



AMERICAN UNIVERSITY OF BEIRUT

HYDRAULIC COMPONENT LIBRARY FOR COMBINED  
FORWARD/INVERSE SIMULATION APPROACH USING  
MODELICA

by  
JOSEPH GEORGES SAAD

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Mechanical Engineering  
to the Department of Mechanical Engineering  
of the Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
April 2014

AMERICAN UNIVERSITY OF BEIRUT

HYDRAULIC COMPONENT LIBRARY FOR COMBINED  
FORWARD/INVERSE SIMULATION APPROACH USING  
MODELICA

by  
JOSEPH GEORGES SAAD

Approved by:



---

Dr. Matthias Liermann, Assistant Professor  
Department of Mechanical Engineering

Advisor



---

Dr. Daniel Asmar, Assistant Professor  
Department of Mechanical Engineering

Member of Committee



---

Dr. Elie Shammas, Assistant Professor  
Department of Mechanical Engineering

Member of Committee

Date of thesis defense: April 11, 2014

AMERICAN UNIVERSITY OF BEIRUT


THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Saad Joseph Georges  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

  
Signature

April 11, 2014  
Date

## ACKNOWLEDGMENTS

I would like to thank all the individuals that stood next to me throughout my graduate years at AUB. It is because of you and your support that I can stand here today and say: I Did it!

I would like to send my gratitude to my thesis advisor Dr. Matthias Liermann who helped me in all the steps and the ups and downs that I have tackled throughout this thesis. It was not an easy task, but with his support and perseverance we were able to eventually overcome all the milestones faced.

Moreover, I would also like thank my thesis committee members, Dr. Daniel Asmar and Dr. Elie Shammass, for their vital encouragement and support.

I would like to send my acknowledgements to the American University of Beirut where I have spent my last couple of years working on this thesis and making great friends and memories.

I would like to thank Ms. Nadine Knesevitch for her help in laying out the format of this thesis.

Last but not least, my acknowledgements and gratitude goes to my family and my close friends who have always been there and supported me from day one. Your support was fruitful and delivered great results. I dedicate this thesis to you.

# AN ABSTRACT OF THE THESIS OF

Joseph Georges Saad for

Master of Mechanical Engineering

Major: Mechatronics Track

Title: Hydraulic Component Library for Combined Forward/Inverse Simulation Approach Using Modelica

Inverse dynamic simulation of hydraulic drives is helpful in early design stages of a hydraulic machine to answer the question whether the drive can meet dynamic load requirements and to predict the energy consumption for required load cycles. While a forward simulation of the hydraulic drive needs an implementation of the controller which generates the control input as a function of the control error, the inverse dynamic simulation can be implemented without control. This is because the required motion is simply defined as a constraint and therefore the control error is always zero.

This thesis illustrates in the literature review the techniques of equation manipulation to get the inverse dynamics. These techniques are implemented in simulators which use equation based modeling language such as Modelica. A simple forward/inverse simulation example of a hydraulic servo-drive is used in Modelica to illustrate the feasibility of these techniques on inverse simulation. After proving that the hydraulic servo-drive example can be simulated in a forward and inverse approach, this thesis proposes to amend one of the current hydraulic libraries in Modelica, `openHydraulics`, to become viable for inverse simulations.

The thesis explains the inverse simulation problems faced with the components of the `openHydraulics` library after modifying them to allow inverse simulations. The thesis concludes in a hydraulic servo-drive example of a gear hobbing machine actuator where the design of this actuator is optimized using the inverse simulation technique from components of the modified library. The modified hydraulics simulator library provides the design engineers with a tool to optimize their designs and produce more efficient systems without the need to tune a controller.

## CONTENTS

ACKNOWLEDGEMENTS.....	V
ABSTRACT.....	VI
LIST OF ILLUSTRATIONS.....	IX
LIST OF TABLES.....	XII
chapter	
I.THESIS OBJECTIVES.....	1
II.INTRODUCTION.....	2
A. Background & Motivation to use Forward/Inverse Simulation.....	2
B. Problem Statement & Solution Methodology .....	5
III.LITERATURE REVIEW .....	7
A. Survey on similar and related work .....	7
B. Equation Manipulation in Modelica .....	13
C. The difference between Equations, Algorithms, & Functions in Modelica...	25
1. Equations in Modelica .....	25
2. Statements/Algorithms in Modelica .....	27
3. Difference between Equations and Algorithms.....	28
4. Functions in Modelica .....	30
IV.PRELIMINARY RESULTS: INVERSE DYNAMIC SIMULATION OF A HYDRAULIC DRIVE .....	32

A. Modeling of Valve .....	33
B. Modeling of Cylinder.....	34
C. Forward and Inverse System Causalization.....	36
D. Problems with Existing Library .....	42
<b>V.INVERSION PROBLEMS IN MODELICA.....</b>	<b>44</b>
A. Inversion of Algorithms .....	45
B. Discontinuities .....	47
C. Saturation .....	48
D. Non-Monotony.....	49
E. Derivative function not found.....	50
F. Non-differentiable Inputs.....	52
<b>VI.OpenHYDRAULICS LIBRARY COMPENENTS &amp; THEIR     INVERSION PROBLEMS.....</b>	<b>54</b>
A. Valve .....	54
B. Cylinder.....	59
C. Pipeline Modifications .....	63
D. Oil Modifications .....	67
<b>VII.APPLICATION EXAMPE: VALVE/CYLINDER DESIGN     OPTIMIZATION.....</b>	<b>70</b>
<b>VIII.CONCLUSION,LIMITATIONS, &amp; FUTURE WORK .....</b>	<b>85</b>



Appendix

I. VALVE.....87

    A. Variable Restriction ..... 88

    B. Variable Restriction Series Valve ..... 90

    C. Second Order Block ..... 90

II. CYLINDER .....93

    A. Chamber Head..... 94

    B. Chamber Rod ..... 96

    C. Cushion ..... 97

        1. CushionTable.....98

        2. Relief Valve .....99

    D. Leakage ..... 100

BIBLIOGRAPHY .....102

# ILLUSTRATIONS

Figure	Page
1. Forward Simulation .....	3
2. Inverse Simulation .....	4
3. Forward simulation only v/s Backward/Forward simulation.....	5
4. Forward Model of Electro-mechanical Actuator (Bals, Hofer, Pfeiffer, & Schallert, 2003).....	8
5. Inverse Model of Electro-mechanical Actuator (Bals, Hofer, Pfeiffer, & Schallert, 2003).....	9
6. Translation of Model in Modelica .....	14
7. Electric Circuit .....	16
8. Causalization Steps .....	18
9. Causality Graph .....	18
10. Structural Incidence Matrix .....	19
11. Lower Triangular Form of Incidence Matrix.....	19
12. Inverse Electric Circuit .....	20
13. Causality Graph of Inverse System.....	21
14. Lower Triangular Form of Inverse System Incidence Matrix .....	22
15. Block Lower Triangular Form of Incidence Matrix .....	22
16. Servo-Hydraulic Linear Actuator .....	32
17. Backward Simulation Approach does not work in Simulink.....	41
18. Hydraulic Servo-Drive System in openHydraulics.....	42
19. Metering Table in openHydraulics Valve Model .....	43

20. Gear Hobbing Machine.....	43
21. example1 Results .....	47
22. Friction v/s Velocity Discontinuous .....	48
23. Limited Output.....	49
24. Friction Model Non-Monotonous .....	50
25. Inversion of Interpolation Table .....	51
26. Inversion of Interpolation Table Simulation.....	51
27. Non-differentiable Equation .....	52
28. Valve Model and Hierarchical Tree.....	55
29. P2A & B2T v/s P2B & A2T .....	56
30. Double Acting Cylinder Modeled.....	59
31. openHydraulics Double Acting Cylinder Model and Hierarchical Tree .....	60
32. Line with No Volume .....	63
33. Pipeline with Volume Model and Hierarchical Tree .....	63
34. Line_mod model .....	67
35. Gear Hobbing Cylinder Actuator.....	71
36. Gear Hobbing Valve/Cylinder Dymola Inverse Circuit .....	71
37. Cylinder Position Input, Velocity Profile, and External Force.....	73
38. Pressure Design Requirements .....	74
39. Pressure Distribution, Energy Consumption, and Valve Opening for Initial Configurations.....	75
40. Pressure Distribution, Energy Consumption, and Valve Opening for Cylinder 2....	78
41. Pressure Distribution, Energy Consumption, and Valve Opening for Cylinder 3....	80
42. Cylinder 3 Modified Valve Opening .....	81
43. Cylinder 3 with Larger Head Chamber Rod Diameter .....	82

44. Gear Hobbing Valve/Cylinder Dymola Forward Circuit.....	83
45. Reference Cylinder Position and Actual Control Output .....	83
46. Valve V4_3CC Model Tree .....	87
47. Variable Restriction .....	88
48. CombiTable1Ds .....	89
49. Variable Restriction Model Tree .....	89
50. Second Order Block.....	90
51. Inverse Second Order Block .....	91
52. Second Order Inverse Result .....	92
53. Cylinder Model Tree.....	94
54. Cylinder Chamber Head .....	95
55. Cushion Model.....	97
56. CombiTable1D.....	98
57. Cushion Table Plot.....	99
58. Relief Valve .....	99
59. Laminar Restriction Model .....	100

## TABLES

Table	Page
1. Forward Simulation Lower Triangular Form .....	37
2. Inverse Simulation Block Lower Triangular Form.....	38
3. Inverse Simulation Lower Triangular Form .....	39
4. Inverse Simulation Lower Triangular Form Larger System.....	39
5. Problems and Solutions.....	44
6. P2A & B2T Interpolation Table Parameters.....	56
7. P2B & A2T Interpolation Table Parameters.....	56
8. Parameters for Generic Oil density calculation .....	68
9. Cylinder Initial Configuration Parameters.....	72
10. Initial Valve Configurations .....	75
11. Cylinder 2 Parameters.....	76
12. Cylinder 3 Parameters.....	79
13. Valve Modified Parameter Configurations.....	80

# CHATER I

## THESIS OBJECTIVES

The thesis has three objectives. The first is to conduct a literature survey on equation handling techniques of equation based modeling language simulators and their use for inverse system simulation. The second objective is to modify the existing opensource hydraulic library, `openHydraulics`, found in Modelica in order to make it feasible for inverse simulations. The third objective is to perform a case study using the components from the modified library. The study is to optimize the design of a gear hobbing machine's hydraulic actuator using components of the modified library to illustrate the importance of inverse simulation.

## CHAPTER II

### INTRODUCTION

This section illustrates the forward and inverse simulation approaches, and discusses the role that the inverse simulation has on the design process in a forward/inverse approach. In addition, this section highlights the problems that the current hydraulic library in Modelica face when it comes to inverse simulations, and proposes to fix the `openHydraulics` library to make it compatible for inverse simulations.

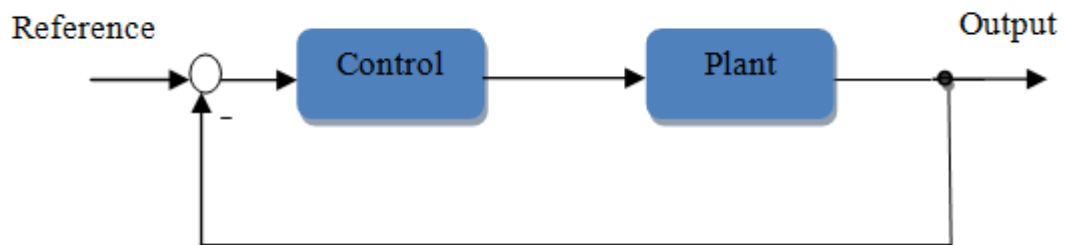
#### **A. Background & Motivation to use Forward/Inverse Simulation**

This section illustrates the differences in the simulation approaches and the importance of a combined forward/inverse simulation on designing new systems.

Mechanical systems have become complex and include many engineering disciplines. This urges companies to spend part of their budget for performing system simulations on their designs before they actually manufacture them.

Usually, system simulations are not introduced in early design stages. Rather they are implemented in later stages to test the dynamics of the system designed or to choose an optimum control that will provide the desired system output. The approach used to perform these simulations is commonly known as the forward simulation approach, where the states of the governing differential equations of the system are calculated in the direction of causality from given control and reference inputs to system outputs. Closed loop controlled systems require a forward simulation implementation that includes the control design. Figure 1 illustrates the forward approach where the

input to the system is the reference and the output is the plant output, such as the motion of a drive for example.



**Figure 1: Forward Simulation**

Design engineers use steady state conditions and engineering assumptions depending on their expertise to design their systems and then they test their systems dynamically to implement a suitable controller. Engineers would like to run simulations in early design stages before implementing a controller at a later stage. Simulations in the design stage aid the design engineers to select the optimal component sizing and parameters of their system; thus, helping to have a more reliable and efficient system once the controller is applied in later stages. Moreover in case the engineer requires to test the system efficiency, he is interested in the system's input and output variables and not in designing or tuning a controller. The forward approach can be considered as a drawback since the engineer will have to design a controller in closed loop systems to check the efficiency.

This is where the backward or inverse simulation approach is introduced and can have a major role for helping design engineers design their systems. Inverse system simulation is also beneficial in cases where the control design is difficult. Figure 2 illustrates the inverse simulation concept where the desired output is fed as an input to

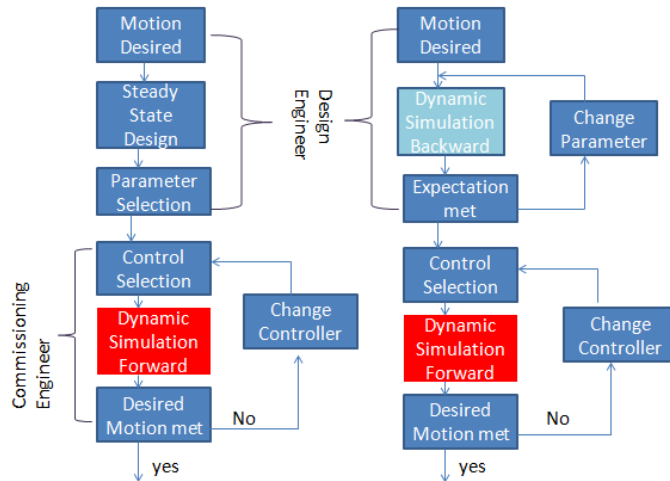


the system plant thus the output of the simulation corresponds to the input that should be fed to the system.



**Figure 2: Inverse Simulation**

Figure 3 illustrates the difference between the conventional forward simulation (on the left) used after the design process is performed through steady state analysis from the engineer's expertise and the backward/forward simulation (on the right) used during and after the design process. The conventional simulation uses only the forward approach after the design parameters have been already chosen by the design engineer and to select an optimal controller for the system. Therefore, everytime system parameters are changed, controller has to be tuned again. Whereas, the backward/forward simulation introduces the backward approach in the design stage to help the design engineer select improved design parameters for his system before the commissioning stage where the commissioning engineer applies the forward simulation approach once to select an optimal controller for the system. The advantage of the backward approach is providing a tool that helps the design engineer in selecting and sizing system components more efficiently without the need to tune a controller everytime the system parameters are changed.



**Figure 3: Forward simulation only v/s Backward/Forward simulation**

The backward/forward approach (on the right) of Figure 3 is desired in order to provide better systems. However, since the boundary conditions of the inverse simulation models are not the same as those of the forward simulation, the engineers usually have to build two separate models for each case.

It is desired to have tools that allow combined forward/inverse simulation to be performed using the same models.

## **B. Problem Statement & Solution Methodology**

Equation based object oriented modeling tools such as Modelica provide an important platform that allows combined forward/inverse simulations due to their acausal property. The problem is that eventhough Modelica allows this combined forward/inverse simulation approach, currently the two hydraulic libraries available in Modelica do not allow the inverse simulation for specific reasons that will be investigated in this thesis. One is developed by Modelon (Modelon, 2012-2013). The other one, which is called *openHydraulics*, is found for free in openModelica (Paredis,

2013). Both do not allow the inversion of hydraulic servo-drive systems using the same models; however, it will be shown later in the thesis that Modelica itself can handle the combined inverse/forward simulation of the hydraulic drive effectively.

Therefore, the aim of this thesis is to modify the opensource `openHydraulics` hydraulics library to enable the system models to be simulated in a combined forward/inverse fashion.

The methodology to reach the thesis goal considered is implemented accordingly. The first step is a literature review where the literature review itself is divided into 3 parts: related work, equation manipulation techniques in Modelica, and differences between equations, algorithms, and functions in Modelica. This is illustrated in section III. Afterwards, the second step is building a test case consisting of a hydraulic valve/cylinder system that is modeled in Modelica. This step, discussed in section IV, shows the feasibility of the forward and inverse simulation of such system in Modelica. The third step deals with investigating inverse simulation problems tackled through modifying the `openHydraulics` library in Modelica. This step is shown in sections V and VI. Finally, the last step is testing the feasibility of the modified `openHydraulics` library with a case study of a hydraulic valve/cylinder servo-drive system taken from components of this library. This study shows the importance of inverse simulation in designing more efficient and optimized systems which is the goal of the thesis. This is illustrated in section VII.

## CHAPTER III

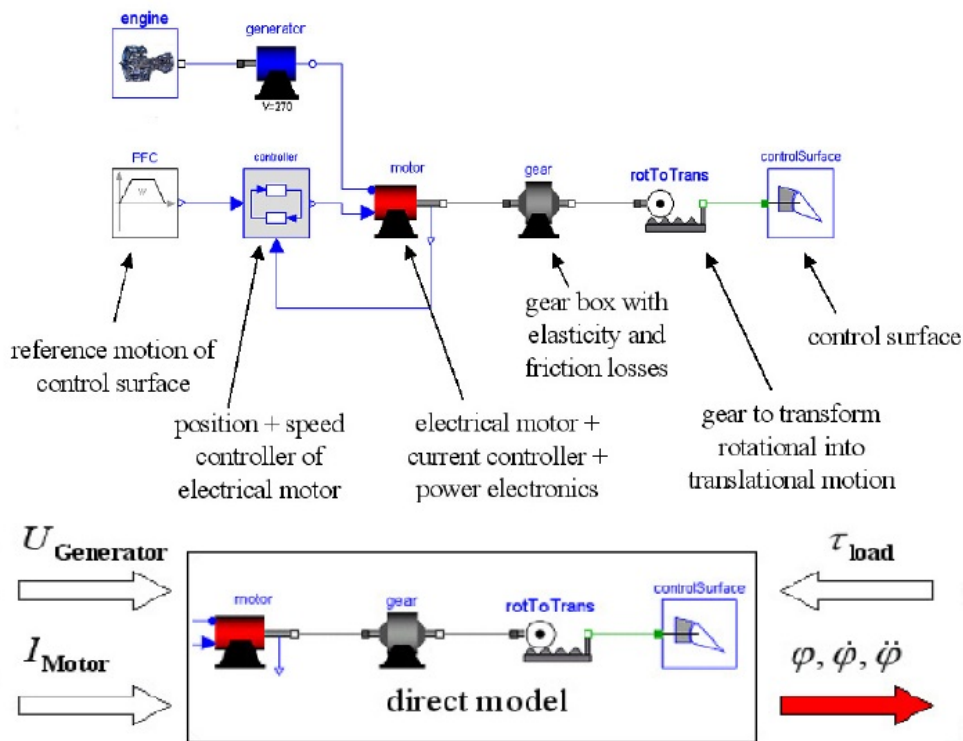
### LITERATURE REVIEW

The following is a literature survey that illustrates the use of inverse simulation in literature. It also explains the equation manipulation techniques used in Modelica at an example of an RL circuit which will be simulated both in a forward and inverse fashion.

#### **A. Survey on similar and related work**

The backward simulation approach from a modeling point of view is not much different from the forward approach. The only difference is that the system inputs and outputs are interchanged. The result of this interchange is still a system of DAEs, Differential Algebraic Equations, that can be solved with the same techniques of any DAE solver (Otter M., 2005). The inverse dynamic simulation is based on two main points which are: inverting the differential equations that describe the system and designing an output trajectory of the system (Froberg, 2006). This means that the output that is desired from the system is assumed to be met, and it is fed to the system as an input; whereas the calculated output of such a simulation are the real required inputs of the system. The benefit of the backward simulation approach is that the simulation does not need for a controller. There is low complexity in models where complicated controller designs from partner companies could be avoided (Bals, Hofer, Pfeiffer, & Schallert, 2003). Inverse simulation can be introduced into early design stages which will facilitate the selection of component sizing for the design engineers.

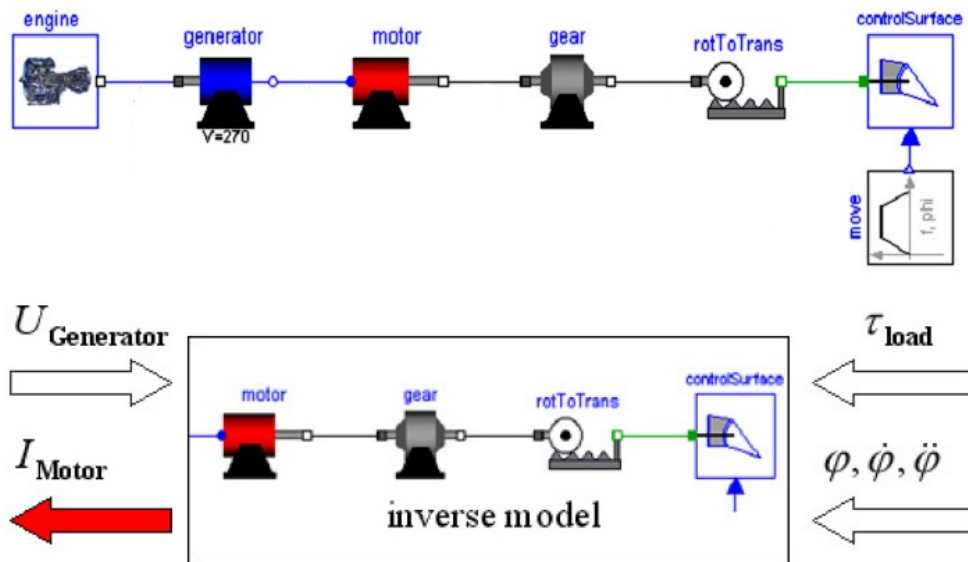
The inverse modeling approach in Bals, Hofer, Pfeiffer, & Schallert (2003) is illustrated at an example of an electro-mechanical actuator model used in an airplane in which both direct (forward) and inverse simulations were conducted using the same models. The figures of the following example are taken from Bals, Hofer, Pfeiffer, & Schallert (2003).



**Figure 4: Forward Model of Electro-mechanical Actuator (Bals, Hofer, Pfeiffer, & Schallert, 2003)**

Figure 4 illustrates the direct or forward model of the electro-mechanical actuator of an airplane. A control unit is required to set the voltage level that the motor should consume from the generator. The motor control unit provides a current command needed to move the control surface according to the reference motion of the control

surface. In its lower part, Figure 4 shows the inputs and outputs of the forward or direct modeling approach on the electro-mechanical actuator. It is seen that the inputs of the system are the motor current  $I_{Motor}$ , the generator voltage  $U_{generator}$ , and the load acting on the control surface  $\tau_{load}$ . Whereas, the outputs of the system are the dynamic motion of the control surface  $\varphi$ ,  $\dot{\varphi}$ , and  $\ddot{\varphi}$ . The consumed electrical power can be calculated from the actual motor current  $I_{Motor}$  and the actual motor voltage  $U_{Motor}$ . Moreover, the fuel consumption can be calculated by the generator model.



**Figure 5: Inverse Model of Electro-mechanical Actuator (Bals, Hofer, Pfeiffer, & Schallert, 2003)**

The inverse model as shown in Figure 5 on the other hand has the applied load at the control surface  $\tau_{load}$ , the predefined dynamic motion, and the generator voltage are inputs to the system; whereas, the output of the system is now the motor current  $I_{Motor}$ . The power is calculated in a similar fashion as that for the direct model.

The difference between inverse models and direct models is the absence of a controller which is not found in the inverse model. The presence of a controller implies a presence of errors in the actual output of the system. That is why the actual control surface position is slightly different than the predefined one  $\varphi$ . This error hardly affects the power consumption. An advantage of the inverse modeling approach is low complexity in models where complicated controller designs from partner companies are not found nor needed (Bals, Hofer, Pfeiffer, & Schallert, 2003).

The methods used to handle the inverse model DAE in Dymola are the same as those for the direct model. The input functions or trajectories to the system must be differentiable to a certain order. For the example illustrated, the input control surface position  $\varphi$  must be at least twice differentiable in order to be able to calculate  $\ddot{\varphi}$ . Therefore, the use of filters is important to ensure this differentiability.

Another paper from Otter M. (2005) deals with inverse models in control using Dymola. The aim of this paper is to design controllers for nonlinear systems using inversion of the plant model. The goal is to replace controllers based on linear inverse models with nonlinear inverse models which result in controllers that are valid for full operating regions of the plant. According to Otter M. (2005), if the models are implemented in Modelica, this results in highly automated control models working within the full operating range.

An inverse model of a DAE is obtained by interchanging the meaning of variables to obtain the real inputs as unknowns and real outputs as known. The result of this interchange is still a DAE. According to Otter M. (2005), an inverse model cannot be used in a controller unless its DAE is stable and has a unique solution. Furthermore, an inverse model should have the degree of its denominator at least equal or larger than

that of the numerator. Therefore, additional poles should be added or filters to fulfill this condition. An alternative way is to control the derivative of the output, not the output itself. In this way one will have an extra pole. This is seen by the following example (Otter M., 2005):

$$y = \frac{(s + 1)}{(s - 2) \cdot (s + 3)} \cdot u \quad \text{III-A-1}$$

The input of the system is  $u$  and the output is  $y$ . The inverse of III-A-1 is:

$$u = \frac{(s - 2) \cdot (s + 3)}{(s + 1)} \cdot y \quad \text{III-A-2}$$

It needs a minimum filter of order 1 for example  $\frac{1}{Ts+1}$  for the transfer function to exist and the inverse model to be correct.

$$u = \frac{(s - 2) \cdot (s + 3)}{(s + 1)} \cdot \frac{1}{Ts + 1} \cdot y \quad \text{III-A-3}$$

Or as discussed, one could control the derivative of the output.

$$\dot{y} = s \cdot y = \frac{s \cdot (s + 1)}{(s - 2) \cdot (s + 3)} \cdot u \quad \text{III-A-4}$$

==> the inverse of the system now is

$$u = \frac{s \cdot (s + 1)}{(s - 2) \cdot (s + 3)} \cdot \dot{y} \quad \text{III-A-5}$$

The inverse simulation approach has attracted attention for the solution of nonlinear control problems where the control response is of interest. It allows



investigation of characteristics needed in a control actuator to ensure that the overall system performance is not degraded (Murray-Smith, 2000). In case of open control when human is providing manual input to the system, the inverse approach can show if a particular task is beyond the human's capabilities due to large system required inputs. Inverse simulations are significant in aircraft flight control and in aircraft handling qualities investigations (Murray-Smith, 2000). The available methods of inverse simulation can be divided into techniques which involve numerical differentiation and iterative techniques which are based upon numerical integration processes. The methods based upon differentiation tend to be at least an order of magnitude faster than those which involve integration and the two approaches tend, therefore, to have different areas of application (Murray-Smith, 2000).

ADVISOR is an advanced vehicle simulator that was developed in 1994 by National Renewable Energy Laboratory to support the US Department of Energy hybrid propulsion system (Markel, 2002). ADVISOR simulates vehicle performance on standard driving cycles between 2.6 and 8 times faster than a representative forward-facing model due to its combined backward/forward approach (Wipke, 1999). Depending on the level of details of the components modeled, the ADVISOR model is considered as a steady state model (Wenzhong, Mi, & Emadi, 2007). The backward model does not require a driver; instead, the acceleration force needed is calculated directly from the desired velocity profile where the torque on the wheel could be computed and calculations run backward component by component to reach the electric energy input needed to provide the resulting speed profile. System loss and efficiency could directly and explicitly be calculated. Moreover, simulations execute quickly using backward simulation approach (Wipke, 1999). One of the weaknesses of the backward

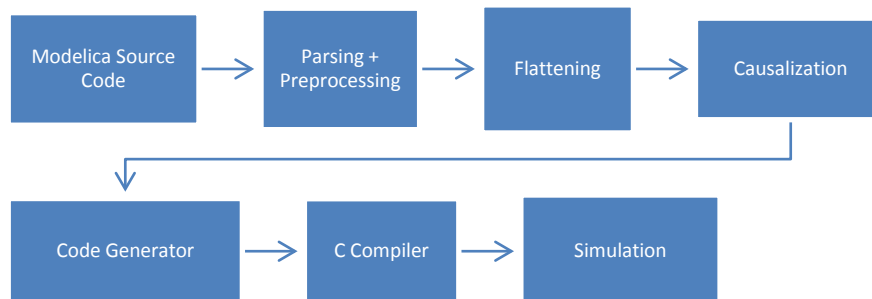
facing approach in ADVISOR is the assumption that the output is met which makes it not suitable to compute the "best effort performance" produced when accelerations of the speed trace exceeds the capabilities of the drivetrain. Moreover, the backward facing approach in ADVISOR uses efficiency maps that are produced by steady-state testing where dynamic effects are not included in the maps which estimates the backward facing model energy use.

This literature survey has illustrated some of the uses of inverse simulation in literature. It shows that inverse simulation has been used in the design process of systems and in enhancing control designs for nonlinear systems. According to literature, inverse simulation is often used in the aerospace industry for testing and designing better airplane components. However, some of the limitations that made the inverse simulation not used more often are that not all systems are invertible and approximations sometimes need to be considered in order to avoid discontinuities that will render the system irreversible.

## **B. Equation Manipulation in Modelica**

A tool that helps the user to build the same model and use it in combined forward/inverse simulation is Modelica, an equation-based object oriented modeling language. The benefit of Modelica is that it uses acausal physical modeling style where the models are defined based on equations rather than assignment statements (Fritzson, 2004). The power of equation based models is that they do not specify a priori which variables are declared as inputs and which are outputs (Fritzson, 2004); however, using assignment statements, variables that are assigned on the left-hand side of an expression are the outputs of the system and variables on the right-hand side are the assigned inputs

of the system. Simulink is one of the tools that is based on variable assignment. Using equation-based object oriented modeling tools such as Modelica facilitates the use of backward simulation approach, because the user only builds one model and can simulate it either in the forward approach or in the backward approach by flipping the system inputs. However, the use of traditional variable assignment tools such as Simulink will require the user to build two separate models, one for the conventional forward approach and the other for the backward approach. As is seen from this section, obtaining an inverse model is not always simple rather involves inversions and tedious manual writing of the equations required to model the system inverse (Åström, Elmqvist, & Mattsson, 1998). The process of how Modelica handles equations is illustrated in Figure 6.



**Figure 6: Translation of Model in Modelica**

First the model is created by the user using Modelica source code with equations as main components. Upon simulation instance, the model is parsed to check whether the use of the Modelica language is correct or wrong. After parsing, preprocessing takes place where the parser will check whether the classes are used correctly such as the `extend` command that connects different submodels. After language

and type checking, flattening takes place. In flattening, the hierarchy of the model structure is destroyed; thus, all the parameters, variables, and equations from all the component models of the system are collected in one global set. This set contains all the DAEs of the system that will be computed in the simulation with standard solvers (Zimmer, 2011).

A system of DAE is represented implicitly in the following form:

$$0 = F \left( \frac{dx(t)}{dt}, x(t), u(t), y(t), t \right)$$

The goal is to transform the implicit DAE to an explicit state-space representation form that is suited to most ODE solvers.

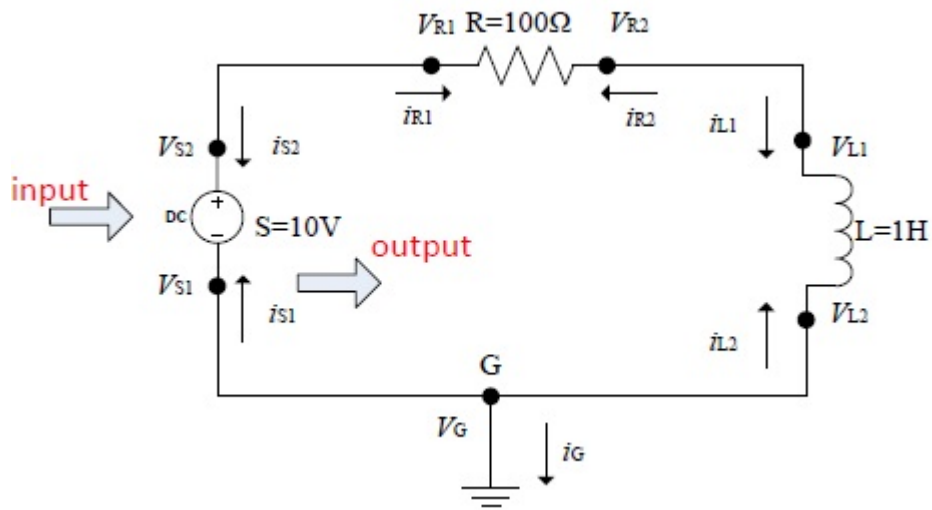
$$\dot{x}(t) = f(x(t), y(t), u(t))$$

$$y(t) = g(x(t), u(t))$$

where,

- $u(t)$  is the vector of input variables
- $x(t)$  is the vector of state variables
- $y(t)$  is the vector of output variables and also include algebraic variables
- $\dot{x}(t)$  is the derivative of the state vector

The following example illustrates the way the equations are automatically manipulated in Modelica. Figure 7 shows an electric circuit. The circuit consists of a voltage source providing an input alternating voltage amplitude of 10 V with a frequency of 10 Hz, a resistor of resistance  $R=100 \Omega$ , and an inductor of inductance  $L=1$  H. The output of the system is considered as the circuit current.



**Figure 7: Electric Circuit**

After the system model is flattened, the resulting equations are shown below.

$$\begin{array}{cccc}
 V_{S2} = V_{R1} & V_{R2} = V_{L1} & V_{L2} = V_G & V_{S1} = V_G \\
 i_{S2} + i_{R1} = 0 & i_{R2} + i_{L1} = 0 & i_{L2} + i_{S1} + i_G = 0 & V_G = 0 \\
 i_{R1} + i_{R2} = 0 & U_R = R \cdot i_{R1} & V_{R1} + U_R = V_{R2} & i_{S2} + i_{S1} = 0 \\
 V_{S1} + 10 = V_{S2} & i_{L1} + i_{L2} = 0 & U_L = L \cdot \frac{di_{L1}}{dt} & V_{L1} + U_L = V_{L2}
 \end{array}$$

The first thing that is done is to eliminate all redundant equations or trivial ones such as  $V_{L2} = V_G$  and leave out all necessary ones. The result of this procedure is the 9 equations with their respective variables as shown below. The equations are numbered in order to be sorted afterwards.

$$\text{Eq.1 } V_G = 0$$

$$\text{Eq.2 } U_R = R \cdot i_{R1}$$

$$\text{Eq.3 } V_{R1} + U_R = V_{L1}$$

$$\text{Eq.4 } V_G + 10 = V_{R1}$$

$$\text{Eq.5 } U_L = L \cdot \frac{di_{L1}}{dt}$$

$$\text{Eq.6 } V_{L1} + U_L = V_G$$

$$\text{Eq.7 } -i_{S1} + i_{R1} = 0$$

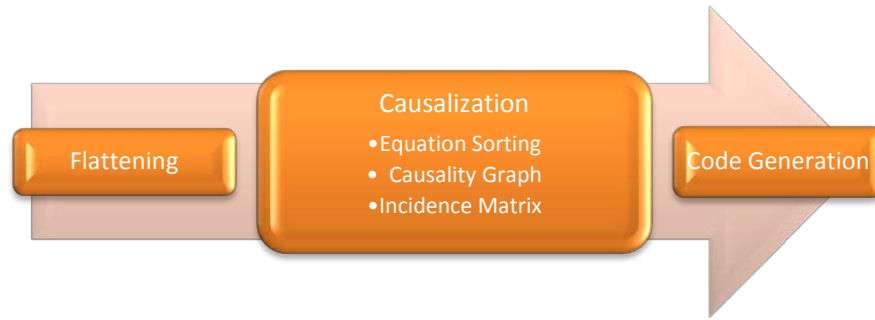
$$\text{Eq.8 } -i_{R1} + i_{L1} = 0$$

$$\text{Eq.9 } -i_{L1} + i_{S1} + i_G = 0$$

After eliminating trivial equations the system is transformed into explicit state-space form where the derivative of the state is explicitly written in terms of the state itself and other related variables. In this case, the only state is  $i_{L1}$  ; therefore, the explicit state equation becomes:

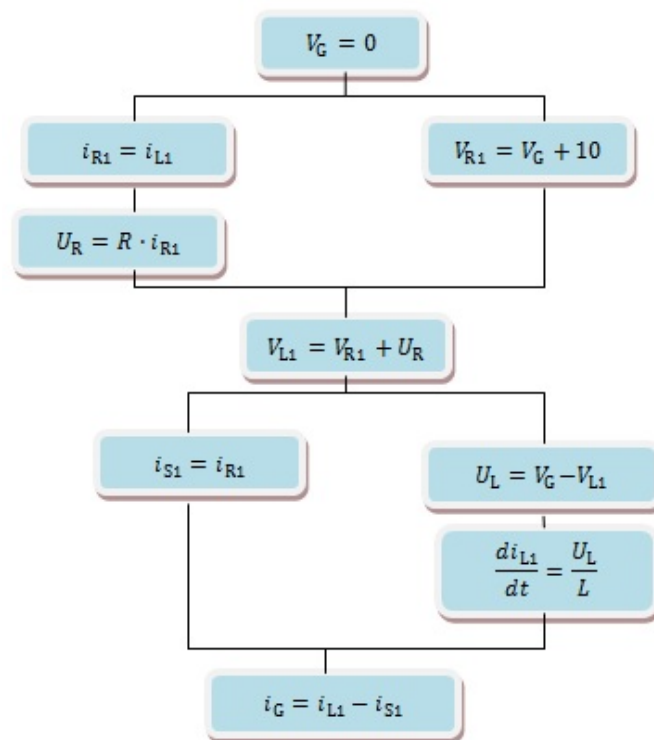
$$\frac{di_{L1}}{dt} = \frac{U_L}{L}$$

After explicitly defining the state equations, the causalization step takes place, Figure 8. This step deals with causalizing the equations; thus, indicating which equation determines which unknown and step by step build the hierarchy of the solution. This step is divided into three parts, displayed in Figure 8: sorting of equations according to the precedence of solution, constructing the causality graph, and building the structural incidence matrix.



**Figure 8: Causalization Steps**

The causality graph is an acyclic directed graph that represents the causality of the system. Figure 9 illustrates the causality graph of the example system.



**Figure 9: Causality Graph**

The non-causal list of equations is represented in a matrix form called structural incidence matrix. An example of such a matrix is shown in the Figure 10.

	$i_G$	$i_{S1}$	$U_L$	$i_{R1}$	$V_G$	$V_{L1}$	$V_{R1}$	$U_R$	$\frac{di_{L1}}{dt}$
Eq.1					X				
Eq.2				X				X	
Eq.3						X	X	X	
Eq.4					X		X		
Eq.5			X						X
Eq.6			X		X	X			
Eq.7		X		X					
Eq.8				X					
Eq.9	X	X							

**Figure 10: Structural Incidence Matrix**

As seen from Figure 10, the rows of the matrix represent the equations and the columns of the matrix represent the equation variables or unknowns and output. In other words, the columns represent vectors  $\dot{x}(t)$  and  $y(t)$ . The states are considered known.

The aim of the causalization as discussed earlier is to sort the equations in a hierarchical fashion. Therefore, the best consequence that is obtained after manipulation of the causality graph Figure 9 is a lower triangular form (LT) of the incidence matrix.

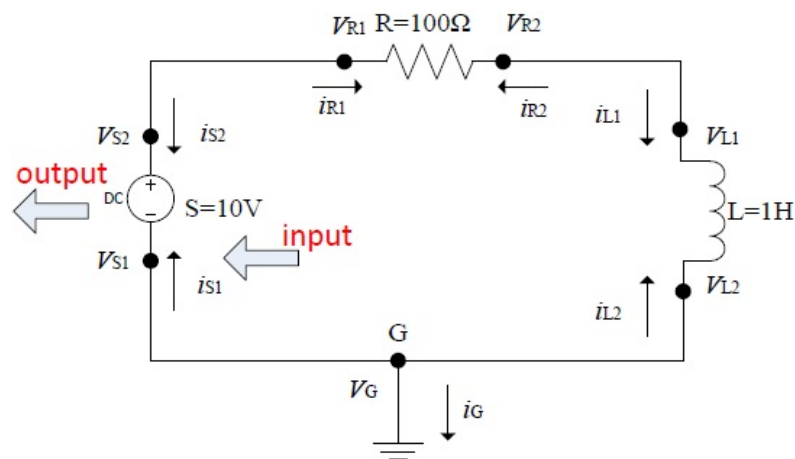
This can be seen in Figure 11.

	$V_G$	$V_{R1}$	$i_{R1}$	$U_R$	$V_{L1}$	$i_{S1}$	$U_L$	$\frac{di_{L1}}{dt}$	$i_G$
Eq.1	X								
Eq.4	X	X							
Eq.8			X						
Eq.2			X	X					
Eq.3		X		X	X				
Eq.7			X			X			
Eq.6	X				X		X		
Eq.5							X	X	
Eq.9						X			X

**Figure 11: Lower Triangular Form of Incidence Matrix**



It is shown from Figure 11 that the causalization of the equations results in a permutation of the structure index matrix to transform into lower triangular form. The solution is straightforward by simply replacing the variables in a forward manner. This LT form follows the same order of causality obtained from the acyclic directed graph. The solution is directly performed through forward substitution of variables through the chronology obtained from the LT matrix and with any ODE solver such as Forward-Euler to increment the states for the next time step.



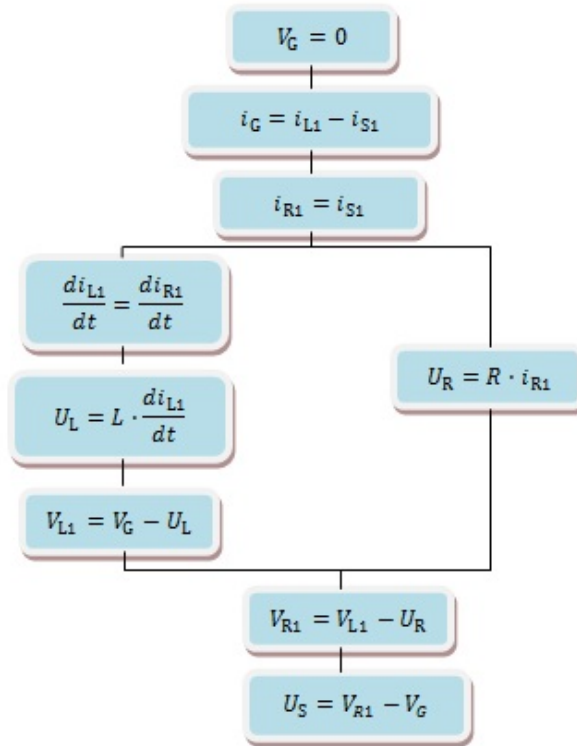
**Figure 12: Inverse Electric Circuit**

In the same manner as previously described, the inverse of this system is solved. Figure 12 shows the inverse circuit which is the same as the forward circuit model with interchanging the input/output boundary conditions. In this case, the input is the current  $i_{S1}$  and the output is the voltage source  $U_S$ . The same equations are used, but by replacing the new input and the new output. Modelica derives equation Eq.8 to become a function of the state derivative  $\frac{di_{L1}}{dt}$  which is considered as an unknown and mandatory to proceed in the solution process. This process is called Dummy-derivative

method used in cases where system is structurally singular due to constraint state equations where one of the state derivatives is considered as an algebraic variable and called dummy derivative (Mattson & Söderlind, 1993). Therefore, Eq.8 becomes Eq.8i shown.

$$\text{Eq.8i} \quad \frac{di_{L1}}{dt} = \frac{di_{R1}}{dt}$$

The new causality graph is shown in the Figure 13.



**Figure 13: Causality Graph of Inverse System**

The LT incidence matrix is then formed and shown in Figure 14.

	$V_G$	$i_G$	$i_{R1}$	$\frac{di_{L1}}{dt}$	$U_L$	$U_R$	$V_{L1}$	$V_{R1}$	$U_S$
Eq.1	x								
Eq.9		x							
Eq.7			x						
Eq.8i				x					
Eq.5				x	x				
Eq.2		x				x			
Eq.6	x				x		x		
Eq.3						x	x	x	
Eq.4	x							x	x

**Figure 14: Lower Triangular Form of Inverse System Incidence Matrix**

However, not all systems can be permuted into LT form. The forward causalization and the LT form can only be used for simple problems. For the vast majority of the other cases, the BLT which stands for the Block Lower Triangular form is used. The BLT form is close to the LT form where the blocks of the BLT matrix at the diagonal are as small as possible. An example of the BLT matrix is shown in the Figure 15.

	$V_G$	$V_{R1}$	$i_{R1}$	$U_R$	$V_{L1}$	$i_{S1}$	$U_L$	$\frac{di_{L1}}{dt}$	$i_G$
Eq.1	x								
Eq.4	x	x							
Eq.8			x	x					
Eq.2			x	x					
Eq.3		x		x	x				
Eq.7			x			x			
Eq.6	x				x		x	x	x
Eq.5							x	x	x
Eq.9						x			x

Block of Coupled Equations

**Figure 15: Block Lower Triangular Form of Incidence Matrix**

Matrices that cannot be permuted into LT form such as in Figure 15 are solved by isolating highlighted blocks on the diagonal of the matrix of Figure 15 and

considering them as coupled systems. The result of these isolations is a similar diagonal as the LT and can be solved by simple forward substitution on an acyclic directed graph.

Dymola solves coupled systems either through symbolic manipulations of equations which will transform the BLT form to an LT form or it solves for these isolated blocks alone in a process called tearing (Elmqvist & Otter, 1994). The concept of tearing is to assume a set of variables of an isolated block to be known, where these variables are called tearing variables, and to solve for the unknowns in the block in a causalized fashion. Some equations within a block might be overconstrained, they are considered as residual equations and thus the solution is not direct but is solved numerically with an iterative solver.

An example of how blocks could be solved:

1.  $a + b = 4$
2.  $a \cdot b = c$

$a$  and  $b$  are unknowns; whereas,  $c$  is known. In order to solve the block,  $a$  is assumed to be known; thus it is called a tearing variable. After this, the equations are resorted and causalized to solve for  $b$  iteratively as follows:

1.  $b = 4 - a = f(a)$
2.  $residual = c - a \cdot b = g(a, b)$

For every  $b$ , we get a different *residual* value where the aim is to minimize the *residual* value. Many methods can be used to solve  $0 = f(x)$  and the mostly used method is the Newton's method. The Newton's method deals with solving for the roots of  $x$  from the equation  $0 = f(x)$  where an initial guess for  $x_0$  is required. The algorithm finds a new value of  $x$  using the following equation (Lindfield & Penny, 2012):

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

If  $y = f(x) = residual$  is smaller than a certain tolerance value, the iteration is stopped and  $x_{n+1}$  becomes the final value otherwise the iteration is repeated to find a new value of  $x_{n+1}$ . Pseudo code:

```

while  $\left| \frac{g(a_n)}{g'(a_n)} \right| > \text{tolerance}$ 
     $a_{n+1} = a_n - \frac{g(a_n)}{g'(a_n)}$ ;
     $iteration = iteration + 1$ ;
     $a_n = a_{n+1}$ ;
     $\frac{g(a_n)}{g'(a_n)}$ ;
end

```

After manipulating the equations and bringing the DAEs into a flattened and sorted causalized system of ODE equations, C code is generated and with the help of any numeric equation solver such as Forward-Euler, the equations of the system model are solved and the simulation results are plotted. Therefore, the last 3 steps of Figure 6 are found in other simulation softwares even if they are based on variable assignment; however, the first 4 steps which are autonomous and important, are found in Modelica compilers. Thus, Modelica modeling language is a valuable tool to perform inverse or backward simulations.

## C. The difference between Equations, Algorithms, & Functions in Modelica

This section illustrates the differences between the equations, algorithms, and functions in Modelica. Some concepts in this section are taken from (Fritzson, 2004).

### *1. Equations in Modelica*

Many common simulation tools such as Matlab and LabView use assignment statements to represent equations found in models. Equation-based object oriented language such as Modelica uses equations as its main representation feature of models. Equations are more flexible to use because they do not have a specific data flow direction or execution order.

The equations provide the acausal feature in Modelica in which the same equation is used regardless of the assigned input and output. For example, the Ohm's Law equation is considered:

$$U=R \cdot i$$

It is written in the same fashion in Modelica no matter what the inputs/outputs are. This equation could be expressed in 3 ways if assignment is to be used depending on the input/output. If voltage is to be calculated from the current intensity and resistance, the assignment equation is:

$$U := R \cdot i$$

If current intensity is to be computed from voltage and resistance, the assigned equation becomes:

$$i := U/R$$

If resistance is to be computed from current intensity and voltage, the equation becomes:

$$R:=U/i$$

In Modelica, equations are classified into 4 groups depending on where they occur:

1. Normal Equations
2. Declaration Equations
3. Modification Equations
4. Initial Equations

Normal equations are found under the equation sections in the Modelica code under the keyword `equation` and terminated by other keywords such as `end`:

```
equation
  . . . . .
  <equations>
  <other keywords>
end
```

Declaration equations are used to declare parameters and constants. For example:

```
constant Integer two = 2;
parameter Real resistance = 200;
```

A declaration equation always holds the meaning that the value given to the parameter will not change during the simulation.

Modification equations are used to assign attributes. This means to assign the initial starting value of an equation at the beginning of the computation. For example:

```
Real frequency (start=200);
```

Initial equations section is used as part of the model in order to set an initial condition constraint to the system. It is declared by an initial equation as follows:

```
initial equation
  <equation>
```

## ***2.Statements/Algorithms in Modelica***

Equations are well suited for physical modeling, but there are situations where computations are better expressed as algorithms, i.e., sequences of statements.

Statements are imperative constructs allowed in algorithm sections. In Modelica, an algorithm section starts with the word `algorithm`. To terminate an algorithm section, one of the following keywords can be used: `public`, `protected`, `algorithm`, `equation`, **or** `end`.

The following is an example of an algorithm section.

```
algorithm
  . . . . .
  <statements>
  <other keywords>
```

An example of an algorithm section between two equation sections is illustrated as follows:

```
equation
  a=b*3;
  c=d;
algorithm
  a1:=c+a;
  a2:=b-4;
  a1:=a2+b;
equation
  e=a1-a2;
  . . . . .
end
```

It is noted that within the algorithm section, certain values of variables are taken from outside the algorithm. The values are called input variables. In the previous example, they are `a`, `b`, and `c`. The variables that receive their values from the algorithm



are called outputs of the algorithm and they must be assigned explicitly in terms of the inputs. In the previous example, they are  $a_1$  and  $a_2$ .

### ***3. Difference between Equations and Algorithms***

After providing a highlight on the equation and algorithm parts respectively, this section will demonstrate some differences between them.

The first difference is that in an equation section, each equation is used in simulating a model; whereas, in an algorithm section which is based on assignments, one can overwrite a previous assignment and thus ending up with only the last result which is needed to proceed with the solution of the model. The following example will illustrate this concept.

```
equation
m = n;
m = p;
```

There are 2 equations in this equation section. Both are included in the solution process of the model.

```
algorithm
m := n;
m := p;
```

However, in this algorithm section, the second assignment statement will overwrite the first one since they represent the same variable and thus  $m:=p$  will only be included in the solution process of the model disregarding  $m:=n$ .

An algorithm can be thought of like a box that takes some variables from the outside, assigns them in a certain fashion, and provides them as outputs to the model. In other words, the assignment statements inside an algorithm section are not important.

Only the outputs from the algorithm section are. The following example illustrates this point.

```
algorithm
m := 0;
n := 1;
for i in 1:5 loop
  m := m + x[i];
  n := n * x[i];
end for;
```

$m$  and  $n$  can be seen to be assigned several times due to the for loop, but at the end the only output that is taken from this algorithm are these 2 assignments:

```
algorithm
m := 1+2+3+4+5;
n := 1*2*3*4*5;
```

This is similar if replaced by these 2 equations:

```
equation
m = 1+2+3+4+5;
n = 1*2*3*4*5;
```

However, if an equation section is to be used initially instead of the algorithm section then this becomes:

```
equation
m = 0;
for i in 1:5 loop
  m = m+i;
```

This would lead to an error, because the system will expand to 6 equations with only 1 variable:

```
equation
m = 0;
m = m + 1;
m = m + 2;
m = m + 3;
m = m + 4;
m = m + 5;
```

Therefore, as a small summary, in an algorithm, it does not matter how many times an equation is assigned a value, what only matters is the final assigned equation

value; however, in an equation section, each equation counts and remains as part in the solution of the whole model.

#### ***4.Functions in Modelica***

Functions are used in almost every mathematical model. Considering Modelica, general functions used in mathematics such as `abs`, `sqrt`, `mod`, etc. are predefined within the language; whereas, other mathematical functions such as `sin`, `cos`, `exp`, etc. are found in the Modelica standard math library `Modelica.Math`. The standard arithmetic operators `+`, `-`, `*`, `/` are also considered as functions and are used directly through the language.

In Modelica, users can define their own functions. The body of a function contains an algorithm section containing assignment statements that are executed once the function is called. The inputs to the function are denoted by the keyword `input`; whereas, the results of the function are denoted by the word `output`. A Modelica function has no memory and always returns the same results given the same arguments. Below is an example of the structure of a function:

```
function f
  input  Real a;
  output Real b, c;
algorithm
  b:= a;
  c:= 3+b;
  b:= c+b;
end f;
```

There is an inter-relation between the algorithm, equation, and function. A function's body is written by an algorithm. However, in order to call a function in a model after this function is already defined, an equation is used. This equation equates the output results of the function with its inputs through the function's name. Therefore,

the correlation between these three terms is noted. The equation calling the function of the above example is shown below:

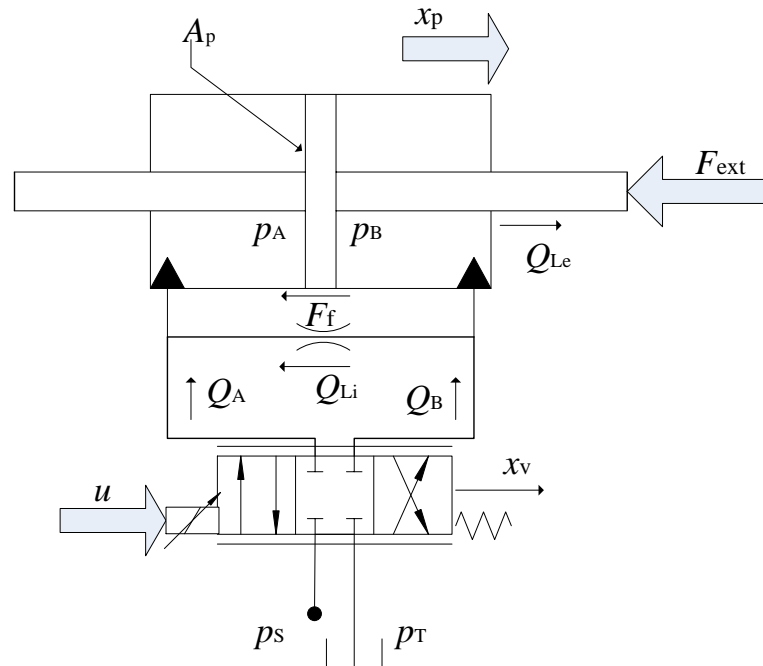
$$(b, c) = f(a);$$

...

## CHAPTER IV

### PRELIMINARY RESULTS: INVERSE DYNAMIC SIMULATION OF A HYDRAULIC DRIVE

Servo-Hydraulic linear axes are used to control position, velocity, or force in applications that require high power-to-weight ratio and system reliability in harsh environmental conditions. It is considered to be a good application to implement both the forward and backward approaches on and demonstrate their advantages. The system studied in this example consists of a synchronizing cylinder and a 4/3 directional control valve, see Figure 16.



**Figure 16: Servo-Hydraulic Linear Actuator**

The equations to describe the system dynamics can be found in many standard textbooks such as (Meritt, 1967), (Watton, 2009), and (Jelali & Kroll, 2003).

## A. Modeling of Valve

The valve modulates the power provided by the pressure source and delivered to the cylinder. As shown in Figure 16, the positive sense for the spool motion of the valve is assumed to the right. This valve is assumed to be critically lapped. Therefore, the flow orifice equations to the cylinder chambers are:

$$Q_A = c_v \text{sg}(x_v) \text{sign}(p_s - p_A) \sqrt{|p_s - p_A|} \dots$$

$$\dots - c_v \text{sg}(-x_v) \text{sign}(p_A - p_T) \sqrt{|p_A - p_T|}$$
**IV-A-1**

$$Q_B = c_v \text{sg}(-x_v) \text{sign}(p_s - p_B) \sqrt{|p_s - p_B|} \dots$$

$$\dots - c_v \text{sg}(x_v) \text{sign}(p_B - p_T) \sqrt{|p_B - p_T|}$$
**IV-A-2**

Where

$$\text{sg}(x_v) = \begin{cases} x_v & \text{for } x_v \geq 0 \\ 0 & \text{for } x_v < 0 \end{cases}$$
**IV-A-3**

The sign of  $x_v$  dictates the direction of flow in the system. The valve spool piston has dynamic characteristics with respect to the electrical input. Manufacturers' catalogues show frequency response plots which can be used to identify the dynamics of the spool position control system. A 2<sup>nd</sup> order approximation is sufficient:

$$\frac{1}{\omega_v^2} \ddot{x}_v + \frac{2D_v}{\omega_v} \dot{x}_v + x_v = K_v u$$
**IV-A-4**

## B. Modeling of Cylinder

Figure 16 shows the cylinder to be modeled. Introducing the positive sense for the velocity of the cylinder to be to the right and assuming that whatever enters the system is positive and whatever leaves the system is negative, the equations used for modeling the cylinder are illustrated below.

From the continuity equation, the flow in every chamber of the cylinder is as follows:

$$Q_A + Q_{Li} = \dot{V}_A + \frac{V_A}{E'(p_A)} \dot{p}_A \quad \text{IV-B-1}$$

$$Q_B - Q_{Li} - Q_{Le} = \dot{V}_B + \frac{V_B}{E'(p_B)} \dot{p}_B \quad \text{IV-B-2}$$

Where  $Q_{Li}$  and  $Q_{le}$  are the internal and external leakage flow. The volumes of the chambers are given by:

$$V_A = V_{A0} + x_p A_p \quad \text{IV-B-3}$$

$$V_B = V_{B0} - x_p A_p \quad \text{IV-B-4}$$

$V_{A0}$  and  $V_{B0}$  are the initial chamber volumes and in our case they are equal because the piston is considered referenced at the cylinder's center.  $A_p$  is the area of the piston.

The pressure dynamics equations are as follows:

$$\dot{p}_A = \frac{1}{c_{hA}} (Q_A - A_p \dot{x}_p) \quad \text{IV-B-5}$$

$$\dot{p}_B = \frac{1}{c_{hB}} (Q_B + A_p \dot{x}_p) \quad \text{IV-B-6}$$

The hydraulic capacitances of each of the 2 chambers are:

$$C_{hA} = \frac{V_{pl,A} + (\frac{S}{2} + x_p)A_p}{E'_A(p_A)} \quad \text{IV-B-7}$$

$$C_{hB} = \frac{V_{pl,B} + (\frac{S}{2} - x_p)A_p}{E'_B(p_B)} \quad \text{IV-B-8}$$

Where  $S$  is the stroke of the cylinder. Newton's 2<sup>nd</sup> law is applied in order to obtain the equation of motion for the cylinder.

$$m_t \ddot{x}_p + F_f(\dot{x}_p) = (p_A - \alpha p_B)A_p - F_{ext} \quad \text{IV-B-9}$$

Where  $m_t$  is the total mass and consists of the piston mass  $m_p$  and the hydraulic fluid mass in the cylinder chambers and the pipelines which are  $m_{A,fl}$  and  $m_{B,fl}$ . In general, the fluid mass is considered very negligible with respect to the piston mass which will lead to the assumption of  $m_t = m_p$ .

The Stribeck friction curve is considered in our case for the friction. The problem with friction when it comes to modeling is at zero velocity where it is discontinuous. It has an equal maximum positive and negative value depending on the direction of travel. The equation is as follows:

$$F_f(\dot{x}_p) = \sigma \dot{x}_p + \text{sign}(\dot{x}_p) [F_{c0} + F_{s0} e^{\frac{-|\dot{x}_p|}{c_s}}] \quad \text{IV-B-10}$$

where  $\sigma$  is the viscous friction coefficient,  $F_{c0}$  is the coulomb friction,  $F_{s0}$  the static friction, and  $c_s$  the stribeck velocity. In order to make it continuous and



monotonous, an approximation can be used (Liermann, 2012). This approximation makes the function invertible:

$$\text{sign}(\dot{x}_p) \approx \frac{2}{\pi} \cdot \arctan(\gamma \dot{x}_p) \quad \text{IV-B-11}$$

thus,

$$|\dot{x}_p| = \dot{x}_p \frac{2}{\pi} \cdot \arctan(\gamma \dot{x}_p) \quad \text{IV-B-12}$$

### C. Forward and Inverse System Causalization

Based on the equations of the models described above, the next step by comparing with the Modelica translation process described in Figure 6 is to write them in explicit form.

For the forward approach, the input to the system is the valve opening  $x_v$  and the output is the piston position. The explicit state differential equations in this case are as follows:

$$\dot{p}_A = \frac{1}{c_{hA}} (Q_A(x_v, p_A) - A_p \dot{x}_p + Q_{Li}) \quad \text{IV-C-1}$$

$$\dot{p}_B = \frac{1}{c_{hB}} (Q_B(x_v, p_B) + A_p \dot{x}_p - Q_{Li}) \quad \text{IV-C-2}$$

$$\ddot{x}_p = \frac{1}{m_t} [(p_A - \alpha p_B) A_p - F_{\text{ext}} - F_f(\dot{x}_p)] \quad \text{IV-C-3}$$

where  $Q_A$  and  $Q_B$  are functions of  $x_v$  as illustrated by equations IV-A-1 and IV-A-2 respectively.

The incidence matrix for the forward simulation is created and is shown in Table 1. The piston position  $\ddot{x}_p$  is calculated first from the initial conditions of the pressures  $p_A$  and  $p_B$  of the cylinder chambers. Then, the new pressure derivatives are calculated as shown by equations IV-C-1 and IV-C-2 in Table 1. Any numeric solver can be applied to calculate the next state obtained by the next time increment. The easiest solver which is applied in our case is the Forward-Euler solver that is based on calculating the next time state from the previous state and its derivative.

$$y(t + \Delta t) = y(t) + \Delta t \cdot \frac{dy}{dt}$$

	$\ddot{x}_p$	$\dot{p}_A$	$\dot{p}_B$
IV-C-3	x		
IV-C-1		x	
IV-C-2			x

**Table 1: Forward Simulation Lower Triangular Form**

Now, in order to solve for the inverse, the boundary conditions are interchanged where the new input to our system is the piston position  $x_p$  and its derivatives, and the desired output is the valve opening  $x_v$ . The explicit equations dealt with are the same equations IV-C-1, IV-C-2, and IV-C-3 of the forward simulation. In this case, equation IV-C-3 is a constraint equation between states and must be derived in order to get this equation as a function of  $\dot{p}_A$  and  $\dot{p}_B$  which are the unknown variables. The equation become as follows:

$$\ddot{x}_p = \frac{1}{m_t} [(\dot{p}_A - \alpha \dot{p}_B) A_p - \dot{F}_{\text{ext}} - F_f(\dot{x}_p)] \quad \text{IV-C-4i}$$

A Dummy-derivative method can be used then (Mattson & Söderlind, 1993) in cases where the system is structurally singular due to constraint state equations. One of the state derivatives is considered as a dummy derivative which is considered now as algebraic variable. If the chamber B pressure derivative is considered as dummy derivative, then  $\dot{p}_B$  becomes  $p_B'$  and its relative state is called dummy state. Therefore, the system will consist now of IV-C-1, IV-C-2, IV-C-3, and IV-C-4i knowing that  $\dot{p}_B$  is an algebraic dummy derivative. This technique is used to keep track of the information that the states are related and to compensate for drift conditions during simulation specially with stiff systems (Bachmann, 2012).

The incidence matrix for this system of equations become as shown in Table 2.

	$\dot{p}_A$	$x_v$	$\dot{p}_B$
IV-C-1	x	x	
IV-C-2		x	x
IV-C-4i	x		x

**Table 2: Inverse Simulation Block Lower Triangular Form**

In order to solve this BLT, one of the ways used by Modelica is through the use of symbolic manipulation by making equation IV-C-4i an explicit function of  $x_v$  through replacing equations IV-C-1 and IV-C-2 in equation IV-C-4i. After manipulating IV-C-4i explicitly as function of  $x_v$ ,  $x_v$  becomes the only unknown that can be solved for as in the case of the forward simulation with the help of the initial state conditions as shown by IV-C-4ii. After calculating  $x_v$  at time  $t=0$ , the pressure derivatives  $\dot{p}_A$  and  $\dot{p}_B$  are calculated from  $x_v$ . The BLT then becomes an LT as shown in Table 3, and the system can be solved with any ODE solver such as the Forward-Euler solver discussed.

$$x_v = \frac{\left( \frac{A_p \cdot \dot{x}_p}{c_{hA}} + \frac{A_p \cdot \dot{x}_p \cdot \alpha^2}{c_{hB}} - \frac{\alpha \cdot K_{ij}}{c_{hB}} (p_B - p_A) + \frac{m_v \cdot \ddot{x}_p + F_f(\dot{x}_p) + F_{ext}}{A_p} \right)}{\left( \frac{\sqrt{p_s - p_A}}{c_{hA}} + \frac{\alpha \cdot \sqrt{p_B - p_T}}{c_{hB}} \right) \cdot c_v} \quad \text{IV-C-4ii}$$

	$x_v$	$\dot{p}_A$	$\dot{p}_B$
IV-C-4ii	x		
IV-C-1	x	x	
IV-C-2	x		x

**Table 3: Inverse Simulation Lower Triangular Form**

In case we need to know the input voltage to operate the valve, we add an additional equation which is:

$$\ddot{x}_v = -2D_v \omega_v \dot{x}_v - \omega_v^2 x_v + K_v \omega_v^2 u \quad \text{IV-C-5}$$

And the inverse incidence matrix becomes as shown in Table 4.

	$x_v$	$u$	$\dot{p}_A$	$\dot{p}_B$
IV-C-4ii	x			
IV-C-5	x	x		
IV-C-1	x		x	
IV-C-2	x			x

**Table 4: Inverse Simulation Lower Triangular Form Larger System**

It is seen from the illustrated technique how straightforward it is for Modelica to solve for the system's inverse by just interchanging the boundary conditions and manipulating the equations to obtain an LT incidence matrix that could be solved through direct substitution of variables and with any ODE solver.

These equations can also be solved by Modelica through tearing. From Table 2,  $\dot{p}_A$  is assumed as a tearing variable. This implies that for every value of  $\dot{p}_A$ ,  $x_v$  is

calculated from IV-C-1 by replacing for the value of  $\dot{p}_A$  and solving for  $x_v$ . Similarly, IV-C-2 calculates  $\dot{p}_B$  by replacing for the value of  $x_v$  obtained IV-C-4i becomes IV-C-6 and Newton's method is used to solve for the residual to make it 0.

$$y = residual = \ddot{x}_p - \frac{1}{m_t} [(\dot{p}_A - \alpha \dot{p}_B) A_p - \dot{F}_{ext} - \dot{F}_f(\dot{x}_p)] \quad \text{IV-C-6}$$

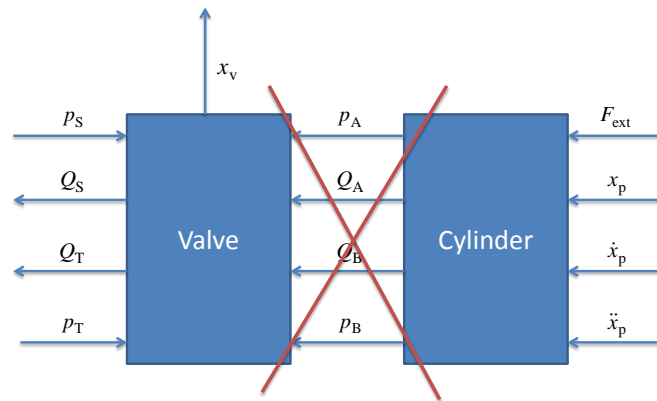
Pseudo code: (Lindfield & Penny, 2012)

```

while  $\left| \frac{y}{y'} \right| > \text{tolerance}$ 
     $\dot{p}_A(n+1) = \dot{p}_A(n) - \left| \frac{y}{y'} \right|;$ 
     $iteration = iteration + 1;$ 
     $\dot{p}_A(n) = \dot{p}_A(n+1);$ 
     $x_v = \frac{c_{hA} \cdot \dot{p}_A(n) + A_p \cdot \dot{x}_p - K_{li} \cdot (p_B - p_A)}{c_v \sqrt{p_s - p_A}};$ 
     $\dot{p}_B = \frac{1}{c_{hB}} (Q_B(x_v, p_B) + A_p \dot{x}_p - Q_{Li})$ 
     $\left| \frac{y}{y'} \right|;$ 
end

```

Figure 17 indicates that if separate models containing the equations of the valve and cylinder were built in Simulink which is based on variable assignment, it is not possible to interchange the direction of the boundary conditions and the data flow in order simulate the system inversely, because the pressures and flows in both the cylinder and valve are interdependent requiring to know their values beforehand to proceed with the inverse simulation. Therefore, in order to perform the simulation in Simulink, the whole system should be implemented after the causalization step with all its equation manipulations as shown earlier.

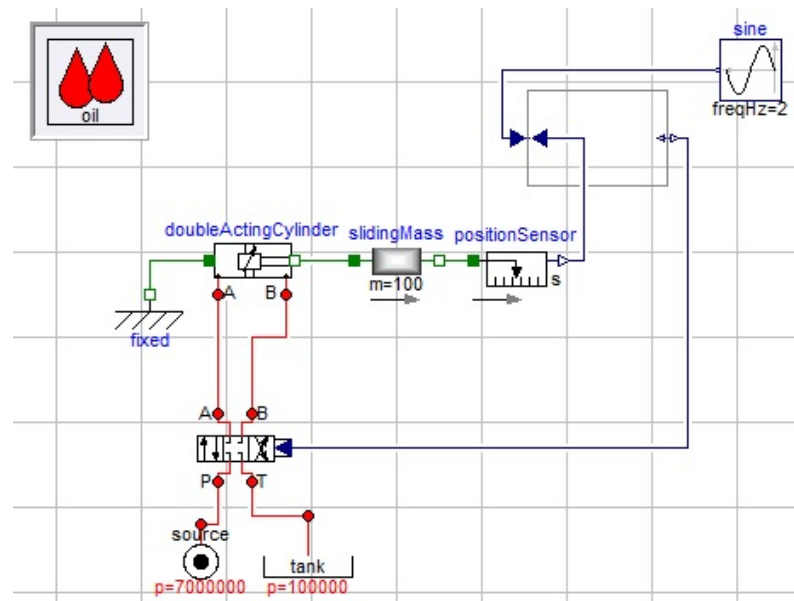


**Figure 17: Backward Simulation Approach does not work in Simulink**

Meanwhile, after writing the equations and building the models in Modelica, it is possible to simulate the hydraulic servo-drive either in a forward fashion by providing voltage input to the valve and accordingly attaining the position and velocity of the hydraulic piston actuator or inversely by providing the position of the of the cylinder as an input and simulating backwards to attain the required valve opening  $x_v$  and accordingly the required valve voltage input  $u$ . This is possible thanks to the feature of flattening where all the equations from all models are found in one global set as discussed earlier, and thus allowing the state space representation of the equations, their sorting, and the solution of the required ones. The simulation results of this servo-drive inverse dynamic example are found in (Saad & Liermann, 2013).

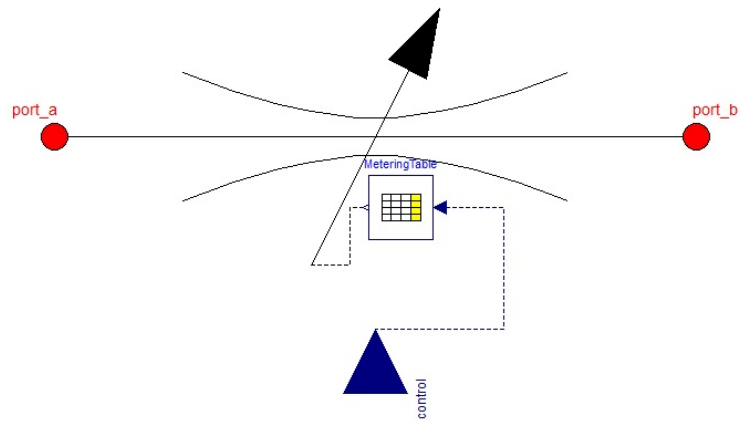
## D. Problems with Existing Library

Eventhough it was shown that it is possible to simulate the system forwardly and inversely in Modelica with the same model for the valve and the cylinder, the current libraries in Dymola for hydraulic systems do not allow so.



**Figure 18: Hydraulic Servo-Drive System in openHydraulics**

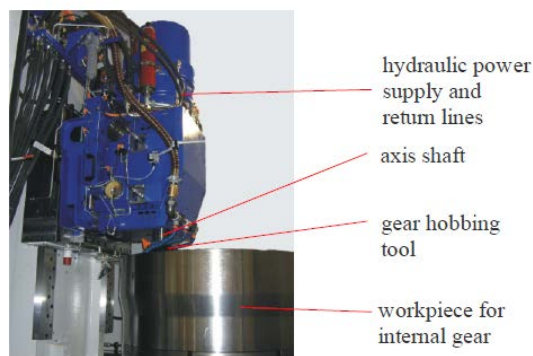
Figure 18 shows the valve and cylinder system modeled using the `openHydraulics` library in Modelica where the inverse system simulation is intended. However, Modelica was not able to simulate that system using the library components. Figure 19 shows one of the problematic parts of the model that caused that made the translation of the system to be impossible, the metering table. It provides the input to the restrictions between the ports of the valve model. This table, by way of implementation, does not allow its output to be derived more than once which in our case leads to aborting the translation process.



**Figure 19: Metering Table in openHydraulics Valve Model**

There are many other similar reasons that cause the hydraulics library not to simulate hydraulic systems inversely. These reasons are explained in depth in sections V and VI. Therefore, the goal of this thesis is to amend the library to make it viable for inverse simulation.

After modifying this library, the inverse simulation is to be tested in a case study. The case study is a hydraulic actuator of a gear hobbing machine shown in Figure 20 where the target is to optimize its design. This is illustrated in section VII.



**Figure 20: Gear Hobbing Machine**



## CHAPTER V

### INVERSION PROBLEMS IN MODELICA

This section illustrates issues that cause inversion problems in Modelica.

Table 5 provides a small summary of the inversion problems that are dealt with in this section.

Problem	Example	Solution
<b>Inversion of algorithms</b>	$m := 2 \cdot n$ $\Rightarrow$ find $n$ ?	Change the algorithm to an equation
<b>Discontinuous Equations</b>	$f(x) = \begin{cases} -x - 2, & x < 0 \\ x + 2, & x > 0 \end{cases}$	In case the point of discontinuity, $x = 0$ , is not reached, the inverse simulation runs properly; else an approximation equation should be used in the discontinuous region to join both equations and provide continuity to the function
<b>Saturation</b>	$f(x) = \begin{cases} 2, & x > 1 \\ x, & -1 \leq x \leq 1 \\ -2, & x < -1 \end{cases}$	The inverse is defined in the region where saturation does not occur. However, in the region of saturation, the inverse is not defined because there are infinite solutions of $x$ for one $f(x)$ . Solution is either changing the saturation regions or working in the regions where there is no saturation.

<b>Non-Monotony</b>	$y = 2 \cdot x^2$	Parabolic and other functions having local minima or maxima have inverse simulation problems. Solution is to avoid local maxima/minima regions in the simulation or approximate these regions to be monotonly increasing or decreasing.
<b>Derivative Function not Found</b>	2 <sup>nd</sup> order derivative in interpolation tables. Functions that need to be derived to be inverted and their derivatives are not defined.	Replace those functions or tables by equations written directly in the model to be simulated.
<b>Non-Differentiable Equations</b>	Critical point transitions are not differentiable (constant input + ramp)	User-defined differentiable inputs through approximating the critical points with higher order smooth equation transitions

**Table 5: Problems and Solutions**

### A. Inversion of Algorithms

Algorithms are not invertible when building inverse systems, since they are predefined statements having a certain direction of causality as discussed in section III-C-2. Therefore, just interchanging the input/output boundary conditions will not result in an inverse model, rather it will result in an error message stating that Modelica cannot differentiate algorithms which is a drawback when simulating the inverse of a model containing an algorithm section in its body.

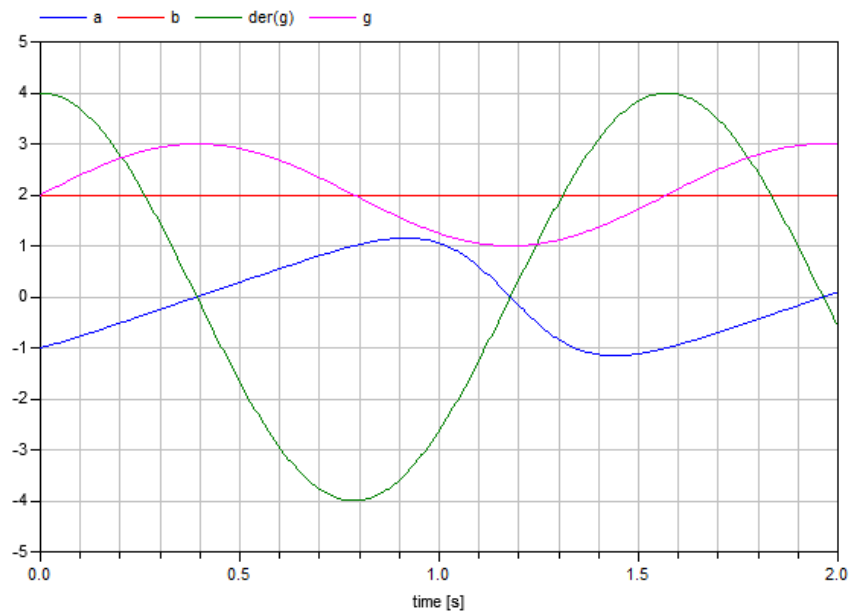
In section III-C-3, it was stated that an algorithm is seen as a block whose output is the only result that matters and is used by the solution of the system model disregarding all the statements and assignments that are found within the algorithm block. In addition to that, the assigned statements within this algorithm block have only one direction of causality and this direction cannot be interchanged autonomously by

Modelica when performing inverse simulations rather the user must interchange their direction of causality manually. However, in section III-C-4, it was also discussed that the body of the written function is an algorithm. Nonetheless, it was also mentioned that calling a function is done through an equation. This gives the functions the properties of equations. Thus, the function is invertible. This is shown in the below.

```
function f
  input Real a;
  output Real b;
algorithm
  b := 2*a;
end f;

model example1
  Real a;
  Real g;
  parameter Real b=2;
equation
  g = 2 + sin(2*b*time);
  der(g) + g*f(a)=0;
end example1;
```

Function `f` has `a` as an input and `b` as an output as defined. In `example1` model, `b` is given a constant input value; whereas, `a` and `g` are calculated from the 2 equations under the equation section. The first equation calculates `g`; then, the second equation will automatically be used to calculate `a` eventhough `a` is given as an input to the function `f`. This example serves as a proof of the inversion capability of the function in Modelica if called within a model knowing that its body is comprised of an algorithm. The simulation results of the model for one period are shown in Figure 21.



**Figure 21: example1 Results**

Algorithm sections are widely used as part in the models of the `openHydraulics` library components and are one of the factors that are causing inverse simulation problems to this library as will be discussed in section VI.

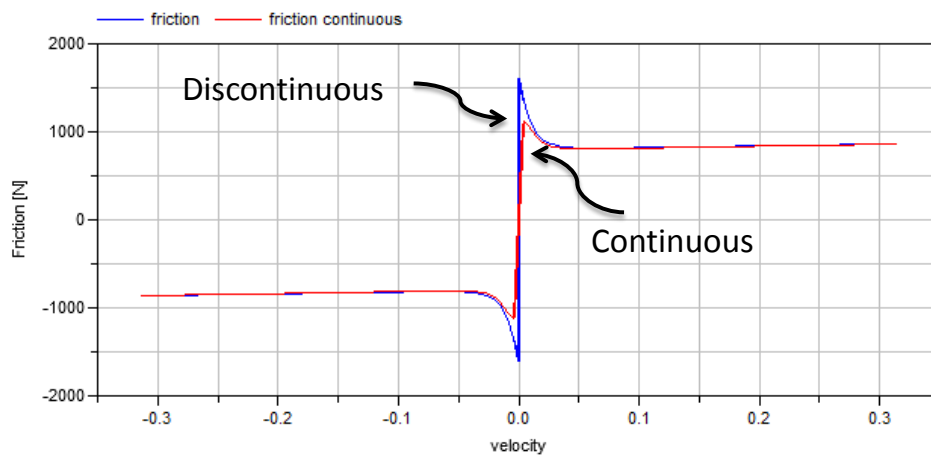
## B. Discontinuities

Some equations do not have an inverse due to discontinuities, saturation, or non-monotony. Examples of physical effects pertaining such characteristics can be friction equations, time delays, backlash, hysteresis, etc. (Otter M., 2005).

In case the non-invertible part of the equation is not needed or reached during the system inverse simulation, Modelica will simulate the system as if there is no discontinuity, because that discontinuity did not affect the simulation.

The Stribeck friction curve discussed in section IV-B and illustrated in equation IV-B-10 is discontinuous at zero-velocity. This is shown in blue in Figure 22.

This discontinuity is symmetric with respect to the origin. It will not allow the inverse of such an equation to be constructed; therefore, approximation in the region of the discontinuity is required in order to make the equation continuous and thus invertible. Figure 22 also shows in red the friction model with an approximation around 0, equation IV-B-11, that takes out the discontinuity problem of inversion; however, that model is still not invertible since it is not monotonous and is dealt with in section V-D.



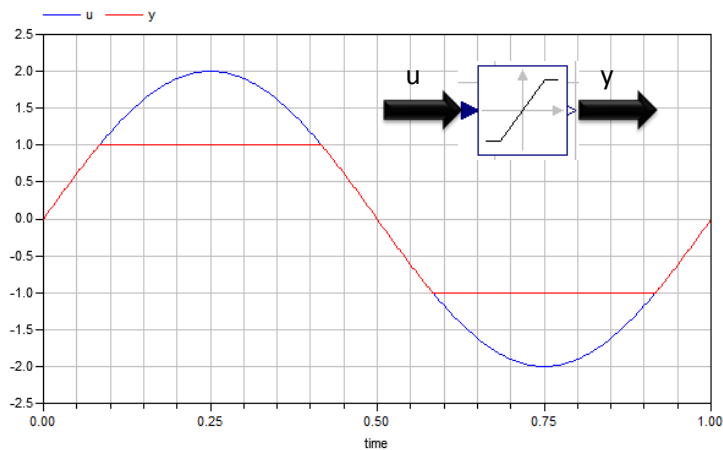
**Figure 22: Friction v/s Velocity Discontinuous**

### C. Saturation

Saturation is the case when the output for a certain region is constant no matter if the input increases. This can be defined as when the output reaches a maximum or minimum limit. Actuator limitation is considered as a saturation, because at the limits, the equation is constant regardless of the increase in the input. Figure 23 shows a saturation function which limits the sinusoidal input equation between the maximum and minimum boundaries of 1 and -1, respectively.

Saturation functions are not invertible, because for the same inverted input, there are infinite number of outputs. The inverse is defined in the region where

saturation does not occur. A solution is either through changing the saturation regions to monotonly increasing with a very slow slope or working in the regions where there is no saturation.



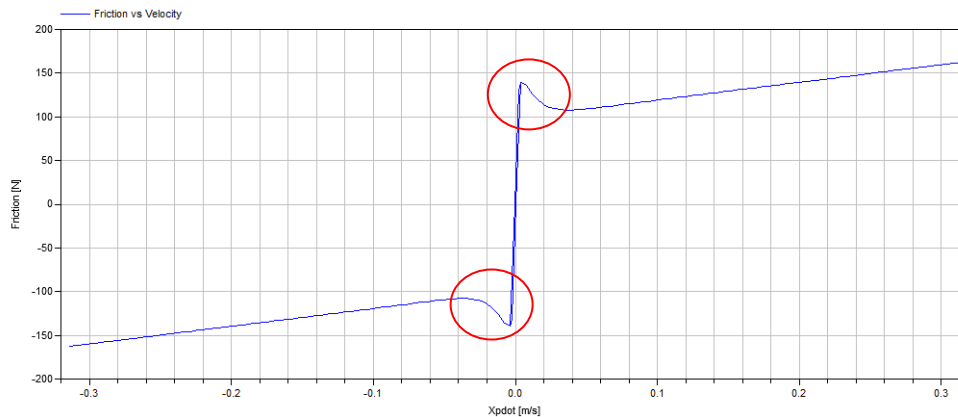
**Figure 23: Limited Output**

#### **D. Non-Monotony**

Other type of equations which are not invertible are the non-monotonous equations such as the parabolic and hyperbolic equations. These equations are not invertible specifically at the regions of local maxima or minima where the slope of the inverted function becomes infinite.

Figure 24 shows a friction model which have two regions indicated by the circles which are not monotonous. In those regions, inversion is not possible, because for the same ordinate there are two abscissas.

In order to solve such problems, approximations should be made in those regions to neglect small local maxima or minima and approximate that the system is continuously increasing or decreasing after these points.

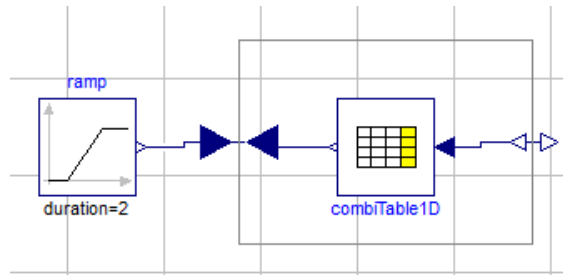


**Figure 24: Friction Model Non-Monotonous**

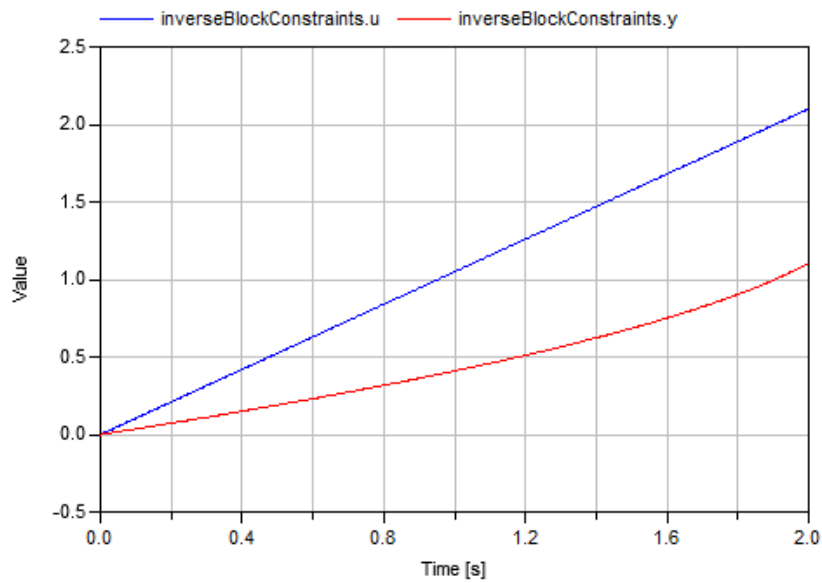
### E. Derivative function not found

In some inverse simulation cases, Modelica requires to derive some equations several times. In case these equations are called up by external functions, these functions might not be written in a way to include their  $n^{\text{th}}$  order derivatives. In these cases Modelica returns an error message indicating the Modelica cannot find a derivative function in order to reduce the DAE index. This type of error is tackled when modifying the pipeline of the `openHydraulics` library and is discussed in section VI-C.

Inversion of interpolation tables in Modelica causes similar problems if the system simulated requires that the table equation be derived twice. A table represents a function  $y = f(u)$  where its role is to interpolate the data provided by the user in the column input  $u$  to obtain the output of the table found in its column table  $y$  which will be fed to the system. The table by itself is invertible as shown in Figure 25 and Figure 26.



**Figure 25: Inversion of Interpolation Table**



**Figure 26: Inversion of Interpolation Table Simulation**

However, with some systems such as the valve, the inversion of such tables is not possible, because Modelica does not have a function that provide the 2<sup>nd</sup> order derivative for the table which will result in an error message.

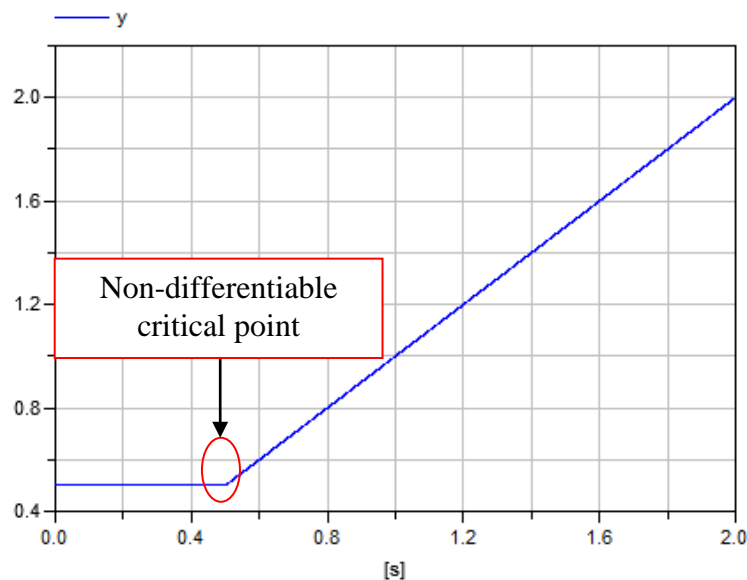
To solve this problem, the table can be replaced with a corresponding equation that represents the same input/output relationship provided by the table. This solution is discussed in the valve modifications section VI-A.



## F. Non-differentiable Inputs

System inputs are very important in inverse simulation. Inputs have to be at least  $n$ -times differentiable where  $n$  is the maximum required derivative in order for the solution to exist. For example, in a hydraulic servo-drive inverse simulation as in the example of section VII, if the input is the cylinder position, it should be twice differentiable if the acceleration is required for the progression of the solution.

The real problem is with constructing an input from different equation configurations. For example, if the input for a certain interval of time is constant and then it becomes ramp, the solver will not be able to solve the solution because the derivative at the critical point of change is infinite. This is shown in Figure 27. This is the major problem faced with inverse simulations in Modelica and specifically in the `openHydraulics` library.



**Figure 27: Non-differentiable Equation**

Using higher-order filters to smooth such inputs do not solve the inverse simulation problem as in section VII. To solve this problem, the user has to build his own input models by making these critical point transitions smooth through approximating them with higher order equations. This technique is used in section VII where the critical points are approximated according to the equation found in (Zeit).

## CHAPTER VI

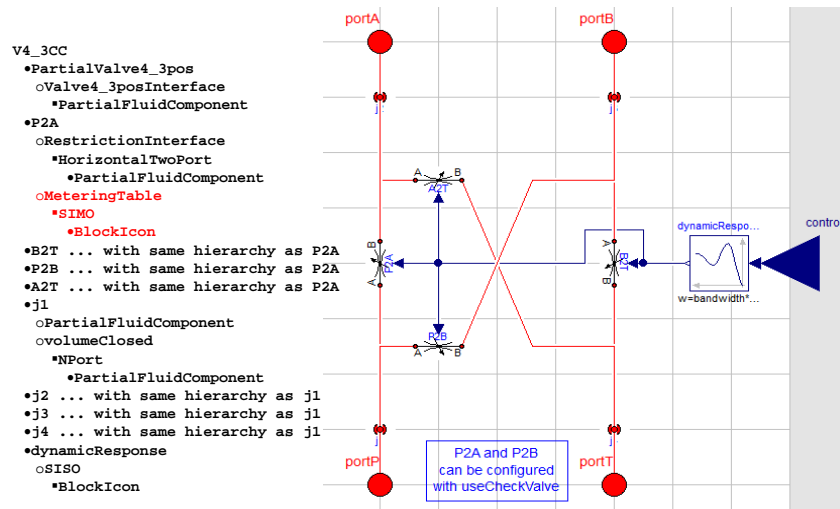
### OpenHydraulics LIBRARY COMPONENTS & THEIR INVERSION PROBLEMS

The aim of this thesis is to modify the basic components of the `openHydraulics` library to enable combined forward/inverse simulations. The component inversion problems along with their performed modifications are illustrated in this section.

#### A. Valve

The valve is a major component in hydraulic systems. In the `openHydraulics` library, the `v4_3cc` directional control valve is considered and its components are shown in Figure 28. It consists of the following components: `A2T` and `B2T` which are variable restriction models, `P2A` and `P2B` which are variable restriction series models having the same properties as variable restriction models, and `dynamicResponse` which is a second order block. The valve has 4 fluid ports and a control input representing the voltage signal input to the valve. It also has 4 junctions that contain closed volumes in each of them.

The model hierarchical tree of the valve is shown in Figure 28. The components causing inverse simulation problems are highlighted in red:



**Figure 28: Valve Model and Hierarchical Tree**

It is seen from the hierarchical tree how complex the valve model is from the high number of models and submodels involved.

The directional control valve  $v4\_3cc$ , as stated, consists of 4 orifices under the name of variable restriction ( $A_{2T}, B_{2T}$ ) and variable series restriction ( $P_{2A}, P_{2B}$ ) in addition to a second order block. The second order block receives the voltage input  $u$  to the valve provided by an external voltage source or a controller and outputs the valve opening  $x_v$  according to the 2<sup>nd</sup> order dynamic relation discussed in equation IV-A-4.  $x_v$  then is provided as the input to all 4 orifices which are triggered by their interpolation table relation depending on the sign of  $x_v$ . The interpolation table ( $MeteringTable$ ) relation provides the orifice opening direction thus the 3 cases of the of the valve opening.

The first case is the positive opening of the valve. This means that orifice  $P_{2A}$  and orifice  $B_{2T}$  are opened allowing flow from the pump to cylinder chamber A and from cylinder chamber B to tank respectively; whereas,  $A_{2T}$  and  $P_{2B}$  are closed blocking the flow from the pump to cylinder chamber B and from cylinder chamber A to tank.

Therefore, the interpolation table trigger parameters between the control input of the table and its output which is named `openFraction` of `P2A` and `B2T` are as shown in Table 6:

<i>Control</i>	<i>openFraction</i>
0	0
1	1

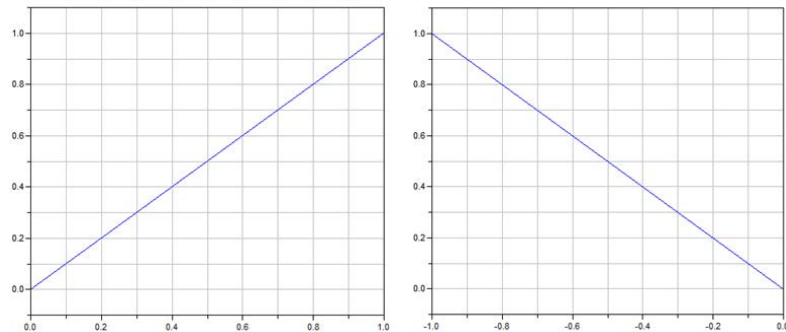
**Table 6: P2A & B2T Interpolation Table Parameters**

The second case is the negative opening of the valve. In this case `P2B` and `A2T` are opened allowing flow from pump to cylinder chamber B and from cylinder chamber A to tank, and `P2A` and `B2T` are closed to block the opposite flow senses. Therefore, the interpolation trigger parameters of `P2B` and `A2T` are as shown in Table 7:

<i>Control</i>	<i>openFraction</i>
0	0
-1	1

**Table 7: P2B & A2T Interpolation Table Parameters**

The first and second cases `MeteringTable` plots are shown in Figure 29.



**Figure 29: P2A & B2T v/s P2B & A2T**

The third case is when the valve is fully closed providing zero flow to the actuators. This case is triggered by the voltage source which provides the valve with zero voltage  $u$ ; thus in turn zero valve opening  $x_v$ .

The valve problem, when performing an inverse simulation between the cylinder and the valve as shown in Figure 18, is with the `MeteringTable` highlighted in red in the hierarchical model tree. In this inverse simulation, the input to the system is the output required cylinder position  $s$  and the output is the voltage input  $u$  to the valve.

The error found for this simulation is highlighted as follows:

**Warning: Cannot find differentiation function:**

***dymTableIpo1Der(v4\_3CC.B2T.MeteringTable.tableID, 2, v4\_3CC.B2T.MeteringTable.u, der(v4\_3CC.B2T.MeteringTable.u))  
with respect to time***

**Failed to differentiate the equation**

***der(v4\_3CC.B2T.MeteringTable.y[1]) = dymTableIpo1Der(v4\_3CC.B2T.MeteringTable.tableID, 2, v4\_3CC.B2T.MeteringTable.u, der(v4\_3CC.B2T.MeteringTable.u));  
in order to reduce the DAE index.***

This error indicates that Dymola cannot find a 2<sup>nd</sup> derivative function between the output and the input of the metering interpolation table found in the variable restriction models. The table only allows a 1<sup>st</sup> derivative between its input and output. Thus, in case we had a first order block between  $x_v$  and  $u$  indicating a 1st order relation between them instead of 2nd order dynamic relation, the valve will not have an inverse simulation problem, because the metering interpolation table allows 1<sup>st</sup> order derivative between its boundaries as already stated. However, the valve modeled in the `openHydraulics` library has a 2<sup>nd</sup> order relation between  $x_v$  and  $u$  see equation IV-A-4.

In order to solve this problem, the interpolation metering table block is removed and replaced by an equation. The equation allows 2<sup>nd</sup> order derivatives. Therefore, a modified variable restriction model and a modified variable series valve

restriction are included in the library. Both replace the `MeteringTable` with the following equation:

```
openFraction = max(min(control, max_contr), min_contr);
```

This equation also limits the `openFraction` within the control limits `max_contr` and `min_contr` as with the `MeteringTable`. Therefore, the `P2A` and `P2B` are replaced by the modified variable series restriction model but with different control limits (`max_contr`, `min_contr`) of (1,0) for `P2A` and (0,-1) for `P2B`. `A2T` and `B2T` are replaced by the modified variable restriction model, and also with different control limits of (0,-1) and (1,0) respectively.

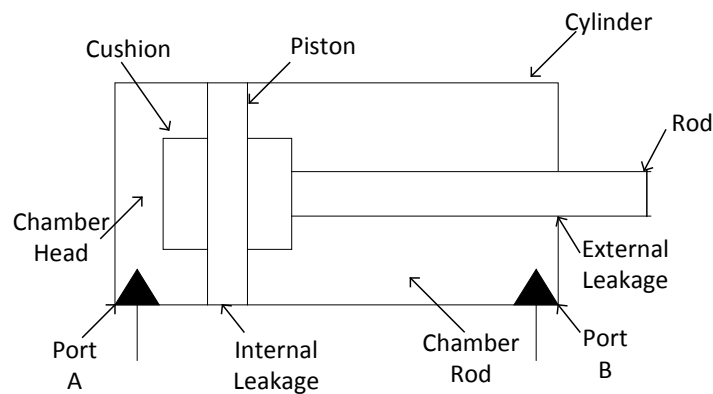
A warning message is also included in the valve to inform the user when the valve exceeds its opening limitation. This is illustrated by the `when` clause:

```
when openFraction>=max_contr or openFraction<=min_contr then  
  Modelica.Utilities.Streams.print("\nWARNING: Valve opening exceeded");  
  Modelica.Utilities.Streams.print("          The valve is undersized");  
end when;
```

After these changes, the modified directional control valve `v4_3CC_mod` works for inverse valve/cylinder simulations.

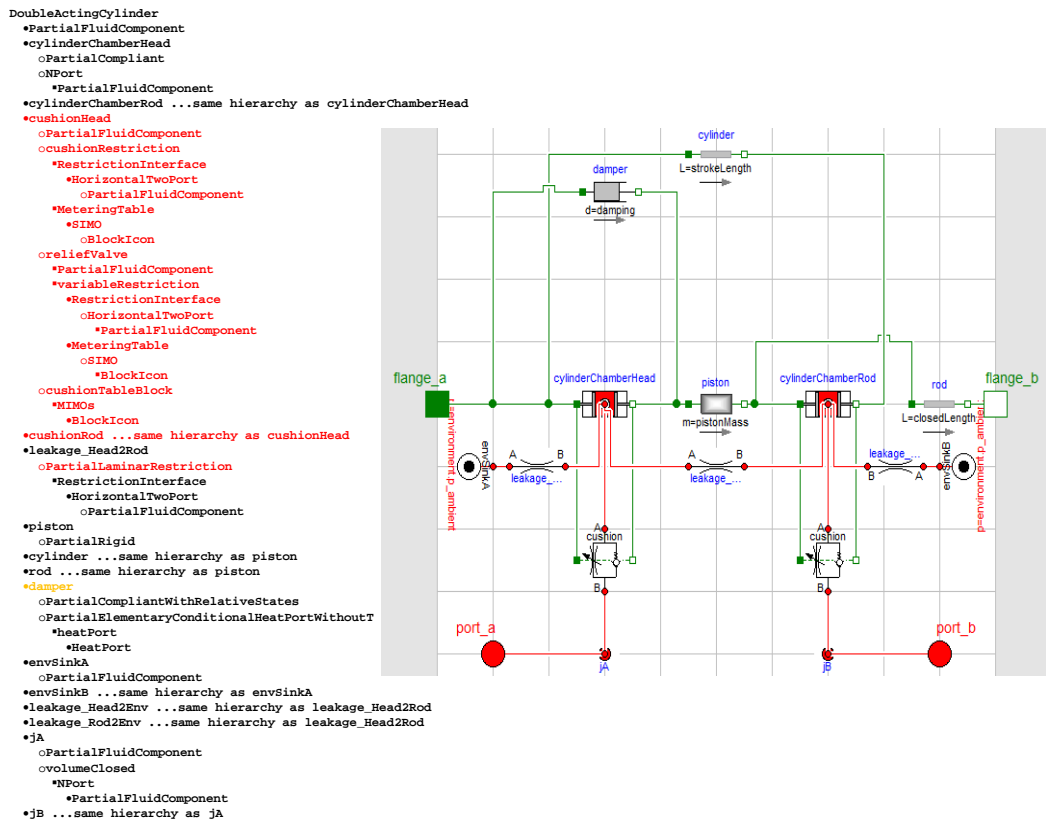
## B. Cylinder

The cylinder is another essential hydraulic component that is found in most hydraulic circuits. It translates the orders provided by a valve in a mechanical fashion. Figure 30 and Figure 31 illustrate the cylinder that is modeled in `openHydraulics` library. It is a double acting cylinder with a single rod consisting of 2 chambers, the rod chamber and the head chamber which does not contain any rod, as shown in Figure 30. Figure 31 shows the model components that are used in `openHydraulics` library that constitute the cylinder. It consists of a piston model, a damper model, a cylinder length model, a rod model, 2 chamber models, internal and 2 external leakage models, and 2 cushion models. Some cylinders have cushions, while others do not; however, the `openHydraulics` cylinder is modeled with cushions knowing that they can be taken out of the model depending on the user's needs.



**Figure 30: Double Acting Cylinder Modeled**





**Figure 31: openHydraulics Double Acting Cylinder Model and Hierarchical Tree**

The double acting cylinder model is complex consisting of many submodels as shown in Figure 31. The model hierarchical tree is illustrated in Figure 31.

Similar to the valve, the double acting cylinder also has inverse simulation errors. The components comprising the cylinder errors are highlighted in red in the hierarchical tree. The modifications made to these components are illustrated afterwards.

The first error encountered is related to section A where the algorithms are not invertible. The error message received by Dymola is the following:

**Warning: Dymola is currently unable to differentiate algorithms.  
Failed to differentiate the equation**

**algorithm**

*doubleActingCylindermodified.leakage\_Rod2Env.port\_a.m\_flow := doubleActingCylindermodified.leakage\_Rod2Env.conductance\*doubleActingCylindermodified.leakage\_Rod2Env.dp;*

*in order to reduce the DAE index.*

This algorithm is found in the `PartialLaminarRestriction` model extended by the `LaminarRestriction` model which is used to represent the 3 leakages of the cylinder. In order to solve this error, the `algorithm` section of the `PartialLaminarRestriction` model is replaced by an `equation` section as follows:

<code>algorithm</code>	<code>equation</code>
<code>port_a.m_flow := conductance*dp;</code>	<code>port_a.m_flow = conductance*dp;</code>

After this replacement, the modified laminar restriction models works properly in inverse simulations.

The second error is caused by the cushions. The cushion model is very complex containing several submodels. It includes 3 interpolation tables. Two under the name of `MeteringTable` and one under the name of `cushionTableBlock`. Due to the density of the model, and once included in valve/cylinder system inverse simulation, the solver has to perform a lot of higher order differentiations that is not feasible to be solved. However, the cushion does not play an important role in inverse simulation, its use is important in forward simulation to decrease the impact between the piston and the cylinder's end position. A rigid end-stop would excite high frequency modes and would considerably slow down simulation time, which is often undesired. It is unlikely though that the cylinder runs into end-stops using the inverse simulation approach. Therefore, to decrease the complexity of the system, the cushion models are taken out from the

modified double acting cylinder and the inverse valve/cylinder simulation runs with no further complications.

The cylinder model includes a damper taken from the Modelica library, but this damper does not include a stribeck friction force. Therefore, a new model called `Damper_withStribeckfriction` is introduced to the `openHydraulics` library under the cylinder category. The Modelica library `damper` model's friction force is only a function of the speed according to the following relation:

$$f = d \cdot v \tag{VI-B-1}$$

where  $f$  is the friction force,  $d$  is the damping constant, and  $v$  is the relative speed which in this case is the piston speed  $\dot{x}_p$ .

The new `Damper_withStribeckfriction` model replaces the friction force of the previous Modelica library `damper` model in equation VI-B-1 with the stribeck friction force of equation IV-B-10 including equations IV-B-11 and IV-B-12. Therefore, the friction force equation is the following:

$$f = \sigma \dot{x}_p + \frac{2}{\pi} \cdot \arctan(\gamma \dot{x}_p) [F_{c0} + F_{s0} e^{\frac{-\dot{x}_p \frac{2}{\pi} \arctan(\gamma \dot{x}_p)}{c_s}}] \tag{VI-B-2}$$

The model also calculates the power loss due to the friction force according to the following equation:

$$P_{lossfriction} = f \cdot \dot{x}_p \tag{VI-B-3}$$

Moreover, the cylinder only contains 1 rod found in the rod chamber. Therefore, an additional option is added where the user is now able to include a rod in

the head chamber in case he needs. In this way, differential or synchronizing cylinders can now be modeled.

Those are the modifications performed on the double acting cylinder of the `openHydraulics` library, which resulted in a modified cylinder allowing inverse simulations.

### C. Pipeline Modifications

There are 2 pipelines in the `openHydraulics` library. One consists only of a friction model with no volume, while the other consists of 3 volumes and 2 friction models. They are represented by Figure 32 and Figure 33 respectively.

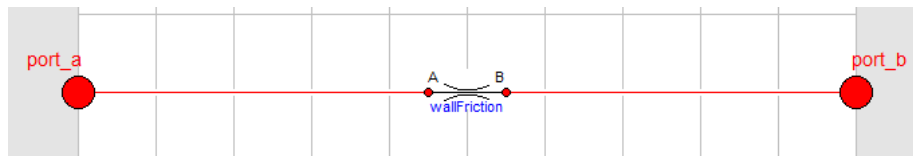


Figure 32: Line with No Volume



```

Line
  •PartialFluidComponent
  •volumeA
    oNPort
      ▪PartialFluidComponent
        •volumeB ...same hierarchy as volumeA
  •volumeMiddle ...same hierarchy as volumeA
  •wallFriction_a
    oRestrictionInterface
      ▪HorizontalTwoPort
        •PartialFluidComponent
  •wallFriction_b ...same hierarchy as wallFriction_b
  
```

Figure 33: Pipeline with Volume Model and Hierarchical Tree

Taking the first model, Figure 32, which consists only of a wall friction model. This model calculates the flowrate from a function called `massFlowRate_dp_WallFriction`. This function takes the pressure difference of the ports, the density at each port, the dynamic viscosity at each port, the length of the pipe  $l$ , the diameter, and the roughness as inputs.

This function first assigns a density value according to the sign of the pressure difference. Therefore, if the pressure difference is positive, then the density at port A is assigned to the function, else the density of port B is assigned. Similarly, the dynamic viscosity is treated in the same fashion. Then, the modified friction coefficient is calculated as shown in VI-C-1:

$$\lambda_2 = \frac{2 \cdot \rho \cdot |dp| \cdot diameter^2}{l \cdot \eta^2} \quad \text{VI-C-1}$$

The Reynold's number is calculated assuming laminar flow as in VI-C-2:

$$Re = \frac{\lambda_2}{64} \quad \text{VI-C-2}$$

Then the calculated Reynold's number is checked whether it is in the laminar region by comparing it to a given value. If it is larger than that value, a modified Reynold's number is calculated according to the turbulent region through equation VI-C-3.

$$Re = -2 \cdot \sqrt{\lambda_2} \cdot \log_{10} \left( \frac{2.51}{\sqrt{\lambda_2}} + 0.27 \cdot \delta \right) \quad \text{VI-C-3}$$

where  $\delta = \frac{\text{roughness}}{\text{diameter}}$  is called the relative roughness.

In case the calculated  $Re < Re_2$  which is the entrance to the turbulent region, the Reynold's number is located in the transitional phase between the turbulent and laminar regions and is calculated through an interpolation function. This function takes the calculated  $Re$ , the Reynold's number leaving the laminar region  $Re_1$ , the Reynold's number at the entrance of the turbulent region  $Re_2$ ,  $\delta$ , and  $\lambda_2$  as inputs to calculate the appropriate  $Re$ .

The mass flowrate is then calculated as VI-C-4 in case  $dp > 0$ . If  $dp < 0$ , the mass flowrate is then negative and multiplied by -1.

$$\dot{m} = \frac{\pi \cdot \text{diameter} \cdot \eta}{4} \cdot Re \quad \text{VI-C-4}$$

The second pipeline model shown in Figure 33 is more comprehensive where it includes the compressibility properties found in the closed volume model. The pipeline includes a closed volume at the beginning of the line, then a friction model covering the first half length, a closed volume at the middle of the line, another friction model covering the next half length, and a closed volume at the end of the pipeline. This is illustrated by the hierarchical tree.

Both models if added to the valve/cylinder system do not allow inverse simulation. Therefore, modifications to the models has to be done. The problem is with the friction model shortly described above.

The first problem is with the oil density and dynamic viscosity which are not constant and have to be calculated at every port. They are required to calculate  $\lambda_2$  and the mass flowrate  $\dot{m}$  according to equations VI-C-1 and VI-C-4 respectively. In forward simulation, the data flow allows the solver to calculate the required densities and viscosities at the ports knowing the pressure difference. However, in inverse simulation, equation VI-C-4 has more than one unknown which includes the density and dynamic viscosity in addition to the pressure difference. In order to be able to solve this issue, an assumption of constant density and dynamic viscosity must be made. This is performed by modifying the oil properties and is discussed in section VI-D.

Another problem is with the Reynold's number. It includes many cases and also an interpolation function depending on the region. This makes the inverse simulation hard to be solved by the numeric solver.

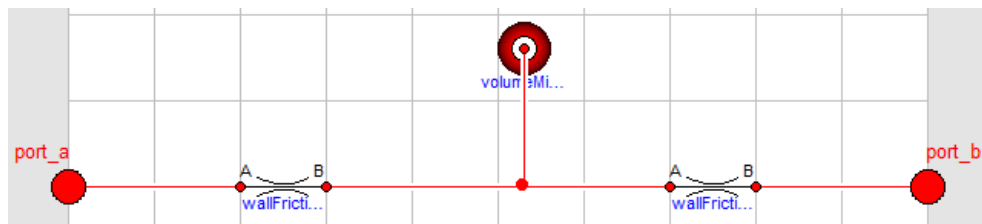
Therefore, a modified wall friction model is included. This model simplifies the cases to only laminar region which is easier for the solver to solve in inverse simulations. This model is called `WallFriction_mod_model` and directly calculates the mass flowrate considering constant density and dynamic viscosity as shown in VI-C-5.

$$\dot{m} = \frac{\pi \cdot \rho \cdot dp \cdot diameter^4}{128 \cdot l \cdot \eta} \quad \text{VI-C-5}$$

From VI-C-5 ,it is seen that now there is a direct relation between the pressure difference and the flowrate that will make the inverse simulation feasible by the solver and can be used in the valve/cylinder system inverse simulation.

However, modifying the pipeline that includes 3 volumes shown in Figure 33 through changing the `WallFriction` models by the `WallFriction_mod_model` models is not feasible

for inverse simulation once included valve/cylinder system. The only combination that works is with inclusion of only one volume placed in the middle of the pipe which will compensate to the whole compressibility properties of line, instead of the use of 3 volumes, in addition to the 2 modified wall friction models at both half lengths of the pipe. This `Line_mod` model is shown in Figure 34.



**Figure 34: Line\_mod model**

This modified line can now be included in the valve/cylinder whole system where the inverse simulation of this system is now feasible.

#### **D. Oil Modifications**

The `openHydraulics` library has two oil models that are found under the Fluids section. The oil model is necessary in every system simulation to provide the oil properties to the components involved.

The first model is the `GenericOil` model where the density of the fluid is calculated by the "Tait equation" function shown in equation VI-D-1:

$$\rho = \frac{\rho_0 \cdot \left(1 - \log \left( \frac{K_0}{1 + K_0} \right)^{\frac{1+p \cdot \left( \frac{1+K_0}{K_0} \right)}} \right)}{1 + aV \cdot (T - T_0)} \quad \text{VI-D-1}$$



where

$$K_0 = K_{00} \cdot e^{-\beta_K \cdot T}$$

**VI-D-2**

The parameters of this equation are illustrated in Table 8.

<i>Parameter</i>	<i>Definition</i>
$\rho_0=870$	Reference density at $p_0$ and $T_0$
$K_0$	Temperature-dependent Bulk modulus
$K_{00}= 8.4e9$	Bulk Modulus at 0K
$\bar{K}_0=10.9$	Constant in Tait equation
$T_0=273.15K$	Reference Temperature
$\beta_K=0.0058$	Temperature Coefficient
$aV=7.7e-4$	Thermal Coefficient
$p_0=101325 \text{ N.m}^{-2}$	Reference Pressure

**Table 8: Parameters for Generic Oil density calculation**

The dynamic viscosity is also variable and is calculated by equation VI-D-3 which is based on the Walter equation for kinematic viscosity:

$$\eta = \rho \cdot e^{-6} \cdot (-0.7 + 10^{10^{9.32-3.65 \log(T)}})$$

**VI-D-3**

The other oil model found in the `openHydraulics` library is called `GenericOilSimple`. In this model, the dynamic viscosity is constant  $\eta = 0.036$ , while the density is calculated by equation VI-D-4:

$$\rho = 870 + 5 \cdot e^{-7} \cdot (p - p_0)$$

**VI-D-4**

where  $p_0 = 101325 \text{ N.m}^{-2}$ .

Both oil models do not allow the inverse simulation of the modified valve/cylinder system with the modified pipeline. This is due to the fact that the mass

flowrate in the friction of the pipeline depends on the density and the difference in pressure. Therefore, if the density is variable and depends on the pressure variation as is the case with both oil models in addition to the pressure difference which is also variable, the inverse simulation becomes hard for the solver to handle.

Therefore, a new modified oil model called `GenericOilSimple_mod` is introduced where in this model both the density and the dynamic viscosity are considered constant and equal to  $\rho = 870$  and  $\eta = 0.036$ . This model allows then the inverse simulation of the valve/cylinder model including the modified pipeline discussed in section VI-C.

## CHAPTER VII

### APPLICATION EXAMPLE: VALVE/CYLINDER DESIGN OPTIMIZATION

After modifying the `openHydraulics` library to allow inverse simulations, this library is tested in this section on a hydraulic servo-drive system to optimize its design.

A gear hobbing machine is a typical example of hydraulic servo-drive systems. It is one type of milling machines that is responsible for manufacturing gears. It has teeth that progressively cut into the workpiece to provide it with its final shape. This machine consists of a hydraulic power supply and return lines, an axis shaft, and a gear hobbing tool called hob attached to the axis shaft. The axis shaft which is a hydraulic actuator moves up and down and has a rotating actuator at the end of the rod. In our case, the workpiece continuously rotates during which the hob scrapes its inner surface in its down stroke until the workpiece has its final shape. Figure 20 shows a picture of the gear hobbing machine. Figure 35 shows the hydraulic cylinder of the machine taken from the `openHydraulics` library.

This section illustrates the importance of inverse simulation in designing systems and selecting their parameters. It makes use of the modified `openHydraulics` library components mentioned in this thesis to build and simulate the hydraulic valve/cylinder component part of the gear hobbing machine. Figure 36 shows the dymola gear hobbing valve/cylinder circuit modeled. It includes a constant supply pressure source, a modified directional control valve, 2 volumes on the cylinder supply and return, modified cylinder, external applied force, tank, position sensor, in addition to pressure and speed sensors.

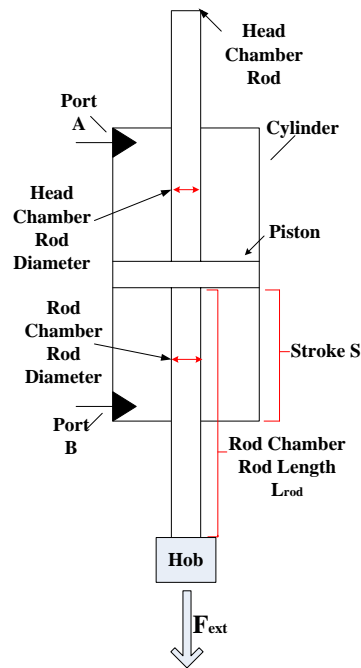


Figure 35: Gear Hobbing Cylinder Actuator

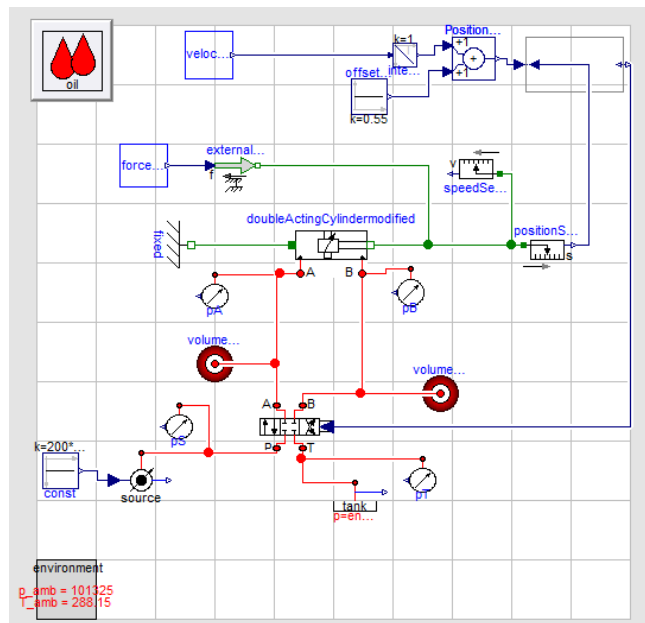


Figure 36: Gear Hobbing Valve/Cylinder Dymola Inverse Circuit

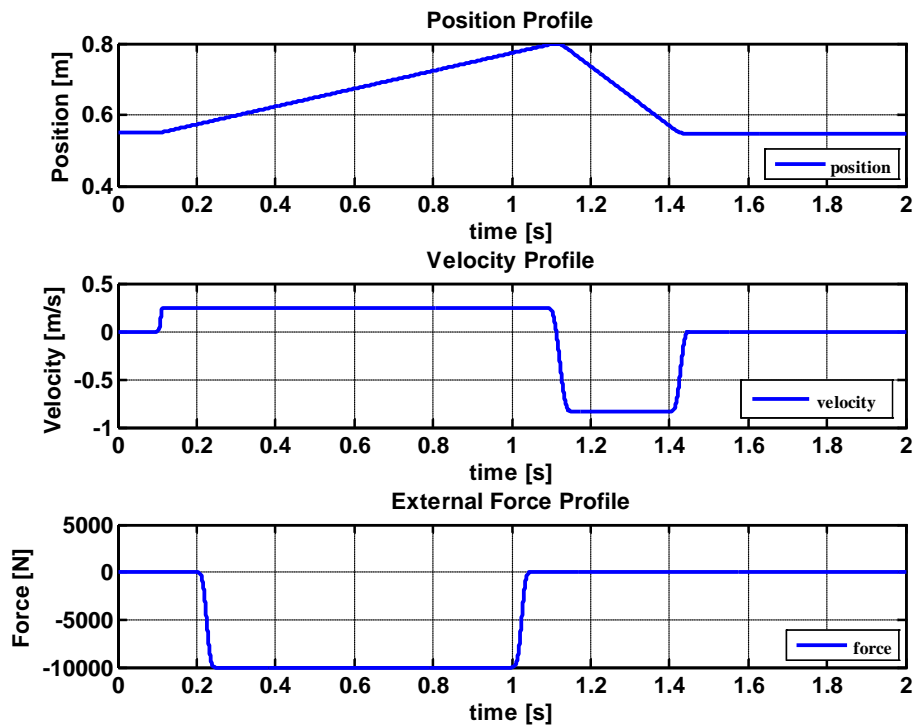
The cylinder initial parameters are illustrated in Table 9 from a commercially available tooling machine.

Initial Cylinder Parameters							
Bore Diameter	Rod Chamber Rod Diameter	Head Chamber Rod Diameter	Stroke Length	Rod Chamber Rod Length	Nominal Flowrate	Nominal Pressure drop	Maximum Rated Pressure
100mm	80mm	80mm	0.3m	0.5m	$0.01 \frac{m^3}{s}$	0.1bar	300bar

**Table 9: Cylinder Initial Configuration Parameters**

The cylinder rod chamber rod length taken in this simulation is  $L_{rod} = 0.5m$ , and the maximum cylinder stroke is  $s = 0.3m$ . The input to the simulation is cylinder position  $x_p$  which is provided to the position sensor. The simulation runs for 2s and  $x_p$  is shown in Figure 37.

In order to acquire this smooth signal, the user must build a differentiable function input for at least a certain minimum order and not rely on filters as discussed in section F. For this reason, the velocity profile shown in Figure 37 is modeled first, where the velocity trajectory changes are modeled as discussed in (Zeitz) assuming a 4th order differentiable smooth curve. In this way, the points of trajectory change are now smooth allowing the solver to derive them and not causing any divergence in the solver solution or causing the solution to become infinite thus leading the simulation to halt.

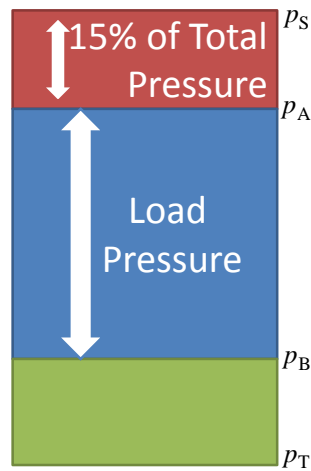


**Figure 37: Cylinder Position Input, Velocity Profile, and External Force**

The velocity profile is integrated and an offset of 0.55m is added to obtain the smooth position input profile of Figure 37. The cylinder starts at 0.55m and remains constant the first 0.1s. Afterwards, the working cycle starts where the position of the rod chamber rod increases to reach its maximum of 0.8m from the cylinder head chamber base at 1.1s. During the working stroke cycle, the external force is applied on the cylinder as shown in Figure 37. The force starts at 0.2s, increases to reach its maximum of 10000N and then returns to 0 at 1.05s which is approximately the end of the working stroke. After 1.1s, the cylinder starts its return stroke and retrieves back to its end position at 1.44s.

The aim is to select the best possible system parameter configurations that fulfill the user requirements and consume the least energy. In this case, the user

requirements for the system is to have a rod chamber rod diameter of  $D_{rod} = 0.08\text{m}$  that allows the contact with high amplitude external forces without having large stresses, and to use as much load pressure as possible keeping around 15% safety pressure difference between the source pressure and cylinder head chamber pressure  $p_s - p_A$  in the working stroke as shown in Figure 38 to maintain controllability.



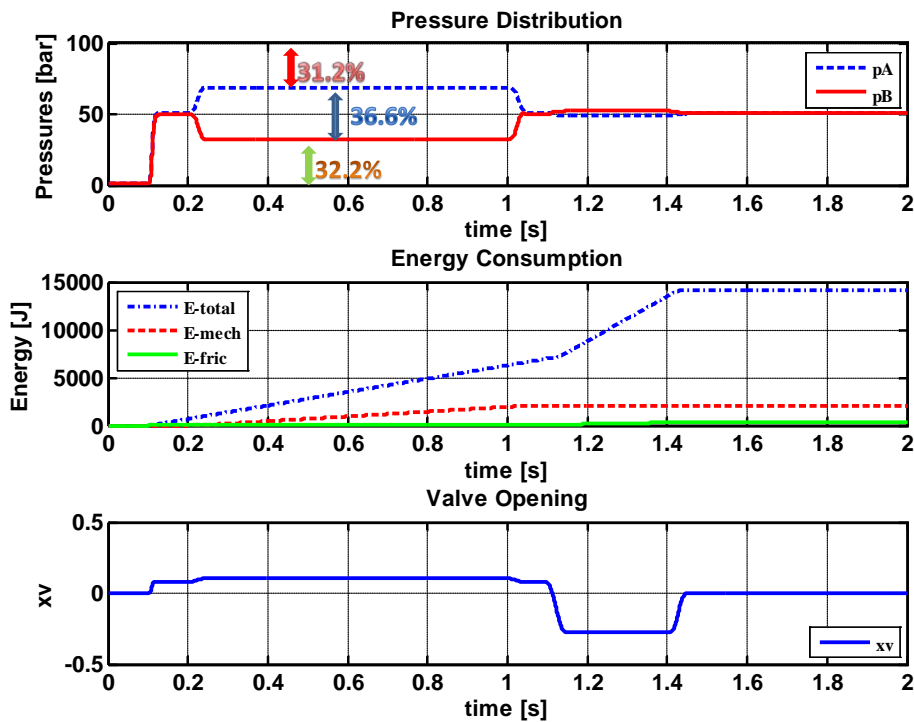
**Figure 38: Pressure Design Requirements**

The valve is chosen to allow sufficiently large flows in order not to run into limitations through the process of simulating different cylinder parameters. After selecting the appropriate cylinder parameters, a smaller valve that fulfills the user requirements is then chosen accordingly. Table 10 illustrates the initial valve parameters. The nominal flow is large as already stated and the bandwidth frequency is 100Hz allowing a fast valve response to changes which results in decreasing the spikes occurred at critical points thus allowing a smoother response.

Valve Parameter Configurations			
Nominal Flow ( $\frac{m^3}{s}$ )	Nominal Pressure drop (bar)	Bandwidth (Hz)	Damping Coefficient
0.0125	100	100	1

**Table 10: Initial Valve Configurations**

The inverse simulation is applied starting with the initial cylinder configurations shown in Table 9. The source pressure input is  $p_s = 100\text{bar}$ . The pressure distribution, the energy consumption, and the valve opening are shown in Figure 39.



**Figure 39: Pressure Distribution, Energy Consumption, and Valve Opening for Initial Configurations**



From the results, it is seen that the cylinder configurations do not fulfill the design requirements. Taking Figure 39, the pressure loss  $p_s - p_A = 31.2\%$  is approximately double the user requirements of 15%. Moreover, the load pressure is only 36.6% of the total allowed pressure resulting in 63.4% pressure losses. The pressures alone indicate that this combination is not efficient and far from the user specifications.

Looking at the energy consumption in Figure 39, the total energy consumed for the whole cycle is high reaching a maximum of 14141.6J. The mechanical energy which is the result from the external force and cylinder speed from the working stroke reaches a maximum of 2000J. The figure shows also the energy resulting from the friction force in the cylinder. The energy loss due to friction is negligible compared to the valve losses.

The valve opening is shown in Figure 39. The valve opens only 10% in the working stroke and 27.5% in the negative direction in the return stroke. The valve is oversized for these system requirements; however, as stated earlier, appropriate valve configuration will be chosen after the cylinder parameters are selected.

In order to reduce the energy consumption, the cylinder piston area is reduced. Table 11 illustrates the parameters of Cylinder 2 where the piston diameter of the initial cylinder is reduced to 91mm thus reducing the area and accordingly the energy consumption. The cylinder is also a synchronizing cylinder in this case.

Cylinder 2 Parameters							
Bore Diameter	Rod Chamber Rod Diameter	Head Chamber Rod Diameter	Stroke Length	Rod Chamber Rod Length	Nominal Flowrate	Nominal Pressure drop	Maximum Rated Pressure
91mm	80mm	80mm	0.3m	0.5m	$0.01 \frac{m^3}{s}$	0.1bar	300bar

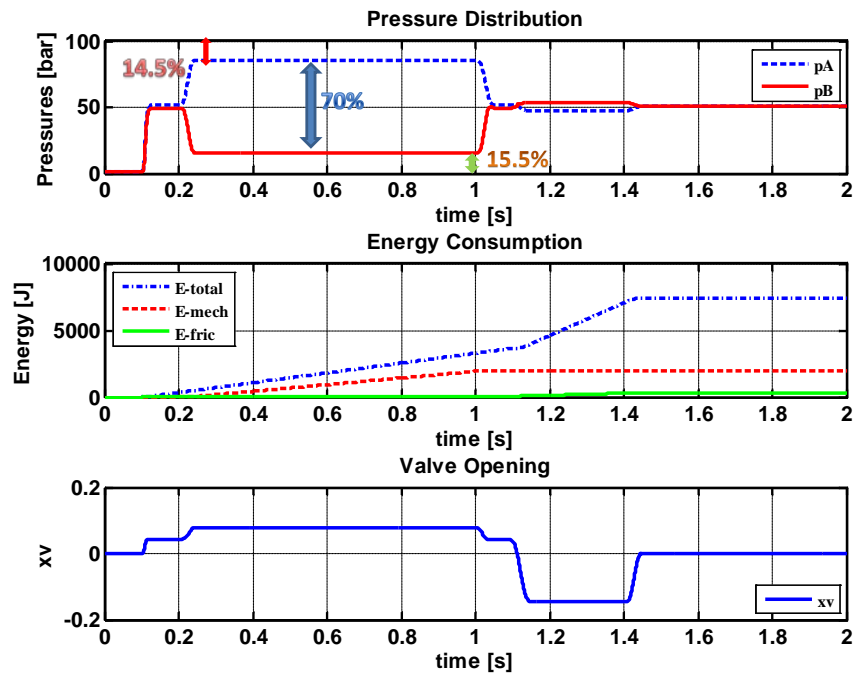
**Table 11: Cylinder 2 Parameters**

The results of the inverse simulation for Cylinder 2 are illustrated in Figure 40.

From Figure 40, the pressure distribution shows that the user requirements are fulfilled where the pressure difference between the pump and the cylinder chamber A is 14.5% which is very close to the 15% safety requirement given by the user. Moreover, the load pressure difference is very large reaching 70% indicating how efficient this cylinder configuration is compared to the initial cylinder.

Figure 40 illustrates the energy consumptions of the system. The total energy consumed reaches 7389J which is approximately half of that consumed by the initial cylinder. This indicates that by decreasing the piston area to by 9mm, 50% of the energy losses are reduced resulting in a more efficient system that is able to provide the user requirements.

Figure 40 represents the valve opening. The valve opens less than that of the previous simulation. This is because the load pressure is higher in this case resulting with higher pressures in ports A and B. Since the flow is the same in the system, the valve opening is less according to equations IV-A-1 and IV-A-2.



**Figure 40: Pressure Distribution, Energy Consumption, and Valve Opening for Cylinder 2**

Since the inverse simulation in the `openHydraulics` library is an easy and important tool to use providing the flexibility to test as many configurations as possible, another cylinder configuration is tested with the aim to further enhance the energy consumption of the system.

Cylinder 3 parameter configuration is shown in Table 12. In this case, the cylinder is a differential cylinder preserving the rod chamber rod diameter to 80mm as specified by the user and decreasing the head chamber rod diameter to 77mm. Moreover, the piston bore diameter is also reduced to 82mm.

Cylinder 3 Parameters							
Bore Diameter	Rod Chamber Rod Diameter	Head Chamber Rod Diameter	Stroke Length	Rod Chamber Rod Length	Nominal Flowrate	Nominal Pressure drop	Maximum Rated Pressure
82mm	80mm	77mm	0.3m	0.5m	$0.01\frac{m^3}{s}$	0.1bar	300bar

**Table 12: Cylinder 3 Parameters**

The results of the inverse simulation of Cylinder 3 are shown in Figure 41. In this simulation, the pump constant pressure is increased to  $p_s = 200\text{bar}$ .

From Figure 41, the pressure  $p_A = 168.8\text{bar}$  at the working cycle which is 15.6%. It fulfills exactly the user requirement. Moreover,  $p_B = 6.15\text{bar}$  thus providing a load pressure of 162.65bar representing 81.2% of the total provided pressure. This result is clearly better than the result of Cylinder 2 providing less pressure losses in the valve.

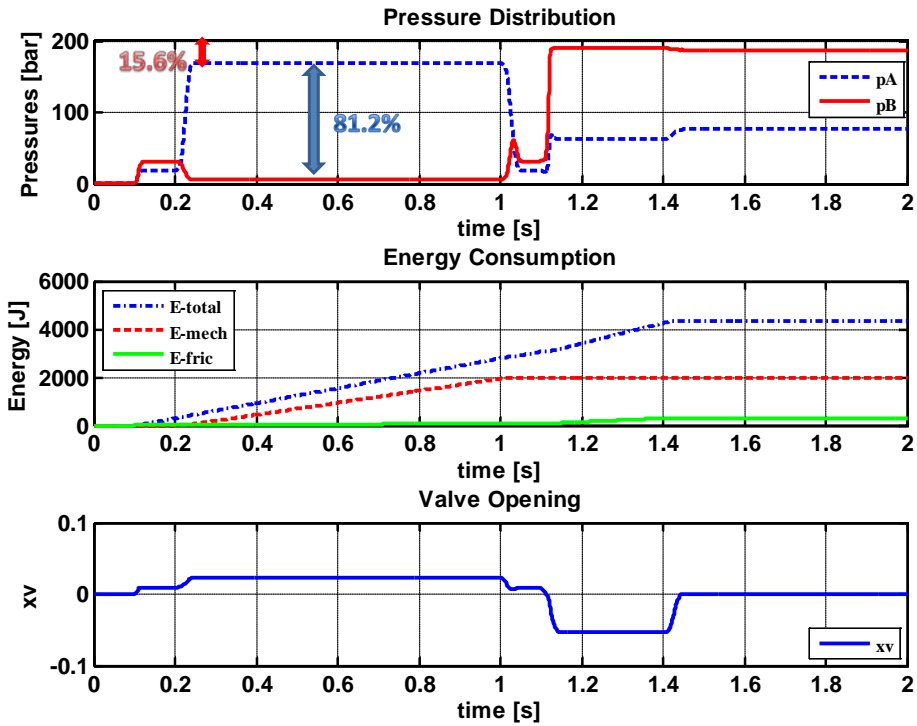
Figure 41 shows the energy consumption of the system. It shows that the total energy consumed in this case reaches 4368J which is way better than that consumed by Cylinder 2 configuration. It consists of 59.1% of the energy consumed by Cylinder 2; thus, saving 40.9% of energy losses. This configuration shows a huge improvement in the efficiency and the optimization of the system and highlights the power of the inverse simulation on selecting appropriate design parameters.

Figure 41 shows the valve opening which is also less than Cylinder 2 for the same reason discussed earlier. It only opens 2.27% in the working stroke and 5.35% in the return stroke. Since Cylinder 3 configurations are chosen as appropriate to design the system, the valve parameters are reduced accordingly. The valve flow is reduced from  $0.0125\frac{m^3}{s}$  to  $0.00083333\frac{m^3}{s}$  thus selecting a smaller valve. Table 13 illustrates the new modified valve parameters and Figure 42 shows the new valve opening. In this case

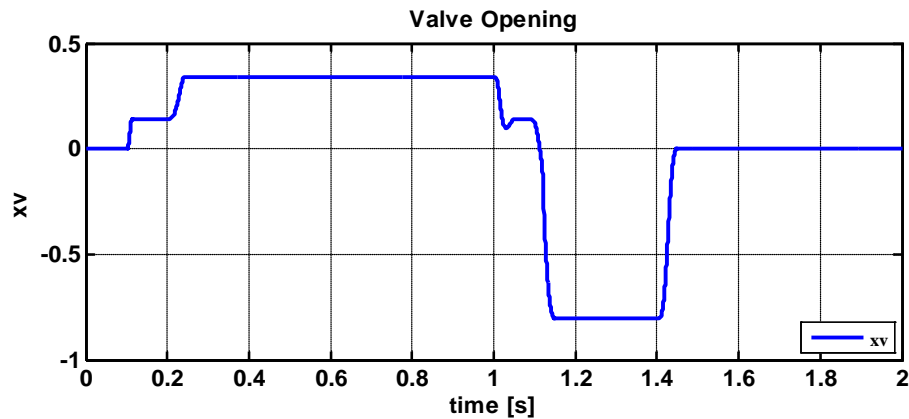
the valve opens around 34% in the working stroke, and then it opens approximately 80% in the negative sense in the return stroke.

Valve Modified Parameter Configurations			
Nominal Flow ( $\frac{m^3}{s}$ )	Nominal Pressure drop (bar)	Bandwidth (Hz)	Damping Coefficient
0.00083333	100	100	1

**Table 13: Valve Modified Parameter Configurations**



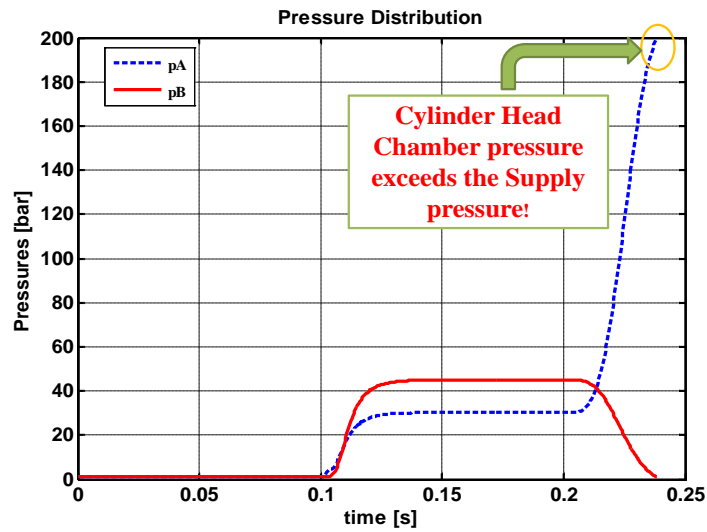
**Figure 41: Pressure Distribution, Energy Consumption, and Valve Opening for Cylinder 3**



**Figure 42: Cylinder 3 Modified Valve Opening**

It is seen now how important the sizing parameters are on the system. They play the major role in the system efficiency and energy consumption. Therefore, it is worth to check what happens if the head chamber rod diameter of Cylinder 3 is increased by only 1mm to become 78mm. Figure 43 illustrates the pressures resulted from the inverse simulation of this system. It is seen that the simulation is not feasible for the whole cycle, because the system with its current inputs and configurations is not physical. It requires a higher input source, because the pressure of the cylinder head chamber is larger than the source 200bar pressure.

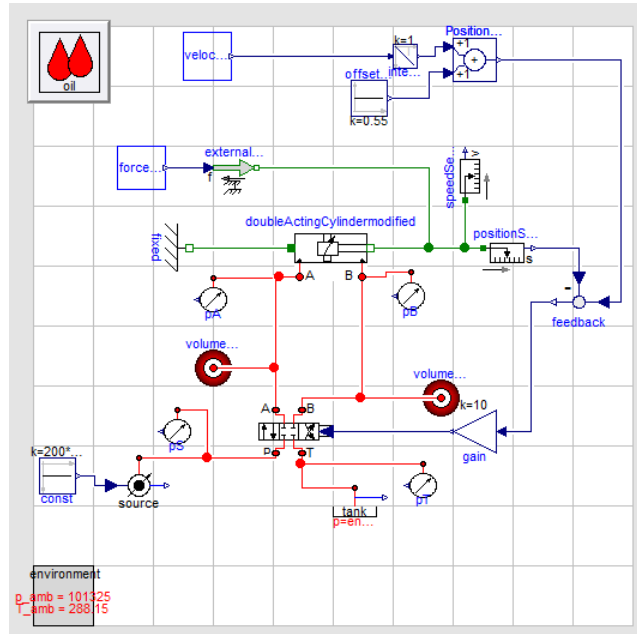
It is interesting how only 1mm can affect the system performance drastically. This is a great advantage that the inverse simulation provides where the user gets acquainted with the effects of the size changes that he makes on the system.



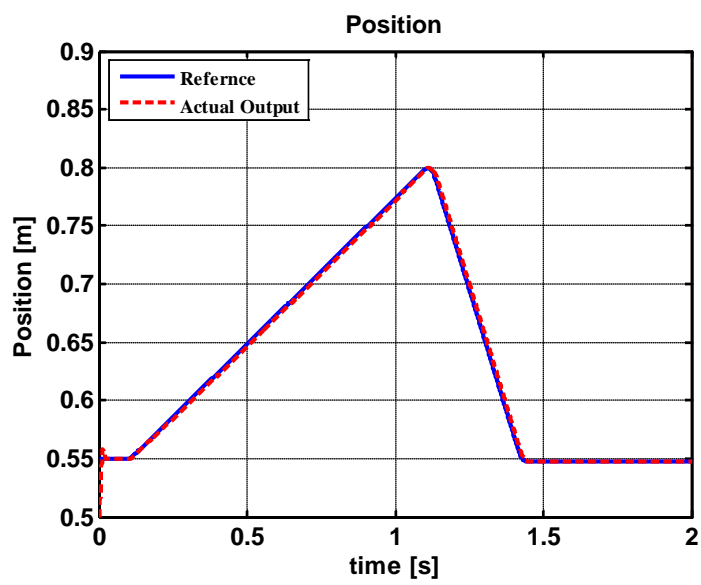
**Figure 43: Cylinder 3 with Larger Head Chamber Rod Diameter**

After selecting the optimal component sizing parameters of the valve and cylinder using inverse simulation, the engineer can now apply the forward simulation using the selected component sizes to tune a controller that will operate the system. This is shown in Figure 44. The input to the closed loop system is the reference position which is subtracted by the actual cylinder position provided by position sensor. This subtraction results with the error that is fed as an input to the proportional P controller that runs the system.

By tuning the system, a P controller gain of value 10 is found reasonable to drive the system. The control response of the actual cylinder position from this controller is shown in Figure 45. In this way, the controller is chosen once at the end of the design stage and after the system optimized parameters are chosen from the inverse simulation. This process saves the engineer a lot of time and effort.



**Figure 44: Gear Hobbing Valve/Cylinder Dymola Forward Circuit**



**Figure 45: Reference Cylinder Position and Actual Control Output**

This section illustrated the benefits of the inverse simulation in designing systems through modifying the design of a gear hobbing servo-drive. It also tested the feasibility of inverse simulation in the modified `openHydraulics` library which is the result



of the work of this thesis. One can say that the design engineers now have a new tool that helps them in designing more efficient systems through selecting optimum parameter sizing without the need to waste time in tuning a controller everytime they alter the system parameters. They only need to select a controller once at the end of the design stage to run their system.

## CHAPTER VIII

### CONCLUSION, LIMITATIONS, & FUTURE WORK

The thesis covers 3 main objectives which are: providing a literature survey on inverse simulation and equation based object oriented modeling language, modifying the existing `openHydraulics` library in Dymola to allow inverse simulations, and inverse simulation of a valve/cylinder example using components of this modified `openHydraulics` library. As a result the feasibility of the library is tested and the importance of inverse simulation in optimizing system designs is illustrated.

The thesis showed the importance of inverse simulation as a tool for design engineers to test and select more efficient systems without the need to tune a controller everytime the system parameters are changed. The engineer only tunes a controller once after the optimum system parameters are chosen. This process saves a lot of time and effort invested by the engineer. However, not all systems are invertible and the user has to provide differentiable inputs to the inverse simulation.

The approach has limitations as well. As systems are put together from components that allow inverse simulation, the overall system may not be invertible even for SISO systems. Further work should be conducted to develop methods to analyze the invertibility of systems.

The modified pipeline discussed in section VI-C has certain limitations. It is not feasible to conduct an inverse simulation with them. The modified pipelines only work with ramp, constant, or sinusoidal inverse system inputs. Combined inputs containing critical transition points such as in the example of section VII and as

discussed in section V-F are a limitation for the use of modified pipelines in inverse simulations. The reasons behind this should be tackled in future research.

Some other future work can be done to enhance the project. For example to check the reasons why some complex systems are hard for the solver to solve. Moreover, similar work can be done on the other hydraulics library in Dymola designed by Modelon to make it feasible also for inverse simulations.

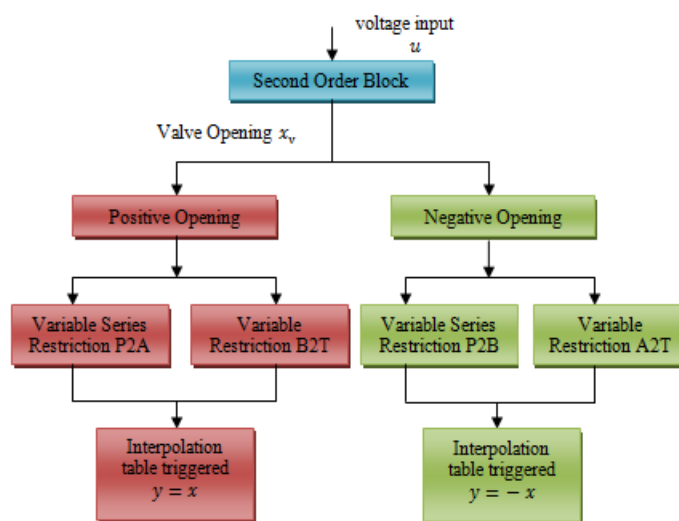
At the end, the thesis provides engineers with a tool that helps them to enhance their designs and produce more efficient systems. The concepts and techniques used in this thesis can be applied to other engineering libraries and disciplines analogically which will provide similar optimized and enhanced systems.

# APPENDIX I

## VALVE

This appendix includes how the valve and cylinder are modeled in the `openHydraulics` library.

Figure 46 shows a model tree of the valve. It illustrates how the valve of Figure 28 functions. The voltage  $u$  is provided as an input to the 2<sup>nd</sup> order block. The output of this block is the valve opening  $x_v$  which is provided as an input to the variable restriction orifices. If  $x_v$  is positive indicating a positive valve opening, this means that the P2A and B2T orifices are triggered providing flow to the cylinder head chamber A and to the tank respectively. In this case, the cylinder moves in the positive sense as indicated in Figure 16. However, if  $x_v$  is negative, P2B and A2T orifices are triggered in this case providing flow to the cylinder rod chamber B and the tank respectively. In this case, the cylinder moves in the negative sense indicated Figure 16.

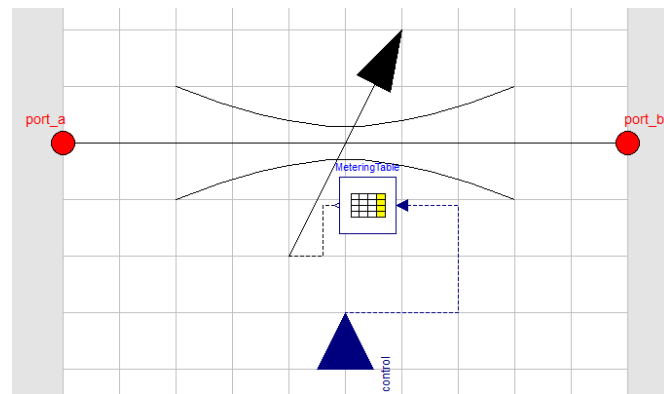


**Figure 46: Valve V4\_3CC Model Tree**

The triggering is done by the parameters of the interpolation metering table, where P2A and B2T tables are triggered by the relation  $y = x$ ; whereas, P2B and A2T tables are triggered by the relation  $y = -x$ . This is shown in Figure 29.

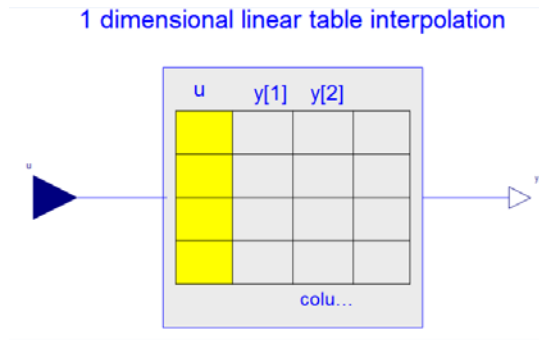
### A. Variable Restriction

The variable restriction model consists of 2 fluid ports and a control input as shown in Figure 47.

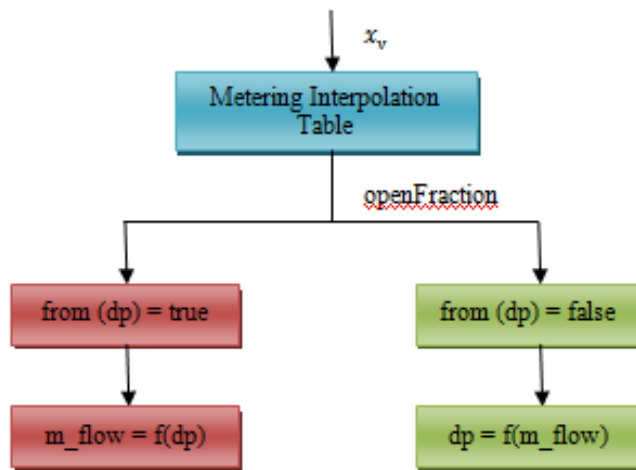


**Figure 47: Variable Restriction**

It includes an interpolation table model called Metering Table which is the same model as `CombiTable1Ds`. The `CombiTable1Ds` shown in Figure 48 is found under the `Modelica.Blocks.Tables` in the Modelica library. The function of this block is to linearly interpolate in one dimension with one input  $u$  and  $n$  outputs. The parameter `columns` defines how many columns to be interpolated from the same input. The grid values of the first column which is the input  $u$  should be defined strictly monotonically increasing. This table causes inversion problems as discussed in section V-E.



**Figure 48: CombiTable1Ds**



**Figure 49: Variable Restriction Model Tree**

Figure 49 illustrates the function of the variable restriction in a model tree. The input to the restriction is the opening  $x_v$ . It is provided as an input to the metering interpolation table, which in turn provides the `openFraction` as its output. The triggering of the table is shown in Figure 29. The function of the variable restriction is to provide the flow rate or the pressure difference as an output depending on the boolean trigger `from (dp)`. In the case of the valve/cylinder inverse simulation, the flow rate is used as an output. The inversion problems of this table are the same as discussed in section V-E.

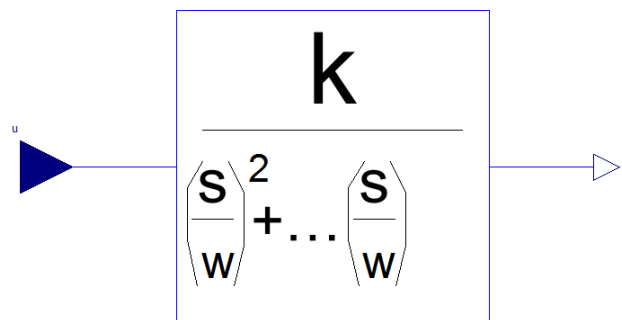
## B. Variable Restriction Series Valve

The variable restriction series valve has the same configuration of components as the variable restriction shown in Figure 47. It is constituted of 2 fluid ports and a control input in addition to the same interpolation table discussed in the variable restriction model which are triggered as shown in Figure 29. The inversion problems of this table are the same as discussed in section V-E.

The difference between the variable restriction series valve and the variable restriction is that the variable restriction series valve only outputs the flow rate as a function of the difference in pressure, but the opposite is not true in this case.

## C. Second Order Block

The dynamic response block in the valve model is taken from the `Second Order` block model found in the Modelica library. The block is shown in Figure 50.

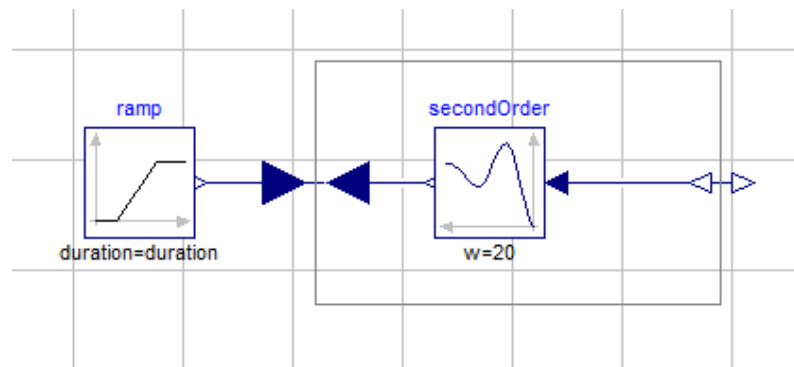


**Figure 50: Second Order Block**

Its function is to define a 2<sup>nd</sup> order function between the input  $u$  and the output of the block  $y$  according to the following relation:

$$y = \frac{k}{\left(\frac{s}{w}\right)^2 + 2 \cdot D \cdot \left(\frac{s}{w}\right) + 1} \cdot u$$

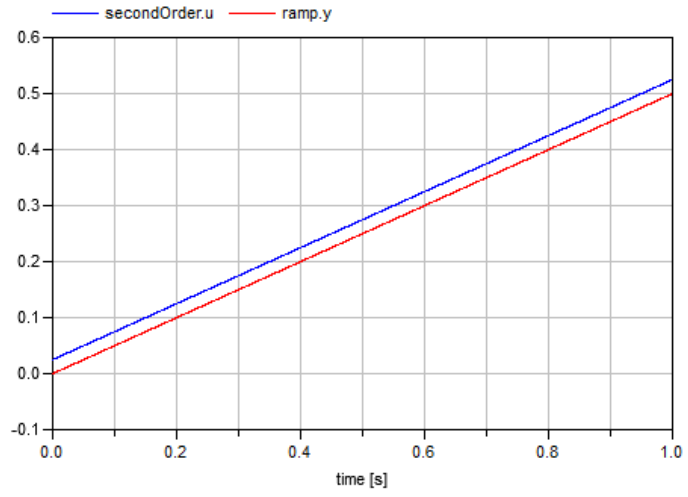
This block is invertible when considered by itself. This is illustrated by the following simple example as shown in Figure 51. The ramp is provided as an input to the input signal connector of the `inverseblockconstraint` which is connected to the output  $y$  of the second order block; whereas, the output signal connector of the `inverseblockconstraint` is connected to the input  $u$  of the second order block. In other words, the input/output boundary conditions of the second order block are interchanged.



**Figure 51: Inverse Second Order Block**

The result of the inverse simulation is shown in Figure 52. It shows the ramp input provided to the output  $y$  connector of the second order block in red and the output of the simulation which is the real input  $u$  in blue.





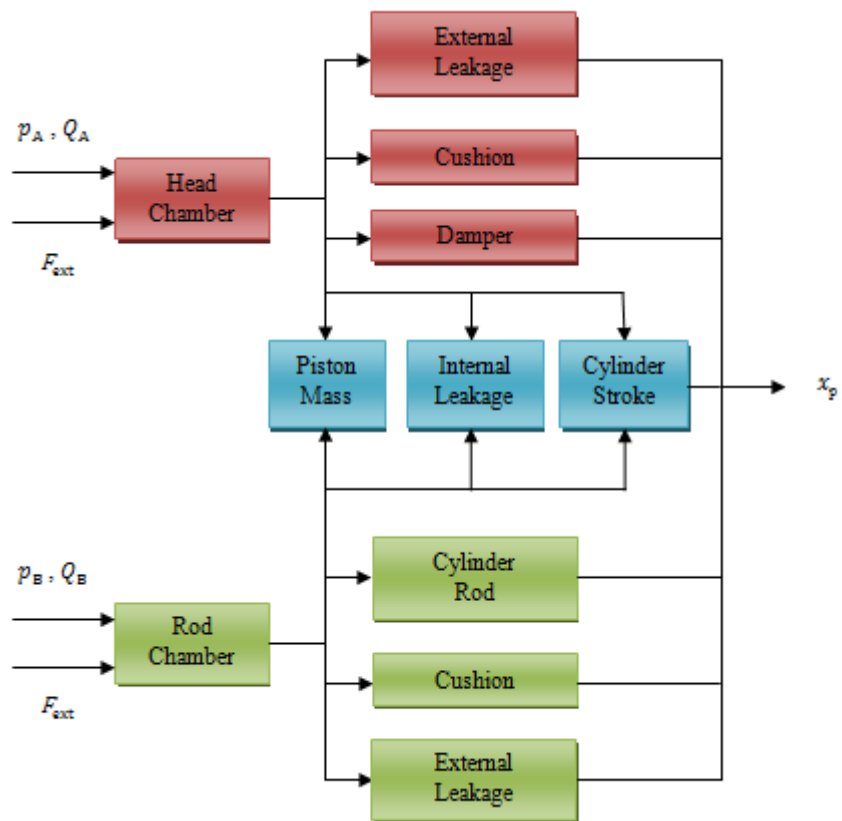
**Figure 52: Second Order Inverse Result**

## APPENDIX II

### CYLINDER

Figure 53 shows the cylinder model tree. The cylinder inputs are the pressure  $p_A$  and flow  $Q_A$  to chamber A and pressure  $p_B$  and flow  $Q_B$  to chamber B in addition the external force  $F_{\text{ext}}$  which can be exerted on either flange A or B. This cylinder does not contain stribek friction, it only contains a damper model taken from the Modelica library. The leakages to the outside environment are modeled by the external leakage models on both chamber sides. Moreover, the internal leakage between both chambers is modeled by the internal leakage model. A cushion is modeled on each chamber to decrease the impact of the piston with the cylinder base once it reaches its end position. The cushion is not necessary to be modeled in inverse simulation, because the advantage of a cushion is found in forward simulation to decrease the piston/cylinder impact. The cylinder is not symmetric. It consists of one rod in the rod chamber, while no rod in the head chamber. The cylinder stroke defines the stroke length that the piston propagates.

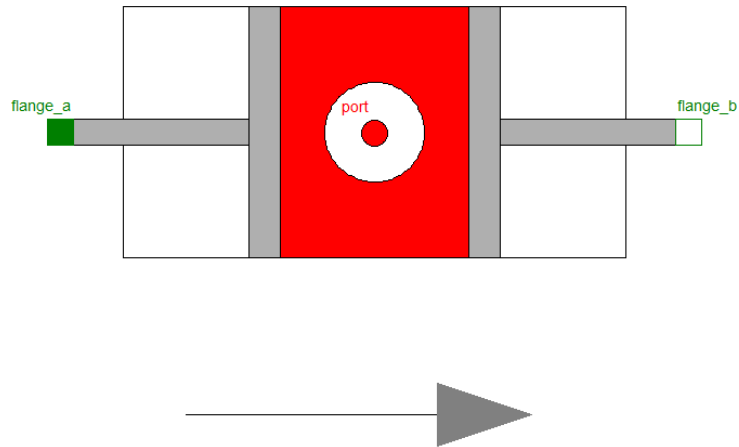
The output of the cylinder is the position of the cylinder rod  $x_p$  and its derivatives which represent the speed and acceleration.



**Figure 53: Cylinder Model Tree**

### A. Chamber Head

The cylinder chamber head component from Figure 31 is taken from the `FluidPower2MechTrans` model found in the Basic section of the `openHydraulics` library. Figure 54 shows the cylinder chamber head component. Flange A is on the cylinder's end and Flange B is on the tip of the piston's end on the side of the head chamber.



**Figure 54: Cylinder Chamber Head**

This model, as can be perceived from its name, relates fluid forces to mechanical forces with their respective work and dynamics involved. It contains an equation to calculate the volume of the fluid within the chamber according to the following equation:

$$V = \max(s_{rel}, 0) \cdot A + V_{res}$$

where  $s_{rel}$  is the relative piston position and  $V_{res}$  is the residual volume. It also calculates the mass of the fluid from the volume and oil density:

$$m = V \cdot \rho_{oil}$$

The speed of the piston is calculated from the derivative of the relative piston position:

$$v_{rel} = \dot{s}_{rel}$$

And the work done by the fluid is calculated through the following equation:

$$w = v_{rel} \cdot (f + f_{contact})$$

where  $f$  is the force exerted and  $f_{contact}$  is the contact force when the end of travel is reached. Moreover, it calculates the force equilibrium equation which is:

$$0 = A \cdot (P_{vol} - P_{ambient}) + f + f_{contact}$$

This model contains an `algorithm` section under which there are 2 `assert` equation conditions. The first `assert` condition is that the Volume  $V > 0$  since the volume in the chamber cannot be negative else it is empty, and the other `assert` condition states the pressure in the chamber  $P_{vol} < \text{max pressure}$ .

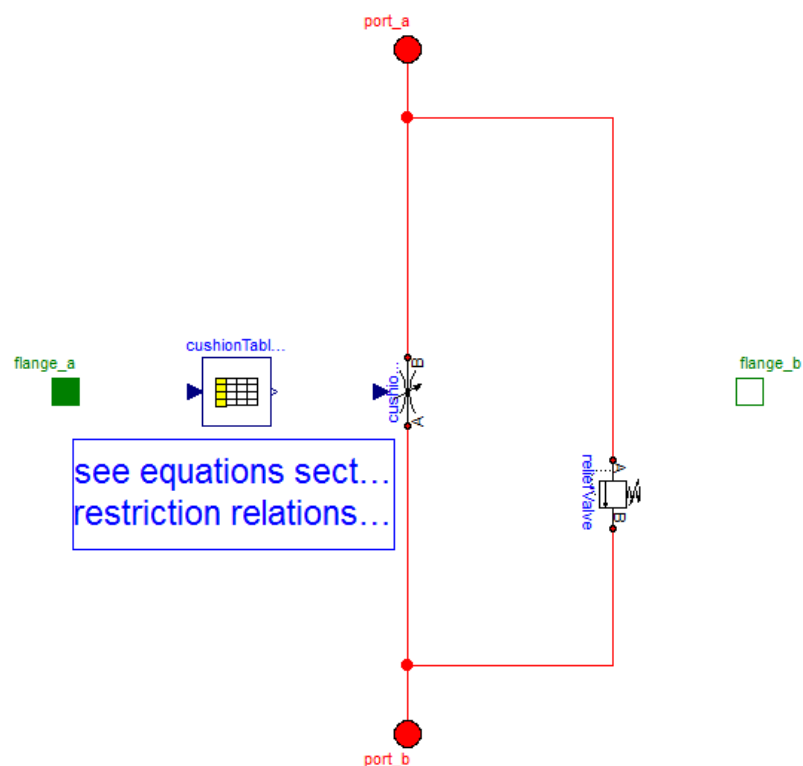
## B. Chamber Rod

The cylinder chamber rod has the same model as the cylinder chamber head shown in Figure 54 which is taken from the `FluidPower2MechTrans` model found in the Basic section of the `openHydraulics` library. The only difference is with area of the chamber used in the calculation, where it is the full area of the piston in the head chamber however it is smaller in the rod chamber side and calculated according to the following equation:

$$A = \frac{\pi}{4} \cdot (D_{piston}^2 - D_{rod}^2)$$

## C. Cushion

There are 2 cushions connected to each cylinder chamber respectively. The aim of the cushion is to decrease the impact between the piston and the cylinder's end. Therefore, instead of having a sudden impulse or impact once the cylinder reaches its end position thus having pressure peaks that may damage the cylinder, the pressure and flow variations may change smoothly using a cushion. It is similar to a damper.

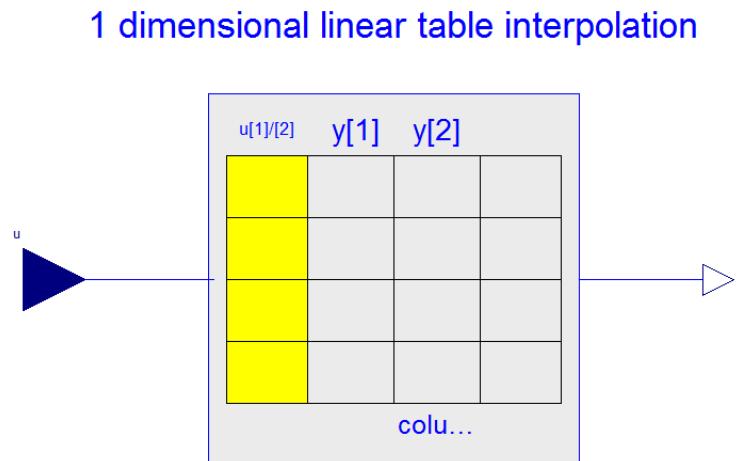


**Figure 55: Cushion Model**

Figure 55 shows the cushion model in the `openHydraulics` library. It consists of 2 mechanical flanges, an interpolation table called `cushionTableBlock`, 2 fluid ports, a cushion restriction, and a relief valve.

## 1. CushionTable

The `cushionTableBlock` uses interpolation table model `CombiTable1D` from `Modelica.Blocks.Tables` in the Modelica library. Figure 56 shows the `CombiTable1D` model.

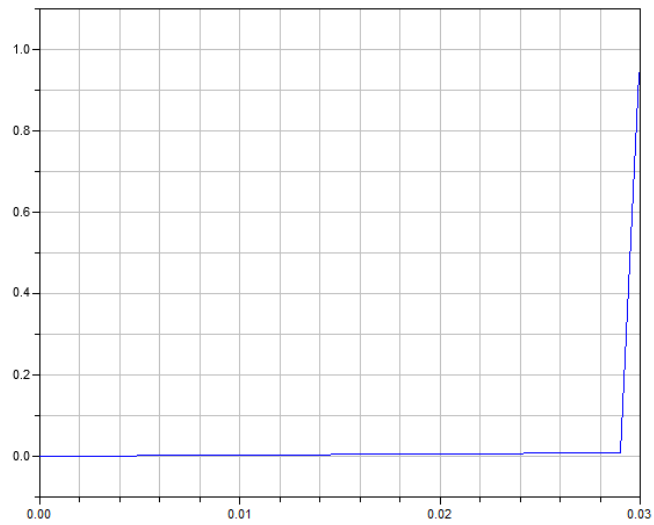


**Figure 56: CombiTable1D**

It is a 1-dimension interpolation table similar to that found in the variable restriction model with the difference that in this case there are n-inputs and n-outputs; whereas in the `ComniTable1Ds` there is 1-input and n-outputs.

The configuration parameters of this table are as follows: [0, 0.001; 0.029, 0.01; 0.03, 1] where the 1st column is the  $s_{rel}$  column and the 2nd column represents fraction of  $Q_{nominal}$ .

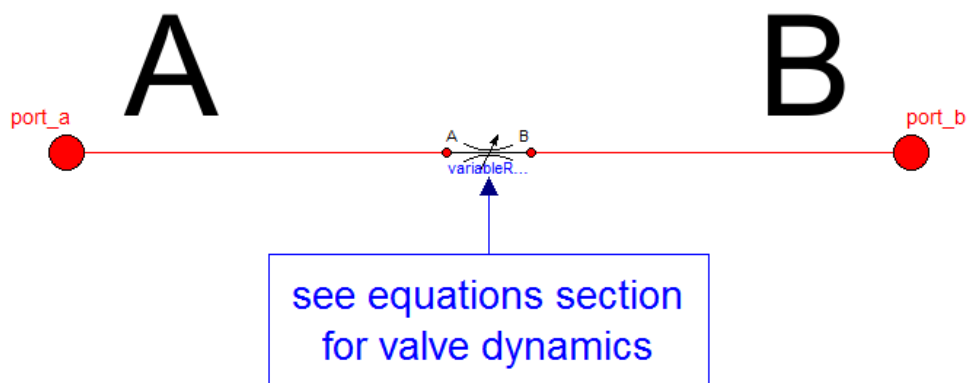
Figure 57 shows the plot of the configuration parameters of the cushion table as specified in the `openHydraulics` library. One should note that the user can specify different parameters relating the  $s_{rel}$  inputs of the table to the  $Q_{nominal}$  outputs depending on his simulation requirements.



**Figure 57: Cushion Table Plot**

## 2. Relief Valve

The relief valve model used in the cushion model is taken from the valves components in the `openHydraulics` library. It consists of 2 fluid ports, and a variable restriction sub-model in addition to the relief valve equations. This is shown in Figure 58.



**Figure 58: Relief Valve**



As stated earlier, the variable restriction model consists of an interpolation table that causes inversion problems.

The relief valve has an algorithm section assigning the control input of the metering table in the variable restriction model to the `valvePosition` real variable used as normalized valve position initialization input between 0 and 1 that can be adjusted manually. The algorithm is the following:

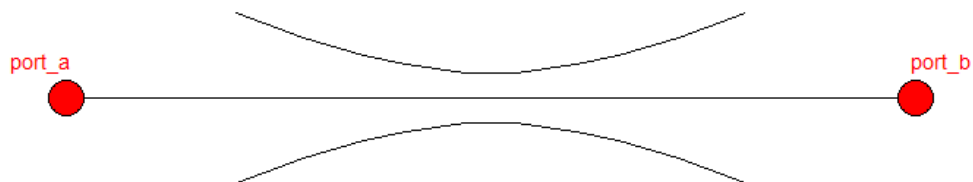
```
variableRestriction.control := valvePosition
```

The algorithm section causes inversion problems as discussed in section V-A if the equation is needed and used during the inverse simulation.

#### D. Leakage

There are 3 leakage models found in the cylinder model. The leakage to the environment on the head chamber side called `leakage_Head2Env`, the leakage to the environment on the rod chamber side called `leakage_Rod2Env`, and the leakage between the head and rod chambers which is the internal leakage called `leakage_Head2Rod`.

These models are based on or taken from the `LaminarRestriction` Model found in the `Basic` section of the `openHydraulics` library and shown in Figure 59.



**Figure 59: Laminar Restriction Model**

This model calculates the conductance of the fluid for laminar flow, where according to Hagen-Poiseuille relation:

$$flow = pressure \times Resistance$$

and the conductance is the inverse of the resistance:

$$conductance = \frac{1}{Resistance}$$

Therefore, this model calculates the conductance according to the following equation:

$$conductance = \frac{\pi \cdot \rho \cdot D^4}{128 \cdot \eta \cdot L}$$

where  $D$  is the restriction diameter,  $\rho$  is the average density,  $L$  is the length of the restriction, and  $\eta$  is the dynamic viscosity.

This model extends the `PartialLaminarRestriction` model which causes inversion problems due to the presence of an algorithm section which states that:

$$portA.Q := conductance \cdot \Delta p$$

This model includes an assert equation to check if the Reynold's number is in the laminar region.

## BIBLIOGRAPHY

Åström, K. J., Elmqvist, H., & Mattsson, S. E. (1998). Evolution of Continuous-Time Modeling and Simulation. *The 12th European Simulation Multiconference*. Manchester.

Bachmann, B. (2012). Mathematical Aspects of Object-Oriented Modeling and Simulation. *9th International Modelica Conference, Tutorial 2*. Munich.

Bals, J., Hofer, G., Pfeiffer, A., & Schallert, C. (2003). Object-Oriented Inverse Modelling of Multi-Domain Aircraft Equipment Systems with Modelica. *Proceedings of the 3rd International Modelica Conference*, (pp. 377-383). Linköping.

Elmqvist, H., & Otter, M. (1994). Methods for Tearing Systems of Equations in Object-Oriented Modeling. *Proceeding ESM'94 European Simulation Multiconference*, (pp. 326-332). Barcelona.

Fritzson, P. (2004). *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*. Linköping: IEEE Press.

Froberg, A. (2006). Inverse Dynamic Simulation of Non-Quadratic MIMO Powertrain Models -Application to Hybrid Vehicles. *Vehicle Power and Propulsion Conference* (pp. 1-6). Windsor: IEEE.

Jelali, M., & Kroll, A. (2003). *Hydraulic Servo-systems* (Vol. 1). London: Springer-Verlag.

Kohmascher, T. (2008). Modellbildung, Analyse, und Auslegung Hydrostatischer Antriebsstrangkonzepete Shaker.

L. Campbell, S., Linh, V. H., & R. Petzold, L. (2011, October 21). *article/Differential-algebraic\_equations*. Retrieved from [www.scholarpedia.org](http://www.scholarpedia.org).

Li Pengfei, L. Y. (July 12-15, 2010). Consistent Initialization of System of Differential-Algebraic Equations for Dynamic Simulation of Centrifugal Chillers. *Internal Compressor Engineering Conference*. Purdue.

Liermann, M. (2012). Backward simulation - A tool for designing more efficient mechatronic systems. *Proceedings of the 9th International Modelica Conference*, (pp. 867-876). Munich.

Lindfield, G. R., & Penny, J. E. (2012). *Numerical Methods using Matlab* (Third Edition ed.). Waltham, MA, USA: Elsevier.

Markel, T. B. (2002). ADVISOR: a system analysis tool for advanced vehicle modeling. *Journal of Power Sources* , 110 (2), 255-266.

Mattson, S. E., & Söderlind, G. (1993). Index Reduction in Differential-Algebraic Equations using Dummy Derivatives. *Siam J. Sci. Comput.* , 14 (3), 677-692.

Mattson, S. E., Olson, H., & Elmqvist, H. (2000). Dynamic Selection of States in Dymola. *Modelica Workshop 2000*, (pp. 61-67). Lund.

Merrit, H. E. (1967). *Hydraulic control systems*. John Wiley and Sons.

Modelon. (2012-2013). <http://www.modelon.com/products/modelica-libraries/hydraulics-library/>. (Modelon) Retrieved from <http://www.modelon.com/>.

Murray-Smith, D. (2000). The inverse simulation approach: a focused review of methods and applications. *Mathematics and Computers in Simulation* , 53, 239-247.

Otter M., T. M. (2005). Nonlinear Inverse Models for Control. *Modelica 4th International Conference*, (pp. 267-279). Hamburg.

Paredis, C. (2013, February 19).

[https://www.modelica.org/libraries\\_old/OpenHydraulics](https://www.modelica.org/libraries_old/OpenHydraulics). Retrieved from [www.modelica.org](http://www.modelica.org).

Saad, J., & Liermann, M. (2013). Inverse Dynamic Simulation of a Hydraulic Drive with Modelica. *Proceedings of the ASME 2013 International Mechanical Engineering Congress & Exposition*. San Diego: ASME IMECE 2013.

Tafazoli S, d. S. (1996). Friction modeling and compensation in tracking control of an electrohydraulic manipulator. *Proc 4th IEEE Mediterranean Symp New Directions in Control and Automation* , 375-380.

Watton, J. (2009). *Fundamentals of Fluid Power Control*. Cambridge University Press.

Wenzhong, G. D., Mi, C., & Emadi, J. (2007). Modeling and simulation of electric hybrid vehicles. *Proceedings of the IEEE* , 95 (4), 729-745.

Wipke, K. C. (1999). ADVISOR 2.1: A User-Friendly Advanced Powertrain Simulation Using a Combined Forward/Backward Approach. *Vehicular Technology, IEEE Transactions* , 48 (6), 1751-1761.

Yong, L. X. (2011, August 17-20). The importance of pump/motor efficiencies for mobile machines operating costs.

Zeitz, M. (n.d.). <http://www.isys.uni-stuttgart.de/lehre/systemdynamik/fls/Hilfsblaetter/PolyRefTrajektorien.pdf>. Retrieved 01 31, 2014, from [www.isys.uni-stuttgart.de](http://www.isys.uni-stuttgart.de).

Zimmer, D. (2011, November 22). [http://www.robotic.de/fileadmin/control/zimm\\_di/Lecture4.pdf](http://www.robotic.de/fileadmin/control/zimm_di/Lecture4.pdf) (Lecture notes). Retrieved from <http://www.robotic.dlr.de>.

