Out: January 15, 2014
In: January 24, 2014

NTNU – Trondheim
Norwegian University of
Science and Technology

Exercise 2

TTK4130 Modeling and Simulation

Problem 1 (Modelica, Dymola, simple two-tank model)

In this problem, we will implement a model of a two-tank system coupled with a pipe with laminar flow[1], in Modelica/Dymola. See Figure 1.

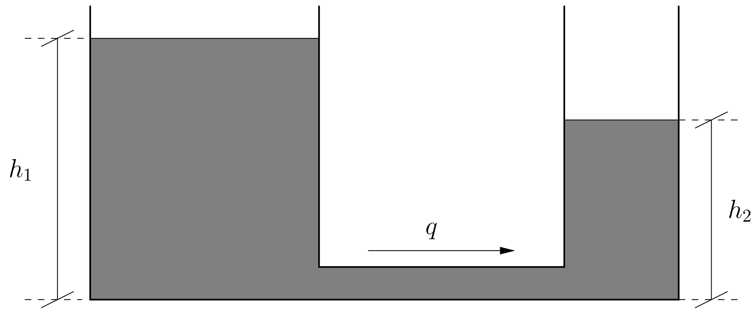

Figure 1: Coupled two-tank system

The mass balance for each tank can be written as (why?)

$$A_1 \frac{\mathrm{d}h_1}{\mathrm{d}t} = -q,$$
$$A_2 \frac{\mathrm{d}h_2}{\mathrm{d}t} = q$$

where A_i is the cross-sectional area of tank i, and h_i is the liquid height. The volume flow rate q is, assuming an incompressible Newtonian fluid flowing through a (long) cylindrical pipe, accurately described by the Hagen-Poiseuille law,

$$q = (p_1 - p_2)\frac{\pi D^4}{128\mu L},$$

where p_i is the bottom pressure of each tank, D is the pipe diameter, μ is the dynamic viscosity and L is the pipe length. Note the sign convention that flow is positive out of tank 1. To couple these equations we need a relation between height and pressure:

$$p_i = \rho g h_i.$$

The values of the parameters can be found from the (incomplete) Modelica model below:

```
model TwoTanks_basic
  // Constants
  constant Real pi = 3.14;
  constant Real g = 9.81;

  // Parameters
  parameter Real A1 =  1.0 "Area of tank 1";
  parameter Real A2 =  2.0 "Area of tank 2";
  parameter Real L =   0.1 "Pipe length";
  parameter Real D =   0.2 "Pipe diameter";
```

[1]The example is adapted from an example in M. Tiller, "Introduction to physical modeling with Modelica", 2001.

```
  parameter Real rho = 0.2 "Fluid density";
  parameter Real mu =  2e−3 "Fluid dynamic viscosity";

  // Variables
  Real p1 "Pressure in tank 1";
  Real p2 "Pressure in tank 2";
  Real h1 "Liquid level in tank 1";
  Real h2 "Liquid level in tank 2";
  Real q "Volume flow rate between tanks";

equation
  // Relation pressure and height

  // Flow between tanks (positive out of tank 1)

  // Mass balances for each tank

end TwoTanks_basic;
```

(a) Implement the two-tank model in Dymola (that is, write in the code above, and add the missing equations in the equation-section). First make a package ('File' → 'New' → 'Package') called TwoTanks, and make a model within the package (right-click the package in the Packages-pane, and choose 'Edit' → 'New class in package' → 'Model') called TwoTanks_basic. Choose 'Window' → 'View' → 'Modelica text' to open the text-view of the model.

How many variables are there? How many equations do you have to implement?

Simulate the model (in the simulation view). Experiment with different initial conditions.

Solution: The equation-part of the model could be implemented as

```
equation
  // Relation pressure and height
  p1 = rho*g*h1;
  p2 = rho*g*h2;

  // Flow between tanks (positive out of tank 1)
  q = pi*D^4/(128*mu*L)*(p1 − p2);

  // Mass balances for each tank
  A1*der(h1) = − q;
  A2*der(h2) = q;
end TwoTanks_basic;
```

There are five variables, so we must implement five equations.

(b) It is desirable that the model contains information about the units of the involved quantities. We could do this by specifying a property for the parameters or variables, such as e.g.

```
parameter Real A1(unit="m2")=1.0 "Area of tank 1";
```

but to help us getting a consistent set of units, the Modelica Standard Library has defined all SI units (open 'Modelica' → 'SIunits' to inspect them). Improve the model above (in a new model, twotanks_SI, in the same package, if you want) with appropriate units from Modelica.SIunits. The code excerpt below should give you some hints (we have also used Modelica.Constants):

```
model TwoTanks_SI
```

```
  import SI = Modelica.SIunits;

  // Constants
  constant Real pi = Modelica.Constants.pi;
  constant Real g = Modelica.Constants.g_n;

  // Parameters
  parameter SI.DynamicViscosity mu =  2e-3 "Fluid dynamic viscosity";

  // Variables
  SI.Length h1 "Liquid level in tank 1";
```

Note that by using the import-statement, we can write SI rather than Modelica.SIunits when we use something from the SIunits library.

Solution:

```
model TwoTanks_SI
  import SI = Modelica.SIunits;

  // Constants
  constant Real pi = Modelica.Constants.pi;
  constant Real g = Modelica.Constants.g_n;

  // Parameters
  parameter SI.Area A1 =  1.0 "Area of tank 1";
  parameter SI.Area A2 =  2.0 "Area of tank 2";
  parameter SI.Length L =   0.1 "Pipe length";
  parameter SI.Length D =   0.2 "Pipe diameter";
  parameter SI.Density rho = 0.2 "Fluid density";
  parameter SI.DynamicViscosity mu =  2e-3 "Fluid dynamic viscosity";

  // Variables
  SI.Pressure p1 "Pressure in tank 1";
  SI.Pressure p2 "Pressure in tank 2";
  SI.Length h1 "Liquid level in tank 1";
  SI.Length h2 "Liquid level in tank 2";
  SI.VolumeFlowRate q "Volume flow rate between tanks";
```

This process could be part of a much larger process, and then it is not practical to model the total process in a single file. By splitting it into parts, it is much easier to get an overview and maintain the overall model, sub-models that are equal need only be modeled once (for instance, tanks), and it is easy to replace/add sub-models. Modelica (and Dymola) is very well suited for this.

(c) What are appropriate connection variables for this type of process? (That is, a process consisting of mass balances and exchange of mass/flow between units.) Several choices can be sensible. In your package, implement a Modelica connector (a type of model) as shown below:

```
connector FlowPort
  import SI = Modelica.SIunits;

  flow SI.VolumeFlowRate q "Volume flow rate from the connection point
                            into the component";
  SI.Pressure p "Thermodynamic pressure in the connection point";
end FlowPort;
```

In connectors, we have "nonflow" variables (effort) which should be equal in the connector, and "flow" variables (prefixed with flow) that should equate to zero. Think voltage, current and Kirchhoffs laws.

Solution: When interconnecting processes of the type we consider here, we need to know the flow, and the "potential for flow" to be able to calculate the flow. The most natural is then perhaps volumetric flow and pressure, as in the code in the example. The product of these has Watt (power) as unit. Alternatives are mass flow instead of volumetric flow (this is used in the Modelica Standard Library), and height/level instead of pressure.

The connector for these type of systems in Modelica Standard Library handles also energy-flow, and fluids (systems) with several components. See Modelica.Fluid.Interfaces.FluidPort.

(d) Implement a tank model (called Tank) and a pipe model (Pipe), and put them together in the following way:

```
model TwoTanks
  Tank Tank1(A=1.0);
  Tank Tank2(A=2.0);
  Pipe Pipe(L=0.1,D=0.2);
equation
  connect(Tank1.flowPort,Pipe.flowPort_a);
  connect(Tank2.flowPort,Pipe.flowPort_b);
end TwoTanks;
```

To help you get started, the Tank-model could be implemented as:

```
model Tank
  // Constants
  constant Real g = Modelica.Constants.g_n;

  // Parameters
  parameter Real A =  1.0 "Area of tank";
  parameter Real rho = 0.2 "Fluid density";

  // Ports
  FlowPort flowPort "fluid flows in or out of tank";

  // Variables
  Real p "Pressure in tank";
  Real h "Liquid level in tank";

equation
  // Relation pressure and height
  p = rho*g*h;

  // Mass balances for each tank
  A*der(h) = -flowPort.q;

  // Set pressure in port
  flowPort.p = p;
end Tank;
```

Solution: The pipe model could look like

```
model Pipe
```

```
  import SI = Modelica.SIunits;

  // Constants
  constant Real pi = Modelica.Constants.pi;

  // Parameters
  parameter SI.Length L =   0.1 "Pipe length";
  parameter SI.Length D =   0.2 "Pipe diameter";
  parameter SI.DynamicViscosity mu =  2e-3 "Fluid dynamic viscosity";

  // Ports
  FlowPort flowPort_a "Port a";
  FlowPort flowPort_b "Port b";

  // Variables
  SI.VolumeFlowRate q "Volume flow rate between tanks";

equation
  // Flow between tanks (positive out of tank 1)
  q = pi*D^4/(128*mu*L)*(flowPort_a.p - flowPort_b.p);

  // Set flow in connectors (positive direction into connector)
  flowPort_a.q = -q;
  flowPort_b.q = q;
end Pipe;
```

The final question is optional:

(e) In this task, we will model the same system using components from the Modelica.Fluid library in the Modelica Standard Library. Make a new model (for instance in the same package as before), and drag-and-drop the models Modelica.Fluid.Vessels.OpenTank (times 2), Modelica.Fluid.Pipes.StaticPipe and Modelica.Fluid.System (contains some parameters common for all sub-models). Connect them by clicking on the connectors.

Then we have two fill in the parameters. For each of the tanks, doubleclick and fill in

- Height of the tank (note; this is not the level)
- Cross sectional area
- Medium. Choose one from the drop-down menu, for instance "Water: Simple liquid water medium".
- Set 'use_portsData' to 'False'.

For the pipe, doubleclick to fill in

- Length
- Diameter
- Medium (same as for tanks)
- Flow model. It will work if you do nothing (then the model "DetailedPipeFlow" will be used). If you choose something else than the default (for instance: "NominalLaminarFlow"), you have to fill in some of the parameters of the flow model.

Simulate and compare.

Solution:

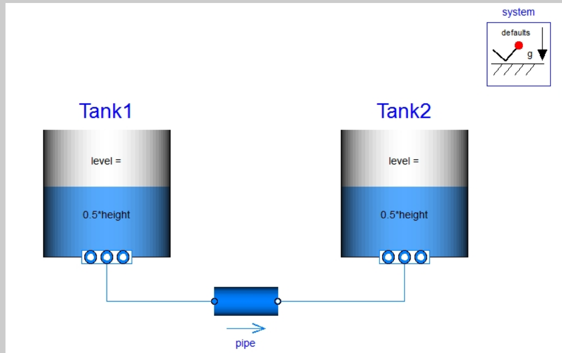

Figure 2: Two-tank system implemented by using components from Modelica.Fluid.

Problem 2 (Positive real transfer functions)

(a) Are the transfer functions

$$H_1(s) = \frac{1}{1+Ts}$$
$$H_2(s) = \frac{s}{s^2+\omega_0^2}$$

positive real?

Solution: The transfer function $H_1(s)$ has a pole in $s = -1/T < 0$ (and thus no poles in $\mathrm{Re}\,[s] > 0$),

$$\mathrm{Re} H_1\,(j\omega) = \mathrm{Re}\frac{1-j\omega T}{(1+j\omega T)\,(1-j\omega T)} = \frac{1}{1+\omega^2 T^2} > 0,$$

and there are no poles on the imaginary axis (or in infinity), which implies that $H_1(s)$ is positive real.

The transfer function $H_2(s)$ has poles in $s = \pm j\omega_0$. We have that

$$\mathrm{Re} H_2\,(j\omega) = \mathrm{Re}\frac{j\omega}{\omega_0^2-\omega^2} = 0, \quad \omega \neq \omega_0,$$

and at the poles,

$$\mathrm{Res}_{s=\pm j\omega_0} H_2(s) = \lim_{s\to\pm j\omega_0} (s \mp j\omega_0)\, H_2(s) = \lim_{s\to\pm j\omega_0} \frac{s}{s \pm j\omega_0} = \frac{1}{2},$$

which shows that $H_2(s)$ is positive real.

(b) Assume that $c > 0$ and $b > 0$ are given constants. For which a is the transfer function

$$H_3(s) = \frac{s+a}{(s+b)(s+c)}$$

positive real?

Solution: The transfer function $H_3(s)$ has poles in $s = -b$ and $s = -c$. We have that

$$\begin{aligned} H_3(j\omega) &= \frac{j\omega + a}{(j\omega + b)(j\omega + c)} \\ &= \frac{(j\omega + a)(b - j\omega)(c - j\omega)}{(b^2 + \omega^2)(c^2 + \omega^2)} \\ &= \frac{abc + \omega^2(b + c - a) + j\left[\omega(bc - ab - ac) - \omega^3\right]}{(b^2 + \omega^2)(c^2 + \omega^2)} \end{aligned}$$

which means that

$$\mathrm{Re} H_3(j\omega) = \frac{abc + \omega^2(b + c - a)}{(b^2 + \omega^2)(c^2 + \omega^2)}.$$

Since the denominator is always positive, it sufficient to require that the nominator,

$$abc + \omega^2(b + c - a) > 0, \quad \forall \omega.$$

We see that $\omega = 0$ requires that $a > 0$ (since b and c are positive), and $\omega \to \infty$ requires that $b + c - a > 0$. That is, we must have

$$0 < a < b + c$$

for $H_3(s)$ to be positive real.

(c) For which $a \geq 0$ is the transfer function

$$H_4(s) = \frac{s^2 + a^2}{s\left(s^2 + \omega_0^2\right)}$$

positive real?

Solution: The transfer function $H_4(s)$ has three simple poles at the imaginary axis, in $s = 0$ and $s = \pm j\omega_0$. From

$$H_4(j\omega) = -j\frac{a^2 - \omega^2}{\omega\left(\omega_0^2 - \omega^2\right)},$$

we see that $\mathrm{Re} H_4(j\omega) = 0$ for all $j\omega$ that are not poles. The residues in $s = 0$ and $s = \pm j\omega_0$ are

$$\mathrm{Res}_{s=0} H_4(s) = \lim_{s \to 0} s H_4(s) = \frac{a^2}{\omega_0^2} > 0,$$

$$\mathrm{Res}_{s=\pm j\omega_0} H_4(s) = \lim_{s \to \pm j\omega_0} (s \mp j\omega_0) H_4(s) = \frac{\omega_0^2 - a^2}{2\omega_0^2},$$

which both are positiv if and only if $a < \omega_0$. That is, $H_4(s)$ is positive real if and only if $a < \omega_0$.

(d) The state-space model for the transfer function $H_1(s)$ is

$$T\dot{y} = -y + u.$$

Find a storage function $V(y) \geq 0$ which can be used to show passivity for the system $u \mapsto y$.

Solution: Consider the quadratic storage function

$$V = \frac{1}{2} T y^2.$$

We have that

$$\dot{V} = T\dot{y}y = yu - y^2,$$

which implies that $u \mapsto y$ is passive.

(e) Given the transfer function

$$H(s) = \frac{(s + z_1) \dots (s + z_m)}{s\,(s + p_1) \dots (s + p_n)}$$

where $\mathrm{Re}[p_i] > 0$, $\mathrm{Re}[z_i] > 0$ and $n > m$. Show that $H(s)$ is positive real if and only if $\mathrm{Re}[H(j\omega)] \geq 0$ for all $\omega \neq 0$.

Solution: $H(s)$ has a simple pole on the imaginary axis in $s = 0$, while the remaining poles are in $s = -p_i < 0$. The residue in $s = 0$ is

$$\mathrm{Res}_{s=0} H(s) = \lim_{s \to 0} sH(s) = \frac{z_1 \cdots z_m}{p_1 \cdots p_n}$$

Since poles and zeros are either real or complex conjugated, $(z_1 \cdots z_m)/(p_1 \cdots p_n)$ is real and positive, which shows the result.