Modeling of an Experimental Batch Plant with Modelica

Katja Poschlad[1], Manuel A. Pereira Remelhe[1], Martin Otter[2]
[1]University of Dortmund, Process Control Laboratory (AST), Germany
[2]German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Germany

Abstract

A Modelica model of an experimental batch plant installed at the Process Control Laboratory (AST) of the University of Dortmund was developed. The plant model consists of tanks, pipes, valves, pumps and a control system to mix, buffer, heat, cool and evaporate a mixture of water and sodium chloride. Details of the most important components, of the plant model and of the sequence control implemented with the Modelica StateGraph library are discussed.

1 Introduction

The Modelica_Fluid library [4] is a free library to be included in the Modelica standard library. It is supposed to be usable for several application areas. Based on this library, a model of a laboratory batch plant was developed with the following goals:

- to further develop the Modelica_Fluid library,
- to demonstrate the ability to model and simulate batch plants with Modelica,
- to have a free, non-trivial, controlled plant model to test and verify new modeling ideas and algorithms,
- to use it as benchmark problem for hybrid control systems.

To achieve these goals and in order to be able to implement the batch plant, new components have been developed such as a much more generic tank model, a model for cooling and heating tanks, an evaporator, a condenser, and several simple components such as a non-horizontal pipe and a controlled valve model. Furthermore, a model for a medium system consisting of water and sodium chloride has been constructed. In this article an overview of this project is given. More details can be found in [5].

2 Description of the Batch Plant

The process under consideration is an evaporation plant for a student lab at the Process Control Laboratory (AST) of the University of Dortmund that evaporates a water sodium chloride mixture so that a higher concentrated solution is produced [1]. The task of the students is to learn how to program the process control system. A picture of the batch plant is shown in figure 1. The flow sheet diagram is shown in figure 2.

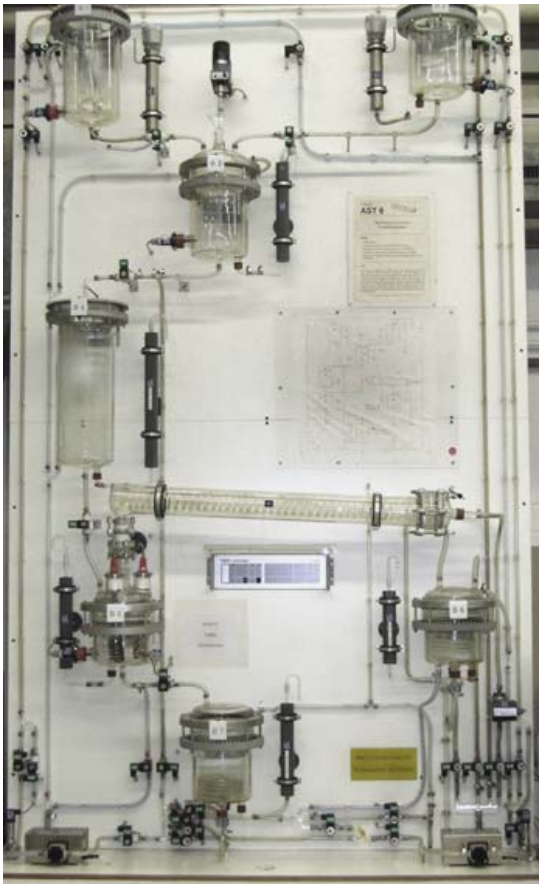


Figure 1: Picture of AST batch plant.

Pure water from tank B1 and concentrated sodium chloride solution from tank B2 are mixed in a mixing tank B3. After buffering in tank B4 the mixture flows to the evaporator B5. Here the water sodium chloride mixture is evaporated until the desired concentration is reached. The steam is condensed in the condenser K1 and cooled afterwards in the cooling tank B6. The concentrated solution is also led to a cooling tank B7. The cooled fluids are pumped back to the charging vessels by the pumps P1 and P2. Between the tanks several valves are present that are regulated by a central control system.

3 Main Fluid Components

The plant is modeled with components of the Modelica_Fluid library, such as pipe and pump models [4]. Several fluid flow components had to be extended or newly developed and implemented, in order to model and simulate this batch plant.

3.1 Generic tank model

A generic tank model was newly implemented that describes a tank which is open to the environment at fixed ambient pressure. Heat transfer to the environment and to the tank walls is neglected. The tank is

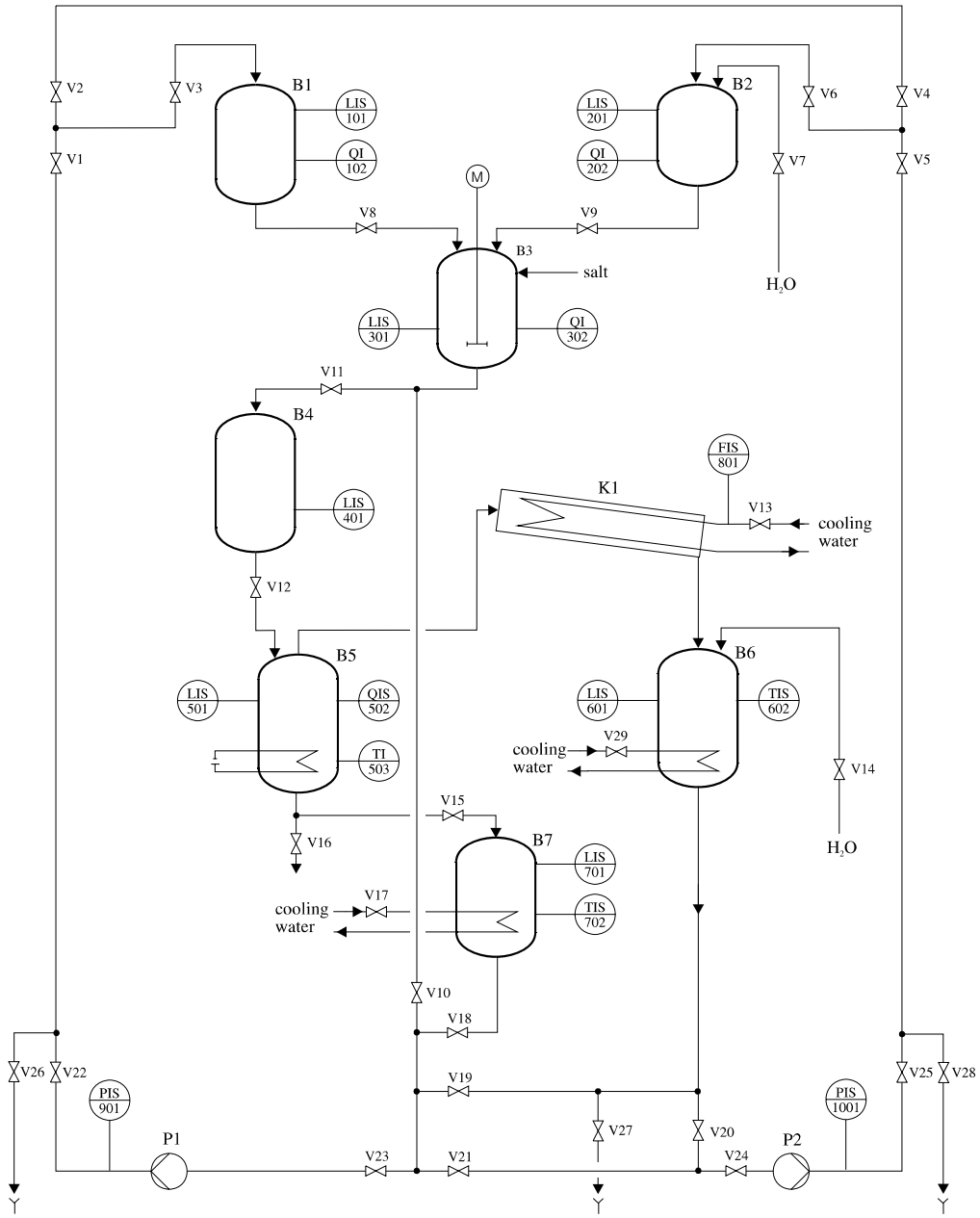


Figure 2: Flow sheet diagram of AST batch plant.

filled with a single or multiple-substance liquid, assumed to have uniform temperature and mass fractions.

The tanks in the AST batch plant, as well as most batch plants in industry, have more than one inlet/outlet at different heights. For this reason, the "Tank" model has a vector of FluidPort connectors at the top and at the bottom of the tank icon, as seen in the screenshot of the tank icon in figure 3.

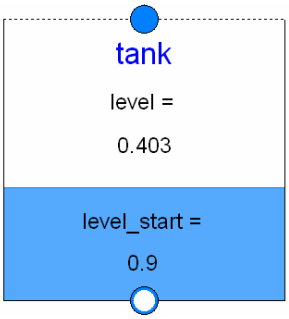


Figure 3: Screen shot of the tank icon

The fill level is dynamically visualized with the blue square and also the actual value is given (in figure 3: "level = 0.403"). This is implemented in the icon annotations by function calls provided from the Beta-release of the UserInteraction library.

The top connectors "topPorts[:]" are assumed to be always above the overflow level. Therefore, fluid from the tank can never flow into them.

The connectors "ports[:] at the bottom of the icon are either attached at the bottom or at the side of the tank and fluid can flow in both directions.

For the top connectors it is sufficient to know the number of connectors, since the pressure at the connectors is always identical to the ambient pressure. The bottom and side connectors are defined by (hydraulic) inner diameters of the inlet/outlet pipe and at which position "portLevel" the connector is present.

The parameter menu of the tank is shown in figure 4.

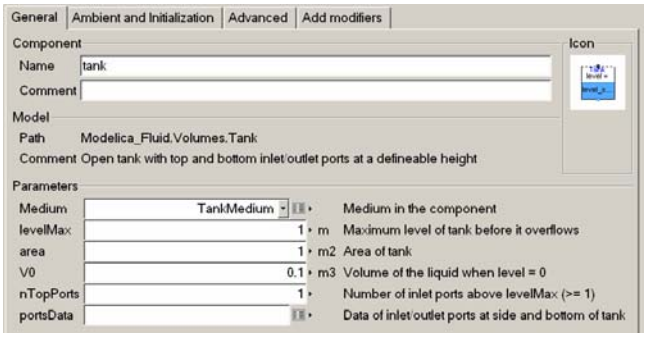


Figure 4: Tank parameter menu

Parameter "levelMax" defines the maximum level until the tank overflows. Currently, this triggers just an assert. Parameter "portsData" is an array of records that defines diameter and port level of the bottom and side connectors. An example is shown in figure 5 below.

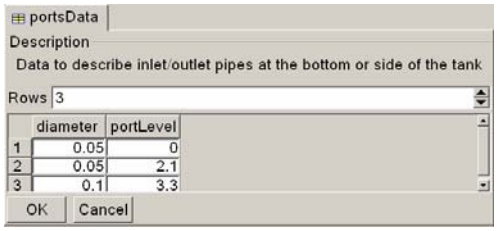


Figure 5: Parameter menu to define tank ports

The tab "Ambient and Initialization", see figure 4, defines the ambient and initial conditions.

Due to the equation based nature of Modelica, the implementation of the *basic* tank equations is straightforward:

```
model tank
  import IF = Modelica_Fluid.Interfaces;
  replaceable package Medium =
   Modelica.Media.Interfaces.PartialMedium;
  IF.FluidPort_a topPorts[:](
      redeclare package Medium = Medium);
  IF.FluidPort_b ports[:](
      redeclare package Medium = Medium);
  Medium.BaseProperties medium(..);
  ...
  // Total quantities
  medium.p = p_ambient;
  V   = area*level + V0 "Volume of fluid";
  m   = V*medium.d "Mass of fluid";
  mXi = m*medium.Xi "Mass of fluid comp.";
  U   = m*medium.u "Internal energy";

  // Mass balances
  der(m) = sum(topPorts.m_flow) +
           sum(ports.m_flow);
  for i in 1:Medium.nXi loop
    der(mXi[i]) = sum(topPorts.mXi_flow[i])
                + sum(ports.mXi_flow[i]);
  end for;

  // Energy balance
  der(U) = sum(topPorts.H_flow) +
           sum(ports.H_flow) -
           p_ambient*der(V)
  ...
end tank;
```

If the fluid is incompressible, the term "p_ambient***der**(V)" is removed from the energy balance. The reason is that otherwise unphysical small temperature changes occur, if the tank level changes. By removing this term, mechanical work is neglected since it is also neglected in the medium model.

As usual in the Modelica_Fluid library (for details, see [4]), mass flow rate and enthalpy flow rate at the ports are computed with the semiLinear(..) operator

in order to describe the potential bidirectional flow of the fluid, e.g.,

```
ports[i].H_flow = semiLinear(
  ports[i].m_flow, ports[i].h, medium.h);
```

Finally an equation for the port pressures has to be provided. For the topPorts, this is simple, because they are by definition always above the fluid level and therefore topPorts.p[i] = p_ambient.

If a bottom or side port ports[i] is below the actual fluid level, the Bernoulli equation can be used to compute the port pressure (it is assumed that the tank has a uniform pressure, temperature and density; the Bernoulli equation holds also in cases where the density is varying with time, provided it has the same value along the path under consideration):

If the fluid flows out of the tank, it is applied from the tank level to the corresponding port. In this situation no significant pressure losses occur, with exception of a potential small one that depends on the shape of the outlet pipe.

If the fluid flows into the tank, the whole kinetic energy is dissipated and does not lead to a pressure increase. Taken this into consideration the basic equation is given as:

```
ports[i].p = p_ambient +
   (level - portLevel[i])*g*medium.d -
   if ports[i].m_flow < 0 then
      ports[i].m_flow^2/
                (2*medium.d*area[i]^2)
   else 0;
```

Severe difficulties occur if the fluid level drops below an inlet/outlet port and therefore the corresponding pipe might no longer contain fluid or is filled only partially with fluid. An example is shown in figure 6:

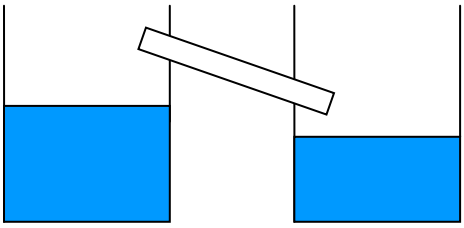


Figure 6: Inlet/outlet pipe over the tank levels

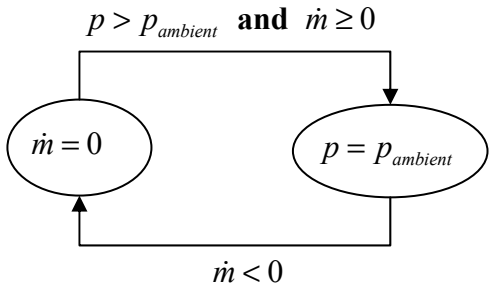
A detailed description of this situation would require modeling the mixture of liquid and air and taking into account that the mixture is not homogenous in a pipe. Since this would complicate the modeling of tanks, pipes and other components considerably, different approximate solutions have been evaluated and verified by a suite of test models.

In the tank, there are now two solutions implemented that can be selected in the "Advanced" menu. The default solution operates in the following way when the fluid level is below a port:

A large pressure loss factor (default: $\zeta_{out} = 10^5$), is used when the fluid flows out of the port and a small pressure loss factor (default: $\zeta_{in} = 0.01$) is used when the fluid flows into the port. The quadratic characteristic around zero mass flow rate is regularized with the Modelica_Fluid.Utilities.regSquare2(..) function, in order to guarantee that at zero mass flow rate the derivative of the pressure drop equation is neither zero nor infinity. Note, when the level is above the pipe port, we have $\zeta_{out} = 1$ and $\zeta_{in} = 0$. Numerically, this means that the pressure drop equation of the Bernoulli equation is a strict monotonically rising characteristic so that a non-linear solver can handle system equations that contain such an equation, if the step size is selected small enough. Physically, this means that a small leakage mass flow rate appears that depends essentially on the value of ζ_{out}. All test models work very reliably with this solution, see, e.g., the models in Modelica_Fluid.Examples.Tanks.

Alternatively, a second solution can be selected that is based on the following considerations:

If the level is below the port and fluid flows from the port into the tank, the port pressure is identical to the ambient pressure. If the fluid tries to flow out of the tank, the mass flow rate is explicitly set to zero. The solution is now to switch between these two equations using the following simple state machine:



This state machine is implemented in Modelica as:

```
m_flow_out = pre (m_flow_out) and
             not port.p > p_ambient or
                    port.m_flow < -1e-6;
0 = if pre(m_flow_out) then m_flow
                       else p-p_ambient;
```

This solution has the advantage that no leakage flow occurs and it works reliably in many situations. It may fail in cases where a non-linear system of equations is no longer well defined, e.g., due to the equation m_flow =0 in one branch of the if statement (it fails for example in the complicated situation of the example Modelica_Fluid.Examples.Tanks.Tanks WithEmptyingPipe2).

A third solution variant would be to take the same basic equations as above, but use a declarative formulation to switch between the two equations based on a parameterized curve description (similarly to the description of ideal electrical switches and to friction in the Modelica standard library). This yields the equations:

```
        m_flow_out = s < 0;
      port.m_flow = if m_flow_out then 0
                                  else s;
port.p-p_ambient = if m_flow_out then s
                                  else 0;
```

Numerical experiments with the test cases show that this third variant is not reliable and often fails. The reason is that this formulation leads to *non-linear mixed* systems of equations having both Real and Boolean variables (`m_flow_out`) as unknowns and the algorithms in Dymola seem to not work well in such a situation. Contrary, the parameterized curve descriptions of, e.g., electrical switches, in the Modelica standard library lead to *linear mixed* systems of equations that are solved very well in Dymola.

We have now two sets of equations, one that is used if the level is above the port level and one when it is below. The question arises, how to switch between these two equation sets. In Modelica one could use a simple if clause:

```
if level > portLevel[i] then
   // Bernoulli equation
else
   // equation to handle empty port
end if;
```

since the relation will automatically trigger a state event and therefore the switching is performed numerically in a reliable way.

However, in some situations, especially, when a tank is empty and all fluid that flows into the tank is at once flowing out at the bottom port, a numerical solver will switch extremely often between the two branches of the if clause, leading to a very small step size so that the simulation is practically stopped, i.e., chattering occurs.

Chattering can always be reduced by introducing an appropriate hysteresis. For a tank this is rather straightforward and also makes physically sense, because the exact time instant when to switch between the two equations is not well defined: A port at the side of the tank has a certain diameter. When the level is above the upper part of the pipe, there is clearly fluid in the port. If the level is below the lower part of the pipe, there is clearly no fluid in the port, but in between the situation is not well defined.

In the tank model the hysteresis is formulated in the following way, using the fraction k of the port diameter as hysteresis distance:

```
aboveLevel[i] =
  level >= (portLevel[i] + diameter[i]/k)
  or pre(aboveLevel[i]) and
  level >= (portLevel[i] - diameter[i]/k)
levelAbovePort[i] = if aboveLevel[i] then
                    level - portLevel[i] else 0;
```

Example Modelica_Fluid.Examples.Tanks.OneTank demonstrates the behavior. See the model setup in figure 7. It consists of a tank where a constant mass flow rate is entering at the top inlet and the fluid flows out at the bottom port with a higher mass flow rate.

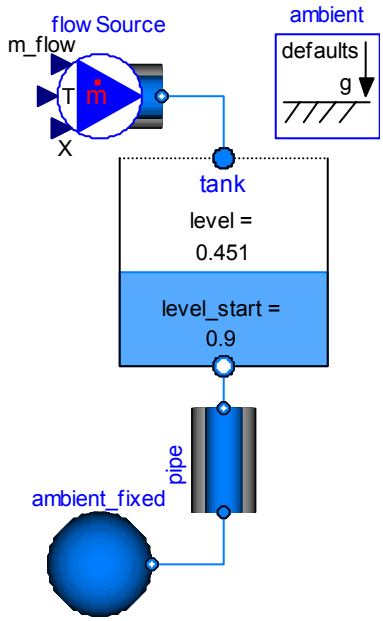


Figure 7: Fluid flows faster out of the tank bottom outlet as it flows in by the top inlet leading to chattering.

A plot of the level is shown in figure 8. The chattering after about 50 s when the tank is empty is clearly visible. Due to the introduced hysteresis, the chattering influences the simulation speed only marginally.

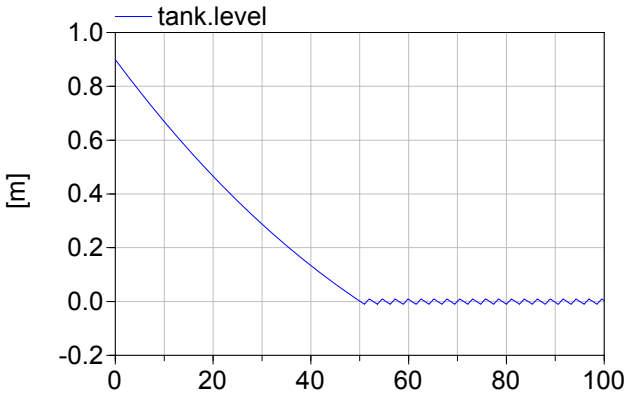


Figure 8: Level of tank from figure 7 over time.

3.2 Evaporator and condenser model

The experimental batch plant contains an evaporator and a condenser, which have to be modeled. In order to simplify the task, the following simplifications are made:

- evaporator and condenser are modeled in one component.
- condensed water is saturated at the end of the condenser.
- no hold up in the condenser.

The evaporator/condenser model is an extension of the generic tank model. A heat port is added to the tank model which extends the heat balance. Furthermore a single fluid port is added which symbolizes the end of the condenser. The icon of the model is giving in 9.

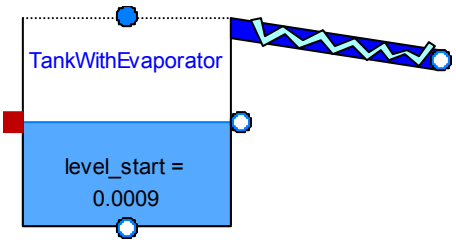


Figure 9: Screen shot of evaporator/condenser model.

The energy balance of the generic tank is changed to

```
der(U) = sum(H_flow_bottomPorts) +
         sum(H_flow_sidePorts) +
         sum(H_flow_topPorts) +
         heatPort.Q_flow +
         Condensed.m_flow*hv -
         p_ambient*der(V);
```

with the enthalpy of the saturated vapor `hv`. Because the assumption is made that the condensed water is saturated, the energy from the cooling water in the condenser is equal to the condensing energy. So only the evaporator needs to be balanced.

The mass flow rate of the condensed water `m_con` for a binary system is calculated by

```
m_con = if p > p_sat then 0 else
        - heatPort.Q_flow /
        (hv-hl+Xi(1)*(cp*dT_dX+dhl_dX));
```

with the supplied heat `heatPort.Q_flow`, the specific enthalpy of the saturated liquid `hl`, the saturated vapour `hv`, the heat capacity `cp`, the partial derivative of the boiling temperature with respect to the mass fraction `dT_dX`, and the partial derivative of the specific enthalpy `hl` with respect to the mass fraction `dhl_dX`. This calculation assumes that only water evaporates and that all the energy is used for evaporation, once the saturation pressure `p_sat` is reached. This can lead to chattering if cold fluid flows into the tank. To avoid the chattering the variable `Q_up` is introduced, that gives the amount of energy used to rise the temperature in the tank up to the boiling point. These considerations lead to the following formulation that avoids chattering:

```
if Q_up > heatPort.Q_flow then
  m_con = 0;
else
  m_con=-( heatPort.Q_flow -Q_up)/
    (hv-hl+Xi(1)*(cp*dT_dX+dhl_dX));
end if;
```

3.3 Other Component Models

Additionally, some simpler component models have been implemented, especially:

- Model class **Ambient** to define default ambient conditions (such as default ambient pressure) with one inner ambient instance and access the data via inner/outer from other elements.
- Model class **ValveDiscrete** to open and close a simple valve model by a Boolean input signal.
- Model class **WallFriction** is renamed to **WallFrictionAndGravity** and an additional term `Δp = height_ab*d*g` for the calculation of the pressure drop due to gravity is added, given the relative height `height_ab` of port_b over port_a.

4 A medium model for mixture of water and sodium-chloride

To model the batch plant it is necessary to construct a model for the system water-sodium chloride. This type of medium is not yet available in the Modelica.Media library and therefore a superclass for two phase models of a mixture was introduced. The subclass relationship of this extension is displayed in figure 10.

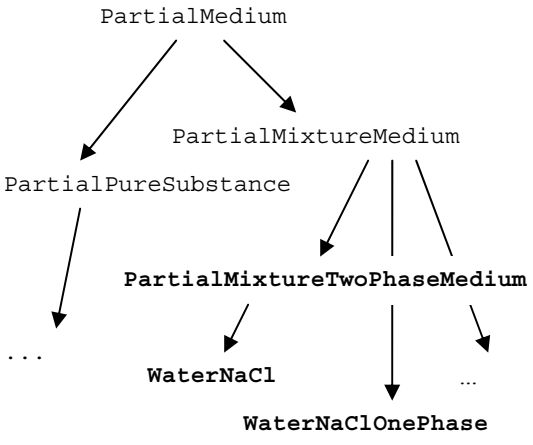


Figure 10: Hierarchical structure of Modelica.Media

For the model of the batch plant the following fluid characteristics are important and therefore included in the medium model:

The density `d`, the specific enthalpy `h` and the dynamic viscosity `eta` are described as a function of temperature `T`, pressure `p`, and composition `Xi`. To model the evaporation, the saturation pressure `p_sat` is needed which is a function of temperature `T` and composition `Xi`.

For the implementation of the medium model the interpolation of Chou [2] is used, as well as the data for the dynamic viscosity of DECHEMA [3]. The resulting terms are as follows:

Density ρ is calculated by

$$\frac{1}{\rho(T,w,p)} = F_1(T) - p\cdot F_2(T) - p^2\cdot F_3(T)$$
$$+w\cdot F_4(T) + w^2\cdot F_5(T) - w\cdot p\cdot F_6(T)$$
$$-w^2\cdot p\cdot F_7(T) - \frac{1}{2}w\cdot p^2\cdot F_8(T)$$

with the pressure p, the temperature T, the mass fraction of sodium chloride w and the terms $F_1 - F_8$ which are provided in the paper of Chou [2].

The specific enthalpy h is calculated by

$$h = h_0 + \int_{T_0}^{T} c_p \cdot dT$$

where the specific enthalpy h_0 under standard condition is defined by

$$h_0(w) = E_1\cdot(1-w) + E_2\cdot w^{1,5}$$
$$+E_3\cdot w^2 + E_4\cdot w^{2,5} + E_5\cdot w^3$$

with the mass fraction w and the parameters $E_1 - E_5$ [2]. The heat capacity c_P is calculated by

$$c_p(x,T) = C_1(x) - C_2(x)\cdot T + C_3(x)\cdot T^2$$

with the mole fraction of sodium chloride x, and the terms $C_1 - C_3$ [2].

To calculate the saturation pressure p_S the following equation is used:

$$\ln(p_s(x,T)) = (1 - G_1\cdot x + G_2\cdot x^2)\cdot$$
$$\left(G_3 - \frac{G_4}{T} - G_5\cdot\ln(T) + G_6\cdot T\right)$$
$$-x\cdot(G_7 - G_8\cdot x + G_9 x^2)$$

with parameters $G_1 - G_9$ from [2].

The viscosity η, which is needed for the calculation of the pressure drop within a pipe, is calculated by

$$\eta(T,w) = \exp$$
$$\left(c_7 + c_1\cdot T^3 + c_2\cdot T^2 + c_3\cdot T + c_4\cdot w^3 + c_5\cdot w^2 + c_6\cdot w\right)$$

with the parameters

$$c_1 = \text{-6.83241E-07}\,\frac{1}{K^3}$$
$$c_2 = 0.000765676\,\frac{1}{K^2}$$
$$c_3 = \text{-0.297018665}\,\frac{1}{K}$$
$$c_4 = \text{-0.299730079}$$
$$c_5 = 2.065267182$$
$$c_6 = 1.546491257$$
$$c_7 = 31.57571595$$

For the vapor-liquid-equilibrium also the specific enthalpy of the water vapor is needed. This is taken from the medium model `StandardWater`.

5 Controller

The controller that regulates the valves, the heating and the cooling is modeled with the Modelica.StateGraph library, i.e., using basically a "sequential function chart" with additional equations. The flow chart with steps and transitions is shown in figure 11. The left triangle is an input connector containing all sensor signals whereas the triangle on the right side is an output connector containing all actuator signals.

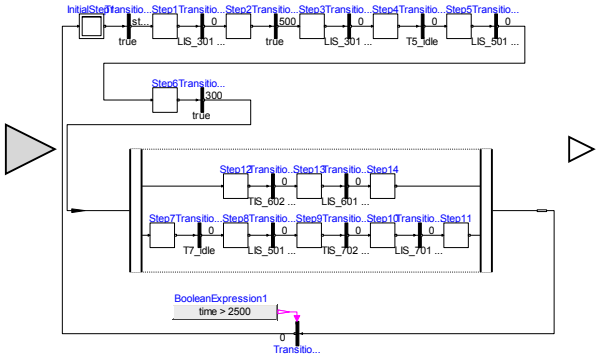


Figure 11: StateGraph model of the controller

In the first step the valve V8 is open so that liquid flows from tank B1 to tank B3 till a minimum level of 0,13 m is reached in B3 (Transition2). In step2 valve V9 opens till a desired concentration is measured in tank B3 (Transition3). Step3 leads the fluid to tank B4 by opening valve V11 till the level in tank B3 is below 0,01m. In Step4 T5_idle shows whether tank B5 can be filled.

```
T5_idle =
  LIS_501 < 0.01 and not V15.open;
```

If Step5 is active, valve V12 opens which leads the liquid to tank B5 till the level T5_batch_level is reached in tank B5. Step6 starts the heating in tank B5 till the desired concentration is reached. Since the evaporated water and the concentrated solution are split up, the controlling has to be split as well which is realized by parallel branches. The upper branch is for the condensed water. Step12 starts the cooling in tank B6 till the temperature is below 20°C. After that the valves V20, V24, V25, V5 and V6 have to be opened so that the pump P2 can pump the water to tank B2. The lower branch controls the concentrated solution. If tank B7 can be filled (T7_idle = true), in Step8 the valve V15 is open till the tank B5 is empty. Then the cooling in tank B7 is switched on till the temperature is low enough. To pump the liquid back to tank B1 with pump P1 the valves V18, V23, V22, V1 and V3 have to be opened. Once both branches are finished, and time > 2500 s, the cycle starts from the beginning.

The controller input connector "`sensors`" contains all measured values such as the level of a tank, the temperature in a tank or the concentration. The measured values are named in analogue to their names in the flow sheet of the batch plant (see figure 2). e.g.

```
sensors.LIS_301 = B3.level;
```

The Boolean outputs are written in the output connector "actuators". E.g.

```
actuator.V9 = Step2.active;
```

where `Step2.active` becomes true when Step1 has been active and the transition `Transistion2`

```
LIS_301 >= 0.13;
```

becomes true.

6 Model of the Batch Plant

With the new components, the new media model, the Modelica_Fluid and the Modelica.StateGraph library, a model of the batch plant can be assembled. The tanks B1, B2, B3 and B4 are modeled with the generic tank model described in chapter 3.1. The tank B5 is modeled with the model for an evaporator/condenser as explained in chapter 3.2. The tanks B6 and B7 are modeled by an extended generic tank model that includes a heat port to model the cooling. The pumps are modeled with the pump model from the Modelica_Fluid library [4]. To model the pressure drops and the variable heights of the tanks the pipes are modeled with the model of a non-horizontal pipe. The valves are modeled such that they are either open or closed. They are controlled by the Boolean input signal open to indicate if the valve is open. Two models have been setup: One using the StandardWater medium model, i.e., modeling only two phase water flow and one model that uses the medium model for the water–sodium chloride mixture which has been discussed in chapter 4. The controller sketched in chapter 5 is used to model the controller of the batch plant. The Modelica model of the batch plant is shown in figure 12.

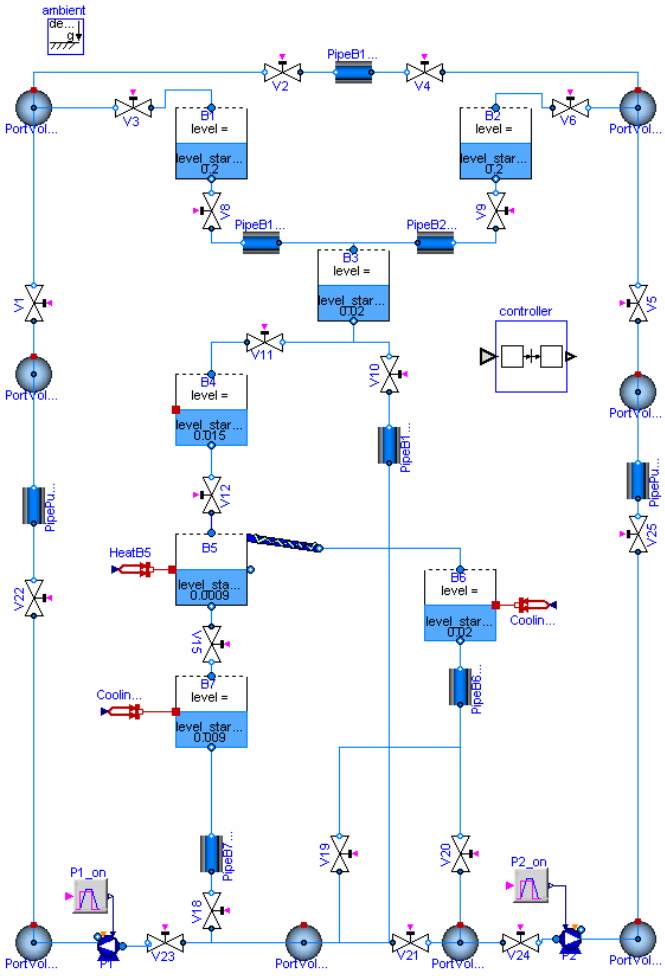


figure 12: Modelica model of the AST batch plant

Modeling this system by pressure drop components only is not possible, since otherwise equations for pressure, composition and specific enthalpy at connectors are missing when two or more connected valves are closed. For this reason, appropriate volumes are introduced at connection points.

The connections from the central controller to the sensors and actuators are not performed graphically because too many signals would need to be connected. Instead, the connections are stated in the textual level with equations: Each valve gets its input if it is open or closed from the controller by e.g.

```
V1.open = controller.sensors.V1;
```

and equally for the other valves. The variables for heating and cooling are Real and not Boolean variables, so the equation looks different to those for the valves. This is shown exemplarily for the heating in tank 5:

```
HeatB5.Q_flow =
  if controller.actuators.T5_Heater
  then 20000 else 0;
```

7 Simulation

It is possible to simulate the batch plant model with the StandardWater medium model completely. Simulation results of the tank levels are shown in figure 13.

For the mixture of water and sodium chloride the simulation stops when one of the pumps should start. Till this unwanted ending the results are plausible and discussed below.

In Figure 14 the fluid levels of all tanks are shown. At the beginning, the level of e.g. tank B5 (black line) stays constant till valve V12 opens. Till this valve closes the level rises quite steeply due to the inflow from tank B4 (magenta line), which shows the same changing just in the opposite way. Afterwards the level in tank B5 keeps rising but slower due to the expansion of the fluid because of the heating. Once some water evaporates, the level of the tank falls as the level of tank B6 (blue crossed line) rises. As soon as the desired concentration is reached the concentrated solution is flowing out of tank B5 into tank B7 (red crossed line) so the level of tank B5 falls steeper. Because of the different diameters of the two tanks B5 and B7 these two curves show different gradients. These results are as expected.

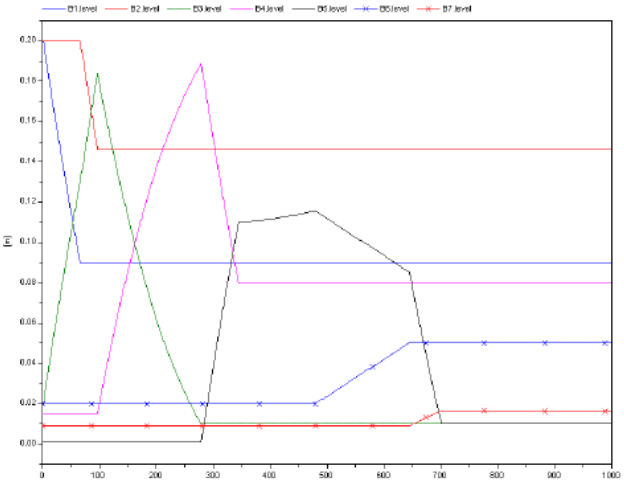


Figure 14: Tank levels of batch plant with water- sodium chloride mixture.

Figure 15 shows in the upper graph the temporal

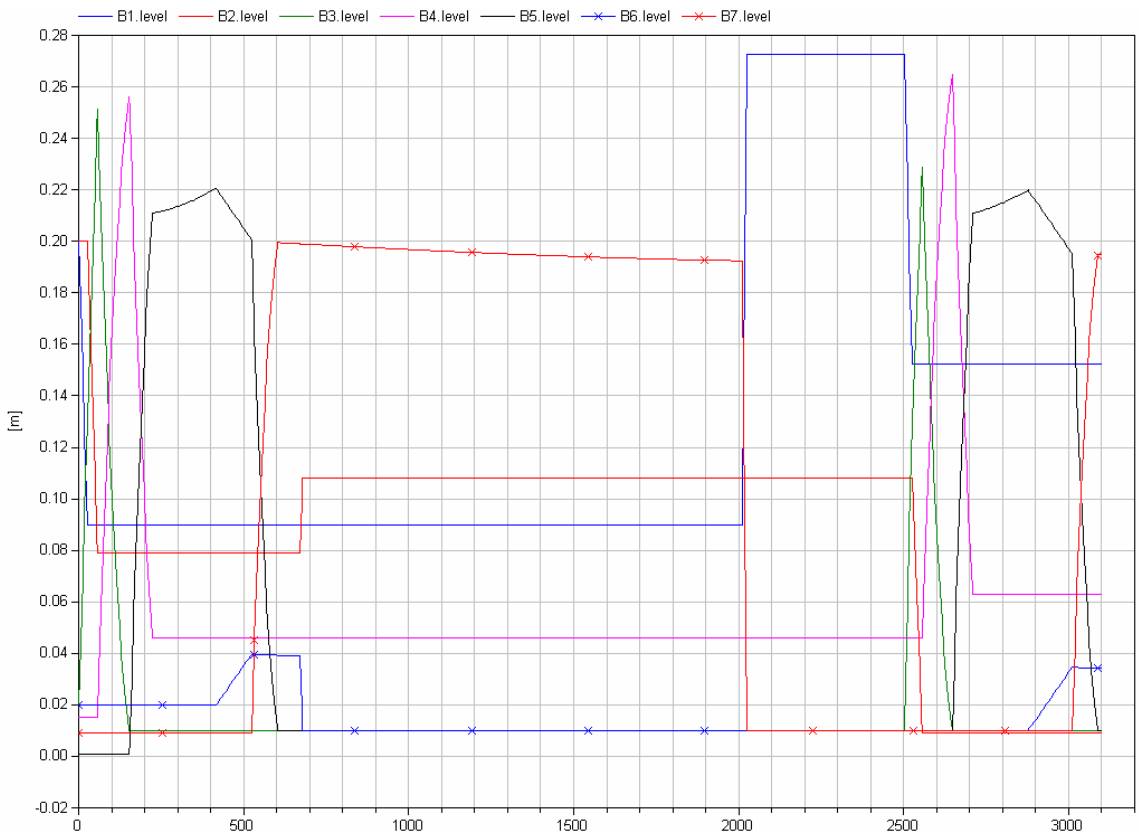


Figure 13: Tank levels of batch plant with StandardWater.

process of the temperature in tank B5 and in the lower image the supplied heat energy is present. When the heat energy changes from 0 to 2000, the temperature starts rising due to the heating. Once the saturation temperature is reached the temperature stays at the boiling temperature which rises only slightly with rising concentration of salt. The supplied energy is then only used for evaporation but not for heating up to higher temperatures than boiling point. This result confirms with reality.

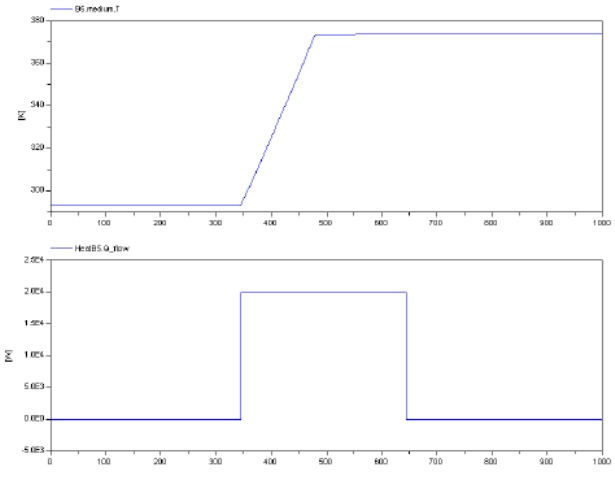


Figure 15: Temporal process of the temperature and the heat flow in tank B5

In Figure 16, the concentration of sodium chloride in tank B3 (blue) and tank B5 (red) are shown over time. For tank B3 the concentration changes when pure water from tank B2 flows into this tank. Once the fluid flow stops, the concentration remains constant. In tank B5 the concentration changes when water is evaporating till the desired concentration is reached and stays constant afterwards.

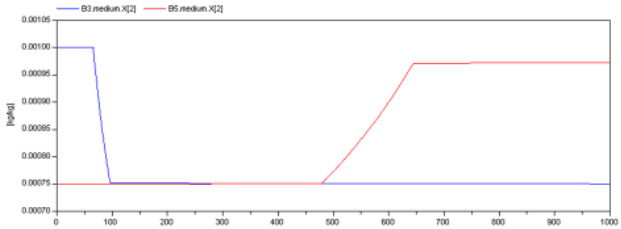


Figure 16: Temporal process of the concentration of NaCl in tanks B3 and B5

All results show qualitatively the expected characteristics.

8 Conclusion and Outlook

The experimental batch plant at AST has been modeled in Modelica. Needed components not available in the Modelica_Fluid library have been implemented. The batch plant model with the Standard-Water medium model, as well as most of the developed components, have been included in version 1.0 Beta 1 of the Modelica_Fluid library and are therefore freely available in open source.

Besides getting the model to work for the whole process duration with the water- sodium chloride mixture, several improvements are desirable: The roughly estimated plant parameters should be more carefully identified and validated with measurement data from the batch plant. The simulation results should be compared with measurement data. The controller should be improved to include exceptional situations such as sensor and actuator failures, as well as stop and shut-down operations. The controller implementation should be made more transparent, which might require extensions to Modelica and tool improvements. Furthermore, it is planned to provide the plant as a benchmark system for tool integration within WP3 of the Network of Excellence HYCON, http://wp3.hycon.bci.uni-dortmund.de/1.0.html.

Acknowledgements

This work was partially funded by the Network of Excellence HYCON, contract number FP6-IST-511368.

References

[1] Kowalewski S., Stursberg O., and Bauer N. (2001): *An Experimental Batch Plant as a Test Case for the Verification of Hybrid Systems*. European Journal of Control 0:1–16.

[2] Chou J.C.S., and Rowe, A.M. (1969): *Enthalpies of aqueous sodium chloride solutions from 32 to 350°F,* Desalination 6: 105-115.

[3] Barthel J., Neueder R., and Meier R. (1998): *Electrolyte data collection Vol.XII Part 3c,* DECHEMA.

[4] Casella F., Otter M., Proells K., Richter C., and Tummescheit H. (2006): *The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks*. Proceedings of the Modelica'2006 conference.

[5] Poschlad K. (2006): *Modellierung einer Batchanlage mit Ablaufsteuerung in Modelica* (2006). Diplomarbeit, Universität Dortmund, Lehrstuhl für Anlagensteuerungstechnik, Prof. Engell.