ISSN 0280–5316
ISRN LUTFD2/TFRT--5653--SE

Modeling and Control of a Paper Dryer Section Using Modelica™

Alessandro Pontremoli

Department of Automatic Control
Lund Institute of Technology
November 2000

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
November 2000

Document Number
ISRN LUTFD2/TFRT--5653--SE

Author(s)
Alessandro Pontremoli

Supervisor
Björn Wittenmark

Sponsoring organisation

Title and subtitle
Modeling and Control of a Paper Dryer Section Using Modelica™
(Modellering och reglering av torksektionen i en pappersmaskin)

Abstract

The main purpose of this project has been to model one steam heated cylinder, which was taken as part of a drying section, moreover the design and structure of an existing process has been taken in consideration.

To develop this work Modelica™ software has been used. Particular attention has been paid to reproduce the two time constants behavior of the dynamics in the pressure control loop. This behavior has been observed in existing industrial processes. Other dynamics in the system include; fluid dynamics concerning momentum flow, convective heat flow and pressure, heat dynamics concerning heat flow and temperature inside the cylinder shell has been included.

The basic control volume, flow and medium models are all inherited from the “ThermoFlow” library, which is still under development in this department.

The control is done via a PI controller, the lambda tuning method is applied to tune it. The performance of the whole system was improved. The main contribution of his thesis is to link previous works, which concern paper web modeling and others that concern modeling of thermo–hydraulic systems. The project has been done in collaboration with the ABB company.

Key words
Paper drying, Thermo–hydraulic systems, Modeling, Object–oriented

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
68

Recipient’s notes

Security classification

Acknowledgments

I want to express my sincere gratitude to everyone who has helped and supported me during this work. Even more, I want to thank everyone who supported and encouraged me during the past years.

I want to give special thanks to:

- My Swedish "Super Advisor" Prof. Björn Wittenmark, who has been providing and supporting me through all this work with precious suggestions from the height of his long experience.
- The Ph.D. students Hubertus Tummescheit and Jonas Eborn, whose infinite patience in helping me with technical questions I'll never forget.
- Eng. Leif Andersson and Rolf Braun for their help in logistic problems.
- All the PhD's and PhD students at the department: Johan Bengtsson, Anton Cervin, Lena de Maré, Jonas Eborn, Johan Eker, Mattias Grundelius, Magnus Gäfvert, Sven Hedlund, Ari Ingimundarson, Bo Lincoln, Erik Möllerstedt, Rasmus Olsson, Mikael Petersson, Anders Robertsson, Henrik Sandberg, Stefan Solyom, Hubertus Tummescheit, and Stephan Velut who have been so nice and who have made me feel at home from the beginning.
- All my office mates, for being so patient and gentle with me, in particular I want to remember: Michael Grebeck, Kuan Luen Ng, Franck Ruffier and Susana Santos.
- The secretaries: Britt Marie Mårtensson, Eva Schildt and Agneta Tuszynski whose gentleness and helpfulness are unique.
- Last but not least, my Italian "Super Advisor" Prof. De Carli Alessandro, who has given me the possibility to go abroad to develop this work and who permitted to enlarge my knowledge in control science outside the academic Italian world.

To My Family,

who have made all this possible and always supported my decisions. I will never be able to pay back what they did and do for me.
A special thought goes to all people who have helped me in the past when I passed through a dark period of my life.

Alessandro

Contents

1. Introduction

History of Paper

It is well known that the word *paper* comes from papyrus, a writing material which was known in Egypt as long ago as around the year 3500 B.C. Papyrus was made by putting together fibers from the stem of the papyrus plant, and after drying in the sunlight was ready to writing.

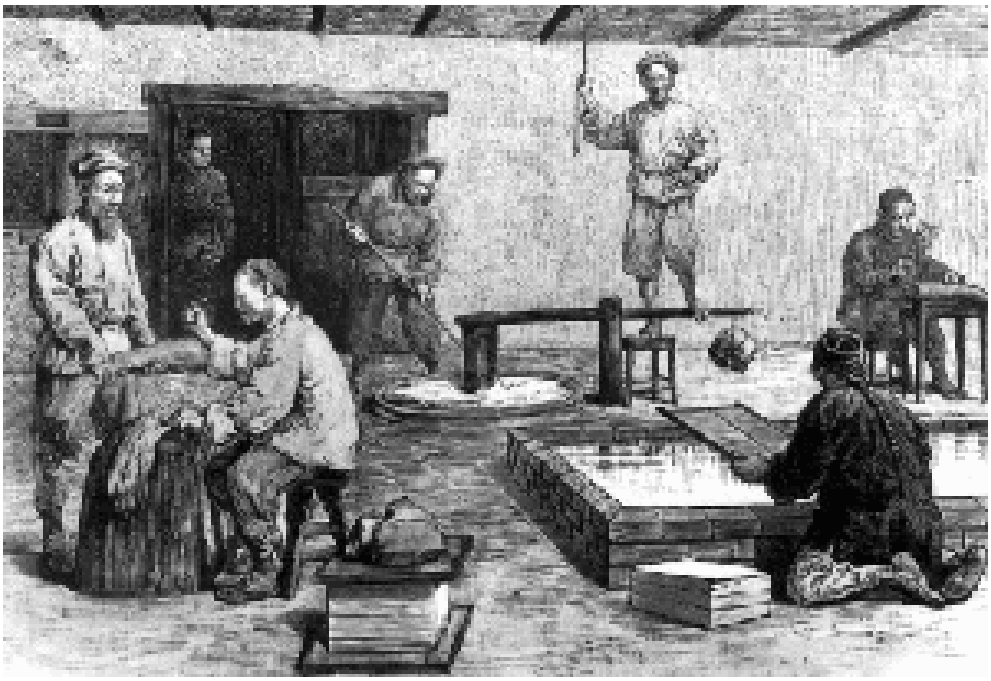

Figure 1.1 Chinese Paper Making 100 A.C. (From [8]).

The first findings of paper were discovered in China and they can be dated around 105 A.C.. In Europe the paper and the papermaking process arrived around the 11th century. It arrived in Spain through the Moors in 1238. Independently of this, paper mills also began to appear in Italy in the middle of the 13th century. In France we find the first mill in 1338, and the Germans followed in 1390. The first plant in Scandinavia was in Linköping in 1523. Production of paper in the beginning was naturally done by hand and was so expensive that it was sold in particular shops called *apothecaries*. Parchment was still largely used, it was above all due to the cheaper way how it was obtained. The invention of the printing press in the 1450 made it cheaper to produce books. The paper gave a better print result than using parchment.

Little by little the paper became the principal and later on the only material used for writing. People understood the potential of this process and new methods of printing were invented.

Around 1600 it was begun to print the first regular newspapers. At the

end of this century the papermaking art was spread all over the Europe, the table 1.1 shows the earliest mill in some European countries.

Year (AD)			
Spain (Xativa)	1150	Holland (Gennep)	1428
France (Herault)	1189	UK (Hertfordshire)	1488
Italy (Fabriano)	1260	Sweden (Motala)	1532
Germany (Nuremberg)	1389	Denmark	1540
Switzerland (Marry)	1400	Russia (Moscow)	1690
Belgium	1407	USA (Germanstown, Pa.)	1690

Table 1.1 Erliest Paper Mill in some European Country.

Due to this development it was necessary to improve the performance of the paper making. It was time to invent the *paper machine*. In 1799 Nicolas-Louis Robert, a Frenchman, patented the first papermaking process. The patent was, however, more an idea than a practical machine.

Consequently it was sold, and arrived in London to the Fourdrinier brothers, who were in the paper business. So, the idea of the paper machine became a real concrete working machine in 1804. In 1820, Thomas Crompton patented a method for drying paper with the help of steam heated cylinders.

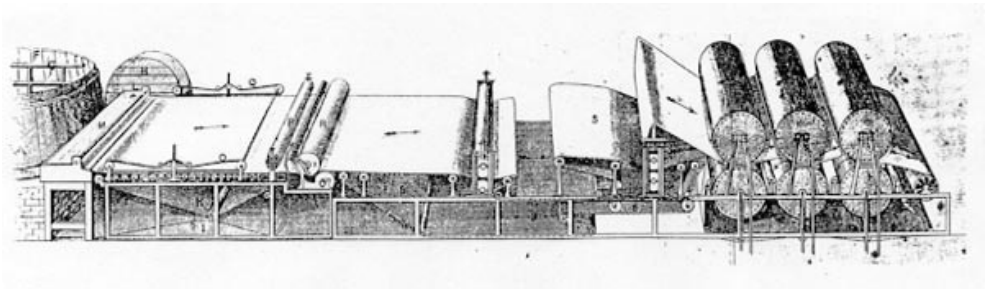

Figure 1.2 Fourdrinier Paper Machine 1850 (From [9]).

The basic principles of todays paper machines had been developed. A rapid technological progress was at hand and a fantastic development began.

2. Pulp and Paper

This chapter is dedicate to the description of the whole papermaking process, in fact it is required and necessary to explain the basics of the papermaking process before to introduce the goal of the present work, further information in [9] and [7].

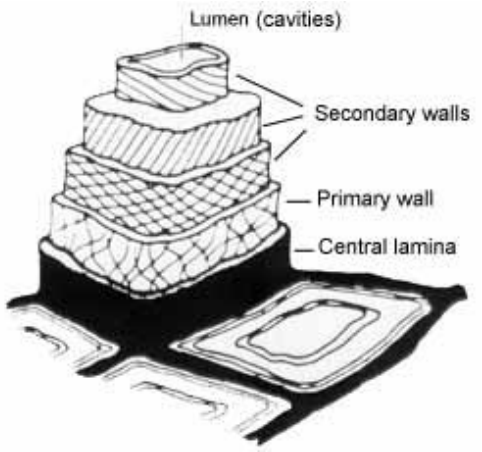

Figure 2.1 Wood Structure (From [9]).

2.1 Wood and Fiber

It is well known that wood is made basically of two components (see Fig (2.1)):

- *Cellulose* is the principal component and its content inside wood is around 45% plus a 30% of hemi-cellulose. The cellulose molecule consists of several chains of interlinked ring-shaped molecules, mostly glucose units. The number can vary from a few hundred to more than 10,000.

- *Lignin* is a resinous material acting as a glue, binding the fibers together, its content in wood is around 5%. The lignin molecule is even more complicated than cellulose molecules, consists of networks of ring-shaped molecules of different kinds.

Other secondary substances, like tree oils, resins and fatty acids contained inside wood, have to be separated from the cellulose to be able to make paper with good characteristics.

A classification of the wood in two basics sets, with respect to structural characteristics, it can be given:

- *Softwood*, in which the fibers have a length of 2–4 mm and the thickness is around 0.03 mm. To this category belong woods coming from coniferous trees.

- *Hardwood*, in which the fibers have a length of 0.5–1.5 mm, maintaining a thickness around 0.03 mm. To this category belong woods coming from deciduous trees.

It is the combination of fibers and lignin that gives the trunk and branches of the tree strength, hardness, and durability.

There are two different basics methods to extract fibers from wood and separate them from lignin and other substances:

- *Chemical dissolution* of the lignin which means that the lignin is dissolved with the aid of chemicals and heat. This process to dissolve lignin using chemicals is also called *chemical pulp*, and the pulp obtained it contains only cellulose.

  This process performed with softwood gives a strong paper, due to the long fibers, and is therefore used when strength is important. On the other hand when the process is performed with hardwood which has short fibers is ideal for making fine paper, and good printing properties are achieved.

- *Mechanical processing* of the wood which means that the fibers are torn away from one another by mechanical actions. This process to divide the fibers each other is also called *mechanical pulp*. The pulp obtained it contains cellulose mixed with other substances, including the lignin.

  This process is performed mostly with spruce and is used principally for newsprint and paperboard.

The propriety and qualities of pulp produced are affected by the kind of pulping process as well as the type of wood used.

Apart from wood, a great deal of other materials like grass, cereals and synthetic materials was tested and sometime employed to product paper, but with poor results. *Waste paper* has long been a major source of fiber raw material for the paper industry and in recent years the use of waste paper has rapidly increased. Despite the use of waste paper is growing related to wood needs and ecological problems. In the other hand it is necessary to note that its use is pretty difficult because of presence of foreign materials.

2.2 Wood Handling

The first part of the papermaking process is almost the same in all plants. The wood arrives in the mill in form of logs or chips. The logs are so passed thorough a machinery which consist in large revolving drums and that take away the bark, the product so obtained is fragmented into chips and it is very important for the next processes to have the same size of the chips.

The chips are then stored in piles and fed from the bottom of the pile through a system of conveyors, which supply the *digester house* in which the pulping process takes place.

2.3 Chemical Pulp

Chemical pulp is produced by cooking wood chips in a pressurized boiler (known as a digester) with a cooking liquor. The chemicals in the liquid liberate the cellulose by dissolving the lignin.

A chemical pulp process consist of two lines:

- *Fiber line*, which take the chips and give, at the end of the process, the pulp which essentially consists of cellulose.
- *Chemical recovery line* takes the used liquor which contains, the cooking chemicals and same dissolved wood substances.

There are several good reasons to take care of the liquor:

- The liquor can be recovered and reused.
- The cooking chemicals, after separation from the liquor, can be reused.
- The dissolved wood substance after separation from the liquor can be used as a fuel.

The main objectives achieved recovering this materials are: money saving and environmental preservation.

The chemical pulp can be produced using two different processes, sulphate process or the sulfite process, they are described below. The properties of chemical pulp can be varied within wide limits depending on the wood raw material and the wanted grade of paper. The typical yield of a chemical pulp is around 45% over wood fed into, the lignin content is around 2% to 5%.

The Sulphate Pulp Process

The name of this process come from the specific chemical used, namely sodium sulphate (Na_2SO_4), which is used to compensate losses of sulphur and sodium in the process. Nowadays these losses are normally low and there is only a little need for sodium sulphate.

The chemicals that participate in the dissolving process are highly alkaline (high pH) and are called *white liquor*. These are sodium hydroxide ($NaOH$) and sodium sulfite (Na_2S). White liquor also contains other chemicals, for example sodium carbonate (Na_2CO_3) and sodium sulphate that, however, do not participate in the lignin dissolving process.

The liquor contained in the pulp after the sulphate cook is called *black liquor* and is washed out of the pulp after the cooking.

The sulphate pulp mills are recognizable because they have a so called “rotten egg” smell. This smell is caused by, among other things, different sulphur-containing gases such as hydrogen sulfite. Nowadays, the gases are collected and burned to minimize the smell problem.

The Sulfite Pulp Process

Almost all the sulfite pulp mills use magnesium (Mg) as the base chemical for the cooking process. Only one Swedish mill uses sodium (Na) and one uses calcium (Ca).

Batch cooking or continuous methods can be employed in the sulfite pulp mills. The machinery used in the digester plants is more or less the

same as in the sulphate pulp mills. In Sweden almost all sulfite processes uses batch cooking and *magnesium bisulfite* ($Mg(HSO_3)_2$) is used as the active chemical. The cooking is done in acid environment due to the bisulfite and the pH is around 4.0. The cooking conditions (temperature, pressure, cooking time) are almost the same as in sulphate pulp cooking.

After the cooking the pulp is washed and screened. The return liquor is evaporated and recovered as much as possible. In sulfite pulp cooking more of the resin remains in the pulp than in sulphate pulp cooking. There are different methods to remove the resin remains.

The precedents are the two methods employed to develop a chemical pulp, but the pulp cooking can be processed in two different basics way as described in following subsection.

Pulp Cooking

The two basics methods are now described:

- *Batch mode*, in which the digester (cooker) is filled up with chips and liquor. The cooking liquor, passing through screens, is carried with a pump to an heat exchanger, here it is heated with steam before fed back to the digester.

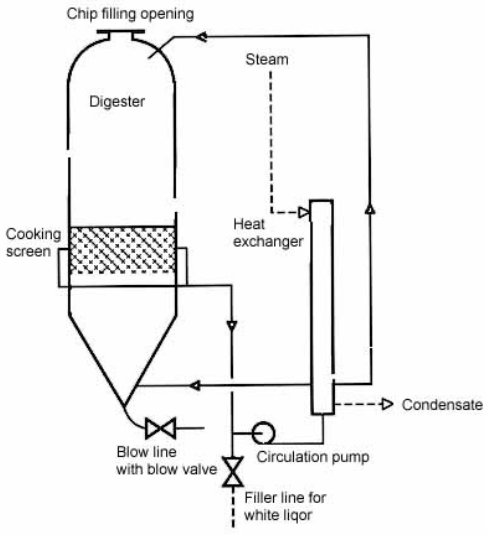

Figure 2.2 Batch Cooking (From [9]).

  The cooking temperature is around 170°C. Time and conditions of cooking depend on the wood row material. At the end of the cooking an outlet valve is opened and the operation is repeated.

  Several digesters are used in the batch cooking process. The number depends on the size of the digesters and the production quantity at the mill.

- *Continuous mode*, in which cooking chips and liquor are fed into the digester in a continuous flow and ready cooked pulp come out continuously from the bottom of the digester.

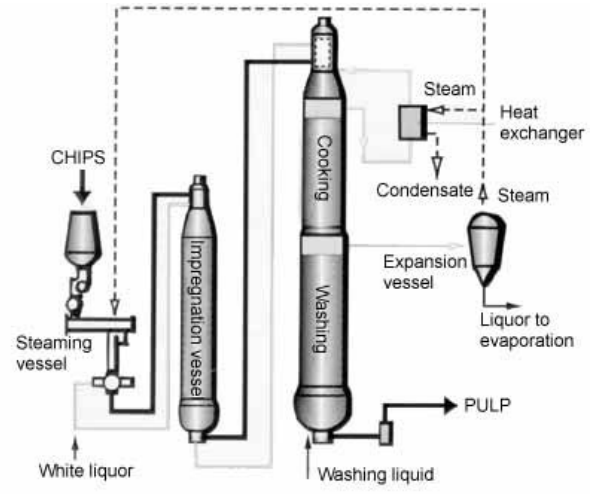

Figure 2.3 Continuous Cooking (From [9]).

The Digester

Basically, the digester is a large vertical pipe in which the chopped wood or pulp moves from top to bottom. The process takes place in several stages:

1. *Presteaming* with steam in a chip hopper.
2. *Steaming* with steam in a steaming vessel.
3. *Soaking* with liquor, which penetrates into the cavities in the chips. This is also helps to remove the air among chips.
4. *Cooking* during which the lignin is dissolved.
5. *Washing* in the lower part of the digester.

The cooking quality is controlled thorough pH, cooking time, and temperature of the cooking bulk so that the proper pulp grade is obtained.

Pulp which has to be bleached should contain as little residual lignin as possible, to keep down the consumption of bleaching chemicals. This is important not only for environmental reasons but also to obtain good production economy.

2.4 Mechanical Pulp

Mechanical pulp is made by fragmenting the wood and releasing the fibers mechanically. The wood pass thorough one or more grinders, in this way the pulp is obtained. To get an homogeneous pulp it is important that the moisture content in the incoming wood is at least 30% and preferably around 47%. Wood of low moisture content is presoaked by immersion or by spraying with water. This process is simpler than the chemical one, mostly because there no chemical recovering.

The wood yield for mechanical pulp is almost 100%, on the the other hand the price to pay for this high rate is that here all the substances remaining in the pulp, such as lignin, resin and some coloring material.

For this kind of pulp there is no use of the chemicals and related chemical problems are avoided, i.e. the smelling trouble.

Softwood is usually used in this process, but some hardwoods can also be used. Large amounts of electrical energy are consumed in the manufacture of mechanical pulp. Mechanical pulp is used to make several different products and the range of applications has increased with the technical developments. However, when strength and purity are the most important properties of the paper chemical pulp is still largely used, moreover in some common use, like newspaper, bleaching can be avoided.

When mechanical pulp is carried out at elevated temperature adding some chemicals it is called *semi-chemical pulp*. In this particular process the wood is presoftened with chemical and thereafter treated as in mechanical pulp. In such kind of process the yield it is around 90%.

Nowaday two different kinds of mechanical pulp are implemented in several different variant:

Groundwood pulp

This pulping process was the first in which wood was used as raw material for papermaking.

Today, groundwood pulp is produced in the following way: barked round wood logs are pressed against a rotating cylinder of sandstone while being sprayed with water to cool and wash the grinding stone. The grains in the surface of the stone penetrate into the surface of the wood which is heated by the friction. This cause bond breaking between fiber and lignin. In this way, little by little the broken fibers became smaller and finally pulp.

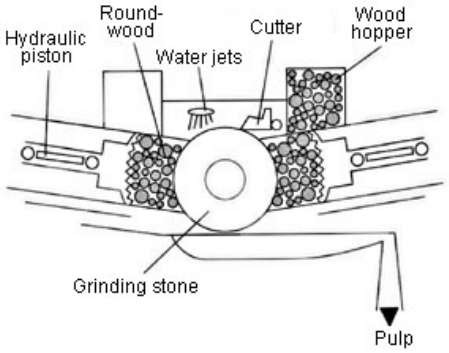

Figure 2.4 Grinder for Groundwood Pulp (From [9]).

The largest grinding stones may have a diameter of 1.8 meters and is powered from a motor of more than 7 MW. The average of production is about 120 tones of pulp per grinder per 24 hours.

A recently developed type of grinder is the pressure grinder in which grinding is processed under positive pressure in a steamed environment. Pressurized groundwood pulp is stronger than normal groundwood pulp.

Refiner pulp

This name is representative for the process, because the pulping is achieved carrying the grinding wood chips against a disc refiner. This has two grinding discs with patterned surfaces and a chip feeding device.

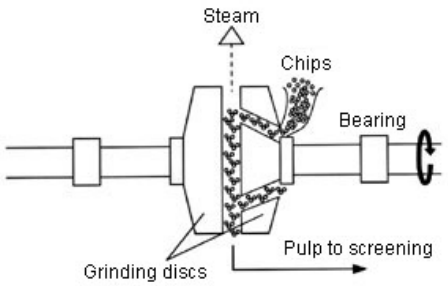

Figure 2.5 Grinder for Refiner Pulp (From [9]).

The commonest refiner pulp are thermo-mechanical pulp, TMP, and chemic-thermo-mechanical pulp, $CTMP$. In both cases, the chips are preheated with steam to about 120°C. In the manufacture of $CTMP$, the chips are also impregnated with sodium sulfite (Na_2SO_3).

2.5 Washing and Bleaching

After cooking the pulp pass thorough the washing process. This is very important in the view that any remaining of liquor leads to an higher consumption of chemicals and this must be avoided, for environmental and economic reasons. The washing consists of an screening action which separate the fiber from other impurities and a filtering action. The screens are made in several different ways and permit to achieve a certain homogeneity.

Bleaching is the next process which the pulp usually is passed thorough. Pulp coming from sulphate pulp is brown colored, pulp coming from sulfite pulp is grey-yellow colored, and pulp coming from mechanical pulp it slightly yellow colored. To obtain good whiteness of the final product the pulp has to be bleached. There are two basic way to bleach pulp:

- *Lignin-removing bleaching* is a continuation of the dissolving lignin cooking process but with softer chemical actions and lower temperature.

- *Lignin-preserving bleaching* uses chemicals which make the lignin lighter in color without removing it from the pulp. This method is mostly used for mechanical pulp in which all the lignin remains.

Brightness for pulp and paper is measured in ISO brightness, which is measured as a percentage of absolute whiteness. Unbleached softwood sulphate pulp has a brightness of about 26% ISO. Sulfite pulp is much brighter, 60–65% ISO. Bleached pulp have brightnesses from 70% to more than 90% ISO. Newsprint has a brightness of 65–70% ISO. Printing and writing papers have brightnesses between 85 and 90% ISO and have to be made from highly bleached pulp.

2.6 Papermaking

The papermaking process is showed in Fig. 2.6 and can be summarized in four stages:

- *stock preparation*,
- *dewatering*,
- *pressing*,
- *drying*.

These are followed by a fifth stage, called *finishing*, which can involve glazing, coating, slitting/winding, sheeting, wrapping/packing, etc.

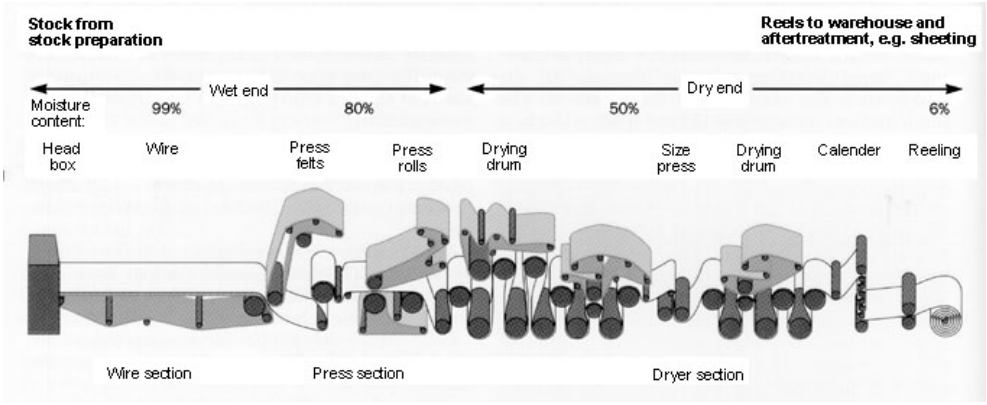

Figure 2.6 Paper Machine (From [9]).

Stock preparation

This process is the first stage of the papermaking, the raw material is the pulp that arrives either from the precedent stage of the same plant or from another paper mill. In the first case the plant process is called *integrated paper production* otherwise *non-integrated mill production*, most Swedish paper mills have their own pulp mills.

Quite often, different types of paper pulp are mixed together either to have the wanted properties or to recover pulp coming from recycled paper. This mixing is usually done in conjunction with stock preparation.

The stock is prepared by beating the incoming pulp. Pulp is put into the beater, and water is added to facilitate the mixing. Mechanical pulp and recycled fiber pulp are not normally beaten as they already have sufficient good properties.

The beater is basically a revolving cylinder and a bed-plate; the fiber are forced to pass thorough them. The revolving and press actions mash and split the fibers, which are forced to absorb water: an homogeneous "slurry" mixing is obtained. In the beater other materials are also added to the slurry to give to the final product the requested property. The most common treatments and additives are:

- **Sizing**, which is a treatment to prevent penetration of moisture and in general water. A typical sizing solution consists of a rosin soap

dispersion mixed with the stock in an amount of 1 to 5 percent of fiber. Since there is no affinity between rosin soap and fiber, it is necessary to use a coupling agent, normally alum ($Al_2(SO_4)_3$).

- **Fillers**, are added to improve surface property such as brightness, opacity, surface smoothness, and ink receptivity. Clay (aluminum silicate), often referred to as kaolin or china clay, is commonly used, but only in a few places in the world (Cornwall, in England, and Georgia, in the United States) are the deposits readily accessible and sufficiently pure to be used for pigment. Another pigment is titanium dioxide (TiO_2), which is the most expensive of the common pigments and is often used together with others.

  Calcium carbonate ($CaCO_3$) is used mainly to improve brightness, opacity, and ink receptivity to printing and magazine stocks. Special uses include the filling of cigarette paper, which give good burning properties.

  To improve surface strength the most commonly used materials are starch, polyacrylamide resins, and natural gums such as locust bean gum and guar gum. The most common type of starch currently used is the modified type known as cationic starch. When dispersed in water, this starch assumes a positive surface charge. Because fiber normally assumes a negative surface charge, there is an affinity between the cationic starch and the fiber.

  Other fillers are zinc oxide, zinc sulfide, hydrated silica, calcium sulfate, hydrated alumina, talc and barium sulfate.

  Since most of these fillers have no affinity for fibers, it is necessary to add an agent such as alum to help hold the filler in the formed sheet. The amount of filler used may vary from 1 to 10% of the fibers, nevertheless many types of fine paper may contain up to 30% of fillers.

- **Dyes**, which is usually added to impart pigment and color to the paper stock. Many so called *direct dyes* with a natural affinity for cellulose fiber are highly absorbed, even from dilute water solution. The so called *basic dyes* have a high affinity for groundwood and unbleached pulp.

Chemicals are also added to the stock to influence its pH. Fillers such as kaolin and chalk are added to enhance the brightness and opacity of the paper.

The stock is cleaned on screens and revolving cleaners to remove sand, metal particles, etc. It is now ready in the *header*, before to be sprayed out onto the wire of the paper machine the stock is dilute with white water to a very low concentration, usually the water mass content is 99%.

Dewatering

This is in practice the first stage of the pulp drying, and it is named *wire section* or *wet section*.

In the wire section the stock is dewatered on one or more wires. The highly diluted (0.2–1% kg of solids/kg of pulp) stock flows onto the wire from a head box and is dewatered. The stock leaves the wire as a web of paper with a dry solids content of about 20%. Depending on the design of the wet section, the paper machines are called:

- *fourdrinier* machines, where dewatering and sheet formation take place on an endless wire felt which is stretched between a number of rolls,
- *twin wire* machines, where the sheet is formed between two wires and dewatering takes place in both directions,
- *hybrid* machines featuring both a fourdrinier and a twin wire.

Pressing

Dewatering of the paper web continues in the press section, where the web is pressed between press rolls. Usually the press section consists of three or four such press roll stages, in which the dry solids content of the paper increases step by step. After the final press roll nips the dry solids content is about 30–50%. The presses are usually fitted with press felts which distribute the pressing pressure on the paper web and suck and remove water. There are three basic different press types:

- *suction press*, consists of a perforated suction roll revolving against a solid roll with a granite or rubber shell: the paper web is constrained to pass through them. The suction roll is provided with a vacuum pump, that provide the suction action.
- *vented nip press* consists of a vented (grooved) solid roll and a counter roll with a smooth surface. In the nip, water is pressed out of the felt down into the grooves and is thrown out by centrifugal force,
- *felt wire press* is a plastic wire passing between the felt and the lower roll. The plastic wire facilitates removal of the expelled water from the press roll nip.

Drying

In the dryer section the paper web is dried to a dry solids content up to 95%. This is accomplished by dryer wires which press the paper web against steam-heated drying cylinders.

The dryer section is encased in a dryer hood. It is vital for good ventilation to be maintained in the dryer hood so that moist air can be removed. The moist air is heat-exchanged and its heat content utilized. This heat can be used, for example, to make hot water or to heat the premises.

Paper machines are often classified on the basis of how the dryer section is designed:

- Multicylinder machines have a large number of steam-heated drying cylinders with diameters of 1.5–1.8 meters (newsprint, fine paper, kraft paper, and other grades).
- Yankee machines have a large drying cylinder (Yankee cylinder) with a diameter of 4–6 meters (one-side glazed paper and other grades).
- Combination machines with one Yankee cylinder and a number of ordinary drying cylinders.

The Yankee cylinder not only dries the paper web but also gives a smooth surface to the paper. This is accomplished by pressing the paper web against the brightly polished Yankee cylinder. The web sticks to the cylinder until it has become dry enough to be released.

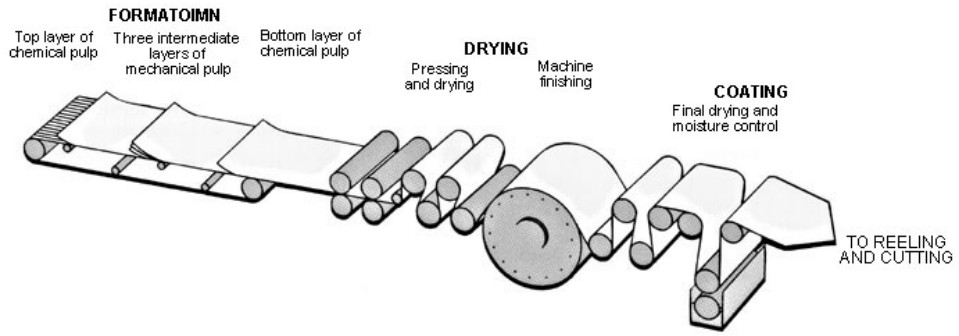

Figure 2.7 Paper Machine showing processes (From [9])

Paper Finishing

After leaving the dryer section the paper is often passed through a finishing machine which consists of a stack of steel rolls. As the paper passes through the nips between the rolls its surface is smoothened. The paper can be surface-treated in different ways. To reduce the amount of dust coming from the surface of the paper in the printing presses the paper is surface sized. This is done with starch in sizing presses which are located in the dryer section or in a separate station. When enhanced brightness and printing property are required the paper is covered with a coating of kaolin, chalk or titanium dioxide. The coat can be applied, as can surface sizing, either in the paper machine or in a separate stage.

When a reel of paper is ready the paper web is broken and led over to a new roll. This is called a reeling drum. The reel is lifted away for further treatment.

Before the paper is delivered the reel on the reeling drum is divided into smaller reels and wrapped. Some paper is delivered in the form of sheets of different formats after sheet cutting in a sheet cutter.

3. Background

Nowaday the papermaking process is one of the most important sectors of Sweden economy and represents a substantial part of GNP, it involves a large amount of energy and natural resources. In this view it is necessary and required a very efficient use of raw materials, besides considering the large amount of energy employed, small improvements in production or in energy utilization, or use of natural resources can represent significant energy savings and profits.

The largest amount of energy is used in the drying operations. From experimental measurements the amount of energy employed in the drying is estimated to be about 2/3 of the total. This, coupled with the fact that Sweden is one of the major paper producer, makes it necessary to consider more deeply the problem of drying.

3.1 Previous Work

A large amount of work is available on drying. Extensive research describing the paper web is available, but little can be found concerning the thermohydraulic part.

Previous work at the Department of Chemical Engineering I at Lund University concerning paper drying includes mainly mathematical modeling of the multi-cylinder dryers and internal transport phenomena within the paper structure (see [10] and [4]).

Other references concern the suction siphon inside the steam heated cylinders, which is placed to remove the amount of water forming by condensation inside the steam heated cylinder. Also work can be found concerning the hood ventilation systems. This study started with an investigation of the references for the drying process. Two references have mainly been taken in consideration. The focus of respective works is briefly described.

In [10] a mathematical model is developed for the drying of paper in a multicylinder drying section. The model is based on the unsteady state, one-dimensional heat conduction equation, which is applied to both the cylinder shell and the paper web. No internal mass transfer phenomena are modeled explicitly. A lumped parameter, the effective thermal conductivity of the paper, implicitly takes, at least partially into account, these phenomena. Great emphasis is placed on finding the proper heat transfer coefficients that are a part of the boundary conditions of the unsteady state heat conduction equations.

Three of this boundary conditions are investigate in detail:

- contact heat transfer at the cylinder-paper interface;
- mass transfer at the paper-fabric-air interfaces;
- laminar condensate flow inside cylinders.

In [4], almost the same mathematical model is developed for the drying of paper in a multicylinder drying section. The model is based on the unsteady state heat conductivity equation, the unsteady state mass balance

for water in the paper web and the unsteady state mass balance for dry solids in the paper web. The model only includes the cylinder shell and the paper web. The dynamic effects in the steam and condensate system and in the drying hood are neglected. These two assumptions are validated with experimental measurements. No internal mass transfer phenomena inside the paper web are explicitly modeled. The moisture content is modeled as a linear function of stock flow. Particular attention is taken in the grade change simulations.

3.2 Goals of the Present Work

Considering the paper models discussed above it can be found out that they use a static equation for the steam pressure inside the steam heated cylinder to get the right corresponding temperature (see equation (3.42) in [4]). In the present work we attempt to develop a model for the thermohydraulic missed part.

The first target is principally to develop a suitable model for the steam heated cylinder, part of a group in a drying section of the paper machine. The drying section can consist of several steam heated cylinders, their number can easily reach 100 units.

The whole model is implemented in the Modelica™ environment, an object oriented language, described in chapter 4.

Brief descriptions of the basics component of the thermohydraulic Submodels are given. The whole thermohydraulic model is mainly composed of two basics components:

- *Flow model*, which contains the momentum and mass flow equations.
- *Control Volume model*, which contains mass and energy balance equations, when a suitable *medium* model is added.

This concept will be clear in chapter 5, where they will be described in detail.

Starting with the cylinder, the whole model is extended by adding pipes, valves, etc.. Using the Control Volume and Flow models almost all of the following models are built up:

- **Cylinder**.
- **Valve**, static or dynamic.
- **Pipe**.
- **Pipe 3 Port**.
- **Tank**.
- **Sink**.
- Cylinder **Shell**.
- Simple **Paper Web**.

Using the previous components as “bricks” the thermohydraulic model is built up. The model is then “tuned” choosing suitable parameter values and checking the corresponding variable values, so trying to obtain a good behavior of the whole system. *Good behavior* means that it is possible to reproduce, disregarding some unavoidable modeling errors, the behavior of a real process.

The second goal is to choose a suitable controller for the cylinder pressure. To achieve this goal two items could be taken in account: cost and benefit. Now let considering the system, it is immediately clear that several controllers are required for the whole plant so each controller could be of very low price. In the other hand good performance are required.

In this view, considering the rate cost/benefit only very simple, effective and efficient controllers can be used. It is desired to improve the pressure stability to improve the stability of the steam temperature, the heat flow, and finally to reach an improvement in drying performance.

4. Modeling Software

4.1 Introduction and State of Art

Nowaday several modeling and simulation environments are available. There are systems that are complex and heterogeneous, involving several different scientific fields: thermodynamics, chemistry, power electronics, mechanics, robotics, and so on.

Despite, the state-of-art of modeling languages many often do not adequately support the structuring of large, complex models, and the process of model evolution in general. This support is usually provided by sophisticated graphical user interfaces, but at the price of specialization to a certain modeling formalism or application domain, or even uniquely to a specific software package. Therefore, they would be useless with regard to the interoperability and heterogeneous problems.

Among the recent research results in modeling and simulation, two concepts have strong relevance to this problem:

- *Object-oriented modeling languages* have already demonstrated how object oriented concepts can be successfully employed to support hierarchical structuring, reuse, and evolution of large and complex models independent from the application domain and specialized graphical formalisms.
- *Non-causal modeling* demonstrate that the traditional simulation abstraction – the input/output block – can be generalized by relaxing the causality constraints, i.e., by not constraining ports to an 'input' or 'output' role early, and that this generalization enables both simpler models and more efficient simulation, while retaining the capability to include submodels with fixed input/output roles.

The combined power of these concepts together with proven technology from existing modeling languages justify a new attempt to introduce interoperability of modeling tools and simulation systems.

4.2 Why Modelica™ Software?

To answer this question[1], the object-oriented, non-causal modeling methodology and the corresponding standard model representation, Modelica™, should be compared with at least four alternatives:

1. Established commercial *general purpose simulation tools*, such as ACSL, EASY5, SIMULINK, System Build and others, are continually developed and Modelica will have to offer significant practical advantages with respect to these,

[1]Was the first thing I wondered when my supervisor told me to use this language !

2. *Special purpose simulation programs* for electronics (Spice, Saber, etc), multibody systems (ADAMS, DADS, SIMPACK, etc), chemical processes (ASPEN Plus, SpeedUp, etc) have specialized GUI and strong model libraries. However, they lack the multi-domain capabilities.

3. Many industrial simulation studies are still done without the use of any general purpose simulation tool, but rather relying on *numerical subroutine libraries and traditional programming languages*. Based on experience with such tools, many users in this category frequently doubt that any general purpose method is capable of offering sufficient efficiency and robustness for their application.

4. An IEEE supported alternative *language standardization effort* is underway: VHDL-AMS.

4.3 Modelica Simulators

In this section just a glance is given of the languages which have been used as starting point for the Modelica design, i.e., Modelica builds upon the experience gained with these languages.

Since the definition of CSSL (Continuous System Simulation Language) in 1967, most modeling languages are essentially block oriented with inputs and outputs and the mathematical models are defined as assignment statements for auxiliary variables and derivatives. Physical equations thus need to be transformed to in a suitable form for calculation. The only aid in transforming the equations to an algorithm for calculating derivatives is automatic sorting of the equations.

The languages that form the base of Modelica, all have general equations, i.e. $expression = expression$, as the basic element. Hierarchical decomposition and reuse are typically supported by some kind of model class. Typically, the languages have provisions to describe physical connection mechanisms, i.e. to associate a set of variables with some kind of port. Such ports can be used at higher hierarchical levels when connecting submodels without having to deal with individual variables.

Examples of object–oriented and/or non-causal modeling languages include: ASCEND, Dymola, gPROMS, MOSES, NMF, ObjectMath, Omola, SIDOPS+, Smile, U.L.M., ALLAN, and VHDL-AMS.

In particular, further information are given about Dymola: the specific language here used to perform the simulations in the present work.

Dymola

Dymola (Dynamic Modeling Language) (http://www.dynasim.se/), was introduced already in 1978 [1] and is based on equations for non-causal modeling, model types for reuse, and submodel invocation for hierarchical modeling.

The Dymola translator utilizes graph theoretical methods for causality assignment, for sorting, and for finding minimal systems of simultaneous equations. Computer algebra is used to solve for the unknowns and to make simplifications of the equations. Constructs for hybrid modeling, including instantaneous equations, was introduced. Crossing functions for

efficient handling of state events are automatically generated. A graphical editor is used to build icons and to make model compositions. Major application areas include multi-body systems, drive-trains, power electronics, and thermal systems.

5. ThermoFlow: a Modelica™ Library

This chapter is essentially dedicated to a brief description of *ThermoFlow*: a thermo-hydraulic library in Modelica™ language[1] further information can be found in [6].

5.1 Topics of the Library

The general goal is to provide Modelica™ with a tool capable to treat problems regarding thermo-hydraulic systems.

Since is not possible to cover the large variety of real thermo-hydraulic systems, the library is built up putting more emphasis and attention to basic building blocks that in complete one. This idea rise since Modelica™ is an objected oriented language and one of its main topics is extensibility. This makes it possible to combine several different basic models to obtain the desired complete one. For this reason a great deal of effort is put to develop basic *Flow Model* (FM) and *Control Volume* (CV) (see Fig. 5.1) . In the CV models is then added a suitable model for the desired *Medium Model* (MM).

Since the medium flow direction is a priori unknown and even more it may change during the simulation, the possibility of bidirectional flow is provided. So, any model is built up with accessing point or *connectors*, which are referred to as *junctions* and neither input nor output.

The library is designed and built up standing the following guidelines:

- One unified library both for lumped and distributed models.
- Separation of the medium submodels, which can be selected through class parameters.
- Both bi- and unidirectional flows are supported.
- Assumptions can be selected through class parameter.

Taking in mind this specifications the main target of the library is to enable the user to quickly assemble complex models playing with the simple basics one.

5.2 Basic Ideas

The basic ideas are to build two basic "*simple*" model and build up the whole thermo-hydraulic library mainly using only these as a base for any other model with some exceptions (e.g. internal conduction). They are then always alternatively interconnected to be able to build up distributed models or to compose more complicate ones, (e.g. pipes). it is now time to consider the two basic blocks:

[1] Still in development in this department.

- ***Control Volume***, which contains, in a mathematical form, the following ideas:
  - **Energy Balance**
  - **Mass Balance**

  A suitable ***Medium Model*** is then added in each CV.
- ***Flow Model***, which contains in a mathematical form, the following ideas:
  - **Momentum Balance**, which can be implemented in several different way depending on the specific application, representative are:
    - *Dynamic Momentum Balance*
    - *Stationary Pressure Drop*

Summarizing, it can be said that energy and mass balances are matters of Control Volume model, and momentum balance are matters of Flow model. The general scheme in Fig. 5.1 show that there is always an alternation of Control Volume and Flow models.

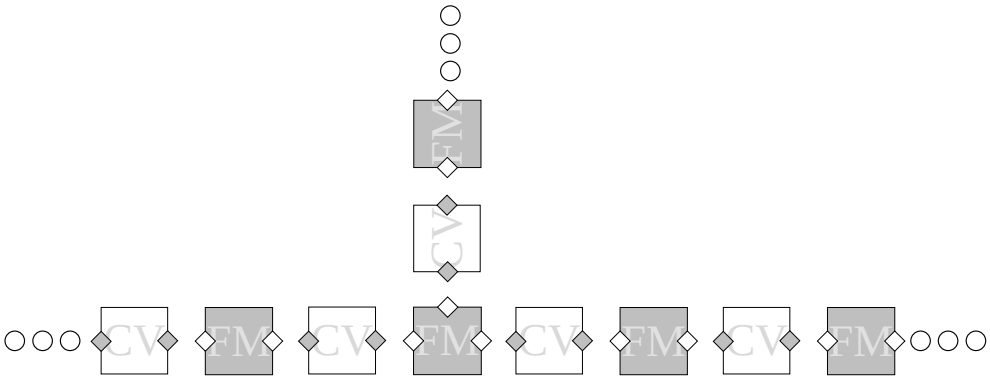

Figure 5.1 Flow Grid

After that all basic ideas are given a simple example is given. Consider a thermohydraulic model with one source and one sink and several different intermediate thermohydraulic models (e.g. pipes, valves, etc.). Then it is sufficient to assign two independent state variables, (e.g. pressure and enthalpy) in the source and just one variable in the sink to be able to get the behavior of the whole system, whenever a medium model (e.g. water, CO_2, etc.) is assigned. These concepts will be clear in the next section.

5.3 Basic Equations

The basic ideas in the previous section are now transformed into mathematical equations[2].

[2]Computers do not understand ideas, yet!

Control Volume – Basic Equations

The Control Volume model is, together with the FM, the most important basic building block of the ThermoFlow library, it consist usually of two connectors (at least one), energy and mass balance equations and an added medium. The "*key*" of the whole library is in the previous sentence, in fact it can be found that any thermohydraulic model contain at least one CV.

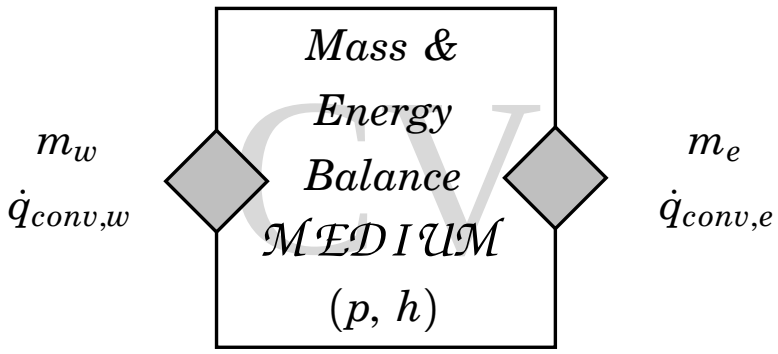

Figure 5.2 Control Volume Model

Consider the *mass* and *energy* balances, which introduce the following equations:

$$M = \rho V \tag{5.1}$$
$$U = uM \tag{5.2}$$

where M is the total mass, U is the total inner energy, V is the total volume, ρ is the density and u is the specific inner energy. All the precedent variables are referred to a general fluid. A differentiation for a *constant volume* V yields:

$$\frac{dM}{dt} = V\frac{d\rho}{dt} = \sum_i^n \dot{m}_i \tag{5.3}$$
$$\frac{dU}{dt} = M\frac{du}{dt} + u\frac{dM}{dt} = \sum_i^n \dot{q}_{conv,i} + \sum_j^l \dot{q}_{tran,j} \tag{5.4}$$

The right hand side of equations (5.3) and (5.4) are given for a Control Volume with n heat flows ($\dot{q}_{conv,i}$) in n corresponding connections to other Control Volumes and l heat transfer ($\dot{q}_{tran,j}$) areas to solid bodies. Flows are taken positive into the Control Volume.

Flow Model – Basic Equations

The Flow Model is, together with the CV, the most important basic building block of the ThermoFlow library, it usually consist of two connectors and a momentum flow equation. By definition the *momentum* equation is:

$$I = \int_V \rho\, w\, dV = \int_{\Delta z}\int_A \rho\, w\, dA\, dz = \dot{m}\,\Delta z \tag{5.5}$$

Figure 5.3 Flow Model

where w is the velocity in z direction, A is the normal flow area and $\dot{m}$ is the mass flow. From the equation 5.5 and for a dynamic FM the *dynamic momentum balance* may be expressed as:

$$\Delta z \frac{d\dot{m}}{dt} = \frac{dI}{dt} = \dot{I}_1 - \dot{I}_2 + (p_1 - p_2)\, A - F_{wall} \tag{5.6}$$

In the same way from the equation (5.5) and for a static FM the *stationary pressure drop* can be expressed as:

$$0 \;=\; (p_1 - p_2)\, A - F_{wall} \tag{5.7}$$

F_{wall} depends on the "flow characteristic", a common assumption is turbulent friction loss:

$$F_{wall} \;=\; \xi \frac{\Delta z}{2D} \dot{m}\, |w| \tag{5.8}$$

Each generic convective heat flow ($\dot{q}_{conv,i}$) in the right hand side of equation (5.4) is calculated in the previous[3] FM which the CV is connected; it may be expressed as:

$$\dot{q}_{conv,i} \;=\; \dot{m}_i\, h_i \tag{5.9}$$

where $\dot{m}_i$ is the incoming mass flow and h_i is the corresponding enthalpy.

Medium Model

It is known that the medium models implementation is a problem, as soon as more elaborate model than of an ideal gas is required. The problem rises since the trade off between fast and accurate implementation, in fact the behavior of a medium is highly complicate, unless it is an ideal gas. In particular it is desirable to avoid iterative calculations. At the moment are implemented the following medium models:

- *Pure Ideal Gas*.
- *Mixture of Ideal Gases*.
- CO_2
- *Water*

[3]Previous respect to the flow direction

Some more models are going to be implemented. The models are all very accurate and they are taken either from some recommended formulations or standards like IAPWS/IF97 for the water. The medium model is then added to the CV. In this way utilizing the mass and energy balance equations, it is possible to get all the state variables and proprieties of the medium.

5.4 Pressure and Enthalpy as States

A very brief explanation is provided to understand the need to use pressure and enthalpy as state instead of internal energy and density (or mass). Whenever there are flows with heat transfer the balance equations (5.3) and (5.4) are not suitable to be implemented, because they may yield a stiff system. Instead, two other independent state variable must be chosen. The principal advantages are:

- To avoid stiffness or at least reduce it.
- Faster computations for the medium proprieties.

If pressure p and enthalpy h are chosen as states the balance equations may be rewritten as:

$$\frac{d\rho}{dt} = \left.\frac{\partial \rho}{\partial p}\right|_h \frac{dp}{dt} + \left.\frac{\partial \rho}{\partial h}\right|_p \frac{dh}{dt} \tag{5.10}$$

$$\frac{du}{dt} = \left.\frac{\partial u}{\partial p}\right|_h \frac{dp}{dt} + \left.\frac{\partial u}{\partial h}\right|_p \frac{dh}{dt} \tag{5.11}$$

To obtain differential equations for pressure and enthalpy the linear system composed of equations 5.10 and 5.11 must be solved.

For certain applications, and when the stiffness it is not a problem temperature and density may be chosen as states. In other cases even pressure and temperature may be chosen as states. They are not independent inside the two phase region In that is why another state variable (e.g. the vapor mass fraction) must be chosen. Temperature and pressure are often chosen because the easy way in which they can measured.

6. ThermoHydraulic Model

This chapter is essentially dedicated to the description of the thermohydraulic model, developed to reproduce the behavior of one steam heated cylinder as part of a drying section of the paper machine. A model has been developed that reproduces morphology and topology of a general plant, taking particular attention to dimensions and sensor placing. Even more, it has be tried to model all dynamics to be able to reproduce the "real" behavior of the system. Particular care is taken to model each single component.

6.1 Model: Overview

Above all, it is necessary to remained the reader that Modelica™ is an object oriented language, one characteristic, among others, allows the user to "reuse" the same model, with the possibility to assign different parameters; further informations may be found in chapter 4 and in [6].

In this section all the basic models , used to build up the whole thermohydraulic model, will be presented and briefly described.

Almost all the basic models grouped in the **PaperDry** library are derived from *ThermoFlow*, a Modelica™ library already described in chapter 5.

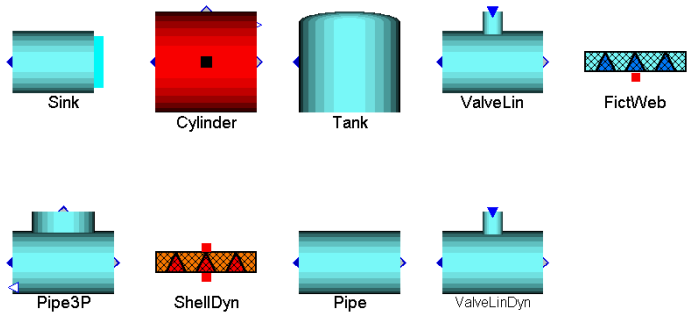

Figure 6.1 Basic Models

To reproduce the thermohydraulic behavior of the steam heated cylinder the following basic models are developed:

- **Pipe**, which is essentially built with the Control Volumes and the Flow Model alternatively connected. It has obviously two accessing points.
- **Pipe 3 Port**, which is built with a central Control Volume and either one or two Flow Models. It has three accessing points.
- **Valve**, which is basically a Flow Model. It is implemented in two different versions:
  - *Static Valve*, which contains just a linear relation from the input signal to the flow area.
  - *Dynamic Valve*, which is a more complicated model than the static one, dynamics is now added.

- **Cylinder**, which is essentially the Control Volume model with an assigned Medium model: water in this case.
- **Tank**, which is one connector Control Volume model with two assigned state variables: pressure and enthalpy.
- **Sink**, which is one connector Control Volume model with an assigned pressure.
- **Cylinder Shell**, which essentially contains discretized diffusion equations.
- **Simple Paper Web**, which is a very simple static model containing just a boundary condition equation.

The models, which are shown in Fig. (6.1), will be treated in more detail in the next sections.

6.2 Some Definitions

To avoid confusion in neophyte readers[1] and before to continuing with the description of the models it is necessary to give some basic definitions:

- *Fluid*: an aggregate of matter in which the molecules are able to *flow* past each other without limit and without fracture planes forming.
- *Liquid*: a state of matter intermediate between that of crystalline substances and gases in which a substance has the capacity to flow under extremely small shear stresses and conforms to the shape of a confining vessel, but is relatively incompressible, lacks the capacity to expand without limit, and can posses a free surface.
- *Medium*: that entity in which objects exist an phenomena take place; examples are free space and various fluids and solids.
- *Steam*: water vapor, or water in its gaseous state; the most widely used working fluid in external combustion engine cycles.
- *Vapor*: a gas at a temperature below the critical temperature, so that it can be liquefied by compression, without lowering the temperature.

All definitions are from [11].

6.3 Pipe Model

The pipe is built up with a series of Flow Model and Control Volume and to be able to properly model the dynamic behavior of the fluid the pipe is made *distributed*. The basic idea to make it is shown in Fig. (6.2), where FM and CV are alternatively interconnected.

A detailed description of the used equation inside CVs and FMs in out of the scope of this work, even if some general guidelines will be provided. Each CV is provided with a Medium Model which consists of two input

[1]This section may be avoided to the expert reader.

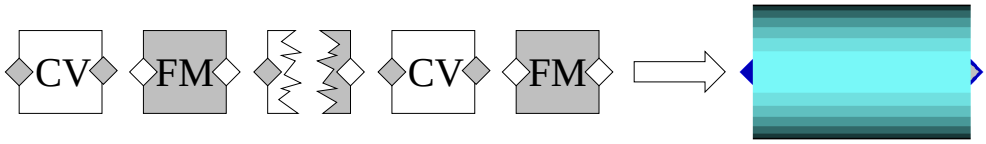

Figure 6.2 Pipe Model

independent state variables. In particular water is chosen as MM and enthalpy h and pressure p as input state variables. Each FM contained in the discretized pipe has to be provided with a suitable equation relating pressure drop and mass flow. In particular a *dynamic* equation is added relating momentum flow, mass flow, and pressure drop it has the same structure as the equation (5.6), but with a slightly different equation for F_{wall}. The following expression is given in each FM for F_{wall}:

$$F_{wall,i} = A \frac{\dot{m}_i^2}{\dot{m}_{0,i}^2} \Delta p_0 \qquad i = 1, \ldots, n \tag{6.1}$$

where n is the number of segments, $F_{wall,i}$ is the friction force between the extremities of each segment, A_i is the flow area, $\dot{m}_i$ is the mass flow, $\dot{m}_{0,i}$ and Δp_0 are respectively the measured pressure drop and mass flow in each FM of the pipe.

The expression for F_{wall} here used is very simple and differs from the equations presented in chapter 5, but it behaves fairly good when Δp and $\dot{m}_0$ are the measured values of pressure drop and mass flow in a pipe in steady state conditions.

6.4 Pipe Three Port Model

The Three Port Pipe is the natural and necessary extension of a pipe. It is natural to extend the pipe by induction and it is necessary to be able to build up complex models. In fact with this component and only with this it is possible to build models containing more than one input–one output. The necessity to have this component raise since real models usually have one generator supplying more than one pipe, e.g. in this model the tank provides steam for several cylinders.

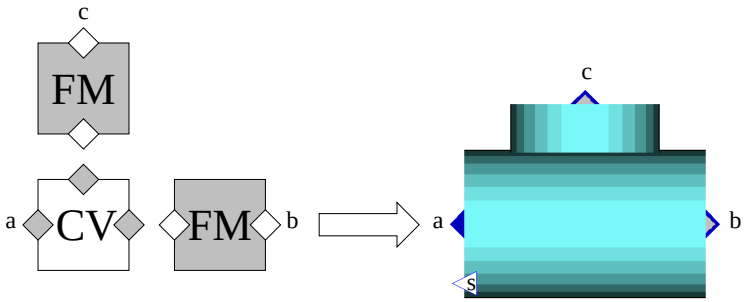

Figure 6.3 Pipe 3 Port Model

The Three Port Pipe is a *lumped* model, it consists of a three port Control Volume and either one or two Flow Models, it depends if the model is used respectively as either two input–one output or one input-two output. Only the one input–two output is here considered and utilized, it is shown in Fig (6.3). As in the Pipe model the Control Volume model is provided with a Medium Model and pressure and enthalpy are used as input state variables. The FMs here used are once more *dynamics* as in the pipe model, thus the same equation (5.6) relating momentum flow, mass flow, and pressure drop is used. For F_{wall} the following expression is given in each FM:

$$F_{wall,i} = A_i \frac{\dot{m}_i^2}{\dot{m}_{0,i}^2} \Delta p_0 \qquad i = 1,2 \tag{6.2}$$

where n is now the number of FMs, $F_{wall,i}$ is the friction force between the extremities of each FM, A_i is the flow area, $\dot{m}_i$ is the mass flow, $\dot{m}_{0,i}$ and Δp_0 are respectively the measured pressure drop and mass flow in each FM of the pipe.

6.5 Valve Model

The valve model is essentially derived from a Flow Model, in particular from a *static* FM. Since a static FM is used, the momentum flow is not used in the equation relating pressure drop and mass flow. In this case the equation (5.7) is utilized. Thus, both dynamic and static valves share the following equation for the F_{wall}:

$$F_{wall} = \frac{S_l}{\rho A_v^2} \dot{m}^2 \tag{6.3}$$

where F_{wall} is the friction force, S_l is the lateral surface, A_v is the flow area, and $\dot{m}$ is the mass flow.

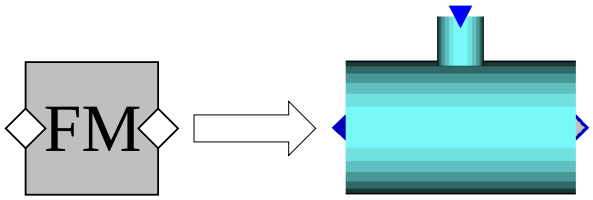

Figure 6.4 Valve Model

The derivation of the valve model is made via *extension* from the basic FM provided from the library, in this way the valve model *inherits* all the desired properties with the possibility to add your own either static or dynamic equations.

Static Valve

The static valve is the simplest possible valve model, just a linear relation, input signal output flow area, is provided to the static FM. The following equation is added:

$$A_v = A\, u_v \qquad u_v \in [0,1] \tag{6.4}$$

where u_v and A_v are respectively the input signal and the output variable flow area.

Dynamic Valve

The dynamic valve here modeled is one step more complicate than the static one, in fact it is modeled with a first order dynamic system. The following equations represent the added dynamic:

$$\dot{x}_v = -\frac{x_v}{\tau} + \frac{u_v}{\tau} \qquad u_v \in [0,1] \tag{6.5}$$

$$A_v = A\,x_v \tag{6.6}$$

where x is the state variable, τ is the time constant of the valve and A, u_v, and A_v are defined as in the static valve.

It has to be noticed that all the areas before mentioned are obviously taken orthogonal to the flow direction and the momentum flow is calculated with the variable area A_v (see the code in Appendix).

6.6 Cylinder Model

This is the principal object of the thermohydraulic model, it is basically a three port Control Volume. The basic CV provided from the ThermoFlow library has been modified to be able to have the correct behavior of the model.

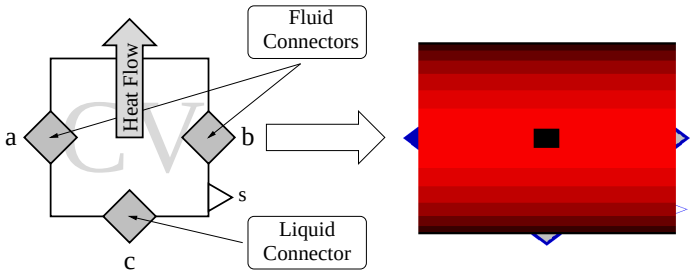

Figure 6.5 Cylinder Model

It is shown in Fig. (6.5) that the connectors **a**, **b** and **c** take the fluid with different properties. A detailed description follows:

- The connectors **a** and **b** take either inlet or outlet fluid with steam properties, i.e. with the medium global properties calculated in the Medium Model added to the CV.
- The connector **c** takes inlet/outlet fluid with liquid propriety, i.e. with the medium liquid propriety calculated in the Medium Model added to the CV.

The cylinder model seems to be fairly different from the reality, in fact in real plants the cylinder is just one input one output device, instead in the model there are three connectors. This can be explained considering the following compound of models:

- *Cylinder*

- *Water Level Controller*
- *Water Level Valve*

And comparing this with the following compound of real devices:

- *Cylinder*, which is a big two connectors rotating pipe and steam is continuously blown into it. The heat flow coming from the blow fluid is utilized to dry the paper running on its external surface shell.
- *Steam–Water Separator*, which is a three connector device, here take place the liquid–vapor separation. Even more the level of the liquid is kept almost constant by a valve.

To get the similarity between the model and the corresponding section of the plant (see Fig. (6.6)) they have to be observed as black boxes. With this position it is possible to consider only the behavior between input **a** and outputs **b** and **d**. In fact both in the real plant and in the model, the

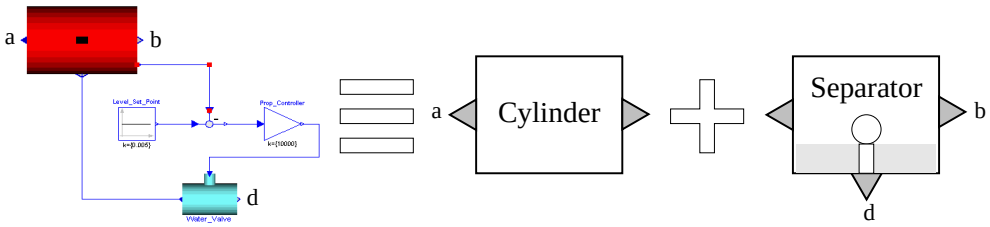

Figure 6.6 Comparison

connector **a** is supplied with overheated steam while the connector **b** blows out wet steam and the connector **d** blows out water. In the model there is the following situation:

- Incoming steam is blown at the left hand side connector of the cylinder.
- Wet steam is blown away at the right hand side connector of the cylinder.
- Water is blown away at the right hand side connector of the valve, when the water level in the cylinder is higher than set point.

In working plants there is the following situation:

- Incoming steam is blown at the left hand side connector of the cylinder.
- Wet steam and water are blown away at the right hand side connector of the cylinder and together are blown into the separator.
- In the separator, water and steam are separated. A special valve, holding a constant pressure, removes the water at the down side connector, while the steam is blown away at the right hand side connector.

Comparing the modeled compound and the real one, disregarding some obvious unavoidable modeling errors, it is easy to understand that they have the same behavior from input **a** to output **b** and from input **a** and output **d**.

Few words will be spent on the connector **s** that appears in the left hand side of the cylinder model in fig. (6.5). In fact it is placed just to be able to measure the water level in the cylinder. It is used as feedback signal in the water-level controller.

6.7 Tank Model

The Tank model shown in Fig. (6.7) is a simple one–connector Control Volume in which two state variable are assigned. A Medium Model is added to the CV and pressure p and enthalpy h are chosen as independent state variable.

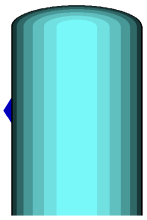

Figure 6.7 Tank Model

This Tank model is an infinity either liquid water or steam source, in which all the state variables in the medium are kept constant.

6.8 Sink Model

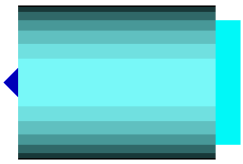

Figure 6.8 Sink Model

The Sink model showed in Fig. (6.8) is a simple one–connector Control Volume in which one state variable is assigned. A Medium Model is added to the CV and pressure p is chosen as fixed variable.

6.9 Cylinder Shell Model

This model, despite all others presented above, is not derived from any other model already implemented in the ThermoFlow library, in fact this

kind of basic models are not implemented, yet. The Cylinder Shell model is a solid body, thus the heat transfer phenomena is reduced just to a simple diffusion phenomena.

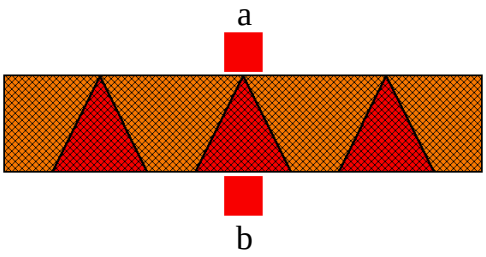

Figure 6.9 Cylinder Shell Model

It is well known that this kind of problems can be treated using the following Fourier's postulate and Fourier's equation (see [2]):

$$\frac{\partial T}{\partial t} = \phi + \frac{\lambda}{\rho c_p} \frac{\partial^2 T}{\partial x^2} \tag{6.7}$$

$$q = -\lambda \frac{\partial T}{\partial x} \tag{6.8}$$

where T is the temperature, t is the time variable, ϕ is the generated heat flux, λ is the diffusion constant, ρ is the density, c_p is the specific heat capacity, q is the heat flux, and x is the space variable.

To be able to apply the precedent equations a space discretization along the flow direction is required and the physical dimensions have to be considered, since the equation are all normalized over an unitary area.

When the equation (6.8) is integrated, the shell is considered as a flat surface and border phenomena are neglected. The precedent approximations are made possible because the wide cylinder dimensions. The boundary condition in the connector **a**, i.e. the equation regarding the heat transfer between the steam in the cylinder and the shell, is inserted in this model, despite the boundary condition in the connector **b** is matter of the model Paper Web model.

6.10 Simple Paper Web Model

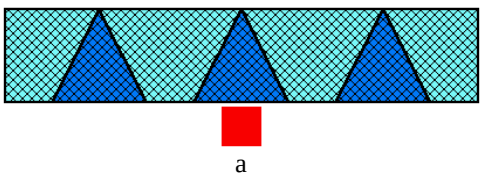

Figure 6.10 Paper Web Model

This Web model is provided just with a boundary condition equation. This model is simple since it is not interesting in the present work to analyze phenomena inside the web. In the other hand it is possible to replace the

web model with another more complicated whenever that phenomena have to be considered. The following boundary equation is given, when an unitary flow area is considered:

$$q = k\,(T_a - T_{ext}) \tag{6.9}$$

where q is the heat flux, k is the thermal conductivity, T_a is the shell contact surface temperature, and T_{ext} is the external surface temperature of the Web.

6.11 Complete Model

This section is dedicated to the description of the complete model shown in Fig. (6.11), disregarding for the moment the pressure control section, which will be described in chapter 7. The model try to reproduce one steam heated cylinder taken as part of a group of twenty cylinders, which are all supplied from a single steam source. One of the problems that has to be solved is how to get the same behavior as twenty cylinder even if only one is modeled. The answer was found by modeling the *Sink_Z* in a special

	Dimensions		
Models	*Diameter*	*Length*	*Width*
Header	0.25	25	–
Pipe	0.15	25	–
Pipe_Z	0.67	25	–
T_Pipe	0.15	1	0.67[2]
Valve_In	0.25	0.3	–
Valve_Out_S	0.027	0.2	–
Valve_Out_W	0.04	0.2	–
Cylinder	1.8	6	–
Shell	0.027[3]	6	1.8π
Web	–	6	1.8π
Tank_S	∞ Dimensions		
Sink_S	∞ Dimensions		
Sink_W	∞ Dimensions		
Sink_Z	∞ Dimensions		

Table 6.1 Model Dimensions

way. This is not a sink as described in section 6.8, but is a model that takes the same pressure, enthalpy and density as the cylinder, even if only

[3]Diameter connector **c**.
[3]Thickness

the pressure is really needed for calculations. In this way the full system behaves as if twenty cylinders are modeled.

All the model have physical dimensions except Tank and Sinks, which have infinity components. All the dimensions are shown in the table 6.1.

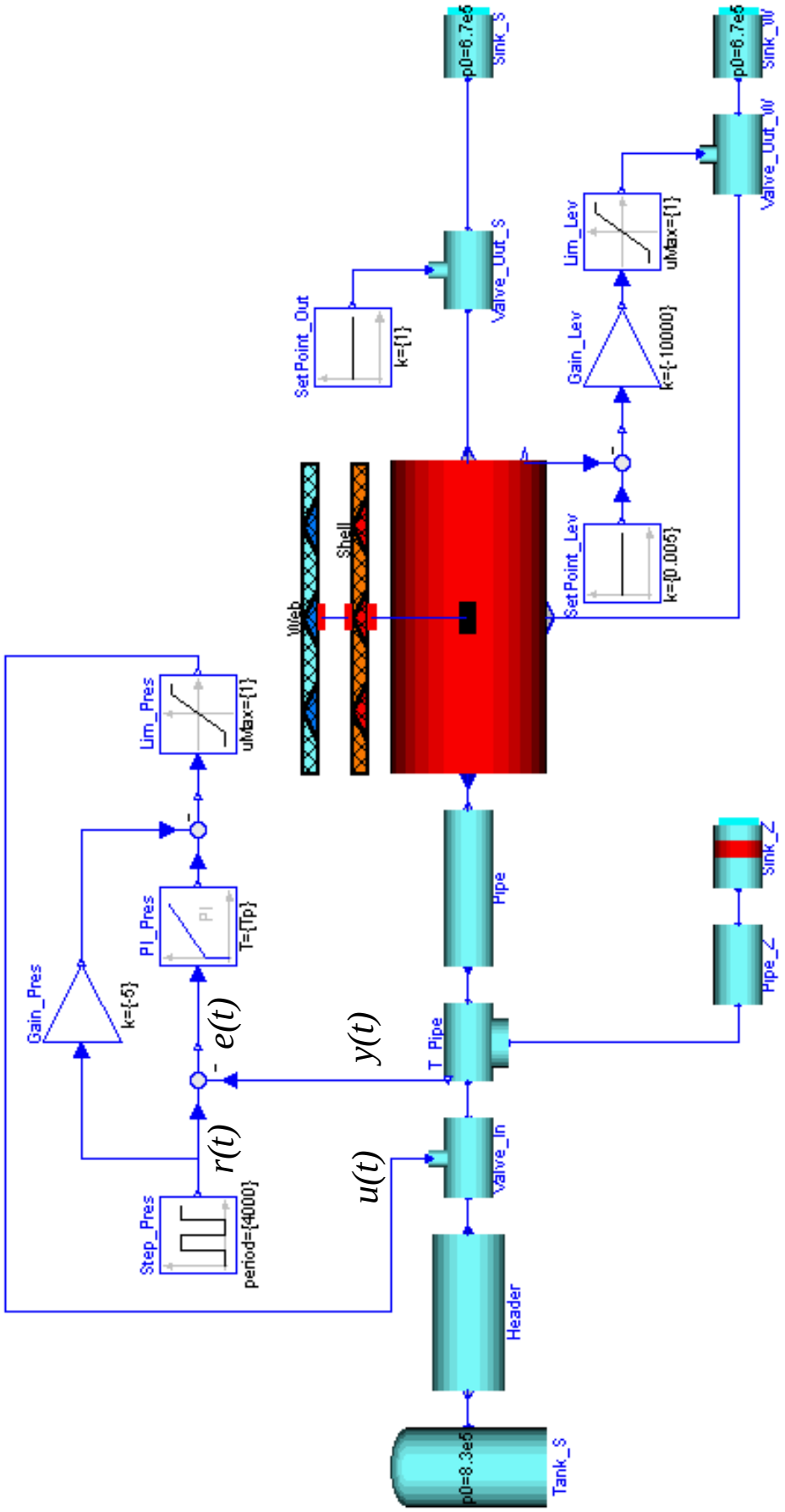

Figure 6.11 Complete Model

6.12 Model Tuning

This section is dedicated to the description of the model tuning, which is done taking data from a real working plant. It has to be said that the available data are fairly confused and incomplete, in fact they consist in few information got in informal way from an engineer of ABB company. Anyway, the data are referred to a particular working plant, whose structure is quite general.

After the complete model had built up, the main problem was to tune it. This operation was fairly difficult mainly for two reasons:

- Few and incomplete data avalaible from real working plants.
- Initial value fitting problems and consequent difficulties to start simulations in Dymola environment.

A list of the few informations and data available follows:

- *Step Response Shape*. This was the first piece of information provided. The Step Response of the system, measured in the pressure sensor, which is positioned in the Three Port model, is showed in Fig. (6.12). From the figure it can be understood that the "real"system is well approximated with a two time constant system. The segment lengths **OA** and **AB** were as well provided, which are respectively around 10 seconds and 150 seconds.
- *Incoming steam pressure*. In the model this is simply the pressure inside the Tank_S. The data values provided for the pressure are between 8 and 10 bars.
- *Incoming Steam Temperature*. In the model this is simply the temperature inside the Tank_S. The data value provided for the temperature is around 460 K.
- *Outcoming liquid water flow*. In the model this is the outlet flow in the connector **c** of the cylinder. The data value provided for this flow is around 0.2 kg/s
- *Outcoming steam flow*. In the model this is the outlet flow in the connector **b**. The data value provided for the outcoming steam is around 15% in mass of the inlet steam flow (connector **a** in the cylinder).

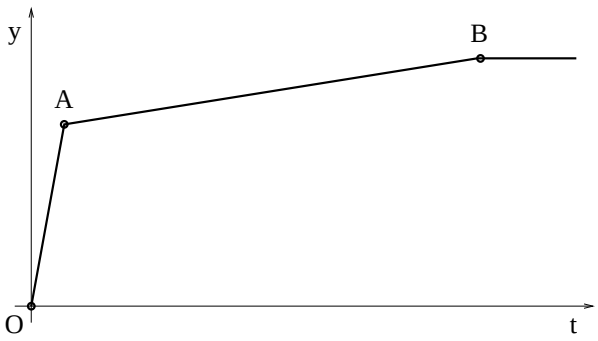

Figure 6.12 Step Response Shape. OA length is around 10 seconds and AB length is around 150 seconds.

7. Pressure Control

This chapter is dedicated mainly to the description of the adopted control pressure strategies. First of all is convenient to introduce the open loop response which make it possible to verify the similarity between the model behavior and the corresponding real system. After this controller and relative tuning methods will be introduced.

7.1 Open Loop Step Response

This section is dedicated to the description of the open loop step response, the sensor signal $y(t)$ (see Fig. (6.11)) is taken as output. The input $u(t)$ is the valve control signal. In Fig. (7.1) is showed the step response for two different set points, moreover are shown the steam temperature inside the cylinder and the cylinder shell external temperature. From the figure it can be seen the delay between the two previous mentioned temperatures, which can be roughly estimated between 5 and 10 seconds.

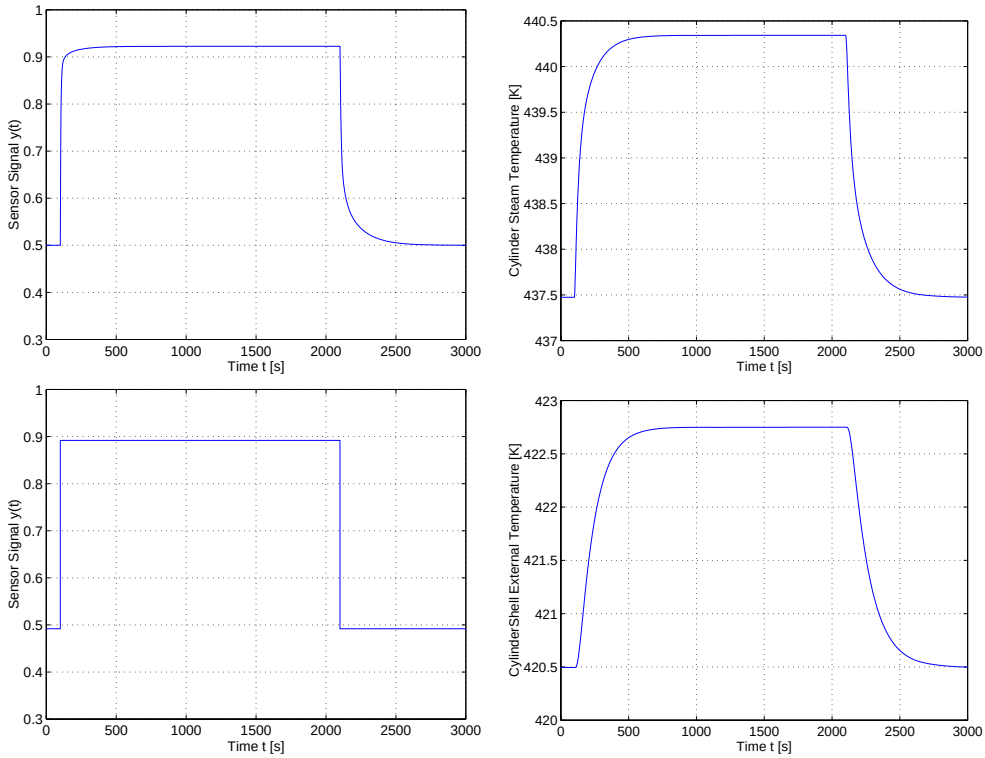

Figure 7.1 Open Loop Step Response.

In Fig. (7.2) the step response is analyzed more in detail and two different the set points are considered. Moreover both step responses are fitted in order to show the two time constant behavior of the system. The step responses are fitted with the simple fitting Matlab command `nlinfit`, which

is provided with the following two time constant fitting function model:

$$y(t) = k_1 + k_2 - k_1\, e^{-\frac{t}{\tau_1}} - k_2\, e^{-\frac{t}{\tau_2}} \tag{7.1}$$

where k_1, k_2, τ_1 and τ_2 are all constants. It is well known that the structure of the function (7.1) is typical of a two time constants linear system. In this way it should be possible to approximate the whole process at each set point with a simple linear two time constant system. In Tab. (7.1) all values for the constants in eq. (7.1) are given. Thus, as shown the Fig.

Constant	*Set Point* = 0.4919	*Set Point* = 0.8919
k_1	0.3845	0.2856
k_2	0.0377	0.1363
τ_1	3.7813	9.2702
τ_2	74.5019	113.1037

Table 7.1 Fitted parameters for the function 7.1 in two different set points.

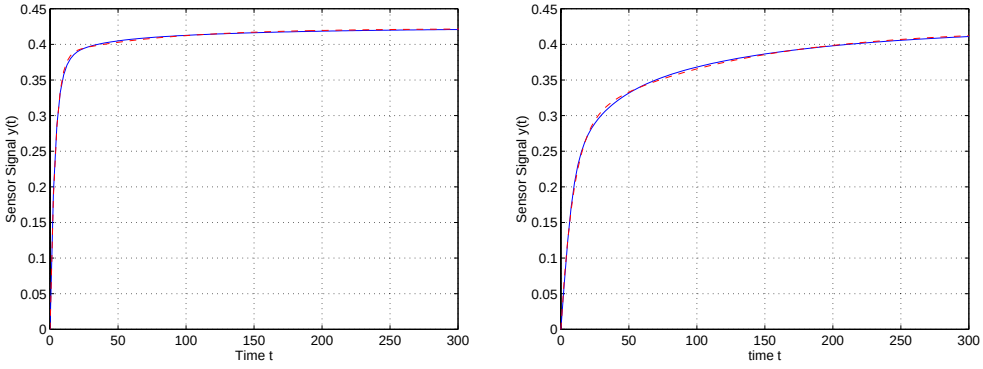

Figure 7.2 Open loop step responses and fitting curves (dashed). Set point = 0.4919 (left). Set point = 0.8919 (right)

(7.2) the fitted curves (dashed lines) agree very well with the respective open loop responses. The behavior of the system can be split in:

- *Fast dynamic* (τ_1), which mainly depends on:
  - valve dynamic, whose time constant is 5 seconds.
  - mass flow in the header, which is discretized to be able to reproduce the flow delay.
- *Slow dynamic* (τ_2), which mainly depends on the pressure wave propagation in the pipe after the valve.

Some more will be spent to describe the system, in fact it seems to be strange that it is possible to approximate the system only with a second order system, as above explained. The explanation can be found observing the sensor signal $y(t)$ and input signal $u(t)$ (see Fig. (6.11)) placement. In fact, they are very close each other respect to the whole system, thus their dynamics are also strongly related. All the states and their placement in the system are showed in Tab. (7.2). The number of states in the discretized models as pipes depends on the discretization number.

Models	*States*	*Models*	*States*
Header	9	Shell	5
Pipe	9	Web	0
Pipe_Z	9	Tank_S	0
T_Pipe	4	Sink_S	0
Valve_In	1	Sink_W	0
Valve_Out_S	0	Sink_Z	0
Valve_Out_W	0	PI controller	1
Cylinder	2	**Total**	40

Table 7.2 Models and states.

7.2 Controller Tuning

The used controller and its tuning will be briefly described. The principal thoughts when an engineer chooses a controller have to be: low costs and high performances. Naturally this it is utopia, a tradeoff is always needed. In this view one of the most simple and effective device is a PI controller, which also well fit in the cost/benefit tradeoff. The general form of a PI controller can be represented by the transfer function:

$$C(s) = k_c \frac{1 + sT_c}{sT_c} \tag{7.2}$$

where k_c is the gain and T_c is the reset time constant.

Sometimes the general form showed in Fig. (7.3) it is not enough and some modifications are needed as in the case to improve the performances of the controlled process. One simple modification is called *set point weighting* (see [5]), which consist to add a feedforward gain to the referement. This modification allows to have overshoot in the step response, which sometimes is useful in order to have quicker set point changes. The scheme of the modified PI controller is shown in Fig. (7.4).

After the controller was chosen, the trouble was to choose a proper method to tune it. The suggestion came from the industry, where the tuning is done using a particular method, which require to fit only one parameter.

Lambda Tuning Method for PI Controllers

This method is very attractive from an industrial point of view, in fact only *one parameter* has to be fitted. In general this method is utilized every time a process may be approximated with a one–time–constant linear system plus a delay. Sometimes, as in the case it is used for low order systems either with or without delay. For further information see [5] and [3].

This method is particularly effective whenever the process is stable and can be approximated by a low order linear system. In fact it is based on pole placement and in particular for stable process as the case on pole cancellation.

It is now time to introduce the PI controller structure specialized for

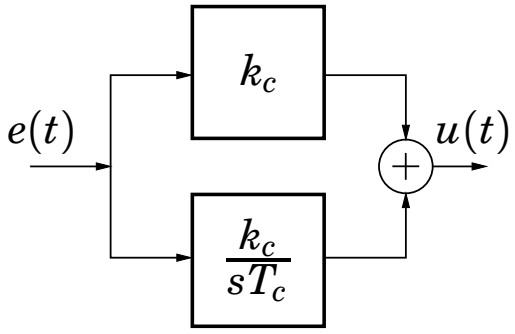

Figure 7.3 General PI controller.

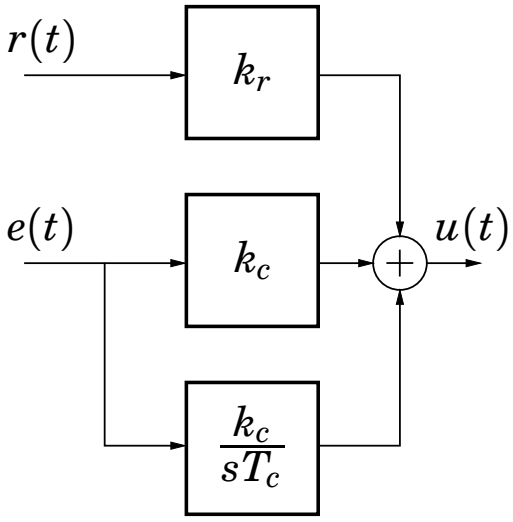

Figure 7.4 Modified PI controller.

the λ tuning method applied to processes without delay:

$$C(s) = \frac{1 + sT_p}{\lambda k_p \, sT_p} \tag{7.3}$$

where k_p is the process gain, T_p is one of the process time constants and λ is the tuning parameter. Since the process approximatively behaves as a two time constants linear system, which of them is it the best choice? In the next section, with the help of simulations, the answer will be (hopefully) found.

7.3 Closed Loop Step Response

In this section will be presented all the simulations and output signals. All the simulation results will be presented using both the PI structure and the λ method. After λ is fitted k_r is changed from 0 to a suitable value.

Before to go thorough the simulations it is good to know how in the industry the tuning is done. The PI controller is before provided with the same gain as the process and after it is provided with one of the two time constants of the system. Which criterium is used, to choose one of them, is not standard.

Now, the results will be presented using all the fitted time constants values (τ_1 and τ_2) given in the table 7.1, moreover because the different behavior of the system in different set point, two mean values for each time constant are also calculated and used to perform a simulation.

Before to plot the results it is necessary to say it is impossible to show all the plots for each configuration of the PI controller with the λ method, thus a choice has to be taken. The system it is not linear, thus the fitting curve and his time constants change for each different set point, see Fig. (7.2).

For this reason approximate mean values respectively for the first and the second time constant are calculated, thus considering λ as parameter,

there will be only two configurations of the controller, whenever the set point weight k_r is neglected.

In this view only few variables will be plotted for the fitted constants τ_1 and τ_2 and more extensive results will be presented for the two mean values, which are defined as:

- τ_m is an approximate mean value of the time constants τ_1.
- τ_M is an approximate mean value of the time constants τ_2.

All the closed loop responses presented are referred to the set point change shown in Fig.(7.1). The controller configurations which use the time constants τ_m and τ_M will be considered more deeply and the following variables will be shown:

- Error signal $e(t)$
- Control Signal $u(t)$
- Sensor signal $y(t)$
- Cylinder temperature
- Shell external temperature

where the $e(t)$, $u(t)$ and $y(t)$ are the signals shown in Fig. (6.11).

Some interesting variables of the system are shown from Fig. (7.5) to Fig. (7.8), while the fitted time constant are used to design the controller. Moreover it is easy to observe the following facts:

- The control signal saturate very easily when τ_1 is used to design the controller.
- The behavior of the system is very similar when different fitted values of the same time constant (e.g. τ_1) are used to design the controller. Compare Fig. (7.5) with Fig. (7.6) and Fig. (7.7) with Fig. (7.8).
- There are no sensible improvement when τ_1 is used changing λ, see Figs. (7.7) and (7.8).

From the previous observations the choice to calculate the time constant mean values seems to be reasonable, some results are going to be presented using the previous mentioned mean values, the results are showed from Fig. (7.9) to (7.10).

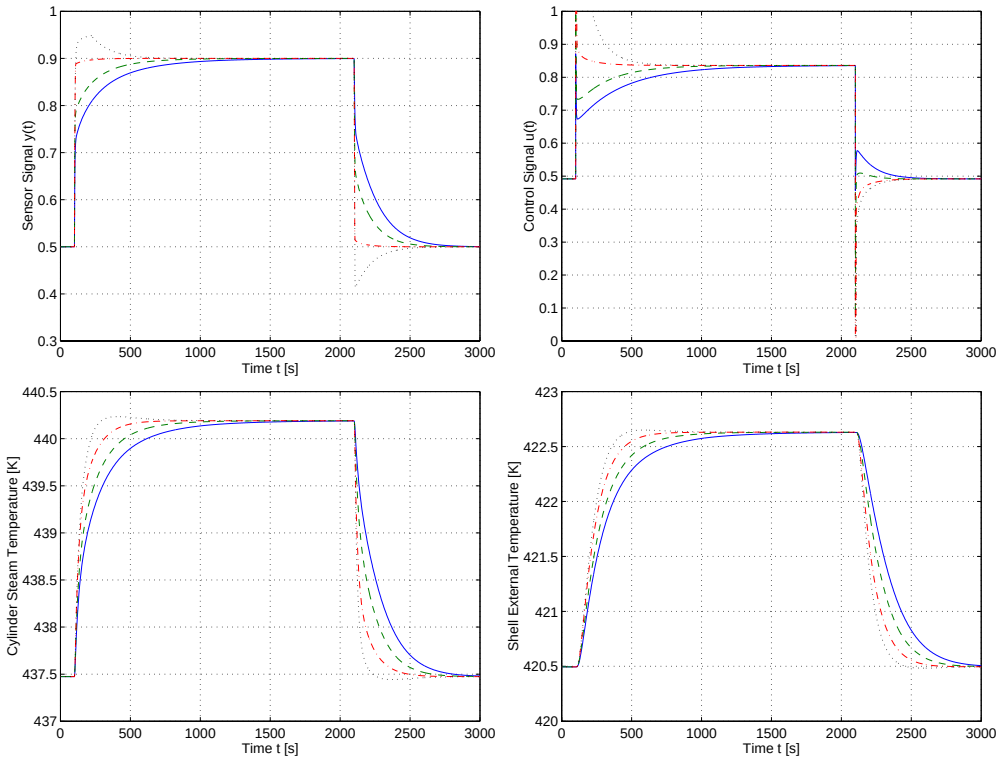

Figure 7.5 $T_p = 113.10$. $k_r = 0$, $\lambda = 1$ (—). $k_r = 0$, $\lambda = 0.5$ (– –). $k_r = 0$, $\lambda = 0.05$ (– · –). $k_r = 5$, $\lambda = 0.05$ (· · ·). The delay between the steam temperature and the external shell temperature can be estimated around 7 seconds.

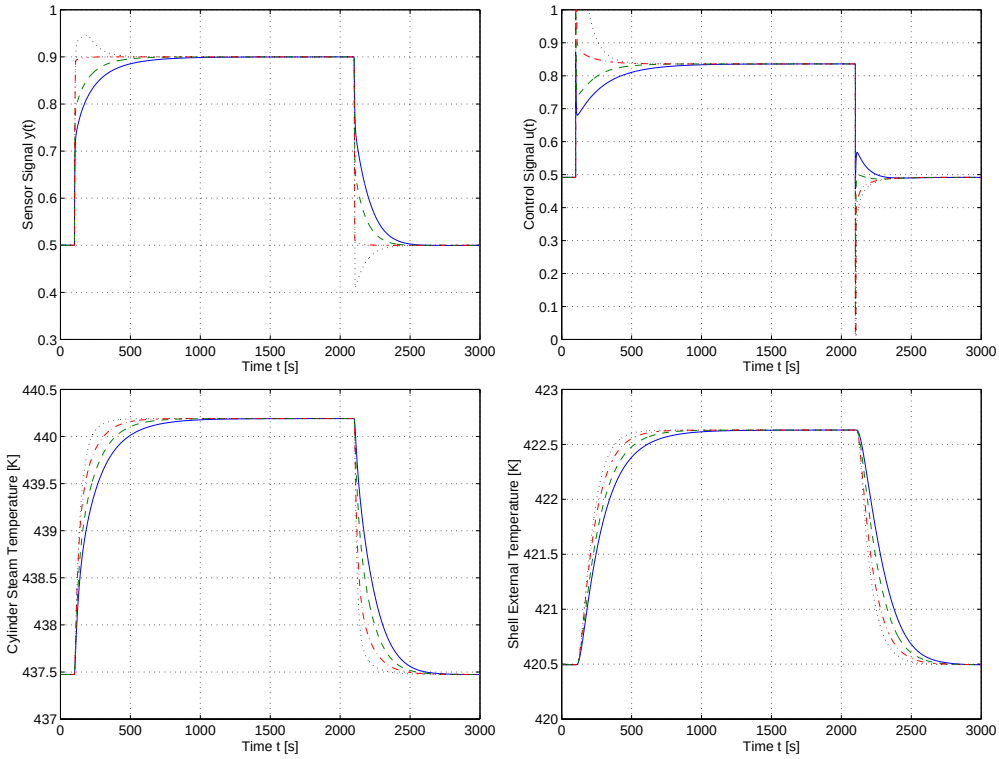

Figure 7.6 $T_p = 74.50$. $k_r = 0$, $\lambda = 1$ (—). $k_r = 0$, $\lambda = 0.5$ (– –). $k_r = 0$, $\lambda = 0.05$ (– · –). $k_r = 5$, $\lambda = 0.05$ (· · ·). The delay between the steam temperature and the external shell temperature can be estimated around 7 seconds.

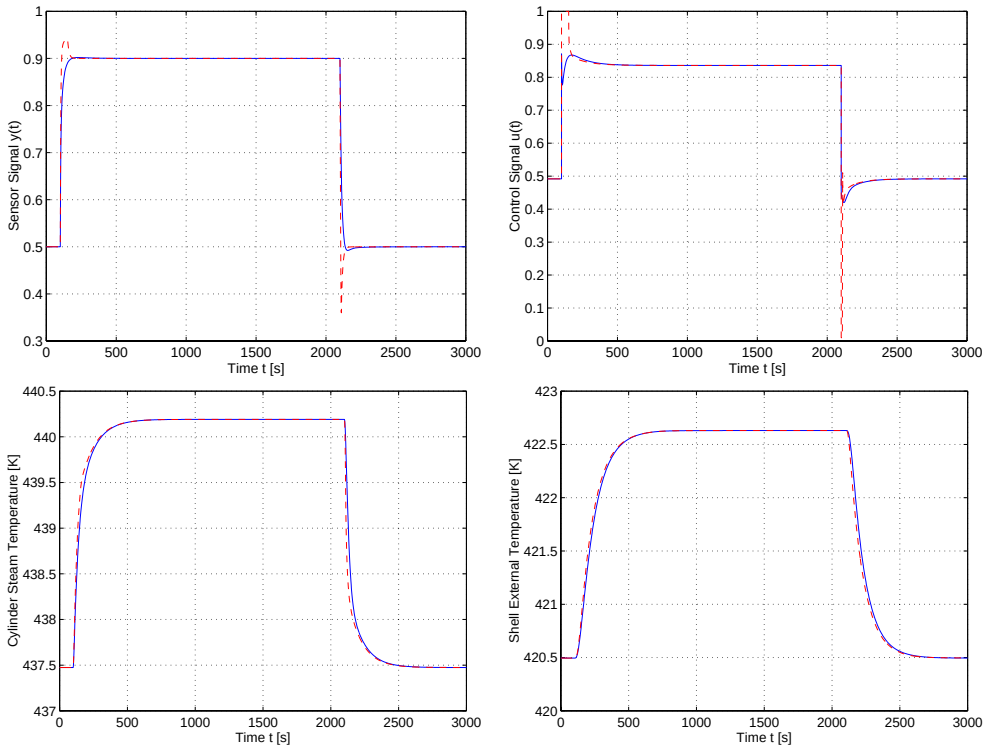

Figure 7.7 $T_p = 9.27$. $k_r = 0$, $\lambda = 1$ (—). $k_r = 5$, $\lambda = 0.05$ ($\cdots$). The delay between the steam temperature and the external shell temperature can be estimated around 7 seconds.

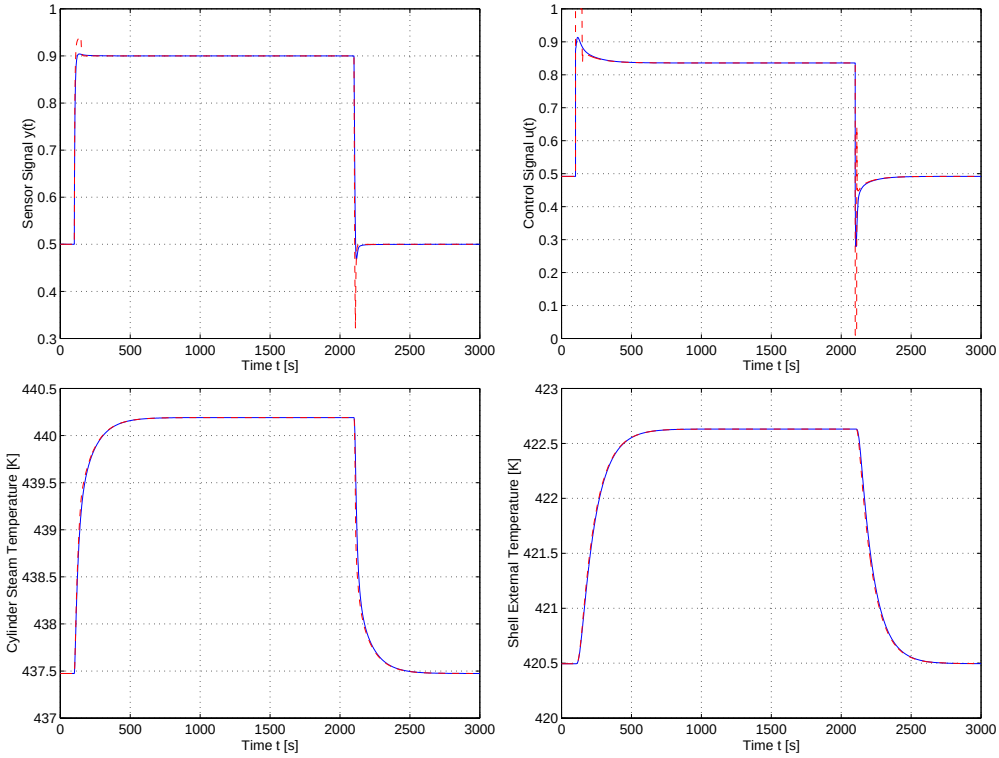

Figure 7.8 $T_p = 3.78$. $k_r = 0$, $\lambda = 1$ (—). $k_r = 5$, $\lambda = 0.05$ ($\cdots$). The delay between the steam temperature and the external shell temperature can be estimated around 7 seconds.

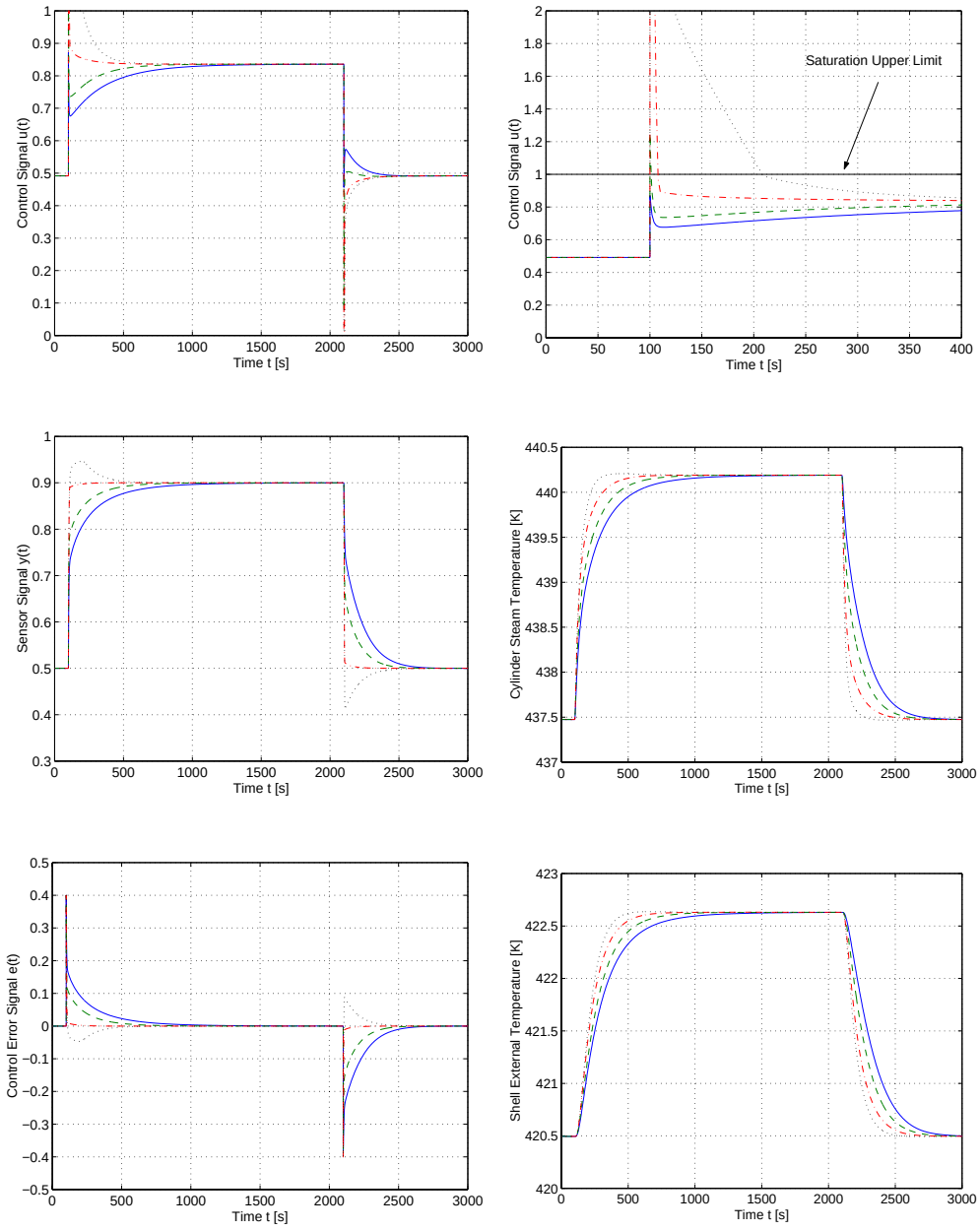

Figure 7.9 $T_p = \tau_M = 94$. $k_r = 0$, $\lambda = 1$ (—). $k_r = 0$, $\lambda = 0.5$ (−−). $k_r = 0$, $\lambda = 0.05$ (−·−·). $k_r = 5$, $\lambda = 0.05$ (···). The delay between the steam temperature and the external shell temperature can be estimated around 7 seconds.

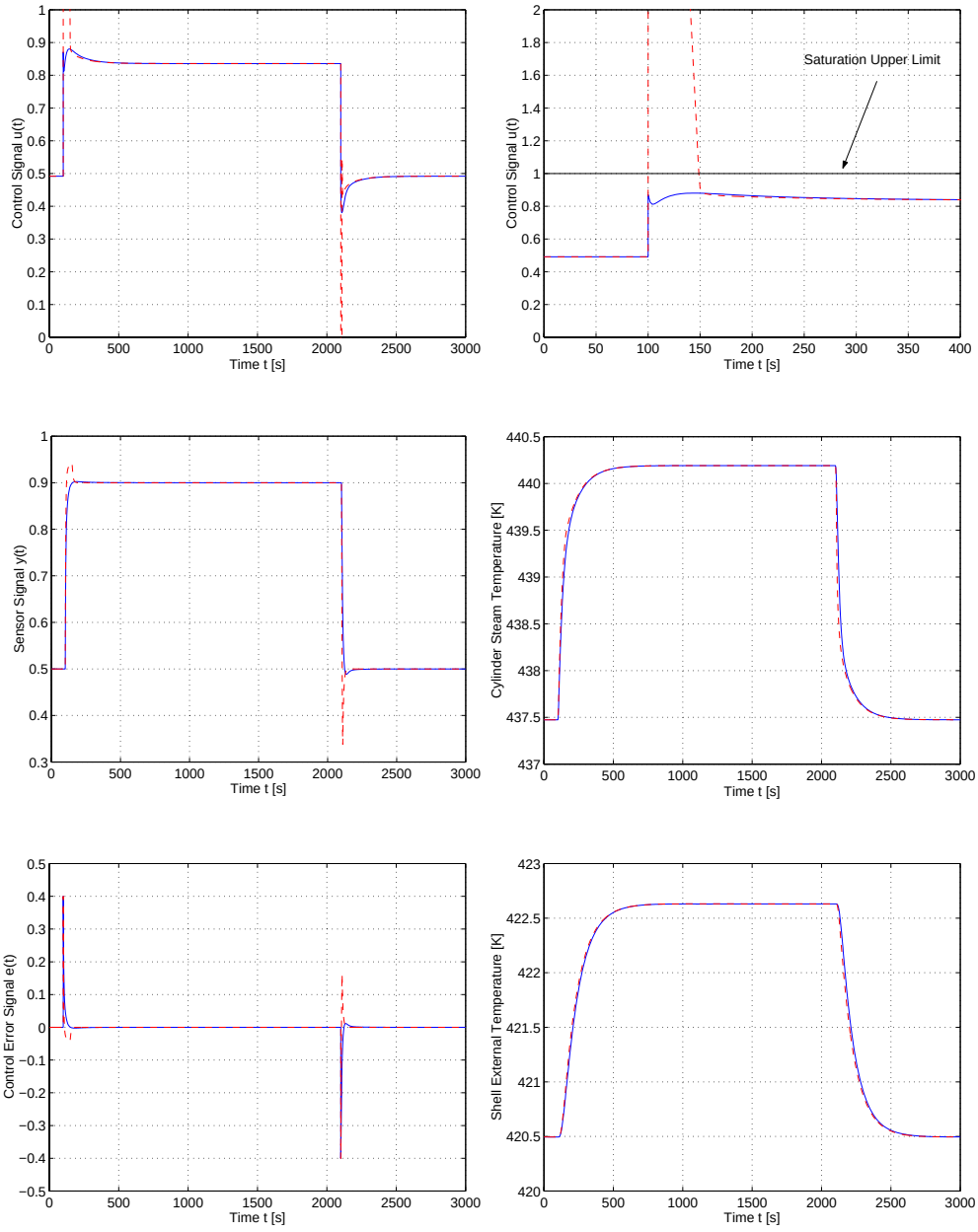

Figure 7.10 $T_p = \tau_m = 6.5$. $k_r = 0$, $\lambda = 1$ (—). $k_r = 5$, $\lambda = 0.05$ ($\cdots$). The delay between the steam temperature and the external shell temperature can be estimated around 7 seconds.

7.4 Disturbances Rejection

In this section will be presented the results relative to the disturbances. In the previous section it is showed that the difference in the responses is small when the time constants τ_M and τ_m are used to tune the controller, instead to use τ_1 and τ_2 for each set point. In this view only the time constant mean value τ_M, which has given the best result in the step response, will be used to study the disturbance rejection proprieties of the closed loop system. In particular the following type of disturbances are going to be applied on the system:

- *Pressure disturbance* in the supplying tank, which can be seen as pressure variations in the supplier or in the header. They can happen for many several different reasons as losses in joints, pressure source malfunctions, etc..
- *Web Disturbance*, which is a variation in the heat flow. It can be seen as a parameter variation in the incoming paper, for example variations in temperature, moisture content, etc..

Since the best results in the step response are obtained with $Tp = \tau_M$, $\lambda = 0.05$ and $k_r = 5$ only disturbance simulations with this controller configuration will be presented. The Fig. (7.11) shows the results when a disturbance in the tank is applied. The results concerning the web disturbances are shown in Fig. (7.12) and they show that is not possible to control properly this kind of disturbance, mainly because the sensor almost does not “fill” the variation in pressure, despite inside the pipe there is a strong mass flow variation. Even more it can be observed that disturbance magnitude increments produce proportional temperature variations.

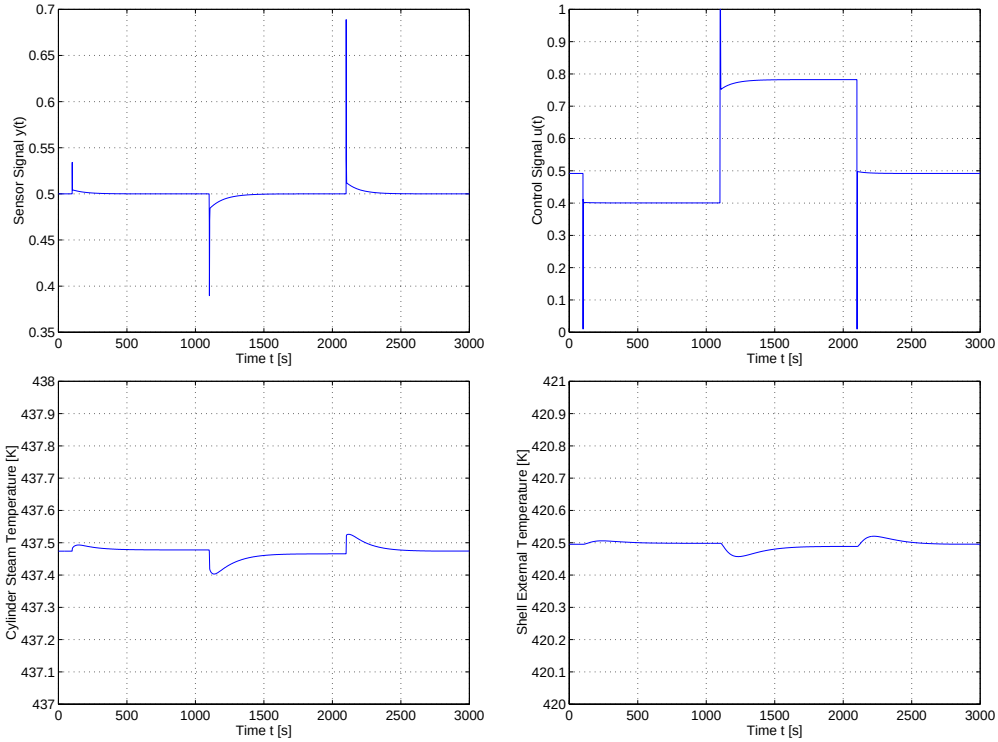

Figure 7.11 Pressure disturbance in the supplying tank. Amplitude ±7.5%.

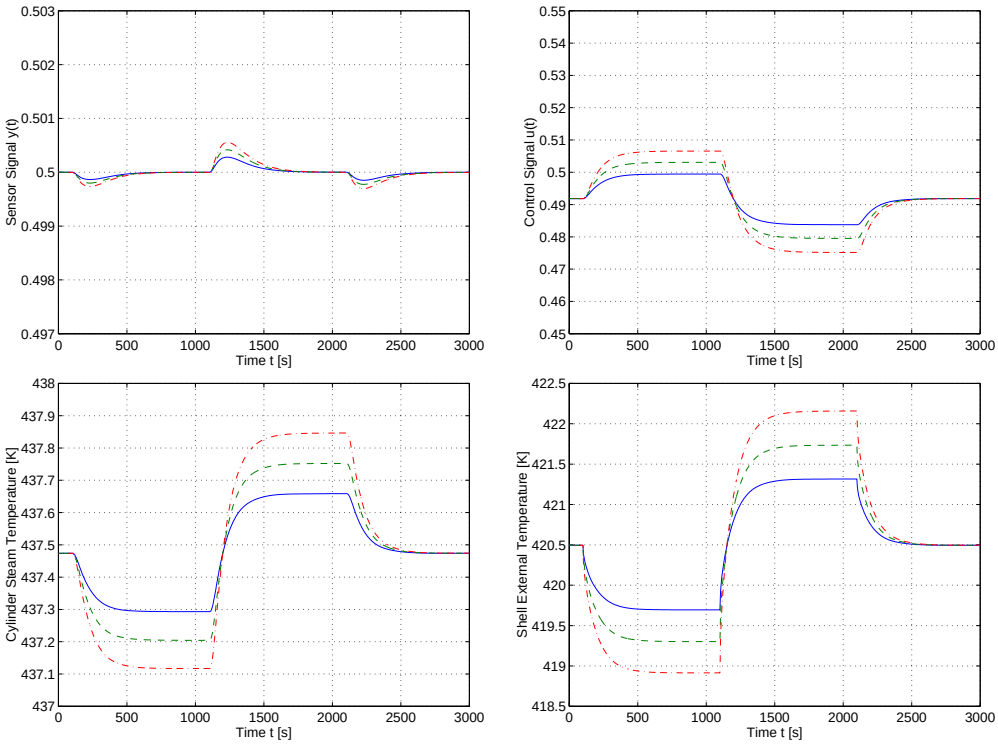

Figure 7.12 Generic heat flow disturbance in the paper web. Amplitude ±5% (—), ±7.5% (– –) and ±10% (– · –).

7.5 Simulation Summary

First of all it has been shown that the PI controller, yields good step response improvements without change of the system structure. In fact, it is very cheap to have only one valve for a whole cylinder group. The performances will be improved very much by putting a valve near each cylinder, but on the other hand the costs will raise a lot. The most important improvement is achieved by the setpoint weight, which permit to have an overshoot in the 3 port pipe, but on the other hand this permits to have quicker changes in pressure inside the cylinder.

The disturbances have been applied in the supplying tank and in paper web. As it has been shown it was possible to reject the pressure disturbance in the tank, instead it was not possible to reject the heat flow disturbance in the web. In fact that in the tank effect only the pressure, while the that other in the web effects both pressure and mass flow, but just the pressure is controlled.

8. Conclusions

In this chapter problems and results will be analyzed. First of all the software is considered, then the model is analyzed and finally some suggestions are given for future work.

8.1 Software

Modelica™ software has been used to model and simulate the model. It is was chosen mainly because it is a Object Oriented Language, which the main features are:

- *Classes and object* handling, which is a general propriety of object oriented languages
- *Non–causal modeling*, which allows to treat the connectors as port and neither as input nor output
- *Hybrid modeling*, which allows to mix in the same model several different system types as: electrical, mechanical, hydraulic, electronics, etc..

Despite these excellent Modelica characteristics, Dymola, the simulator used, was not as good. The main problem was due to the initial conditions calculus, in fact if the system under analysis is far from the steady states conditions the simulation does not start at all. This is due to the Newton method, which does not converge. At the moment a great deal of effort is put by the designers to solve this problem. On the other hand, it can be said that this simulator was initially designed to simulate robotic systems in which the initial conditions are not a problem and they can easily calculated.

8.2 Model

First of all a new sub-library called PaperDry had been developed, which was built up using the basic models provided from the ThermoFlow library, then the thermohydraulic model was as well built up. The trouble was now to choose the parameters of each model, as dimensions, heat transfer coefficients and so on. The answer came from an engineer of ABB company, who provided me, even if in informal way, same data and dimensions. The next step was to get the same behavior as the real process and this was quite hard even because the difficulty to start simulations each time that the parameters was changed. Anyway at the end the same behavior as in real plants was reproduced.

8.3 Controller and Simulations

Since the model had built up was time to think about performance improvements. The goal was to improve the performances without change

the process structure, and using a low cost device. The controller that well fit in this view it is a PI controller, one of the simplest form of it was used. The results has been pretty good, improvements in set point changes and disturbances rejection were obtained. The main results are:

- The step response is quicker, which means that the process is ready at new working conditions in less time, this also reflect on money savings
- The disturbances in pressure in the supplier tank can be properly rejected, which means that in case of malfunctions the system works in the same set point.
- The disturbances in heat flow changes inside the web can not properly rejected even if they do not effect very much the performances of the system.

Overall, it can be said that the system behaves as the real one and the controller improves a lot the performances.

8.4 Future Work

The work it has been very interesting and a lot results were found, but improvements and extensions should be done. In the following list some ideas about future work is provided:

- The model can be easily extended to simulate several groups
- The paper web model can be changed with a complex one to be able to get all the phenomena inside it
- The structure of the system can be changed in order to improve the performances of the real process
- The controller can be replaced with a more complicate and powerful one, even if this will increase the costs

The previous suggestions does not claim to be complete, but they only give some start points for future work.

A. Modelica™ Code

The following code is part of the PaperDry library; it inherits basic models from the ThermoFlow library, which is still under development at the Department automatic control at Lund University.

The latest version of the ThermoFlow library is available on the web page `http://www.control.lth.se/ hubertus/ThermoFlow/`.

```
model Cylinder
  constant Real pi=3.14159;
  parameter SIunits.Diameter D_cyl=1 "Cylinder Diameter";
  parameter SIunits.Length L_cyl=5 "Cylinder Axial Length";
  //parameter SIunits.Pressure p_max=3e5;
  //parameter SIunits.Pressure p_min=1e5;
  //SIunits.Length Lev;
  extends
    Modelica.ThermoFlow.ControlVolumes.ControlVolumeThreePortSingleDynamic2V1L(
    V0=L_cyl*D_cyl^2*pi/4,
    p(start=ones(1)*2.5e5),
    h(start=ones(1)*1.0e6),
    redeclare model Medium extends
        Modelica.ThermoFlow.MediumModels.Water.SaturatedWaterSteamMedium_ph;
      end Medium);
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[1, 1],
      component=[20, 20]),
    Window(
      x=0.21,
      y=0.09,
      width=0.62,
      height=0.77),
    Icon(
      Rectangle(extent=[-4, 4; 4, -4], style(color=41, fillColor=41)),
      Rectangle(extent=[-100, 100; 101, -100], style(
          color=0,
          gradient=2,
          fillColor=41)),
      Rectangle(extent=[-10, 10; 10, -10], style(color=0, fillColor=0))));
  Modelica.Blocks.Interfaces.OutPort S_Lev annotation (
    extent=[110, 64; 90, 84],
    rotation=-180,
    layer="icon");
equation
  hl = props[1].hl;
  dl = props[1].dl;
  S_Lev.signal[1] = 1 - sqr(1 - (1 - props[1].x)*props[1].d/props[1].dl);
end Cylinder;
```

```
model FictWallDist
  parameter SIunits.Temp_C Tweb=85;
  parameter SIunits.Length D=1.8 "Cyl Diameter";
  parameter SIunits.Length L=6 "Cyl Axial Length";
  parameter SIunits.ThermalConductivity Kp=200;
  annotation (
    Coordsys(
```

```
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Icon(
      Rectangle(extent=[-100, 20; 100, -20], style(
          color=0,
          thickness=2,
          fillColor=67,
          fillPattern=10)),
      Text(
        extent=[0, 60; 100, 20],
        string="%name",
        style(color=0, thickness=2)),
      Polygon(points=[-60, 20; -80, -20; -40, -20; -60, 20], style(
          color=0,
          thickness=2,
          fillColor=69,
          fillPattern=10)),
      Polygon(points=[60, 20; 40, -20; 80, -20; 60, 20], style(
          color=0,
          thickness=2,
          fillColor=69,
          fillPattern=10)),
      Polygon(points=[0, 20; -20, -20; 20, -20; 0, 20], style(
          color=0,
          thickness=2,
          fillColor=69,
          fillPattern=10))),
    Window(
      x=0.13,
      y=0.08,
      width=0.63,
      height=0.75));
  Modelica.ThermoFlow.Interfaces.HeatTransferLumped q annotation (extent=[-20
        , -50; 20, -10]);
  Modelica.Blocks.Interfaces.InPort u annotation (extent=[-10, 10; 10, 30],
      rotation=-90);
equation
  q.q = L*D*3.14159*Kp*(q.T - Tweb - 273.15)*(1 + u.signal[1]);
end FictWallDist;
```

```
model Pipe
  constant Real pi=3.14159;
  parameter SIunits.Length D=1.0 "Pipe Diameter";
  parameter SIunits.Length L=1.0 "Pipe Length";
  parameter Integer n=1 "Number of Segments";
  parameter SIunits.Pressure p0l=10.0e5;
  parameter SIunits.Pressure p0r=9.0e5;
  parameter SIunits.Enthalpy h0=2.5e6;
  parameter SIunits.MassFlow mdot0=0.1;
  extends
    Modelica.ThermoFlow.ControlVolumes.ControlVolumeTwoPortSingleDistributed(
    mdot(start=ones(n + 1)*mdot0),
    p(start=if n == 1 then ones(n)*p0l else linspace(p0l, p0r, n)),
    h(start=ones(n)*h0),
    A0=D^2*pi/4,
    Dhyd0=D,
    V0=L*D^2*pi/4,
    L0=L,
    redeclare model Medium extends
        Modelica.ThermoFlow.MediumModels.Water.WaterSteamMedium_ph;    end
```

```
      Medium,
    redeclare model PLoss extends
        Modelica.ThermoFlow.TestModels.ComponentLib.Pipes.PipePressureLossDistributed
        ;    end PLoss);
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Window(
      x=0.29,
      y=0.11,
      width=0.6,
      height=0.69),
    Icon(Rectangle(extent=[-100, 62; 100, -62], style(gradient=2, fillColor=67
         ))));
equation
  dV = zeros(n);
end Pipe;
```

```
model Pipe3P
  constant Real pi=3.14159;
  parameter SIunits.Volume V0=0.4 "Central Section Volume";
  parameter SIunits.Length Lb=1.0 "Outlet Pipe Lengths";
  parameter SIunits.Length Lc=1.0 "Outlet Pipe Lengths";
  parameter SIunits.Length Db=0.15 "Port B Diameter";
  parameter SIunits.Length Dc=1.15 "Port C Diameter";
  parameter SIunits.Pressure p0=1e5 "Initial Condition";
  parameter SIunits.Enthalpy h0=1e6 "Initial Condition";
  extends Modelica.ThermoFlow.Interfaces.FlowThreePortSingle;
  parameter SIunits.Pressure p_max=12.0e5;
  parameter SIunits.Pressure p_min=7.0e5;
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Window(
      x=0.24,
      y=0.05,
      width=0.53,
      height=0.77),
    Icon(Rectangle(extent=[-100, 62; 100, -62], style(gradient=2, fillColor=67
         )), Rectangle(extent=[-62, 100; 62, 60], style(gradient=1, fillColor=
           67))));
  Modelica.ThermoFlow.ControlVolumes.ControlVolumeThreePortSingleDynamic
    ThreePort(
    V0=V0,
    p(start=ones(1)*p0),
    h(start=ones(1)*h0),
    redeclare model Medium extends
        Modelica.ThermoFlow.MediumModels.Water.WaterSteamMedium_ph;    end
      Medium) annotation (extent=[-20, -20; 20, 20]);
  Modelica.Blocks.Interfaces.OutPort S_Pres annotation (extent=[-110, -60; -90
       , -40], rotation=-180);
  Modelica.ThermoFlow.PartialHydraulicModels.FlowModelTwoPortSingleLumpedDyn
    OutPortB(
    redeclare model PLoss extends
        Modelica.ThermoFlow.TestModels.ComponentLib.Pipes.PipePressureLossLumped
        ;    end PLoss,
    L=Lb,
```

```
    Dhyd=Db,
    A=pi*Db^2/4) annotation (extent=[40, -20; 80, 20]);
  Modelica.ThermoFlow.PartialHydraulicModels.FlowModelTwoPortSingleLumpedDyn
    OutPortC(
    redeclare model PLoss extends
        Modelica.ThermoFlow.TestModels.ComponentLib.Pipes.PipePressureLossLumped
        ;    end PLoss,
    L=Lc,
    Dhyd=Dc,
    A=pi*Dc^2/4) annotation (extent=[-20, 38; 20, 78], rotation=90);
  connect(ThreePort.a, a) annotation (points=[-20, 0; -100, 0]);
  connect(OutPortB.b, b) annotation (points=[80, 0; 100, 0]);
  connect(ThreePort.b, OutPortB.a) annotation (points=[20, 0; 40, 0]);
  connect(OutPortC.b, c) annotation (points=[1.11022e-15, 78; 0, 100]);
  connect(OutPortC.a, ThreePort.c) annotation (points=[-1.11022e-15, 38; 0, 20
        ]);
equation
  S_Pres.signal[1] = (ThreePort.a.p - p_min)/(p_max - p_min);
  // R.signal[1]=Dc;
end Pipe3P;
```

```
model ShellDyn
  constant Real pi=3.14159;
  constant Real CtoK=273.15 "Celsius To Kelvin";
  parameter Integer n(min=3) = 4 "Discretization points";
  parameter SIunits.Length d=0.025 "Thickness Shell";
  parameter SIunits.Length D=1.5 "Cylinder Diameter Length";
  parameter SIunits.Length L=5 "Cylinder Axial Length";
  parameter SIunits.ThermalConductivity lambda=45
    "Cylinder Shell Thermal Conductivity";
  parameter SIunits.CoefficientOfHeatTransfer hc=1500
    "Heat tranfer coefficient Shell-...";
  parameter SIunits.SpecificHeatCapacity cp=500 "Specific Heat Capacity";
  parameter SIunits.Density rho=7300 "Density";
  SIunits.Temp_K T[n + 1](start=400*ones(n + 1));
  SIunits.HeatFlux q[n];
  SIunits.Length dx;
  SIunits.Area A;
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Diagram,
    Icon(
      Text(
        extent=[0, 60; 100, 20],
        string="%name",
        style(color=0, thickness=2)),
      Rectangle(extent=[-100, 20; 100, -20], style(
          color=0,
          thickness=2,
          fillColor=45,
          fillPattern=10)),
      Polygon(points=[-60, 20; -40, -20; -80, -20; -60, 20], style(
          color=0,
          thickness=2,
          fillColor=41,
          fillPattern=10)),
      Polygon(points=[0, 20; 20, -20; -20, -20; 0, 20], style(
          color=0,
```

```
          thickness=2,
          fillColor=41,
          fillPattern=10)),
      Polygon(points=[60, 20; 80, -20; 40, -20; 60, 20], style(
          color=0,
          thickness=2,
          fillColor=41,
          fillPattern=10))),
    Window(
      x=0.12,
      y=0.18,
      width=0.73,
      height=0.95));
  Modelica.ThermoFlow.Interfaces.HeatTransferLumped q_S annotation (extent=[-
        20, -50; 20, -10]);
  Modelica.ThermoFlow.Interfaces.HeatTransferLumped q_N annotation (extent=[-
        20, 10; 20, 50]);
equation
  dx = d/n;
  A = L*D*pi;
  q_S.q = A*hc*(q_S.T - T[1]);
  der(T[1])*rho*cp*2*dx*dx = lambda*(6*q_S.q*dx/(lambda*A) - 7*T[1] + 8*T[2]
     - T[3]);
  for i in 2:n loop
    der(T[i])*cp*rho*dx*dx = lambda*(T[i - 1] - 2*T[i] + T[i + 1]);
  end for;
  der(T[n + 1])*rho*cp*2*dx*dx = lambda*(6*q_N.q*dx/(lambda*A) - 7*T[n + 1] +
    8*T[n] - T[n - 1]);
  for i in 1:n loop
    q[i]*dx = lambda*A*(T[i] - T[i + 1]);
  end for;
  T[n + 1] = q_N.T;
end ShellDyn;
```

```
model Sink
  parameter SIunits.Pressure p0=1.0e5;
  parameter SIunits.Enthalpy h0=2.4e6;
  parameter SIunits.Volume V0=1.0;
  extends Modelica.ThermoFlow.Interfaces.FlowOnePortSingle;
  extends Modelica.ThermoFlow.Interfaces.BalanceSetSingleLumped(
    de=zeros(1),
    dV=zeros(1),
    p(start=ones(1)*p0),
    h(start=ones(1)*h0));
  extends Modelica.ThermoFlow.MediumModels.Water.SaturatedWaterSteamMedium_ph(
      p(start=ones(n)*p0), h(start=h0));
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Window(
      x=0.14,
      y=0.09,
      width=0.69,
      height=0.84),
    Icon(Rectangle(extent=[60, 50; 80, -50], style(color=65, fillColor=65)),
        Rectangle(extent=[-100, 62; 60, -62], style(
          color=0,
          gradient=2,
          fillColor=67))));
```

```
equation
  p[1] = p0;
  V[1] = V0;
  dm[1] = 0.0;
  q_source[1] = 0.0;
  a.h = h[1];
  a.d = d[1];
  a.p = p[1];
  h[1] = h0;
  a.dG = 0;
end Sink;
```

```
model Sink_V
  extends Modelica.ThermoFlow.Interfaces.FlowOnePortSingle;
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Window(
      x=0.33,
      y=0.31,
      width=0.63,
      height=0.76),
    Icon(
      Rectangle(extent=[60, 50; 80, -50], style(color=65, fillColor=65)),
      Rectangle(extent=[-100, 62; 60, -62], style(gradient=2, fillColor=67)),
      Rectangle(extent=[-20, 62; 20, -62], style(gradient=2, fillColor=41))));
equation
  a.dG = 0;
end Sink_V;
```

```
model Source_Dist
  parameter Real height1=1 "Height of first ramp";
  parameter Real height2=-height1 "Height of  second ramp";
  parameter Real duration1=2 "Duration of ramp";
  parameter Real duration2=duration1 "Duration of ramp";
  parameter Real offset=0 "Offset of output signal";
  parameter SIunits.Time start1=1000
    "Output = offset for: start2 < time < start1";
  parameter SIunits.Time start2=2500
    "Output = offset for: start2 < time < start1";
  parameter SIunits.Time start3=4000
    "Output = offset for: start3 < time < start1";
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Diagram,
    Icon(
      Line(points=[-80, 80; -80, -90], style(color=8)),
      Polygon(points=[-80, 90; -88, 68; -72, 68; -80, 90], style(color=8,
            fillColor=8)),
      Line(points=[-90, -20; 80, -20], style(color=8)),
      Polygon(points=[90, -20; 68, -12; 68, -28; 90, -20], style(
          color=8,
          fillColor=8,
          fillPattern=1)),
```

```
      Line(points=[-80, 0; -60, 0; -50, -60; -20, -60; 0, 60; 30, 60; 40, 0;
            60, 0], style(color=0)),
      Rectangle(extent=[-100, 100; 100, -100]),
      Text(extent=[-100, -100; 100, -140], string="%name")),
    Window(
      x=0.28,
      y=0.12,
      width=0.52,
      height=0.73));
  Modelica.Blocks.Interfaces.OutPort y annotation (extent=[90, -10; 110, 10]);
  Modelica.Blocks.Sources.Ramp Ramp1(
    height={height1},
    duration={duration1},
    offset={offset},
    startTime={start1}) annotation (extent=[-80, 40; -60, 60]);
  Modelica.Blocks.Sources.Ramp Ramp2(
    height={-height1 + height2},
    duration={duration1 + duration2},
    offset={0},
    startTime={start2}) annotation (extent=[-80, -10; -60, 10]);
  Modelica.Blocks.Math.Add3 Adder annotation (extent=[20, -10; 40, 10]);
  Modelica.Blocks.Sources.Ramp Ramp3(
    height={-height2},
    duration={duration2},
    offset={0},
    startTime={start3}) annotation (extent=[-80, -60; -60, -40]);
  connect(Adder.outPort, y) annotation (points=[41, 0; 100, 0]);
  connect(Ramp1.outPort, Adder.inPort1) annotation (points=[-59, 50; -20, 50;
        -20, 8; 18, 8]);
  connect(Adder.inPort2, Ramp2.outPort) annotation (points=[18, 0; -59, 0]);
  connect(Ramp3.outPort, Adder.inPort3) annotation (points=[-59, -50; -20, -50
        ; -20, -8; 18, -8]);
end Source_Dist;
```

```
model TankDist
  parameter SIunits.Pressure p0=3.0e5;
  parameter SIunits.Enthalpy h0=2.5e6;
  parameter SIunits.Volume V0=1.0;
  extends Modelica.ThermoFlow.Interfaces.FlowOnePortSingle;
  extends Modelica.ThermoFlow.Interfaces.BalanceSetSingleLumped(
    de=zeros(1),
    dV=zeros(1),
    p(start=ones(1)*p0),
    h(start=ones(1)*h0));
  extends Modelica.ThermoFlow.MediumModels.Water.SaturatedWaterSteamMedium_ph(
      p(start=ones(n)*p0), h(start=h0));
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Window(
      x=0.07,
      y=0.04,
      width=0.51,
      height=0.75),
    Diagram,
    Icon(
      Ellipse(extent=[-100, 100; 100, 60], style(gradient=3, fillColor=67)),
      Rectangle(extent=[-100, 80; 100, -100], style(gradient=1, fillColor=67))
        ,
```

```
      Text(
        extent=[-80, -100; 80, -140],
        string="%name",
        style(
          color=73,
          pattern=0,
          fillColor=74))));
  Modelica.Blocks.Interfaces.InPort u annotation (extent=[-10, 90; 10, 110],
      rotation=-90);
equation
  p[1] = p0;
  V[1] = V0;
  dm[1] = 0.0;
  q_source[1] = 0.0;
  a.h = h[1];
  a.d = d[1];
  a.p = p[1]*(1 + u.signal[1]);
  h[1] = h0;
  a.dG = a.G_norm;
end TankDist;
```

```
model ThHydDisturb
  constant Real pi=3.14159;
  // ================================================================
  // ThermoHydraulic Parameters
  // ================================================================
  parameter SIunits.Pressure p_tank=8.3e5;
  parameter SIunits.Pressure p_sink=6.7e5;
  // ================================================================
  // Dimensional Parameters
  // ================================================================
  parameter SIunits.Diameter D=1.80 "Cylinder Diameter";
  parameter SIunits.Length Lc=6.0 "Cylinder Axial Length";
  // ================================================================
  // Controller Parameters
  // ================================================================
  constant Real Tp1up=3.7813 "First Time Constant Step Up";
  constant Real Tp2up=74.5019 "Second Time Constant Step Up";
  constant Real Tp1dw=9.2702 "First Time Constant Step Down";
  constant Real Tp2dw=113.1037 "Second Time Constant Step Down";
  parameter Real Tp=94 "Controller Current Time Constant";
  parameter Real Kp=1.055 "Open Loop Gain";
  parameter Real lambda=0.05 "lambda parameter";
  annotation (
    Coordsys(
      extent=[-200, -100; 200, 100],
      grid=[1, 1],
      component=[20, 20]),
    Diagram(
      Text(
        extent=[170, -15; 190, -25],
        string="p0=%p_sink",
        style(color=0)),
      Text(
        extent=[-200, -15; -180, -25],
        string="p0=%p_tank",
        style(color=0)),
      Text(
        extent=[170, -85; 190, -95],
        string="p0=%p_sink",
        style(color=0))),
```

```
  Window(
    x=0.02,
    y=0.09,
    width=0.91,
    height=0.82),
  Icon(
    Rectangle(extent=[-200, 100; 200, -100], style(color=0)),
    Text(
      extent=[-190, 50; 190, -50],
      string="Disturbancies",
      style(color=0)),
    Text(extent=[-200, -100; 200, -150], string="%name"),
    Rectangle(extent=[-200, 100; 200, -100], style(color=0)),
    Text(extent=[-200, -100; 200, -150], string="%name")));
Modelica.ThermoFlow.PaperDry.Cylinder Cyl(D_cyl=D, L_cyl=Lc) annotation (
    extent=[-10, 0; 70, -40]);
Modelica.Blocks.Sources.Constant SetPoint_Out(k={1}) annotation (extent=[90
      , 0; 110, 20]);
Modelica.ThermoFlow.PaperDry.ValveLin Valve_Out_S(D=0.15, L=0.2) annotation
  (extent=[110, -30; 130, -10]);
Modelica.ThermoFlow.PaperDry.ValveLin Valve_Out_W(D=0.05, L=0.2) annotation
  (extent=[140, -100; 160, -80]);
Modelica.Blocks.Math.Feedback Fback_Lev annotation (extent=[60, -50; 80, -70
      ]);
Modelica.Blocks.Sources.Constant SetPoint_Lev(k={0.005}) annotation (extent=
      [34, -70; 54, -50]);
Modelica.Blocks.Nonlinear.Limiter Lim_Lev(uMax={1}, uMin={0.01}) annotation
  (extent=[120, -70; 140, -50]);
Modelica.ThermoFlow.PaperDry.Sink_V Sink_Z annotation (extent=[-40, -100;
      -20, -80]);
Modelica.ThermoFlow.PaperDry.Pipe3P T_Pipe(
  V0=1,
  Lb=1.5,
  Lc=5,
  Db=0.15,
  Dc=0.67,
  p0=750000,
  h0=2.8e6,
  p_max=p_tank,
  p_min=p_sink) annotation (extent=[-90, -10; -70, -30]);
Modelica.ThermoFlow.PaperDry.Pipe Pipe(
  D=0.15,
  L=25,
  n=3,
  h0=2.808e6,
  L0=30,
  Dhyd0=0.15) annotation (extent=[-60, -30; -20, -10]);
Modelica.ThermoFlow.PaperDry.ValveLinDyn Valve_In(
  T=5,
  D=0.25,
  L=0.50) annotation (extent=[-120, -30; -100, -10]);
Modelica.ThermoFlow.PaperDry.Pipe Pipe_Z(
  D=0.67,
  L=25,
  n=3,
  h0=2.808e6,
  L0=30,
  Dhyd0=0.67) annotation (extent=[-50, -80; -70, -100], rotation=180);
Modelica.ThermoFlow.PaperDry.Pipe Header(
  D=0.25,
  L=10,
  n=3,
```

```
    h0=2.808e6,
    L0=10,
    Dhyd0=0.25) annotation (extent=[-130, -35; -170, -5]);
  Modelica.Blocks.Math.Gain Gain_Lev(k={-10000}) annotation (extent=[90, -70;
        110, -50]);
  Modelica.ThermoFlow.PaperDry.ShellDyn Shell annotation (extent=[-10, -2; 70
        , 18]);
  Modelica.ThermoFlow.PaperDry.TankDist Tank_S(p0=p_tank) annotation (extent=[
        -180, -40; -200, 0]);
  Modelica.ThermoFlow.PaperDry.Source_Dist Dist_Tank(
    height1=0.15,
    height2=-0.15,
    duration2=2,
    offset=0,
    start1=1000,
    start2=2500,
    start3=4000) annotation (extent=[-200, 40; -180, 60]);
  Modelica.ThermoFlow.PaperDry.SinkDist Sink_S(p0=p_sink) annotation (extent=[
        170, -30; 190, -10]);
  Modelica.ThermoFlow.PaperDry.FictWallDist Web annotation (extent=[-10, 10;
        70, 30]);
  Modelica.ThermoFlow.PaperDry.Source_Dist Dist_Web(
    height1=0,
    height2=0,
    duration1=2,
    duration2=2,
    offset=0,
    start1=1000,
    start2=2500,
    start3=4000) annotation (extent=[30, 40; 50, 60]);
  Modelica.ThermoFlow.PaperDry.Source_Dist Dist_Sink_S(
    height1=0,
    height2=0,
    duration1=2,
    duration2=2,
    offset=0,
    start1=1000,
    start2=2500,
    start3=4000) annotation (extent=[140, 40; 160, 60]);
  Modelica.ThermoFlow.PaperDry.Sink Sink_W(
    p0=p_sink,
    h0=5e5,
    V0=10.0,
    n=1) annotation (extent=[170, -100; 190, -80]);
  Modelica.Blocks.Math.Feedback Fback_Pres annotation (extent=[-100, 40; -80,
        60]);
  Modelica.Blocks.Nonlinear.Limiter Lim_Pres(uMax={1}, uMin={0.001})
    annotation (extent=[-10, 40; 10, 60]);
  Modelica.Blocks.Continuous.PI PI_Pres(k={1/(lambda*Kp)}, T={Tp}) annotation
    (extent=[-70, 40; -50, 60]);
  Modelica.Blocks.Sources.Pulse Step_Pres(
    amplitude={0.4},
    period={4000},
    offset={0.25},
    startTime={1000}) annotation (extent=[-140, 40; -120, 60]);
  Modelica.Blocks.Math.Feedback Fback_Pres2 annotation (extent=[-40, 60; -20,
        40]);
  Modelica.Blocks.Math.Gain Gain_Pres(k={-5}) annotation (extent=[-100, 70;
        -80, 90]);
  connect(SetPoint_Out.outPort, Valve_Out_S.u) annotation (points=[111, 10;
        120, 10; 120, -10]);
  connect(Cyl.b, Valve_Out_S.a) annotation (points=[70, -20; 110, -20]);
```

```
  connect(Cyl.c, Valve_Out_W.a) annotation (points=[30, -40; 30, -90; 140, -90
      ]);
  connect(SetPoint_Lev.outPort, Fback_Lev.inPort1) annotation (points=[55, -60
      ; 62, -60]);
  connect(Lim_Lev.outPort, Valve_Out_W.u) annotation (points=[141, -60; 150,
      -60; 150, -80]);
  connect(T_Pipe.a, Valve_In.b) annotation (points=[-90, -20; -100, -20]);
  connect(T_Pipe.b, Pipe.a) annotation (points=[-70, -20; -60, -20]);
  connect(Pipe.b, Cyl.a) annotation (points=[-20, -20; -10, -20]);
  connect(Pipe_Z.b, Sink_Z.a) annotation (points=[-50, -90; -40, -90]);
  connect(Pipe_Z.a, T_Pipe.c) annotation (points=[-70, -90; -80, -90; -80, -30
      ]);
  connect(Header.a, Valve_In.a) annotation (points=[-130, -20; -120, -20]);
  connect(Gain_Lev.outPort, Lim_Lev.inPort) annotation (points=[111, -60; 118
      , -60]);
  connect(Fback_Lev.outPort, Gain_Lev.inPort) annotation (points=[79, -60; 88
      , -60]);
  connect(Shell.q_S, Cyl.q) annotation (points=[30, 5; 30, -20]);
  connect(Cyl.S_Lev, Fback_Lev.inPort2) annotation (points=[70, -34.8; 70,
      -34.8; 70, -52]);
  connect(Header.b, Tank_S.a) annotation (points=[-170, -20; -180, -20]);
  connect(Dist_Tank.y, Tank_S.u) annotation (points=[-180, 50; -170, 50; -170
      , 20; -190, 20; -190, 0]);
  connect(Sink_S.a, Valve_Out_S.b) annotation (points=[170, -20; 130, -20]);
  connect(Web.q, Shell.q_N) annotation (points=[30, 17; 30, 11]);
  connect(Dist_Sink_S.y, Sink_S.u) annotation (points=[160, 50; 178, 50; 178,
      -14]);
  connect(Dist_Web.y, Web.u) annotation (points=[50, 50; 60, 50; 60, 30; 30,
      30; 30, 22]);
  connect(Valve_Out_W.b, Sink_W.a) annotation (points=[160, -90; 170, -90]);
  connect(Fback_Pres.outPort, PI_Pres.inPort) annotation (points=[-81, 50; -72
      , 50]);
  connect(Step_Pres.outPort, Fback_Pres.inPort1) annotation (points=[-119, 50
      ; -98, 50]);
  connect(PI_Pres.outPort, Fback_Pres2.inPort1) annotation (points=[-49, 50;
      -38, 50]);
  connect(Fback_Pres2.outPort, Lim_Pres.inPort) annotation (points=[-21, 50;
      -12, 50]);
  connect(Gain_Pres.outPort, Fback_Pres2.inPort2) annotation (points=[-79, 80
      ; -30, 80; -30, 58]);
  connect(Step_Pres.outPort, Gain_Pres.inPort) annotation (points=[-119, 50;
      -110, 50; -110, 80; -102, 80]);
  connect(Lim_Pres.outPort, Valve_In.u) annotation (points=[11, 50; 20, 50; 20
      , 100; -150, 100; -150, 10; -110, 10; -110, -10]);
  connect(Fback_Pres.inPort2, T_Pipe.S_Pres) annotation (points=[-90, 42; -90
      , -15]);
equation
  Sink_Z.a.p = Cyl.p[1];
  Sink_Z.a.d = Cyl.d[1];
  Sink_Z.a.h = Cyl.h[1];
end ThHydDisturb;
```

```
model ValveLin
  constant Real pi=3.14159;
  parameter SIunits.Diameter D=1.5e-1 "Valve Diameter";
  parameter SIunits.Length L=0.3 "Valve Length";
  SIunits.Area Av;
  extends
    Modelica.ThermoFlow.PartialHydraulicModels.FlowModelTwoPortSingleLumped(
    A=D^2*pi/4,
    Dhyd=D,
```

```
    L=L);
  replaceable model water1
    extends ThemoFlow.MediumModels.SaturatedSteamWater_ph;
    annotation (Coordsys(
        extent=[-100, -100; 100, 100],
        grid=[2, 2],
        component=[20, 20]));
  end water1;
  replaceable model water
    extends ThermoFlow.MediumModels.Water.SaturatedWaterSteamMedium_ph;
    annotation (Coordsys(
        extent=[-100, -100; 100, 100],
        grid=[2, 2],
        component=[20, 20]));
  end water;
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Icon(
      Line(points=[-20, 60; -20, 100; 20, 100; 20, 60], style(color=0)),
      Rectangle(extent=[-100, 62; 100, -62], style(
          color=0,
          gradient=2,
          fillColor=67)),
      Rectangle(extent=[-20, 100; 20, 56], style(
          color=0,
          pattern=0,
          gradient=1,
          fillColor=67))),
    Window(
      x=0.14,
      y=0.06,
      width=0.8,
      height=0.79));
  Modelica.Blocks.Interfaces.InPort u annotation (extent=[-10, 90; 10, 110],
      rotation=-90);
equation
  Av = A*u.signal[1];
  Ploss[1] = if mdot[1] > 0 then mdot[1]^2/Av^2/a.d else -mdot[1]^2/Av^2/b.d;
  G_norm[1] = if mdot[1] > 0 then mdot[1]*mdot[1]/a.d/Av else -mdot[1]*mdot[1]
    /b.d/Av;
  0 = (a.p - b.p)*Av - Ploss[1]*dz*Dhyd*pi;
end ValveLin;
```

```
model ValveLinDyn
  constant Real pi=3.14159;
  parameter Real T=10 "Valve time constant";
  Real x;
  parameter SIunits.Diameter D=1.5e-1 "Valve Diameter";
  parameter SIunits.Length L=0.3 "Valve Length";
  SIunits.Area Av;
  extends
    Modelica.ThermoFlow.PartialHydraulicModels.FlowModelTwoPortSingleLumped(
    A=D^2*pi/4,
    Dhyd=D,
    L=L);
  replaceable model water1
    extends ThemoFlow.MediumModels.SaturatedSteamWater_ph;
    annotation (Coordsys(
```

```
        extent=[-100, -100; 100, 100],
        grid=[2, 2],
        component=[20, 20]));
  end water1;
  replaceable model water
    extends ThermoFlow.MediumModels.Water.SaturatedWaterSteamMedium_ph;
    annotation (Coordsys(
        extent=[-100, -100; 100, 100],
        grid=[2, 2],
        component=[20, 20]));
  end water;
  annotation (
    Coordsys(
      extent=[-100, -100; 100, 100],
      grid=[2, 2],
      component=[20, 20]),
    Icon(
      Line(points=[-20, 60; -20, 100; 20, 100; 20, 60], style(color=0)),
      Rectangle(extent=[-100, 62; 100, -62], style(
          color=0,
          gradient=2,
          fillColor=67)),
      Rectangle(extent=[-20, 100; 20, 56], style(
          color=0,
          pattern=0,
          gradient=1,
          fillColor=67))),
    Window(
      x=0.11,
      y=0.03,
      width=0.8,
      height=0.79));
  Modelica.Blocks.Interfaces.InPort u annotation (extent=[-10, 90; 10, 110],
      rotation=-90);
equation
  der(x) = -x/T + u.signal[1]/T;
  Av = A*x;
  Ploss[1] = if mdot[1] > 0 then mdot[1]^2/Av^2/a.d else -mdot[1]^2/Av^2/b.d;
  G_norm[1] = if mdot[1] > 0 then mdot[1]*mdot[1]/a.d/Av else -mdot[1]*mdot[1]
    /b.d/Av;
  0 = (a.p - b.p)*Av - Ploss[1]*dz*Dhyd*pi;
end ValveLinDyn;
```

B. Bibliography

[1] Elmqvist H. *A Structured Model Language for Large Continuous Systems*. PhD thesis, Lund University Dept. of Automatic Control, 1978.

[2] Barducci I. *Trasmissione del Calore*. Collana di Fisica Tecnica, 1996.

[3] Åström K. J. Panagopoulos H., Hägglund T. The lambda method for tuning PI controllers. Technical report, Department of Automatic Control, 1997.

[4] Håkar Persson. *Dynamic Modeling and Simulation of Multi-Cylinder Paper Dryers*. PhD thesis, Lund University Dept. of Chemical Engineering I, 1998.

[5] Hägglund T. Åström K. J. *PID Controllers: Theory, Design and Tuning*. International Society of America, 1995.

[6] Wagner F. Tummescheit H., J. Eborn J. Thermoflow: a thermo-hydraulic library in modelica. draft version. Technical report, Lund University, 2000.

[7] World Wide Web. http://www.britannica.com/.

[8] World Wide Web. http://www.ipst.edu/amp/museum_invention_paper.htm.

[9] World Wide Web. http://www.skogssverige.se/.

[10] Björn Wilhelmsson. *An Experimental and Theoretical Study of Multi-Cylinder Paper Drying*. PhD thesis, Lund University Dept. of Chemical Engineering I, 1995.

[11] McGraw-Hill Zanichelli. *Dizionario Enciclopedico Scientifico e Tecnico*. Zanichelli, 1998.