Out: March 29, 2019, 8:00
Deadline: April 25, 2019, 20:00

NTNU – Trondheim
Norwegian University of Science and Technology

Assignment 11
TTK4130 Modeling and Simulation

Problem 1 (Transmission lines, PDEs, discretization methods. 50 %)

NB: This is a computer exercise, and can therefore be solved in groups of 2 students. If you do so, please write down the name of your group partner in your answer.

The model for a hydraulic transmission line can be written as the partial differential equation (PDE):

$$\frac{\partial p(x,t)}{\partial t} = -cZ_0 \frac{\partial q(x,t)}{\partial x} \tag{1a}$$

$$\frac{\partial q(x,t)}{\partial t} = -\frac{c}{Z_0}\frac{\partial p(x,t)}{\partial x} - \frac{F(q(x,t))}{\rho_0}, \tag{1b}$$

where the sonic velocity c $[m/s]$ and the line impedance Z_0 $[kg/(m^4 s)]$ are defined as

$$c = \sqrt{\frac{\beta}{\rho_0}} \tag{2}$$

$$Z_0 = \frac{\rho_0 c}{A} = \frac{\sqrt{\rho_0 \beta}}{A}, \tag{3}$$

where β is the fluid bulk modulus (the ratio between the pressure increase and the relative decrease of volume) $[N/m^2]$, ρ_0 is the fluid density $[kg/m^3]$, A is the cross sectional area $[m^2]$, and $F(q)$ is the friction force as a function of the volume flow $[N]$.

The book outlines two different methods for simulating the transmission line model. The first one is described in section 4.5.10 to 4.5.16 in the book, and is based on approximating the irrational transfer functions of the PDE model by rational transfer function. These approximations are necessary since irrational transfer functions cannot be simulated directly (e.g. in Simulink).

On the other hand, the second method, which is explained in section 4.6 and will be used in this exercise, is based on discretizing the spatial dimension into a set of small volumes called "finite volumes" (see Figure (1)).

Inside these volumes, the pressure and volume flow are assumed constant for a given time, and a Helmholtz resonator model -or equivalently a mass and moment balance- is used to relate the values of these variables at each "finite volume".

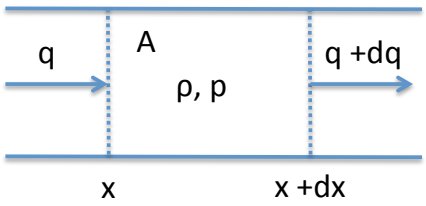

Figure 1: Finite volume element for hydraulic transmission line.

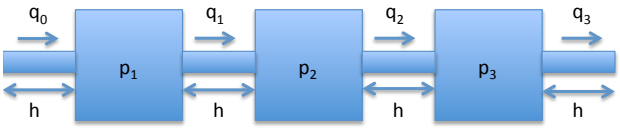

Figure 2: A chain of interconnected Helmholtz resonators ducts of length h representing a transmission line in admittance form.

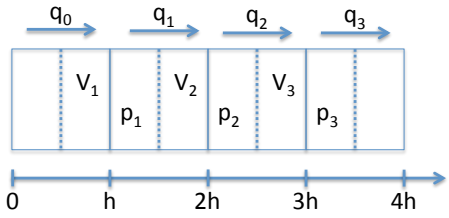

Figure 3: Spatial discretization of a transmission line with pressure inputs at both sides to get a chain of Helmholtz resonators.

This approach will lead to a state-space representation of the system, where the volume flows at each end of the transmission line are the inputs, and the pressures at each end will be the outputs.

In this exercise, the transmission line of length L is divided into n smaller finite "volumes" with length $h = \frac{L}{n}$, as shown in Figure (2) and Figure (3).

Furthermore, for the first part of this problem, we assume a simple linear friction model in (1b), which is given by $F = \rho_0 B q$, where $B = \frac{8\nu_0}{r_0^2}$.

Hint for the whole problem: Read sections 4.5 and 4.6 in the book.

(a) Set up a model for the system by setting up a mass and moment balance for each finite volume.
Hint: Read sections 4.6.1 to 4.6.5 in the book.

Solution: The model is given in section 4.6.5:

$$\dot{p}_i = \frac{c^2\rho_0}{Ah}\left(q_{i-1} - q_i\right), \quad i = 1, \ldots, N$$
$$\dot{q}_{i-1} = \frac{A}{h\rho_0}\left(p_{i-1} - p_i\right) - Bq_{i-1}, \quad i = 2, \ldots, N$$
$$q_0 = q_{in}, \quad q_N = q_{out}.$$

Note that these equations can also be derived by using the finite differences

$$\frac{\partial q_i}{\partial x} = \frac{q_i - q_{i-1}}{h} + O(h)$$
$$\frac{\partial p_{i-1}}{\partial x} = \frac{p_i - p_{i-1}}{h} + O(h).$$

(b) We will now write the model for the transmission line as a linear state-space model.

As mentioned above, the volume flow into the first volume (q_0) and the volume flow out of the last volume (q_N), which constitute the boundary conditions of the transmission line PDE, will be the inputs of the linear state-space model.

In order to do so, gather all variables in the state variable $\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} p_1 & q_1 & p_2 & q_2 & \cdots & q_{N-1} & p_N \end{bmatrix}^T ,$$

and write the model as

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} , \end{aligned}$$

where the inputs and outputs are

$$\mathbf{u} = \begin{bmatrix} q_0 \\ q_N \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} p_1 \\ p_N \end{bmatrix} .$$

Find the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ for the n finite volumes.

Solution: The matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & -M & 0 & 0 & \dots & 0 & 0 \\ N & -B & -N & 0 & \dots & 0 & 0 \\ 0 & M & 0 & -M & \dots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & \dots & M & 0 & -M & 0 \\ 0 & 0 & \dots & 0 & N & -B & -N \\ 0 & 0 & \dots & 0 & 0 & M & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} M & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & -M \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

where M and N are given as

$$M = \frac{c^2 \rho_0}{Ah} = \frac{\beta}{Ah}, \quad N = \frac{A}{h\rho_0}.$$

Note the band pattern of $\mathbf{A}$ (the matrix is tridiagonal). Also note how the rows repeat, i.e. $\mathbf{A}$ can be written as

$$\mathbf{A} = \begin{bmatrix} 0 & -M & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ & \mathbf{A_1} & & & \dots & & \mathbf{0} & & \\ & \vdots & & & \ddots & & \vdots & & \\ & \mathbf{0} & & & \dots & & \mathbf{A_1} & & \\ 0 & 0 & 0 & 0 & \dots & 0 & N & -B & -N \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & M & 0 \end{bmatrix},$$

where

$$\mathbf{A}_1 = \begin{bmatrix} N & -B & -N & 0 \\ 0 & M & 0 & -M \end{bmatrix}.$$

(c) Implement the transmission line model in Matlab/Simulink. Use the parameters values:

Parameter	Symbol	Value	Unit
Length	L	20.0	m
Bulk modulus	β	$1.96 \cdot 10^9$	Pa
Density	ρ_0	1000	kg/m^3
Pipe radius	r_0	$6.0 \cdot 10^{-3}$	m
Kinematic viscosity	ν_0	$8.0 \cdot 10^{-5}$	m^2/s

You may assume that the inputs q_0 and q_N are given by step functions.

Add your Matlab scripts and/or a figure of your Simulink diagrams to your answer (PDF file).

Hint: One way to implement this model involves using a state-space *block in Simulink, with inputs and outputs as given in part b. and as shown in Figure 4. If you do so, you may use a mask over this subsystem or a Matlab script for setting the parameter and variable values. This approach is convenient since the inputs can be implemented by* step *blocks in Simulink. Nevertheless, feel free to implement the transmission line model as you see fit.*

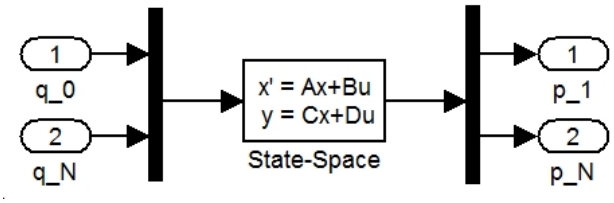

Figure 4: State-space model in Simulink.

Solution: One of infinite many correct solutions can involve a Matlab script for initialization and a Simulink diagram as shown below:

```
area = r0^2*pi;
h = L/n;
M = beta/(area*h);
N = area/(h*rho0);
B = 8*nu0/r0^2;
x0 = zeros(2*n-1,1);
mat = [M 0 -M 0; 0 N -B -N];
A = [0 -M zeros(1,(2*n-3)); N -B -N zeros(1,(2*n-4))];
x0(1:2) = [p0; q0];
for i = 1:1:n-2
addmat = [zeros(2,2*i-1) mat zeros(2,2*(n-2-i))];
A = [A;addmat];
x0((2*(i+1)-1):2*(i+1)) = [p0; q0];
end
x0(2*n-1) = p0;
A = [A; zeros(1, (2*n-3)) M 0];
B = [M 0 ; zeros((2*n-3),2); 0 -M];
C = [1 zeros(1, 2*n-2); zeros(1, 2*n-2) 1];
D = zeros(2,2);
```

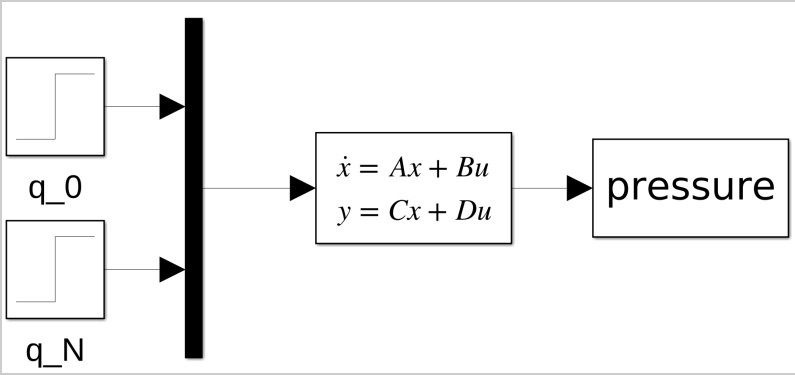

Figure 5: Simulink diagram that simulates the transmission line model.

(d) Simulate the system for 5 seconds and for $n = 5, 10$ and 40 using the inputs

$$q_0 = \begin{cases} 0 & t < 1s \\ 0,001 & t \geq 1s \end{cases}, \qquad q_N = \begin{cases} 0 & t < 2s \\ 0,001 & t \geq 2s \end{cases}.$$

Add plots for p_1 and p_N as a function of time to your answer (PDF file), where you compare the results for different values of n. Comment on the results.

Furthermore, add Bode plots of the transfer functions from q_0 to p_1 and to p_N for the different values of n. Comment on the plots.

Solution: The simulations are shown in Figure 6 and 8.

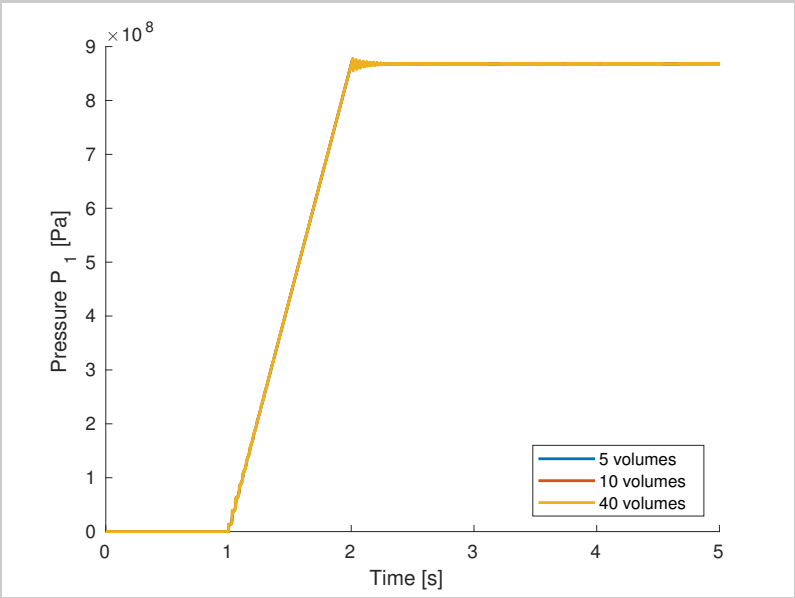

Figure 6: Simulations results for $p1$.

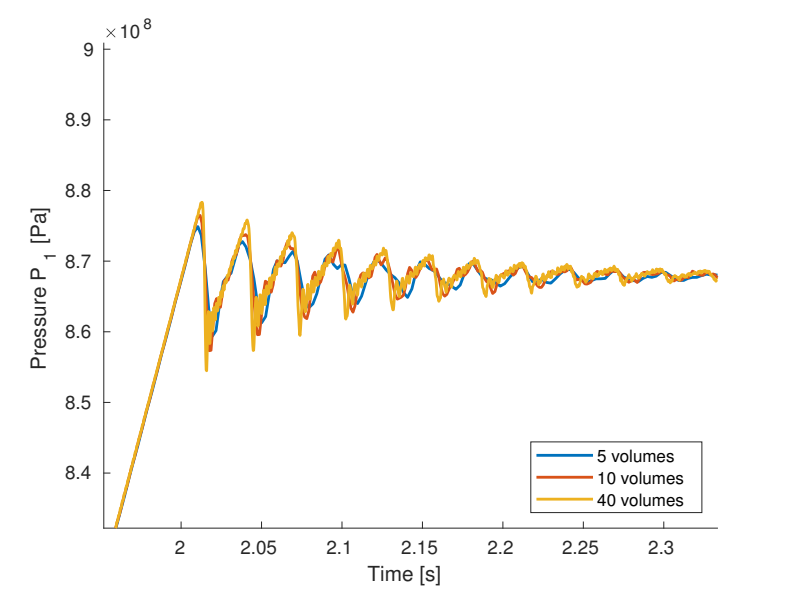

Figure 7: Simulations results for $p1$. Close up: Oscillations.

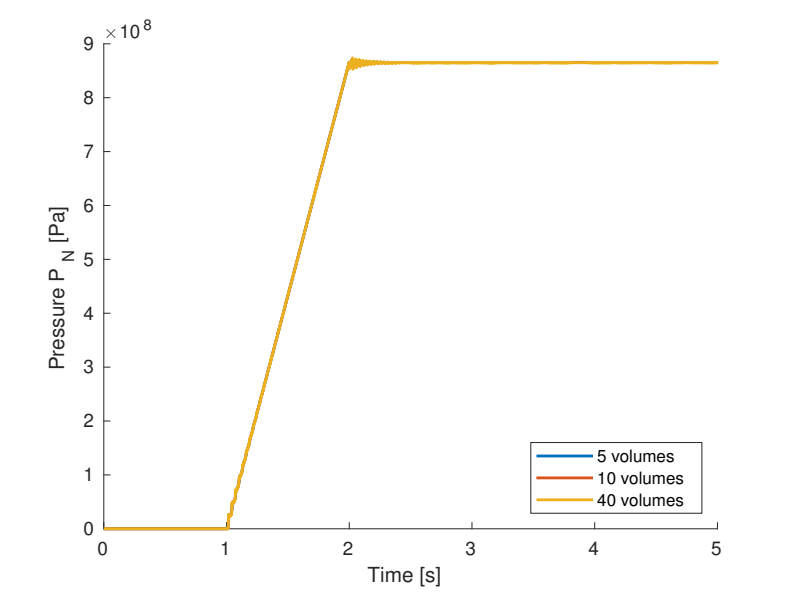

Figure 8: Simulations results for $p1$.

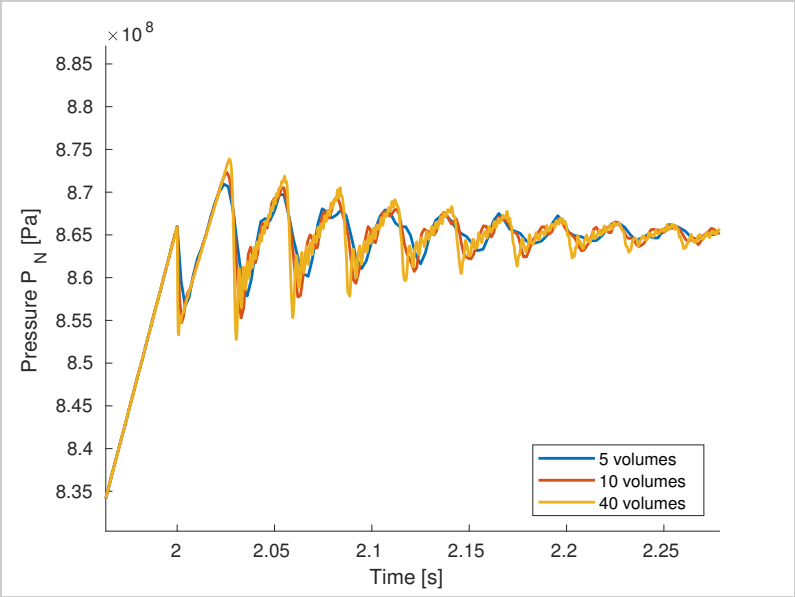

Figure 9: Simulations results for $p1$. Close up: Oscillations.

When the inflow increases without the outflow increasing, then the pressure increases. Pressure stops increasing once the outflow matches the inflow. In addition to these macroscopic effects, there are oscillations corresponding to pressure waves going back and forth in the transmission line. This is shown in Figure 7 and Figure 9. We observe that for low values of n, the pressure waves are smoothed out.

The Bode plots are shown in Figure 10 and 11.

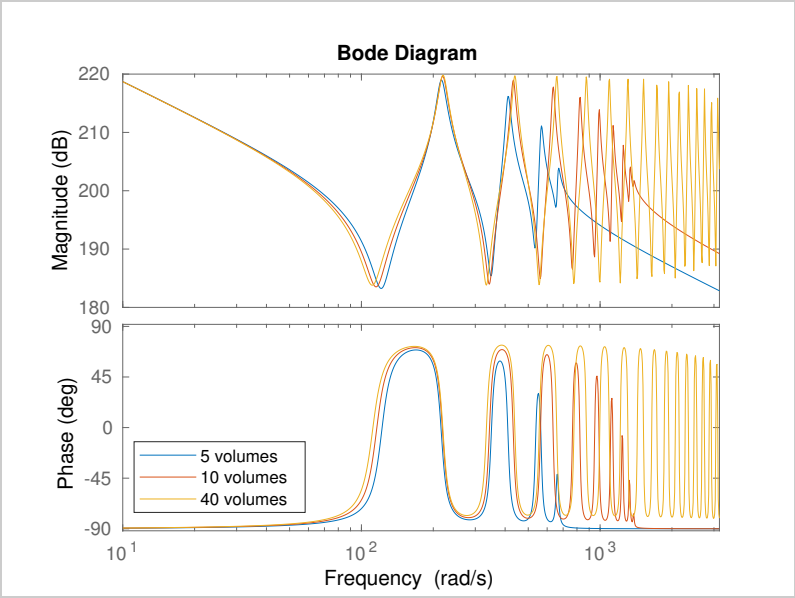

Figure 10: Frequency response from q_0 to p_1.

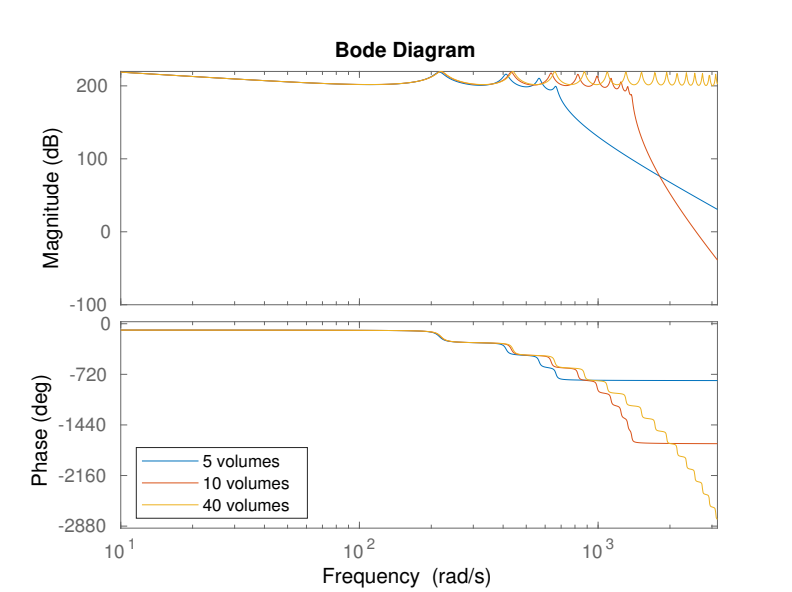

Figure 11: Frequency response from q_0 to p_N.

There is no delay (other than a phase-shift) from the inflow to the pressure in the first volume element, while there is a large delay from the inflow to the pressure in the last volume element. Furthermore, we observe resonances in each Bode plot. The number of resonances depend on the number of finite volume elements.

We will now investigate another friction model, which includes linear friction and diffusion. Hence, the partial differential equation for the volume flow is now

$$\frac{\partial q}{\partial t} = -\frac{A}{\rho_0}\frac{\partial p}{\partial x} + \nu_0\left(\frac{\partial^2 q}{\partial x^2} - \frac{8}{r_0^2}q\right). \tag{4}$$

(e) Use the finite difference of first order

$$\frac{\partial^2 q_i}{\partial x^2} = \frac{q_i - 2q_{i-1} + q_{i-2}}{h^2} + O(h).$$

to discretize (4) in a similar manner as done in part b.
Finally, express the discretized transmission line system as a linear state-space model.

Solution: The model becomes

$$\dot{p}_i = \frac{c^2\rho_0}{Ah}\left(q_{i-1} - q_i\right), \quad i = 1, \ldots, N$$

$$\dot{q}_{i-1} = \frac{A}{h\rho_0}\left(p_{i-1} - p_i\right) - \left(\frac{2\nu_0}{h^2} + \frac{8\nu_0}{r_0^2}\right)q_{i-1} + \frac{\nu_0}{h^2}\left(q_{i-2} + q_i\right), \quad i = 2, \ldots, N$$

$$q_0 = q_{in}, \quad q_N = q_{out}$$

This can be written on state-space form,

$$\dot{\mathbf{x}} = \mathbf{A}_v\mathbf{x} + \mathbf{B}_v\mathbf{u},$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u},$$

where $\mathbf{A}_v$ is given as

$$\mathbf{A}_v = \begin{bmatrix} 0 & -M & 0 & 0 & \cdots & & \mathbf{0} & \\ N & -d & -N & k & \cdots & & & \\ & \mathbf{0} & \mathbf{A}_2 & & \cdots & & \mathbf{0} & \\ & \vdots & & & \ddots & & \vdots & \\ & \mathbf{0} & & & \cdots & \mathbf{A}_2 & \mathbf{0} & \\ & & & & & M \;\; 0 & -M & 0 \\ & \mathbf{0} & & & \cdots & k \;\; N & -d & -N \\ & & & & & 0 \;\; 0 & M & 0 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} M & 0 & -M & 0 & 0 \\ k & N & -d & -N & k \end{bmatrix}$$

$$M = \frac{\beta}{Ah},\; N = \frac{A}{h\rho_0},\; d = \frac{2\nu_0}{h^2} + \frac{8\nu_0}{r_0^2},\; k = \frac{\nu_0}{h^2},\; \mathbf{u} = \begin{bmatrix} q_0 \\ q_N \end{bmatrix}$$

and $\mathbf{B}_v$ as

$$\mathbf{B}_v = \begin{bmatrix} M & 0 \\ k & 0 \\ \vdots & \vdots \\ 0 & k \\ 0 & -M \end{bmatrix}$$

and $\mathbf{C}$ and $\mathbf{D}$ as in part b.

Problem 2 (The Modelica language, replaceable, redeclare. 50 %)

NB: This is a computer exercise, and can therefore be solved in groups of 2 students. If you do so, please write down the name of your group partner in your answer.

The Modelica language offers several configuration management features that allow us to define new classes by modifying exiting ones in different ways. One of such features is given by the keyword extends, which was used in assignment 3 problem 1. By means of this keyword, one can expand an existing class into a new one by adding new variables, parameters and equations.

Another important way of defining new classes is by replacing particular components (instances of some another class) with other components. This is achieved by using the keywords replaceable and redeclare. In such case, the components in the original class are declared "replaceable", and in the definition of the new class these components are "redeclared". The main advantage of using the keywords replaceable and redeclare is that new classes can be created with the same connecting setup as the original class, i.e. one does not need to reconnect anything.

The main objective of this problem is to show how the keywords replaceable and redeclare work.

Download the Modellica package TankSystem.mo, which has been uploaded together with this file.

(a) Explain what the classes of this package are. In particular, explain which roll do these classes play in order to implement the tank model TankLevelPIDSystem.

Make a diagram that illustrates the different components and connections of this model.

Hint: The names of the classes and variables should give you a clue.

Solution: There are two open tanks: tank 1 and tank 2. The outlet of a liquid source goes into tank 1 and the outlet of tank 1 goes into tank 2. For each tank, its level is measured and sent to a PID controller, which controls the position of the outlet valve.
The closed-loop system makes sure that the level of the tanks are kept at their given references.

(b) In both the function LimitValue and the class LiquidSource, the keyword algorithm is used.
What is this keyword used for? Compare it to the keyword equation.

Solution: The keyword algorithm defines an environment for "usual" programming: Declarations are defined, and they are executed by the program in the given order.
On the other hand, the keyword equation defines an environment with equations that the program will determine how to solve depending on the overall structure of the model, the selected solver and parameters. In particular, the order of the equations is irrelevant.

(c) What does the function LimitValue do?

Solution: It implements a saturation function. Note that the order of the inputs matter.

In the model TankLevelPIDSystem, we use classical PID-controllers to control the level of the tanks. In order to test different control solutions, we would like to implement another model, where PID-controllers with bounded derivative and integral effect are used.

(d) Implement PID-controllers with bounded derivative and integral effect as a block class.
Do not modify the other classes.
Remember that the transfer function for such a controller is given by

$$H(s) = \frac{u}{e}(s) = \beta K \frac{1 + T_i s}{1 + \beta T_i s} \frac{1 + T_d s}{1 + \alpha T_d s}, \tag{5}$$

where e is the deviation, u is the control input and $K > 0$, $T_i > T_d > 0$, $\beta \geq 1 \geq \alpha > 0$.
Add your implementation to your answer (PDF file).
Hint 1: See how the class PIDController is implemented.
Hint 2: A realization of such a controller is given in assignment 2 problem 3.b.

Solution:

```
block PIDBoundedController
  extends TankSystem.BaseController;
  parameter Real a = 1;
  parameter Real b = 1;
equation
  a * Td * der(x1) + x1 = (a - 1) * error;
  b * Ti * der(x2) + x2 = (b - 1) / a * (error + x1);
  controlOutput = K * ((error + x1) / a + x2);
end PIDBoundedController;
```

(e) Finally, use the keywords replaceable and redeclare to define a new model, where both classical PID-controllers of TankLevelPIDSystem are replaced by PID-controllers with bounded derivative and integral effect.
Use the same values for K, T_i and T_d as in the original model, and the values $\alpha = 1$ and $\beta = 100$.
Add your implementation to your answer (PDF file).
Finally, simulate both models for half an hour and enclose a plot to your answer (PDF file), where the levels of both tanks for the different controllers are compared.

Hint 1: Read sections 2.5 in "Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica".

Hint 2: The model TankLevelPIDSystem only requires minimal modifications, and the definiton of the new model can be done in one line.

Solution: Modified TankLevelPIDSystem model:

```
//... code...
  replaceable TankSystem.PIDController levelController1(ref = 0.25, K = 10,
  Ti = 500, Td = 10);
  replaceable TankSystem.PIDController levelController2(ref = 0.40, K = 10,
  Ti = 500, Td = 10);
// ... code ...
```

New model:

```
model TankLevelPIDBoundedSystem = TankSystem.TankLevelPIDSystem(
redeclare TankSystem.PIDBoundedController levelController1(ref = 0.25, K = 10,
Ti = 500, Td = 10, a = 1, b = 100),
redeclare TankSystem.PIDBoundedController levelController2(ref = 0.40, K = 10,
Ti = 500, Td = 10, a = 1, b = 100));
```

The results for both controller systems are shown in Figure 12.

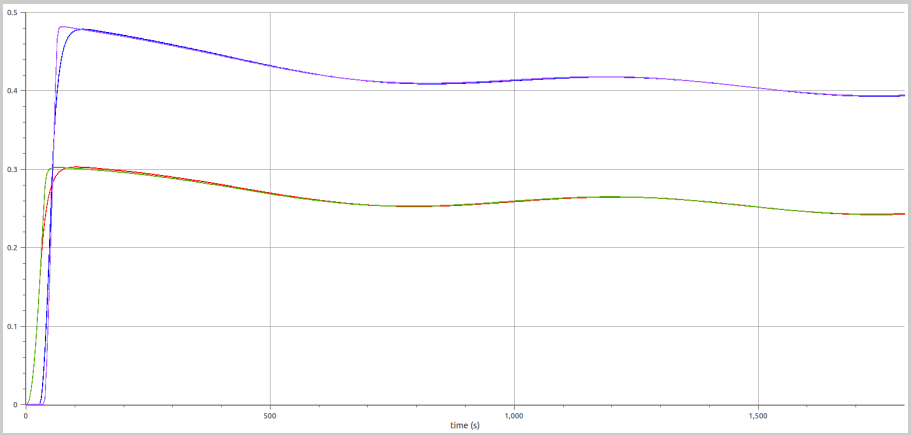

Figure 12: Results for classical PID controller: Level of tank 1 (red) and tank 2 (blue). Results for PID controller with bounded effects: Level of tank 1 (green) and tank 2 (purple).

We will now implement a new tank model, where the level of the tank or the temperature of the fluid inside it can be controlled.

As in the previous models, this new model will consist of two tanks: Tank 1 and tank 2, where the outlet flow of tank 1 goes into tank 2. However, each tank will now have its own liquid source. The PID-controller for tank 1 will still control the tank level, while the PID-controller for tank 2 will control the temperature of the fluid inside it. Moreover, tank 1 and 2 are in contact, and there will be some heat transfer between them.

Since the balance laws and abstraction required for this model are not the main objective of this exercise, they have been done for you.

Download the file AdditionalClasses.txt, which has been uploaded together with this file. Add the code there to the TankSystem package (replace the OpenTank file with its new version).

The block OpenTankWithTemperature extends the block OpenTank with an energy balance that gives the temperature of the fluid inside the tank. The block ContactSurface and the connector HeatSurface are used to model the heat transfer between two tanks in contact.

However, in order to implement the new model a liquid source that has the temperature of the liquid as a variable is necessary, as well as a suitable connector.

(f) Implement the block LiquidSourceWithTemperature and the connector LiquidFlowWithTemperature based on the block LiquidSource and the connector LiquidFlow, respectively. Use the keywords extends, replaceable and redeclare depending on the situation. For simplicty, assume that the liquid out of the source has constant temperature.

Add your implementations to your answer (PDF file).

Hint 1: Study the handed classes for hints on the implementation and required variable names.

Hint 2: The block LiquidSource requires a slightly modification.

Solution: The LiquidSource block was slightly modified:

```
// ... code...
  replaceable connector LiquidConnector = TankSystem.LiquidFlow;
  LiquidConnector qOut;
// ... code...
```

The LiquidSourceWithTemperature block is defined by extending (extends) and redeclaring the connector LiquidFlow (declare):

```
block LiquidSourceWithTemperature
  extends TankSystem.LiquidSource(redeclare connector LiquidConnector
  = TankSystem.LiquidFlowWithTemperature);
  parameter Real T = 298;
equation
  qOut.T = T;
end LiquidSourceWithTemperature;
```

The LiquidFlowWithTemperature connector is implemented by extending the LiquidFlow connector:

```
connector LiquidFlowWithTemperature
  extends LiquidFlow;
  Real T;
end LiquidFlowWithTemperature;
```

(g) Implement the new model, where the level of tank 1 is controlled and the temperature of tank 2 is controlled, as described above. You can choose which type of PID controller to use.

Connect the different blocks using the graphical interface, and add a figure with the connected

model to your answer.

Use the same values as before for the parameters that were present in the previous models, except for the initial level of the tanks, which should be set equal to 0.1 m, and the reference for the temperature in tank 2, which should be set to 323 K.

Furthermore, set the initial temperature of the liquid inside both tanks and the liquid source of tank 1 to 298 K, and the temperature of the liquid source of tank 2 to 348 K.

For the heat transfer between the tanks, set the contact surface equal to 1 m^2 and the heat transfer coefficient equal to 0.5. Furthermore, set the specific heat capacity of the fluid to 0.004 $J\,kg^{-1}$.

Simulate the new model for half an hour, and add two plots to your answer: one with the level of both tanks, and another plot with their temperatures.

Hint 1: See `TankLevelPIDSystem` *for inspiration.*

Hint 2: Make sure that `measureLevel` *is true for Tank 1, but false for Tank 2.*

Solution:

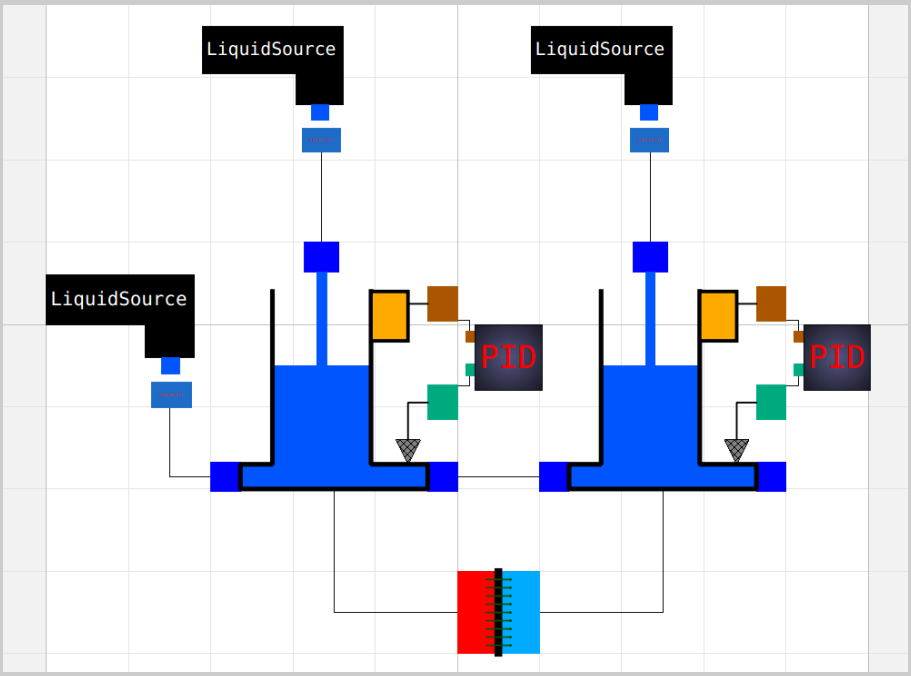

Figure 13: The final system should look something like this.

For this particular solution, a bounded and a classical PID-controller were chosen for tank 1 and 2, respectively. The simulation results are shown in Figure 14 and Figure 15.

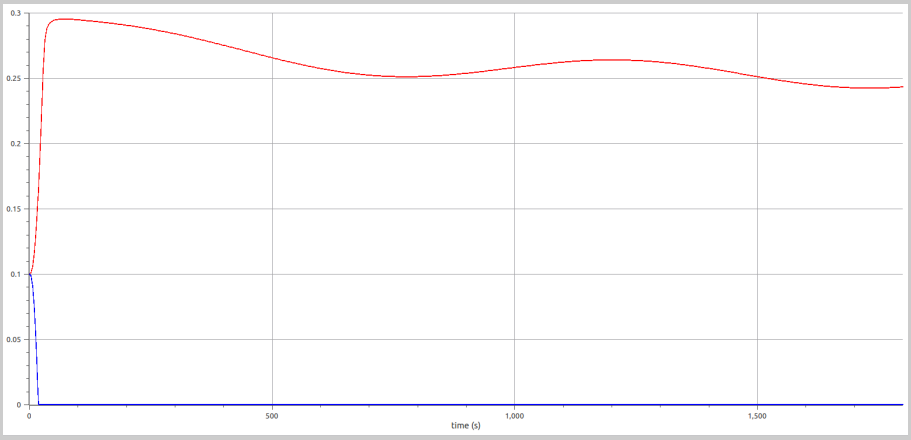

Figure 14: Level of tank 1 (red) and tank 2 (blue).

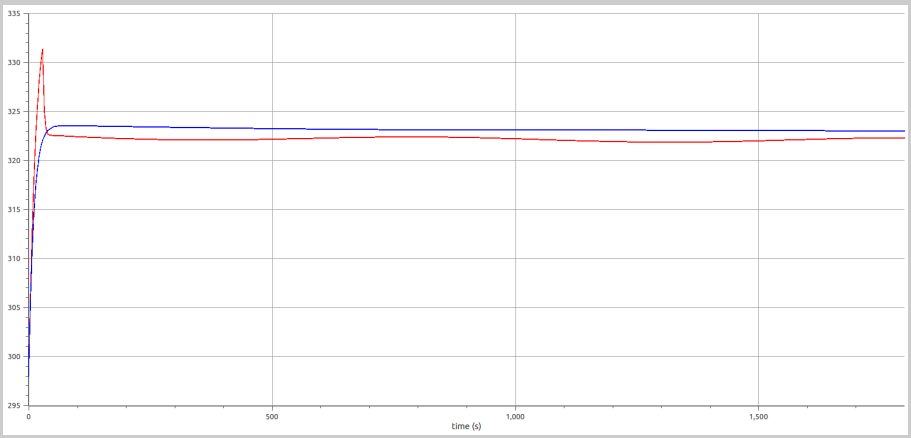

Figure 15: Temperature of tank 1 (red) and tank 2 (blue).

(h) **(Optional)** Custom the different blocks and connectors by adding icons.

Solution: See Figure 13 for an example.