.phi=p_out.phi;
Flow=p_out.phi*60e3;
```

interface:

type	R							
ports	name	domain	type	terminals	orientation	size	unit	description
	p_in	hydraulic	real	1	in	1		
	p_out	hydraulic	real	1	out	1		
	Dpump	signal	real	1	in	1		
	Gin	signal	real	1	out	1		

information:

name	Rotary_pump.Internal_Leakage
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

MTF MTF**implementation:**

```

parameters
  real A= 0.000675; //piston crosssectional area m^2
  real n= 8; //number of pistons
  real D= 0.05; //pitch diameter m
  real Cc= 5.0966e+000; //Constant Friction when displacement=100e-6
  real Cp= 5.4058e-007; //Pressure dependent friction
  real Cn= -1.3992e-002; //Speed dependent friction

variables
  real T_loss; //overall toque losses
real Flow;
equations
  Dpump = A*n*D/(2*pi); //pump displacement per radian(m^3/rad)
  T_loss=Cc+Cp*(P_out.p-P_in.p)+Cn*P_motor.omega;
  P_motor.T = T_loss+A*n*D/(2*pi)*(P_out.p-P_in.p)*tan(Angle);
//flow rate of pump which A*n*D/(2*pi)is the pump displacement per radian(m^3/rad)
  P_out.phi = A*n*D/(2*pi)*P_motor.omega*tan(Angle);
  P_in.phi = P_out.phi;
  Flow=P_out.phi*60e3;

```

interface:

type	MTF							
ports	name	domain	type	terminals	orientation	size	unit	description
	P_motor	rotation	real	1	in	1		
	P_out	hydraulic	real	1	out	1		
	Angle	signal	real	1	in	1		
	P_in	hydraulic	real	1	in	1		
	Dpump	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	not_equal	P_motor, P_out				

information:

name	Rotary_pump.MTF
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\MTF.emx
TimeStamp	2011-11-29 16:15:31

R Outlet_Leakage**implementation:**

```

variables
  real dp; //the differential pressure between pump inlet and outlet
  real Gex; //the internal conductance
  real Flow;

```


equations

```
Gex=Gin/2;
dp = p_outlet.p-p_tank.p;
p_outlet.phi = Gex * dp;
p_tank.phi=p_outlet.phi;
Flow=p_outlet.phi*60e3;
```

interface:

type	R							
ports	name	domain	type	terminals	orientation	size	unit	description
	p_outlet	hydraulic	real	1	in	1		
	p_tank	hydraulic	real	1	out	1		
	Gin	signal	real	1	in	1		

information:

name	Rotary_pump.Outlet_Leakage
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask



PC

implementation:

```
parameters
  real Pset= 300e5; //set point of Max pressure
  real K = 10e5;//Pressure deviation from setpoint for full pump flow
variables
  real MaxAngle;
  real gain;

initialequations
  MaxAngle = pi/4; //the maximum angle of swash-plate
equations
// No flow when P_out>Pset
// Pump flow proportional to (Pset - P) when Pset>P
if P_out>Pset then
  angle = 0.0;
else
// Limit flow between 0 and 100% of max flow
  gain = min([1.0, (Pset - P_out)/K]);
  angle = MaxAngle*gain;
end;
```

interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	P_out	signal	real	1	in	1		
	angle	signal	real	1	out	1		

information:

name	Rotary_pump.PC
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False

name	DocumentationMask
------	-------------------

Se Tank

implementation:

```
equations
  p.e = 1e5;
```

interface:

type	Se							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	1	in	1		
restrictions	kind	priority	type	ports				
	causality	fixed	out	p				

information:

name	Rotary_pump.Tank
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

0 ZeroJunction

implementation:

```
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	Rotary_pump.ZeroJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction1

implementation:

```
equations
```

```
sum (direct (p.f)) = 0;
equal (collect (p.e));
effort = first (p.e);
```

interface:

type	ZeroJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_in	p				

information:

name	Rotary_pump.ZeroJunction1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

**Sine****implementation:**

```
parameters
  real amplitude = 1.0;           // amplitude of the wave
  real omega = 1.0 {rad/s};      // angular frequency of the wave
  real motionstart=5;
variables
  boolean hidden change;
  real hidden half;
equations
  "calculate at least 2 points per cycle to get a triangle"
  half = pi / omega;
  change = frequencyevent (half, half / 2);
  if time < motionstart then
    output=0;
  else
    "calculate the sine wave"
    output = amplitude * sin ( omega * time);
  end;
```

interface:

type	WaveGenerator							
ports	name	domain	type	terminals	orientation	size	unit	description
	output	signal	real	1	out	1		

information:

name	Sine
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Signal\Sources\WaveGenerator-Sine.emx
TimeStamp	2007-9-27 16:12:6

Se Tank**implementation:**

```

parameters
  real effort = 0;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

interface:

type	Se																									
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th colspan="2">description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td colspan="2"></td> </tr> </tbody> </table>								name	domain	type	terminals	orientation	size	unit	description		p	power	real	1	out	1			
name	domain	type	terminals	orientation	size	unit	description																			
p	power	real	1	out	1																					
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>fixed</td> <td>out</td> <td>p</td> </tr> </tbody> </table>								kind	priority	type	ports	causality	fixed	out	p										
kind	priority	type	ports																							
causality	fixed	out	p																							

information:

name	Tank
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

Se Tank2**implementation:**

```

parameters
  real effort = 0;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

interface:

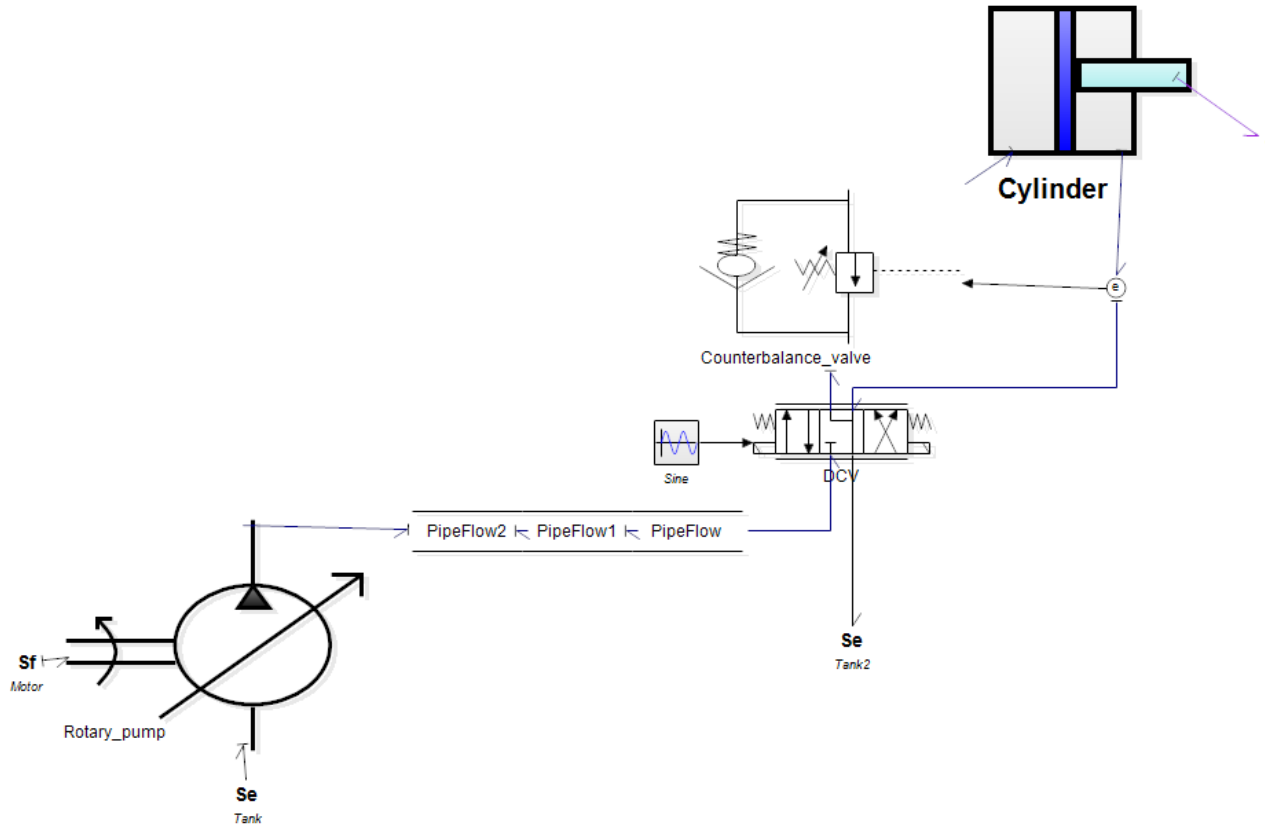
type	Se																									
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th colspan="2">description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td colspan="2"></td> </tr> </tbody> </table>								name	domain	type	terminals	orientation	size	unit	description		p	power	real	1	in	1			
name	domain	type	terminals	orientation	size	unit	description																			
p	power	real	1	in	1																					
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>fixed</td> <td>out</td> <td>p</td> </tr> </tbody> </table>								kind	priority	type	ports	causality	fixed	out	p										
kind	priority	type	ports																							
causality	fixed	out	p																							

information:

name	Tank2
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

model **model**

implementation:

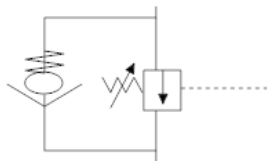


interface:

type Mainmodel

information:

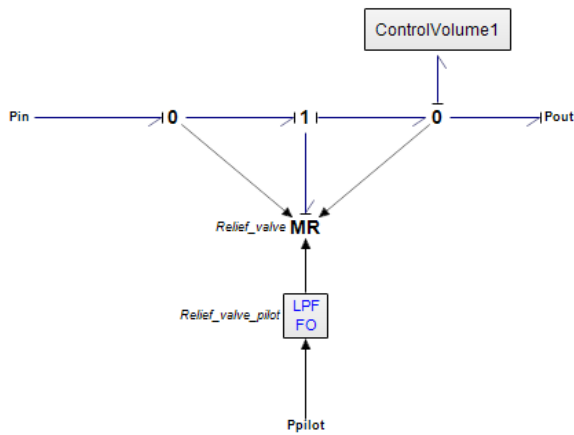
name	model
Version	4.6
IsMainModel	0
KeepParameterValues	False
LibraryPath	Hydraulic components\KBC Advanced.emx
TimeStamp	2016-5-30 15:02:39



Counterbalance_valve

Counterbalance_valve (Advanced)

implementation:



interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	Pin	hydraulic	real	1	in	1		
	Pout	hydraulic	real	1	out	1		
	Ppilot	signal	real	1	in	1		

information:

name	Counterbalance_valve
implementation	Advanced
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask

ControlVolume1 (contrl_volume1)

implementation:

```
// Control volume
parameters
  real bulkmodulus= 1.6e9;
  real ControlVolume = 0.01;

variables
  real Qinit, Pressure, Q;

initialequations
// Initial oil volume in control volume at 1 bar
  Qinit = 1.0e5*ControlVolume/bulkmodulus;

equations
// Integrating flows to find oil volume
  Q = int(cv.phi,Qinit);

  if Q>0.0 then
    cv.p = bulkmodulus/ControlVolume*Q;
  else
    cv.p = 0.0;// No pressure when no oil present in volume
  end;
// Calculating pressure in bar for plotting
  Pressure = cv.p/1e5;
```

interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	cv	hydraulic	real	1	in	1		

information:

name	Counterbalance_valve.ControlVolume1
implementation	contrl_volume1
Version	4.0
LibraryPath	Template\Submodel-Equation.emx

IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask

1 OneJunction14

implementation:

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction																														
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>any</td> <td>none</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>flow</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>							name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																								
p	hydraulic	real	any	none	1																										
flow	signal	real	1	out	1																										
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality constraint</td> <td>one_out</td> <td>p</td> <td></td> </tr> </tbody> </table>							kind	priority	type	ports	causality constraint	one_out	p																	
kind	priority	type	ports																												
causality constraint	one_out	p																													

information:

name	Counterbalance_valve.OneJunction14
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

MR Relief_valve

implementation:

```
parameters
  real density = 861.8;
  //Counterbalance valve settings
  real Popening = 25e5;
  real D=0.025;
  real GLeak = 1.0e-16;
  real Pset=200e5;

variables
  real hidden FlowArea_max;
  real FlowAreacv {m2},FlowArearv {m2};
  real hidden x,y;
  real flow;
  real hidden K;
  real hidden hoistflow;
  real hidden slackflow;

initialequations
  //Maximum flow area for checkvalve
  FlowArea_max=2*pi*D;

equations
  x=D/4*min([1,max([0.000001,pHoist/5e5])]);
  y=D/4*min([1,max([0.000001,pSlack/Pset])]);
  // K = 1.30+0.20*(FlowArea_max/FlowArea)^2;
  if p.p>0.0 then
  //Flow through checkvalve in hoist direction
  FlowAreacv=FlowArea_max*x;
  K = 1.30+0.20*(FlowArea_max/FlowAreacv)^2;
  hoistflow = FlowAreacv*(2*abs(p.p)/(density*K))^0.5*sign(p.p);
  else
  hoistflow = abs(p.p)*GLeak*sign(p.e);
  end;

  if Ppilot>Popening then
  //Flow through checkvalve in hoist direction
  FlowArearv= FlowArea_max*y;
  K = 1.30+0.20*(FlowArea_max/FlowArearv)^2;
  slackflow = FlowArearv*(2*abs(p.p)/(density*K))^0.5*sign(p.p);
  else
  slackflow = abs(p.p)*GLeak*sign(p.e);
  end;

p.phi = hoistflow +slackflow;
//Calculating flow in l/min for plotting purposes
```

```
flow = p.phi*60.0e3;
```

interface:

type		MR						
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	1	in	1		
	Ppilot	signal	real	1	in	1		
	pHoist	signal	real	1	in	1		
	pSlack	signal	real	1	in	1		

information:

name	Counterbalance_valve.Relief_valve
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\MR.emx
TimeStamp	2011-11-29 16:09:29
AllowLibraryUpdate	True
name	DocumentationMask

LPF	Relief_valve_pilot
FO	

implementation:

```
parameters
  real BW = 0.4 {Hz};      // Bandwidth
variables
  real rate, state, w;
equations
  w = 2*3.1415926536*BW;
  rate = (input - state) * w;
  state = int (rate);
  output = state;
```

interface:

type		LowPassFilter						
ports	name	domain	type	terminals	orientation	size	unit	description
	input	signal	real	1	in	1		
	output	signal	real	1	out	1		

information:

name	Counterbalance_valve.Relief_valve_pilot
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Signal\Filters\LowPassFilter-FO.emx
TimeStamp	2007-9-27 15:24:41
AllowLibraryUpdate	True
name	DocumentationMask
name	svgFilename

0 ZeroJunction**implementation:**

```
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
```

interface:

type		ZeroJunction						
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	effort	signal	real	1	out	1		

restrictions	kind	priority	type	ports

causality constraint one_in p

information:

name	Counterbalance_valve.ZeroJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16

0 ZeroJunction2

implementation:

```

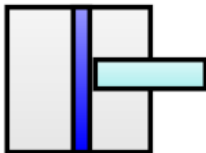
equations
  sum (direct (p.f)) = 0;
  equal (collect (p.e));
  effort = first (p.e);
  
```

interface:

type	ZeroJunction																								
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>any</td> <td>none</td> <td>1</td> <td></td> <td></td> </tr> <tr> <td>effort</td> <td>signal</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			effort	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
effort	signal	real	1	out	1																				
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>constraint</td> <td>one_in</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	constraint	one_in	p																
kind	priority	type	ports																						
causality	constraint	one_in	p																						

information:

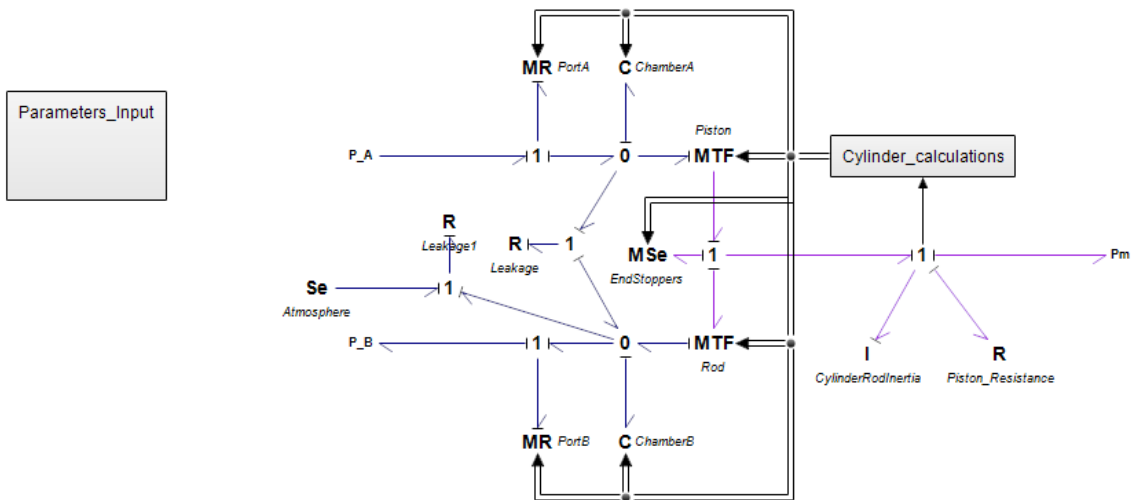
name	Counterbalance_valve.ZeroJunction2
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\ZeroJunction.emx
TimeStamp	2011-11-29 16:45:16



Cylinder

Cylinder (Advanced)

implementation:



interface:

type	Submodel																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>P_A</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	P_A	hydraulic	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
P_A	hydraulic	real	1	in	1												

P_B	hydraulic	real	1	out	1
Pm	mechanical	real	1	out	1

information:

name	Cylinder
implementation	Advanced
Version	4.0
LibraryPath	Template\Submodel-Graphical.emx
IsMainModel	1
KeepParameterValues	True
TimeStamp	2007-11-1 22:32:34
AllowLibraryUpdate	False
name	DocumentationMask
name	ScriptFilename
string	D:\Master Thesis\picture\cylinder.png
name	MaskChoice
int	1
name	BitmapFilename
string	D:\Master Thesis\picture\cylinder.png

Se Atmosphere**implementation:**

```

parameters
  real effort = 1e5;
variables
  real flow;
equations
  p.e = effort;
  flow = p.f;

```

interface:

type	Se																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>power</td> <td>real</td> <td>1</td> <td>out</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	power	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description										
p	power	real	1	out	1												
restrictions	<table border="1"> <thead> <tr> <th>kind</th> <th>priority</th> <th>type</th> <th>ports</th> </tr> </thead> <tbody> <tr> <td>causality</td> <td>fixed</td> <td>out</td> <td>p</td> </tr> </tbody> </table>	kind	priority	type	ports	causality	fixed	out	p								
kind	priority	type	ports														
causality	fixed	out	p														

information:

name	Cylinder.Atmosphere
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\Se.emx
TimeStamp	2011-11-29 16:36:49
AllowLibraryUpdate	True
name	DocumentationMask

C ChamberA**implementation:**

```

// Calculating pressure in chamber A
parameters
  real global bulkmodulus;
  real global dcy1 ;
  real global dcy1_outside;
  real global E,v;
variables
  real Voil, Pressure;
  real bulkmodulus_eff; //effective bulk modulus due to dilation
initialequations

if dcy1>10*(dcyl_outside-dcyl) then
  bulkmodulus_eff = 1/(1/bulkmodulus+(dcyl_outside+dcyl)/(E*(dcyl-dcyl_outside)));
else
  bulkmodulus_eff = 1/(1/bulkmodulus+2/E*((dcyl_outside^2+dcyl^2)/(dcyl_outside^2-dcyl^2)+v));
end;

equations
// Integrating flow to find oil quantity

```

```

Voil = int(p.phi,Cylcalc[3,1]);
if Voil>0.0 then
  p.p = bulkmodulus_eff/Cylcalc[5,1]*Voil;
else
  p.p = 0.0; // No pressure when no oil present in volume
end;

// Calculating pressure in bar for plotting
Pressure = p.p/1e5;

```

interface:

type	C						
ports	name	domain	type	terminals	orientation	size	unit description
	p	hydraulic	real	1	in	1	
	Cylcalc	signal	real	1	in	[9,1]	
restrictions	kind	priority	type	ports			
	causality	preferred	out	p			

information:

name	Cylinder.ChamberA
LibraryPath	Bond Graph\C.emx
TimeStamp	2011-11-29 15:58:35
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

C ChamberB**implementation:**

```

// Calculating pressure in chamber B
parameters
  real global bulkmodulus;
  real global dcyl ;
  real global dcyl_outside;
  real global E,v;
variables
  real Voil, Pressure;
  real bulkmodulus_eff; //effective bulk modulus due to dilation
initialequations

if dcyl>10*(dcyl_outside-dcyl) then
  bulkmodulus_eff = 1/(1/bulkmodulus+(dcyl_outside+dcyl)/(E*(dcyl-dcyl_outside)));
else
  bulkmodulus_eff = 1/(1/bulkmodulus+2/E*((dcyl_outside^2+dcyl^2)/(dcyl_outside^2-dcyl^2)+v));
end;

equations
// Integrating flow to find oil volume
Voil = int(p.phi,Cylcalc[4,1]);
if Voil>0.0 then
  p.p = bulkmodulus/Cylcalc[6,1]*Voil;
else
  p.p = 0.0; // No pressure when no oil present in volume
end;

// Calculating pressure in bar for plotting
Pressure = p.p/1e5;

```

interface:

type	C						
ports	name	domain	type	terminals	orientation	size	unit description
	p	hydraulic	real	1	in	1	
	Cylcalc	signal	real	1	in	[9,1]	
restrictions	kind	priority	type	ports			
	causality	preferred	out	p			

information:

name	Cylinder.ChamberB
LibraryPath	Bond Graph\C.emx
TimeStamp	2011-11-29 15:58:35
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

CylinderRodInertia**implementation:**

```

parameters
  real Rodmass=500; //Mass of cylinder rod and other translating (motion) inertias
variables
  real Rodmomentum, Rodspeed, Rodposition;
equations
  //Integrating force to find momentum
  Rodmomentum = int(p.e);
  //Finding speed based on momentum
  p.f = Rodmomentum/Rodmass;
  Rodspeed = p.f;
  //Integrating speed to find position of rod
  Rodposition = int(Rodspeed);

```

interface:

type	I																
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>mechanical</td><td>real</td><td>1</td><td>in</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	mechanical	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	mechanical	real	1	in	1												
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>preferred</td><td>in</td><td>p</td></tr> </tbody> </table>	kind	priority	type	ports	causality	preferred	in	p								
kind	priority	type	ports														
causality	preferred	in	p														

information:

name	Cylinder.CylinderRodInertia
LibraryPath	Bond Graph\I.emx
TimeStamp	2011-11-29 15:55:55
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask

Cylinder_calculations

Cylinder_calculations**implementation:**

```

// This is a submodel making calculations for a hydraulic cylinder.
// The model use rod position as input, and calculates output values
// such as flow area for inlet and outlet ports, initial fluid volume
// and current physical volume in chamber A and B,piston and rod area
// and force on cylinder piston/rod when reaching either of the end positions.
// The results are sent to other parts of the model in an signal array.
// Cylcalc = [Ainlet;Aoutlet;VAoilinit;VBoilinit;VA;VB;Apist;Arod;Fbumper];

parameters
  real global bulkmodulus;
  // Defining initial conditions, rod position and pressures
  real global rodposinit, PAinit, PBinit;
  // Defining port dimensions, diameters, stroke and deadvolume for the cylinder
  real global dinlet, doutlet, dcyl, drod, Stroke, vdead;
  // Defining maximum pressure, deflection at maximum pressure and damping ratio at end strokes
  real global Pmax, deflection, dampingratio;//sequeece the end stopper to 0.0001m

variables

```

```

real Ainlet, Aoutlet;
real Apist, Arod;
real rodpos;
real VAinit, VBinit, VAoilinit, VBoilinit, VA, VB;
real kbumper, cbumper, Fbumper;

initialequations
// Calculating area of inlet and outlet ports
Ainlet = 0.25*pi*dinlet^2;
Aoutlet = 0.25*pi*doutlet^2;

// Calculating area on piston and rod side
Apist = 0.25*pi*dcyl^2;
Arod = Apist-0.25*pi*drod^2;

// Calculating initial physical oil volumes in A and B chamber
VAinit = vdead + Apist*rodposinit;
VBinit = vdead + Arod*(Stroke-rodposinit);
// Calculating initial oil quantity in A and B chamber
VAoilinit = PAinit*VAinit/bulkmodulus;
VBoilinit = PBinit*VBinit/bulkmodulus;

// Calculating stiffness and damping for bumper
kbumper = Pmax*Apist/deflection;
cbumper = dampingratio*2*sqrt(Pmax*Apist/9.81*kbumper);

equations
// Calculating rod position
rodpos = int(rodspeed,rodposinit);

//Calculating physical volume in A and B chamber
VA = vdead + Apist*rodpos;
VB = vdead + Arod*(Stroke-rodpos);

// Checking if minimum or maximum stroke has been reached
if rodpos> Stroke then
  Fbumper = kbumper*(rodpos-Stroke) + cbumper*max([0,rodspeed]); //Maximum stroke reached
else
  if rodpos< 0 then
    Fbumper = kbumper*(rodpos) + cbumper*min([0,rodspeed]); // Minimum stroke reached
  else
    Fbumper = 0.0; // Normal stroke - no bumper forces
  end;
end;

// Calculation results in a matrix
Cylcalc = [Ainlet;Aoutlet;VAoilinit;VBoilinit;VA;VB;Apist;Arod;Fbumper];

```

interface:

type	Submodel							
ports	name	domain	type	terminals	orientation	size	unit	description
	rodspeed	signal	real	1	in	1		
	Cylcalc	signal	real	1	out	[9,1]		

information:

name	Cylinder.Cylinder_calculations
Version	4.0
LibraryPath	Template\Submodel-Equation.emx
IsMainModel	1
KeepParameterValues	False
TimeStamp	2007-11-1 22:32:1
AllowLibraryUpdate	False
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

MSe EndStoppers**implementation:**

```

// Force when cylinder reaches maximum or minimum stroke (Damper and spring)
equations
  p.e = Cylcalc[9,1];

```

interface:

type	MSe

ports	name	domain	type	terminals	orientation	size	unit	description
	p	power	real	1	in	1		
	Cylcalc	signal	real	1	in	[9,1]		

restrictions	kind	priority	type	ports
	causality	fixed	out	p

information:

name	Cylinder.EndStoppers
LibraryPath	Bond Graph\MSe.emx
TimeStamp	2011-11-29 16:12:33
Version	4.2
IsMainModel	1
KeepParameterValues	False
AllowLibraryUpdate	True
name	DocumentationMask
name	MaskChoice
int	1
name	BitmapFilename
name	ScriptFilename
name	Scaling
double	1

R Leakage**implementation:**

```

parameters
  real global d cyl,Stroke;
  real viscosity=32e-6;
  real c=0.00001; //the clearance between piston and cylinder house

variables
  real flow;
equations
  p.phi = pi*d cyl*c^3/(3*viscosity*Stroke)*p.p;
  flow = p.phi*60e3;

```

interface:

type	R																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	hydraulic	real	1	in	1												

information:

name	Cylinder.Leakage
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

R Leakage1**implementation:**

```

parameters
  real global d cyl,Stroke;
  real viscosity=32e-6;
  real c=0.00001; //the clearance between piston and cylinder house

variables
  real flow;
equations
  p.phi = pi*d cyl*c^3/(3*viscosity*Stroke)*p.p;
  flow = p.phi*60e3;

```

interface:

type	R																
ports	<table border="1"> <thead> <tr> <th>name</th> <th>domain</th> <th>type</th> <th>terminals</th> <th>orientation</th> <th>size</th> <th>unit</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>p</td> <td>hydraulic</td> <td>real</td> <td>1</td> <td>in</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	1	in	1		
name	domain	type	terminals	orientation	size	unit	description										
p	hydraulic	real	1	in	1												

information:

name	Cylinder.Leakage1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\R.emx
TimeStamp	2011-11-29 16:35:37
AllowLibraryUpdate	True
name	DocumentationMask

1 OneJunction**implementation:**

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>any</td><td>none</td><td>1</td><td></td><td></td></tr> <tr><td>flow</td><td>signal</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
flow	signal	real	1	out	1																				
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>constraint</td><td>one_out</td><td>p</td></tr> </tbody> </table>	kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																						
causality	constraint	one_out	p																						

information:

name	Cylinder.OneJunction
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction1**implementation:**

```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction																								
ports	<table border="1"> <thead> <tr><th>name</th><th>domain</th><th>type</th><th>terminals</th><th>orientation</th><th>size</th><th>unit</th><th>description</th></tr> </thead> <tbody> <tr><td>p</td><td>hydraulic</td><td>real</td><td>any</td><td>none</td><td>1</td><td></td><td></td></tr> <tr><td>flow</td><td>signal</td><td>real</td><td>1</td><td>out</td><td>1</td><td></td><td></td></tr> </tbody> </table>	name	domain	type	terminals	orientation	size	unit	description	p	hydraulic	real	any	none	1			flow	signal	real	1	out	1		
name	domain	type	terminals	orientation	size	unit	description																		
p	hydraulic	real	any	none	1																				
flow	signal	real	1	out	1																				
restrictions	<table border="1"> <thead> <tr><th>kind</th><th>priority</th><th>type</th><th>ports</th></tr> </thead> <tbody> <tr><td>causality</td><td>constraint</td><td>one_out</td><td>p</td></tr> </tbody> </table>	kind	priority	type	ports	causality	constraint	one_out	p																
kind	priority	type	ports																						
causality	constraint	one_out	p																						

information:

name	Cylinder.OneJunction1
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction10**implementation:**

```
equations
```

```

sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);

```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction10
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction13**implementation:**

```

equations
sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);

```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction13
Version	4.2
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction15**implementation:**

```

equations
sum (direct (p.e)) = 0;
equal (collect (p.f));
flow = first (p.f);

```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	hydraulic	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction15
Version	4.2

IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2011-11-29 16:17:51

1 OneJunction3

implementation:

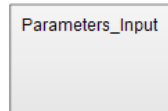
```
equations
  sum (direct (p.e)) = 0;
  equal (collect (p.f));
  flow = first (p.f);
```

interface:

type	OneJunction							
ports	name	domain	type	terminals	orientation	size	unit	description
	p	mechanical	real	any	none	1		
	flow	signal	real	1	out	1		
restrictions	kind	priority	type	ports				
	causality	constraint	one_out	p				

information:

name	Cylinder.OneJunction3
Version	4.0
Version	4.0
IsMainModel	1
KeepParameterValues	False
LibraryPath	Bond Graph\OneJunction.emx
TimeStamp	2007-9-27 9:51:18



Parameters_Input

implementation:

```
parameters
  real global bulkmodulus = 1.6e9;
  real global E = 180e9; //Youngp
```

Technical problem of Appendix C,

Due to a technical problem of 20sim software, the modelling code profile can not be exported. If the readers wish to review the modelling details of “Advanced models”, then please refer to the modelling details in 20sim files.