ABSTRACT
Inverse dynamic simulation of hydraulic drives is helpful in early design stages of hydraulic machines to answer the question whether the drive can meet dynamic load requirements and at the same time 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 paper surveys examples of successful use of inverse dynamic simulation in engineering. We use the example of a hydraulic servo-drive to explain the procedure how to generate a state space description of the inverse problem from the given system of differential algebraic equations. Equation based modeling languages such as Modelica lend themselves naturally for inverse simulation because the definitions of which variables of the model are inputs and which are outputs is not made explicit in the model itself.

NOMENCLATURE

\[\begin{align*}
E & \quad \text{Bulk modulus} & & \text{Pa} \\
L & \quad \text{Inductance} & & \text{H} \\
Q & \quad \text{Flow rate} & & \text{m}^3 \text{s}^{-1} \\
S & \quad \text{Cylinder stroke} & & \text{m} \\
V & \quad \text{Cylinder volume} & & \text{m}^3 \\
p & \quad \text{Pressure} & & \text{Pa} \\
u & \quad \text{Input valve voltage} & & \text{V} \\
\alpha & \quad \text{Cylinder chamber areas ratio} & & \text{m}^{-1} \\
\sigma & \quad \text{Viscous friction coefficient} & & \text{N} \text{m} \text{s}^{-1} \\
\gamma & \quad \text{Approximation factor} & & \text{rad}^{-1} \\
\omega_v & \quad \text{Natural undamped valve frequency} & & \text{rad} \text{s}^{-1} \\
A_p & \quad \text{Cylinder piston area} & & \text{m}^2 \\
D_v & \quad \text{Valve damping ratio} & & \text{m}^{-1} \\
F_{c0} & \quad \text{Coulomb friction} & & \text{N} \\
F_{ext} & \quad \text{External force} & & \text{N} \\
F_f & \quad \text{Friction force} & & \text{N} \\
F_{s0} & \quad \text{Static friction} & & \text{N} \\
K_v & \quad \text{Valve voltage gain} & & \text{V} \text{m}^{-1} \\
Q_{Le} & \quad \text{External cylinder Leakage} & & \text{m}^3 \text{s}^{-1} \\
Q_{Li} & \quad \text{Internal cylinder leakage} & & \text{m}^3 \text{s}^{-1} \\
V_{pl} & \quad \text{Bottom dead center volume} & & \text{m}^3 \\
V_0 & \quad \text{Cylinder initial chamber volume} & & \text{m}^3 \\
U_L & \quad \text{Inductor voltage} & & \text{V} \\
c_h & \quad \text{Hydraulic capacitance} & & \text{m}^3 \text{Pa}^{-1} \text{s}^{-1} \\
c_s & \quad \text{Stribeck velocity} & & \text{m} \text{s}^{-1} \\
c_v & \quad \text{Valve flow gain} & & \text{m}^3 \text{V}^{-1} \text{s}^{-1} \\
i_{L1} & \quad \text{Inductor current} & & \text{A} \\
m_t & \quad \text{Total mass} & & \text{kg} \\
x_v & \quad \text{Valve opening} & & \text{m} \\
x_p & \quad \text{Cylinder piston position} & & \text{m} \\
x_p & \quad \text{Cylinder piston velocity} & & \text{m} \text{s}^{-1} \\
x_{\dot{p}} & \quad \text{Cylinder piston acceleration} & & \text{m} \text{s}^{-2} \\
\end{align*}\]
**INTRODUCTION & BACKGROUND**

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 motion/force obtained from the a control input signal.

In case the engineer wants to analyze 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. Therefore, usually design engineers use steady state conditions and engineering assumptions depending on their expertise to design their systems. However, it would be an advantage for engineers to use the same simulation models in early design stages when efficiency of system configurations are analyzed as in stages when the control design takes place. 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. 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 motion/force output is fed as an input to the system thus the output of the simulation corresponds to the control input that would otherwise be generated by the controller.

The backward simulation approach from a modeling point of view is not much different from the forward approach. The only difference is 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 [1]. 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 [2]. 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 assumes that the desired output is met; thus, there is no need for a controller. There is low complexity in models where complicated controller designs from partner companies can be avoided [3]. Inverse simulation can be introduced into early design stages to facilitate the selection and sizing of components.

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 the intention is to select an optimal controller for the system. 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 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.

The backward/forward approach (on the right) of Figure 3 is desired in order to provide better systems. It is desired to have tools that allow combined forward/inverse simulation to be performed using the same models. These tools will facilitate the communication among engineers in the firm having different tasks and will reduce the possibility of errors that might be faced.
To illustrate the concept of backward/forward simulation in this paper, we use 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 [4]. The power of equation based models is that they do not specify a priori which variables are declared as inputs and which are outputs [4]; however, using assignment statements, variables that are assigned on the left-hand side of an equation 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 changing the system inputs. On the other hand, the use of traditional variable assignment tools such as Simulink will require the user to build two separate models, one for the traditional forward approach and the other for the backward approach. This process can even be quite tedious, as this paper will demonstrate [5]. The process of how Modelica handles equations is illustrated in the next section.

**EQUATION MANIPULATION IN MODELICA**

The process of how Modelica simulates a model makes it an important platform to test and use the backward modeling approach. This process of translating a model is shown in Figure 4 and is illustrated in this section with the help of an electric circuit example that is simulated in both forward and inverse fashion.

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 syntax is correct or wrong. After parsing, preprocessing takes place where the model will check whether the classes are used correctly such as the extend command that connects different sub-models. After syntax 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 solved [6].

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

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

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,

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

The following example illustrates the way the equations are automatically manipulated in Modelica. Figure 5 below shows an electric circuit. The circuit consists of a voltage source providing an input alternating voltage amplitude of 10V with a frequency of 10Hz, a resistor of resistance \(R=100\Omega\), and an inductor of inductance \(L=1H\). The output of the system is considered as the current circuit.
After the system model is flattened, the resulting equations are shown below.

\[
\begin{align*}
V_{S2} &= V_{R1} \\
i_{s2} + i_{R1} &= 0 \\
i_{R1} + i_{R2} &= 0 \\
V_{S1} + 10 &= V_{S2} \\
V_{L1} + V_{L2} &= V_G \\
V_{G} &= 0 \\
i_{s2} + i_{S1} &= 0
\end{align*}
\]

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.

Eq.1 \( V_G = 0 \)  
Eq.2 \( U_R = R \cdot i_{R1} \)  
Eq.3 \( V_{R1} + U_R = V_{L1} \)  
Eq.4 \( V_G + 10 = V_{R1} \)  
Eq.5 \( U_L = L \cdot \frac{di_{L1}}{dt} \)  
Eq.6 \( V_{L1} + U_L = V_G \)  
Eq.7 \( -i_{S1} + i_{R1} = 0 \)  
Eq.8 \( -i_{R1} + i_{L1} = 0 \)  
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:

\[
\frac{di_{L1}}{dt} = \frac{U_L}{L}
\]

As seen from Figure 8, the rows of the matrix represent the equations and the columns of the matrix represent the equation variables or unknowns and output.
The aim of the causalization as discussed earlier is to sort the equations in a hierarchical fashion. Therefore, the best sequence that is obtained after performing the causality graph in Figure 7, is a lower triangular form of the incidence matrix. This can be seen in Figure 9.

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

Figure 9: Lower Triangular Form of Incidence Matrix

It is shown from Figure 9 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.

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

Figure 9: Lower Triangular Form of Incidence Matrix

After solving for the forward system, the inverse of this system is solved. Figure 10 shows the inverse circuit which is the same as the forward circuit model with interchanging the input/output boundary conditions. The same technique of equation manipulation is used to find the inverse. 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 divides 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. Therefore, Eq.8 becomes Eq.8i shown.

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

The new causality graph is shown in the Figure 11.

$$\begin{align*}
\begin{array}{c}
V_B = 0 \\
i_C = i_{S1} - i_{B1} \\
i_{R1} = i_{S1} \\
\frac{di_{L1}}{dt} = \frac{di_{R1}}{dt} \\
u_C = L \frac{di_{L1}}{dt} \\
u_R = R \cdot i_{R1} \\
u_B = V_{L1} - u_B \\
u_R = V_{R1} - u_B \\
\end{array}
\end{align*}$$

Figure 11: Causality Graph of Inverse System

The LT incidence matrix is then formed and shown in Figure 12.

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

Figure 12: 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 13.

Matrices that cannot be permuted into LT form such as in Figure 13 are solved by isolating highlighted blocks on the diagonal of the matrix of Figure 13 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.

\[
\begin{array}{ccccccc}
V_G & V_{R1} & i_{R1} & U_R & V_{L1} & i_{S1} & U_L & \frac{di_{L1}}{dt} & i_G \\
Eq.1 & x & x & x & x & x & x & x & x \\
Eq.4 & x & x & x & x & x & x & x & x \\
Eq.8 & x & x & x & x & x & x & x & x \\
Eq.2 & x & x & x & x & x & x & x & x \\
Eq.3 & x & x & x & x & x & x & x & x \\
Eq.7 & x & x & x & x & x & x & x & x \\
Eq.6 & x & x & x & x & x & x & x & x \\
Eq.5 & x & x & x & x & x & x & x & x \\
Eq.9 & x & x & x & x & x & x & x & x \\
\end{array}
\]

**Figure 13: Block Lower Triangular Form of Incidence Matrix**

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 [7]. 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 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 [8]:

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

If \(y = f(x) = \text{residual}\) 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 \(x_{n+1}\). This can be written in pseudo code as:

```plaintext
while |\(\frac{g'(a_n)}{g(a_n)}\)| > tolerance
\[a_{n+1} = a_n - \frac{g(a_n)}{g'(a_n)};\]
iteration = iteration + 1;
\[a_n = a_{n+1};\]
\[g(a_n), g'(a_n)\]
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 4 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 only in Modelica. Thus, Modelica modeling language is a valuable and easy tool to perform inverse or backward simulations.

**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 14.
MODELING OF VALVE

The valve modulates the power provided by the pressure source and delivered to the cylinder. As shown in Figure 14, 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:

\[
Q_A = c_A \text{sgn}(x_v) \text{sign}(p_s - p_A) \sqrt{|p_s - p_A|} \\
Q_A = c_A \text{sgn}(-x_v) \text{sign}(p_A - p_T) \sqrt{|p_A - p_T|}
\]

\[
Q_s = c_s \text{sgn}(x_v) \text{sign}(p_s - p_B) \sqrt{|p_s - p_B|} - c_s \text{sgn}(x_v) \text{sign}(p_B - p_T) \sqrt{|p_B - p_T|}
\]

where

\[
\text{sgn}(x_v) = \begin{cases} 
  x_v & \text{for } x_v \geq 0 \\
  0 & \text{for } x_v < 0
\end{cases}
\]

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. These characteristics involve many parameters which are hard to find accurately; however, manufacturers' catalogues often show frequency response plots which can be used to identify the dynamics of the spool position control system. Often a 2nd order approximation is sufficient:

\[
\frac{1}{\omega_v^2} \ddot{x}_v + 2\zeta \frac{\omega_v}{\omega_v} \dot{x}_v + x_v = K_v u
\]

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:

\[
Q_A + Q_{Li} = V_A + \frac{V_A}{E(p_A)} \dot{p}_A
\]

\[
Q_B - Q_{Le} = V_B + \frac{V_B}{E(p_B)} \dot{p}_B
\]

The volumes of the chambers are given by:

\[
V_A = V_{A0} + x_p A_p
\]

\[
V_B = V_{B0} - x_p A_p
\]

The pressure dynamics equations are as follows:

\[
\dot{p}_A = \frac{1}{c_{hA}} (Q_A - A_p \dot{x}_p + Q_{Li})
\]

\[
\dot{p}_B = \frac{1}{c_{hB}} (Q_B + A_p \dot{x}_p)
\]

The hydraulic capacitances of each of the 2 chambers are:

\[
c_{hA} = \frac{V_{pA} + \left(\frac{S}{2} + x_p\right)A_p}{E_A(p_A)}
\]

\[
c_{hB} = \frac{V_{pB} + \left(\frac{S}{2} - x_p\right)A_p}{E_B(p_B)}
\]

Newton's 2nd law is applied in order obtain the equation of motion for the cylinder.

\[
m \ddot{x}_p + F_l(x_p) = (p_A - p_B)A_p - F_{ext}
\]

Where \( m \) 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 \( m_{B,fl} \).

The Striebeck 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:

\[
F_l(x_p) = \sigma \dot{x}_p \left[ F_{c0} + F_{c0} e^{-\frac{|x_p|}{c_s}} \right]
\]

In order to make it continuous and monotonous, an approximation can be used [10]. This approximation makes the function invertible:

\[
\text{sign}(\dot{x}_p) \approx \frac{2}{\pi} \cdot \arctan(y\dot{x}_p)
\]

thus,

\[
|\dot{x}_p| = \frac{2}{\pi} \cdot \arctan(y\dot{x}_p)
\]

INVERSE SIMULATION AND EFFICIENCY ANALYSIS OF SYSTEM

The explicit state differential equations in this case are as follows:

\[
\dot{\dot{x}}_p = \frac{1}{c_{hA}} (Q_A(x_v, p_A) - A_p \dot{x}_p + Q_{Li})
\]

\[
\dot{\dot{p}}_B = \frac{1}{c_{hB}} (Q_B(x_v, p_B) + A_p \dot{x}_p - Q_{Li})
\]
where $Q_A$ and $Q_B$ are functions of $x_v$ as illustrated by equations 5.1-1 and 5.1-2 respectively.

The incidence matrix for the forward simulation is created and is shown in Table 1. The piston position $x_p$ 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 5.3-17 and 5.3-18 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.

### Table 1: Forward Simulation Lower Triangular Form

<table>
<thead>
<tr>
<th></th>
<th>$\dot{x}_p$</th>
<th>$\dot{p}_A$</th>
<th>$\dot{p}_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3-19</td>
<td>$x$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3-17</td>
<td></td>
<td>$x$</td>
<td></td>
</tr>
<tr>
<td>5.3-18</td>
<td></td>
<td></td>
<td>$x$</td>
</tr>
</tbody>
</table>

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 5.3-17, 5.3-18, and 5.3-19 of the forward simulation. In this case, equation 5.3-19 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:

$$\ddot{x}_p = \frac{1}{m_r} [(p_A \cdot c_B) \cdot \dot{p}_A \cdot \dot{F}_\text{ext} - F_\text{f} (\dot{x}_p)]$$  \hspace{1cm} \text{5.3-19}$$

A Dummy-derivative method can be used then [11] 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 $p_B^\prime$ and its relative state is called dummy state. Therefore, the system will consist now of 5.3-17, 5.3-18, 5.3-19, and 5.3-20i 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 [12].

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

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

<table>
<thead>
<tr>
<th></th>
<th>$\dot{p}_A$</th>
<th>$x_v$</th>
<th>$\dot{p}_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3-17</td>
<td>$x$</td>
<td>$x$</td>
<td></td>
</tr>
<tr>
<td>5.3-18</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>5.3-20i</td>
<td>$x$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In order to solve this BLT, one of the ways used by Modelica is through the use of symbolic manipulation by making equation 5.3-20i an explicit function of $x_v$ through replacing equations 5.3-17 and 5.3-18 in equation 5.3-20i. After manipulating 5.3-20i 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 in equation 5.3-21.

$$x_v = \frac{(A_p \cdot \dot{x}_p + A_p \cdot \dot{v} \cdot a^2 / a \cdot b \cdot (p_B - p_A) + m \cdot \dot{x}_p + \dot{F}_v (x_v) + \dot{F}_\text{ext})}{A_p}$$ \hspace{1cm} \text{5.3-21}$$

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 above.

### Table 3: Inverse Simulation Lower Triangular Form

<table>
<thead>
<tr>
<th></th>
<th>$x_v$</th>
<th>$\dot{p}_A$</th>
<th>$\dot{p}_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3-21</td>
<td>$x$</td>
<td>$x$</td>
<td></td>
</tr>
<tr>
<td>5.3-17</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>5.3-18</td>
<td>$x$</td>
<td></td>
<td>$x$</td>
</tr>
</tbody>
</table>

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

$$\ddot{x}_v = -2D_v \omega_s \dot{x}_v - \omega_s^2 x_v + K_v \omega_s^2 u$$ \hspace{1cm} \text{5.3-22}$$

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

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

<table>
<thead>
<tr>
<th></th>
<th>$x_v$</th>
<th>$u$</th>
<th>$\dot{p}_A$</th>
<th>$\dot{p}_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3-21</td>
<td>$x$</td>
<td></td>
<td>$x$</td>
<td></td>
</tr>
<tr>
<td>5.3-22</td>
<td>$x$</td>
<td></td>
<td>$x$</td>
<td></td>
</tr>
<tr>
<td>5.3-17</td>
<td>$x$</td>
<td>$x$</td>
<td></td>
<td>$x$</td>
</tr>
<tr>
<td>5.3-18</td>
<td>$x$</td>
<td></td>
<td></td>
<td>$x$</td>
</tr>
</tbody>
</table>

Figure 15 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.

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 $u$ as it was shown. 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. It was shown that it is feasible to build separate valve and cylinder models and simulate the system inversely; however, currently the two hydraulic libraries available in Modelica do not allow the inverse simulation of hydraulic drive for reasons that are still under investigation by the authors of this paper.

Table 5 represents some of the parameters that were used in the simulation of our model. Because the aim of the inverse or backward simulation is to help the design engineer in his selection of the sizing parameters of the system, 2 cases for different piston diameter sizes are considered in the simulation and the results of the 2 simulations are then compared in order to see which size is more efficient to design the system knowing that the design parameters provided in this paper are not the best choices but just to provide an idea about the technique and that the design engineer should keep simulating to have an optimal design.

Figure 16 shows the Blocks.Math.InverseBlockConstraints that was used to directly interchange system inputs and output; thus, providing the real output desired of our system which is the cylinder position $x_p$ as an input and calculating inversely the required input which is the voltage $u$. An important note when applying the inverse is that the input to the system should be differentiable at least to the number of times one needs to differentiate it for example in our case to acquire an acceleration which is minimum twice differentiable. This is done either by introducing a filter to the input signal or the signal is differentiable by itself as in our case where it is a sine wave [3].

<table>
<thead>
<tr>
<th>Table 5: Simulation Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{ext}$</td>
<td>1000 N</td>
</tr>
<tr>
<td>$p_s$</td>
<td>75E5 Pa</td>
</tr>
<tr>
<td>$p_T$</td>
<td>0 Pa</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1 (synchronizing cylinder)</td>
</tr>
<tr>
<td>$c_V$</td>
<td>$1.78174 \cdot 10^{-7}$</td>
</tr>
<tr>
<td>$c_s$</td>
<td>$0.01 \frac{m}{s}$</td>
</tr>
<tr>
<td>$D_v$</td>
<td>0.5</td>
</tr>
<tr>
<td>$K_v$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\omega_v$</td>
<td>$628.32 \ \text{rad} \frac{s}{s}$</td>
</tr>
<tr>
<td>$m_i$</td>
<td>19.523 kg</td>
</tr>
<tr>
<td>$D$ (piston large diameter)</td>
<td>0.04 m / 0.05 m</td>
</tr>
<tr>
<td>$d$ (piston small diameter)</td>
<td>0.025 m</td>
</tr>
<tr>
<td>$S$</td>
<td>0.2 m</td>
</tr>
<tr>
<td>$E$</td>
<td>15000E5 Pa</td>
</tr>
<tr>
<td>$Q_{li}$</td>
<td>0</td>
</tr>
<tr>
<td>$Q_{le}$</td>
<td>0</td>
</tr>
<tr>
<td>$V_{p1,a}$</td>
<td>$10^{-3} \text{ m}^3$</td>
</tr>
<tr>
<td>$V_{p1,b}$</td>
<td>$10^{-3} \text{ m}^3$</td>
</tr>
<tr>
<td>$F_{c0}$</td>
<td>100 N</td>
</tr>
<tr>
<td>$F_{s0}$</td>
<td>100 N</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>800</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$10^4 \frac{N \text{ s}}{m}$</td>
</tr>
</tbody>
</table>

The system model setup in Modelica is seen in Figure 16 below.

Figure 16 shows the output desired for our piston which is fed as input to the inverse simulation.

![Figure 15: Backward Simulation Approach in Simulink](image1)

![Figure 16: Dymola System Model](image2)
The output of the simulation, which is the real voltage input to the valve is shown in Figure 18 for two different configurations. It is seen that the voltage required for the valve to provide this output position peaks to -12.92V at 0.5s from start of simulation for a cylinder piston outer diameter of D=0.05m chosen by the design engineer; whereas, it peaks at around -6.51V for a smaller chosen piston outer diameter of D=0.04m.

In terms of power consumption, Figure 19 and Figure 20 illustrate each of the power consumption for the 2 cases. The figures show that for a bigger piston diameter, the power input peaks at 2231.31W at t=0.5s and the power losses in the valve which also peaks at 1866.31W at t=0.5s are higher than those for a smaller piston diameter where the input power required peaks at 1398.54W at t=0.5s and the maximum power loss from the valve is 1033.6W at t=0.5s.

Moreover, and most importantly, the efficiency of the system in both cases is compared and shown Figure 21. It is seen that for the smaller piston diameter, the efficiency is higher than for a larger diameter. However, one should note that the efficiencies are still low and the design engineer can adjust his parameters in order to reach more efficient systems.

One more comparison obtained from the simulation and seen in Figure 22 is that the valve is undersized in the case of the larger cylinder piston indicating the need of a larger valve with a larger opening; whereas, for the smaller cylinder, the valve does not operate above its maximum opening limits thus it is working in range and its size allows the system to reach its desired output.

Eventhough the valve allows the system having the smaller cylinder to operate, it does not mean that it is the best valve choice, because with the help of the inverse simulation, the
design engineer is entitled to test all the parameter combination possible to select the most efficient valve and cylinder size to be implemented in the later stages of design.

**CONCLUSION**

This paper illustrates the concept of inverse or backward simulation and its importance when applied on the design stage of systems. As illustrated, it is the same as the forward simulation with the interchange of the input/output boundary conditions. It is tedious to obtain the inverse systems by hand; however, Modelica with its equation manipulation techniques provides a great advantage because it allows the inversion of systems automatically. This paper demonstrates the importance of inverse simulation in Modelica. We believe that Modelica can become a platform on which design engineers can rely on for system sizing and control design at the same time.

The example of hydraulic servo-drive demonstrated the advantages that the backward or inverse simulation provides to the design engineer in selecting his system design parameters before starting to apply a controller to the system. In this way, better and more efficient systems consuming less energy can be designed in the future. The hydraulic servo-drive example did not make use of already available libraries. Currently, the authors of this paper are investigating the reasons why the current hydraulic libraries in Modelica do not allow inverse simulations and develop an alternative library which will be compatible with both forward and inverse simulations.

**ACKNOWLEDGMENTS**

The authors would like to express their thanks to the American University of Beirut for its support.

**REFERENCES**


