Physical network modeling of a reverse osmosis purification unit

Joel Assarsson

Simon Thoulouis

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6027
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2017 by Joel Assarsson & Simon Thoulouis. All rights reserved.
Printed in Sweden by Tryckeriet i E-huset
Lund 2017

Abstract

The use of computer models in research and development is a powerful tool to make the engineering process more effective. By providing accurate simulations, the need for working prototypes and extensive testing is reduced. This Master's Thesis investigates the Gambro WRO 300 H reverse osmosis water purification system with the aim of creating a working and easily adaptable computer model of the machine. For this purpose, the physical network modeling approach provided by the MathWorks Simscape environment has been used, and the experimental data from the machine is fitted to an a priori model of the reverse osmosis process - the solution-diffusion model. The result is a Simscape model of the WRO 300 H as well as four experimental model designs implemented using the same custom Simscape library. While the models do capture the overall behaviour of the WRO 300 H and provide interesting insights in the experimental designs at a qualitative level, there are significant improvements that could be made to enhance the accuracy of the membrane model by modeling its internal physical phenomena.

Acknowledgements

This Master's Thesis was conducted at Baxter Lund, and we would like to sincerely acknowledge the helpful support and technical insight provided by Bo Wennberg, Michael Pettersson and Robert Hallström at Baxter, as well as Fredrik Angsmark for the resources pooled and help offered with managing the project.

We would also like to express our gratitude for the interest in this project shown by MathWorks, and the helpful guidance to the Simscape environment provided by Juan Sagarduy.

Finally we would like to thank Tore Hägglund, for the guidance provided to us throughout the duration of this project.

Contents

1

Introduction

1.1 Background

The main function of the kidneys is to regulate the balance of electrolytes in the human body and remove waste products from the blood stream. In case of reduced kidney function or kidney failure, dialysis treatment is used to replace the vital functions of the kidney.

There are several different kinds of dialysis treatments, which are all based on the same principles - the exchange of waste products and electrolytes between a dialysate solution and the blood of the patient. In this project, the focus is on machinery related to hemodialysis, where the blood of the patient is allowed to pass through a dialysis machine at the same time as the dialysate fluid. The two fluids flow simultaneously past opposing sides of a semipermeable membrane, where the dialysate fluid absorbs waste particles and water from the blood while larger particles, such as blood cells, remain separated from the dialysate by the membrane [NIDDK, 2017].

Hemodialysis treatments requires large volumes of clinically pure water. A healthy person generally only subjects their blood to contact with water by drinking, approximately 14 L of water every week. In contrast, a person undergoing dialysis treatment can expose their blood to upwards of 600 L of water every week through the use of dialysis equipment [Hoenich and Ward, 2016], while the unusual means of exposure carry significantly increased risks. Ingested water is absorbed into the blood through the intestinal walls, which act as a barrier. While a low amount of toxins present in the water can be dangerous, the relatively small volume of water coupled with the protective properties of the intestines means that the risk of infection is comparably low. For patients undergoing dialysis treatment however, a large volume of water comes into contact with the bloodstream without these protective mechanisms, which means that even small amounts of toxins present in the dialysate can be dangerous. Consequently, it is of utmost importance that the water intended for use in dialysis treatment is as clean as possible [Hoenich and Ward, 2016].

For this purpose, clinically pure water is usually supplied to the dialysis machine directly from a nearby water purification unit. In this project, the Gambro WRO 300 H has been studied, it is a water purification unit that runs simultaneous to the dialysis machine and produces water of very high purity by use of a reverse osmosis process. The machine is designed to be simple and compact enough for in-home dialysis treatment, as well as use in clinics [Gambro, 2010].

1.2 Motivation

At present, no computer model of the Gambro WRO 300 H water purification system exists. The advantages with such a model are motivated by the opportunities offered by model-based design and engineering.

Model-based engineering is an approach to product development that integrates a model of the system in the development process. The use of computer simulations in the design process can reduces the need for prototypes, especially in the early stages of the engineering process [Gordon, 2012]. For a finished product, like the WRO 300 H, a model of the systems allow for analysis of the process, and simulation of certain anomalous circumstances that are impossible or impractical to reproduce in a lab environment. Furthermore, it is possible to simulate modifications of the system such as the introduction of regulator designs to suppress disturbances.

1.3 Goal

The project aims to create a computer model for the Gambro WRO 300 H which provides fair estimations of the system at a reasonable interval of temperature, inlet-water conductivity and membrane module pressure. While the accuracy of the model is important, the model should primarily be adaptable and reusable.

As a part of this process, an evaluation of the explored software possibilities should be performed, in regards to both specific and general requirements for a computer model.

As a proof of concept a simulated regulator that handles typical disturbances is to be designed and implemented in the computer model, to demonstrate the possibilities of model-based engineering.

1.4 Method

The Master's Thesis project is conducted at Baxter International. Baxter produces a broad range of medical products and specialize in dialysis equipment.

To complete the project, a suitable mathematical model for a reverse osmosis membrane should be selected, based on the literature and research available. The specific requirements placed on the computer model should be the deciding factor in the choice of model complexity. The choice of modeling tools and software is to be motivated by the suitability of the tools in regards to this project.

An adequate identification method is to be selected, based on the model form, available resources, as well as the limitations and freedoms offered by the WRO 300 H machine and the measurement possibilities. Through physical measurements, the parameters of the system are identified, from which the model can be constructed and implemented. The performance of the model is evaluated from a qualitative and quantitative view by simulating the system, and validation is done against the physical system.

2

Reverse Osmosis Theory

2.1 Osmosis

Osmosis and osmotic pressure

Osmosis is a naturally occurring process when two solutions of different chemical concentrations are separated by a semipermeable membrane. The membrane allows for the transfer of solvent fluids, while the transfer of solutes and suspended particles are rejected by the membrane [Baker, 2004].

The chemical potential between the two sides of the membrane leads to a solvent flow across it. The flow is directed such that it equalizes the concentrations of the two solutions. In a U-formed loop, such as in Fig. 2.1, this results in a volume increase on one side of the membrane and with that, an increase in hydrostatic pressure. This pressure counteracts the solvent transfer, and as such limits the chemical equalization of the two solutions.

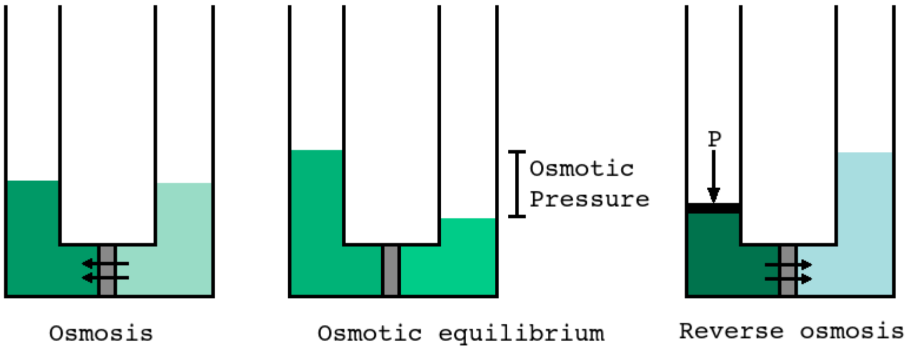


Figure 2.1 Principles of osmosis and reverse osmosis

The limit for which a given chemical potential can cause a liquid solvent transfer over a semipermeable membrane to a region of higher hydrostatic pressure is called the ***osmotic pressure***. The osmotic pressure is related to the chemical potential across the membrane, and as the two solutions are equalized, the osmotic pressure drops while the hydrostatic pressure grows. When the two pressures are equal, no solute transfer will occur, and the solutions are in equilibrium [Baker, 2004].

Reverse osmosis

If an external pressure is applied to the high-concentration side of the setup, this pressure also counteracts the osmotic pressure. The resulting net pressure decides the direction of the solvent flow. Thus, with an applied pressure that is high enough, the osmosis process is reversed and the solvent flow runs in the direction from the higher concentration side of the membrane to the lower concentration side [Baker, 2004].

This process is simply called reverse osmosis, and it can be used to dilute a solvent to produce a fluid of very high purity. The typical solutes such as salts and minerals are separated from the produced water, but equally important is that the reverse osmosis membrane also acts as a barrier that rejects virus and bacteria from contaminating the water [Kucera, 2010]. This technology is used in the Gambro WRO 300 H to produce clinically clean water for use in dialysis treatment.

2.2 Mathematical model of reverse osmosis

There are several popular models to describe the flow of solutes and solvents in a reverse osmosis process, where the goal usually is to relate these flows to controllable variables such as pressure and concentration difference between the solutions. The different models serve distinct purposes, and are based on different assumptions when deriving them. As such, they have varying degrees of complexity and usability in this project.

The models can be divided into three group: those based on irreversible thermodynamics, homogeneous membrane models, and porous membrane models [Williams, 2009].

Choice of model

A model can be derived from the assumption that flows can be explained by phenomenological relationships in a system close to thermodynamic equilibrium. These models, based on irreversible thermodynamics (IT), treat the membrane as a ***black box***. A mathematical model based on these assumptions doesn't include any parameters relating to actual physical properties of the membrane. Instead, it is solely expressed in flows and concentrations [Williams, 2009].

The porous membrane models by contrast, assume that the flows can be explained by transport through tiny pores in the semipermeable membrane. The porous models generally have a higher degree of complexity and include physical parameters describing the structure and properties of the membrane and its pores [Williams, 2009].

The homogeneous membrane models include the Solution-Diffusion (SD) model and variants thereof. The SD-models are the most popular choices in modern times of describing reverse osmosis transports [Baker, 2004]. SD-models generally have a lower degree of complexity compared to the porous models, while also describing the physical parameters of the membrane to some extent, most importantly it parametrizes the permeability of solvents and solutes through the membrane.

The aim of this project is not to represent the reverse osmosis membrane as accurately as possible, it should instead be a simple and usable component in a larger model of the complete WRO 300 H machinery. With this aim in mind, and because of its appropriate level of complexity, the solution-diffusion model is the membrane model used in this project.

Solution-diffusion model

All the mentioned transport models are based on the assumption that the driving force of the solvent through the RO-membrane is the gradient of its chemical potential. Furthermore, in this derivation it is assumed that the components being transported is pure water as the solvent, and common salt as the solute.

The Solution-Diffusion model assumes that the pressure within the membrane through which transportation takes place is uniform and that the chemical potential gradient can be expressed as a concentration gradient. The assumption that diffusion is driven by a concentration gradient is expressed in Fick's law, where the flux of a component *i* can be described as:

$$J_i = -D_i \frac{dC_i}{dz} \tag{2.1}$$

where D_i is the component diffusivity and C_i is the component concentration within the membrane module.

To relate this to the controllable aspects of the process, it is necessary to equate the chemical potential on both edges of the membrane [Wijmans and Baker, 1995].

For the feed side of the membrane, this results in the following expression:

$$c_{i0(m)} = K_i \cdot c_{i0} \tag{2.2}$$

where c_{i0} denotes the concentration of water on the feed side of the membrane, $c_{i0(m)}$ is the corresponding concentration inside the membrane and K_i is the sorption coefficient for the membrane.

According to Wijmans and Baker [1995], the corresponding potential on the permeate side of the membrane is:

$$c_{il(m)} = K_i \cdot c_{il} \cdot \exp\left(-v_i \frac{(P_0 - P_l)}{RT}\right) \tag{2.3}$$

where v_i is the water volume inside the membrane, P_0 is the pressure at the feed side, P_l is the pressure at the permeate side, R the gas constant and T the membrane temperature.

Insertion of these two expressions into Fick's law results in an expression for water flux:

$$J_i = \frac{D_i K_i}{l}\left[c_{i0} - c_{il} \cdot \exp\left(-v_i \frac{(P_0 - P_l)}{RT}\right)\right] \tag{2.4}$$

At osmotic equilibrium, the flux of water across the membrane is zero, and the hydrostatic pressure is equal to the osmotic pressure, which leads to the following relation between c_{il} and c_{i0}:

$$J_i = 0 \tag{2.5}$$

$$\Delta\pi = P_0 - P_l \tag{2.6}$$

$$\Longrightarrow$$

$$c_{il} = c_{i0} \cdot \exp\left(\frac{v_i(\Delta\Pi)}{RT}\right) \tag{2.7}$$

where $\Delta\pi$ is the osmotic pressure across the membrane.

Combining Eqs. (2.4) and (2.7) gives the following flux equation:

$$J_i = \frac{D_i K_i c_{i0}}{l}\left[1 - \exp\left(\frac{-v_i(\Delta P - \Delta\Pi)}{RT}\right)\right] \tag{2.8}$$

where ΔP is the hydraulic pressure differential across the membrane, l is the membrane thickness.

Under normal conditions of reverse osmosis the exponential factor in this expression is small. A Taylor approximation of degree zero $1-e^x \approx -x$ when x is small results in the approximate expression:

$$J_i = A(\Delta P - \Delta\Pi) \tag{2.9}$$

A is the permeability constant of water through the membrane, which can be calculated as follows:

$$A = \frac{D_i K_i c_{i0} v_i}{lRT} \tag{2.10}$$

According to Wijmans and Baker [1995], the corresponding calculation for the salt transportation results in the expression:

$$J_j = \frac{D_j K_j}{l}\left[c_{j0} - c_{jl}\cdot\exp\left(-v_j\frac{(P_0-P_l)}{RT}\right)\right] \tag{2.11}$$

Since the total salt volume present in the membrane v_j is typically very small the expression can be approximated to:

$$J_j = \frac{D_j K_j}{l}(c_{j0} - c_{jl}) \tag{2.12}$$

which can be written as:

$$J_j = B(c_{j0} - c_{jl}) \tag{2.13}$$

where B is the permeability of salt through the membrane.

It should be noted that a linearization of the system has been performed around a point where the salt volume is very small and the differential pressure is small in relation to the temperature.

In practice the membrane module is constituted by several sheets of membrane spirally coiled around a central pipeline. This can be difficult to properly describe through analytic methods, but a fair approximation [Mejvik and Olin, 2012] of the water flow can be made by assuming that the pressures and chemical potentials is equal across the entire homogeneous membrane area:

$$F_p = A_m A(\Delta P - \Delta\Pi) \tag{2.14}$$

where F_p is the permeate water flow and A_m is the total membrane area.

The Solution-Diffusion model assumes that the pressure inside the membrane is uniform and equal to the pressure on the feed side of it. In practice, pressure along the membrane feed side can vary, and in most applications the membrane is inserted into a membrane module, which is fed from one side and emptied from the other.

The pressure in such a module can vary, but under the assumption that the reject water is at a relatively high pressure compared to the product flow and assumed to be close to feed pressure levels, a reasonable assumption of the net pressure ΔP is:

$$\Delta P = \frac{P_f + P_r}{2} - P_p \tag{2.15}$$

where P_f is feed water pressure, P_r is reject pressure and P_P is product pressure.

According to Williams [2009], the osmotic pressure can be calculated as:

$$\Pi = \frac{iRT}{V_w} ln(X_s) \tag{2.16}$$

where V_w is water volume in the membrane and X_s is the molar fraction of solved salts. The correction factor i is the so called Van't Hoff factor and relates the concentration of molecule particles to the concentration of particles once solved. Sodium chloride has a Van't Hoff factor of two as each molecule of $NaCl$ dissolved in water produces two free ions.

According to [Dow, 2007] Eq. (2.16) can be approximated to:

$$\Pi = 1.12 \cdot T \cdot \sum_n M_n \tag{2.17}$$

where M_n is the molal concentration of the n:th ion species.

The salt flow across the membrane is:

$$F_{p(s)} = AmB(c_f - c_p) \tag{2.18}$$

It should be highlighted that the flow of water across the membrane is expressed as a function of pressure while the salt flow is primarily driven by a concentration difference.

A salt rejection rate can be defined as:

$$\mathbb{R} = 1 - \frac{c_p}{c_f} \tag{2.19}$$

Rearranging this expression [Williams, 2009] leads to the following equation:

$$\frac{1}{\mathbb{R}} = 1 + \frac{B}{A} \cdot \frac{1}{\Delta P - \Delta \Pi} \tag{2.20}$$

This clearly shows that the rejection rate is pressure dependent. A higher pressure at equal chemical distribution results in a larger water flow across the membrane, while the salt flow remains the same.

3 Equipment

3.1 Gambro WRO 300 H

The Gambro WRO 300 H is a water purification machine that utilizes reverse osmosis to produce clinically clean water for use with dialysis machines. For proper function of the machinery, the water fed to the system needs to be potable and filtered (<5 µm). Furthermore, the water needs to be very soft (<0.3°dH), otherwise significant scaling of the reverse osmosis membrane may occur [Gambro, 2010].

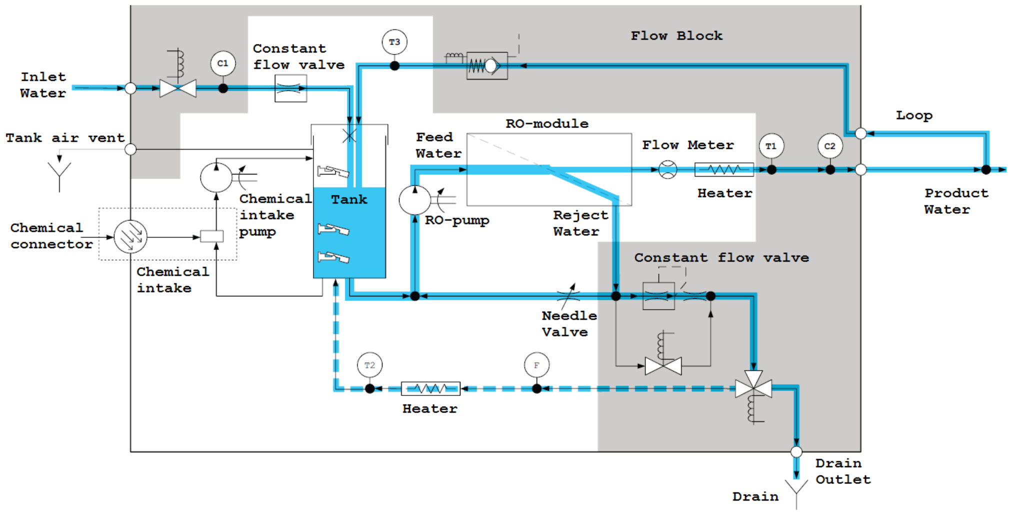
Flow diagram

A flow diagram of the WRO 300 H is presented in Fig. 3.1. The WRO 300 H is built around a central loop in which the membrane module is fed with high pressure water from a water pump, while a needle valve allows for circulation of the water while maintaining a high module pressure.

By adjusting the needle valve, the module pressure across the reverse osmosis membrane is altered, higher pressure leads to a larger product water flow, as in Eq. (2.14). Because the pressure is adjusted with the needle valve, this also means that the circulation flow decreases with higher pressure. A lack of circulation in the membrane module leads to scaling and fouling of the membrane, while the higher pressure increases the load on the machine and it's components. Consequently, a lower module pressure and higher circulation flow is preferred.

Water is fed to the central loop from an internal water tank. The product water is delivered to the dialysis machine, while any excess unused water is diverted back to the internal water tank in the WRO 300 H. The reject water is discharged from the central loop at a constant rate through a constant flow valve. A small portion of the reject water is then fed back to the water tank when the machine is in water save mode.

The internal tank is fed with fresh water from the water inlet. Sensors and heaters are placed throughout the machine in accordance with Fig. 3.1.



Credit: Gambro Lundia AB, modifications have been made

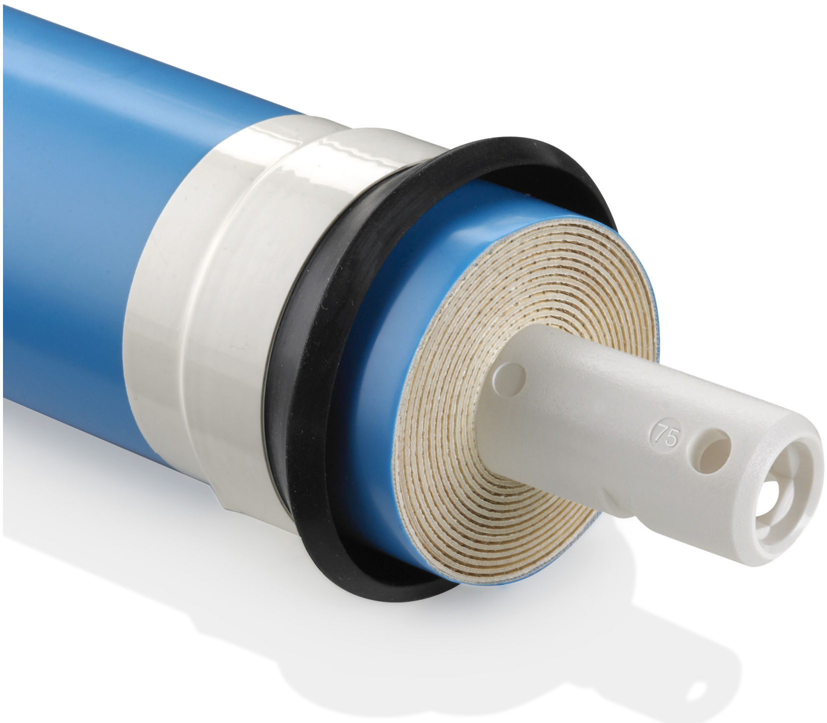
Figure 3.1 Flow diagram, WRO 300H.

Membrane module

The membrane module in the WRO 300 H is designed in a spiral configuration where several membrane sheets are folded around an inner permeate collection channel. The sheets extend from a central pipe and wraps around it to produce the spiral configuration of the cylindrical membrane module, as seen in Fig. 3.2.

The feed water flows from end to end of the membrane module under high pressure, where a portion of it permeates through the membrane and runs through the collection channels to the central pipe, which leads to the product flow outlet.

The fluid that did not permeate through the membrane is called the reject fluid, and is recirculated in the feed water loop, while a portion of it is drained. While most of the solute particles are led out of the membrane module by the reject water flow, a portion of the particles attach to the membrane sheets. This causes a fouling of the membrane which can damage the efficiency of the module in the long term. Low reject water flow through the membrane aggravates this effect [Gambro, 2008].



Credit: Dow Water & Process Solutions

Figure 3.2 A spiral wound membrane, with product flow pipe portruding from the center.

While the general filtering properties of the module certainly depend on the design properties of the membrane, such as its total membrane area, membrane thickness, and the permeability characteristics of the membrane materials, there are considerable variations between individual membrane modules. Due to aging, fouling, scaling and production irregularities, each module has unique properties and a computer model that adequately describes one membrane might not be suitable for another of the same design.

Methods of measurements

To conveniently measure levels of salts, bacteria and other trace elements in water, the WRO 300 H uses a related quantity - water conductivity. The conductivity of a solution is linearly correlated to its salinity [Apps, 2013], with different coefficients for different chemical components. Similarly, bacterial activity in a solution is related to the conductivity of the solution [Cady et al., 1978]. As such, conductivity sensors is the standard tool for online measurement of water purity in purification units such as the WRO 300 H.

The built in measurement system of the WRO 300 H is not designed for system identification purposes, but rather for system supervision and control. It simply needs to alert when signs of reduced function or failure appears. As such, the machine has been modified during the course of the project, to enable accurate modeling of the reverse osmosis membrane.

Conductivity and temperature The conductivity sensors on the WRO 300 H measures the conductivity of the inlet and product water. The sensitivity of the conductivity sensors is about $\pm 10\%$ or $\pm$ 10 µS/cm, whichever is higher. The sensor has a significant inherent delay due to how the sensor system is implemented. The measurement is averaged over a time interval and acts as a low-pass filter [Gambro, 2008].

The measured conductivity is not only related to the chemical composition of the water, but also varies with temperature. The actual conductivity value is calculated by the machine software using both the measured temperature and the raw conductivity measurements. Ideally the temperature and conductivity sensors would be placed as close to one another as possible. As can be seen in the flow diagram 3.1, this is true for the product water, but not for the inlet water.

While there is no temperature sensor close to the inlet conductivity sensor, there are two placed in the product loop and another one in the water saving loop. Under normal working conditions the temperature is only slightly varying throughout the machine which suggests that the separation of conductivity and temperature sensors, while not ideal, doesn't introduce any significant measurement errors when comparing the product flow temperature to the rest of the machine. However, the low flow in the water saving and product return results in a significant temperature difference in comparison to the rest of the system. As such, these temperature readings should not be used for any identification purposes.

Flow and pressure The WRO 300H only has one flow sensor, which is placed in the product water outlet. According to the solution-diffusion model, this flow is not directly correlated to any other flows through the machine or past the membrane, but rather, it's related to the pressure across it. While there could be interesting details that would reveal themselves with further flow measurements, they are not strictly necessary for a functional identification of the membrane.

The WRO 300 H has no built-in pressure sensors, but instead has a socket for pressure measurements at the reject end of the membrane module. The pressure is assumed to be uniform throughout the membrane module, which should not be understood as an absolute fact. The spiral configuration of the membrane suggests a certain radial pressure drop, while the way the feed and reject water connections are made to either end of the cylindrical membrane module, further suggests a longitudinal pressure drop in the module. The assumption of uniform pressure throughout the module thus implies that these pressure drops are assumed to be small compared to both the hydraulic pressure in the feed water loop, and the pressure drop across the membrane.

Concerns All in all, there are two concerns with the available measurement systems in the unmodified machine that must be addressed when taking measurements for identification purposes - conductivity measurement of the feed water flow, and conductivity measurement accuracy for the product water flow.

The conductivity of the feed water must be measured to enable a parameter estimation of the RO-membrane. The only way to perform such a measurement, is to insert a sensor in the feed water loop, between the RO-pump and the RO-membrane. This will undoubtedly impact the behavior of the system by introducing further hydraulic resistances in a high-pressure part of the machine, but as an exact estimation of the RO-membrane parameters is of utmost priority the benefits of this approach is considered to outweigh the drawbacks.

While an accuracy of $\pm$ 10% is acceptable in the inlet water and feed water measurements, an absolute error of $\pm$ 10 µS/cm in the product flow measurements would translate into a relative error between $\pm$ 100 – 200% for a product water flow of 4 – 10 µS/cm, far beyond any acceptable limits. Furthermore, the built-in measurement system is only graded down to 1 µS/cm, which also provides a rounding error that completely renders the measurements unusable for identification purposes. Luckily, this problem is easily solved by fitting the product loop with an external conductivity measurement system.

3.2 External Measurements

Conductivity

WTW Cond 3110 The WTW Cond 3310 is a handheld conductometer, which combines conductivity and temperature measurement equipment. It has a measuring range of 0.0 to 199.9 µS/cm at a resolution of 0.1. The temperature range is between −5.0 and 105 °C at a resolution of 0.1.

Pressure

Keller LEO 1 & LEO 2 The Keller manometers are handheld pressure sensors with a range of 0.0 to 300.0 bar and an accuracy of $< 0.1\%$ of full scale at room temperature. The operating temperature is between 0 – 50 °C.

Keller PA-23 The Keller PA-23 is a piezoresistive pressure transmitter with a range from 0.0 to 200.0 bar, with an accuracy pf $\pm 0.2\%$ of full scale. The operating temperature is between 0 – 100 °C.

3.3 Logging

The on-board measurements from the WRO 300 H have been logged using GXL 3.2 from the Baxter Service Tools library. The pressure measurements from the Keller PA-23 were transmitted to a computer and then logged in LabVIEW by using a National Instruments NI-9215 analog input module and cDAQ-9171 data acquisition system.

4

Simscape

4.1 Background

The toolbox used in this project to build the computer model of the WRO 300H is MathWorks Simscape, an extension of the MathWorks Simulink environment that focuses on the modeling of systems from a physical network approach. In this approach, the system is represented as a network of components that exchange energy with each other through their ports [Mathworks, 2017a].

The entire system is completely defined by the domains and components it uses, as well as the connections between the components. When simulating, the network is expressed as a system of differential equations which is solved simultaneously for each time step [Miller, 2010].

4.2 Domains and connections in Simscape

A domain in a Simscape model defines the physical quantities for a certain type of connection which the components can manipulate, and a set of general rules that apply for them [Miller, 2010]. In an electrical circuit for example, Kirchoff's Voltage Law (KVL) and Kirchoff's Current Law (KCL) states that the sum of the electrical potential differences around any closed circuit must be zero, and that sum of electrical currents into and out from a single node must be equal. In Simscape, these physical quantities are expressed as ***through*** and ***across*** variables. These are called conjugate variables, and together define the energy exchange between the components completely [Mathworks, 2017d]. KVL implies that the voltage at any component connection attached to the same node is equal, which defines the voltage as the ***across*** variable of an electrical circuit, and the difference in electrical potential between two nodes is equal to the voltage ***across*** a component between them.

On the other hand, KCL states that the electrical current through a component must be balanced, and that no electrical charges can leave a closed circuit. Thus, the electrical current is defined as the ***through*** variable of an electrical circuit and the current ***through*** a component is equal to the sum of all currents entering or exiting the component.

These relations are defined in the `domain` declaration of a Simscape library. The conjugate variables are defined in the `variables` blocks, where through variables use the additional keywords `Balancing = true` [Mathworks, 2017d].

Listing 4.1 Variable declaration in an electrical circuit domain

```
domain electrical
...
    variables
        v = { 0 , 'V' };     % Defines voltage as an across variable
    end

    variables(Balancing = true)
        i = { 0 , 'A' };     % Defines current as a through variable
    end
end
```

4.3 Components in Simscape

A domain in Simscape provides a very basic set of rules for how the connections in a certain domain work. The components use these connections to create a physical network that Simscape can solve.

Besides defining a set of connections that acts on a certain domain, the components also must define a set of balancing equations that describe how the component affects its connections. A simple electrical resistor for instance might look like listing 4.2.

Listing 4.2 Class definition for a simple resistor

```
component resistor
    nodes
        p = foundation.electrical.electrical;
        n = foundation.electrical.electrical;
    end
    variables
        i = { 0, 'A' };      % Current
        v = { 0, 'V' };      % Voltage
    end
    parameters
        R = { 1, 'Ohm' };    % Resistance
    end
    branches
        i : p.i -> n.i;
    end
    equations
        v == p.v - n.v;      % Equations,
        v == i*R;            % not assignments
    end
end
```

The `nodes` block defines the two physical connections of the resistor which act on the electrical domain, while the `variables` and `parameters` blocks define the two variables used in the fundamental equations of the component, as well as the resistance of the component. The `branches` statement declares that the current through the component is its internal current from node `p` to node `n`. This ensures that the component itself complies with KCL and defines the positive direction of the current. Finally, the `equations` block relates the difference in electrical potential across the resistor to the current through it, in accordance with Ohm's law.

4.4 Physical network modeling

Advantages

By defining the necessary domains and components, a relatively small library of objects can be used to describe large and complex systems, similarly to how a traditional circuit diagram uses a set of a few basic components to build a larger system.

In this modeling approach, all physical connections in the actual system are represented as connections between model components. The modeling is reduced down to programming correct and computationally robust components, and connecting them in a manner that corresponds to the desired system. This makes for a very modular and flexible model, where individual components or basic designs features are easily altered.

The aim of this project is to create a usable and reusable model, which is easily adaptable to accommodate for simple internal and external changes to the machinery and its setup. As such, the physical network modeling approach is a favorable way to achieve this, in part because of the inherent flexibility of the method, but also because of the general strictness of physical network models in adherence to basic physical phenomenon. Physically impossible or erroneous models quickly reveal themselves, usually in the compilation phase, as the system becomes insolvable [Mathworks, 2017b].

Due to the authors previous knowledge and experience with the Matlab/Simulink environment, Simscape was the logical choice in this project as the modeling tool for a physical network model of the Gambro WRO 300 H.

Drawbacks

While the physical network modeling approach provides an intuitive work flow, it also comes with a few drawbacks. Computationally, the strictness of the general model and simulation of it, means that a great deal of care must be taken when programming the model components. These tend to become highly idealized, which can introduce singularities to the model if special care is not given to close-to-zero cases [Mathworks, 2017b].

Furthermore, the variable-step solver used by Simscape means that special consideration should be given to zero crossings in the model. Because of the variable-step solver, the dynamics around such crossings will be accurately captured. However, consecutive zero crossings will significantly slow down the simulation of a system, because of diminishing time steps [Mathworks, 2017c]. Generally, these situations should be avoided in the early design phases of the model.

5

Model design choices

5.1 Design process

To produce an accurate physical network model, one must assure that the individual components of the model accurately represent components of the machinery, and that the way that these components interact with one another corresponds to the physical interaction between the components in the physical WRO 300 H machinery.

A certain leeway has been taken as related to the first of these two points. Certain components such as the central pump and valves are hard to measure and parameterize accurately, and a different approach has been used to solve this problem. Instead of a completely accurate representation of the individual components, the method used is to Fermi estimate many of the individual component parameters, in particular those of less importance, and then use one or two fundamental parameters to tune the components in such a way that the average behavior of the entire physical network model matches the machinery.

This has rapidly sped up the modeling process by sharply reducing the necessary number of measurements and need for accurate component data sheets, albeit that the reusability and adaptability of the model has suffered.

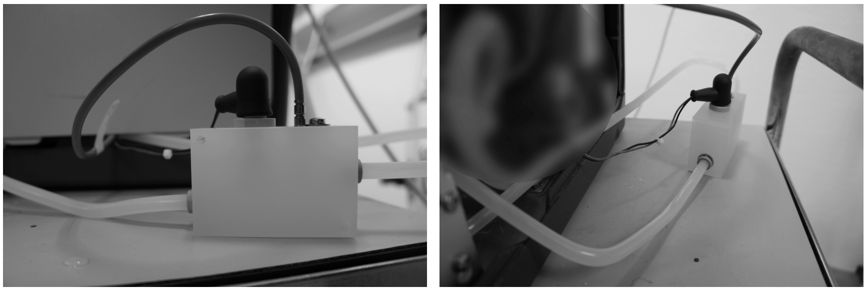
As a whole, the Simscape model of the Gambro WRO 300 H has been developed concurrently with measurements of the system and modifications to the machinery. The reason for this work flow is that a qualitative understanding about Simscape and important phenomenon in the computer model has eased the process of system identification, by giving a clear view of what elements in the machine that must be accurately measured and represented in the model.

The crucial component in this regard is unsurprisingly the reverse osmosis membrane module itself. Accordingly, the primary objective when performing measurements on the WRO 300 H, has been to accurately estimate the permeability parameters of the membrane module.

5.2 Modifications

The standard measuring equipment in the WRO 300 H was quickly determined to be insufficient to accurately determine the permeability parameters of the RO membrane module experimentally. To handle this problem, two modifications were made to the machine and lab setup.

Feed flow conductivity



(a) Flow block (b) Feed flow loop

Figure 5.1 The modified feed flow loop.

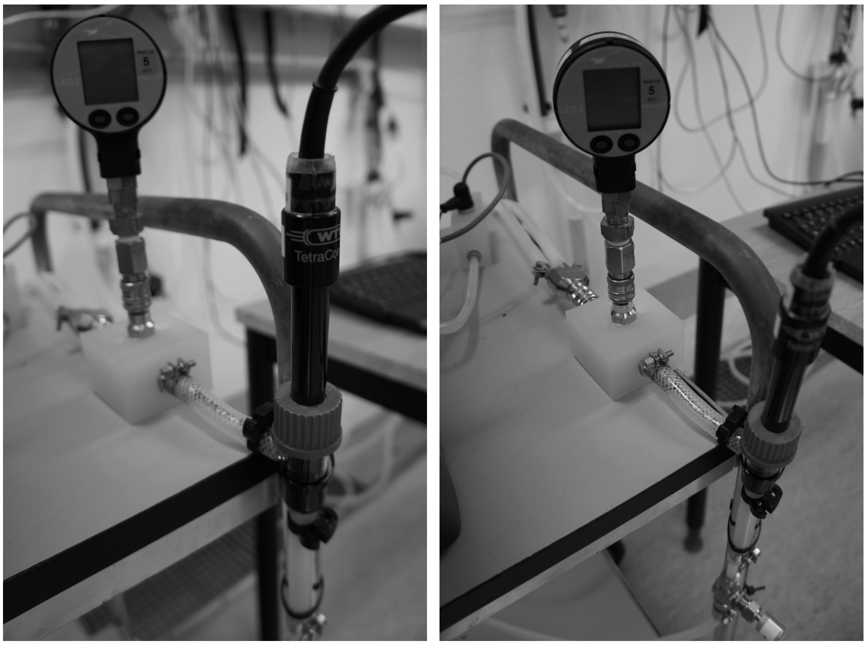
To enable conductivity measurements on the feed water flow, an extended piece of flow tubes were inserted in the machinery, between the pump and the membrane module. As seen in Fig. 5.1, a specially designed block was placed in the loop, with sensors for conductivity and temperature measurements attached to it. The measurements were performed by the standard system of the WRO 300 H, by replacing sensor slots that were deemed dispensable. This was made possible by the fact that the feed loop sensors were identical to the standard sensors on the WRO 300 H machinery.

Product flow accuracy

The precision of the standard sensors in the WRO 300 H were found to be lacking in regards to the product flow conductivity. This was solved by placing a hand held sensor between the product loop and the drain, as seen in Fig. 5.2.

5.3 Limitations

A great deal of limitations has been made to the overall design of the computer model. The reason for these often are very practical—if a phenomenon is hard to measure or model in comparison to the benefits a successful model gives in regards



(a) WTW Cond 3110 conductivity sensor (b) Keller LEO 2 pressure gauge

Figure 5.2 The product loop with attached conductivity and pressure sensors.

to achieving the goals of the project, it is almost certainly preferable to work around it.

Heat exchange modeling

Modeling the temperature distribution of the system, was initially regarded as a secondary or superfluous task. Later tests confirmed this, as the temperature throughout the machinery revealed itself to be almost identical within the standard measurement accuracy for the equipment, although clearly time-varying. Because of these reasons and the increased complexity that heat exchange modeling would incur on the project, the complete modeling of heat exchange between internal components was scrapped in favor of simply declaring the temperature as a global parameter for the entire model.

This leads to a primary limitation of the model. The temperature of the system varies with several factors—ambient temperature, inlet water temperature, pump power output and flow rates through the machinery. After a few minutes of running, the WRO reaches a thermal equilibrium. Since the Simscape implementation does not include a heat exchange model, this transient state is not represented in the physical network model of the WRO 300 H.

Transients and delays

While the exclusion of heat exchange modeling limits the representation of transients and delays in the system, there is one other important aspect in which the representation of delays are limited.

The volume flow through the system is regarded as an incompressible flow, which implies that there is no inertia in regards to the pressure propagation within a single component or a single node. This should not be any problem due to the small flow velocity compared to the speed of sound in the fluid.

However, the propagation of the salt mass flow rate within an individual component should be relative to the flow velocity and longitudinal length of the component, rather than the speed of sound. In the Simscape model, the salt mass flow rate propagates instantly, which cannot be justified with the same reasoning as the pressure propagation.

Consequently, the representation of internal delays in the model in regards to the salt transfer becomes limited. While an analytical solution could be constructed to solve this problem, there is no way to measure and validate these dynamics. Because of this, the representation of transients and delays has been given a lower priority, and are not accurately included in the Simscape model.

An important limitation is imposed on the model by the technical traits of the measurement system. Due to the low-pass filtering properties of the conductivity sensors it is infeasible to acquire accurate measurements of the transients of the system with existing equipment. The guidelines advice the operator of the machine to flush the system for a few minutes before initiating the dialysis treatment to ensure a high quality of product water. The WRO 300 H is typically run for hours at a time at approximately stationary outputs. Rather than to modify the equipment to incorporate the relatively short-term transient in the model it was thus reasoned that these dynamics are of low importance in the context of how the model would be utilized.

Salt representation

The WRO 300 H is used to separate water from various bacteria, viruses and minerals. The computer model drastically simplifies this by assuming that all contaminants are dissolved salts, which obey the relations set up by the solution-diffusion model. Furthermore, to simplify the conversion between the measured quantity (conductivity) and the fundamental property of the model (salt mass), all dissolved salts are assumed to be sodium chloride, which has a constant conversion factor between the two quantities. The simplification reduces the model complexity at the cost of a certain precision loss, and is justified by the fact that there exists a correlation between bacterial activity and solvent conductivity [Cady et al., 1978].

Furthermore, the salt volume is assumed to be negligible, and the density of the fluid is assumed to be unaffected by the salt concentration in the water. This is motivated by the fact that the total salt masses in the system are extremely small in this implementation, and inclusion of this factor would incur an additional degree of complexity when calculating the water parameters for each individual component in every time step.

6

Measurements

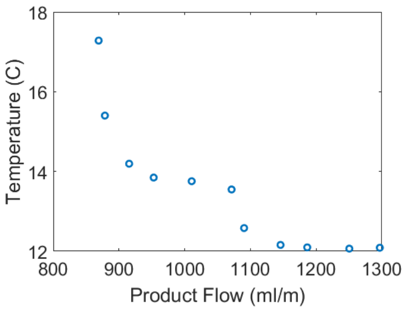
6.1 Method

All measurements were made under the special circumstance of constant product loop pressure, this was intended to simulate the back pressure caused by a connected dialysis machine. The pressure was monitored by a hand held pressure gauge attached to the product loop, and controlled by a manual gauge, as seen in Fig. 5.2. The membrane pressure was controlled by one of two methods in each measurement series, either by steering it with the RO-pump speed setting, or by adjusting the needle valve. When performing parameter estimation, the system was allowed to reach stable and steady values before measuring, this was repeated for several module pressures. In cases where the data is presented without a time axis the measurement series have been compiled into mean values representing the steady state value for each of the set control values.

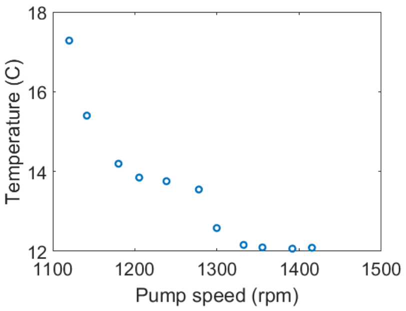
6.2 Measurements

Temperature measurements

Fig. 6.2(a) shows the water temperature measured in the feed sensor. It is evident that the temperature varies significantly with the pump speed. In the figure the pump speed is represented as a percentage number of a maximum pump speed. 35 % is roughly equivalent to 870 rpm while 45 % is approximately 1300 rpm. The temperature seems to vary non-linearly with pump speed and product flow rate. See Figs. 6.1(a) and 6.1(b) which show the relationship between temperature and pump speed, and product water flow rate respectively. Note that pump speed and product flow rate are roughly linearly proportional in the entire region, as seen in Fig. 6.2(b).

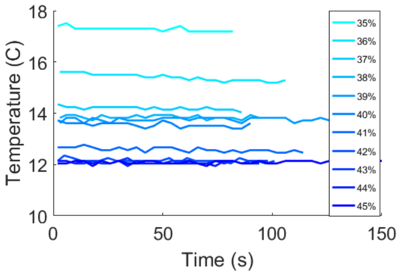


(a) Temperature as a function of pump speed.

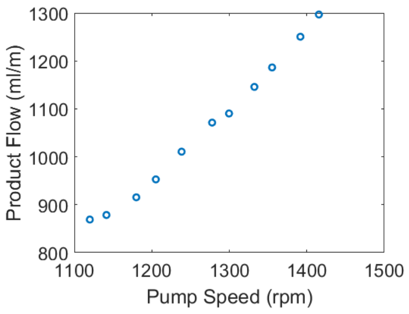


(b) Temperature as a function of flow rate.

Figure 6.1 Temperature at different pump speeds.



(a) Temperature at various pump speeds.



(b) Product flow rate as a function of pump speed.

Figure 6.2 Temperature measurements and flow rate as a function of pump speed.

Inlet conductivity

The inlet water conductivity and temperature changed on a daily basis during experimentation but so slowly varying it can be considered constant during one measurement series. See Fig. 6.3. It should be noted that the conductivity measurement is assumed to be accurate within 10% or ± 10 µS/cm, whichever is largest. Accordingly there could be variations up to ± 17 µS/cm that are undetected.

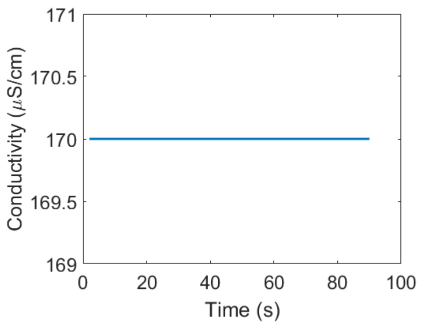
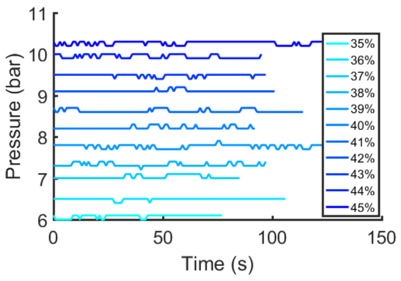


Figure 6.3 The measured inlet water conductivity during one measurement series.

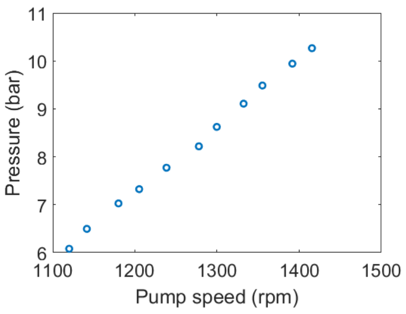
Pressure and water flow

The net driving pressure was calculated by subtracting the product loop pressure (constant $P_{prod} = 1.5$) and osmotic pressure from the measured module pressure. The osmotic pressure was calculated by translating the feed and product flow conductivities to a corresponding molal concentration of NaCl, and then applying Eq. (2.17) to the measurements.

Fig. 6.4(a) shows the module pressure measured at different pump speeds. Figs. 6.4(b) shows the relationship between pump speed and module pressure. As seen in Figs. 6.4(b) and 6.2(b), both product water flow rate and module pressure are close to linearly dependent to the controlling pump speed. The data represented in the first set of figures is based on an experiment where the membrane pressure was controlled by way of varying the pump speed. When the corresponding data was collected in an experiment where the pressure was varied using the needle pump the data was slightly different, as seen in Fig. 6.6(a).

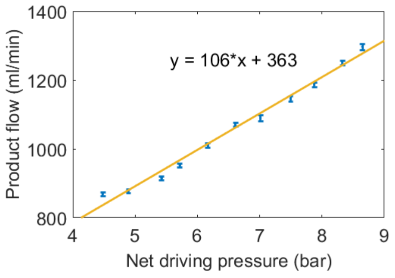


(a) Module pressure at various pump speeds.

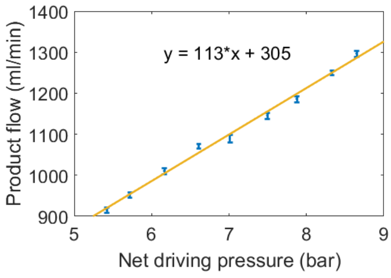


(b) Module pressure as a function of pump speed

Figure 6.4 Module pressure, controlled by pump speed setting.

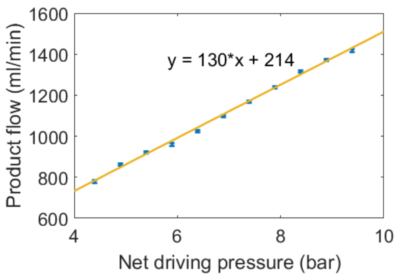


(a) Linear estimation of water permeability.

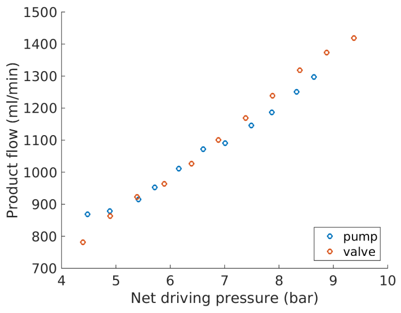


(b) Linear estimation of water permeability, outliers removed.

Figure 6.5 Product flow as a function of net driving pressure, with linear estimation of membrane water permeability. Module pressure controlled by pump speed.



(a) Linear estimation of water permeability.



(b) Product flow as a function of net driving pressure, two different modes of control.

Figure 6.6 Product flow as a function of net driving pressure when controlled by needle valve. Comparison between different modes of control.

Salt flow

Fig. 6.7 shows the typical salt flow as a function of the concentration difference. The figure includes two measurements, one from a series collected while controlling the membrane pressure and flow with the pump, and one while controlling by the needle valve. The figure highlights the non-linearity of the slope and the apparent difference between modes of control.

Fig. 6.8 shows a salt flow measurement. Instead of salt flow, the product conductivity has been plotted against the product flow rate, both for pump controlled and needle controlled measurement series.

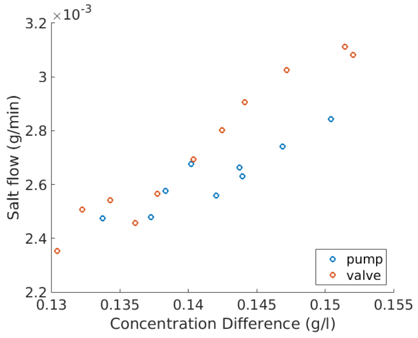
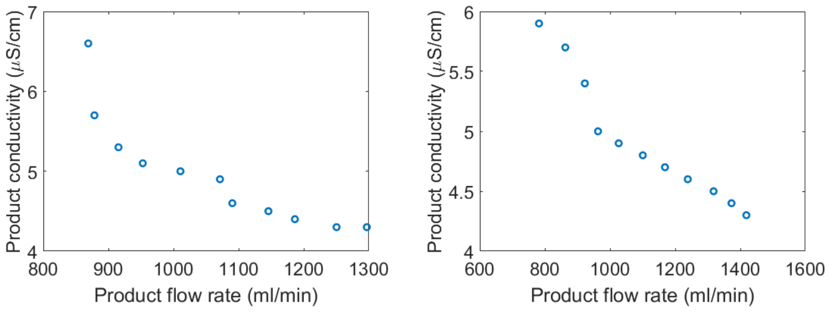


Figure 6.7 Salt flow as a function of concentration difference.



(a) Pump controlled.

(b) Needle controlled

Figure 6.8 Product conductivity as a function of product flow rate.

7

Implementation

7.1 Saline hydraulic domain

The domain used for the Simscape model of the WRO 300H is a saline water domain, which is an extension of the hydraulic domain in the Simscape foundation library:

Listing 7.1 Hydraulic domain class definition

```
domain hydraulic
% Hydraulic Domain
% Copyright 2005-2013 The MathWorks, Inc.
  parameters
    density       = { 850   , 'kg/m^3' }; % Fluid density
    viscosity_kin = { 18e-6 , 'm^2/s'  }; % Kinematic viscosity
    bulk          = { 0.8e9 , 'Pa'     }; % Bulk modulus at atm.
                                          % pressure and no gas
    alpha         = { 0.005 , '1'      }; % Relative amount
                                          % of trapped air
  end

  variables
    p = { 0 , 'Pa' };
  end

  variables(Balancing = true)
    q = { 0 , 'm^3/s' };
  end
end
```

In this domain declaration, the pressure is defined as the across variable of the domain, while the volume flow of water is the through variable of it. The domain design mirrors the example given of the electrical domain in 4.1. This follows naturally from the behaviour of fluids which corresponds to the voltage and current laws of Kirchoff. This basic relation has not been changed in the saline hydraulic domain, and the same hydraulic relations have been used to calculate the flows and pressure drops across the system.

The way the hydraulic domain has been extended to the saline hydraulic domain, is primarily by the inclusion of a mass flow as an additional through variable in the domain, which represents the flow of salts through the network.

Listing 7.2 Changes to conjugate variables

```
domain salt_dom
    % Saline hydraulic Domain
...
    variables(Balancing = true)
        q = { 1e-3 , 'm^3/s' }; % Flow rate
        msalt = {1e-3, 'kg/s'}; % Salt mass rate
    end
end
```

This quantity does not have a corresponding across variable that it interacts with throughout the physical network, instead it is defined in each individual component by relating it to the volume flow of water through it. This is done by ensuring that the salt concentration of the flow through a component is consistent with either the concentration entering it, or that of the accumulated water inside the component.

Furthermore, the saline hydraulic domain vastly extends the water parameters of the domain, by defining pressure and temperature tables for the density, kinematic viscosity, bulk modulus and thermal expansion coefficient. Correct values for these parameters are then extrapolated at runtime by using the defined temperature of the system and the pressure in the individual components.

7.2 Components

A number of components have been designed for use with the saline hydraulic domain, some of which are adaptations of hydraulic components in the standard Simscape hydraulic library or thermal liquid library. Out of the components designed from scratch, the model for the internal water tank and membrane model are central to the Simscape implementation.

Internal water tank

The internal water tank supplies the central loop of the machine with water, and is itself supplied from three different nodes—the main inlet, the product loop, and the reject loop. The water tank is also vented in case of overflow, the node definitions are shown in listing 7.3.

Listing 7.3 Water tank node definitions

```
component watertank
...
    nodes
        IN = saltdomain.salt_dom;    %IN:top
        LOOP = saltdomain.salt_dom; %LOOP:top
        VENT = saltdomain.salt_dom; %VENT:top
        OUT = saltdomain.salt_dom;  %OUT:bottom
    end
```

For each of these nodes, the fluid flow through a certain node is calculated by defining the pressure on either side of the node, and then using a formulation of Bernoulli's principle to calculate the volumetric flow through it, as seen in listing 7.4.

Listing 7.4 Flow calculations to OUT from water tank

```
component watertank
...
    equations
        let
            % ACROSS
            p_out    = OUT.p;
...
            % OUT
            p = if ge(h,0), p_amb + rho_OUT*g*h else p_amb end;
...
        in
...
            % OUT
            if (h>0)
                rho_OUT*q_out^2 == a_out^2*2*(p-p_out);
            else
                q_out == 0;
            end
...
        end
    end
```

In this implementation, `p` is the internal pressure at node `OUT`, `p_out` is the pressure at the node connected to `OUT`. The corresponding flow through node `OUT` is `q_out`, while `a_out` is the cross-section area of the node. The water density is `rho_OUT` and `p_amb` is the ambient pressure. The total height of the internal water volume is `h`, and `g` is the gravitational acceleration.

The correct salt mass flow away from the tank through a node is calculated by assuming that the concentration of the water leaving the tank is the same as the water inside the tank. The salt mass inside the tank is calculated by declaring that time derivative of the internally accumulated salt mass is the sum of all salt mass flows entering and leaving the water tank. The salt levels are thus assumed to be uniform throughout the water tank.

Listing 7.5 Calculation of accumulated salt mass, and salt mass flow to OUT

```
component watertank
...
    equations
...
        in
...
            macc.der == msalt_in + msalt_out + msalt_vent + msalt_loop;

             % OUT
            if (V>0) && (q_out <0)
                (msalt_out/(q_out)) == (macc/V);
            else
                msalt_out == 0;
            end
...
        end
    end
```

Reverse osmosis membrane

The RO membrane module contains three nodes, and uses the permeability constants for water and salt transfers as the main parameters for the components.

Listing 7.6 Node and parameter definitions for RO module

```
component RO_module
...
    nodes
        FEED = saltdomain.salt_dom; % feed:left
        PROD = saltdomain.salt_dom; % product:right
        REJ = saltdomain.salt_dom;  % reject:right
    end

    parameters
        % Reverse osmosis consants
        Kw = {129.6394e-3, 'l/(bar*min)'};
        Kw_konst = {213.1505e-3, 'l/min'};
        Ks = {0.035392, 'l/min'};
...
        p_loss = {0.01, '1'} ;      % reject pressure loss ratio
    end

    parameters(Access=protected)
        Nsalt = {0.058, 'kg/mol'};          % molar mass of salt
        kosm = {1.19, 'psi*kg/(K*mol)'};    % constant osmosis
...
    end
```

The parameters are unique to a certain membrane, and have been evaluated by performing measurements on a membrane module. In both the cases where the pressure was controlled by needle vent and the pump, the measurements incicated a linear relationship between product flow net driving pressure (Figs. 6.5(b) and 6.6(a). The figures have been supplemented with linear approximations based on least-square estimates. Both the slope and the offset are slightly different from each other. As seen in Figs. 6.6(a) and 6.6(b) the linear equations mapping pressure to water flow does not pass through the origin. To ensure correct behaviour in the pressure interval of interest, this positive bias is included in the model as seen in listing 7.6. The water flow across the membrane is calculated in accordance with Eq. (2.14), with the aforementioned constant added.

The assumed linear relationship between salt flow and concentration difference was not corroborated by the data 6.7. Instead the permeability has been assumed to be linear within a smaller range than presented in the measurement figures. While the salt mass permeability estimation suffers from a positive bias, this has been disregarded in the implementation, and the salt mass flow across the membrane is calculated using Eq. (2.18), as seen in 7.7

Listing 7.7 RO module equations

```
component RO_module
...
    equations
    ...
        in
            % Pressure loss from inlet to drain
            p_rej == p_feed*(1-p_loss);

            % Flow equation, conservation of water flow rates
            q_feed + q_prod + q_rej == 0;
            % Conservation of salt flow rates
            msalt_feed + msalt_prod + msalt_rej == 0;

            % Concentration gradient across membrane
            if (q_feed>0 && q_prod<0)
                deltaC == ((msalt_prod/q_prod)-(msalt_feed/q_feed));
            else
                deltaC == 0;
            end

            % Calculation of pressure and osmotic pressure
            deltaP == p_prod-(p_feed+p_rej)/2;

            % Two ions for each molecule of salt
            deltaPosm == 2*kosm*T*deltaC/(rho*Nsalt);

            %Water through membrane
            q_prod == (deltaP-deltaPosm)*Kw-Kw_konst;
            %Salt through membrane
            msalt_prod == deltaC*Ks;
        end
    end
```

The original assumption was that there was no pressure loss between the feed and reject nodes of the membrane module. As shown by the positive bias in the water permeability measurements such as in Fig. 6.5(b), this assumption may be doubtful. An ad hoc solution has been made to assume a percentual pressure loss between the feed and reject nodes.

The hydraulic pressure difference `deltaP` is calculated by using the average pressure between the feed and reject nodes and the osmotic pressure `deltaPosm` is calculated according to Eq. (2.16).

The permeability constants are measured and estimated as related to the reject pressure. When assuming that no relative pressure loss exists, `p_loss = 0`, the model complies with this in a correct way. When a loss ratio is added to the model, the measured permeability constants are applied to the average pressure in the membrane module, which differs from the reject pressure. As such, using the `p_loss` factor in the model will also introduce an error to the system.

7.3 WRO 300 H model design

Subsystems

A number of subsystems are included in the model, these are used to avoid cluttering in situations where more extensive calculations with physical signals are to be made. Most of the used subsystems represent certain parts of the machinery or lab setup which are regulated in some way, and these subsystems include the regulator blocks as well as the controlled hydraulic components.

Constant flow valve The constant flow valve is constructed by placing an ideal flow sensor in line with a controlled valve, and using a PI-controller to regulate the valve. A reference flow input is passed from the supersystem, while a short delay is introduced to the flow measurements to avoid numerical singularities.

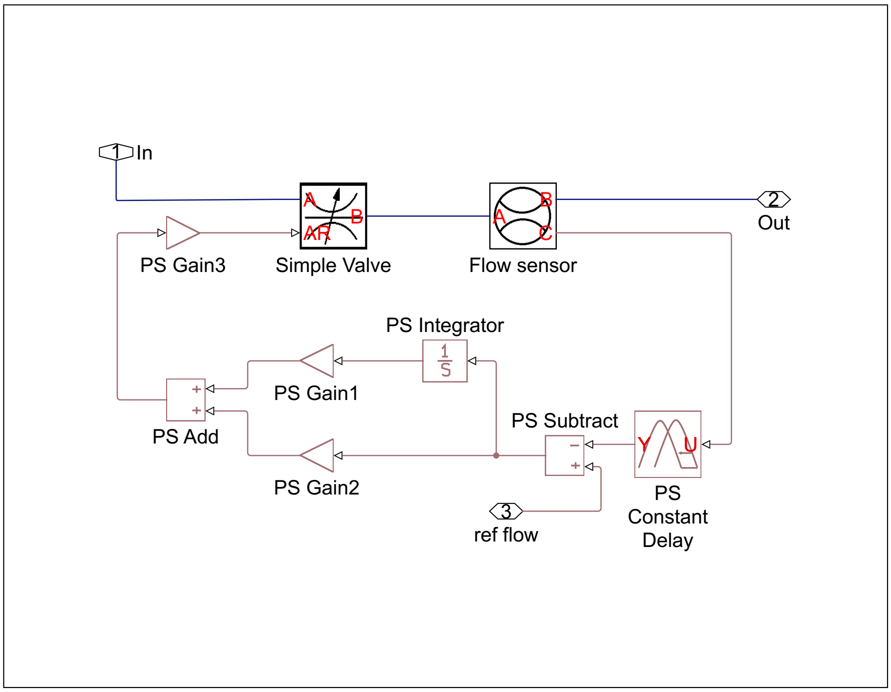


Figure 7.1 Constant flow valve

Product loop The product loop is constructed by placing an ideal pressure sensor in parallel to a controlled valve, which is regulated by a PI-controller. The reference pressure is passed from the supersystem, and a short delay is introduced to the pressure measurements to ensure numerical stability. This setup ensures that the differential pressure between the input node and drain node is equal to the reference pressure.

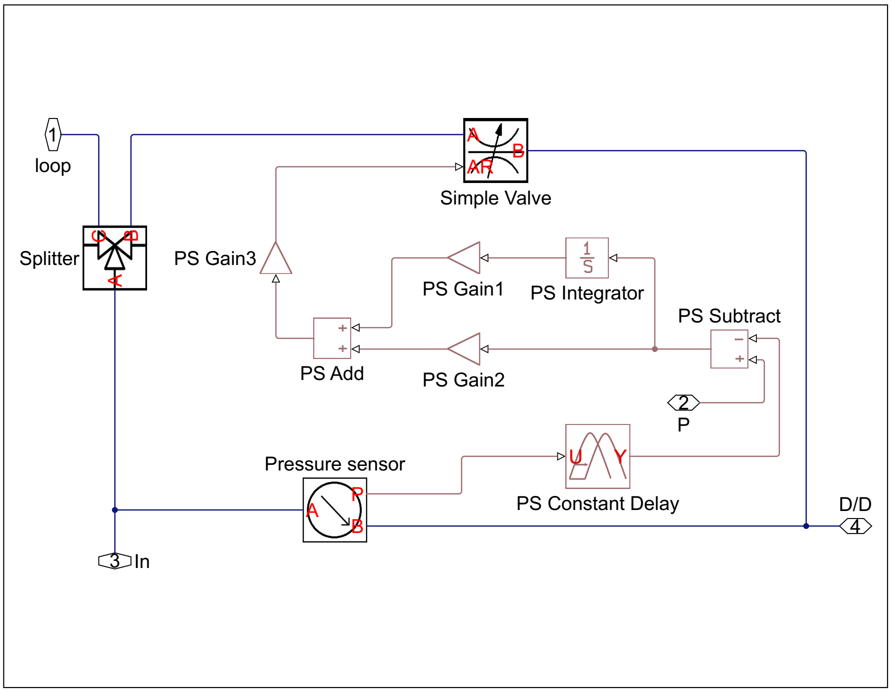


Figure 7.2 Product loop

Water saving The water saving subsystems consists of a splitter and a regulated valve as the hydraulic components. The total flow and the tank return flow are measured by ideal flow sensors, and a PI-controller ensures that the percentual flow to tank return is equal to the reference percentage, which is passed from the supersystem. A delay is introduced before the PI-controller to ensure numerical stability.

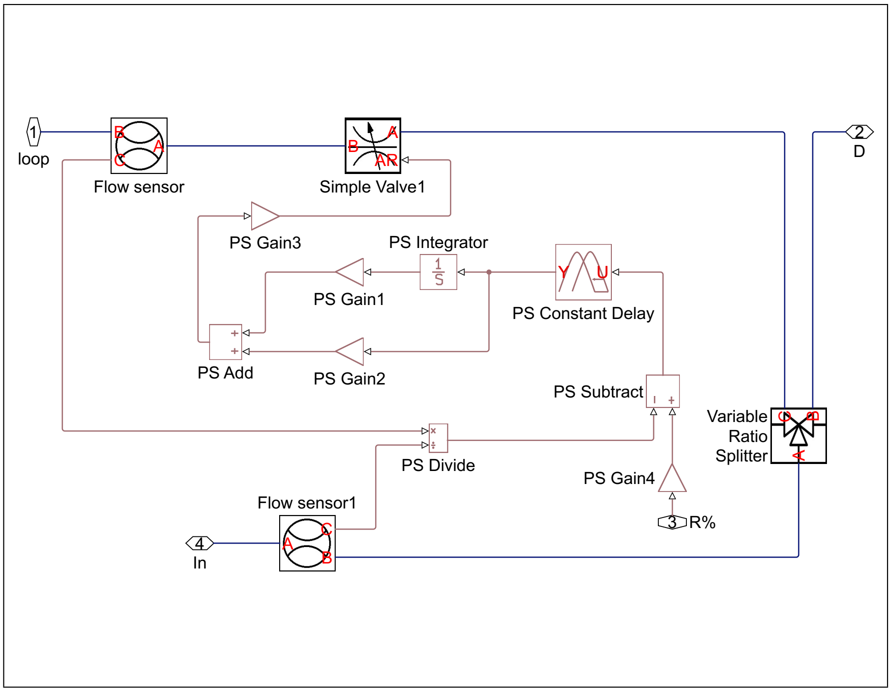


Figure 7.3 Water saving

Top level model

The Simscape model is built around the central loop between the water tank and the RO-membrane module. Both connections to drain are completed by using self-regulating subsystems that models the ideal behaviour of the respective connections. The RO pump is set to a fixed value and the valve in the central loop is tuned to ensure a conforming overall behaviour in the loop at the pressure region of interest.

Most connections are assumed to be frictionless and without pressure drops, but hydraulic resistive tubes have been included at certain points in the model to ensure that the flow behaviour conforms with the physical process. In particular, resistive tubes are placed before drainage points to avoid internal nodes with atmospheric pressure. The complete model layout is shown in Fig. 7.4.

Model layout

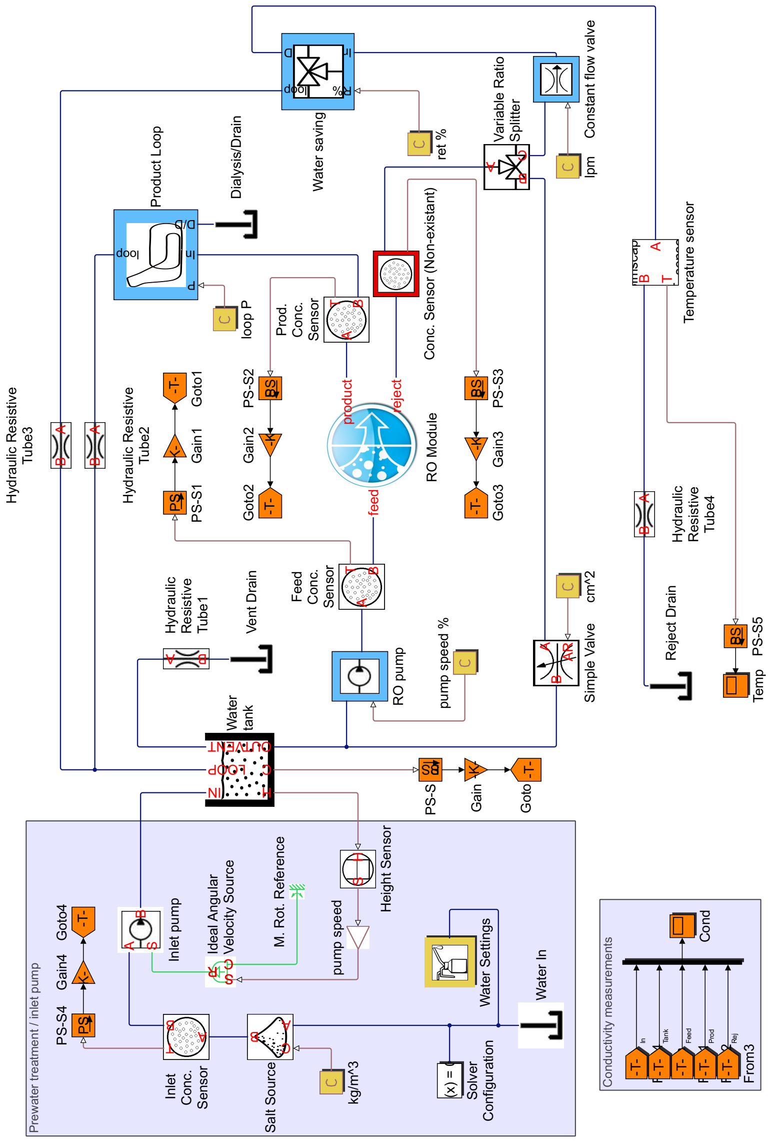


Figure 7.4 WRO 300 H model

7.4 Regulated model design

A regulated model design has been made where a subsystem represents a regulated version of the needle valve in the central loop, the controlled variable is the product flow from the membrane module. An ideal flow sensor measures this quantity and passes the value to regulator subsystem. The reference flow is set in the top-level system, and passed to the subsystem.

Subsystems

Regulated Needle Valve The regulated needle valve subsystem uses a simple valve which is regulated by a PI-controller. The measured product flow and reference flow is passed from the supersystem, and a delay is introduced to the measured product flow to ensure numerical stability. As this controlled valve is going to interact with the regulated valve in the product loop subsystem, it is important to choose the delay and time constants of this PI-controller so as to make it slower than the one in the product loop subsystem. This mitigates the risk of oscillating or unwanted behaviour.

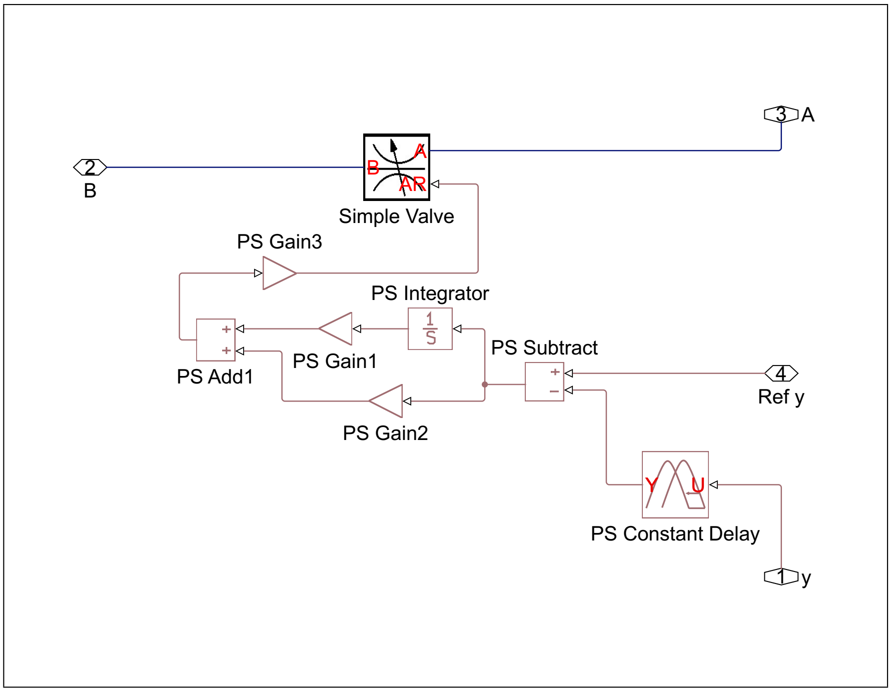


Figure 7.5 Regulated Needle Valve

Model layout

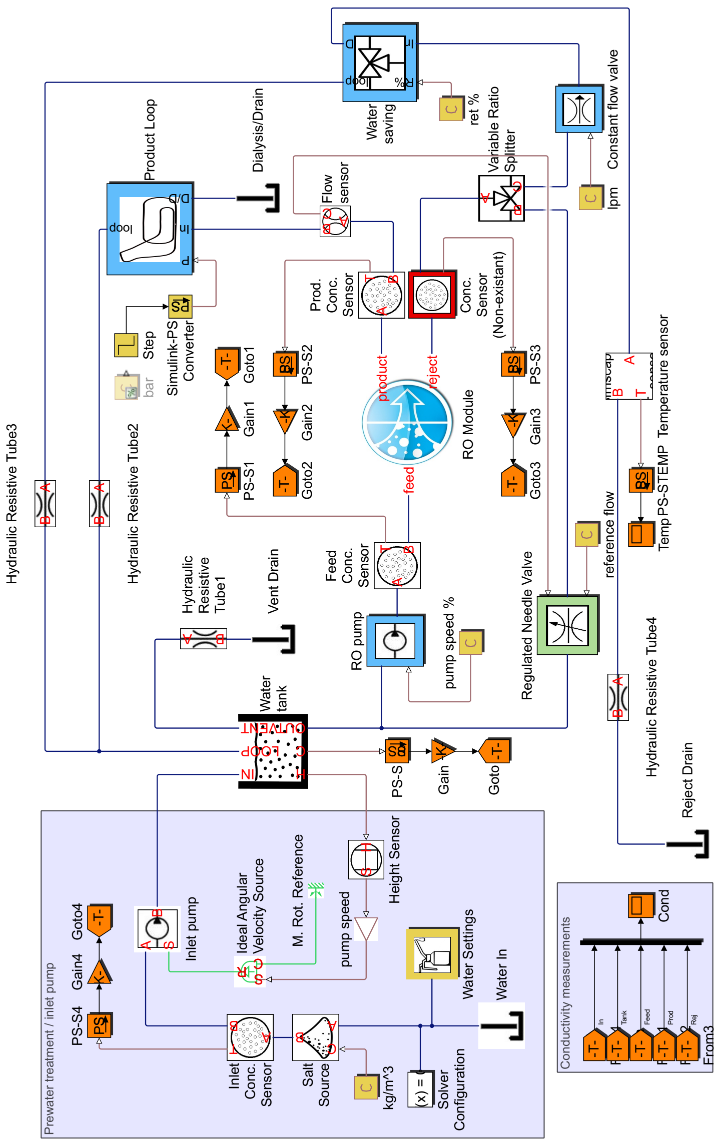


Figure 7.6 WRO 300 H with regulated needle valve

7.5 Modified model designs

Three additional models have been designed, where different ways of connecting multiple RO-membrane modules are tested. Two different series circuits have been designed, as well as a parallel circuit. All three designs use a single pump to pressurise the membrane modules. The reject flow series circuit uses a single reject loop valve, while the other designs uses two separate valves.

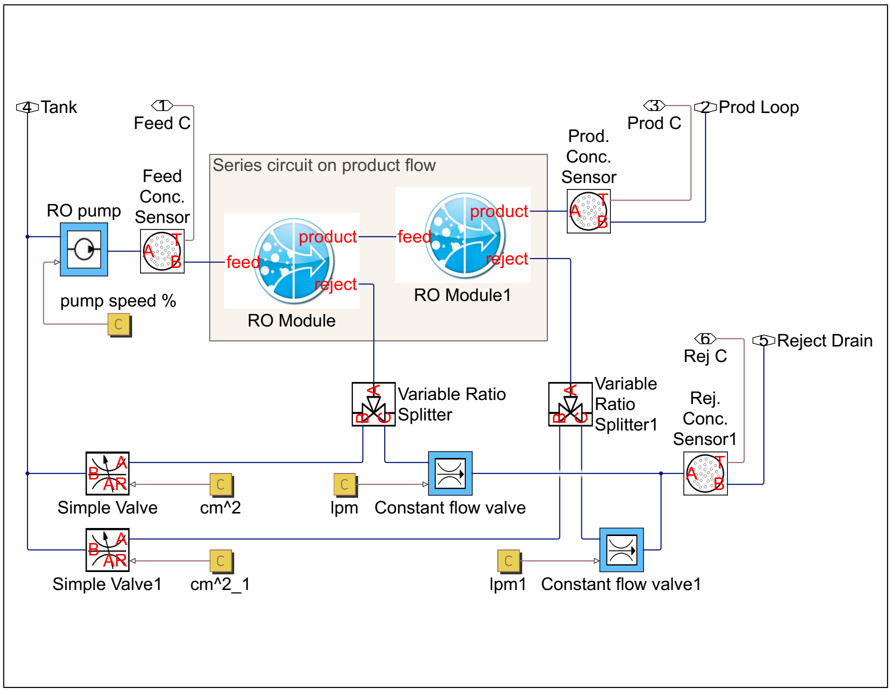


Figure 7.7 Series circuit central loop, as related to product flow

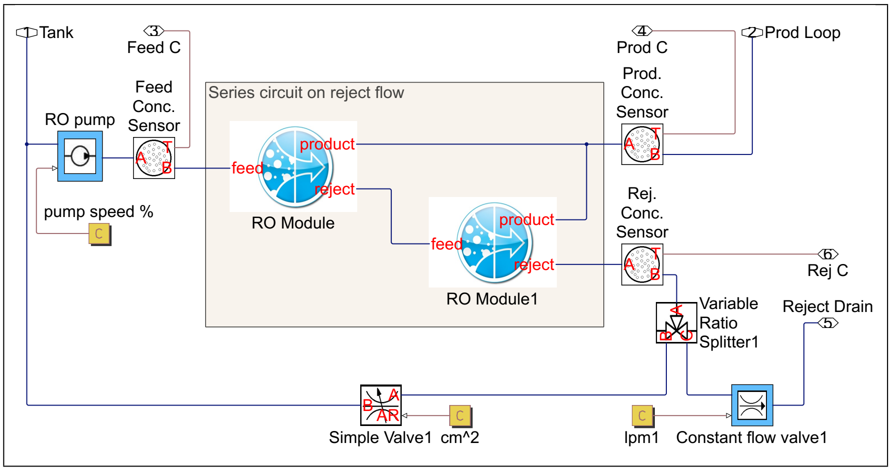


Figure 7.8 Series circuit central loop, as related to reject flow

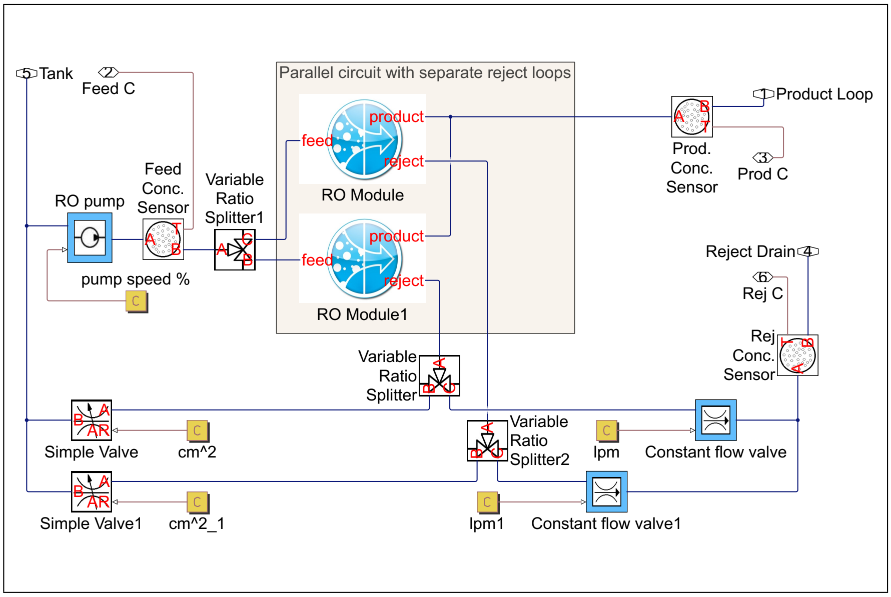


Figure 7.9 Parallel circuit central loop

8

Results

8.1 WRO 300 H model

Verification

A quick verification was made by comparing the mean product flow as a function of the net driving pressure, and the salt mass flow as a function of the concentration difference, to ensure that the model accurately replicates the estimated parameters.

As seen in Fig. 8.1, the model accurately replicates the mean product flow through the membrane, as represented by the mesaurements presented in Fig. 6.6(a).

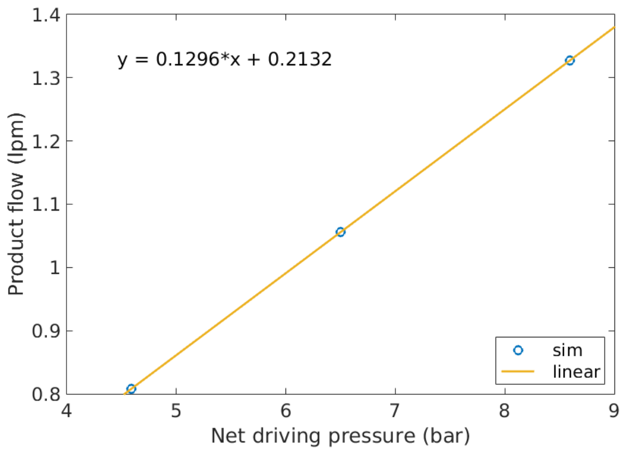


Figure 8.1 Linear estimation of the product flow as a function of the net driving pressure, with relative pressure loss `p_loss=0` in the membrane module.

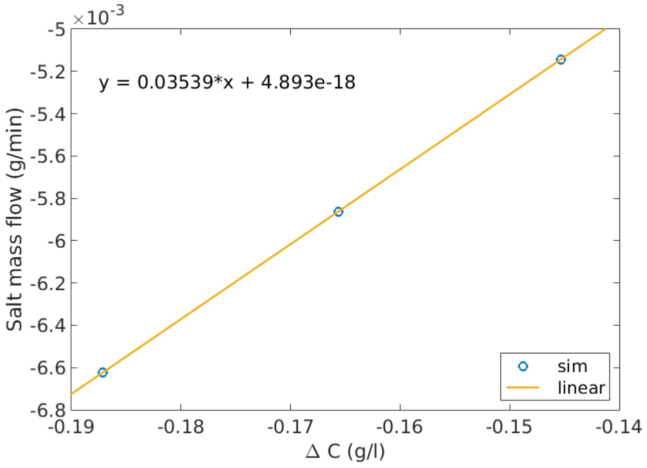


Figure 8.2 Linear estimation of the salt mass flow as a function of the concentration difference, with relative pressure loss `p_loss=0` in the membrane module.

As seen in Fig. 8.2, the model accurately replicates the mean salt mass flow through the membrane.

Oscillations

When simulating the model, an oscillating behaviour emerges, which is especially evident when observing the feed pressure, such as in Fig. 8.3.

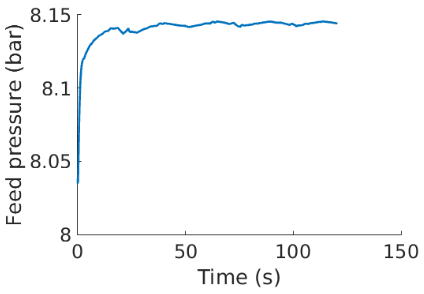
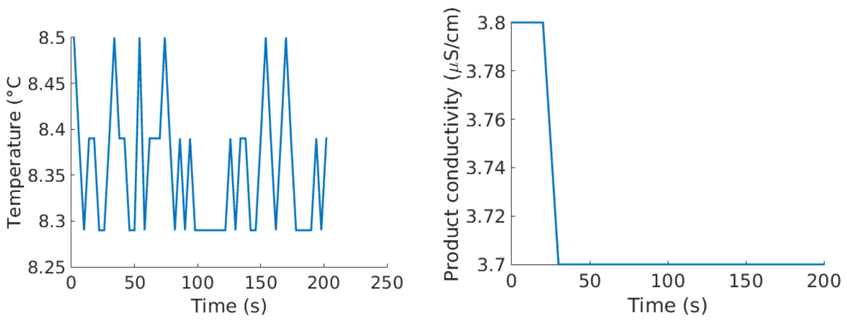


Figure 8.3 Feed pressure oscillations

Load disturbances

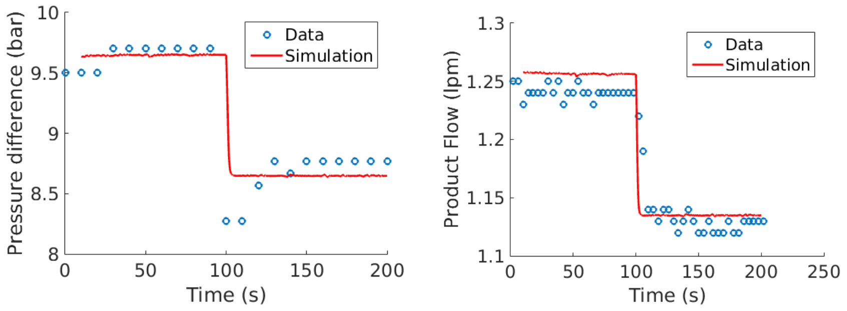
Several sets of load disturbance simulations were performed with the WRO 300 H machinery and simulated with the Simscape model, by changing the product loop counter-pressure. Overall, the temperature and product flow conductivity in the WRO were unaffected by the disturbance, as exemplified by Figs. 8.4(a) and 8.4(b).



(a) Feed water temperature (b) Product flow conductivity

Figure 8.4 Measured data, load disturbance at `t=100`.

The product flow is changed by a load disturbance to the product loop. In the first example, a step disturbance is introduced to the WRO 300 H machine, as well as the Simscape model at time `t=100`, when the product loop pressure is increased from 0.5 bar to 1.93 bar. The resulting change in pressure difference is shown in figure 8.5(a) and corresponding product flow change is shown in figure 8.5(b).



(a) Differential pressure (b) Product flow

Figure 8.5 Comparison between machine and simulated model, product loop pressure is changed at `t=100` from 0.5 bar to 1.93 bar.

The second example shows the reverse case when the product loop pressure drops. At time `t=100` the product loop pressure drops from 1.5 bar to 1.03 bar. The resulting change in pressure difference is shown in Fig. 8.6(a) and corresponding product flow change is shown in Fig. 8.6(b).

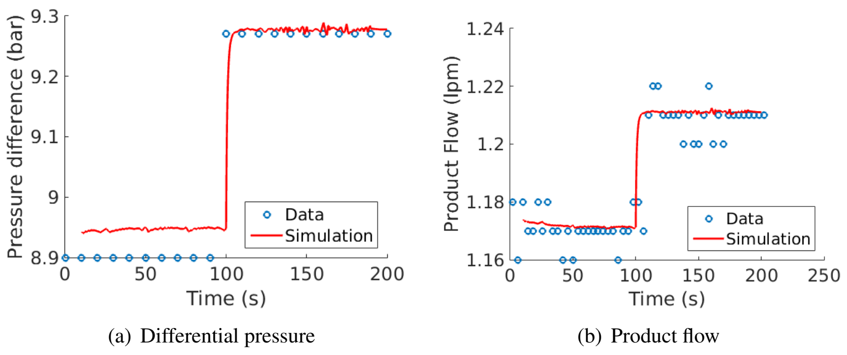


Figure 8.6 Comparison between machine and simulated model, product loop pressure is changed at `t=100` from 1.5 bar to 1.03 bar.

8.2 Regulated model

Load disturbance rejection

A load disturbance simulation was performed with the regulated model, where the product loop pressure is changed at time `t=50` from 1.5 bar to 1.0 bar, as seen in figure 8.7.

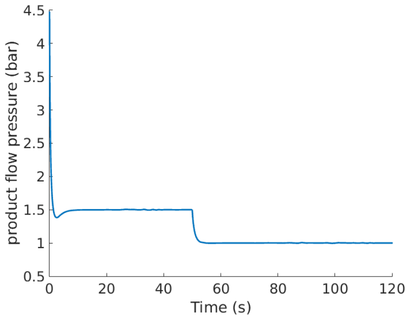
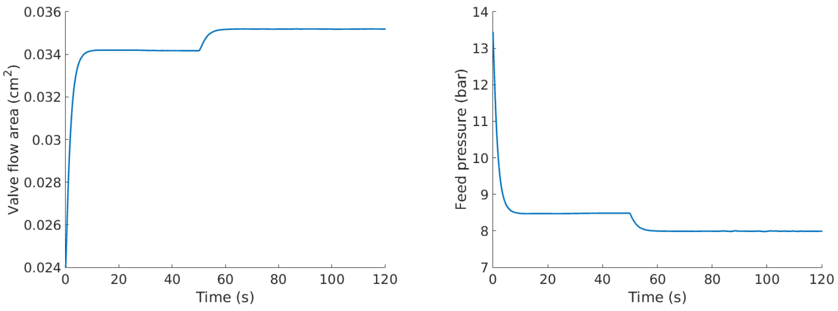


Figure 8.7 Pressure drop in the product loop.

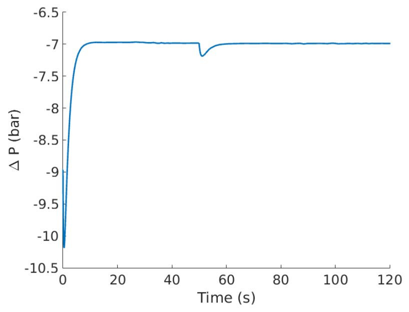
This caused the controller to change the area of the needle valve in accordance with Fig. 8.8(a), which caused the feed pressure drop shown in 8.8(b).



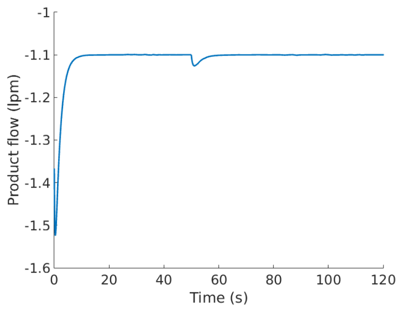
(a) Valve area (b) Feed pressure drop

Figure 8.8 Valve area regulation and corresponding feed pressure drop

The resulting hydraulic differential pressure is shown in Fig. 8.9(a), and the load disturbance is rejected by the controller as shown in Fig. 8.9(b).



(a) Hydraulic differential pressure



(b) Product flow

Figure 8.9 Resulting differential pressure and corresponding product flow

As shown in Fig. 8.10, the flow through the membrane is increased.

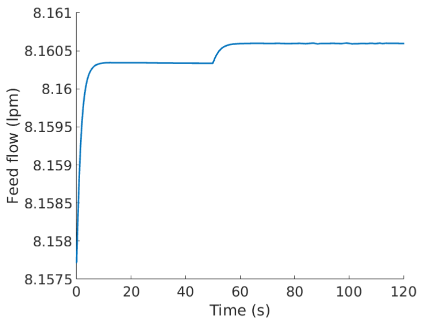
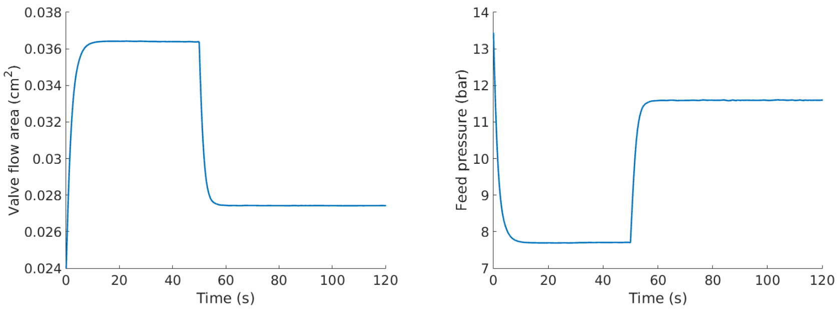


Figure 8.10 Resulting feed flow through the membrane

Reference value change

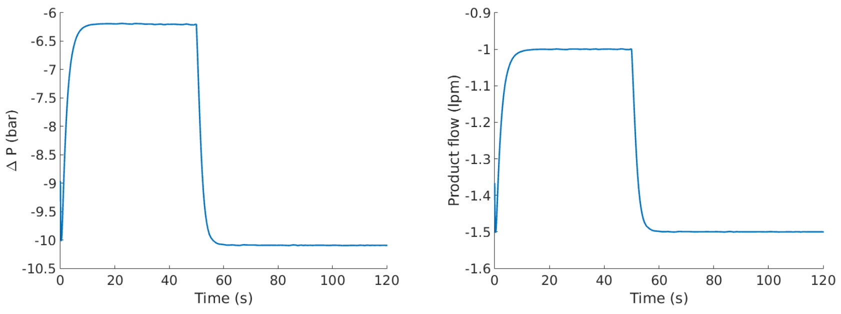
A reference value change simulation was performed with the regulated model, where the reference product flow is changed at time `t=50` from 1.0 lpm to 1.5 lpm. This caused the controller to change the area of the needle valve in accordance with Fig. 8.11(a), which caused the feed pressure drop shown in 8.11(b).



(a) Valve area (b) Feed pressure

Figure 8.11 Valve area regulation due to reference change and corresponding feed pressure increase

The resulting hydraulic differential pressure is shown in Fig. 8.12(a), and the reference flow change is accurately performed by the controller as shown in Fig. 8.12(b).



(a) Hydraulic differential pressure (b) Product flow

Figure 8.12 Resulting differential pressure and corresponding product flow

As shown in figure 8.10, the flow through the membrane is decreased when performing this positive reference flow change, as the valve area is diminished.

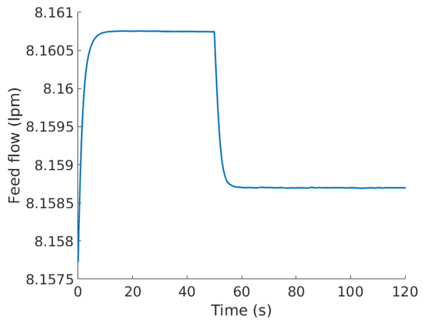


Figure 8.13 Resulting feed flow through the membrane

Oscillations

The oscillating behaviour of the WRO 300 H model has disappeared in the regulated Simscape model, as seen in Fig. 8.14.

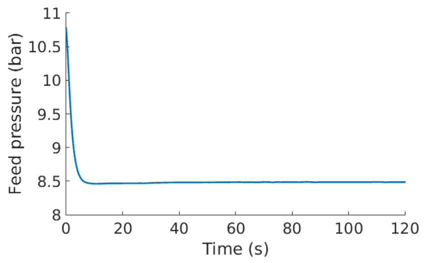


Figure 8.14 Feed pressure

Conductivities

The product flow conductivity is related to the rejection rate, which is pressure dependent, as seen in Eq. (2.20). When performing the load disturbance simulation, the product flow conductivity changes very little, from 10.65 µS/cm at time `t=50`, to 11.03 µS/cm at time `t=100`, as seen in Fig. 8.15. This is natural, as the differential pressure in the module returns to the original value after the load disturbance rejection, and the feed flow conductivity changes very little.

When performing the reference flow change, the conductivity change is somewhat more evident, from 11.21 µS/cm at time `t=50` to 9.163 µS/cm at time `t=100`, as seen in Fig. 8.16. The higher rejection rate caused by the increased differential pressure is mitigated by the fact that the feed flow conductivity also increases substantially.

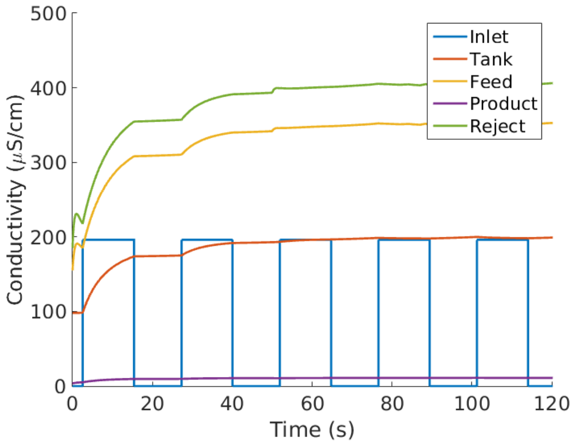


Figure 8.15 Conductivities of the system when performing load disturbance

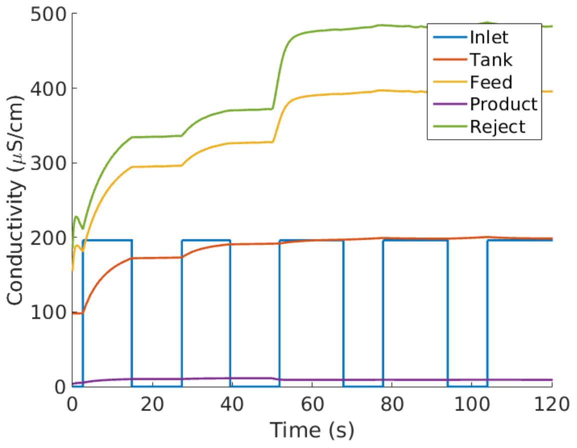


Figure 8.16 Conductivities of the system when performing reference flow change

8.3 Alternate models

Parallel circuit

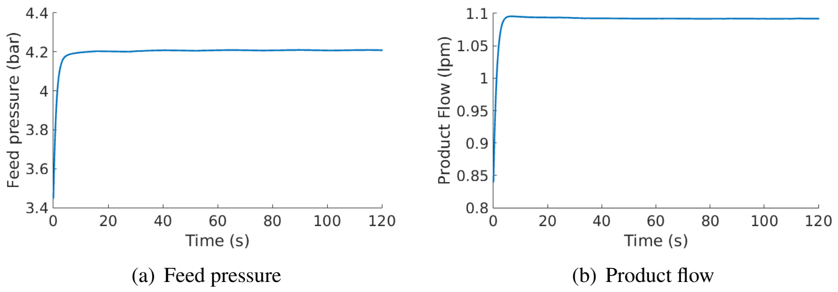


Figure 8.17 Feed pressure and product flow for a parallel circuit simulation

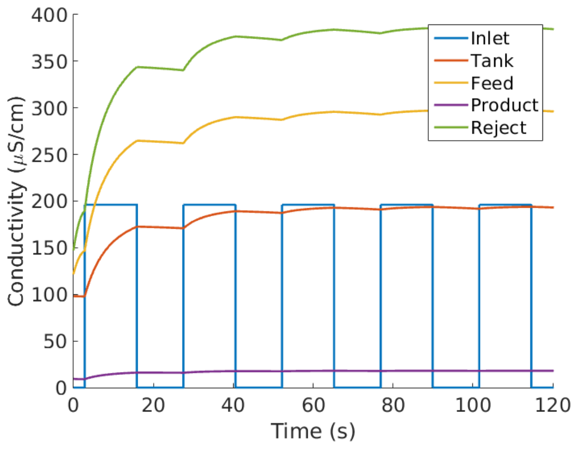
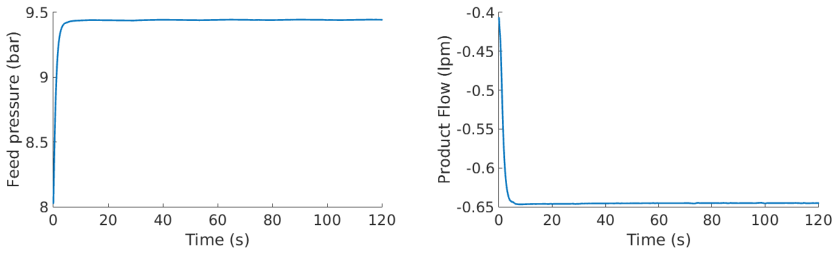


Figure 8.18 Conductivities for a parallel circuit simulation

By implementing a parallel circuit in the product loop, the product flow of the standard one-membrane model is achievable at a much lower pressure, as seen in Figs. 8.17(a) and 8.17(b). The conductivity of the product flow is drastically increased to about 17.9 µS/cm, as seen in 8.18. This can be explained by the fact that the rejection rate is pressure dependent, in accordance with Eq. (2.20)

Product flow series circuit simulation



(a) Feed pressure (b) Product flow

Figure 8.19 Feed pressure and product flow for a product flow series circuit simulation

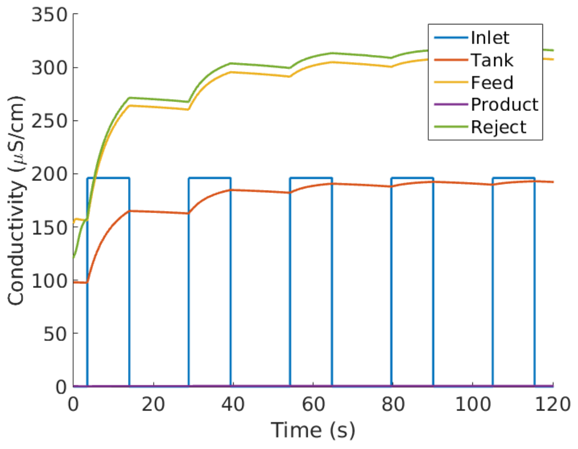
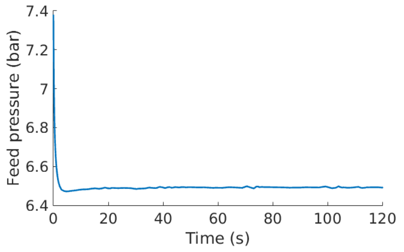


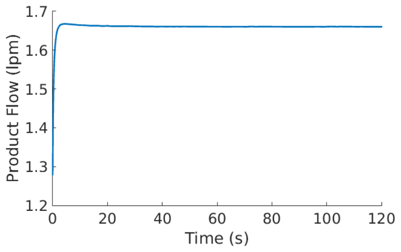
Figure 8.20 Conductivities for a product flow series circuit simulation

Using a series circuit in regards to the product flow, leads to a sharply reduced conductivity, about 0.69 µS/cm as seen in 8.20. The product flow is also significantly lowered to about 0.65 lpm while operating at similar pressure as the WRO 300 H, as seen in 8.19(a) and 8.19(b).

Reject flow series circuit simulation



(a) Feed pressure



(b) Product flow

Figure 8.21 Feed pressure and product flow for a reject flow series circuit simulation

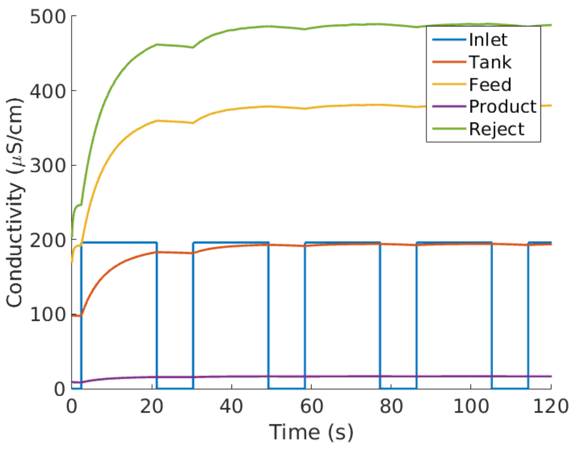


Figure 8.22 Conductivities for a reject flow series circuit simulation

While operating at a somewhat lower pressure than normal use of the WRO 300 H, there is a significantly higher product flow from a reject flow series circuit, as seen in 8.21(a) and 8.21(b). The conductivity is also increased to about 16.5 µS/cm as seen in 8.22.

9

Discussion

9.1 Membrane model

It was decided to base the identification process on an a priori mathematical model. Of several such models the linear Solution-Diffusion model was chosen. It was reasoned that one of the simpler a priori models would present several benefits over a more complex model or one based on more experimental identification. The decision to dismiss an experimental modeling technique, such as frequency domain modeling, was deemed wise after further acquaintance with the measurement conditions. The available sensors and measurement software was not designed to represent transients or anything other than low frequency dynamics in an accurate manner. While it certainly is possible to alter existing measurement hardware or software, such alterations would be both time-consuming and outside of the declared scope and goal of the thesis.

As justified by the measurement results, the linear model was found to be adequate for describing the behaviour of the membrane in the region of interest where the machine is operated. These results, coupled with issues of controllability of interesting state variables helped confirm the choice of modeling method.

The membrane model parameters have been implemented in the model in a manner meant to facilitate simulation of membranes with diverse parameter values. The simple Solution-Diffusion model implemented here bundles all material parameters into a single permeability constant, one for salt flows and one for water flows. These permeability constants are easily altered in the model.

The identification of the water permeability constant proved to be a straightforward task with clear results, as there is a similarity in results between the two modes of performing measurements. While a salt permeability of the membrane was estimated, the result cannot be considered as definitive as there were significant differences between the different modes of operation.

It should be expected that permeation qualities vary between individual membranes of the same model. To ensure the highest simulation quality the membrane model component should be calibrated to the module it is intended to emulate.

Possible amendments

When estimating the water permeability, the assumption is made that the membrane module pressure is equal throughout the module with no pressure drops. As the pressure is measured at the reject side of the membrane module, and the bias is positive as seen in Fig. 6.4(b), one should probably cast doubt on this assumption. If the average pressure is significantly higher than the measured reject pressure, this would offset the graph to the right, removing the positive bias.

Similarly, the conductivity is only measured before the membrane module, and the assumption is made that the concentration drop in the membrane module is not affected by the mode of controlling the module pressure. This assumption is doubtful as well, as using the pump to increase the module pressure will increase the flow through the module, while using the needle valve to increase the module pressure will lower the flow through it. This should affect the way the concentration drops inside the membrane module and could be the cause for the problems with replicating the salt permeability estimation for different modes of conducting measurements.

If these assumptions indeed are incorrect, this affects membrane module parameter estimations. A simple modeling of these phenomena would be to measure both pressure and conductivity at both the feed and reject endpoints, and using the average of these quantities to estimate the permeability parameters of the membrane. This would require further modifications to the lab setup and future work would probably benefit from improved measurements that better match the conditions of the theoretical model.

9.2 Simscape implementation

WRO 300 H model

The Simscape model correlates with the physical machinery in important aspects, and delivers the correct flow of product water with the correct conductivity when simulating the system. The load disturbance measurements and simulations deliver satisfactory results, as seen in Figs. 8.5(b) and 8.6(b).

The sharp oscillations of the model are most probably caused by some state being close to singularity, which is supported by the fact that they disappear in the regulated model. A small leakage flow should be introduced to the components that doesn’t already have it to account for this behaviour and maintain numerical integrity [Mathworks, 2017e].

The foremost drawback of the Simscape model is that it does not accurately represent the delays of the system when propagating the salt concentration through the system. While every node correctly propagates this instantly, the same should not be true to a non-accumulating component of known length, such as valves and piping. While several models of different complexity are available to represent this behaviour, a complete representation of salt diffusion in the piping of the system might not be needed if the flow rate is significantly higher than the salt transfer by diffusion. Instead, one might assume that the propagation of salt concentration is entirely related to the flow rate and longitudinal length of the component. This is not easily implemented in Simscape, as the delays need to be interpreted and reduced down to ordinary differential equations to be solved with the ODE solver used by Simscape. A way to implement this might be to use a number of discrete internal states that accumulates a salt masses and delivers this to the nodes in accordance with the delay of the component and direction of the flow.

Regulated Simscape model and modified models

Regulated model The regulated model design does regulate the product flow in a satisfactory manner. When performing load disturbances to the system, the controller quickly regulates the system to the correct product flow, as seen in Fig. 8.9(b), and reference flow changes are done efficiently as seen in Fig. 8.12(b)

While the controller correctly regulates the system in regards to product flow, one could imagine that it would be beneficial to create a multivariate controller that uses both the needle valve and the pump speed to regulate both the product water flow rate, as well as its conductivity.

Modified models While these designs cannot be verified without constructing a working prototype, the qualitative behaviour of these models should be accurate enough to draw basic conclusions about the behaviour of these designs, in particular when compared to each other or the WRO 300 H model.

The parallel circuit delivers a comparable flow to the WRO 300 H at a much lower module pressure, but since the rejection rate is directly related to the membrane pressure, this also increases the conductivity of the product flow drastically.

The product flow series circuit on the other hand delivers water with a far lower conductivity, which is to be expected.

The reject flow series circuit delivers a far higher product flow at somewhat lower pressure than normal for a WRO 300 H. The conductivity increases somewhat with the lower pressure, as is expected. This model simply represents an ideal membrane module that has doubled in area, without inflicting any internal pressure drops. A more advanced model of the membrane module pressure would probably give more detailed information about the actual consequences of such a setup.

10

Conclusion

A Simscape model was designed to simulate the behaviour of the Gambro WRO 300 H as related to the production of purified water. The model accurately simulates water flow across the membrane within a standard operating interval, while the simulation of salt flows and resulting conductivity is less accurate. This is caused by a large uncertainty in the identification process in regards to the permeability of salts across the membrane.

Several possible amendments to the model and membrane representation have been identified. Further work could improve the accuracy and simulation capacity of the model. The suitability of the Solution-Diffusion model in this application cannot be accurately determined without exploring these possibilities.

The constructed Simscape library is highly adaptable, and can be used to design and simulate variations on the existing machinery.

Bibliography

Abbas, A. (2006). “Model predictive control of a reverse osmosis desalination unit”. *Desalination* **194**, pp. 268–280.

Apps, T. (2013). *Measuring salinity, TDS and conductivity*. Apps Laboratories. Ranceby, Australia.

Baker, R. W. (2004). *Membrane Technology and Applications*. Wiley, Hoboken, New Jersey.

Cady, P., Dufour, S. W., Shaw, J., and Kraeger, S. J. (1978). “Electrical impedance measurements: rapid method for detecting and monitoring microorganisms”. *Journal of Clinical Microbiology* **7**, pp. 265–272.

Dow (2007). *FILMTEC™Reverse Osmosis Membranes Technical Manual*. Dow Liquid Separations. Midland, Michigan.

Gambier, A., Wellenreuther, A., and Badreddin, E. (2003). “Control system design of reverse osmosis plants by using advanced optimization techniques”. *Desalination and Water Treatment* **10**, pp. 200–209.

Gambro (2008). *WRO 300 H Service Manual*. Gambro Lundia AB. Lund, Sweden.

Gambro (2010). *WRO 300 H Operators Manual*. Gambro Lundia AB. Lund, Sweden.

Gordon, L. (2012). “Does model-based engineering make sense?” *Machine Design* **84**, p. 48.

Hoenich, N. and Ward, R. A. (2016). *Maintaining water quality for hemodialysis*. UpToDate. Waltham, Massachusetts.

Kucera, J. (2010). *Reverse Osmosis - Design, Processes, and Applications for Engineers*. Scrivener and Wiley, Hoboken, New Jersey.

Mathworks (2017a). *Basic Principles of Modeling Physical Networks*. The MathWorks. Natick, Massachusetts.

Mathworks (2017b). *Modeling Best Practices*. The MathWorks. Natick, Massachusetts.

Mathworks (2017c). *Important Concepts and Choices in Physical Simulation*. The MathWorks. Natick, Massachusetts.

Mathworks (2017d). *Declare Through and Across Variables for a Domain*. The MathWorks. Natick, Massachusetts.

Mathworks (2017e). *Simulating Hydraulic Models*. The MathWorks. Natick, Massachusetts.

Mejvik, S. and Olin, H. (2012). *Model Based Engineering of a Reverse Osmosis Water Purification Plant*. LUTFD2/TFRT–5903–SE. MA thesis. Lund University, Department of Automatic Control, Lund, Sweden.

Miller, S. (2010). *Modeling Physical Systems as Physical Networks with the Simscape Language*. The MathWorks. Natick, Massachusetts.

NIDDK (2017). *Hemodialysis*. National Institute of Diabetes, Digestive, and Kidney Diseases. Bethesda, Maryland.

Sobana, S. and Panda, R. C. (2011). “Identification, modelling, and control of continuous reverse osmosis desalination system: a review”. *Separation Science and Technology* **46**, pp. 551–560.

Sobana, S. and Panda, R. C. (2014). “Modeling and control of reverse osmosis desalination process using centralized and decentralized techniques”. *Desalination* **334**, pp. 243–251.

Wijmans, J. G. and Baker, R. W. (1995). “The solution-diffusion model: a review”. *Journal of Membrane Science* **107**, pp. 1–21.

Williams, M. E. (2009). *A Review of Reverse Osmosis Theory*. EET Corporation. Harriman, Tennessee.

A

Simscape source code

A.1 Domain

```
domain salt_dom
    % Saline hydraulic Domain
    % Defines the saline hydraulic domain.
    % Defines lookup tables and available node parameters

    parameters (Size=variable)
        % Default liquid property tables for water
        % Rows of the tables correspond to the temperature vector
        % Columns of the tables correspond to the pressure vector

        T_TLU = {[273.1600:10:373.16]', 'K'  }; % Temperature vector
        p_TLU = {[0.01, 0.1, 5:5:50],   'MPa'}; % Pressure vector

        rho_TLU = {[
999.8   999.8   1002.3   1004.8   1007.3   1009.7   1012.2   1014.5   1016.9   1019.2
1021.5  1023.8
999.7   999.7   1002.0   1004.4   1006.7   1009.0   1011.3   1013.5   1015.8   1018.0
1020.2  1022.3
998.2   998.2   1000.4   1002.7   1004.9   1007.1   1009.3   1011.5   1013.6   1015.7
1017.8  1019.9
995.6   995.6   997.8    1000.0   1002.2   1004.3   1006.5   1008.6   1010.7   1012.7
1014.8  1016.8
992.2   992.2   994.4    996.5    998.6    1000.8   1002.9   1004.9   1007.0   1009.0
1011.0  1013.0
988.0   988.0   990.2    992.3    994.4    996.5    998.6    1000.7   1002.7   1004.7
1006.7  1008.7
983.2   983.2   985.3    987.5    989.6    991.7    993.8    995.8    997.9    999.9
1001.9  1003.9
977.8   977.8   979.9    982.1    984.2    986.3    988.4    990.5    992.5    994.6
996.6   998.6
971.8   971.8   974.0    976.2    978.3    980.5    982.6    984.7    986.8    988.8
990.9   992.9
965.3   965.3   967.5    969.8    972.0    974.2    976.3    978.5    980.6    982.7
984.7   986.8
958.8   958.8   960.6    962.9    965.2    967.4    969.6    971.8    974.0    976.1
978.2   980.3
        ], 'kg/m^3'}; % Density table
        nu_TLU = {[
1.7917  1.7914  1.7763  1.7615  1.7473  1.7338  1.7208  1.7085  1.6967  1.6854
1.6747  1.6645
1.3061  1.3059  1.2986  1.2914  1.2845  1.2779  1.2716  1.2656  1.2598  1.2543
1.2491  1.2441
1.0032  1.0032  0.9995  0.9959  0.9924  0.9891  0.9859  0.9829  0.9801  0.9774
0.9748  0.9724
0.8006  0.8005  0.7987  0.7970  0.7953  0.7937  0.7922  0.7908  0.7894  0.7881
0.7870  0.7859
0.6577  0.6577  0.6570  0.6562  0.6555  0.6548  0.6543  0.6537  0.6532  0.6528
0.6523  0.6520
0.5530  0.5530  0.5529  0.5527  0.5526  0.5525  0.5524  0.5524  0.5524  0.5524
0.5524  0.5525
0.4739  0.4739  0.4741  0.4743  0.4745  0.4747  0.4750  0.4753  0.4755  0.4758
0.4762  0.4765
0.4127  0.4127  0.4131  0.4135  0.4139  0.4143  0.4148  0.4152  0.4157  0.4162
0.4167  0.4172
0.3643  0.3643  0.3648  0.3654  0.3659  0.3665  0.3670  0.3676  0.3682  0.3688
0.3694  0.3700
0.3254  0.3254  0.3261  0.3267  0.3273  0.3280  0.3286  0.3293  0.3299  0.3306
0.3312  0.3319
0.2866  0.2866  0.2945  0.2952  0.2959  0.2965  0.2972  0.2979  0.2986  0.2993
0.3000  0.3007
        ], 'mm^2/s'}; % Kinematic viscosity table

        beta_TLU = {[
1.9649  1.9654  1.9929  2.0213  2.0499  2.0787  2.1078  2.1372  2.1668  2.1966
2.2268  2.2572
2.0913  2.0918  2.1191  2.1471  2.1752  2.2036  2.2322  2.2610  2.2899  2.3191
2.3485  2.3781
2.1786  2.1791  2.2067  2.2350  2.2634  2.2919  2.3206  2.3494  2.3784  2.4075
2.4367  2.4661
2.2332  2.2337  2.2619  2.2908  2.3197  2.3487  2.3778  2.4070  2.4363  2.4657
2.4951  2.5246
2.2600  2.2605  2.2895  2.3191  2.3487  2.3784  2.4081  2.4378  2.4675  2.4973
2.5272  2.5571
2.2638  2.2638  2.2936  2.3240  2.3544  2.3848  2.4151  2.4455  2.4758  2.5062
2.5365  2.5669
2.2472  2.2472  2.2779  2.3092  2.3403  2.3715  2.4025  2.4336  2.4646  2.4955
2.5264  2.5573
2.2140  2.2140  2.2455  2.2776  2.3096  2.3415  2.3733  2.4050  2.4367  2.4683
2.4998  2.5313
2.1668  2.1668  2.1992  2.2321  2.2649  2.2976  2.3301  2.3626  2.3949  2.4272
2.4593  2.4914
2.1081  2.1081  2.1413  2.1750  2.2086  2.2421  2.2754  2.3085  2.3415  2.3744
2.4072  2.4399
2.0494  2.0494  2.0739  2.1084  2.1428  2.1770  2.2110  2.2449  2.2786  2.3121
2.3455  2.3788
        ], 'GPa'}; % Isothermal bulk modulus table
        alpha_TLU = {1e-4 * [
-0.6790  -0.6760  -0.4900  -0.3070  -0.1300  0.0410  0.2060  0.3650  0.5190
0.6680  0.8110  0.9490
 0.8780   0.8810   0.9980   1.1140   1.2270  1.3370  1.4450  1.5490  1.6510
1.7500  1.8460  1.9390
 2.0680   2.0690   2.1410   2.2130   2.2830  2.3520  2.4200  2.4860  2.5510
2.6140  2.6760  2.7360
 3.0340   3.0350   3.0740   3.1130   3.1520  3.1910  3.2290  3.2670  3.3050
3.3410  3.3770  3.4130
 3.8550   3.8560   3.8700   3.8840   3.8990  3.9140  3.9300  3.9460  3.9610
3.9770  3.9930  4.0090
 4.5780   4.5780   4.5720   4.5660   4.5620  4.5580  4.5550  4.5520  4.5500
4.5490  4.5480  4.5470
 5.2330   5.2330   5.2090   5.1860   5.1650  5.1440  5.1250  5.1070  5.0900
5.0740  5.0590  5.0450
 5.8400   5.8400   5.8010   5.7620   5.7250  5.6910  5.6570  5.6250  5.5950
5.5660  5.5380  5.5120
 6.4140   6.4140   6.3600   6.3070   6.2560  6.2080  6.1610  6.1170  6.0740
6.0330  5.9940  5.9560
 6.9670   6.9670   6.8980   6.8310   6.7660  6.7050  6.6460  6.5890  6.5350
6.4820  6.4320  6.3840
 7.5190   7.5190   7.4230   7.3420   7.2640  7.1890  7.1180  7.0490  6.9830
6.9200  6.8590  6.8000
        ], '1/K'}; % Isobaric thermal expansion coefficient table
    end

    parameters
        T_min            = {273.16,   'K'      }; % Minimum valid temperature
        T_max            = {373.16,   'K'      }; % Maximum valid temperature
        p_min            = {0.01,     'MPa'    }; % Minimum valid pressure
        p_max            = {50,       'MPa'    }; % Maximum valid pressure
        p_atm            = {0.101325, 'MPa'    }; % Atmospheric pressure
        T                = {290,      'K'      }; % System temperature
    end

    variables
        p = { 0 , 'Pa' }; % Pressure
    end

    variables(Balancing = true)
        q = { 1e-3 , 'm^3/s' }; % Flow rate
        msalt = {1e-3, 'kg/s'}; % Salt mass rate
    end
end
```

A.2 Components

Abstract components

Branch

```
component(Hidden=true) branch
    % Saline Hydraulic Branch
    % Defines a hydraulic branch with external saline hydraulic conserving
    % ports A and B. Also defines associated through and across variables q
    % and p. Accesses temperature and atmospheric pressure from node A and
    % transfers them to node B.

    nodes
        A = saltdomain.salt_dom; % A:left
        B = saltdomain.salt_dom; % B:right
    end

    variables
        q       = {1e-3, 'm^3/s'};  % Liquid volume flow
        p       = {0, 'Pa'};        % Pressure differential
        msalt   = {1e-3, 'kg/s'};   % Salt mass flow
    end

    function setup
        % ACCESS TEMPERATURE AND ATMOSPHERIC PRESSURE FROM NODE
        T       = A.T;
        p_atm   = A.p_atm;

        % TEMPERATURE AND ATMOSPHERIC PRESSURE PROPOGATION
        B.T == T;
        B.p_atm == p_atm;
    end

    branches
        q : A.q -> B.q;                 % Define flow q from A to B
        msalt : A.msalt -> B.msalt; % Define flow msalt from A to B
    end

    equations
        p == A.p - B.p;                 % Define pressure p between A and B
    end
end
```

Three Way Branch

```
component (Hidden=true) branch_threeway
    % Saline Hydraulic Three Way Branch
    % Defines a hydraulic branch with external hydraulic-conserving
    % ports A, B and C. Also defines associated through and across
    % variables q and p between all nodes. Accesses temperature and
    % atmospheric pressure from node A and transfers them to node B
    % and C.

    nodes
        A = saltdomain.salt_dom; % A:bottom
        B = saltdomain.salt_dom; % B:top
        C = saltdomain.salt_dom; % C:top
    end

    variables
        % Liquid volume flows
        q_A = { 1e-3 , 'm^3/s' };
        q_B = { 1e-3 , 'm^3/s' };
        q_C = { 1e-3 , 'm^3/s' };

        % Salt mass flows
        msalt_A = {1e-3, 'kg/s'};
        msalt_B = {1e-3, 'kg/s'};
        msalt_C = {1e-3, 'kg/s'};
    end

    function setup
        % ACCESS TEMPERATURE AND ATMOSPHERIC PRESSURE FROM NODE
        T       = A.T;
        p_atm   = A.p_atm;

        % TEMPERATURE AND ATMOSPHERIC PRESSURE PROPOGATION
        B.T == T;
        C.T == T;

        B.p_atm == p_atm;
        C.p_atm == p_atm;
    end

    branches
        % Define liquid flows q_X from external node X to internal node
        q_A : A.q -> *;
        q_B : B.q -> *;
        q_C : C.q -> *;

        % Define salt flows msalt_X from external node X to internal node
        msalt_A : A.msalt -> *;
        msalt_B : B.msalt -> *;
        msalt_C : C.msalt -> *;
    end



    equations
        q_A+q_B+q_C == 0;                % Liquid volume flow equilibrium
        msalt_A+msalt_B+msalt_C == 0;    % Salt mass flow equilibrium
    end
end
```

Two Port Steady

```
component (Hidden=true) two_port_steady
    % Two Port Component without Liquid Volume (Saline Hydraulic)
    % This base component defines a saline water block with two ports.

    nodes
        A = saltdomain.salt_dom; % A:bottom
        B = saltdomain.salt_dom; % B:top
    end

    variables (Access=protected)
        % Through variables
        q_A = {1, 'm^3/s'}; % flow rate into port A
        q_B = {1, 'm^3/s'}; % flow rate into port B
    end

    function setup
        % ACCESS TEMPERATURE AND ATMOSPHERIC PRESSURE FROM NODE
        T       = A.T;
        p_atm   = A.p_atm;

        % TEMPERATURE AND ATMOSPHERIC PRESSURE PROPOGATION
        B.T == T;
        B.p_atm == p_atm;
    end

    branches
        % Defines liquid volume flows q_X from X to internal node
        q_A : A.q -> *;
        q_B : B.q -> *;
    end
end
```

Hydraulic Components

RO module

```
component RO_module
    % RO Module
    % This block models a reverse osmosis process, with continuous flow
    % from FEED to REJ on one side of the membrane, and an output PROD of
    % purified water on the other side. The component is based on the
    % solution-diffusion model of reverse osmosis, with a constant osmosis
    % factor in the calculation of the osmotic pressure.
    % Furthermore, a constant has been added to the permeability of water
    % through the RO-membrane to give better accuracy in the region of use,
    % while also limiting the interval for which the model is appropriate.

    nodes
        FEED = saltdomain.salt_dom; % feed:left
        PROD = saltdomain.salt_dom; % product:right
        REJ = saltdomain.salt_dom;  % reject:right
    end

    parameters
        % Reverse osmosis consants
        Kw = {129.6394e-3, 'l/(bar*min)'};
        Kw_konst = {213.1505e-3, 'l/min'};
        Ks = {0.035392, 'l/min'};

        % RO-module parameters
        a_feed = {0.01, 'm^2'};      % feed inlet area
        a_rej= {0.01, 'm^2'};        % reject outlet area
        a_prod= {0.01, 'm^2'};       % product outlet area
        vol_tank = {0.005, 'm^3'};   % tank capacity
        p_loss = {0.01, '1'} ;       % reject pressure loss ratio

    end

    parameters(Access=protected)
        Nsalt = {0.058, 'kg/mol'};          % molar mass of salt
        kosm = {1.19, 'psi*kg/(K*mol)'};    % constant osmosis
        P_amb = {0, 'Pa'};                  % relative ambient pressure
        rho = {1e3, 'kg/m^3'};              % density
        T = {293, 'K'};                     % water temperature
        p_atm = {0.101325, 'MPa'};          % atmospheric pressure
    end

    variables
        q_feed = {0.1, 'm^3/s'};     % water flow (port A)
        q_prod = {0.1, 'm^3/s'};     % water flow (port B)
        q_rej = {0.1, 'm^3/s'};      % water flow (port C)

        msalt_feed = {0.001, 'kg/s'};    % salt mass flow (port A)
        msalt_prod = {0.0001, 'kg/s'};   % salt mass flow (port B)
        msalt_rej = {0.0009, 'kg/s'};    % salt mass flow (port C)

        deltaC = {0.001, 'kg/m^3'}; % salt concentration differential
        deltaPosm = {0.1e5, 'Pa'};  % osmotic pressure differential
        deltaP = {0.1e5, 'Pa'};     % pressure differential
    end
```

```
branches
    % Liquid volume flows q_X from X to internal node
    q_feed: FEED.q -> *;
    q_prod: PROD.q -> *;
    q_rej: REJ.q -> *;

    % Salt mass flows msalt_X from X to internal node
    msalt_feed: FEED.msalt -> *;
    msalt_prod: PROD.msalt -> *;
    msalt_rej: REJ.msalt -> *;
end

function setup
    % ACCESS TEMPERATURE AND ATMOSPHERIC PRESSURE FROM NODE
    T       = FEED.T;
    p_atm   = FEED.p_atm;

    % TEMPERATURE AND ATMOSPHERIC PRESSURE PROPOGATION
    PROD.T == T;
    REJ.T == T;

    PROD.p_atm == p_atm;
    REJ.p_atm == p_atm;
end

equations
    let
        % Across variables
        p_feed = FEED.p;
        p_prod = PROD.p;
        p_rej = REJ.p;

        % ACCESS WATER PARAMETERS FROM DOMAIN
        T_TLU   = FEED.T_TLU;
        p_TLU   = FEED.p_TLU;
        rho_TLU = FEED.rho_TLU;

        rho = tablelookup(T_TLU, p_TLU, rho_TLU, T, p_prod+p_atm, ...
            interpolation=linear, extrapolation=linear);
        in
            % Pressure loss from inlet to drain
            p_rej == p_feed*(1-p_loss);

            % Flow equation, conservation of water flow rates
            q_feed + q_prod + q_rej == 0;

            % Conservation of salt flow rates
            msalt_feed + msalt_prod + msalt_rej == 0;

            % Concentration gradient across membrane
            if (q_feed>0 && q_prod<0)
                deltaC == ((msalt_prod/q_prod)-(msalt_feed/q_feed));
            else
                deltaC == 0;
            end

            % Calculation of pressure and osmotic pressure
            deltaP == p_prod-(p_feed+p_rej)/2;
            % Two ions for each molecule of salt
            deltaPosm == 2*kosm*T*deltaC/(rho*Nsalt);

            %Water through membrane
            q_prod == (deltaP-deltaPosm)*Kw-Kw_konst;

            %Salt through membrane
            msalt_prod == deltaC*Ks;
        end
    end
end
```

Fixed Displacement Pump

```
component fixed_displacement_pump < saltdomain.two_port_steady
    % Fixed-Displacement Pump
    % This block models a simple fixed-displacement hydraulic pump in a saline
    % liquid network. Pump leakeage is set by the volumetric efficiency.
    % Mechanical losses in the drive shaft are modeled by applying a friction
    % torque proportional to the pressure difference. The pump does not
    % exchange heat with the surroundings. Port S is the mechanical rotational
    % conserving port representing the drive shaft. Positive rotation results
    % in liquid flowing from port A to port B.

    % Extended and adapted from the corresponding thermal liquid
    % fixed-displacement pump designed by Mathworks.
    % https://se.mathworks.com/help/physmod/hydro/ref/fixeddisplacementpumptl.html

    nodes
        S = foundation.mechanical.rotational.rotational; % S:bottom
    end

    parameters
        % Pump volume displacement
        % Volumetric efficiency at nominal conditions
        % Nominal shaft angular velocity
        % Nominal liquid pressure
        % Nominal liquid density
        % Nominal liquid kinematic viscosity
        % No-load shaft friction torque
        % Friction torque proportionality constant
        % Cross-sectional area at ports A and B
        % Characteristic longitudinal length

        volume_displacement   = {6e-6, 'm^3/rad'};
        volumetric_efficiency = {0.92, '1'      };
        omega_nominal         = {188,  'rad/s'  };
        p_nominal             = {10,   'MPa'    };
        rho_nominal           = {1000, 'kg/m^3' };
        nu_nominal            = {0.8,  'cSt'    };
        no_load_torque        = {0.05, 'N*m'    };
        K_friction            = {6e-7, 'N*m/Pa' };
        area                  = {0.01, 'm^2'    };
        length                = {0.1,  'm'      };
    end

    parameters (Access=protected, Hidden=true)
        % Hagen-Poiseuille coefficient
        % Angular velocity friction threshold
        % System temperature
        % Atmospheric pressure

        K_leakage       = {8e-14,    'm^3'   };
        omega_threshold = {0.01,     'rad/s' };
        T               = {290,      'K'     };
        p_atm           = {0.101325, 'MPa'   };
    end

    variables (Access=protected)
        torque = {0, 'N*m'}; % Shaft torque
        msalt = {1e-3, 'kg/s'}; %salt mass flow through pump
    end
    function setup
        % Check parameter range
        if volume_displacement <= 0
            pm_error('simscape:GreaterThanZero', 'Pump volume displacement')
        end
        if volumetric_efficiency <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Volumetric efficiency at nominal conditions')
        end
        if volumetric_efficiency > 1
            pm_error('simscape:LessThan', ...
                'Volumetric efficiency at nominal conditions', '1')
        end
        if omega_nominal <= 0
            pm_error('simscape:GreaterThanZero', 'Nominal shaft angular velocity')
        end
        if p_nominal <= 0
            pm_error('simscape:GreaterThanZero', 'Nominal liquid pressure')
        end
        if rho_nominal <= 0
            pm_error('simscape:GreaterThanZero', 'Nominal liquid density')
        end
        if nu_nominal <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Nominal liquid kinematic viscosity')
        end
        if no_load_torque < 0
            pm_error('simscape:GreaterThanOrEqualToZero', ...
                'No-load shaft friction torque')
        end
        if K_friction <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Shaft friction torque versus pressure proportionality constant')
        end
        if area <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Cross-sectional area at ports A and B')
        end
        if length <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Characteristic longitudinal length')
        end

        % Hagen-Poiseuelle leakage coefficient at nominal operating conditions.
        K_leakage = volume_displacement * omega_nominal * ...
            (1 - volumetric_efficiency) * nu_nominal * rho_nominal / p_nominal;
    end

    branches
        torque: S.t -> *;            % Define torque from S to internal node
        msalt : A.msalt -> B.msalt; % Define salt flow from A to B
    end

    equations
        let
            % ACCESS WATER PARAMETERS FROM DOMAIN
            T_TLU   = A.T_TLU;
            p_TLU   = A.p_TLU;
            rho_TLU = A.rho_TLU;
            nu_TLU = A.nu_TLU;
            % Across variables
            p_A   = A.p;
            p_B   = B.p;
            omega = S.w;

            nu  = tablelookup(T_TLU, p_TLU, nu_TLU, T, p_B+p_atm, ...
                interpolation=linear, extrapolation=linear);
            rho = tablelookup(T_TLU, p_TLU, rho_TLU, T, p_B+p_atm, ...
                interpolation=linear, extrapolation=linear);

            % Liquid volume flow from port A to port B
            q = q_A;

            % Leakage flow
            leakage = K_leakage * (p_B - p_A) / (nu*rho);

            % Shaft friction
            delta_p_abs = if ge(p_B, p_A), p_B - p_A else p_A - p_B end;
            friction_torque = no_load_torque + K_friction * delta_p_abs;
        in
            % Liquid volume flow through the pump
            q == volume_displacement * omega - leakage;

            % Torque balance
            torque == (p_B - p_A) * volume_displacement + friction_torque ...
                * tanh(4*omega/omega_threshold);

            % Mass balance
            q_A + q_B == 0;
        end
    end
end
```

Resistive Tube

```
component resistive_tube < saltdomain.branch
    % Hydraulic Resistive Tube
    % This block models hydraulic pipelines with circular and
    % noncircular cross sections and accounts for resistive property only.
    % To account for local resistances such as bends, fittings, inlet and
    % outlet losses, and so on, all the resistances are converted into their
    % equivalent lengths, and then the total length of all the resistances is
    % added to the pipe geometrical length.
    %
    % Connections A and B are hydraulic conserving ports. The block positive
    % direction is from port A to port B. This means that the flow rate is
    % positive if fluid flows from A to B, and the pressure loss is determined
    % as p = p_A - p_B.

    % Extended from the corresponding isothermal resistive tube designed by mathworks.
    % https://se.mathworks.com/help/physmod/simscape/ref/pipetl.html

    parameters
         % Tube cross section type
        % Tube internal diameter
        % Noncircular tube cross-sectional area
        % Noncircular tube hydraulic diameter
        % Geometrical shape factor
        % Tube length
        % Aggregate equivalent length of local resistances
        % Internal surface roughness height
        % Laminar flow upper margin
        % Turbulent flow lower margin

        cs_type   = { 1,         '1'     };
        d_in      = { 0.01,      'm'     };
        area      = { 1e-4,      'm^2'   };
        D_h       = { 1.12e-2,   'm'     };
        s_factor  = { 64,        '1'     };
        length    = { 5,         'm'     };
        length_ad = { 1,         'm'     };
        roughness = { 15e-6,     'm'     };
        Re_lam    = { 2000,      '1'     };
        Re_turb   = { 4000,      '1'     };
    end

    parameters(Access=private)
        length_eff      = { 6, 'm' };           % Effective tube length
        rel_roughness   = { 0.001, '1' };       % Relative roughness coefficient
        area_           = { 0, 'm^2' };         % Effective tube area
        D_h_            = { 0, 'm' };           % Effective circular diameter
        T               = {290,  'K'      };    % System temperature
        p_atm           = {0.101325, 'MPa'};    % Atmospheric pressure
    end

    function setup %#simple
        % Reassign area and D_h if circular cross-section
        if cs_type == 1
            % Circular-section tube
            if d_in <= 0
                pm_error('simscape:GreaterThanZero','Tube internal diameter')
            end
            area_ = pi*d_in^2/4;
            D_h_  = d_in;
    else
        if area <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Noncircular tube cross-sectional area')
        end
        if D_h <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Noncircular tube hydraulic diameter')
        end
        area_ = area;
        D_h_  = D_h;
    end

    % Remaining parameter range checking
    if s_factor <= 0
        pm_error('simscape:GreaterThanZero','Geometrical shape factor')
    end
    if length <= 0
        pm_error('simscape:GreaterThanZero','Tube length')
    end
    if length_ad < 0
        pm_error('simscape:GreaterThanOrEqualToZero', ...
            'Aggregate equivalent length of local resistances')
    end
    if roughness < 0
        pm_error('simscape:GreaterThanOrEqualToZero', ...
            'Internal surface roughness height')
    end
    if Re_lam <= 0
        pm_error('simscape:GreaterThanZero','Laminar flow upper margin')
    end
    if Re_turb <= Re_lam
        pm_error('simscape:GreaterThan','Turbulent flow lower margin', ...
            'Laminar flow upper margin')
    end

    % Derived constants
    length_eff = length + length_ad;
    rel_roughness = roughness/D_h_;
end

equations
    let
        % ACCESS WATER PARAMETERS FROM DOMAIN
        T_TLU    = A.T_TLU;
        p_TLU    = A.p_TLU;
        nu_TLU   = A.nu_TLU;
        rho_TLU  = A.rho_TLU;

        nu  = tablelookup(T_TLU, p_TLU, nu_TLU, T, B.p+p_atm, ...
            interpolation=linear, extrapolation=linear);
        rho  = tablelookup(T_TLU, p_TLU, rho_TLU, T, B.p+p_atm, ...
            interpolation=linear, extrapolation=linear);

        % FLOW WORK
        dir = if ge(q,0), 1 else -1 end; % Flow direction, no zero crossing
        Re  = q*dir*D_h_/area_/nu;

        % Friction coefficient in turbulent regime
        friction_coefficient = ...
            if le(Re, Re_lam/10), 0 ...
            else 1/(-1.8*log10( 6.9/Re + (rel_roughness/3.7)^1.11))^2 ...
            end;
            % Pressure drops
            dp_laminar   = s_factor*nu*length_eff/D_h_^2/area_/2*rho*q;
            dp_turbulent = friction_coefficient ...
                * length_eff/D_h_/area_^2/2*rho*q*q*dir;

            % Transition function. Simplest polynomial function ensuring
            % continuity of the pressure loss and its first derivative.
            Re_centered= (Re - Re_lam)/(Re_turb - Re_lam);
            transition = ...
                if le(Re, Re_lam), 0 ...
                elseif le(Re, Re_turb), 3 * Re_centered^2 - 2 * Re_centered^3 ...
                else 1 ...
                end;

            % Actual pressure loss
            pr_loss =...
                if le(Re, Re_lam), dp_laminar ...
                elseif le(Re, Re_turb), dp_laminar + transition ...
                    * (dp_turbulent - dp_laminar) ...
                else dp_turbulent ...
                end;
        in
            p == pr_loss;
        end
    end
end
```

Simple Valve

```
component simple_valve < saltdomain.branch
    % Simple Valve
    % Simple valve model with variable valve area.
    % Input AR directly assigns the valve area, but is limited in both direction
    % by the component parameters min_area and max_area.

    inputs
        area_in = {1, 'cm^2'}; % AR: Left
    end

    parameters
        min_area          = {1e-10, 'm^2'}; % Minimum restriction area
        max_area          = {0.005, 'm^2'}; % Maximum restriction area
        area              = {0.01,  'm^2'}; % Cross-sectional area at ports A and B
        length            = {0.1,   'm'  }; % Characteristic longitudinal length
        Cd                = {0.7,   '1'  }; % Discharge coefficient
        pressure_recovery = {1,     '1'  }; % Pressure recovery
        %                                     1 - On
        %                                     0 - Off
        Re_c              = {12,    '1'  }; % Critical Reynolds number
    end

    parameters (Access=protected, Hidden=true)
        T                     = {290,   'K'     }; % System temperature
        p_atm                 = {0.101325, 'MPa'}; % Atmospheric pressure
    end

    function setup
        % Check parameter range
        if ~((pressure_recovery == 0) || (pressure_recovery == 1))
            pm_error('simscape:Equal', 'Pressure reocvery', 'On or Off')
        end
        if area <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Cross-sectional area at ports A and B')
        end
        if min_area <= 0
            pm_error('simscape:GreaterThanZero', 'Minimum restriction area')
        end
        if min_area >= area
            pm_error('simscape:LessThan', 'Minimum restriction area', ...
                'Cross-sectional area at ports A and B')
        end
        if max_area >= area
            pm_error('simscape:LessThan', 'Maximum restriction area', ...
                'Cross-sectional area at ports A and B')
        end
        if max_area < min_area
            pm_error('simscape:GreaterThanOrEqual', 'Maximum restriction area', ...
                'Minimum restriction area')
        end
        if length <= 0
            pm_error('simscape:GreaterThanZero', ...
                'Characteristic longitudinal length')
        end
        if Cd <= 0
            pm_error('simscape:GreaterThanZero', 'Discharge coefficient')
        end
        if Cd > 1
            pm_error('simscape:LessThanOrEqual', 'Discharge coefficient', '1')
        end
        if Re_c <= 0
            pm_error('simscape:GreaterThanZero', 'Critical Reynolds number')
        end
    end

    equations
        let
            % ACCESS WATER PARAMETERS FROM DOMAIN
            T_TLU   = A.T_TLU;
            p_TLU   = A.p_TLU;
            rho_TLU = A.rho_TLU;
            nu_TLU = A.nu_TLU;

            p_B   = B.p;

            nu  = tablelookup(T_TLU, p_TLU, nu_TLU, T, p_B+p_atm, ...
                interpolation=linear, extrapolation=linear);
            rho = tablelookup(T_TLU, p_TLU, rho_TLU, T, p_B+p_atm, ...
                interpolation=linear, extrapolation=linear);

            % Critical mass flow rate for flow transition from laminar to turbulent
            q_c = Re_c * sqrt(pi/4 * restriction_area) * nu;

            % In turbulent regime, pressure drop is quadratic with respect to mass
            % flow rate. In laminar regime, pressure drop is linear with respect to
            % mass flow rate. Modify mass flow rate square to model
            % transitional smoothly.

            q_square = q * sqrt(q^2 + q_c^2);

            % The restriction area saturates at the minimum and maximum areas.
            restriction_area = ...
                if area_in <= min_area, min_area ...
                elseif area_in >= max_area, max_area ...
                else area_in ...
                end;

            % Pressure differential across the restriction
            area_ratio = restriction_area/area;
            pressure_differential = q_square * (1 - area_ratio^2) ...
                / (2 * rho * Cd^2 * restriction_area^2)*rho^2;

            % Ratio of overall pressure loss to pressure differential
            % across the restriction
            pressure_loss_ratio = ...
                if pressure_recovery == 0, 1 ...
                else (sqrt(1 - area_ratio^2*(1 - Cd^2)) - Cd*area_ratio) ...
                        / (sqrt(1 - area_ratio^2*(1 - Cd^2)) + Cd*area_ratio) ...
                end;
        in
            % Overall pressure loss after partial pressure recovery
            p == pressure_differential * pressure_loss_ratio;
        end
    end
end
```

Variable Ratio Splitter

```
component variable_ratio_splitter < saltdomain.branch_threeway
    % Variable Ratio Splitter
    % Component that splits the volume and mass flow into two different
    % branches. The ratio between the flows can be changed by adjusting the
    % output cross-section areas. The salt concentration in both outputs
    % is equal. Cannot be used as a merger.

    parameters
        a_B = {1, 'cm^2'}; %Output B cross-section area
        a_C = {1, 'cm^2'}; %Output C cross-section area
    end

    parameters (Access=protected, Hidden=true)
        T                      = {290,    'K'      }; % System temperature
        p_atm                  = {0.101325, 'MPa'}; % Atmospheric pressure
    end

    equations
        let
            % ACCESS WATER PARAMETERS FROM DOMAIN
            T_TLU    = A.T_TLU;
            p_TLU    = A.p_TLU;
            rho_TLU = A.rho_TLU;
            nu_TLU = A.nu_TLU;

            % Pressure p_X at node X
            p_A = A.p;
            p_B = B.p;
            p_C = C.p;

            rho = tablelookup(T_TLU, p_TLU, rho_TLU, T, p_A+p_atm, ...
                interpolation=linear, extrapolation=linear);
        in
            % FLOW WORK
            rho*q_B^2 == a_B^2*2*(p_A-p_B);
            rho*q_C^2 == a_C^2*2*(p_A-p_C);

            % Equal concentration at node B and C
            if (q_A>0)
                msalt_B/q_B == msalt_C/q_C;
            else
                msalt_A==0; % Define flows as zero at singularity
            end
        end
    end
end
```

Water Tank

```
component watertank
    % Watertank
    % Idealized water tank with no leakage and uniform cross-section area.
    % All node inputs are pressure dependent, but only allows flow in one direction.
    % IN and LOOP only allows positive flow, OUT and VENT only allows negative flow.
    % C outputs the concentration (salt mass divided by solution volume).
    % H outputs the tank water level.

    outputs
        H = {0, 'cm'};       %H:bottom
        C = {0, 'kg/m^3'};   %C:bottom
    end

    nodes
        IN = saltdomain.salt_dom;   %IN:top
        LOOP = saltdomain.salt_dom; %LOOP:top
        VENT = saltdomain.salt_dom; %VENT:top
        OUT = saltdomain.salt_dom;  %OUT:bottom
    end

    parameters
        % Tank parameters
        A_tank = { 100, 'cm^2' };       % Surface area
        p_amb = { 0, 'Pa' };            % Ambient pressure
        V_init = { 500 , 'cm^3'};       % Initial volume
        C_init = { 100e-3, 'kg/m^3'};   % Initial concentration

        % Node parameters
        a_in   = { 1.0, 'cm^2' };    % Inlet pipe area
        a_loop = { 1.0, 'cm^2' };    % Loop pipe area
        a_vent = { 5.0, 'cm^2' };    % Vent pipe area
        a_out = { 1.0, 'cm^2' };     % Outlet pipe area

        h_vent = {10, 'cm' };        %Vent height
    end

    parameters(Access = private)
        % Water paramters and internal parameters
        g               = { 9.81, 'm/s^2'   };    % Gravity
        rho_INTERNAL    = { 1000, 'kg/m^3' };    % Density
        rho_OUT         = { 1000, 'kg/m^3' };    % Density
        T               = { 290,  'K'       };    % System temperature
        p_atm           = {0.101325,  'MPa'};    % Atmospheric pressure
    end

    variables
        % Through and across variables
        q_in = { 0 , 'm^3/s'};  %flow rate from inlet
        msalt_in = {0, 'kg/s'}; %salt rate from inlet

        q_loop = { 0 , 'm^3/s'};  %flow rate from loop
        msalt_loop = {0, 'kg/s'}; %salt rate from loop

        q_vent = { 0 , 'm^3/s'};  %flow rate to vent
        msalt_vent = {0, 'kg/s'}; %salt rate to vent

        q_out = { 0 , 'm^3/s'};  %flow rate to outlet
        msalt_out = {0, 'kg/s'}; %salt rate to outlet
        % Internal variables
        V = { 500, 'cm^3' }; % Volume
        macc = {0.1,'kg'};   % Accumulated Salt mass
    end

    function setup
        % PARAMETER CHECKING
        if A_tank<=0
            error('Area must be strictly positive.')
        end

        if (p_amb<0)
            error('Ambient pressure must be positive.')
        end
        if (a_out<0) || (a_in<0) || (a_loop<0) || (a_vent<0)
            error('Pipe area must be positive')
        end

        % ACCESS TEMPERATURE AND ATMOSPHERIC PRESSURE FROM NODE
        T       = IN.T;
        p_atm   = IN.p_atm;

        % TEMPERATURE AND ATMOSPHERIC PRESSURE PROPOGATION
        OUT.T == T;
        VENT.T == T;
        LOOP.T == T;

        OUT.p_atm == p_atm;
        VENT.p_atm == p_atm;
        LOOP.p_atm == p_atm;

        % INITIALISATION
        V = V_init;
        macc=C_init*V_init;
    end

    branches
        % Defines flows q_X and msalt_X from X to internal node
        q_in : IN.q -> *;
        msalt_in : IN.msalt -> *;

        q_loop : LOOP.q -> *;
        msalt_loop : LOOP.msalt -> *;

        q_vent : VENT.q -> *;
        msalt_vent : VENT.msalt -> *;

        q_out : OUT.q -> *;
        msalt_out : OUT.msalt -> *;
    end

    equations
        let
            % ACROSS
            p_out    = OUT.p;
            p_in     = IN.p;
            p_loop   = LOOP.p;
            p_vent   = VENT.p;

            % ACCESS WATER PARAMETERS FROM DOMAIN
            T_TLU    = IN.T_TLU;
            p_TLU    = IN.p_TLU;
            rho_TLU = IN.rho_TLU;
        rho_INTERNAL = tablelookup(T_TLU, p_TLU, rho_TLU, T, p_atm, ...
            interpolation=linear, extrapolation=linear);
        rho_OUT = tablelookup(T_TLU, p_TLU, rho_TLU, T, p_out+p_atm, ...
            interpolation=linear, extrapolation=linear);

        % INTERNAL
        h = V/A_tank;

        % OUT
        p = if ge(h,0), p_amb + rho_OUT*g*h else p_amb end;

        % VENT
        p_vi = if ge(h,h_vent), p_amb + rho_INTERNAL*g*(h-h_vent) else p_amb end;
    in
        % HEIGHT AND CONCENTRATION PHYSICAL SIGNALS
        H == h;
        if (V>0)
            C == macc/V;
        else
            C == 0;
        end

        % VOLUME EQUATION
        V.der == q_in + q_out + q_loop + q_vent;

        % FLOW WORK
        % IN
        if(p_in>p_amb)
            rho_INTERNAL*q_in^2 == a_in^2*2*(p_in-p_amb);
        else
            q_in==0;
        end

        % LOOP
        if(p_loop>p_amb)
            rho_INTERNAL*q_loop^2 == a_loop^2*2*(p_loop-p_amb);
        else
            q_loop==0;
        end

        % OUT
        if (h>0)
            rho_OUT*q_out^2 == a_out^2*2*(p-p_out);
        else
            q_out == 0;
        end

        % VENT
        if(h>h_vent)
            rho_INTERNAL*q_vent^2 == a_vent^2*2*(p_vi-p_vent);
        else
            q_vent == 0;
        end

        % SALT FLOWS
        % Mass conservation (salt)
        macc.der == msalt_in + msalt_out + msalt_vent + msalt_loop;
```

```
            % OUT
            if (V>0) && (q_out <0)
                (msalt_out/(q_out)) == (macc/V);
            else
                msalt_out == 0;
            end

            % VENT
            if (V>0) && (q_vent <0)
                (msalt_vent/(q_vent)) == (macc/V);
            else
                msalt_vent == 0;
            end
        end
    end
end
```

Sources, references and settings

Salt Source

```
component salt_source
    % Salt Source
    % This block represents an ideal source of salt mass-flow, with no
    % energy losses. Water must flow in direction from A to B, input C
    % decides the salt concentration as the salt mass flow divided by the
    % solution volume flow.

    inputs
        C = {0, 'kg/m^3'} %C:bottom
    end

    nodes
        A = saltdomain.salt_dom; % A:bottom
        B = saltdomain.salt_dom; % B:top
    end

    variables
        p = {0, 'Pa'};       % Pressure (gauge)
        msalt = {0,'kg/s'}; % Salt flow
        q = {0,'m^3/s'};     % water flow
    end

    branches
        q : A.q -> B.q;               % Define flow q from A to B
        msalt : A.msalt -> B.msalt; % Define flow msalt from A to B
    end

    function setup
        % ACCESS TEMPERATURE AND ATMOSPHERIC PRESSURE FROM NODE
        T       = A.T;
        p_atm   = A.p_atm;

        % TEMPERATURE AND ATMOSPHERIC PRESSURE PROPOGATION
        B.T == T;
        B.p_atm == p_atm;
    end

    equations
        p == A.p - B.p;      % Define pressure between A and B

        % Define msalt to give correct concentration C
        if q>0
            msalt/q == C;
        else
            msalt == 0;      % msalt defined to zero at singularity
        end
        p == 0;              % Zero pressure loss
    end
end
```

Saline Hydraulic Reference

```
component salt_ref
    % Saline hydraulic reference
    % This block represents a connection to atmosphere. It has one saline
    % hydraulic conserving port. Connect to it hydraulic ports of other
    % blocks that are considered directly connected to atmosphere.

    nodes
        P = saltdomain.salt_dom; %:top
    end

    connections
        connect(P, *);
    end
end
```

Water Settings

```
component(Propagation=source) watersettings
    % Water Settings
    % The block assigns fluid properties for all components assembled in
    % a particular loop. The loop detection is performed automatically and the
    % block is considered as part of the loop if it is hydraulically connected
    % to at least one of the loop components. If no Hydraulic Fluid block is
    % connected to the loop, the default properties of the Custom Hydraulic
    % Fluid block are assigned.

    parameters
        %T_min            = {273.16,   'K'       }; % Minimum valid temperature
        %T_max            = {373.16,   'K'       }; % Maximum valid temperature
        %p_min            = {0.01,     'MPa'     }; % Minimum valid pressure
        %p_max            = {50,       'MPa'     }; % Maximum valid pressure
        p_atm            = {0.101325, 'MPa'     }; % Atmospheric pressure
        T                = {290,      'K'       }; % System temperature
    end

    nodes
        G = saltdomain.salt_dom; % :right
    end

    function setup
    %G.T_min            = T_min;
    %G.T_max            = T_max;
    %G.p_min            = p_min;
    %G.p_max            = p_max;
    G.p_atm            = p_atm;
    G.T                = T;
    end
end
```

Sensors

Temperature Sensor

```
component T_sensor < saltdomain.branch
    % Temperature sensor
    % Ideal temperature sensor, does not obstruct neither volume or mass flows
    % in any way, and does not inflict any pressure drops.
    % T_OUT outputs the temperature at node A.

    outputs
        T = {0, 'K'}; % T:Right
    end

    equations
        p==0;       % Zero pressure loss
        T == A.T;   % Output temperature at A
    end
end
```

Concentration Sensor

```
component conc_sensor < saltdomain.branch
    % Concentration sensor
    % Ideal concentration sensor, does not obstruct neither volume or mass flows
    % in any way, and does not inflict any pressure drops.
    % C outputs the solution concentration as the salt mass flow through
    % the component divided by the liquid volume flow.

    outputs
        C = {0, 'kg/m^3'}; % T:Right
    end

    equations
        p==0;       %Zero pressure loss

        % Calculate salt concentration
        if (q>0) || (q<0)
            C == msalt/q;
        else
            C == 0; % Defined as zero when approaching singularity
        end
    end
end
```

Flow Sensor

```
component flow_sensor < saltdomain.branch
    % Flow sensor
    % Ideal flow sensor, does not obstruct volume or mass flows in any way,
    % Does not inflict any pressure drops.
    % Outputs the liquid volume flow rate through the component

    outputs
        Q = {0, 'lpm'}; % C:Right
    end

    equations
        p==0;           % Zero pressure loss
        Q ==q;          % Output q, liquid flow from A to B
    end
end
```

Height Sensor

```
component height_sensor
    % Height Sensor
    % Takes the measured water level H and sends signal 1 to SIGNAL when
    % the level is below the minimum allowed water level. Sends 0 to SIGNAL
    % when the water level reaches the maximum allowed level.

    inputs
        h = {1,'cm'}        %H:right
    end

    outputs
        SIGNAL = {0, '1'};  %S:left
    end

    variables (Event = true, Access = private)
        ON = {value = 0, priority = priority.high};
    end

    parameters
        h_LOW = {4, 'cm'};  %Minimum allowed water level
        h_HI = {8, 'cm'};   %Maximum allowed water level
    end

    equations
        SIGNAL == ON;
    end

    events
        when edge(h<h_LOW)
            ON = 1;             % Output 1 when h reaches low limit
        elsewhen edge(h>h_HI)   % Output 0 when h reaches high limit
            ON = 0;
        end
    end
end
```

Pressure Sensor

```
component pressure_sensor < saltdomain.branch
    % Pressure sensor
    % Ideal pressure sensor that measures the differential pressure between
    % two nodes. Does not allow any mass or volume flow over it.
    % Output pressure differential P between A and B.

    outputs
        P = {0, 'bar'}; % P:Right
    end

    equations
        q==0;        % Zero liquid volume flow
        msalt==0;    % Zero salt mass flow
        P==p;        % Pressure differential
    end
end
```

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER´S THESIS *Date of issue* March 2017 *Document Number* ISRN LUTFD2/TFRT--6027--SE
Author(s) Joel Assarsson Simon Thoulouis	*Supervisor* Bo Wennberg, Baxter, Lund Tore Hägglund, Dept. of Automatic Control, Lund University, Sweden Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner) *Sponsoring organization*

Title and subtitle

Physical network modeling of a reverse osmosis purification unit

Abstract

The use of computer models in research and development is a powerful tool to make the engineering process more effective. By providing accurate simulations, the need for working prototypes and extensive testing is reduced. This Master's Thesis investigates the Gambro WRO 300 H reverse osmosis water purification system with the aim of creating a working and easily adaptable computer model of the machine. For this purpose, the physical network modeling approach provided by the MathWorks Simscape environment has been used, and the experimental data from the machine is fitted to an a priori model of the reverse osmosis process – the solution-diffusion model. The result is a Simscape model of the WRO 300 H as well as four experimental model designs implemented using the same custom Simscape library. While the models do capture the overall behaviour of the WRO 300 H and provide interesting insights in the experimental designs at a qualitative level, there are significant improvements that could be made to enhance the accuracy of the membrane model by modeling its internal physical phenomena.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-98	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/