*International Journal of Fluid Power*, 2015
Vol. 16, No. 3, 163–177, http://dx.doi.org/10.1080/14399776.2015.1107386

Taylor & Francis
Taylor & Francis Group

# Improved sizing of hydraulic servo-drives through inverse simulation approach using Modelica and modified OpenHydraulics library

Joseph Saad and Matthias Liermann*

*Department of Mechanical Engineering, American University of Beirut, Beirut, Lebanon.*

(*Received 26 March 2015; accepted 9 October 2015*)

The inverse dynamic simulation allows a designer to test the dynamic performance of a closed loop controlled drive without the need to parametrize a feedback controller. Equation based object oriented modeling languages such as Modelica are suitable to build models that can be simulated both, in a forward or inverse fashion. The same model can be used to simulate either in forward fashion or inversely, depending purely on which boundary conditions are specified. To evaluate the usefulness and limitations of the inverse simulation approach for the sizing of a hydraulic servo drive, an open source Modelica library OpenHydraulics is modified to enable inverse simulation. The library is then used for a case study about the sizing of a valve/cylinder hydraulic servo-drive. The inverse simulation supports an intuitive, iterative design approach towards the sizing problem of closed loop controlled drives, such as hydraulic or electric servo drives. The designer can change the size of system components and at any time assess the drive's efficiency for a typical loading cycle without the need to implement feedback control. It is shown in a test case of an existing hydraulic servo drive of a tooling machine, how changes in design parameters can be instructed through using the inverse simulation approach. In this case, the energy losses could be decreased by 69 %.

**Keywords:** Inverse simulation; inverse dynamics; fluid power; design; Modelica; hydraulic servo-drive; efficiency; simulation; equation based modeling; design optimization; gear hobbing machine; sizing problem

## 1. Introduction

System simulation has become a mandatory step in designing and optimizing mechanical systems. Our aim with this paper is to evaluate whether inverse simulation with standard Modelica libraries can be a useful tool for engineers to *intuitively* improve hydraulic servo axes sizing in an early stage of design with respect to efficiency and fulfillment of motion requirements. Modelica is a widely known equation based modeling language. It is a meta-language that is used to define dynamic multi-physics lumped parameter simulation models. The findings of this paper are transferable to other equation based modeling languages.

It can be argued that the sizing problem of a hydraulic axis can be supported efficiently with conventional forward simulation using batch simulations or optimization algorithms. However, forward simulation always requires the implementation of a feedback control and therefore the complexity of the sizing problem is increased by at least one additional parameter (assuming proportional control). A technical sales engineer or a system development engineer finds this confusing because both, the change of system sizing parameters and change of control parameters affect the performance. It is not always obvious what the limit of performance is with a certain configuration because the controller plays such an important role. On this background the need to support design engineers with assistance in the control design has clearly been recognized. Many research projects were conducted to automate the control design step, see also Schlemmer and Murrenhoff (2007) and Liermann and Murrenhoff (2005), however without breakthrough in widespread application.

Using inverse simulation is another approach to the same problem. Inverse simulation means that actuator motion and forces are provided as constraint equations to the model and the result of the simulation are the trajectories of system states and required control inputs. In this paper we describe a method in which the same simulation model can be used for the sizing problem with inverse simulation and with only minor modifications to develop a suitable controller in forward simulation mode. This is demonstrated in Figure 1 which shows the similarity of an inverse and forward simulation implementation of a hydraulic servo-axis. The model of the hydraulic system implemented with the equation based modeling language Modelica is identical for both cases. Note that in the inverse simulation case the motion requirement and external forces are given as constraints to the actuator motion, while in the forward simulation case the required motion is given as a reference input to the feedback control.

As has been shown in Liermann (2012) and Saad and Liermann (2013), the hydraulic servo-axis dynamics is invertible and should in principle automatically work with a Modelica implementation of the model. However, the specific implementation of a model library can cause

*Email: matthias.liermann@aub.edu.lb

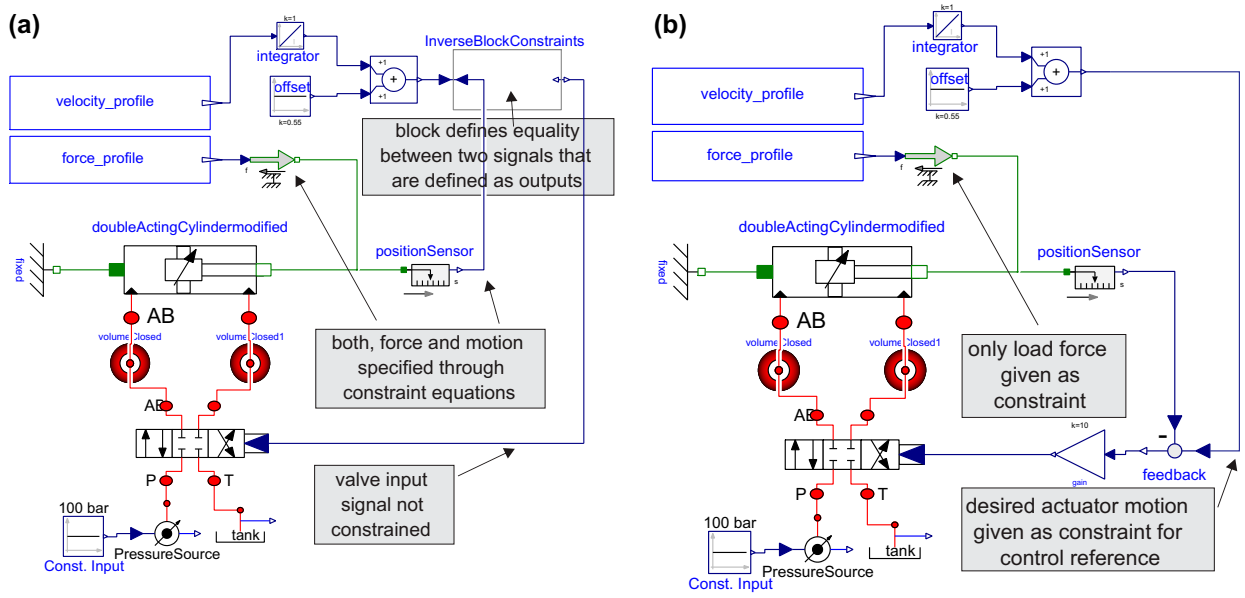



Figure 1. Inverse (a) and forward (b) simulation of servo axis.

inverse simulations to fail. This is the case with all available commercial and open source Modelica libraries for hydraulic systems. Previous studies (Liermann 2012; Saad and Liermann 2013) have therefore only used simplified, custom made models to demonstrate the feasibility of the inverse simulation concept with hydraulic drives. In the frame of this work we modified the Modelica open source library OpenHydraulics to render it suitable for inverse simulation and for its use for the sizing problem of a hydraulic servo-drive. The following section presents a brief overview over the use of inverse simulation in the area of motion control.

### 1.1. *Inverse dynamics simulation*

There are many applications for inverse simulation in motion control. A prominent area of application is in the field of robotics. It is frequently of interest to retrieve the inverse dynamics of multiple degree of freedom robotic manipulators. The aim is to study optimum trajectories and related drive shaft accelerations in order to know the required actuation power of the motors and to keep them within rated limits (Luca and Book 2008). The inverse simulation approach has also attracted attention for the solution of tracking control problems Clayton et al. (2008); Devasia (2002). It allows the generation of feed-forward control inputs and reduces the tracking error significantly.

There are different ways to obtain the inverse dynamics. The most obvious approach, one would assume, is to solve the differential algebraic equations analytically. While this is straightforward for linear systems, it is often more complex and sometimes impossible for nonlinear systems. However, the use of equation based modeling languages, such as Modelica, have led to new approaches where the inverse dynamics can be automatically obtained from nonlinear models (Gräber 2014; Thümmel et al. 2005). This results in controllers that are valid for full operating regions of the plant. In fact, even if a model inverse cannot be found analytically with the algebraic methods of Modelica simulators, the inverse can still be computed numerically. Background information on the equation handling of Modelica compilers can be found in Cellier and Elmqvist (1993), Carpanzano and Maffezzoni (1998) and Fritzson (2014).

Since obtaining the analytical inverse system dynamics is often difficult to achieve manually, various techniques have been developed which approximate the inputs to dynamic systems based on repeated solution of conventional forward simulation models. In his review on the inverse simulation approach, Murray-Smith (2000) concentrates on problem scenarios where it is assumed that the explicit state equation for the forward dynamics is available. It is stated that in this case 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 (Hess et al. 1991).

A third approach to system inversion is based on high-gain feedback, see Figure 2. When the forward plant dynamics is placed into a unity feedback loop with a high gain $K$, the error $e$ between reference trajectory $y_{\text{ref}}$ and output $y^\star$ is small. That means that $u^\star$ is a good approximation for the required input to achieve the reference

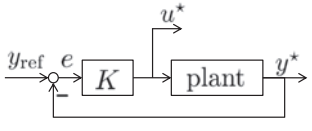



Figure 2. Feedback based inversion.

trajectory $y_{ref}$. Buchholz and von Grünhagen (2007) present a good introduction into this method and apply it to a helicopter flight dynamics model. In the review on feedback control based system inversion, it is pointed out in Murray-Smith (2011), that this alternative to system dynamics inversion is potentially very fast. Issues with the method are faced for high feedback gains when the closed loop behavior becomes oscillatory or unstable. Unfortunately this is frequently the case for hydraulic servo-systems. Applications and discussions of this method can be found in Murray-Smith (2014) for the dynamic analysis of an underwater vehicle and in Tagawa et al. (2011) for the control of a hydraulic actuator.

Besides the inverse dynamics problem in robotics and the nonlinear feedforward control applications, another interesting application of inverse simulation is found in the field of design, in particular for component sizing of closed loop controlled drives. For the sizing of a drive train for a motion control problem, it is important to know the dynamic loads for each component. For drives that consist of only a few components, such calculations can be made with simple tools, and by reducing all but the dominant dynamic components to quasi-static models. Whereas for more complicated systems, dedicated simulation software solutions can be used. One such example is the advanced vehicle simulator ADVISOR that was developed in 1994 by National Renewable Energy Laboratory to support the US Department of Energy hybrid propulsion system (Markel et al. 2002). ADVISOR simulates vehicle performance on standard driving cycles with a combined backward/forward approach (Wipke et al. 1999). Drive train component models are modeled with steady state characteristics only. The vehicle acceleration is considered with a simple inverse dynamic model and fed with the required drive cycle velocity. The implementation does not require a driver; instead, the needed acceleration force is calculated with an inverse model of the vehicle dynamics from the desired velocity profile. Bond graph models lend themselves naturally to an inverse simulation approach as explained in Feki et al. (2008). Sizing of hybrid drive trains with inverse simulation based on bond graph models was successfully applied in Bideaux et al. (2005).

Largely unrecognized by the scientific community, inverse simulation is commonly being used in system configurators of drive and control companies such as the Siemens SIZER Configuration Tool (Ambros 2011). This specialized software calculates dynamic shaft forces in an inverse fashion from predefined typical motion scenarios. It thus helps to select and size an electrical drive solution from available product ranges.

Design is driven by performance requirements, which are often dynamic in nature. Therefore it makes sense to use dynamic simulation tools to prove the fulfillment of those requirements at an early stage of the design. Besides maximum loads and actuator limitations, energy consumption is of interest. Of course, conventional forward simulation can be used to determine those quantities, but in some cases the controller which is necessary to run the simulation is complex to design and maybe considered proprietary knowledge which cannot be made public. Such a case is reported by Bals et al. (2003), where the power consumption of hydraulic, electrical, mechanical, and pneumatic systems for an aircraft flap control is analyzed for predefined load profiles using inverse simulations in Modelica. The benefit of inverse simulation is that the use of proprietary controllers from partner companies can be avoided. To optimize the efficiency and power consumption of components, their sizing is altered and tested using inverse simulation to select the optimal size.

The scope of this paper is similar as in Bals et al. (2003), to use inverse simulation of Modelica models for component selection and sizing, in particular for hydraulic servo axes, while to avoid the necessity to design a feedback control in each iteration step. Preliminary work has been published by the authors. In Liermann (2012) we showed how the design risk can be reduced by using inverse simulation in an early phase of design and demonstrated that by exercising a case study similar to the one which we use in this paper in Section 4. The difference, however, is that in this paper a modified version of a standard Modelica library OpenHydraulicsParedis (2008, (2014) is used and with that the full potential of the object oriented character of Modelica. In our first paper (Liermann 2012) only elementary equations were implemented and all of them were contained in a single flat model. In Saad and Liermann (2013) we gave an overview of the translation process of Modelica models and explained why for conventional simulation software such as Simulink, inverse simulation is not possible on the basis of a component library, and why it is possible with equation based modeling languages. Signal-diagram based simulators such as Simulink have a set causality direction and lack the essential transformation process of model flattening. Flattening means that the equations of all components of a model diagram are assembled in one set before they are sorted and algebraically manipulated in the causalisation step (Casella 2011). In all previous work very simple and custom made models of hydraulic servo-systems were used because available open source and commercial model libraries for hydraulic systems could not be used for the inverse simulation approach. The contribution of this paper is the report of the issues that arise when the inverse simulation approach is applied on existing models that were not built with this approach in mind. Those issues are not specific to the implementation in Modelica but apply also to other equation based modeling languages.

### 1.2. *Paper outline*

Section 2 presents the basic equations of the example problem of a hydraulic valve cylinder drive. In Section 3, an overview over possible reasons for failure of inverse

simulation of Modelica models is given. Modifications to the OpenHydraulics Modelica library to overcome issues with inverse simulation are explained. In Section 4, the benefits of the inverse simulation for a sizing task are demonstrated with a case study. It clearly shows the versatility of inverse simulation for designing more efficient systems without the need to tune a controller. The paper concludes with a discussion of the results.

## 2. Example system – hydraulic servo drive

Servo-hydraulic linear axes are used in many engineering systems and can be used to control position, velocity, or force. They have particular advantages in applications that require high power-to-weight ratio and system reliability in harsh environmental conditions. Figure 3 depicts a schematic of a hydraulic servo-axis consisting of a double rod cylinder, a 4/3 way directional control valve, a constant pressure supply, a pathway representing internal leakage and a position control loop with controller.

The basic equations used to describe and model the system dynamics can be found in standard textbooks such as Jelali (2003), Merrit (1967), Watton (2009) and Murrenhoff (2008). The equations are briefly summarized here, even though their implementation in the Modelica Library OpenModelica is more complicated.

The valve is characterized by four hydraulic connections (pressure supply, line A and B, and return line) and an input signal $u$. The input signal modulates the position of a valve spool which connects or disconnects the hydraulic ways according to the indicated valve symbol schematics. The equations for volumetric flow into line A and B can be given in their basic form (symmetric, zero lapped valve) as:

$$\begin{aligned} Q_A &= c_V\,\mathrm{sg}(x_V)\,\mathrm{sign}(p_S - p_A)\sqrt{|p_S - p_A|} - c_V\,\mathrm{sg}(-x_V)\,\mathrm{sign}(p_A - p_T)\sqrt{|p_A - p_T|} \\ Q_B &= c_V\,\mathrm{sg}(-x_V)\,\mathrm{sign}(p_S - p_B)\sqrt{|p_S - p_B|} - c_V\,\mathrm{sg}(x_V)\,\mathrm{sign}(p_B - p_T)\sqrt{|p_B - p_T|} \end{aligned} \tag{1}$$

where $x_V$ is the normalized spool position of the valve and $c_V$ the valve flow gain. The partial opening of the fluid path $\mathrm{sg}(x_V)$ is a function of the valve spool position and defined as:

$$\mathrm{sg}(x_V) = \begin{cases} 0, & \text{for } x_V < 0 \\ x_V, & \text{for } x_V \geq 0 \end{cases} \tag{2}$$

The valve spool position $x_V$ is related dynamically to the control signal $u$. The dynamic relationship is characterized by the type of actuation and the interaction of the spool with the fluid flow. It is often approximated by a linear second order differential equation with damping ratio $D_V$ and natural undamped frequency $\omega_V$.

$$\ddot{x}_V + 2D_V\omega_V\dot{x}_V + \omega_V^2 x_V = \omega_V^2 u \tag{3}$$

The cylinder is the second relevant system component and characterized by two variable volume chambers and a piston separating them. The piston motion is determined by the balance of forces acting upon it, generated from pressures, friction and external loads. The chamber pressures are a result of compression of fluid in them. The basic equations are:

$$\ddot{x}_p = \frac{1}{m_t(x_p)}\left[(p_A - \alpha p_B)A_p - F_f(\dot{x}_p) - F_{ext}\right] \tag{4}$$

$$\dot{p}_A = \frac{1}{C_{h,A}}\left[Q_A - A_p\dot{x}_p + Q_{Li}(p_A, p_B)\right] \tag{5}$$

$$\dot{p}_B = \frac{1}{C_{h,B}}\left[Q_B + \alpha A_p\dot{x}_p - Q_{Li}(p_A, p_B)\right] \tag{6}$$

with $m_t$ the total mass of accelerated oil, piston and rods, $C_h$ the hydraulic capacity of the chambers, $\alpha$ the piston area ratio and all other variables explained from Figure 3.

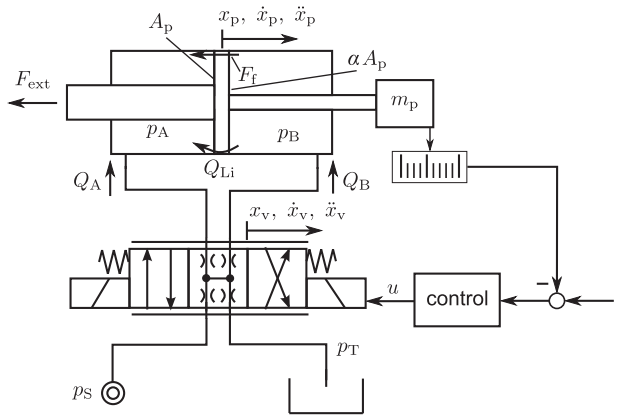



Figure 3. Closed loop controlled hydraulic servo-axis schematic diagram.

## 3. Inversion of OpenHydraulics Modelica library components

This section first illustrates general issues that cause inversion problems with models formulated in equation based modeling languages such as Modelica. In the second part of the section, specific changes of the valve and cylinder models of the OpenHydraulics Modelica library are discussed.

### 3.1. General inversion problems

A simulation model of a hydraulic system consists of many interconnected components that are instantiations of models defined in component libraries. Prior to the execution of a simulation, a compiler flattens and sorts the model equations. Flattening is the process of assembling the equations of the system together in one single set of equations. They are taken out of their separate definition in library component models and put into a single

set of differential algebraic equations. In this process trivial equations are eliminated and the index of the problem is reduced as far as possible by algebraic equation manipulation with the goal to reach a minimal set of equations in block-lower-triangular form Saad and Liermann (2013), Casella (2011). The compiler treats systems intended for inverse simulation in the same way as it treats systems intended for forward simulation. The difference lies only in the definition of boundary conditions, which, however, has profound consequences on the sorting and algebraic manipulation process. Depending on the model implementation and the physical equation itself, errors can occur in the translation process or in the simulation stage. A common error in the translation occurs for example, if a derivative of a variable is needed but not defined. But it is also possible that a model compiles without errors and fails during execution of the simulation. A typical error in the simulation is that state variables become singular which can be caused by discontinuities in the model equations and other forms of non bijective properties. The invertibility of a function is not checked in the translation process. Even if a model contains partially non-invertible functions, the simulation progresses fine as long as the system state does not reach the corresponding conditions. No global property of invertibility is required for the simulation to progress. Table 1 provides a summary of inversion problems that are dealt with in the scope of this work. The list may not be exhaustive, but should be a helpful resource for similar projects.

The first error category in Table 1 refers to errors related to assignment statements that are found in algorithm sections of models. Modelica and other equation based modeling languages define relationships between states and algebraic variables as equations versus assignments. Those equations are transformed and sorted automatically by a compiler. This automatic handling of equations makes the equation based modeling approach unique and is the reason why it can be used for a combined forward and inverse simulation approach. Nevertheless it is sometimes desired to define input-output relationships as assignments, for example to implement a discrete control law. This can be done with algorithm sections. Another use for assignment statements is in functions. Functions have a clearly defined input-output relationship, i.e. causality. They are used in model equations but are defined outside of the models. Figure 4 shows two models that use assignment statements. Both models contain an equation that equates the variable *out* with the simulation time. The model *Test_alg2* defines the variable *inp* implicitly through the function *f* in an equation *out = f(inp)*. This equation can be solved, even though the function *f* is defined using assignment statements. In the model *Test_alg2* the variable *inp* remains

Table 1. Common problems and fixes when translating simulation models for inverse simulation.

| Error category | Example | Comment and Solution |
|---|---|---|
| Algorithms | $m := 2 \cdot n \Rightarrow$ find $n$? | Algorithms can appear in *functions* and *models*. If the assignment $m := 2 \cdot n$ appears in a function, it can be inverted. If a model contains an algorithm section instead of an equation section, it cannot be inverted. **Fix:** Change algorithm to equation, if possible |
| Inv. dynamics unstable | $G(s) = \frac{s-1}{s(s+5)}$ | Inverse dynamics are unstable if the non-inverted system is minimum phase. **Fix**: approximate inverse plant model. |
| Discontinuity | $f(x) = \begin{cases} -x-2 & \text{if } x<0, \\ x+2 & \text{if } x \geq 0 \end{cases}$ | In case the point of discontinuity, is not reached, the inverse simulation runs properly. **Fix:** Define continuous and smooth approximation equation to join both cases |
| Saturation | $f(x) = \begin{cases} -x-2 & \text{if } x<0, \\ x & \text{if } -1 \leq x \leq 1 \\ x+2 & \text{if } x>1 \end{cases}$ | Saturation limits frequently occur in models. 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 multiple solutions. **Fix:** Eliminate saturation limits or avoid running into them. Approximation of saturation by line with small slope. |
| Local min and max | $g(x) = x^2$ | Non-monotonous functions with local minima or maxima have inverse simulation problems. **Fix:** Avoid local maxima/minima regions in the simulation |
| Derivative not found | This error can occur in the following cases: (a) $2^{nd}$ or higher order derivatives of interpolation tables needed (b) Attempt to get derivative of discontinuous functions (c) Inputs are not differentiable | Many causes exist for the error message that a derivative function could not be found. Inputs have to be defined smooth or filtered with pre-filters to a satisfying degree. Interpolation tables, even if they are defined smooth, can only be differentiated once. Input filters may cause discontinuities due to their intitial state definitions. **Fix:** Replace problematic functions and tables with differentiable functions. |

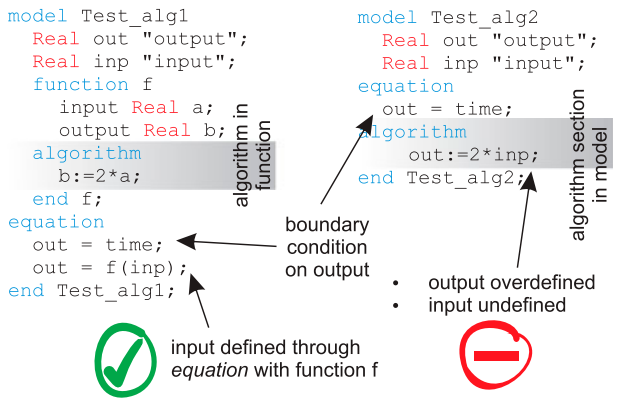



Figure 4. Two models using assignment statements. The left model contains an assignment as part of a function definition. The right hand side model contains the assignments in an algorithm section of the model. The latter model fails during execution because the variable *out* is overdefined.

undefined. The assignment statement *out*:=2*inp* is in an algorithm section and the compiler will not attempt to solve this as an equation. In effect, the variable *out* is over-determined because it is already constrained by the equation *out* = *time*. The simulation fails. One might argue that if it is possible to invert an algorithm that is part of a function it must also be possible to invert an algorithm that is part of an algorithm section in a model. The difference is that the function is evoked as part of an equation. Therefore the variable *out* is not over-determined because the equation can be solved for the variable *inp*. In the algorithm section the causality direction is clearly defined and it would go against the intention of the modeler to invert it.

The second error category refers to systems, where the inverse dynamics is unstable, i.e. for non-minimum phase systems. The problem can be explained at the example of linear non-minimum phase systems, which are characterized by zeros or poles with positive real part. The inverse of a linear system is easily obtained as the inverse of a transfer function. If the transfer function of the non-inverted system possesses a zero with positive real part, then the inverse is a system with an unstable pole. The inverse simulation will not produce useful results in that case. It may be difficult to prove for a non-linear system that it is minimum phase. A good way to check this is by linearizing the system around a number of operating points or to perform many simulations. The proposed fix to this problem is to approximate the non-minimum phase system by a minimum phase system or to choose another input for which the system is not minimum phase. This error was not encountered during this research. For more explanation on this case we refer to Bals et al. (2003)

The cases ‘discontinuity’, ‘saturation’, and ‘local min and max’ in Table 1 describe different forms of non-bijective functions. The simulation will fail when a variable progresses into one of these cases and it is required to invert the non-bijective relationship. The suggested solution to avoid these situations, or to redefine the equations to make them bijective in the required region, may seem unsatisfactory. It is a common case that a motion reference trajectory is provided in such a form that it can actually not be reproduced. It is also part of the design of many devices to have backlash, and hysteresis and static friction. The question therefore arises, how the inverse simulation can be meaningful if all those characteristics need to be removed or avoided? The answer depends on what the scope of use of the inverse simulation is. If the scope is to size system components so that a required motion trajectory can actually be fulfilled, then the failing of the simulation because of actuator saturations provides exactly the information that was sought after. The designer now has the choice to reduce motion requirements in order to complete the simulation or to choose another size for one of the system components. Another benefit that we highlight in this paper about using inverse simulation is that it can be used to study the energy consumption of a drive system for given load cycles. Eliminating effects like backlash, and hysteresis affects the controllability but not the energy consumption. As we show in our design example, friction terms do not have to be eliminated. They just have to be modified and expressed as bijective equations. We conclude that the inverse simulation makes sense for the purpose of system sizing and to predict the energy consumption of given load cycles.

Modelica is used by more and more researchers for the development and analysis of dynamic multi-physics systems. Its strengths are its formal, equation based semantics and its object-oriented character. Especially the availability of ready-to-use simulation libraries in the Modelica Standard Library and further library contributions from the scientific community makes the use of Modelica very appealing. The OpenHydraulics library (Paredis 2008, 2014) is an open-source project which is seamlessly integrated with the open-source Modelica Standard Library. However, inverse simulation of a hydraulic servo-axis according to Figure 1 is not possible using OpenHydraulics. The causes for this are difficult to identify because often Modelica models are strongly hierarchical and modular and the error messages are not easy to interpret. In the remainder of this section the specific changes in the valve and cylinder model are discussed for the inverse simulation to work.

### 3.2. *Valve model analysis and modifications*

The valve is a major component in a hydraulic servo-axis. The basic valve flow equations are given in Equation (1). The flows are a function of port pressures $p_{A,B,P,T}$ and the valve spool position $x_V$, which determines the opening of the flow paths. The function **sg** is only invertible for positive arguments, but Equation (1) as a whole can be solved for the spool position without restriction.

In the openHydraulics library, the directional control valve *V4_3CC* is considered and its components are shown in Figure 5. It can be seen that the model is build strongly hierarchical. Fourteen submodels are visible in the top hierarchy, some of which contain only trivial equations, such as the interface ports portP,portA,portB,-portT, and control. The flow equations of Equation (1) are represented in the submodels P2A,A2T,P2B, and B2T, which are instantiations of a basic component called VariableRestrictionValve. The ports of the submodels are connected to the main valve ports through lines with small volumes j1,j2,j3, and j4, which are instantiations of a component called NJunction. Finally, a signal block called dynamicResponse connects the control input with the input of the variable restriction blocks. Upon its translation, the valve model consists of a total of 185 variables and 185 equations nested throughout all its submodels.

When performing an inverse simulation of a hydraulic servo drive as in Figure 1 it is not always easy to identify causes of translation problems due to the nested structure of the models. In the inverse simulation, the cylinder position trajectory and its derivatives are given and the simulation computes the valve control signal. The error message that the compiler of Modelica outputs with respect to the valve, is that the second order derivative of *openFraction*, the partial opening of the restrictions cannot be found. Analysis shows that the restriction model VariableRestrictionValve uses a look-up table to map between the dynamic spool position $x_V$ and the valve opening variable *openFraction*. This look-up table is a convenient implementation of the function $openFraction = \mathrm{sg}(x_V)$ in Equation (1). It is an instantiation of the CombiTable1Ds of the Modelica standard library.

$$openFraction = \mathrm{interpolate}(\mathrm{openFractionData_i}, x_{\mathrm{VData.i}}) \tag{7}$$

With just a few data points the look-up table can be used to characterize positive and negative overlap between the spool and sleeve of the valve, or nonlinear opening characteristics. However, even though defined smooth, the interpolation function CombiTable1Ds defines only first order derivatives. Therefore, since the second derivative is needed, the error occurs in the compiling stage of the servo-axis model.

It is easy to see why the second order derivative of the valve opening is needed. The valve control signal $u$ is an output to the simulation in the inverse simulation case. Since there is a second order dynamic relation between and valve control signal $u$ and spool position $x_V$, the second order derivative of $x_V$ is needed. The second order derivative of $x_V$ and of *openFraction* are directly related but the interpolation function in Equation (7) is only defined for first order derivatives.

In order to solve this problem, the interpolation look-up table block is removed and replaced by an equation.

$$openFraction = \begin{cases} -1 & \text{if} x_V < x_{V\min}, \\ \frac{x_V}{x_{V\max}} & \text{if} x_{V\min} \le x_V \le x_{V\max} \\ 1 & \text{if} x_{V\max} < x_V, \end{cases} \tag{8}$$

The equation allows second order derivatives and is invertible within the extreme valve positions. For the variable restriction P2A between pump and actuator port A, the limiting parameters are $x_{V\min} = 0$ and $x_{V\max} = 1$. For the variable restriction P2B between pump and actuator port B, the limiting parameters are $x_{V\min} = -1$ and $x_{V\max} = 0$. Similarly for the other restrictions A2T and B2T.

A warning message can be included in the valve model to inform the user when the valve exceeds its opening limitations. This is illustrated by the when clause:

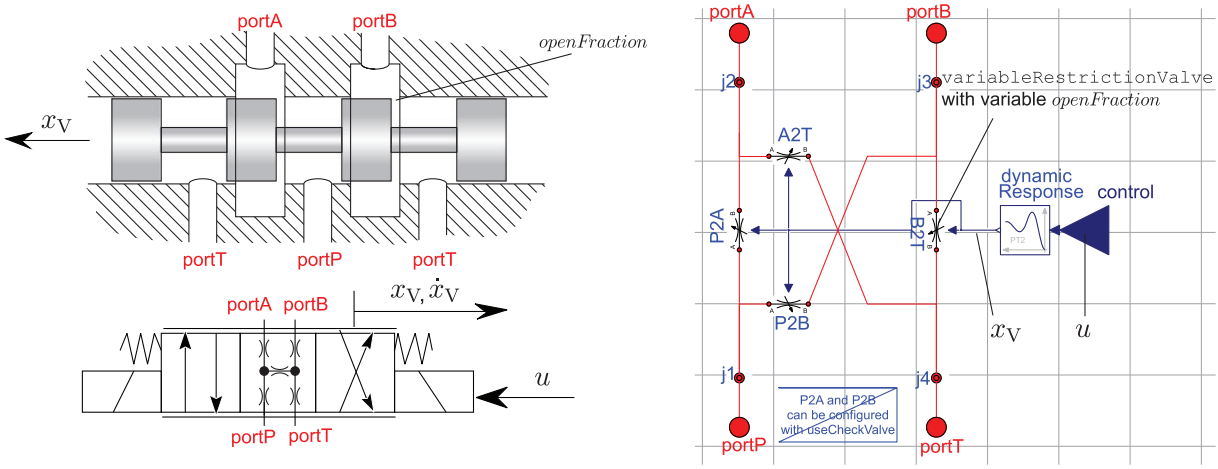



Figure 5. Valve model. Top left shows spool sleeve operating principle of valve. Bottom left shows valve symbol. Right shows the graphical representation of the Modelica model V4_3CC. Component and variable names used in graphical representation of Modelica model (right side) are explained through graphic on the left side.

```
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 *V*4_3*CC*_*mod* works for inverse simulations.

### 3.3. Cylinder model analysis and modifications

The cylinder is the second essential hydraulic component for a hydraulic servo-axis. It transforms the flow and pressure provided from the valve into mechanical force and motion. Figure 6 illustrates the cylinder that is modeled in the OpenHydraulics library.

It is a double acting cylinder with a single rod consisting of two chambers, the rod and head chamber. Between the mechanical interfaces of the chambers is a simple mass to describe the inertia of the piston. A damper model describes the viscous damping behaviour of the piston in the cylinder. Geometric constraints are placed between the mechanical connectors to parameterize the size of the cylinder, piston and rod. Variable restriction models are used to model the external and internal leakage. Two cushion models with hydraulic damping are connected in parallel to the mechanical connectors of the chambers.

Several errors occurred in the translation of the model for inverse simulation which are related to two root causes. The first cause is related to an algorithm section employed in the LaminarRestriction model. The second cause is related to the end position cushioning models.

The flow through a laminar restriction is simply linear with respect to the pressure difference across it.

$$\dot{m}_{\text{lam}} = C_{\text{lam}} \Delta p \tag{9}$$

The conductance $C_{\text{lam}}$ of a annular leakage passage is a function of the diameter $D$ of the passage, its length $L$, the fluid density $\rho$ and the viscosity $\eta$ according to the Poisson's equation

$$C_{\text{lam}} = \frac{\pi D^4 \rho}{128 \eta L} \tag{10}$$

For a certain reason, the flow equation Equation (9) has been coded as an assignment (':=') in an algorithm section of the model LaminarRestriction.

```
algorithm
port_a.m_flow := conductance*dp;
```

An assignment in an algorithm section of a model cannot be interpreted as an equation, compare Table 1 and explanation in Section 3.1. In this case the resolve for the error is simply declare the flow relationship by an equation as follows:

```
equation
port_a.m_flow = conductance*dp;
```

With this replacement of a causal assignment by an *a*causal equation, the modified laminar restriction model works properly in inverse simulations.

The second error is caused by the end position cushioning. The cushion model is a complex system containing several submodels. It includes three look-up tables.

The working principle of the end position cushioning is that the outflow $\dot{m}_{\text{cushion.out}}$ from a chamber is restricted as the piston approaches its end position.

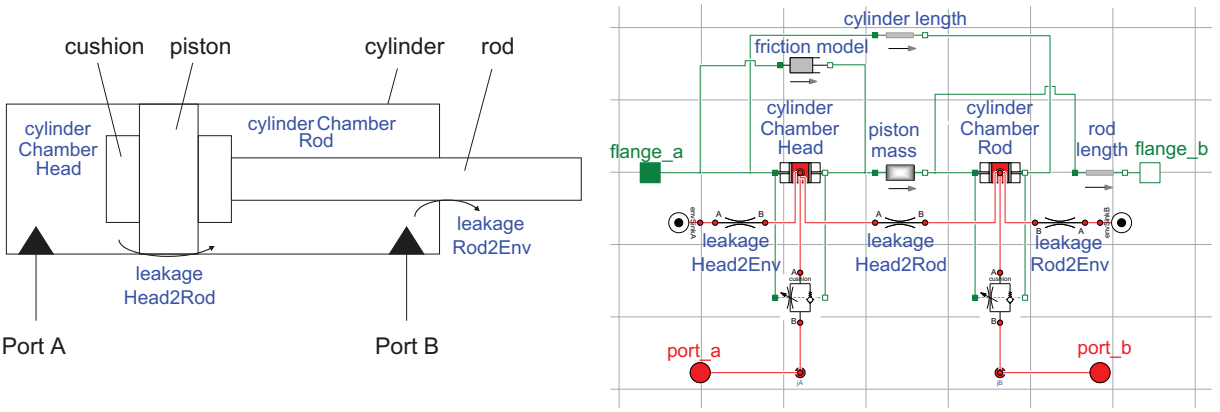



Figure 6. Cylinder model schematic diagram.

$$\dot{m}_{\text{cushion.out}} = f(\Delta p, interp(x_{\text{P.i}}, \text{flowFractionCushion}_{\text{i}})) \quad (11)$$

As previously the case in the valve model (Section 3.2), the look-up tables serve as a convenient and flexible way to parameterize the opening fraction of the variable restriction. However, the problem with the look-up table is that the relationship between input and output is only once differentiable with respect to time. Therefore, higher order derivatives of the flow cannot be calculated.

The solution in this case is to just remove the end position cushioning from the cylinder model with the following reasoning: The use of end position cushioning is recommended in forward simulation even if in reality the cylinder does not have cushioning. Without cushioning, high frequency modes are excited during an undamped impact between the piston and the cylinder head which considerably slows down simulation time. Using inverse simulation for sizing purposes, it is not required to continue a simulation once cylinder end stops have been reached because this immediately indicates insufficient sizing or wrong choice of motion trajectory. By defining an appropriate motion trajectory, it is easy to avoid that the cylinder runs into end-stops. Without cushion models, the modified double acting cylinder runs with no further complications in the inverse simulation case.

## 4. Case study: improved sizing of servo-axis

This section illustrates the versatility of inverse simulation for system design and component sizing. The test application is a servo-hydraulic linear drive of a gear shaping machine. The results are presented in the form of a case study. The aim is to show how inverse simulation together with design intuition helps in finding design alterations that meet motion requirements while energy efficiency is improved. The inverse simulation makes use of the modified OpenHydraulics library components described in this paper.

### 4.1. Optimization parameters and methodology

Sizes and loads for this example are taken from a tooling machine servo-axis, namely a gear shaping machine. Figure 7 shows the gear shaping spindle actuator and a typical load cycle. The gear shaping machine produces internal and external gears with a reciprocating toothed disk cutter. The cutter rotates with the workpiece during cutting and gradually feeds into it. The complete work-cycle of a reciprocating cutting process consists of acceleration, cutting, deceleration and return stroke. During acceleration phase the spindle is brought to cutting speed (0.23 m/s). The cutting force (10 kN) acts as external load on the spindle upon contact between cutter and workpiece. The cutting loads are proportional to the length of the engaged cutting edge and the feed depth. A work cycle shown in the right of Figure 7 represents a heavy duty work cycle for this type of machine. After emergence from workpiece the spindle is decelerated to reverse motion with return stroke velocity (0.8 m/s).

Figure 8 shows the sizing parameters of the drive under consideration for improvement and the design constraints. In this test case, the cylinder and upper rod diameters should be modified such that the velocity and force profile can be fulfilled for a supply pressure of 100 bar while minimizing the energy losses. The lower rod diameter is fixed at 80 mm for considerations of structural integrity. The valve motion control requires a certain pressure drop across the valve to maintain controllability. This is because reducing the pressure difference across the valve also reduces the flow gain Murrenhoff (2008). An empirical rule of thumb is that it should not decrease below 15 % of the pressure supply. The larger the margin, the higher the controllability.

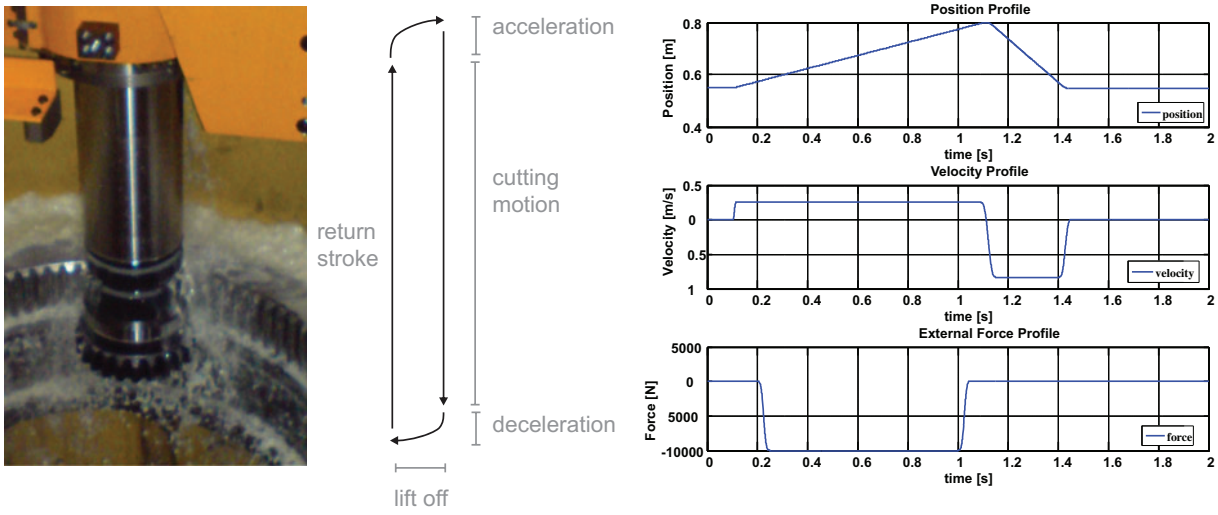



Figure 7. (a) Gear shaping spindle with gear shaped cutter cutting internal gear. (b) Typical load profile during rough cutting.

Also, in the return line a certain pressure drop should be maintained. In addition to keeping controllability at a sufficient level, pressure in the return line helps to avoid cavitation. The simulation model of the setup is shown in Figure 1. It is assumed that pressure is supplied from a constant pressure source.

Even though the word optimization is used in this context, it is not meant to refer to a mathematical algorithm that finds the maximum according to a specific design criteria. Rather a methodology of improvement is proposed that follows engineering intuition. The methodology of optimization that would intuitively be adopted by a designer for this case study is depicted in Figure 9. It is clear that the nominal size of the valve has no effect on the system losses. The valve opening is completely determined by the control and corresponds directly to the motion and force trajectories. Therefore, in a first step for the simulation, the nominal valve size is usually oversized. The cylinder size determines the amount of flow consumed from the constant pressure supply and therefore has a large impact on the energy consumption. Through a couple of design iterations, the cylinder sizes can be changed such that the pressure drop constraints are fulfilled while the energy consumption is minimized. The inverse simulation approach supports the iterative design procedure, because the controller of the drive does not need to be adapted for each sizing iteration. After choosing the appropriate cylinder dimensions, an optimal valve size can be chosen. The nominal valve size should not be too large, as large valves may be more expensive, have larger leakage and lower bandwidth than small valves. Also, generally it is important to make good use of the valve control resolution in middle position of the valve spool.

### 4.2. *Results of Sizing Methodology with Inverse Simulation*

Two evolution cycles for the parameter sets are depicted on the right side of Figure 9. The initial size is taken from a commercially available gear shaping machine. The goal is to find a parameter set that minimizes the energy consumption per working cycle and that is at the same time capable to fulfil the motion requirements. The inverse simulation is applied starting with the initial configuration. The pressure supply is 100 bar. The pressure distribution, the energy consumption, and the valve opening are shown in Figure 10. Sometimes, instead of the energy consumption, the instantaneous expended power is shown. The energy consumption that is shown in the middle graph is the integral of the instantaneous expended power. The goal of the parameter variation is to minimize the total energy consumption at the end of the motion cycle.

In the top diagram it can be seen that at the start of the motion (t = 0.1 s) both chamber pressures rise from their (default) initial conditions to a level around half of the supply pressure. A small difference in pressures is needed to accelerate the piston and tool and to overcome friction. But this pressure difference is small and can hardly be noticed. The pressure drop across the valve in the feed line ($p_S - p_A$) is equal to the pressure drop in the return line ($p_B - p_T$). This is because the cylinder is a double rod type with equal piston areas ($\alpha = 1$) and the same mass flow rate that enters chamber A also leaves chamber B. Since the valve opening is equal for feed and return line, the pressure drops are the same. At t = 0.2 s the tool engages and the cutting load is applied. The actuator pressure difference, often called load pressure $p_A - p_B$, is noticeable, but it is only 37% of the total pressure drop. The load force during cutting is only a fraction of what the cylinder could potentially deliver if the load pressure would be higher. Load pressure and cutting force are related through the piston surface area. Larger piston areas lead to higher flow requirement, and therefore high energy consumption, which is seen from the middle plot. The energy consumption is linear with consumed flow because the supply pressure is constant. For the original cylinder size the total energy consumption amounts to 14.1 kJ for one work cycle. From this it is clear that to reduce energy losses, the piston surface area should be decreased. Of the total consumed energy,

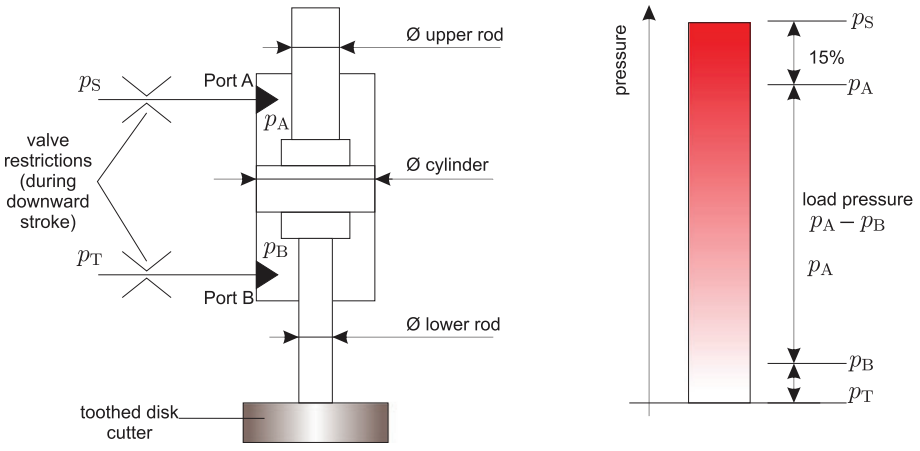
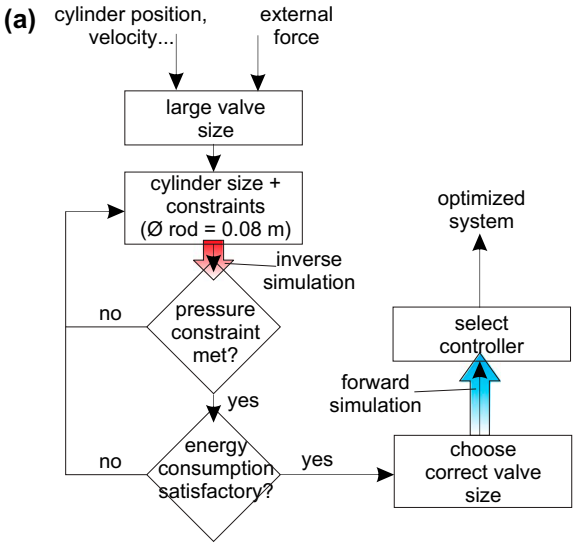



Figure 8. Optimization parameters and load pressure design requirement.




(b)

| | Ø Cylinder | Ø lower rod (constant) | Ø upper rod | supply pressure |
|---|---|---|---|---|
| original | 100 mm | 80 mm | 80 mm | 100 bar |
| 1st Evolution | 91 mm | 80 mm | 80mm | 100 bar |
| 2nd Evolution | 82 mm | 80mm | 77 mm | 200 bar |

Figure 9. (a) Optimization strategy (b) Evolution cycle of parameter sets.

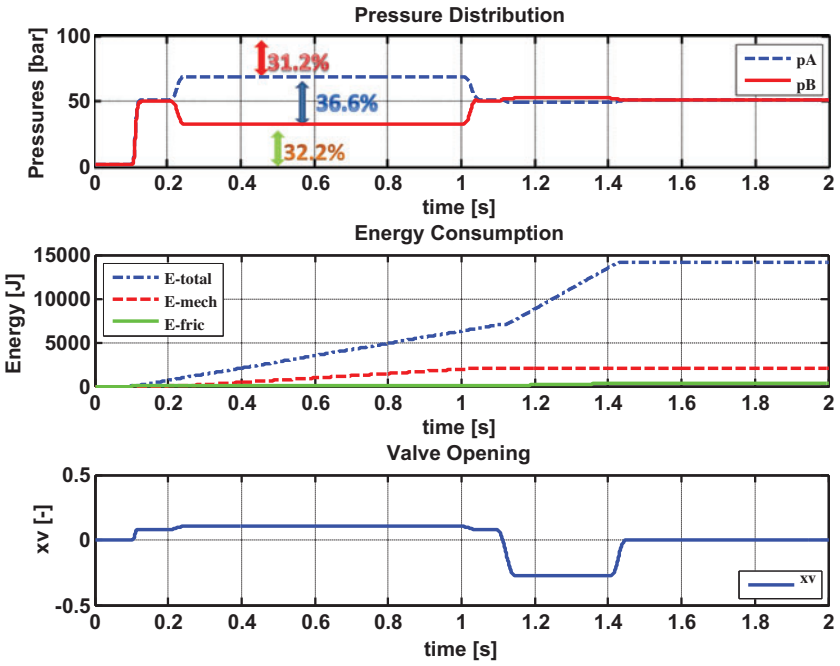



Figure 10. Simulation result with initial design: actuator pressures, energy consumption, valve opening.

most is spent on throttling losses, while only 2 kJ is brought into the cutting process. Friction losses turn out to be almost negligible. From the bottom plot it is seen that the valve opens only 10 % in the working stroke and 27.5 % in the return stroke. The valve is oversized for these system requirements; however, as stated earlier, an appropriate valve will be chosen after the cylinder parameters are selected.

In order to reduce the energy consumption, with only a few iterations, parameter set two in Figure 9(b) is reached, where the cylinder bore is reduced by 9 mm. Simulation results are presented in Figure 11. The top plot shows that the pressure margin is reduced now to the desired level of around 15% and that the load pressure difference now reaches up to 70%. The smaller piston surfaces lead to reduced flow consumption and therefore reduced throttling losses. The total energy consumed is reduced to 7.4 kJ which is approximately half of that consumed by the initial configuration. The valve opens less than for the initial simulation because of the reduced flow demand.

The inverse simulation allows the engineer to focus on the sizing task without getting distracted into questions of control design. If, for example an alternative

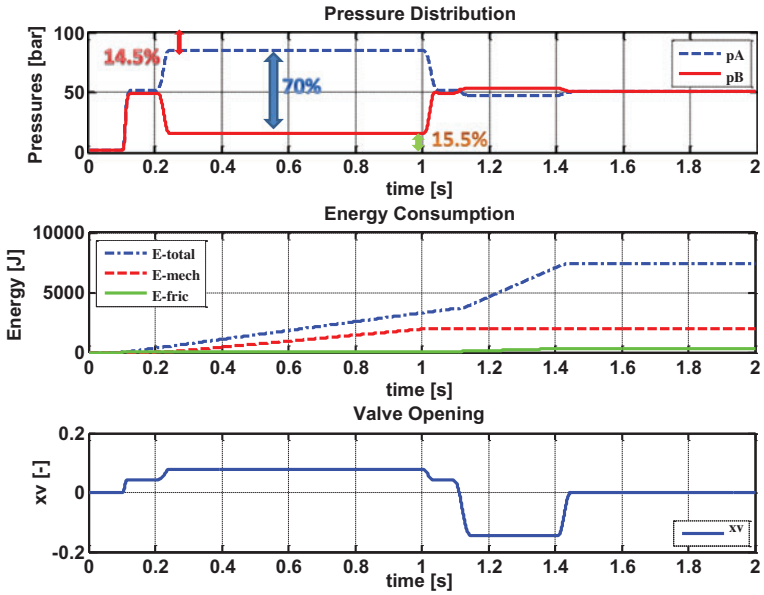



Figure 11. Simulation result for first design iteration: actuator pressures, energy consumption, valve opening.

parameter set is to be searched for 200 bar pressure supply at the same level of efficiency, this can be done intuitively without any further considerations about control parameters. The second design iteration in Figure 9(b) is reached quickly in a few steps. Besides changing the cylinder bore diameter to 82 mm, also the upper rod diameter is reduced by 3 mm to 77mm. Synchronizing double rod cylinders, where both rods have equal diameters, behave symmetric. Prediction of their behavior is comparatively easy. Differential cylinders, where rod diameters are unequal have non-intuitive pressure curves, as can be seen from the results of the inverse simulation shown in Figure 12. The supply pressure is now increased to 200 bar. The upper chamber pressure reaches 168.8 bar, leaving a desired pressure margin of 15.6%. In the bottom chamber, the pressure drops to only 6 bar which could be considered a little too low to prevent cavitation. The load pressure $p_A - p_B$ now amounts to 81.2% of the total provided pressure. The total energy consumption of the system for this parameter set is 4.4 kJ which is a further reduction of 59.1% compared to the energy consumed by the first design iteration. The potential for energy saving lies in increasing the load pressure during the return stroke, while keeping the load pressure at the desired level during the working stroke. This required to have unequal rod diameters.

With this example it has been shown, how, without the need to parameterize a controller, a huge improvement in the efficiency could be obtained through appropriate sizing of a hydraulic cylinder, while in the process being able to make sure that motion requirements are fulfilled. The procedure highlights the power of the inverse simulation. According to Figure 8 two more steps in the optimization process remain. First, an appropriate valve has to be chosen. It is found that a valve with a nominal flow of 30 l/min at 35 bar nominal pressure is large enough to realize the required motion, with maximum opening during return stroke of 80%. Finally, a controller should be designed to verify that the system performance can be achieved in closed loop.

### 4.3. *Control design with forward simulation*

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. The switch from inverse to forward simulation is shown in Figure 1. The input to the closed loop system now is no longer the position trajectory as constraint on the piston motion directly. It is now the reference position from which an error signal is calculated and given as input to a controller. The controller produces the control signal for the valve. As a first attempt, proportional control is employed. Tuning is straight forward since only one parameter is varied, and it is found that a value of 300 $\frac{100\%}{\mathrm{m}}$ is a good compromise between system damping and load stiffness. The control response for the final system configuration with differential cylinder and 200 bar pressure supply is shown in Figure 13.

The top plot shows the reference position together with the actual position for the whole duty cycle. The position error is shown in the middle plot. The bottom plot shows the valve opening fraction. After an initial position offset of 5 mm, the piston quickly moves to its initial position at 0.55 mm. Then the duty cycle begins, first with acceleration of the piston followed by a period of constant velocity, while the cutting load applies. The deviation from the desired trajectory during the working

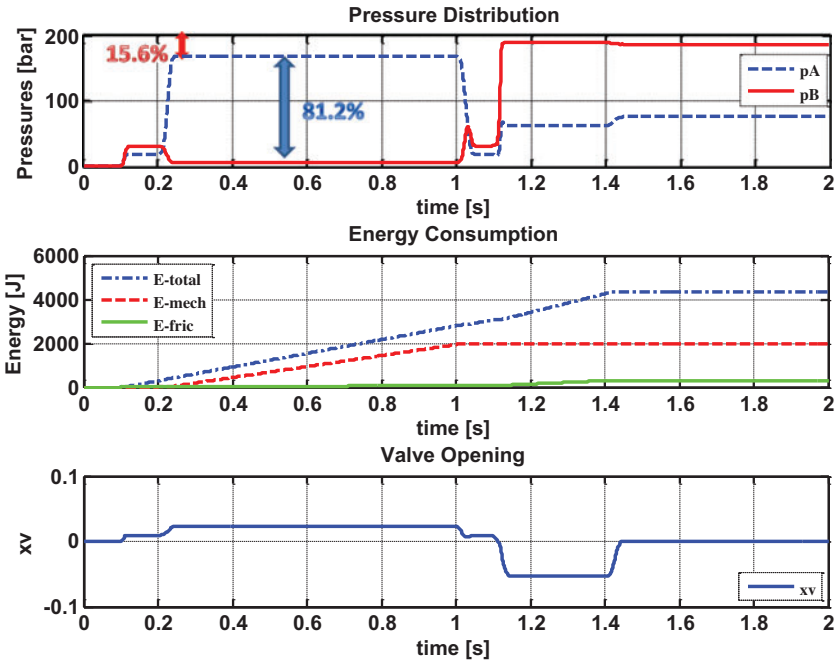



Figure 12. Simulation result for second design iteration: actuator pressures, energy consumption, valve opening.

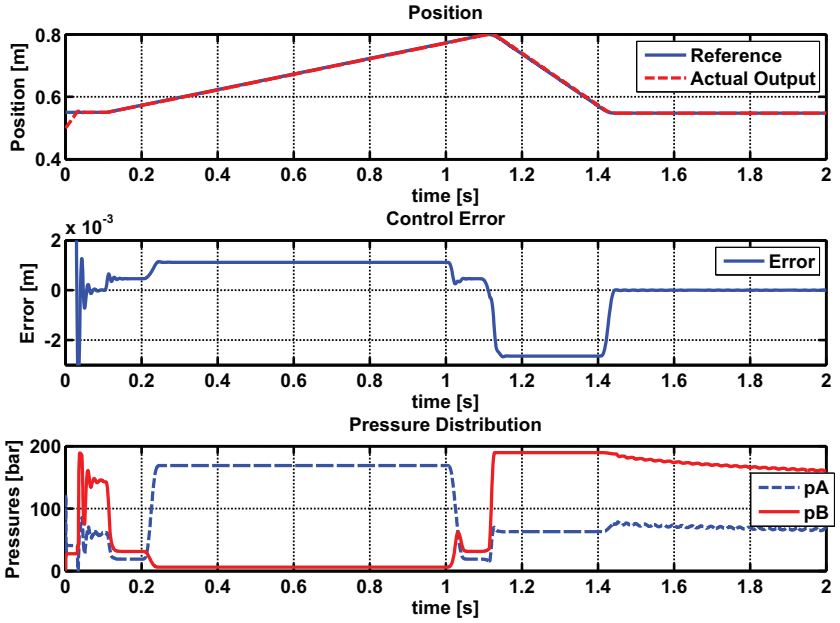



Figure 13. Simulation result with forward simulation of optimized system.

stroke is 1.1 mm. The error in vertical position during reciprocation of the tool translates into a small error of the gear surface shape. The magnitude of 1.1 mm is not unusual during rough cutting. To achieve a higher accuracy during fine cutting, the feed rate would need to be reduced. At 1.1 s the return stroke begins. The control precision is reduced during the return stroke because of the high speed, even though no external load is applied. More sophisticated control concepts can be applied to reduce the positioning error during the working stroke, but this is not the scope of this research.

This section illustrated the benefits of the inverse simulation in selecting system parameters through modifying the design of a gear shaping machine hydraulic servo axis. It also tested the feasibility of inverse simulation with the modified openHydraulic library which is the result of the work of this paper. Using the inverse simulation approach, it is possible to postpone the task

to design a controller to a later stage of design. The dynamic simulation can nevertheless be used for system sizing and gives valuable insight to the design engineer. The process saves time, reduces the design risk and increases potential for communication and collaboration between engineers of different departments responsible for sizing and control design.

## 5. Discussion of results

The aim of this paper is to show usefulness of the inverse simulation approach with equation based simulation models for the design of hydraulic drives. In particular we examine the versatility standard Modelica libraries as a tool for engineers to intuitively improve drive component sizing. It has been previously shown how simplified Modelica models have a potential to solve sizing problems with inverse simulation Bals et al. (2003); Liermann (2012). However, more complex models such as those found in Modelica standard libraries often have features that prohibit the use of inverse simulation. This paper gave an overview over typical issues that may be faced. An example was demonstrated with the open source Modelica library OpenHydraulics how to render two of its components suitable for inverse simulation.

As part of the discussion it should be mentioned that the main obstacle for making models fit for inverse simulation, is to be able to interpret error messages produced by the Modelica compilers. Modelica models that are built hierarchical tend to be very nested and the number of equations accumulates quickly. An example is the valve and the cylinder considered in the case study. The valve flow, which can be modeled basically by Equations (1)–(3) consists of 185 variables in its implementation in OpenHydraulics. The cylinder model, represented in simplified form in Equation (4-6), consists of 300 variables in the OpenHydraulics implementation. This is not a problem in terms of computational efficiency, but it makes troubleshooting a model difficult, even when the physics of the example is well understood by the user. The changes that were ultimately made, seem trivial compared to the complexity of the error messages.

In this case study we considered a system that theoretically is input-output invertible, (Saad and Liermann 2013). In other cases, this may not be known a priori. The interesting thing is, that despite the fact that the system is not fully state observable, the inverse simulation works flawlessly. Not being state observable means that the motion and force trajectories do not correspond to a unique combination of cylinder pressures but only to a unique load pressure difference. However, it turns out that it is not important to define proper values for the initial pressures in the cylinder for the simulation to give sensible results. This can be seen by comparing Figures 12 and 13. Even though the initial pressures in forward and backward simulation are different, because there is an initial motion that happens in the forward simulation only, for the rest of the duty cycle, the exact same pressure levels can be observed.

The open source modeling language Modelica allows to build models easily due to the physics-based interfaces of components and modular structure. Components are preset with parameters that are common for the field of application. A simulation returns sensible results even without the necessity of having to scrutinize every single parameter such as ambient temperature, friction, or leakage coefficients. It is a great practical advantage that with the use of inverse simulation, parameterization of a controller is rendered unnecessary. This can open the use of dynamic simulation to a much wider scope of potential users. Admittedly, the case study presented in this paper was not very complex. Only two parameters, the cylinder bore and the upper piston rod diameter were considered for alteration. Optimization of two parameters is easily done with a trial and error approach. The jump in complexity, however, is large from a two parameter to a three parameter optimization, should the control gain have been included in the consideration. The design constraint involved two features - the pressure margin between actuator and supply pressure and between actuator and tank pressure. The design objective was simply the minimization of power losses. To formulate a proper optimization routine for this seemingly simple task would have been quite complex. In this area, where a formal optimization algorithm may increase the problem's complexity rather than simplify it, and where the conventional forward simulation approach is yet very demanding, the inverse simulation seems to be ideal. Drive applications may represent this category very well, as the example of the SIZER tool underlines (Ambros 2011).

## 6. Conclusion

We conclude with this study that it is feasible and of high practical value to use dynamic simulation models based on standard Modelica libraries to assist design engineers in the component sizing problem with inverse simulation. The study points out typical problems that arise and lists possible solutions or workarounds. To be able to study a hydraulic drive, we explained which necessary changes had to be made to an open source library OpenHydraulics for it to work with the inverse simulation approach. We explained how the inverse simulation approach assists an intuitive engineering methodology to optimize systems through iterations and demonstrated how with this methodology the modified library could be used to successfully optimize the energy consumption of a gear shaping spindle actuator.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

## Funding

We gratefully acknowledge the funding by the American University of Beirut, University Research Board for its support to conduct this research [102544-21071].

## ORCID

*Joseph Saad* http://orcid.org/0000-0002-0030-1351
*Matthias Liermann* http://orcid.org/0000-0002-0165-0078

## References

Ambros, M., 2011. Engineering tool SIZER. *In*: *Antriebstag 2011*. Fürth: Siemens Kundenveranstaltung.

Bals, J., et al., 2003. Object-Oriented Inverse Modelling of Multi-Domain Aircraft Equipment Systems with Modelica. *In*: *Proceedings of the 3 International Modelica Conference*, November 3-4, 2003 Linköping, 377–384.

Bideaux, E., et al., 2005. Design of a hybrid vehicle powertrain using an inverse methodology. *In*: C.R. Burrows, K.A. Edge and D.N. Johnson, eds. *Power transmission and motion control 2005*. Fürth: John Whiley & Sons Inc, 317–332.

Buchholz, J.J. and von Grünhagen, W., 2007. *Inversion Impossible?* Munich: GRIN Publishing GmbH.

Carpanzano, E. and Maffezzoni, C., 1998. Symbolic manipulation techniques for model simplification in object-oriented modelling of large scale continuous systems. *Mathematics and computers in simulation*, 48, 133–150.

Casella, F., 2011. *Symbolic manipulation for the simulation of object-oriented models*, last downloaded Jan 2015.

Cellier, F. and Elmqvist, H., 1993. Automated formula manipulation supports object-oriented continuous-system modeling. *Control Systems, IEEE*, 13, 28–38.

Clayton, G., *et al.*, 2008. Inverse-feedforward of charge-controlled piezopositioners, Mechatronics 18273–281, special Section on Optimized System Performances Through Balanced Control Strategies The 4th IFAC Symposium on Mechatronic Systems G Mechatronics 2006

Devasia, S., 2002. Should model-based inverse inputs be used as feedforward under plant uncertainty? *Automatic Control, IEEE Transactions on*, 47, 1865–1871.

Feki, M.E., et al., 2008. Structural properties of inverse models represented by bond graph. *In*: *Proceedings of the 17th world congress, the international federation of automatic control*, July 6-11, 2008 Seoul, Korea: 236–241.

Fritzson, P., 2014. *Principles of object-oriented modeling and simulation with modelica 3.3: A cyber-physical approach*. Piscataway: Wiley-IEEE Press.

Gräber, M., 2014. Exploiting actuator limits with feedforward control based on inverse models. *In*: *Proceedings of the 10th international modelica conference*, March 10-12, 2014, Lund, Sweden: 637–645.

Hess, R.A., Gao, C. and Wang, S.H., 1991. Generalized technique for inverse simulation applied to aircraft maneuvers. *Journal for guidance, control, and dynamics*, 14, 920–926.

Jelali, M., 2003. *Hydraulic servo systems: Modelling, idendification and control*, London: Springer.

Liermann, M., 2012. Backward simulation - A tool for designing more efficient mechatronic systems. *In*: *Proceedings of the 9th International MODELICA Conference*, September 3-5, 2012, Munich, Germany, 867–876.

Liermann, M. and Murrenhoff, H., 2005. Knowledge based tools for the design of servo-hydraulic closed loop control, *In*: *International symposium on power transmission and motion control (PTMC)*, Bath, England, 17–28.

Luca, A.D. and Book, W., 2008. *Springer handbook of robotics*, chapter 13. Berlin-Heilderberg: Springer, 287–317.

Markel, T., et al., 2002. Advisor: A systems analysis tool for advanced vehicle modeling. *Journal of power sources*, 110, 255–266.

Merrit, H.E., 1967. *Hydraulic control systems*, New York, NY: John Whiley & Sons Inc.

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

Murray-Smith, D.J., 2011. Feedback methods for inverse simulation of dynamic models for engineering systems applications. *Mathematical and Computer Modelling of Dynamical Systems*, 17, 515-541.

Murray-Smith, D.J., 2014. Inverse simulation and analysis of underwater vehicle dynamics using feedback principles. *Mathematical and computer modelling of dynamical systems*, 20, 45-56.

Murrenhoff, H., 2008. *Servohydraulik - geregelte hydraulische Antriebe [Servo-hydraulics - closed loop controlled hydraulic drives]*. Aachen: Shaker Verlag.

Paredis, C.J.J., 2008. An open-source modelica library of fluid power models, in Fluid Power and Motion. *Control*, 2008, 77–90.

Paredis, C.J.J., 2014. *Openhydraulics*,

Saad, J. and Liermann, M., 2013. Inverse dynamic simulation of a hydraulic drive with modelica. *In*: *Proceedings of the ASME 2013 international mechanical engineering congress & exposition*, San Diego: ASME IMECE: 63310

Schlemmer, K. and Murrenhoff, H., 2007. A concept of a knowledge-based assistance system for servohydraulic drive design. *In*: *Proceedings of IMECE2007 ASME international mechanical engineering congress and exposition*, Seattle, Washington, USA, pp. IMECE2007-41402

Tagawa, Y., Tu, J.Y. and Stoten, D.P., 2011. Inverse dynamics compensation via Gsimulation of feedback control systems. *Proceedings of the institution of mechanical engineers, Part I: Journal of systems and control engineering*, 225, 137–153.

Thümmel, M., et al., 2005. Nonlinear inverse models for control. *In*: *Proceedings of the 4th international modelica conference*, March 7-8, 2005 Hamburg: 267–279.

Watton, J., 2009. *Fundamentals of fluid power control*, New York: Cambridge University Press.

Wipke, K., Cuddy, M. and Burch, S., 1999. Advisor 2.1: a user-friendly advanced powertrain simulation using a combined backward/forward approach, Vehicular Technology. *IEEE Transactions on*, 48 , 1751–1761.