NTNU

Norwegian University of
Science and Technology

Model Predictive Control of mixed solar and electric heating

Erik Holth

Master of Science in Engineering Cybernetics

Submission date:	August 2009
Supervisor:	Tor Arne Johansen, ITK
Co-supervisor:	John Bernhard Rekstad, UIO

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

A model of a heat system consisting of a heat storage, a solar collector and an application shall be modeled. The heat storage shall supply the application with heat and hot water. The water in the heat storage shall be heated by either a heating element or by flowing through a solar collector exposed by solar radiation. Weather dynamics such as fluctuating solar radiation and outside temperature shall be incorporated in the model. A controller shall be developed to maintain a reference temperature in the application, with more efficient use of heating element power than todays existing solutions.

Assignment given: 25. February 2009
Supervisor: Tor Arne Johansen, ITK

Contents

List of Figures

List of Tables

Chapter 1

Summary

In this report we will model a heat system consisting of a heat storage tank and an application. The heat storage tank is supplied by a heating element and heated water from a solar collector. The main objective of the heat system is to mainatian a reference temperature in the application (a house). Weather forecasts will be used as weather data affecting the heat system. We will assume that the weather forecasts and the actual weather will be the same. The heat sytem will consist of simplified nonlinear differential equations and be controlled by a model predictive controller (mpc). The mpc controller will use a linearized model of the nonlinear process. The average predicted outside temperature from the weather forecasts will be used as nominal value for the same temperature in the linearized model in the mpc controller. The mpc controller will measure some disturbances to make more efficient control. The most imortant disturbance will be the temperature of the water coming out of the solar collector, that will flow into the heat storage. By measuring this temperature, the mpc controller can apply it to its predictor and make sure that the power of the heating element in the heat storage is reduced when solar collector heated water is available. This is to make sure that the heat storage has enough capacity to receive the heated water from the solar collector, while still maintaining a reasonable temperature in the heat storage. Simulation with different weighting of the inputs in the mpc controller will show that heating element power consumption is influenced by these weights.

Chapter 2

Introduction

2.1 Motivation

Humanity is facing an energy problem. Frequently we come up with solutions that are supposed to be more energy efficient, and that improve the efficiency of technology further and further. In the past years, a focus regarding this issue has been in terms of pollution. How can we satisfy our energy needs with less pollution of the environment? In addition to this, researches have spent time searching for alternative energy sources and also alternative methods of generating the same energy sources. While some of the resulting methods are sustainable, most of them have a different negative effect on the environment. One example is the production of biofuel. Generating energy from biofuel emits less carbon dioxide, but does not necessarily mean that less carbon dioxide is emitted into the atmosphere. The reason for this is that the source for making biofuel, usually different kinds of plants, has a large environmently effect. Cutting down large areas of plants to produce biofuel has a large effect on the state of the earth, since plants play an important role in the greenhouse effect. Moreover, plants are nutrition and housing for many types of of animals and organisms.

We want to find methods that make our energy consumption more efficient. Methods that minimizes the amount of energy that is wasted into unwanted forms and locations.

Solar Power is an energy source that researchers spend time on. Some of the leading researchers in the field are situated in countries like Norway. There are also many products in the marketplace. Most of the solar collectors that heats up water works on a "take when available" basis. This means that a heat storage tank filled with water is heated up by circulating its water through a solar collector when solar radiation is present. This can supply applications with free heat. One often end up with two separate heat storage tanks. One regular supplied by electric power and one supplied by solar power. Since solar power is not always available, having an electric powered backup is usually necessary. The problem is that such solutions require a lot of installation, and are often expensive. The bigger capacity one wants on the heat storage tank of the solar collector, the the more expensive and complicated installation is. With smaller tanks one often faces the problem that the energy capacity is to small to fill up the potential amount of heated water that the solar collector can deliver. Another problem is that heat storage tanks which is only supplied by solar power has a fluctuating temperature. When temperature is too low, unfavourable growth of bacterias can occur, which precludes the use of the heated water in hot water consumptions such as showering etc. The solutions that combine solar collector with another type of energy supply to heat up one common heat storage tank also face the capacity problem. Having used electric power to fill up the capacity of the heat storage when solar radiation is available to heat up water, is not cost effective.

2.2 Existing technology

Aventa [7] is a Norwegian company that does research in this field and deliver solar collectors. They also deliver a control system called mc:floor that controls heated water that circulates through pipes in floors. These pipes can be connected to their solar collectors. The control principle is based on measuring inside and outside temperature, and to use feedback control to compensate for those temperature changes. This control unit can be equipped with a communication module that can send energy data. The German Company Vaillant Group [4] is situated in many European countries, among others in Norway. Among all their energy research technologies is the solar heat technology. They can deliver energy saving solutions in many aspects that can adapt to the energy solution your application may already have. Figure 2.1 shows two of their configurations.

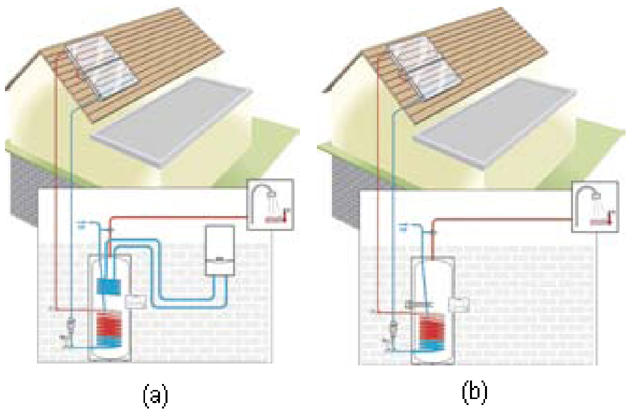

Figure 2.1: (a) Solar collector system supplied by gas, oil or pellet oven, (b) solar collector system supplied by electricity

Vaillant Group [4] can deliver all these systems with their network service system vrnetDialog, which is a diagnostic tool that gives an alert if any unusual behavior may appear. Their control system auroStep ensures that the solar collector is emptied for water when not operating, to prevent frost damage and overheating. If your application is supplied by central heating, the circuit from this and the circuit from the solar collector has to be connected. Vaillant Group delivers a hydraulic block that controls the energy supply from these circuits. A picture of such a hydraulic block is shown in figure 2.2.

Figure 2.2: Hydraulic block connecting central heating and solar heating circuits

Simpler solutions are also available, like the solar collector of the global company Apricus shown on figure 2.3

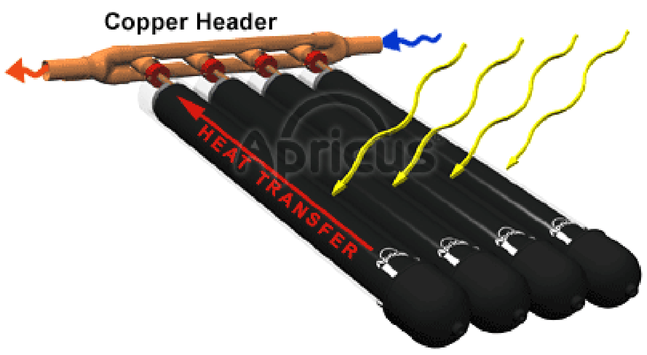

Figure 2.3: Apricus heating solution

The water in figure 2.3 is circulated through the copper header, while the heat is transfered from the evacuated heating tubes exposed by the radiation. This solution requires less installation and maintenance than the ones from Aventa [7] and Vaillant Group [4] but is less efficient.

The Luxembourg company Solelec [10] delivers a different kind of solar collector solution. Their solar collectors are installed outside dwellings, typically on the roof or nearby. Nearly all models are the direct-gain type, consisting of flat panels in which water circulates. Other types may use dish or trough mirrors to concentrate sunlight on a collector tube filled with water, brine or other heat transfer fluids. A storage vessel/container is placed indoors or out. Circulation is caused by natural convection or by a small electric pump. At night, or when insufficient sunlight is present, circulation through the panel can be stopped by closing a valve and/or stopping the circulating pump, to keep hot water in the storage tank from cooling. Depending on the local climate, freeze protection and prevention of overheating will be incorporated in their design, installation, and operation. A picture of one of their direct-gain solar heater panels with integral storage tank is shown in figure 2.4.

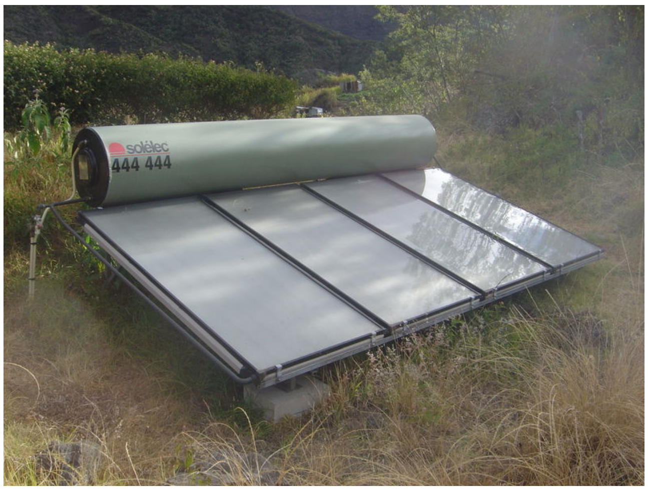

Figure 2.4: Solelec integral direct-gain solar heater

2.3 Weather Forecast

Estimating the future weather is a task that requires much calculation power. Supercomputers are used to do this and in Norway, The Institute of Meteorology is able to deliver predictions of fog, pressure, wind direction, wind speed, cloudiness, low clouds, medium clouds, high clouds, humidity and temperature in Norway. These data are updated twice a day, and gives the mentioned forecast every hour and minimum 48 hours in to the future. For every update all the data are updated based on new information available. This means that data at a given time inside the data range an hour before and an hour after update can be different. This also means that right after update 12 hours are added and at least 60 hours of future detailed data will be available. You can get forecasts further into the future but it's not detailed, which means that it's less reliable and it's not available for every hour. Based on these forecasts, parameters in a model of an energy system, for instance a home can be set.

2.4 Problem formulation

What if one could use weather predictions as a parameter in a control system in an application such as a home with a solar collector. Since most applications have limited capacity of storing energy, one regular problem is that the unit that stores energy is already full when solar energy is available. As we will see in chapter 5, we can make a model of such a system. Since these systems most likely are nonlinear, we can make a linearization at certain nominal values to make control a lot easier. This means that we will get better estimations of how the energy in the application will flow than we would get if we just used linear models that was not adaptive. This in combination with using feed forward control will make an improvement on todays existing solutions mentioned in section 2.2. Matlab with Simulink will be used through this whole demonstration on how we can use model predictive control (mpc) to improve efficiency of a heat system supplied by heated water from a solar collector. Predictions used will both be from a predictor in a mpc controller and based on measured disturbances, which will be used in a feed forward control.

Chapter 3

Solar Energy

3.1 Optic theory

The following optic theory is collected from [3]. By Snell's law, we have that:

$$n_1 sin\theta_1 = n_2 sin\theta_2 \tag{3.1}$$

where θ_1 and θ_2 are the entrance and exit angles and n_1 and n_2 are refractive indices shown in figure 3.1. P and Q in figure 3.1 are the light vectors and v_1 and v_2 are the different velocities, since light changes velocity from one medium to another.

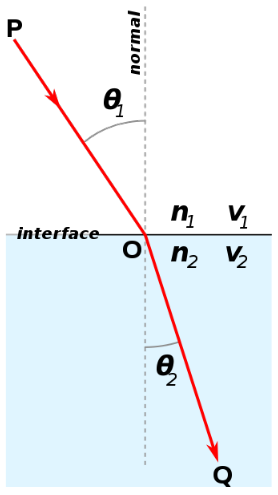

Figure 3.1: Angels and indices in Snell's law

When the solar light passes through the atmosphere, it will be partially reflected and partially transmitted, depending of the angle and polarization of the light. We call the reflection coefficient R and the transmission coefficient T. The sum of R and T equals one.

We will get different characteristics depending on how the light is polarized. Figure 3.2 shows the amplitude coefficients of reflection and transmission denoted by the vectors $r_\perp$,$t_\perp$,$r_{||}$,$t_{||}$ which is directed towards an interface.

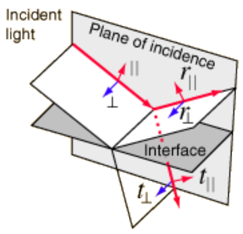

Figure 3.2: Polarized light vectors

If the light is polarized such that the electric field of the light is perpendicular to the plane of incidence in figure 3.2, we will only have a $r_\perp$ and $t_\perp$. If the light is polarized such that the electric field of the light is in the plane of incidence in figure 3.2, we will only have a $r_{||}$ and $t_{||}$. These vectors give amplitude, but we are interested in the the reflection and transmission coefficients denoted by $R_\perp$,$T_\perp$,$R_{||}$,$T_{||}$. In some formalisms they satisfy:

$$R = r^2 \tag{3.2}$$

$$T = \left(\frac{n_2 \cos\theta_2}{n_1 \cos\theta_1}\right) t^2 \tag{3.3}$$

Augustin-Jean Fresnel has deduced the following equation for reflection and transmission coefficients of light

$$R_\perp = \left[\frac{sin(\theta_2 - \theta_1)}{sin(\theta_2 + \theta_1)}\right]^2 \tag{3.4}$$

$$R_{||} = \left[\frac{tan(\theta_2 - \theta_1)}{tan(\theta_2 + \theta_1)}\right]^2 \tag{3.5}$$

where θ_1 and θ_2 are the same as in figure 3.1.

Using Snell's law given by (3.1) on (3.4) and (3.5) we get:

$$R_{\perp} = \left[\frac{n_1 cos\theta_1 - n_2\sqrt{1-(\frac{n_1}{n_2}sin\theta_1)^2}}{n_1 cos\theta_1 + n_2\sqrt{1-(\frac{n_1}{n_2}sin\theta_1)^2}}\right]^2 \tag{3.6}$$

$$R_{||} = \left[\frac{n_1\sqrt{1-(\frac{n_1}{n_2}sin\theta_1)^2} - n_2 cos\theta_1}{n_1\sqrt{1-(\frac{n_1}{n_2}sin\theta_1)^2} + n_2 cos\theta_1}\right]^2 \tag{3.7}$$

Since the solar light is unpolarized, i.e an equal mix of the two coefficients, the reflection coefficient is given by [3]:

$$R = \frac{R_{\perp} + R_{||}}{2} \tag{3.8}$$

Since R is the amount of light reflected, the transmitted amount T is:

$$T = 1 - R \tag{3.9}$$

We can see from (3.9) that the sum of reflected and transmitted light equals 1. Since neither T nor R can be negative, this means that:

$$1 \geq R \geq 0 \tag{3.10}$$

$$1 \geq T \geq 0 \tag{3.11}$$

3.2 Fixed surface radiation

We have to be able to calculate how much radiation effect that radiates towards a solar collector, which in our case is a fixed plane surface mounted on the roof of the application (a house). To be able to do this calculation, equations gathered from [8] has been used. Figure 3.3 shows directed solar radiation towards a plane, which in our case is the solar collector.

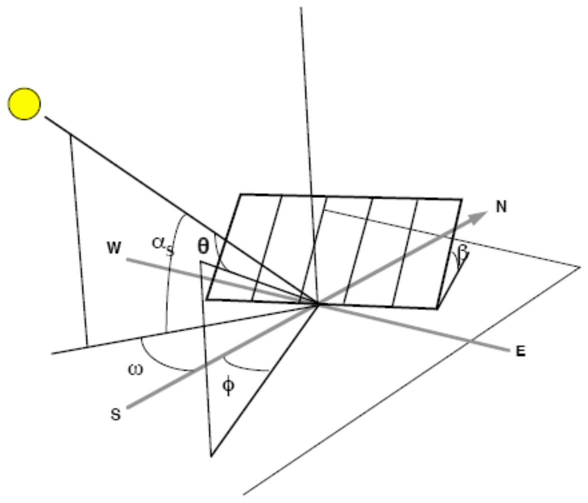

Figure 3.3: Solar radiation towards a fixed surface

Figure 3.3 shows the majority of the angles needed for this calculation. The axis on the figure is oriented towards true north (N on figure 3.3) The only angle we are interested in is θ, which is angle between the solar beam and the perpendicular vector of the fixed plane. This angle depends on ϕ, ω and β on figure 3.3. The angle ϕ is the azimuth angle, that is the angle between the orientation of the fixed plane and south (S). The angle ω is the time angle, that is the angle between the horizontal projection of the sun beam and south (S). We will have $\omega = 0$ at the longitude that passes through London at 12 o'clock GMT(Greenwich Mean Time). Then the longitude degree of any location on earth can be used to determine the angle ω at that particular location. The earth is divided into longitudes with 15° between them and it's exactly one hour (sun time) time difference between each. This means that ω will increase 15° per hour. The angle β is simply the tilt angle of the fixed plane. Very often the solar collector is mounted on the roof, fixed to the roof surface as shown in figure 2.1. In that case the tilt of the roof and of the solar collector β will be the same.

Since the earth orbits around itself and the sun, these angles will change with respect to time. The angle θ will also be dependent on the declination angle δ. This angle will vary through the year because the earth is tilted in terms on its travel path around the sun. This is shown on figure 3.4.

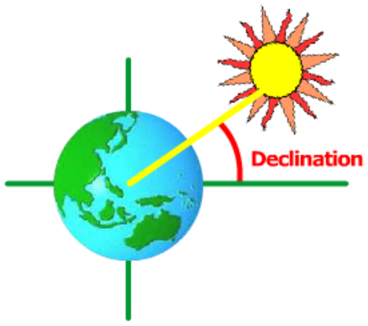

Figure 3.4: Declination angle

Because of this tilt, the declination angle δ will vary depending on where the earth is located on its path around the sun. The equation of δ is given by [8]:

$$\delta = 23.45^\circ sin\left[360^\circ \frac{(284+n)}{365}\right] \tag{3.12}$$

where n is the daynumber of the year. Leap years are neglected.

The latitude angle φ is fixed in time at a fixed location. The latitude is 0 at equator, -90° at the south pole and 90° at the north pole.

The total radiation flux towards the surface is $I_0 Acos(\theta)$ where I_0 is the directed radiation, A is the surface area and the last term is cosine of the described angle θ. The term $cos(\theta)$ is deducted in [8] to be:

$$\begin{aligned} cos(\theta) &= sin(\delta)sin(\varphi)cos(\beta) - sin(\delta)cos(\varphi)sin(\beta)cos(\phi) \\ &+ cos(\delta)cos(\varphi)cos(\beta)cos(\omega) + cos(\delta)sin(\varphi)sin(\beta)cos(\phi)cos(\omega) \\ &+ cos(\delta)sin(\beta)sin(\phi)sin(\omega) \end{aligned} \tag{3.13}$$

where the different angels in 3.13 were explained above and in figure 3.3

We will get a characterization on the radiation shown on figure 3.5.

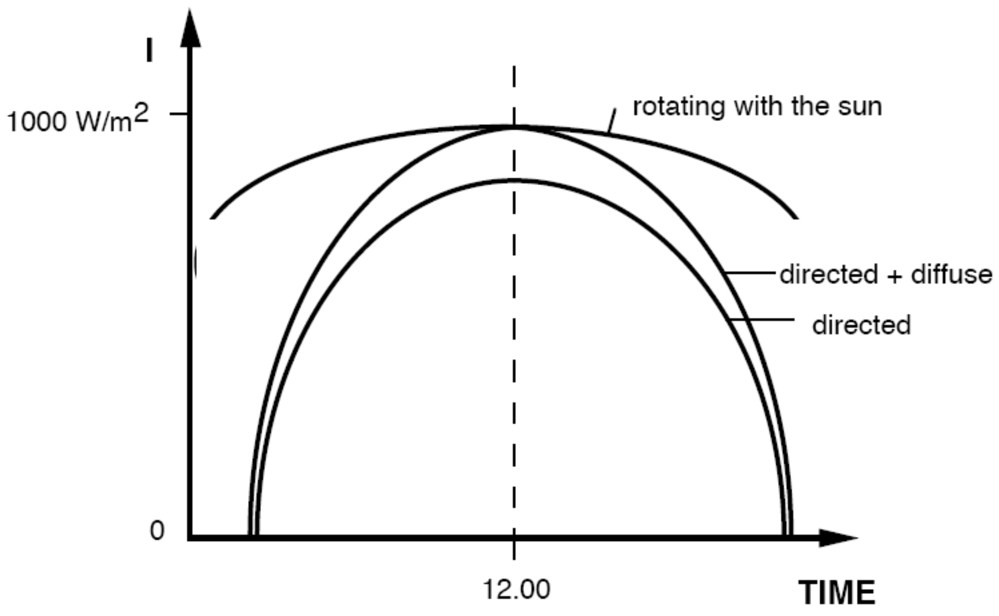

Figure 3.5: Global daily radiation

Figure 3.5 also shows the added effect if the solar collector followed the sun to set the term $cos(\theta)$ equal to one at all time. The radiation from the sun still has to travel through a thicker layer of atmosphere when the sun is not right above the surface. This is the reason why the radiation still decays before and after 12.00 on figure 3.5. Figure 3.5 also shows the effect of diffuse radiation, which we will neglect in this case.

3.3 Approximation of radiation

We want to approximate the radiation weakening through the atmosphere. The solar radiation hits the atmosphere with an angle called θ_1. θ_1 is the same angle as in figure 3.1, assuming the interface is the atmosphere. We will assume θ_1 to be the same as equation (3.13), but with with tilt angle β set to zero. It's reasonable to assume that the atmosphere has no tilting. The equation for θ_1 then simplifies to be:

$$cos(\theta_1) = sin(\delta)sin(\varphi)cos(\beta) + cos(\delta)cos(\varphi)cos(\beta)cos(\omega) \tag{3.14}$$

In reality, n_1 and n_2 in 3.1 are difficult to estimate because the atmosphere have many different layers. Aventa [7] have deducted an approximative formula for the radiation transmitted fraction T according to the theory mentioned above and measured data. The formula is:

$$T = 1 - \underbrace{\frac{a}{\sqrt{cos(\theta_1)}}}_{\text{reflection}} - \underbrace{\left(1 - \frac{a}{\sqrt{cos(\theta_1)}}\right)\left(\frac{b}{cos(\theta_1) + c}\right)}_{\text{absorption}} \tag{3.15}$$

The reflection term of (3.15) represents the reflection of the radiation, which increases when θ_1 in (3.14) increases. The absorption term in (3.15) represents the absorption, which only acts on the non reflected part of the radiation. When θ_1 increases, the radiation will also have a longer distance to travel, which increases the absorption in the atmosphere. Refraction in the atmosphere is small and will be neglected in this case. The coefficients a, b and c in (3.15) are approximated by measurements by Aventa [7] in the Norwegian city Oslo. We will use these constants in the case study. The coefficients a, b and c are shown in table 3.1.

constant	value
a	0.15
b	0.14
c	0.1

Table 3.1: Approximated coefficients in 3.15 for Norway

Clouds on the sky play an important role on how much radiation that reach the surface of the earth. This means that we can not neglect that factor. Measurements done by Aventa [7] says that on a typical summer day in Norway, about 960 $\frac{W}{m^2}$ of maximum radiation flux passes through the atmosphere and reaches a directed surface, assuming no clouds on the sky. If the sky is 100 % covered by clouds, measurements say that about 20 $\frac{W}{m^2}$

reaches the same surface. Based on these measurements, a linear subtraction will be used to incorporate the radiation that reaches the surface, that is 9.4 $\frac{W}{m^2 * \%_{cloudiness}}$. $\%_{cloudiness}$ is the cloudiness percentage that will be available from weather forecasts, which we will come back to in chapter 4. The solar constant I_{sol} is the directed irradiation flux received by a surface from the sun outside the atmosphere. According to Aventa [7] $I_{sol} = 1360 \ \frac{W}{m^2}$. We can then calculate the resulting Irradiation flux $\frac{I}{A}$ where A is the surface area. The equation is approximated to be:

$$\frac{I}{A} = \underbrace{(I_{sol} * T - (9.4 * \%_{cloudiness}))}_{I_0} cos(\theta) \tag{3.16}$$

where I_0 in (3.16) is the directed radiation flux mentioned in section 3.2, T is from equation (3.15), $cos(\theta)$ is from equation (3.13) and $\%_{cloudiness}$ is the cloudiness percentage mentioned. In reality, the exact radiation that passes through the atmosphere and clouds will be much more complicated to calculate. The $\%_{cloudiness}$ that is gathered from weather forecasts discussed in chapter 4, gives a percentage of how big part of the sky is covered by clouds. The thickness and location of the clouds will have a big impact on the amount of solar radiation that hits a surface. In many situations, clouds can increase the irradiation flux. This can happen when the clouds don't block the directed radiation beams towards the fixed surface, but reflects beams with other directions. One will then get a positive contribution by the clouds and not negative as in equation (3.16). The apparent positive contribution by the clouds is the biggest among the denoted diffuse radiations shown in figure 3.5. Reflection from surrounding topography also contributes to diffuse radiation.

3.4 Solar Collector

The operational equations of the solar collector based on average temperature are gathered from [8]. The equations are:

$$
\begin{aligned}
P_G &= c_w \frac{dm}{dt} A (T_{out} - T_{in}) && (3.17) \\
T_w &= \frac{T_{in} + T_{out}}{2} && (3.18) \\
\eta &= \frac{P_G}{IA} && (3.19) \\
\eta &= \eta_0 - \frac{K_1}{I}(T_w - T_0) - \frac{K_2}{I}(T_w - T_0)^2 && (3.20)
\end{aligned}
$$

where the different letters are as explained in table 3.2.

letter	explenation	unit
P_G	power gain on flowing water	W
c_w	heat capacity of water	$\frac{J}{kg°C}$
$\frac{dm}{dt}$	flowrate of water per area	$\frac{kg}{m^2s}$
A	area of collector	m^2
T_{in}	water temperature in to collector	$°C$
T_{out}	water temperature out of collector (later called v_1)	$°C$
T_w	average water temperature in collector	$°C$
T_0	outside temperature (later called v_4)	$°C$
η	collector efficiency due to Irradiation	unitless
η_0, K_1, K_2	collector efficiency parameters	unitless
I	Irradiation to collector	$\frac{W}{m^2}$

Table 3.2: Parameters of solar collector

We call I irradiation since the solar collector receives radiation from the sun. Equation (3.17)-(3.20) are based on average water temperature, and don't give the temperature of the water in terms of differential equations or as a function of the location in the solar collector. The simplification will be used when modeling the solar collector, but it has its limitations as we have not deducted any differential equations for it. Optimizing the amount of energy supplied to the water that runs through the collector at a particular irradiation flux and outside temperature will not be possible with the simplified modeling method used in equations (3.17)-(3.20).

Inserting (3.17) into (3.19) and (3.18) into (3.20) gives:

$$\eta = \frac{c_w \frac{dm}{dt}(T_{out} - T_{in})}{I} \quad (3.21)$$

$$\eta = \eta_0 - \frac{K_1}{2I}(T_{out} + T_{in} - 2T_0) - \frac{K_2}{4I}(T_{out} + T_{in} - 2T_0)^2 \quad (3.22)$$

Since we are interested in finding T_{out}, we insert (3.21) into (3.22) and solve in terms of T_{out} and we get the second order equation:

$$BT_{out}^2 + CT_{out} + D = 0 \quad (3.23)$$

where:

$$B = \frac{1}{4}K_2 \quad (3.24)$$

$$C = c_w\frac{dm}{dt} + \frac{1}{2}K_1 + \frac{1}{2}K_2T_{in} - K_2T_0 \quad (3.25)$$

$$D = (-c_w\frac{dm}{dt} + \frac{1}{2}K_1 + \frac{1}{4}K_2T_{in} - K_2T_0)T_{in} + (K_2T_0 - K_1)T_0 - I\eta_0 \quad (3.26)$$

Because of the physical limits of the solar collector, η in (3.19) and (3.20) is limited to:

$$\eta_0 \geq \eta \geq 0 \quad (3.27)$$

We can see from equation (3.20), that the variable η depends on the solar collector parameters η_0, K_1 and K_2. These parameters varies dependent on the kind of solar collector chosen. You can get different types of collectors with different quantity of cover glass, which affects these parameters. Typical parameters for η_0 is shown in table 3.3.

cover glass	η_0
without	0.95
single sheet	0.87 - 0.90
double sheet	0.75

Table 3.3: Typical parameter range for η_0

Typical range of parameter K_1 and K_2 is shown in table 3.4.

Equation (3.19) and (3.20) shows that it's desirable to have η as high as possible, which means that we want the parameter η_0 as high as possible

parameter	range	unit
K_1	2 - 3	$\frac{W}{m^2 {}^\circ C}$
K_2	0.02 - 0.04	$\frac{W}{m^2 {}^\circ C^2}$

Table 3.4: Typical parameter range for K_1 and K_2

and the parameters K_1 and K_2 as low as possible. Unfortunately one will face a trade off situation. Less cover sheets on the solar collector results in a higher η_0, which can be seen on table 3.3. With fewer cover glasses, the parameters K_1 and K_2 will also be higher. The reason for this is that more of the energy that is radiated on the solar collector will be emitted back to its surroundings due to less isolation. More cover glasses results in a lower η_0, but also a reduction in K_1 and K_2. What choice one should take of these trade off parameters, depends on what the solar collector shall be used for. If one wish to operate with small temperature difference between desired water temperature in the solar collector and its surroundings, a high η_0 is desirable (few or no cover glass). Then those temperatures will be reached fast. If one wish higher temperatures in the water in the solar collector, one should consider using a collector with one or several cover glasses, since more cover glass results in less loss to the surroundings.

The parameters of the chosen solar collector in this task is received by Aventa [7]. The parameters of this type of collector is given in table 3.5.

parameter	value	unit
η_0	0.8	unitless
K_1	3	$\frac{W}{m^2 {}^\circ C}$
K_2	0.03	$\frac{W}{m^2 {}^\circ C^2}$

Table 3.5: Parameter values for Aventa solar collector

A picture of the solar collector with the parameters in table 3.5 is shown in figure 3.6.

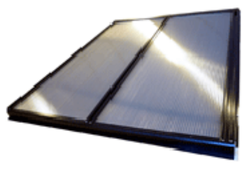

Figure 3.6: Solar Collector from Aventa

Inserting the parameters given in table 3.5 and taking the inequality mentioned in (3.27) into account, we get the feasible region with graph for η shown in figure 3.7.

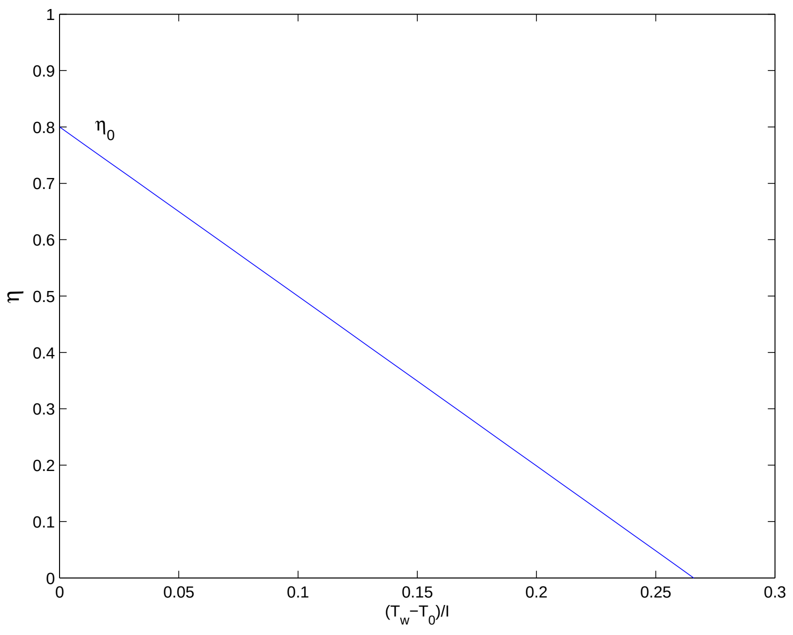

Figure 3.7: Solar Collector Efficiency

3.5 Implementation

Radiation flux is implemented in getRadiationPerM2.m in the appendix A.3. The radiation flux require some weather data (weather forecasts), which was shown in equation (3.16). This will be explained in chapter 4. The radiation flux and the irradiation flux will of course be the same, since the radiation minus loss sent by the sun to a particular surface will be the same as the irradiation received by the same surface. So when radiation and irradiation are mentioned, it will be the same, but seen from either the sun or from the solar collector. The radiation flux is implemented such that it can't be negative, that is set to zero whenever equation (3.16) gives a negative number.

The implementation of the solar collector is coded in getSolarTempOut.m in appendix A.4. The code implements the valves many solar collectors have. These valves makes sure that $T_{out} \geq T_{in}$ and that $T_{out} \leq 100°C$ to avoid boiling. If all the water carrying components in the system operates at a pressure over 1 atm (standard), boiling would have been prevented at 100 $°C$, but it is neglected in this case. The code is implemented in an embedded matlab function block, which looks like on figure 3.8.

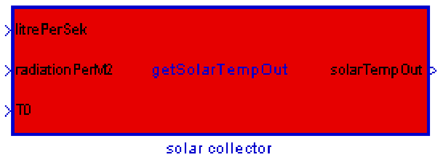

Figure 3.8: Solar collector functionality in embedded Matlab block

Chapter 4

Weather forecast

4.1 Matlab and XML

The weather forecasts of the Norwegian Institute of Metrology (MET) was mentioned in the introduction. To be able to gather weather forecasts from MET, xml must be used. All the data is published through the application programming interface (API) called locationforecast in MET's weatherapi [6]. All the forecasts available are published in xml format and the publishing method was mentioned in section 2.3. As can be seen in the API, one inputs the graphical coordinates i.e. latitude and longitude. An optional choice of height above sea level can be included if wanted. This option will be set to zero if not specified.

To be able to import the xml file to Matlab for navigation and extracting of data, Matlab's Document Object Model (DOM) has to be used. The function xmlread in Matlab reads a URL or filename and returns a Document Object Model node representing the parsed document [5]. One can then navigate through the node and extract all the necessary information one wants to use. The collection of data is done in the Matlab file main.m in appendix A.1 and the Matlab function getWeatherData.m shown in appendix A.2. The collection method used is adaptive, and it collects all the hours available of detailed weather forecasts.

4.2 Testing scenario

Temperature and cloudiness starting at 12 o'clock (12.00) July 1 2009 in Longyearbyen in Svalbard was chosen as testing scenario. This location has been choosen since the outside temperature is low and solar radiation is present. Then heating will be required at all time, and contribution from the solar collector will be a part of the heating of our later modeled system. We will set our time t = 0 at the first data point, that is at 12 o'clock. The coordinates of the location where found using Google Earth, and the coordinates are shown in table 4.1.

coordinates	value	unit
latitude	78.11	degrees
longitude	15.34	degrees
altitude above sea	40	meters

Table 4.1: Geographical Coordinates of testing site

Temperature and radiation with cloudiness are the weather forecasts that will be used as actual weather surroundings in our model. One could choose more weather forecast types than those two to make the model more realistic, but it would also make the model more complicated to model. The coordinates and time of the location will be used to calculate the radiation at any given time using equation (3.14) and (3.16). When the weather forecast was collected from the testing scenario, it was available with 54 data points. We will use the temperature data in the forecast to find nominal value of the outside temperature when simulating a heat system with and without control. The temperature forecast will also be used as a disturbance in form of actual outside temperature in the system we will come back to in chapter 5. Cloudiness will be used to reduce the radiation as shown in equation (3.16) at the same heat system. In figure 4.1 temperature and cloudiness are shown at our testing scenario. Because of our number of data points, the duration of the data will be from t = 0 to t = 53, where t is in hours. The duration of our simulations will therefore be limited to t = 53, when we apply these surroundings to our simulations. When we apply these data ass surroundings to our heat system, Lagrangian interpolation will be used between the hourly data points, since our simulation will be done in seconds.

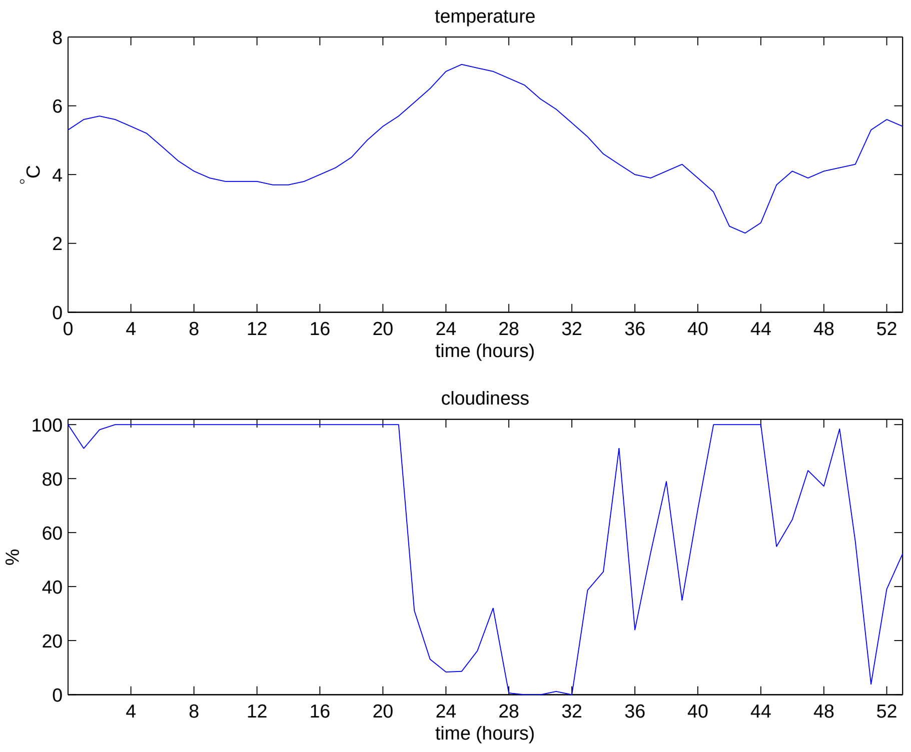

Figure 4.1: Temperature and cloudiness at our testing scenario starting at 12 o'clock

Our solar collector will be oriented with tilt angle $\beta = 45°$ directed towards the east ($\phi = 90°$) where β and ϕ are the angles from equation 3.13 in chapter 3. It is usually more desirable to to direct the solar collector towards south ($\phi = 0°$), but the timing of the cloudiness resulted in more radiation hitting the fixed surface with the chosen direction in our testing scenario.

The radiation flux at this location is shown in figure 4.2 if we neglect the cloudiness. Radiation on the same collector with cloudiness incorporated, deducted from equation (3.16) in chapter 3, is shown in figure 4.3.

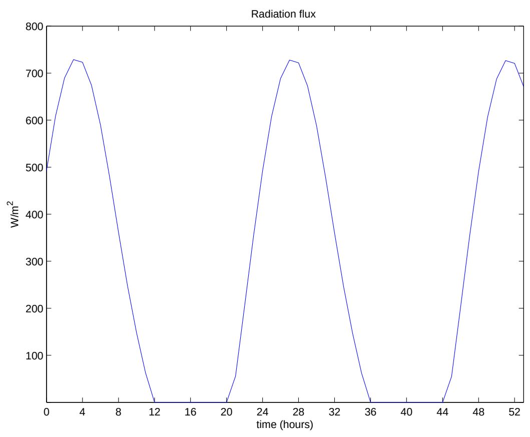

Figure 4.2: Radiation on the fixed surface with neglected cloudiness in Longyearbyen starting at 12 o'clock

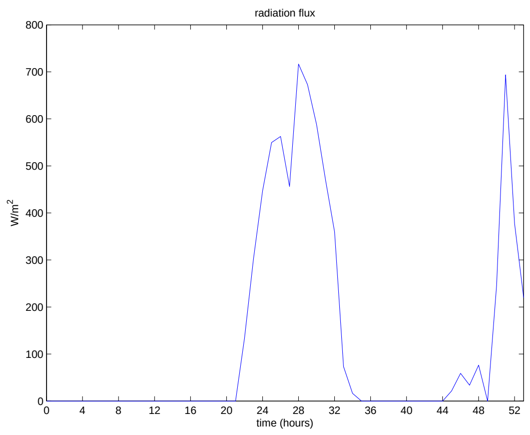

Figure 4.3: Radiation on the fixed surface with cloudiness in Longyearbyen starting at 12 o'clock

The radiation with cloudiness and the temperature at this testing scenario will be used further, when we are going to expose a simplified system to influence by weather surroundings.

Chapter 5

Heat system

5.1 Modeling

To demonstrate the control principle, we are going to model the dynamics of a simplified heat system with a solar collector. The system will consist of an application (for example a house), which will be heated by energy stored in a heat storage tank. Figure 5.1 shows the heat system.

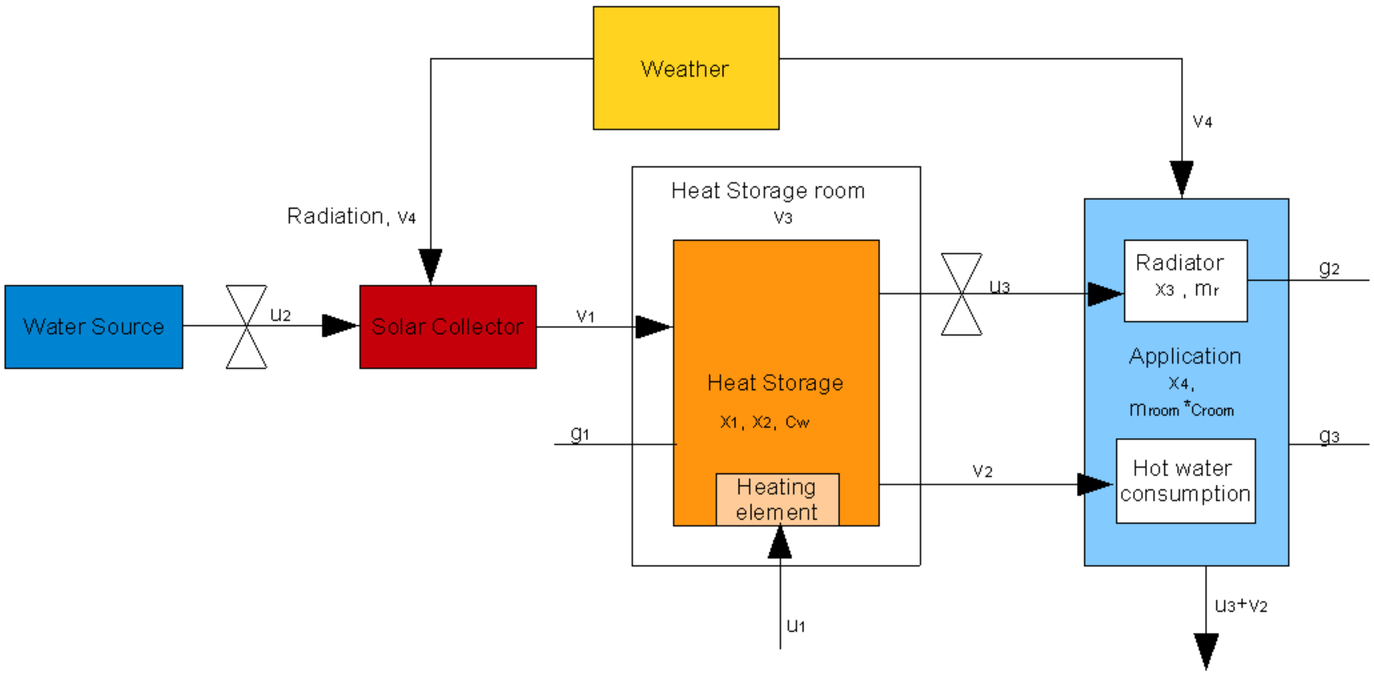

Figure 5.1: Heat system

The application is heated by water running through a radiator which is placed in the application in the heat system. The heat system will be simplified in terms that we use approximation in our modeling of the energy flow. The system will be nonlinear, but will be linearized around nominal values when we apply control to it in chapter 8.

We can see from figure 5.1 that we have a water source where we assume unlimited water supply. This is general water supply in to the system from the water supplier. We will assume that this water has a constant temperature at $5°C$, which will be the lower bound of the temperature v_1 in to heat storage tank. The water then goes through the solar collector with flowrate u_2. The solar collector will either heat up the water, or maintain the same temperature depending on the presence of solar radiation. If the solar collector shall be able to heat up the water, it must be exposed to radiation. The flowrate in and out of the solar collector is the same since the solar collector has no excess storage. We assume that the solar collector is full of water at all time for simplification. This flowrate together with the radiation and the outside temperature v_4 decides how much the solar collector heats up the water flowing through it. The water then goes out of the solar collector with temperature v_1 and flowrate u_2 and into the heat storage. The heat storage is a tank consisting of water. The heat storage is placed inside an isolated room with constant temperature v_3. The heat storage has a heating element which can be supplied by electric power u_1. This power can help heating up the water when the water temperature out of the solar collector can't reach up to the heat storage's desired temperature. The heat storage has temperature x_1 and we assume that the temperature is the same everywhere in the heat storage. (If needed this can be achieved with stirring, but is a reasonable simplification). The heat storage has mass x_2 with specific heat capacity c_w. The coefficient of surface conductance between heat storage and heat storage room is g_1. The water can then go from the heat storage through either the radiator or to generic flow (general hot water tapping). The radiator has temperature x_3, contains water with constant mass m_r and specific heat capacity c_w. The coefficient of surface conductance between radiator and application is g_2. The heat storage tank can also be tapped by generic flowrate v_2. When the water passes through the radiator, it will heat up the application, which has a temperature x_4, a mass and specific heat capacity product $m_{room} * c_{room}$ and a coefficient of surface conductance between application and surroundings g_3. After the water has passed either the radiator or to the generic consumption, it ends up in the drain.

5.2 Deduction of equations

It's known from [1] that the accumulation of energy E per time is:

$$\frac{dE}{dt} = mc\frac{dx}{dt} \tag{5.1}$$

for a given mass m with specific heat capacity c and temperature x.

A simplified cooling law will be used in our system, which says that the power removed from a given mass is proportional to the temperature difference between the mass itself and its surroundings.

The overall system consists of states denoted by x, inputs denoted by u and disturbances denoted by v, while the rest is considered constants. Basically we have five contributions to the energy stored in the heat storage. These are power to heating element u_1, water flow in with a given flowrate u_2 and temperature v_1, water flow out with a given flowrate u_3 and temperature x_1, generic flowrate in terms of hot water tapping with flowrate v_2 and temperature x_1 and power loss due to temperature difference of heat storage and surroundings. We assume by simplification that there is no heat capacity in the heating element, which means that all the power supplied to the heating element goes directly into the water in the heat storage with no delay. Since the heat capacity in the heating element is much faster than in the water in the heat storage, this simplification is reasonable. The water flow out of heat storage is divided in two parts. The first is flow to radiator in application with flowrate u_3 and temperature x_1. The other is the generic flowrate v_2. The differential equations of the last two states, radiator temperature x_3 and application temperature x_4 consists of fewer parts, since the simplified model has fewer contributions. The radiator has constant water mass m_r, which means that it's full of water at all time. Otherwise equation (5.4) and equation (5.5) are deducted the same way as equation (5.2), that is by adding all the contributions and the use of (5.1). The coefficient of surface conductance g_1, g_2 and g_3 will be proportional coefficient used with the simplified cooling law. These coefficients would in reality not be constant, but be variables depending on many factors. The assumed constant g_1 will for instance in reality be dependent on the temperature and mass in the heat storage tank. The coefficient of surface conductance g_2 will in reality depend on the radiator temperature x_3. The constant g_3 would be dependent on many factors, like the application temperature x_4, the outside temperature v_4, complicated weather conditions (wind, humidity etc.) and topographical influences on the application. The heat capacity product $m_{room} * c_{room}$ would in reality depend on the actual temperature x_4 in the application, and many other factors. The temperature v_3 in the room of the heat storage is assumed to be constant, but would in reality depend on temperature of its surroundings and the temperature in the heat storage itself. We have also assumed that the heat storage room is separated from the application, such that the heat storage only affects the application temperature x_4 through the radiator and not through the heat loss of the heat storage. The four differential equations we end up with is listed in (5.2)-(5.5), while table 5.1 lists up all the variables and parameters in those equations. Table 5.2 lists all the units for the states, inputs and disturbances, where $^{\circ}C$ is degree Celsius, kg is kilograms, $\frac{l}{s}$ is liters per second and kW is kilowatt. We assume 1 $\frac{l}{s} = 1\ \frac{kg}{s}$, due to the density of water. This means that the system equations will be:

$$\frac{dx_1}{dt} = \frac{1}{c_w x_2}(1000u_1 + c_w u_2 v_1 - c_w x_1 u_3 - c_w x_1 v_2 - g_1(x_1 - v_3)) \tag{5.2}$$

$$\frac{dx_2}{dt} = u_2 - u_3 - v_2 \tag{5.3}$$

$$\frac{dx_3}{dt} = \frac{1}{m_r c_w}((c_w(x_1 - x_3)u_3 - g_2(x_3 - x_4)) \tag{5.4}$$

$$\frac{dx_4}{dt} = \frac{1}{m_{room} c_{room}}(g_2(x_3 - x_4) - g_3(x_4 - v_4)) \tag{5.5}$$

The factor 1000 in (5.2) is due to u_1 being given in kW in order to achieve favorable scaling of equation for numerical reasons, as explained in section 8.2.

Equation (5.2) is sensitive to a low heat storage mass x_2. According to the equation, the heat storage temperature $x_1 \to \pm\infty$ when $x_2 \to 0$ if we assume that the expression $(1000u_1 + c_w u_2 v_1 - c_w x_1 u_3 - c_w x_1 v_2 - g_1(x_1 - v_3)) \neq 0$. The temperature would in reality never go to infinity with zero mass, but damage would probably occur if heating element power is applied to an empty heat storage tank.

variables	explanation
states:	
x_1	heat storage temperature
x_2	heat storage mass
x_3	radiator temperature
x_4	application temperature
inputs:	
u_1	heating element power
u_2	flowrate through solar collector and into heat storage
u_3	flowrate out of heat storage and into radiator
disturbances:	
v_1	temperature in to heat storage tank (out of solar collector)
v_2	generic flowrate due to hot water tapping
v_3	constant surrounding temperature in heat storage room
v_4	outside temperature

parameters	explanation
m_r	mass of water in radiator
c_w	specific heat capacity of water
$m_{room}c_{room}$	energy per temperature multiplicity in application
g_1	coefficient of surface conductance between heat storage and heat storage room
g_2	coefficient of surface conductance between radiator and application
g_3	coefficient of surface conductance between application and surroundings (outside)

Table 5.1: System variables and constants

variable	unit
x_1	$^\circ C$
x_2	kg
x_3	$^\circ C$
x_4	$^\circ C$
u_1	kW
u_2	$\frac{l}{s}$
u_3	$\frac{l}{s}$
v_1	°C
v_2	$\frac{l}{s}$
v_3	°C
v_4	°C

Table 5.2: System variable units

5.3 Deduction of parameter values

To deduct our constants in our application, we have to decide its dimensions. The application case used will be consisting of a $100m^2$ house with standard roof height and an average proportion of walls and windows. The solar collector has an area of 30 m^2 and will be placed on the roof. The solar collector will be oriented towards east ($\phi = 90°$) with tilt angle $\beta = 45°$. The angles ϕ and β were explained in chapter 3 and shown in figure 3.3. This is the same orientation as we had collected weather forecasts from in chapter 4. We will start by deducting the power requirement for this example. Measurements done by Aventa [7] shows that a simplified house requires $45\frac{W}{m^2}$ to maintain a normal indoor temperature with an outside temperature at $-20°C$. A linear simplification of power required to maintain a given inside reference temperature T_{ref} is shown in figure 5.2.

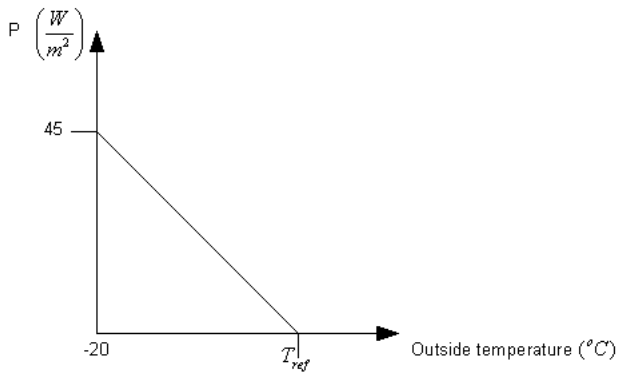

Figure 5.2: Needed power

By measurements [7] it's also known that regular human activity in such a home contribute with approximately $5°C$. We assume this contribution in our modeling of the system. This means that whatever temperature the heating system gives to our application, we will get an extra temperature contribution of $5°C$ for free due to this human activity. This contribution will be implemented as a bias in the application temperature model. This means that the actual room temperature x_4 will be:

$$T_{room} = T_{ref} + T_{free} \tag{5.6}$$

If we want a reference temperature of x_4 at $20°C$, T_{ref} in figure 5.2 will be $15°C$. The coefficient of surface conductance g_3 in our example will then be the gradient of the linear line shown in figure 5.2. Since our application is a 100 m^2 house we get:

$$g_3 = -\frac{dP}{dT} = \frac{4500W}{T_{ref} + 20°C} \approx 130\frac{W}{°C} \tag{5.7}$$

The negative sign before $\frac{dP}{dT}$ is because we need g_3 to be positive since it's a proportional constant of loss and we subtract it in (5.5).

The coefficient of surface conductance g_1 will vary according to the water level or mass in the heat tank. Our simplification sets g_1 to a constant of $100\frac{W}{m^2}$. The choice of value is verified by Aventa [7].

To deduct the coefficient of surface conductance for the radiator g_2 a little trial and error has been done. We want this coefficient of surface conductance to be sufficiently larger than g_1, but small enough to give a reasonable time constant to differential equation (5.5). This has therefore been set to $500\frac{W}{m^2}$. Such a radiator gives in average $12\frac{W}{m^2 °C}$ [9]. By examine the dimensions of such a radiator, it will contain approximately 50 kg of water, with a total area of $40m^2$. The mass and heat capacity of a house is rather complicated to calculate. This is because of the walls and isolation, which carries most of this factor. The mass and heat capacity of the air in the house is neglectable. The product $m_{room}c_{room}$ has therefore been deducted from measurements by Aventa [7]. They have given a mean of measurements to $5\frac{kWh}{°C}$ in a $100m^2$ house where h is hour, which is the same as $1.8 * 10^7\frac{J}{°C}$. Parameter values are listed in table 5.3.

parameter	value	unit
m_r	50	kg
c_w	4184	$\frac{J}{kg°C}$
$m_{room}c_{room}$	$1.8 * 10^7$	$\frac{J}{°C}$
g_1	100	$\frac{W}{°C}$
g_2	500	$\frac{W}{°C}$
g_3	130	$\frac{W}{°C}$
v_3	20	°C

Table 5.3: System parameter values

5.4 Control specification

The capacity of the heat storage tank will be 1000 kg, which means that the heat storage mass x_2 can't surpass that. We will have an upper limit on heating element power u_1 set to 10 kW. The flowrates u_2 and u_3 will be upper limited to 0.1 $\frac{l}{s}$. This also applies to the generic flowrate v_2. We also want to limit the temperature in the heat tank to $100°C$ to prevent the water from boiling. The lower limit of heat temperature x_1 is set to the temperature of the incoming water from the water supplier. We assume this temperature is $5°C$. For the heating system to have full control over the application temperature x_4, outside temperature must be lower than inside reference temperature. If not cooling must take place, which our system can not do. In this case the system will be turned off. This means that we don't allow the flowrates u_2 and u_3 to be negative. Neither the radiator temperature x_3 nor the application temperature x_4 can exceed the heat storage temperature x_1 under operation, assuming a reasonable outside temperature. This means that the limits shown in table 5.4 are operational limits, not physical limits.

variable	lower limit	upper limit	unit
x_1	5	100	$°C$
x_2	0	1000	kg
x_3	5	100	$°C$
x_4	v_4	100	$°C$
u_1	0	10	kW
u_2	0	0.1	$\frac{l}{s}$
u_3	0	0.1	$\frac{l}{s}$
v_2	0	0.1	$\frac{l}{s}$

Table 5.4: Variable limits under operation

5.5 Simulating uncontrolled system

We will first start by simulating the heating of the uncontrolled system (5.2)-(5.5) to verify that the deducted parameters are reasonable. An overview of the system in Matlab is shown in B.1. All our simulations will be done in Simulink.

We will start by simulating the heating of the water in the heat storage from 5 to 70°C at maximum heating element power u_1 at 10 kW. This is to illustrate the dynamics of the heat storage. The heater storage mass x_2 will be at its maximum of 1000 kg at all time, and the flowrate of incoming water u_2 and outgoing water u_3 will be set to zero. This means that no energy will come from the solar collector. The generic flowrate v_2 will also be set to zero. The heat storage room temperature v_3 will be set to 20°C as mentioned in table 5.3. We get the response shown in figure 5.3.

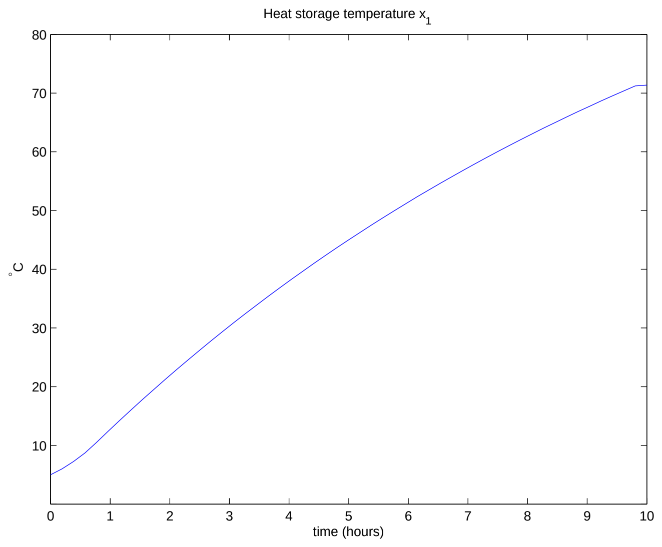

Figure 5.3: Heat storage temperature x_4 with energy from heating element u_1 only

Figure 5.3 shows that it almost takes 10 hours before all the water in the 1000 kg tank is heated up to 70°C. If we integrate equation (5.1) on both sides we get that:

$$E = mc\Delta x \tag{5.8}$$

where E is energy, m is mass, c is heat capacity and Δx is the temperature difference. Inserting m = 1000 kg, c = c_w mentioned in table 5.3 and Δx = 70 - 5 = 65°C, (5.8) gives the amount of energy needed to make this increase in temperature:

$$E = mc\Delta x = 1000kg * 4184\frac{J}{kg^{\circ}C} * 65^{\circ}C \approx 2.72 * 10^{8}J \tag{5.9}$$

With 10 kW of input power which is 10000 $\frac{J}{s}$, we find the time required to heat up the water is:

$$10000\frac{J}{s} * t = 2.72 * 10^{8}J \Rightarrow t \approx 27196s \approx 7.55hours \tag{5.10}$$

where t is time and s is seconds in (5.10). The reason that our system takes longer time to reach $70^{\circ}C$ than the solution of equation (5.10), is that the heat system incorporates energy loss to the heat storage room. The solution of equation (5.10) assumes no loss to its surroundings, but is calculated to show that the deducted system parameters and dynamics are reasonable.

Now we will simulate the heating of the application temperature x_4 to $20^{\circ}C$ in two scenarios.

Scenario 1:

We want to illustrate the dynamics of the application temperature x_4. The heat storage temperature operates at a constant temperature x_1 at 70 $^{\circ}C$ and with maximum heat storage mass x_2. We assume for simplicity that the solar collector heats up all the passing water to the heat storage's steady state temperature at $70^{\circ}C$. This means that the only heating element power we need to use is to compensate for the heat storage loss to its surroundings. We set the flowrates u_2 and u_3 at its maximum at 0.1 $\frac{l}{s}$. Then we maintain maximum heat storage mass x_2. We assume the generic flowrate v_2 to be zero also in this case. The surrounding temperature in the heat storage room v_3 is a constant $20^{\circ}C$. The incoming and outgoing energy carried with u_2 and u_3 are equal. We then find out by solving equation (5.2) that the heating element power u_1 must be 5 kW to compensate for the heat storage energy loss to its surroundings. We assume a constant outside temperature v_4 at $5^{\circ}C$. We also assume that the application has only been heated by human activity. This means that the initial application temperature x_4 will be $5^{\circ}C$, since equal temperature between application and surroundings leads to no conduction. The initial radiator temperature x_3 will then also be 5 $^{\circ}C$. With constant heat storage temperature x_1 at $70^{\circ}C$ and heat storage mass x_2 at 1000 kg, we get the response for radiator temperature x_3 and application temperature x_4 shown in figure 5.4.

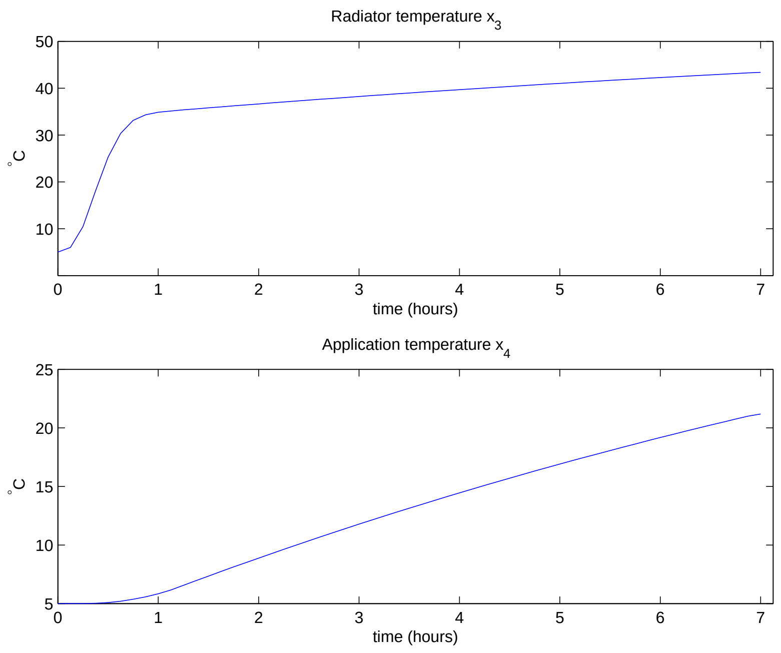

Figure 5.4: Response of radiator and application temperature

From figure 5.4 we can see that it takes between 6 and 7 hours before the application temperature x_4 reaches its desired temperature at $20°C$. The radiator temperature x_3 has a much quicker response due to much less energy needed to heat it up. It flats out after about 1 hour since the convection between radiator and application increases.

Scenario 2:

We want to illustrate how the application temperature x_4 is influenced by variation in the outside temperature. We start by deciding a set of nominal values for the different states, inputs and disturbances. The same method will be used in chapter 6 when we want to linearize the model for use with control. The decided set of nominal values are shown in table 5.5.

states	value	unit
x_1	70	$^\circ C$
x_2	750	kg
x_4	20	$^\circ C$
disturbances	value	unit
v_1	70	$^\circ C$
v_2	0	$\frac{l}{s}$
v_3	20	$^\circ C$
v_4	4.796	$^\circ C$

Table 5.5: Preset nominal states and disturbances

The nominal outside temperature v_4 is the average of the temperature forecast gathered from our testing scenario at Longyearbyen shown in chapter 4. The inputs u_1, u_2, u_3 and the state x_3 are found by finding the equilibrium points for equation (5.2)-(5.5), that is inserting the preset values in table 5.5 and set the equation (5.2)-(5.5) equal zero. This gives the values shown in table 5.6.

states	value	unit
x_3	23.953	$^\circ C$
inputs	value	unit
u_1	5	kW
u_2	0.0103	$\frac{l}{s}$
u_3	0.0103	$^\circ C$

Table 5.6: Equilibrium points based on preset values

Replacing the the outside temperature v_4 with the temperature data gathered from the weather forecast at our testing scenario, gives the response shown in figure 5.5 during the 53 hours we had of available data.

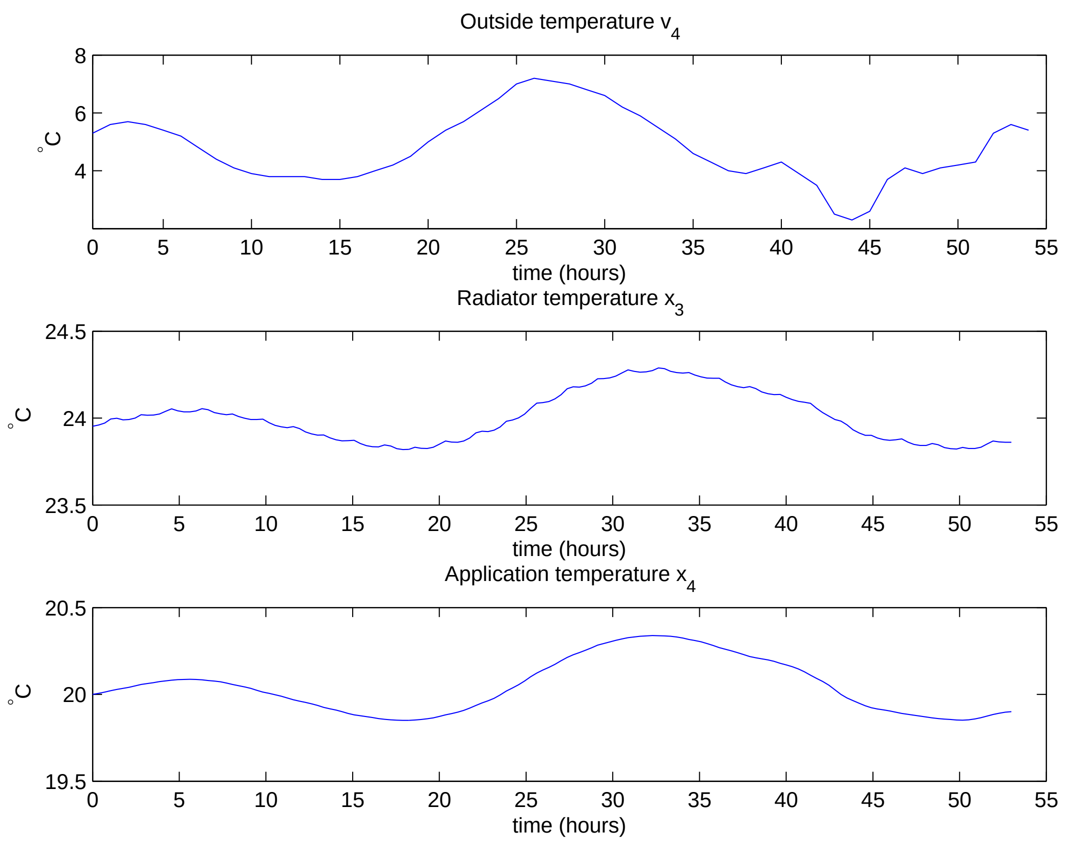

Figure 5.5: Outside temperature v_4, radiator temperature x_3 and application temperature x_4

We will later introduce control to our system. The solar collector in figure 5.1 will then be exposed to the radiation flux with incorporated cloudiness from our testing scenario shown in figure 4.3 from chapter 4. The outside temperature will be the same as in figure 5.5.

Chapter 6

Linearization

6.1 Linearizing the nonlinear heat system model

To make control of our nonlinear system (5.2)-(5.5) simpler, a linearization is favorable. The system linearization will be on the form:

$$\Delta\dot{x} = A\Delta x + B\Delta u + C\Delta v \tag{6.1}$$

where x is states, u is inputs and v is disturbances. $\dot{x}$ is the time derivative $\frac{dx}{dt}$.

If we call $f_i = \frac{dx_i}{dt}$ in (5.2)-(5.5) we get:

$$\frac{\partial f_1}{\partial x_1} = -\frac{1}{x_2}(u_3 + v_2 + \frac{g_1}{c_w}) \tag{6.2}$$

$$\frac{\partial f_1}{\partial x_2} = -\frac{1}{x_2^2}(\frac{1000u_1}{c_w} + u_2v_1 - x_1u_3 - x_1v_2 - \frac{g_1}{c_w}(x_1 - v_3)) \tag{6.3}$$

$$\frac{\partial f_3}{\partial x_1} = \frac{u_3}{m_r} \tag{6.4}$$

$$\frac{\partial f_3}{\partial x_3} = -(\frac{u_3}{m_r} + \frac{g_2}{m_rc_w}) \tag{6.5}$$

$$\frac{\partial f_3}{\partial x_4} = \frac{g_2}{m_rc_w} \tag{6.6}$$

$$\frac{\partial f_4}{\partial x_3} = \frac{g_2}{m_{room}c_{room}} \tag{6.7}$$

$$\frac{\partial f_4}{\partial x_4} = -\frac{g_2 + g_3}{m_{room}c_{room}} \tag{6.8}$$

$$\frac{\partial f_1}{\partial u_1} = \frac{1000}{c_wx_2} \tag{6.9}$$

$$\frac{\partial f_1}{\partial u_2} = \frac{v_1}{x_2} \tag{6.10}$$

$$\frac{\partial f_1}{\partial u_3} = -\frac{x_1}{x_2} \tag{6.11}$$

$$\frac{\partial f_2}{\partial u_2} = 1 \tag{6.12}$$

$$\frac{\partial f_2}{\partial u_3} = -1 \tag{6.13}$$

$$\frac{\partial f_3}{\partial u_3} = \frac{x_1 - x_3}{m_r} \tag{6.14}$$

$$\frac{\partial f_1}{\partial v_1} = \frac{u_2}{x_2} \tag{6.15}$$

$$\frac{\partial f_1}{\partial v_2} = -\frac{x_1}{x_2} \tag{6.16}$$

$$\frac{\partial f_1}{\partial v_3} = \frac{g_1}{c_wx_2} \tag{6.17}$$

$$\frac{\partial f_2}{\partial v_2} = -1 \tag{6.18}$$

$$\frac{\partial f_4}{\partial v_4} = \frac{g_3}{m_{room}c_{room}} \tag{6.19}$$

where the different components goes in the matrices as follows:

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} \tag{6.20}$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} & \frac{\partial f_3}{\partial u_3} \\ \frac{\partial f_4}{\partial u_1} & \frac{\partial f_4}{\partial u_2} & \frac{\partial f_4}{\partial u_3} \end{bmatrix} \tag{6.21}$$

$$C = \begin{bmatrix} \frac{\partial f_1}{\partial v_1} & \frac{\partial f_1}{\partial v_2} & \frac{\partial f_1}{\partial v_3} & \frac{\partial f_1}{\partial v_4} \\ \frac{\partial f_2}{\partial v_1} & \frac{\partial f_2}{\partial v_2} & \frac{\partial f_2}{\partial v_3} & \frac{\partial f_2}{\partial v_4} \\ \frac{\partial f_3}{\partial v_1} & \frac{\partial f_3}{\partial v_2} & \frac{\partial f_3}{\partial v_3} & \frac{\partial f_3}{\partial v_4} \\ \frac{\partial f_4}{\partial v_1} & \frac{\partial f_4}{\partial v_2} & \frac{\partial f_4}{\partial v_3} & \frac{\partial f_4}{\partial v_4} \end{bmatrix} \tag{6.22}$$

and the components in A, B and C that are not listed in (6.2)-(6.19) are zero.

Since we want to minimize the use of the actual input u_1 and not the deviation from it's nominal value, we have to incorporate an offset into the linearized model. Since:

$$\Delta x = x - x_{nominal} \tag{6.23}$$
$$\Delta u = u - u_{nominal} \tag{6.24}$$
$$\Delta v = v - v_{nominal} \tag{6.25}$$

where $x_{nominal}$, $u_{nominal}$ and $v_{nominal}$ are the nominal values used to linearize the nonlinear differential equations in (5.2)-(5.5). Inserting (6.23)-(6.25) into (6.1) gives:

$$\Delta\dot{x} = Ax + Bu + Cv + \underbrace{(-Ax_{nominal} - Bu_{nominal} - Cv_{nominal})}_{\text{offset values}} \tag{6.26}$$

which we can rewrite as:

$$\Delta\dot{x} = Ax + Bu + Cv + \begin{bmatrix} bias_1 \\ bias_2 \\ bias_3 \\ bias_4 \end{bmatrix} \tag{6.27}$$

where the bias elements are the components in the 4x1 vector containing the offset values.

If the nonlinear model of the plant is described as:

$$\dot{x} = f(x, u, v) \tag{6.28}$$

then an approximated expression for the differential equation described by $\dot{x}$ is:

$$\dot{x} \approx f(x_{nominal}, u_{nominal}, v_{nominal}) + \Delta\dot{x} \tag{6.29}$$

where $f(x_{nominal}, u_{nominal}, v_{nominal})$ is a vector with constant coefficients. These coefficients is the differential parts of the nonlinear model (5.2-5.5) after inserting the nominal values. We will choose our nominal values at equilibrium points in (5.2-5.5). This means that the term $f(x_{nominal}, u_{nominal}, v_{nominal})$ will be zero. We then end up with:

$$\dot{x} \approx Ax + Bu + Cv + \begin{bmatrix} bias_1 \\ bias_2 \\ bias_3 \\ bias_4 \end{bmatrix} \tag{6.30}$$

The offset values gives a constant 4x1 vector. These values can be seen as a constant disturbance, and we can can extend the number of disturbances by four to incorporate these offsets. We can rewrite our linearized model where the inputs and disturbances are collected into one vector. The rewritten linearized system will be:

$$\begin{aligned} \dot{x} &= Ax + Dw & (6.31) \\ y &= Gx + Hw & (6.32) \end{aligned}$$

While A in (6.31) remains as in (6.20), D will include the matrices (6.21), (6.22) and the offset values. A measurement state y (6.32) has also been created. This will be equal to x, such that the matrix G will be the identity matrix with same dimension as A. The H matrix will contain only zeros with the same dimension as D. Since we will measure the states, the matrix G and H are redundant, that is (6.32) could be simplified to $y = x$. Since our controller in chapter 8 requires these measurement matrices, (6.32) contains G and H.

We use the following partition of D which goes in to the linearized model in (6.31):

$$D = [D_1 D_2] \tag{6.33}$$

where

$$D_1 = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} & \frac{\partial f_1}{\partial v_1} & \frac{\partial f_1}{\partial v_2} & \frac{\partial f_1}{\partial v_3} & 0 \\ 0 & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} & 0 & \frac{\partial f_2}{\partial v_2} & 0 & 0 \\ 0 & 0 & \frac{\partial f_3}{\partial u_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial f_4}{\partial v_4} \end{bmatrix} \tag{6.34}$$

$$D_2 = \begin{bmatrix} bias_1 & 0 & 0 & 0 \\ 0 & bias_2 & 0 & 0 \\ 0 & 0 & bias_3 & 0 \\ 0 & 0 & 0 & bias_4 \end{bmatrix} \tag{6.35}$$

where the partial derivatives that equals zero are set to zero in 6.34. The w vector becomes:

$$w = \begin{bmatrix} u_1 & u_2 & u_3 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{bmatrix}^T \tag{6.36}$$

and the elements v_5 - v_8 are constant disturbance set to 1.

6.2 Applying Nominal Values

Since we need to find nominal values, i.e. operating points which we linearize the model, we must first make some assumptions for some of the variables described in table 5.1. We want specific reference values in heat storage temperature, heat storage mass and application temperature. Maintaining a comfortable application temperature is the main objective of the heat system, but it must be done under controlled conditions, i.e. maintaining reasonable values in the the other states. Water can not be too hot, and water mass of the heater storage can not be negative or exceed its maximum capacity. Since we can not do anything with the disturbances, we must approximate some operational values for them. Since we have four differential equations (5.2)-(5.5), we can calculate 4 variables at steady state by setting these equations to zero.

We will insert the values of our scenario calculated at Longyearbyen in Svalbard. The chosen linearization values are shown in table 6.1, and the rest of the values will be calculated by setting the time derivative in the the nonlinear equations (5.2)-(5.5) to zero, i.e. equilibrium.

set letters	value	unit
x_1	70	$°C$
x_2	750	kg
x_4	20	$°C$
v_1	70	$°C$
v_2	0	$\frac{l}{s}$

Table 6.1: Approximated Nominal Values

The different letters in table 6.1 were explained in table 5.1.

This leaves us with remaining 5 unknowns, that is the state x_3, the inputs u_1, u_2 and u_3 and the disturbance v_4 also explained in 5.1. The nonlinear equations described in (5.2)-(5.5) will be used to decide four of the remaining five linearization points, since we have only four equations. The outside temperature v_4 will be set to the average temperature of the temperature graph gathered from our testing scenario in Longyearbyen in Svalbard. The remaining 4 unknowns will be calculated setting the four differential equations in (5.2)-(5.5) to zero, which lead to the following equations:

$$\begin{aligned} x_3 &= x_4(1+\frac{g_3}{g_2})-\frac{g_3}{g_2}v_4 && (6.37)\\ u_3 &= \frac{g_2(x_3-x_4)}{cw(x_1-x_3)} && (6.38)\\ u_2 &= u_3 && (6.39)\\ u_1 &= \frac{1}{1000}(-c_w u_2 v_1 + c_w x_1(u_3+v_2)+g_1(x_1-v_3)) && (6.40)\end{aligned}$$

Calculating the remaining values based on (6.37)-(6.40) gives the values shown in table 6.2.

calculated	value	unit
v_4	4.7963	$°C$
x_3	23.953	$°C$
u_1	5	kW
u_2	0.0103	$\frac{l}{s}$
u_3	0.0103	$\frac{l}{s}$

Table 6.2: Calculated Nominal Values

These values will then be inserted in the partial derivatives (6.2)-(6.19) which goes into the matrices (6.20)-(6.22) of the linearized model.

The used method of finding nominal values based on equilibrium points works fine in many cases. In this case we can immediately see that this causes problems for one partial derivative (6.3) which was:

$$\frac{\partial f_1}{\partial x_2} = -\frac{1}{x_2^2}(\frac{1000u_1}{c_w} + u_2v_1 - x_1u_3 - x_1v_2 - \frac{g_1}{c_w}(x_1 - v_3)) \quad (6.41)$$

By comparing (5.2) and (6.41), we can see that they are similar due to derivative rules. This means that (6.41) will be equal to zero, which is true if all the other values stays at their nominal values. Once we operate outside the exact nominal values, it means that the linearized model interprets that the derivative of the temperature in the heat storage x_1 as independent of the mass x_2 in the tank, which is not true. Differential equation (5.2) is strongly nonlinear with respect to x_2. The reason for this is due to division by x_2^2 which can be seen in equation (6.41). These kind of problems can be solved in several ways. One way is to use other nominal values than equilibrium values in which to linearize our model. Since equation (6.41) is strongly nonlinear, linearizing around other values would still be inaccurate. Another solution to the problem is to introduce constraint on the heat storage temperature x_1 around its chosen nominal value when applying control to the system, which will reduce the problem of the nonlinearity.

Chapter 7

Model predictive control

7.1 Introduction

Model predictive control (mpc) [2] is a control strategy that uses a model of a process to predict its output at future time instants within a prediction horizon. The controller then calculates a control sequence that minimizes an objective function subject to physical and operational constraints. The future outputs for a determined prediction horizon P are predicted at each instant t using this process model. These predicted outputs y(t+k|t) for k = 1...P depends on the known values up to instant t(past inputs and outputs) and on the future control signals u(t+k|t), k=0...P-1, which are the control signals sent to the system and calculated. The set of future control signals are calculated by optimizing a determined set of reference criterions. The control horizon M decides how many inputs are calculated. If the control horizon is less than the prediction horizon, the final computed move usually fills the remainder of the prediction horizon. The criterion usually takes the form of a quadratic function of the errors between the predicted output signal and the reference. These errors can be weighted according to what outputs is important to keep close to the reference. Constraints on inputs and outputs can also be incorporated in the objective function. The first control signal calculated is applied while the rest of the signals are rejected. Rejecting the rest of the control signals is done since new information (one more set of inputs and outputs) are available. The horizon is displaced one step towards the future and the controller recalculates a new control sequence. An example of the mpc strategy mentioned is shown in figure 7.1 [5] with prediction horizon P = 9 and control horizon M = 4. The input constraints u_{min} and u_{max}, output constraints y_{min} and y_{max} and output reference r are also shown. The example is single input single output (SISO).

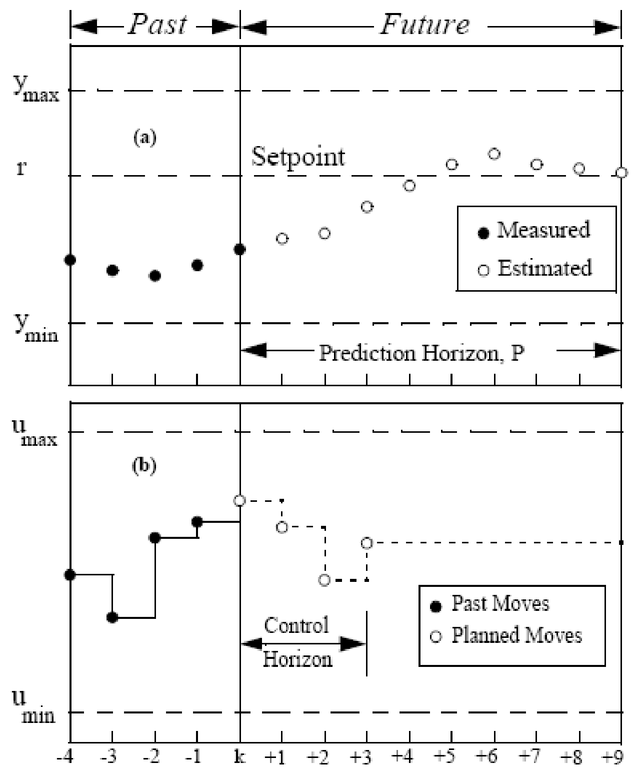

Figure 7.1: Strategy of mpc example showing prediction horizon P=9(a) and control horizon M=4(b)

Figure 7.2 shows a general plant with a mpc controller. The plant contains measured disturbances v(k), unmeasured disturbances $d(k)$, references $r(k)$, a constant unmeasured disturbance on the output and the measured output $y_m(k)$. The plant model may be nonlinear, while the model of the plant in the mpc controller is linearized. We see in figure 7.2 that both the measured output and the measured disturbance are fed into the mpc controller. Since the heat system deducted in chapter 5 only had one unmeasured disturbance, we chose to denote it v(the same letter as the measured disturbances). Even though we have denoted the unmeasured disturbance v in the heat system, d wil be used in this chapter to seperate the different disturbance types.

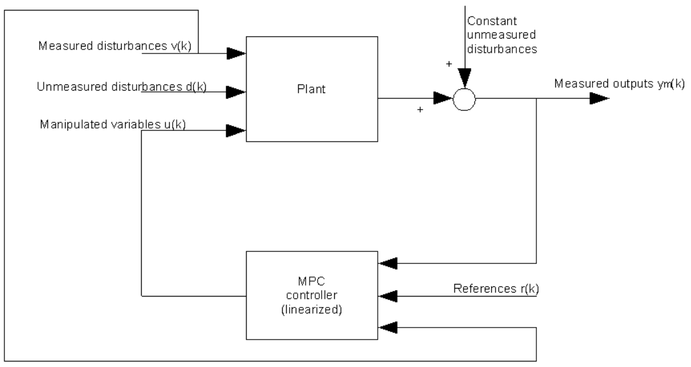
Measured disturbances v(k)
Unmeasured disturbances d(k)
Manipulated variables u(k)
Plant
Constant unmeasured disturbances
+
+
Measured outputs ym(k)
MPC controller (linearized)
References r(k)

Figure 7.2: Plant and mpc controller

The basic structure of the mpc controller [2] in figure 7.2 is shown in figure 7.3. The past inputs in figure 7.3 are calculated from the mpc controller itself, while the measured disturbances are measured from the original plants input and fed into the mpc controller. This is a sort of feed forward control. The past outputs are delivered from the measured outputs of the plant being controlled.

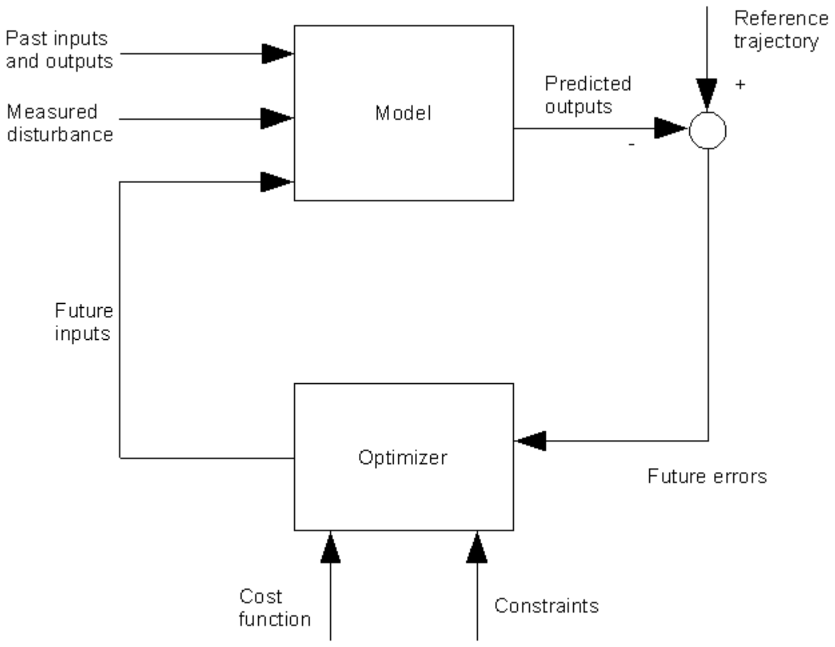

Figure 7.3: Basic structure of mpc

7.2 Model Predictive Control toolbox

7.2.1 Introduction

The Model Predictive Control (MPC) toolbox [5] is a collection of software that helps you design, analyze, and implement an advanced mpc algorithm. The Toolbox is available in the Simulink library in Matlab [5].

7.2.2 Prediction model

The linear model used in the MPC Toolbox for prediction and optimization is shown in figure 7.4.

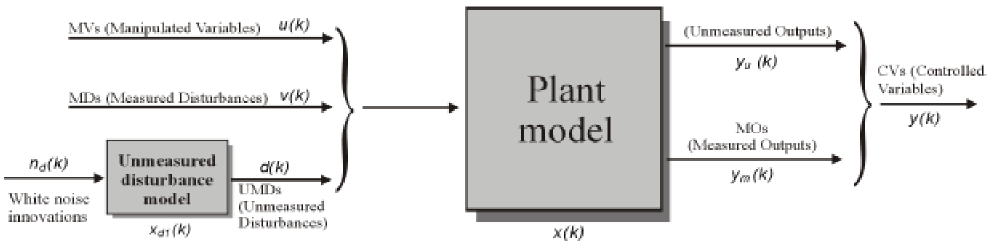
MVs (Manipulated Variables) u(k)
MDs (Measured Disturbances) v(k)
n_d(k)
White noise innovations
Unmeasured disturbance model
x_d1(k)
d(k)
UMDs (Unmeasured Disturbances)
Plant model
x(k)
(Unmeasured Outputs)
y_u(k)
MOs (Measured Outputs)
y_m(k)
CVs (Controlled Variables)
y(k)

Figure 7.4: Prediction Model

The model of the plant in figure 7.4 is a linear time-invariant (LTI) system described by the equations:

$$x(k+1) = Ax(k) + B_u u(k) + B_v v(k) + B_d d(k) \quad (7.1)$$

$$y_m(k) = C_m x(k) + D_{vm} v(k) + D_{dm} d(k) \quad (7.2)$$

$$y_u(k) = C_u x(k) + D_{vu} v(k) + D_{du} d(k) + D_{uu} u(k) \quad (7.3)$$

where x(k) is the n_x-dimensional state vector of the plant, u(k) is the n_u-dimensional vector of manipulated variables (MV), i.e., the command inputs, v(k) is the n_v-dimensional vector of measured disturbances (MD), d(k) is the n_d-dimensional vector of unmeasured disturbances (UD) entering the plant, $y_m(k)$ is the vector of measured outputs (MO), and $y_u(k)$ is the vector of unmeasured outputs (UO). The overall output vector y(k) collects $y_m(k)$ and $y_u(k)$. In the above equations d(k) collects both state disturbances ($B_d \neq 0$) and output disturbances ($D_d \neq 0$). The MPC toolbox collects matrices with the same capital letter into one matrix. The same applies to the vectors being multiplied by those matrices. They are just separated in parts in (7.1)-(7.3) to show the different contributions.

If the actual plant being controlled consists of unmeasured disturbance, the input disturbance model in figure 7.4 is the linear model:

$$x_{d1}(k+1) = \bar{A}x_{d1}(k) + \bar{B}n_d(k) \quad (7.4)$$
$$d(k) = \bar{C}x_{d1}(k) + \bar{D}n_d(k) \quad (7.5)$$

where $x_{d1}(k)$ is the state of the unmeasured input disturbance model and d(k) is the actual unmeasured input disturbance. White noise is chosen as input in equation (7.4) and (7.5) in figure 7.4.

7.2.3 State estimation model

At the beginning of each sampling instant the controller estimates the current plant state. Accurate knowledge of the state improves prediction accuracy, which, in turn, improves controller performance. Figure 7.4 shows the model used for state estimation

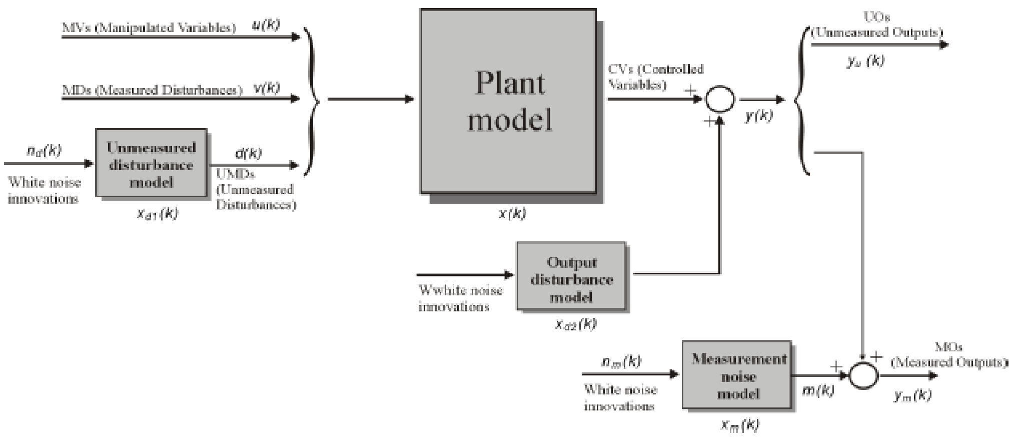

Figure 7.5: State Estimation Model

The measurement noise can be modeled with a linear model driven by Gaussian white noise or other forms of inputs. The Gaussian white noise $n_m(k)$ and $n_d(k)$ has zero mean and unit covariance matrix. If white noise is chosen as input, the measurement noise model on figure 7.5 will be the linear model:

$$x_m(k+1) = \tilde{A}x_m(k) + \tilde{B}n_m(k) \quad (7.6)$$
$$m(k) = \tilde{C}x_m(k) + \tilde{D}n_m(k) \quad (7.7)$$

where $x_m(k)$ is the state of the measurement noise model and m(k) is the measurement noise.

The output disturbance model $x_{d2}(k)$ in figure 7.5 is a collection of integrators driven by white noise on measured outputs.

7.2.4 State observer

The state observer in mpc toolbox is designed to provide estimates of $x(k)$, $x_d(k)$, $x_m(k)$, where $x(k)$ is the state of the plant model, $x_d(k)$ is the overall state of the input and output disturbance model, $x_m(k)$ is the state of the measurement noise model. The estimates are computed from the measured output $y_m(k)$ by the linear state observer:

$$\begin{bmatrix} \hat{x}(k|k) \\ \hat{x}_d(k|k) \\ \hat{x}_m(k|k) \end{bmatrix} = \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{x}_d(k|k-1) \\ \hat{x}_m(k|k-1) \end{bmatrix} + M(y_m(k) - \hat{y}_m(k)) \tag{7.8}$$

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{x}_d(k+1|k) \\ \hat{x}_m(k+1|k) \end{bmatrix} = \begin{bmatrix} A\hat{x}(k|k) + B_u u(k) + B_v v(k) + B_d \bar{C}\hat{x}_d(k|k) \\ \bar{A}\hat{x}_d(k|k) \\ \tilde{A}\hat{x}_m(k|k) \end{bmatrix} \tag{7.9}$$

$$\begin{aligned} \hat{y}_m(k) = C_m\hat{x}(k|k-1) + D_{vm}v(k) + D_{dm}\bar{C}\hat{x}_d(k|k-1) \\ + \tilde{C}\bar{x}_m(k|k-1) \end{aligned} \tag{7.10}$$

where "m"' denotes the rows of C,D corresponding to measured outputs. The states $\hat{x}$ and $\hat{y}$ are estimations of states and outputs, k is the different time instants and the matrices in (7.8)-(7.10) was listed in equation (7.1)-(7.3), (7.4)-(7.5) and (7.6)-(7.7). M is a gain matrix designed using Kalman filtering techniques, if noise or unmeasured disturbance are present.

7.2.5 Cost function

The cost function in the mpc toolbox is on the form:

$$\begin{aligned} J = &\left(\begin{bmatrix} u(0) \\ \cdots \\ u(p-1) \end{bmatrix} - \begin{bmatrix} u_{target}(0) \\ \cdots \\ u_{target}(p-1) \end{bmatrix} \right)^T W_u^2 \left(\begin{bmatrix} u(0) \\ \cdots \\ u(p-1) \end{bmatrix} - \begin{bmatrix} u_{target}(0) \\ \cdots \\ u_{target}(p-1) \end{bmatrix} \right) \\ + &\left(\begin{bmatrix} y(1) \\ \cdots \\ y(p) \end{bmatrix} - \begin{bmatrix} r(1) \\ \cdots \\ r(p) \end{bmatrix} \right)^T W_y^2 \left(\begin{bmatrix} y(1) \\ \cdots \\ y(p) \end{bmatrix} - \begin{bmatrix} r(1) \\ \cdots \\ r(p) \end{bmatrix} \right) + \rho_\varepsilon \varepsilon^2 \end{aligned} \tag{7.11}$$

where u is input is, y is measurements and r is reference. W_u and W_y are the different weights on the deviations and p is the prediction horizon. Notice that the weights are squared. The term $\rho_\varepsilon \varepsilon^2$ incorporates the constraints. The constraints on u and y are relaxed by introducing the slack variable

$\varepsilon \geq 0$. Usually only constraints on outputs are relaxed, but the MPC toolbox can relax inputs as well. The weight ρ_ε on the slack variable ε penalizes the violation of the constraints. The larger ρ_ε with respect to input and output weights, the more the constraint violation is penalized. The algorithm implemented in the mpc toolbox uses different procedures depending on the presence of constraints when optimizing. If all the bounds are infinite, the slack variable ε is removed, and the problem in equation 7.11 is solved analytically. Otherwise a Quadratic Programming (QP) solver is used.

7.2.6 Horizons

The mpc toolbox contains three properties for horizons. The control interval sets the elapsed time between successive controller moves. The prediction horizon sets the number of control intervals over which the controller predicts its outputs when computing controller moves. The control horizon sets the number of moves computed. If less than the prediction horizon, the final computed move fills the remainder of the prediction horizon.

Chapter 8

Heat system with mpc control

8.1 Overview

Recall the heat system described by equations (5.2)-(5.5) in chapter 5 with states, disturbances, inputs and parameters described in table 5.1 and with parameter values listed in table 5.3. The heat system will be controlled with model predictive control, i.e Matlab's MPC toolbox. The heat system will also include the solar collector exposed by radiation deducted in chapter 3 and outside temperature gathered from chapter 4. The effect of the outside temperature on the solar collector is also incorporated. The chosen testing scenario in Longyearbyen in Svalbard from chapter 4 will be the location of our controlled system. Recall from figure 5.5 in chapter 5, that the duration of the simulation of the uncontrolled system was 53 hours, due to the length of the available weather forecast. This also holds for the controlled system. The Simulink model of the controlled heat system is shown in appendix B.7.

8.2 Implementing linearized model

Recall our linearized system (6.31)-(6.32) of our nonlinear heating process from chapter 6 with:

$$\begin{aligned} \dot{x} &= Ax + Dw && (8.1) \\ y &= Gx + Hw && (8.2) \end{aligned}$$

where all the states are collected in the x matrix and all the inputs (manipulated variables and disturbances) are collected in w from (6.36). The matrix A and D is as mentioned in (6.20) and (6.33). G and H are just measurement matrices and in our case G equals I (identity matrix) and

H will consist off only zeros. As mentioned in chapter 6, the measurement matrices G and H were required in the MPC toolbox.

Our nonlinear process from equation (5.2)-(5.5) is implemented with no output disturbances nor measurement noise. This means that we set the output disturbance component x_{d2} from figure 7.5 and the measurement noise component x_m from equation (7.4) to zero in our linearized model of the process.

The linearized system is discretized using the matlab function c2d, with a sampling time of 15 minutes. We put these matrices into the mpc toolbox in Simulink. Recall the states, inputs and disturbances from table 5.1 in chapter 5. We set the states x_1-x_4 as measured outputs and their nominal values as references listen in table 5.5 and 5.6.The disturbances v_1, v_3 and v_4 will be set to measured disturbances. So will also the constant disturbances v_5-v_8 set to 1 from (6.36), which took care of our biases from (6.35). We had one unmeasured disturbance, which was the generic flowrate v_2 (hot water consumption). The disturbance v_2 will be considered to be zero most of the time, which means that we will choose to set the disturbance component x_{d1} from equation (7.4) to zero as well. Our manipulated variables will be the inputs u_1-u_3. The heating element power u_1 is scaled in kW to avoid numerical problems in the QP solver of the mpc controller. The implementation of this mpc controller is done in the m-file mpcInit.m in appendix A.5.

8.3 Simulation

8.3.1 Setup

The simulation in Simulink is done in seconds, that is a 190800 second duration (53 hours). The temperature and radiation with cloudiness data from our testing site is recalled in figure 8.1.

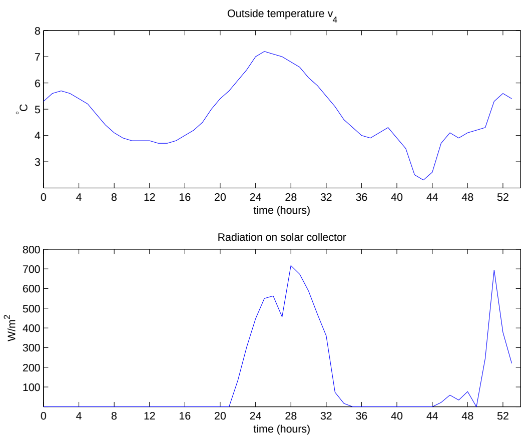

Figure 8.1: Temperature and radiation at testing scenario

These two data vectors are inserted into the temperature and radiation block in the controlled system. Lagrangian interpolation is used to fill the necessary points between every hour. These blocks will act as simulated surroundings. The generic flowrate v_2 is implemented as a square pulse in the middle of the simulation duration, that is between the 26'th and 27'th hour. The pulse lasts for 15 minutes with amplitude 0.1 $\frac{l}{s}$ as shown in figure 8.2.

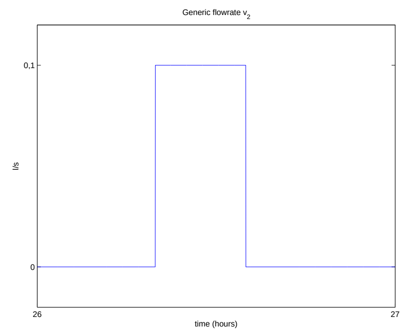

Figure 8.2: Generic flowrate due to hot water tapping

The mpc controller is set with the three horizon properties

property	value	unit
control interval	15	minutes
prediction horizon	24	hours
control horizon	2	hours

Table 8.1: mpc controller horizons

Taking into account the dynamics of the heat system, the duration of the simulation and the desire to have a long prediction and control horizon, these horizons are reasonable. The prediction and control horizons could be increased, but it would take much longer to simulate since much more calculations has to be done. If the linearized model of the nonlinear process is inaccurate, the gain of having long prediction and control horizons are also limited.

We start by tuning the mpc controller for normal operation. We want to put weights on the inputs u_1-u_3, that is how much we want to penalize the usage of them. With normal weighting, we want to penalize the use of heating element power u_1 in the heat storage tank to save power consumption. With low weighting we want to see how much more power is used doing the same job, that is keeping the application temperature x_4 at its reference value at $20°C$. This means that we will also use weight on the output x_4, to penalize the deviation from its reference value.

Recall the problem with the partial derivative (6.41) from chapter 6. We found out that our linearized model of the nonlinear process (5.2)-(5.5) was implemented in such a way that the linearized model interpreted that the temperature in the heater tank x_1 was independent of the change in mass x_2 in the tank. If we penalize the use of u_1, this means that our controller will increase the usage of the flowrate u_2 into the heat storage tank, thinking that it will increase the temperature since it carries energy. Neglecting the negative contribution from the increase in heat storage mass will be a big mistake, and will cause the heating of the application to fail. That is why we will use constraint on the heat storage temperature x_1. Since our linearized model is inaccurate due to the partial derivative problem 6.41, we will reduce this problem by introducing the mentioned constraint, which will result in the heat storage tank operating closer to its nominal value. We want to keep the temperature x_1 in the heat storage tank as low as possible when it's supplied by heating element power to minimize energy loss. However, the heat storage temperature x_1 must be high enough to supply the application with heat and to avoid bacterias. We will set an upper limit on x_1 to avoid boiling, which means x_1 will be constrained between 70 and $100°C$. The constraint may still be violated, due to for instance, inaccuracy of the linearized model or a large unexpected disturbance. In any case, a constraint will limit a states violation of the desired constraint interval.

The initial values of the states, inputs and disturbances will be set to their nominal values listed in table 6.1 and table 6.2.

8.3.2 Simulating with normal weight on input u_1

The set of weights on inputs u_1-u_3 used in the normal operating case is shown in table 8.2 and the constraints are shown in table 8.3. The units were listed in table 5.4.

weights	value
u_1	10
u_2	100
u_2	100
x_1	0
x_2	0
x_3	0
x_4	1000

Table 8.2: Normal weights

variable	lower limit	upper limit
x_1	70	100
x_2	0	1000
x_3	0	100
x_4	0	100
u_1	0	10
u_2	0	0.1
u_3	0	0.1

Table 8.3: Normal constraints

Even though we don't care about the usage of input u_2 and u_3 inside its constraints, we have weighted it to avoid singularities in the QP solver in the mpc algorithm. Besides the mentioned constraint on heat storage temperature x_1, the rest of the states and inputs are constrained based on what was listed in table 5.4. We have relaxed the lower constraint of x_3 and x_4 to zero compared to table 5.4.

The response of the states when simulating under these conditions is shown in figure 8.3. Figure 8.3 shows that all the states over or undershoots until a few control intervals has passed. The application temperature x_4 tracks well on its reference value after these few intervals. We also see the effect of the generic flowrate v_2, which makes the heat storage temperature x_1 drop a little under its constraint until it tracks back. Figure 8.4 shows the the inputs under the same simulation. Notice the descent of heating element power u_1 when the temperature of the incoming water v_1 rises as shown in figure 8.5. The middle peak on both u_1 and u_2 in figure 8.4 is due to the generic flowrate v_2 that taps a lot of energy from the heat storage tank.

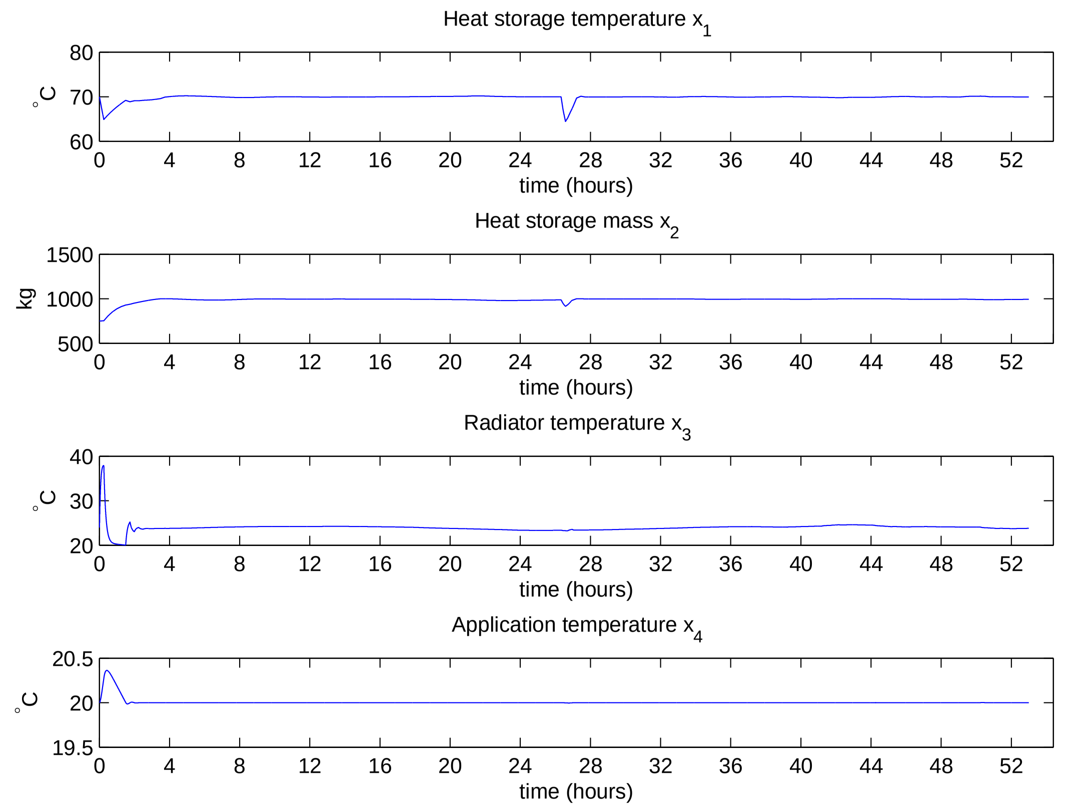

Figure 8.3: States with normal weight on u_1

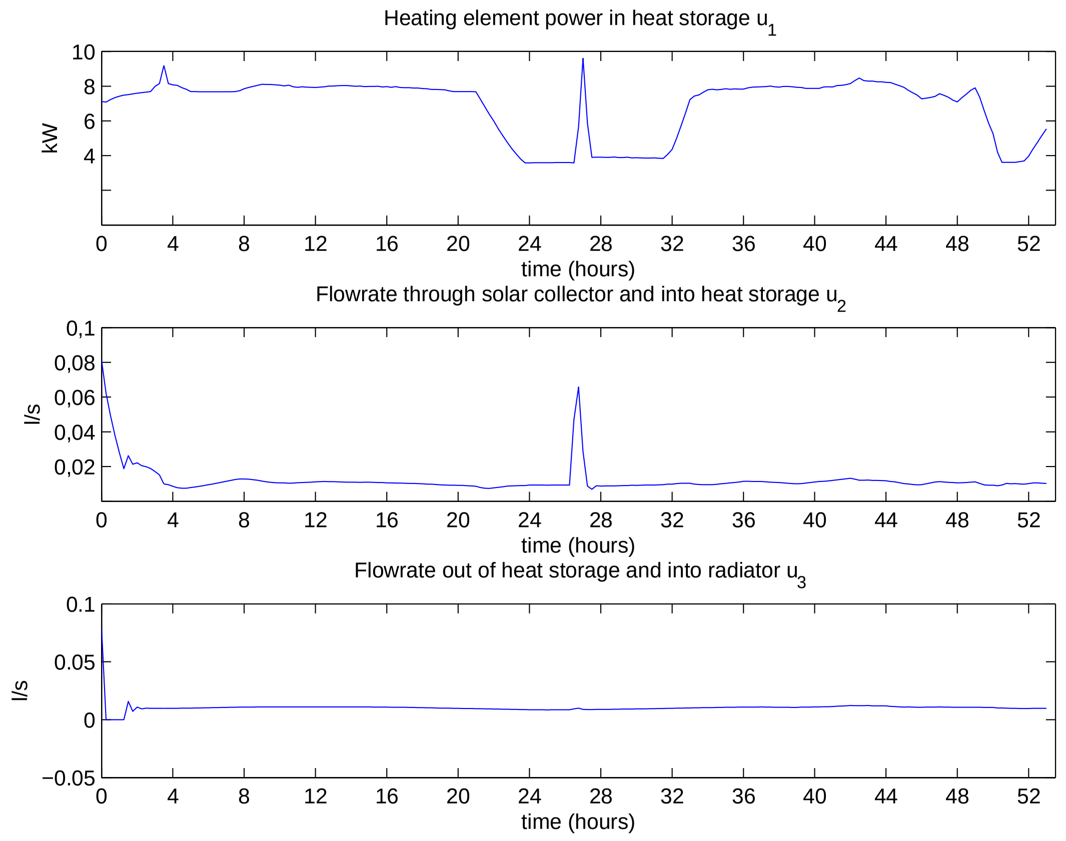

Figure 8.4: Inputs with normal weight on u_1

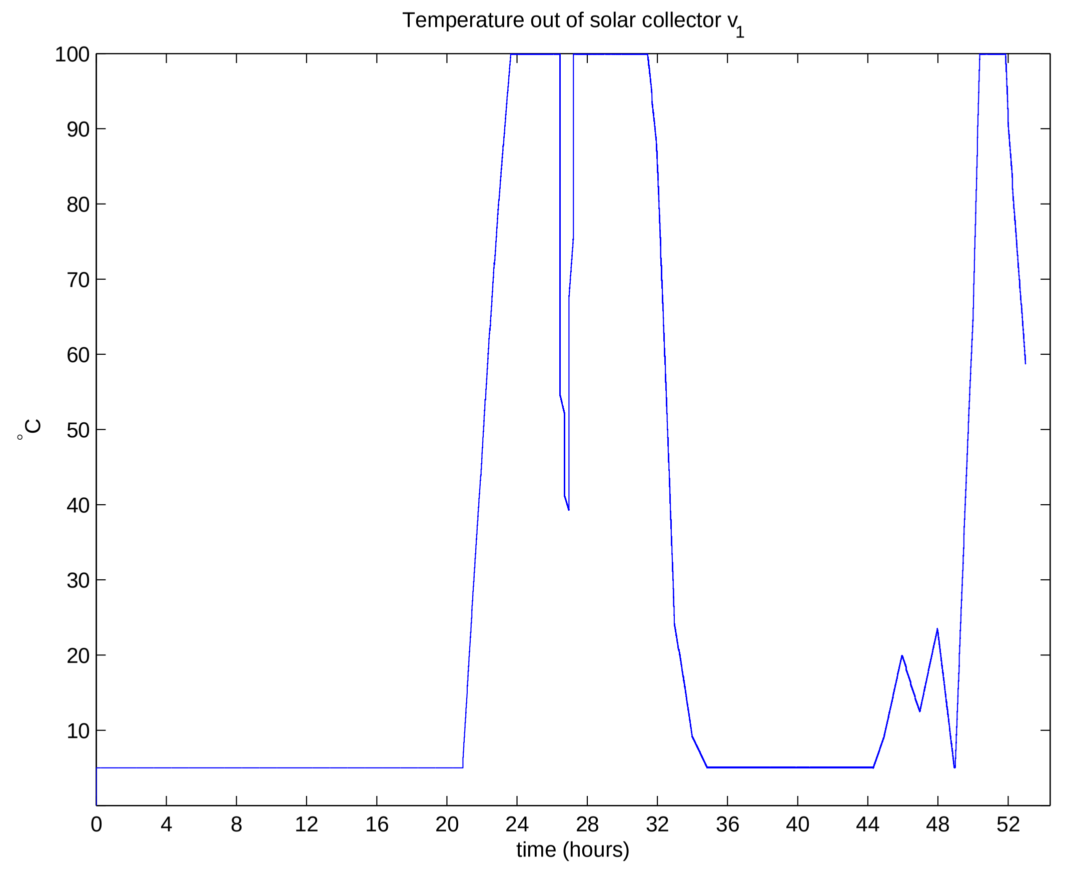

Figure 8.5: Temperature out of solar collector with normal weight on u_1

8.3.3 Simulating with low weight on input u_1

We want to simulate with lower weighting on u_1 to demonstrate the effects. We reduce the weight on u_1 from 10 to 1, that is a reduction of a factor of 100 (since the weights are squared in (7.11)). The rest of the weights will be the same as in table 8.2. Since the weights on the input u_1 are lower, it will be more desirable for the mpc controller to heat up the application temperature x_4 with high heating element power u_1 and the flowrates u_2 and u_3 as low as possible. This will cause the controller to desire to heat up the heat storage temperature x_1 to an unreasonable temperature. It will also try to empty the heat storage tank, to minimize the total usage of u_2 and u_3, while still satisfying the reference of the application temperature x_4. Recall that our linearized model interpreted that the heat storage mass x_2 did not have any effect on the derivative of heat storage temperature x_1. When the mass of the heat storage tank $x_2 \rightarrow 0$, the heat storage temperature $x_1 \rightarrow \infty$. Because of the upper constraints on the heat storage temperature x_1, the peak will be limited. Oscillations is still unavoidable because of this linearization choice. The oscillations can be reduced by having a higher lower constraint on the heat storage mass x_2, which have been done in this case. The lower constraint on x_2 has been set to 500 kg.

We can see from the plots of the states in figure 8.6, that the heat storage temperature x_1 is leaping against its upper constraint on temperature due to the low weighting on heating element power u_1. We also see that the heat storage mass x_2 is leaping on its lower constraint on 500 kg, just as expected. We also see the oscillations on the radiator temperature x_3 and the small oscillations on the application temperature x_4 around its reference point. The oscillations are also reflected on the inputs in figure 8.7. We see that the heating element power u_1 runs at the upper constrained power until the heat storage temperature x_1 reaches its upper constraint. Then the power drops until the heat storage temperature x_1 drops, and the heating element power u_1 goes back to top. As we can see it's a struggle between optimizing with respect to the given weights, while still maintaining the constraints. The total use of heating element power u_1 is substantially higher with low weight on u_1 as expected. The temperature of the incoming water v_1 is shown in figure 8.8. It also oscillates a lot more than with the normal weighting. Recall that the lower limit of the temperature out of the solar collector v_1 was $5°C$. The part of figure 8.8 where v_1 reaches zero is when no water is flowing through the solar collector, as is defined and coded in getSolarTempOut.m in appendix A.4.

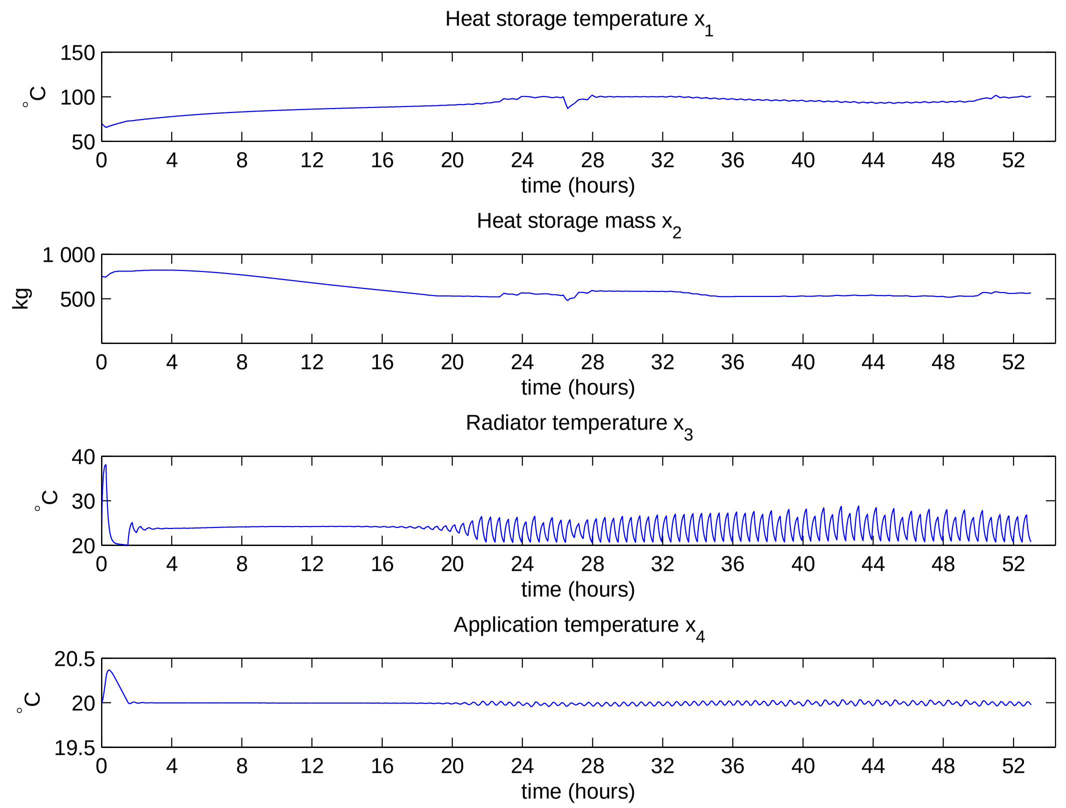

Figure 8.6: States with low weight on u_1

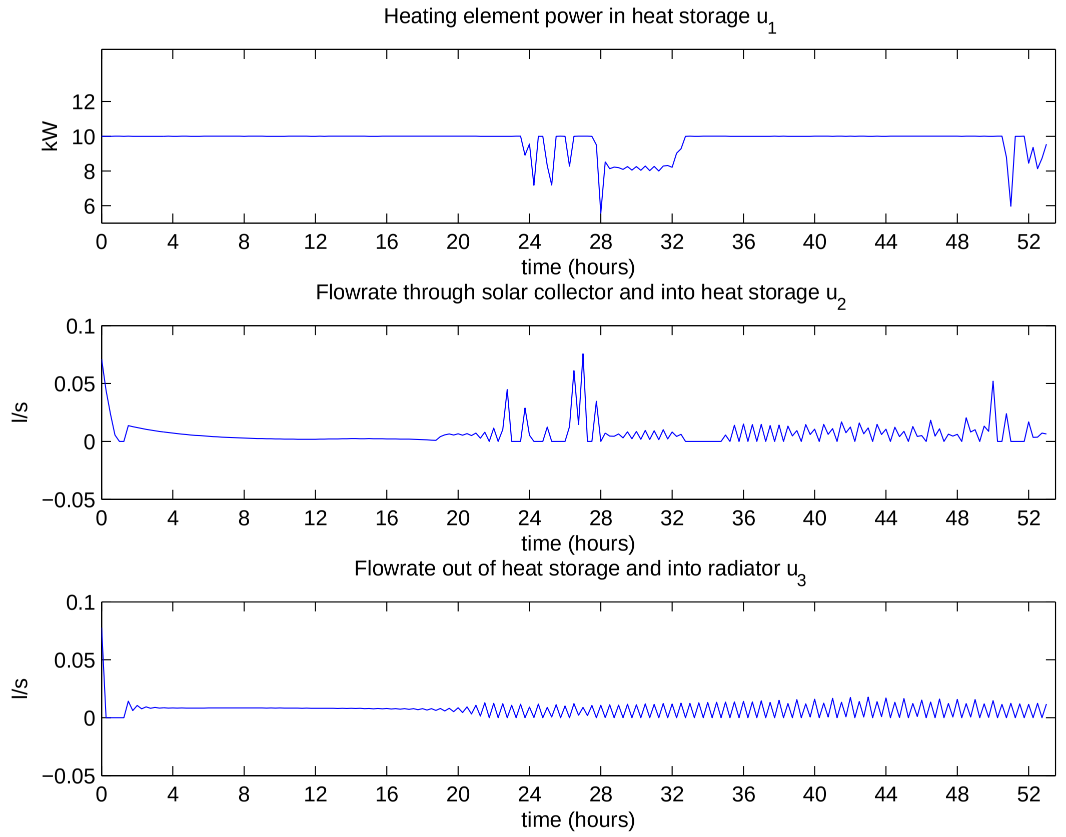

Figure 8.7: Inputs with low weight on u_1

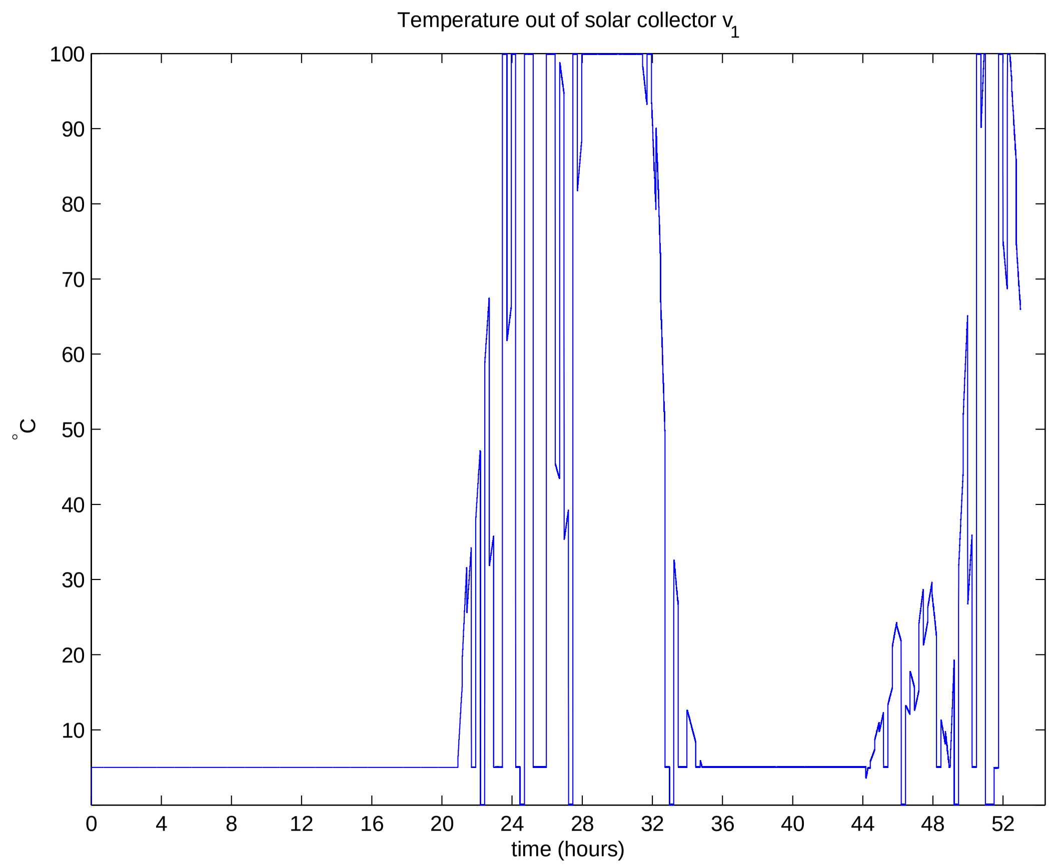

Figure 8.8: Temperature out of solar collector with low weight on u_1

8.3.4 Energy consumption in the simulated scenarios

As was mentioned, the energy consumption when simulating with normal weight on input u_1 was substantially lower than with low weight on the same input. The total energy consumption is found by numerical integrating the two u_1 graphs using the trapezoid method. Converting to kWh (kilowatt-hours) we get the total consumption under the 53 simulated hours shown in table 8.4.

weight on u_1	Energy consumption
normal	363.85 kWh
low	515.8 kWh

Table 8.4: Energy consumption in kWh

Chapter 9

Discussion and Conclusion

9.1 Discussion

The modeled heat system consisting of an application (a house) and a heat storage tank, with supplied energy from heating element power and a solar collector was very simplified. However, it was created to show a principle, that is that on can fill a energy storage in a more efficient way using model predictive control. Many variables that influences such a heat system were neglected. Components like wind, diffuse radiation and humidity were disregarded. Parameters such as coefficient of surface conductance was regarded as constants, while they will vary depending on many variables in reality. Cloudiness was regarded as a negative contribution to the solar radiation, while in many occurrences it can contribute positively with diffuse radiation. Since the weather forecasts did not give any information on where the clouds were located on the sky at a particular time, we had to simplify.

The model of the the solar collector was based on average temperature. We ended up with solving a second order equation to find the temperature of the water coming out of the solar collector. More complicated models of the temperature of the water in the solar collector can be developed. Models consisting of differential equations, where one can describe the differential in temperature of a particular water particle as a function of flowrate, radiation and location in the solar collector can be modeled. Such a solar collector model can be incorporated into the dynamics of the heat system. One can then use a controller to optimize the amount of energy gathered from the solar collector, by tuning the flowrate, while still using a mpc controller to ensure that the heat storage has enough capacity to receive the solar heated water. This will lead to less consumption of electric power, and will be a much better way to model the solar collector.

Recall that the hot water used in the radiator or as generic hot water consumption ended up in the drain after it was used. This was done to simplify the model. It is however an inefficient solution. Circulating the water from the radiator and back through the solar collector or directly into the heat storage, depending on whether solar radiation is available is more efficient. A lot less energy would then be removed form the heat system, which would again lead to less consumption of electric power. Other forms of backups to the solar collector than a heating element supplied by electric power could be incorporated in the heat system. Oil, gas and pellets is a possibility used in many systems. Extension and modification of the model would then had to be done.

The only exploit of the weather forecasts in our heat system was applying the average temperature as a nominal value when linearizing the nonlinear model (5.2)-(5.5). Otherwise the weather forecast was only used as actual surroundings to be able to simulate a scenario. A more advanced controller could be developed based on all the available weather forecast data, that is an adaptive controller that incorporated the weather forecasts in the predictor of the mpc controller. This would have resulted in more efficient mpc controller if the weather forecasts were accurate. If the solar collector in addition were modeled with differential equations, as mentioned above, the weather forecast could be incorporated in the linearized mpc model in a more predictive way. The weather forecast data could then be incorporated as a bias in the differential equations in such a way that for every hour in the prediction horizon, the bias in the differential equations were changed.

A more complicated nonlinear mpc controller could also be used to avoid the problems of linearizing a nonlinear model with strong nonlinearities. The nonlinear mpc controller is however more complicated and requires more computer power. Piecewise linear models can also be used, with different nominal values depending on measurements from the nonlinear process.

Recall that our testing scenario from chapter 4 was chosen since it is a cold place even during summer, with low outside temperature. This means that a heat system exposed by these surroundings, requires a positive energy supply to be able to maintain the reference temperature in the application in the heat system. One can imagine other scenarios where the outside temperature is higher than the reference temperature of the application. Cooling would then be required, which the system deducted in chapter 5 is not able to do. If cooling is required, one can have a controller in the input of the radiator which switches to another water circuit. The other circuit could come directly from the water supplier to the radiator with low water temperature. The reverse effect of the one deducted in chapter 5 would then occur, which means that the radiator would cool the application. The water in the radiator, which would be heated from the application, could then be circulated back through the solar collector or directly in the heat storage to conserve that energy. If no radiation was present at that particular time, it would be unnecessary to pass the water through the solar collector. No radiation during the whole testing scenario would also make the mpc controller used in chapter 8 redundant, since all the required heating would have to come from the heating element in the heat storage tank. All the mentioned functionality requires however more circuits which results in more complicated modeling with more comprehensive equations.

9.2 Conclusion

We have found out that using model predictive control on a heat system containing a solar collector can be efficient. Our goal was to be able to reduce consumption of electric power when heating an application in order to reduce consumption of electric power. We wanted to be able to fill the heat storage in our heat system with heated water from the sun, and to prevent that the heat storage had already filled up its capacity when heated water from the solar collector was available. Weather data in forms of forecasts was collected from a testing scenario and used with deducted formulas to create surroundings which the heat system was exposed by. The testing scenario was chosen from a location far north, to ensure that the outside temperature would be lower than the inside reference temperature of the application. The average temperature from the weather forecast was used as a nominal value for the linearization of the nonlinear model (5.2)-(5.5). The mpc controller managed by a sort of feed forward, to decrease the usage of electric power in the heat storage tank when heated water from the solar collector was available. The controller measured the water temperature out of the solar collector and applied it to its predictor. The outside temperature was also measured and applied in the same way. With normal weight on heating element power u_1 and constraint on the heat storage temperature x_1, we managed to have a low power loss while still maintaining a reasonable temperature in the heat storage tank to avoid bacterias. It also resulted in the heat storage having capacity to receive heated water from the solar collector when it was available. The heat storage temperature had to be lower constrained, since our linear model was inaccurate. The linearized model was inaccurate due to the the strong nonlinearity in the differential equation (5.2) describing the heat storage temperature in the nonlinear model.

When the heating element power u_1 was low weighted, we saw that a lot more energy was used to maintain the same objective, namely to hold the reference temperature in our application. With the low u_1 weighting, the heat storage temperature had to be upper constrained, due to the mentioned linearized inaccuracy. The low weighting on u_1 made it more optimal to use much electric power and with low flowrates in and out of the tank. This resulted in the heat storage having significantly less capacity to receive heated water from the solar collector. It also resulted in oscillations in both states and inputs, since the low weight on u_1 resulted in the heat storage temperature x_1 pushing against its upper constraint.

Using constraints on the outputs in the controlled simulations made the inaccurate linearized model of the nonlinear process acting more reasonable. This was also expected, since the different weights on inputs have less to say in the cost function of the mpc controller when a state is close to one of its constraints. It's also well known that a mpc controller works better the more accurate the linearized model describing the nonlinear process is.

The chosen testing scenario represented the situation where the outside temperature was low enough that the application needed to be heated. Solar radiation was also present in the test scenario. As mentioned in section 9.1, a cooling functionality would have had to be incorporated if the outside temperature was higher than the reference temperature of the application. If enough hot water was tapped from the heat storage tank in terms of hot water consumption, the application of the heat system could also have had problems maintaining its reference temperature. Other energy contributions mentioned in section 9.1 could then be used to contribute to ensure that the heat system had enough energy to maintain the reference temperature in the application and in the heat storage. The heating element power u_1 would also run on full power if the energy consumption from the heat storage tank was high enough. This means that the mentioned conclusions only holds for scenarios where solar radiation is present, and where moderate amounts of energy are drawn out of the heat storage. If for instance the outside temperature is to low, the heating element power u_1 would have to run at maximum power to maintain the reference temperature of the application. At low temperatures, less solar radiation is usually present and the loss in the solar collector is also higher. It is also reasonable that the less free energy that is present, the less gain is obtained by controlling flow of free and costly energy. The big dilemma is that free energy is present when one need it least, and storing energy gained during summer for use during winter is very difficult.

Despite of all the simplifications and assumptions made, we have been able to show a principle that can be developed further with more accurate and complicated models and controllers.

Appendices

Appendix A

m files

A.1 main.m

```

global latitude;
global longitude;
%Geografic coordinates
%Max 4 desimal resolution
latitude='78.11';
longitude ='15.34';
metersAboveSea='40';

%Tilt and orientation of solar collector
tilt = 45; %in degrees, zero degrees is flat
orientation = 90; %in degrees (0=south),(90=east),(180=north) and (270=west)

%% script for collecting data

close all;
%Deleting old Error file
if (exist('Error.txt'))
    delete('Error.txt');
end

%Errorfile and written to status
global Error;
global ErrorStatus;
ErrorStatus=1;

Error = fopen('Error.txt','a');

%API version
api='1.6';

%Create URL, check URL and load in and save data
url=strcat('http://api.met.no/weatherapi/locationforecast/',api,'?lat=',latitude,';lon=',...
longitude ,';msl=',metersAboveSea);
xmlFile='weather.xml';
xmlCheck='check.xml';

[check,status] = urlwrite(url,xmlCheck);

if (status)
    delete(xmlCheck);
    urlwrite(url,xmlFile);
else
    fprintf(Error,'Could not read from URL \n');
    ErrorStatus = 0;
end

%Create DOM list from XML

DOMnode = xmlread('modelvalidation.xml');
listMeta = DOMnode.getElementsByTagName('meta');


%Get length of list
global from;
```

```

from = char(listMeta.item(0).item(1).getAttribute('from'));
to = char(listMeta.item(0).item(1).getAttribute('to'));

temp = regexp(from, 'T', 'split');
temp = temp(2);
temp =  regexp(char(temp), ':', 'split');
fromTime = str2num(char(temp(1)));
temp = regexp(to, 'T', 'split');
temp = temp(2);
temp =  regexp(char(temp), ':', 'split');
toTime = str2num(char(temp(1)));

clear temp;

if (fromTime < toTime)
    extraLength = toTime - fromTime;
elseif (fromTime > toTime)
    extraLength = toTime+24 - fromTime;
else
    extraLength = 0;
end

global totalLength;
totalLength = 49+extraLength;

%Get element list
global allListItems;
allListItems = DOMnode.getElementsByTagName('location');
%%

%Data info

%Available Hour Data in Norway

%fog : unit = "percent"
%pressure : unit="hPa"
%temperatureProbability : #unit="probabilitycode" (Only available every six'th hour)
%highClouds : unit = "percent"
%windDirection : unit = "deg"
%mediumClouds unit = "percent"
%windSpeed : unit = "mps"
%cloudiness : unit percent
%lowClouds : unit="percent"
%windProbability : #unit="probabilitycode" (Only available every six'th hour)
%humidity : unit="percent"
%temperature : unit="celcius"

%# probabilitycode : certainty values.

%0 : [90% , 100%]
%1 : [50% , 90%>
%2 : [ 0% , 50%>

%%

%Check if coordinates is inside boundary
name = char(listMeta.item(0).item(1).getAttribute('name'));

if (strcmp(name,'YR'))

%start getWeatherData

global cloudiness;
global temperature;

temperature = getWeatherData('temperature');
cloudiness = getWeatherData('cloudiness');
%end getWeatherData



%start getRadiation
radiation = getRadiationPerM2(tilt,orientation);  %getRadiation(tiltAngle [degrees], Orientation [degrees])
%end getRadiation

else
    fprintf(Error,'Outside coordinate boundary \n');
    ErrorStatus = 0;
end

%closing Error.txt
fclose(Error);

```

```
%Deleting Error.txt if it's empty or exit run of script if not empty
if (ErrorStatus)
    delete('Error.txt');
else
    return;
end


mpcInit;
```

A.2 getWeatherData.m

```
function weatherData = getWeatherData(dataType)

    %global values

    global totalLength;
    global allListItems;
    global Error;
    global ErrorStatus;

    %end global values

    switch dataType

        case 'fog'
            normalListnumber = 1;
            extendedListnumber = 1;
            data = 'percent';

        case 'pressure'
            normalListnumber = 3;
            extendedListnumber = 3;
            data = 'value';

        case 'temperatureProbability'
            normalListnumber = 5;
            extendedListnumber = 5;
            data = 'value';

        case 'highClouds'
            normalListnumber = 5;
            extendedListnumber = 7;
            data = 'percent';

        case 'windDirection'
            normalListnumber = 7;
            extendedListnumber = 9;
            data = 'deg';

        case 'mediumClouds'
            normalListnumber = 9;
            extendedListnumber = 11;
            data = 'percent';

        case 'windSpeed'
            normalListnumber = 11;
            extendedListnumber = 13;
            data = 'mps';

        case 'cloudiness'
            normalListnumber = 13;
            extendedListnumber = 15;
            data = 'percent';

        case 'lowClouds'
            normalListnumber = 15;
            extendedListnumber = 17;
            data = 'percent';

        case 'windProbability'
            normalListnumber = 19;
            extendedListnumber = 19;
            data = 'value';

        case 'humidity'
            normalListnumber = 17;
            extendedListnumber = 21;
            data = 'value';

        case 'temperature'
            normalListnumber = 19;
            extendedListnumber = 23;
            data = 'value';

    end


    delta = 5;
    forLength1 = (totalLength-5)*delta;
    delta=4;
    forLength2Start = forLength1+delta;
    forLength2Stop = forLength1+(3*delta);
    clear delta;
    l=1;



    for k=0:5:forLength1

        thisListItem = allListItems.item(k);

        if (strcmp(dataType,thisListItem.item(normalListnumber).getTagName))

            valueList(l) = str2num(thisListItem.item(normalListnumber).getAttribute(data));
            l=l+1;

        elseif (strcmp(dataType,thisListItem.item(extendedListnumber).getTagName))

            valueList(l) = str2num(thisListItem.item(extendedListnumber).getAttribute(data));
            l=l+1;

        elseif (strcmp(dataType,'temperatureProbability')||strcmp(dataType,'windProbability'))


        else
            fprintf(Error,'Problem with ListIndecies in loop 1 for %s \n',dataType);
            ErrorStatus = 0;
        end

    end

    for k = forLength2Start:4:forLength2Stop

        thisListItem = allListItems.item(k);

        if (strcmp(dataType,thisListItem.item(normalListnumber).getTagName))

            valueList(l) = str2num(thisListItem.item(normalListnumber).getAttribute(data));
            l=l+1;

        elseif (strcmp(dataType,thisListItem.item(extendedListnumber).getTagName))

            valueList(l) = str2num(thisListItem.item(extendedListnumber).getAttribute(data));
            l=l+1;

        elseif (strcmp(dataType,'temperatureProbability')||strcmp(dataType,'windProbability'))

        else

            fprintf(Error,'Problem with ListIndecies in loop 2 for %s \n',dataType);
            ErrorStatus = 0;
        end


    end


        k=k+3;
        thisListItem = allListItems.item(k);

        if (strcmp(dataType,thisListItem.item(normalListnumber).getTagName))

            valueList(l) = str2num(thisListItem.item(normalListnumber).getAttribute(data));


        elseif (strcmp(dataType,thisListItem.item(extendedListnumber).getTagName))

            valueList(l) = str2num(thisListItem.item(extendedListnumber).getAttribute(data));

        elseif (strcmp(dataType,'temperatureProbability')||strcmp(dataType,'windProbability'))


        else

            fprintf(Error,'Problem with ListIndecies in loop 3 for %s \n',dataType);
            ErrorStatus = 0;
        end



        weatherData = valueList;
end
```

A.3 getRadiationPerM2.m

```
function radiation = getRadiationPerM2(tiltAngle, orientation)

    %global values

    global from;
    global totalLength;
    global latitude;
    global longitude;
    global cloudiness;

    %end global values

    latitudeNum = str2num(latitude);
    longitudeNum = str2num(longitude);

    %days in the different months of year starting with january. We neglect
    %leap year for simplification

    month = [31 28 31 30 31 30 31 31 30 31 30 31];


    temp = regexp(from, 'T', 'split');
    temp1 = temp(1);
    temp2 = temp(2);
    fromDate = regexp(char(temp1), '-', 'split');
    fromClock = regexp(char(temp2), ':', 'split');

    fromMonth = str2num(char(fromDate(2)));
    fromDay = str2num(char(fromDate(3)));
    firstWholeHourGMT = str2num(char(fromClock(1)));

    clear temp1 temp2;

    % Calculation of which day of the year the prediction starts

    daynumberStart = sum(month(1:fromMonth-1)) + fromDay;


    %omega is angle in degree between south and the horizontal projection of the sun beam.

    delta = 15;

    if(firstWholeHourGMT < 12)

        omegaStart = 360-(12-firstWholeHourGMT)*delta+longitudeNum;

    elseif(firstWholeHourGMT > 12)

        omegaStart = (firstWholeHourGMT-12)*delta + longitudeNum;

    else
        omegaStart = longitudeNum;
    end


    omega = [omegaStart:15:omegaStart+15*totalLength];



    %Checking how many hours left of the day to set right daynumber

    hoursLeftOfDay = 24-firstWholeHourGMT; %variable between 1 and 24;
    k = totalLength-24-hoursLeftOfDay;

    if (k>24)

        l = k-24;
        extraDays = [zeros(1,hoursLeftOfDay) ones(1,24) 2*ones(1,24) 3*ones(1,l)];


    else
        extraDays = [zeros(1,hoursLeftOfDay) ones(1,24) 2*ones(1,k)];

    end

    clear k l;

    %Calculating declination vector

    for k=1:totalLength

```

```
        declinationVector(k) = 23.45*sind(360*(284+daynumberStart+extraDays(k))/365);

    end

    clear k;

    %Calculating cosThetal which is angle divider through atmosphere. Since
    %atmosphere has no tilt beta = 0

    for k=1:totalLength
        cosAtmosphereAngle(k) =  sind(declinationVector(k))*sind(latitudeNum) +...
        cosd(declinationVector(k))*cosd(latitudeNum)*cosd(omega(k));

    end

    %Calculating cosFixedPlaneAngle which is projection angle between fixed normal line of plane and
    %sun beam

     for k=1:totalLength

         cosFixedPlaneAngle(k) = sind(declinationVector(k))*sind(latitudeNum)*cosd(tiltAngle)-...
         sind(declinationVector(k))*cosd(latitudeNum)*sind(tiltAngle)*cosd(orientation)+...
         cosd(declinationVector(k))*cosd(latitudeNum)*cosd(tiltAngle)*cosd(omega(k))+...
         cosd(declinationVector(k))*sind(latitudeNum)*sind(tiltAngle)*cosd(orientation)*cosd(omega(k))+...
         cosd(declinationVector(k))*sind(tiltAngle)*sind(orientation)*sind(omega(k));


     end



     %Start of Calcualting Radiation per m^2

     ISol = 1360;  % The solar Constant ISol = 1360 W/m^2

     a=0.15;       %Constants approximated for Oslo
     b=0.14;
     c=0.1;

     for k=1:totalLength

         %radiation will be set to zero if any of these angles are negative
         if(cosAtmosphereAngle(k) <=0 || cosFixedPlaneAngle(k)<=0 )

             valueList(k) = 0;   %radiation

         else
             %radiation and reduction because of cloudiness. Approximated
             %reduction is 9.4 W/m^2%

             valueList(k) =  (ISol*(1-(a/sqrt(cosAtmosphereAngle(k))+((1-a/sqrt(cosAtmosphereAngle(k)))...
             *(b/(cosAtmosphereAngle(k)+c)))))-(9.4*cloudiness(k)))*cosFixedPlaneAngle(k);


             if(valueList(k) <= 0)

                 valueList(k) = 0;      %can't have negative radiation
             end
         end
     end

     %end of Calcualting Radiation if clear skies

     radiation = valueList;


end
```

A.4 getSolarTempOut.m

```

function solarTempOut = getSolarTempOut(litrePerSek,radiationPerM2,T0)

% function that takes in flowrate in to collector, radiation and
% outside temperature

    tIn = 5; %[Celsius] assuming constant temperature in to collector

    if (radiationPerM2 <=0) % No point calculating anything if radiation is zero or less
        solarTempOut=tIn;
        return
    end

    if (litrePerSek == 0)
        solarTempOut = 0;
        return
    end


    cw = 4184; %[J/Celsius*kg] Heat capacity of water

    %CollectorParameters
    eta0 = 0.8;
    k1 = 3;    %[W/m^2Celsius]
    k2 = 0.03; %[W/m^2Celsius^2]
    collectorArea = 30; %[m^2]
    %end CollectorParameters



    litrePerSekPerM2 = litrePerSek/collectorArea;

    b = 0.25*k2;
    c = cw*litrePerSekPerM2+0.5*k1+0.5*k2*tIn-k2*T0;
    d = (-cw*litrePerSekPerM2+0.5*k1+0.25*k2*tIn-k2*T0)*tIn + (k2*T0-k1)*T0 - eta0*radiationPerM2;



    %have to check that eta stays innside it's boundry. It must be between
    %zero and eta0. an outside temperature varmer than water temperature
    %will not give positive contribution to eta

    temp1 = (-c/(2*b))+(sqrt(c^2-4*b*d))/(2*b);   %temporary temperature

    temp2 = eta0-(k1/radiationPerM2)*(((temp1+tIn)/2)-T0)-(k2/radiationPerM2)*(((temp1+tIn)/2)-T0)^2;   %temporary eta

    if (((temp1+tIn)/2)< T0 || (c^2-4*b*d)< 0 )       %keeps (tW-T0)/I positive and checks that tOut is real
       eta = 0.8;
       tOut1 = (eta*radiationPerM2)/(cw*litrePerSekPerM2) + tIn;

    elseif (temp2<0)

        tOut1 = tIn;

    else

        tOut1 = temp1;

    end

    if (tOut1 >= 100)

        tOut1 = 100;
    end

    solarTempOut=tOut1;
end


```

A.5 mpcInit.m

```
%% initiating constants

mr = 50; %[kg]
cw = 4184; %[j/(kg celsius)]
g1 = 100; %[w/celsius]
g2 = 500; %[w/celsius]
g3 = 130; %[w/celsius]
mRoomCroom = 1.8e7; %[j/celsius]


%% getting global data
global temperature;
global totalLength;


%% initial linearization points

%states
x1 = 70; %[degrees celsius]
x2 = 750; %[litres]
x4 = 20; % [degrees celsius]

%disturbances
v1 = 70; %[degrees celsius]
v2 = 0; %[litres/s]
v3 = 20; %[degrees celsius] [constant]
v4 = sum(temperature)/totalLength; %[degrees celsius]


%% Finding rest of nominal values
% solving by setting all nonlinear derrivatives to zero and back
% substitution

x3 = x4*(1+(g3/g2)) - (g3/g2)*v4;
u3 = (g2*(x3-x4)) / (cw*(x1-x3));
u2 = u3;
u1 = ((-cw*u2*v1 + cw*x1*(u3+v2) + g1*(x1-v3)))/1000;

%inserting in vectors
xNom = [x1 x2 x3 x4]';
uNom = [u1 u2 u3]';
vNom = [v1 v2 v3 v4]';

%% Obtaining matrix components

a11 = (-1/x2)*(u3+v2+(g1/cw));
a12 = (-1/x2^2)*(((1000*u1)/cw)+(u2*v1)-(x1*u3)-(x1*v2)-(g1/cw)*(x1-v3));
a31 = u3/mr;
a33 = -((u3/mr)+(g2/(mr*cw)));
a34 = g2/(mr*cw);
a43 = g2/(mRoomCroom);
a44 = -((g2+g3)/mRoomCroom);

b11 = 1000/(cw*x2);
b12 = v1/x2;
b13 = -(x1/x2);
b22 = 1;
b23 = -1;
b33 = (x1-x3)/mr;

c11 = u2/x2;
c12 = -(x1/x2);
c13 = g1/(cw*x2);
c22 = -1;
c44 = g3/mRoomCroom;

A = [a11 a12 0 0 ; 0 0 0 0 ; a31 0 a33 a34; 0 0 a43 a44];
B = [b11 b12 b13 ; 0 b22 b23 ; 0 0 b33 ; 0 0 0];
C = [c11 c12 c13 0 ; 0 c22 0 0 ; 0 0 0 0 ; 0 0 0 c44];

temp = -A*xNom-B*uNom-C*vNom;
temp1 = temp(1);
temp2 = temp(2);
temp3 = temp(3);
temp4 = temp(4);

W = [temp1 0 0 0 ; 0 temp2 0 0 ; 0 0 temp3 0 ; 0 0 0 temp4];

clear temp temp1 temp2 temp3 temp4;

D = [B C W];
```

```
G = eye(4);
H = zeros(4,11);

%creating discrete model
ts = 900;
model = c2d(ss(A,D,G,H),ts);

%clear D G H;

%Assign types of I/O variables
model=setmpcsignals(model,'MV',[1 2 3],'MD',[4 6 7 8 9 10 11],'UD',[5]);



%Assign names to I/O variables

model.InputName  = {'u1';'u2';'u3';'v1';'v2';'v3';'v4';'v5';'v6';'v7';'v8'};
model.Outputname = {'x1';'x2';'x3';'x4'};


%Define I/O constraints and units
clear InputSpecs Outputspecs;

InputSpecs(1) = struct('Min',0,'Max',10,'RateMin',-Inf,'Ratemax',Inf,'Units','kW');
InputSpecs(2) = struct('Min',0,'Max',0.1,'RateMin',-Inf,'Ratemax',Inf,'Units','liters per second');
InputSpecs(3) = struct('Min',0,'Max',0.1,'RateMin',-Inf,'Ratemax',Inf,'Units','liters per second');

Outputspecs(1) = struct('Min',70,'Max',100,'Units','degrees celsius');
Outputspecs(2) = struct('Min',0,'Max',1000,'Units','kg');
Outputspecs(3) = struct('Min',0,'Max',100,'Units','degrees celsius');
Outputspecs(4) = struct('Min',0,'Max',100,'Units','degrees celsius');



%Define prediction and control horizon and set up the MPC object
p = 96;
m = 8;

Weights = struct('ManipulatedVariables',[10 100 100],'ManipulatedVariablesRate',[0 0 0],...
'OutputVariables',[0 0 0 1000]);

MyMPC = mpc(model,ts,p,m,Weights,InputSpecs,Outputspecs);
setoutdist(MyMPC,'remove',[1 2 3 4]);
```

Appendix B

Simulink diagrams

B.1 Heat system

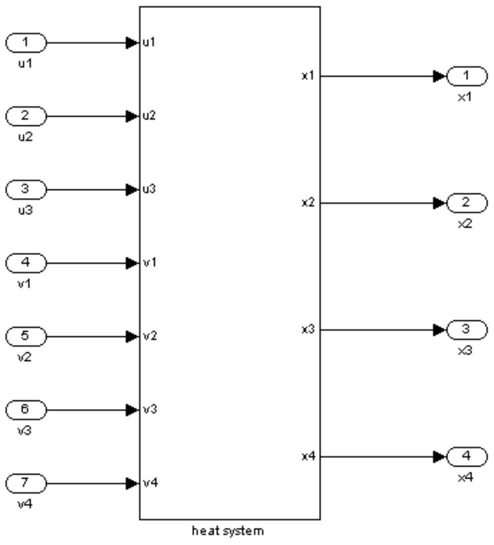

Figure B.1: Heat System Overview

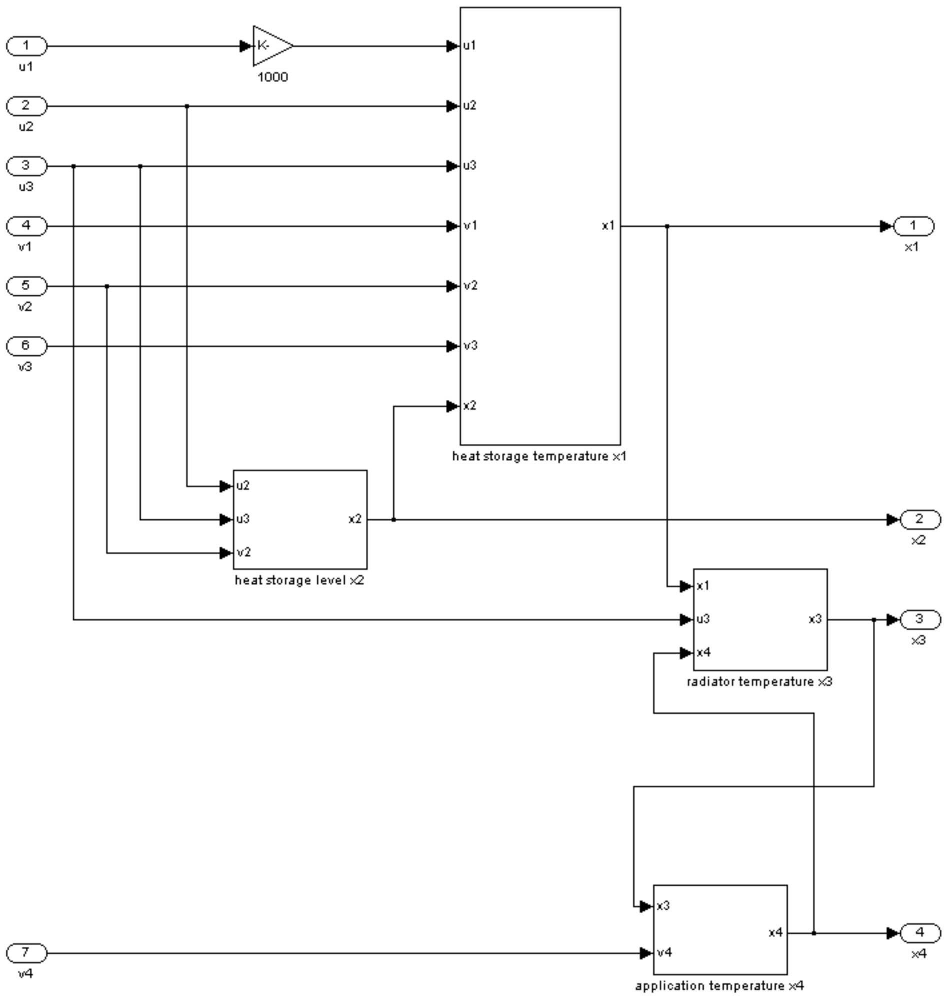

Figure B.2: Heat system

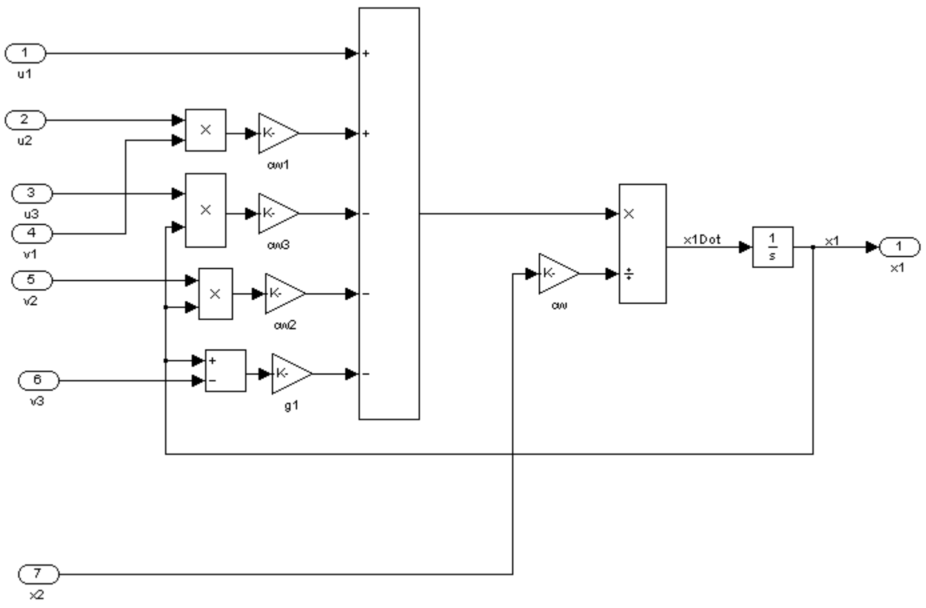

Figure B.3: Heat temperature x_1

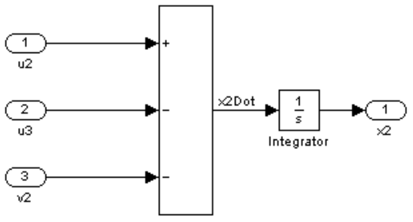

Figure B.4: Heat storage mass x_2

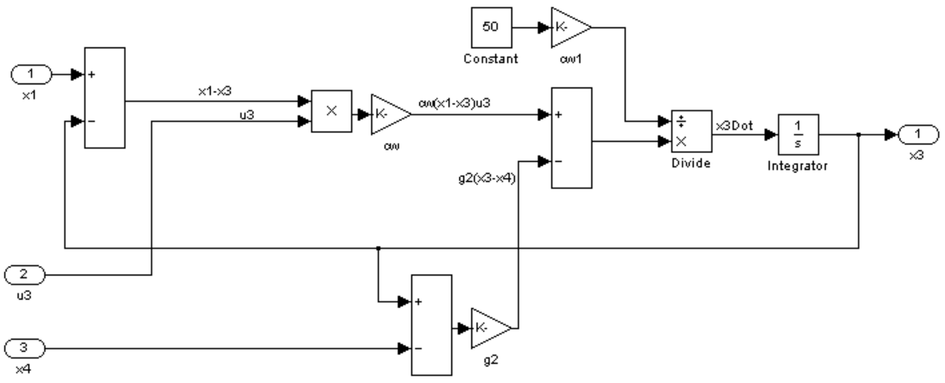

Figure B.5: Radiator temperature x_3

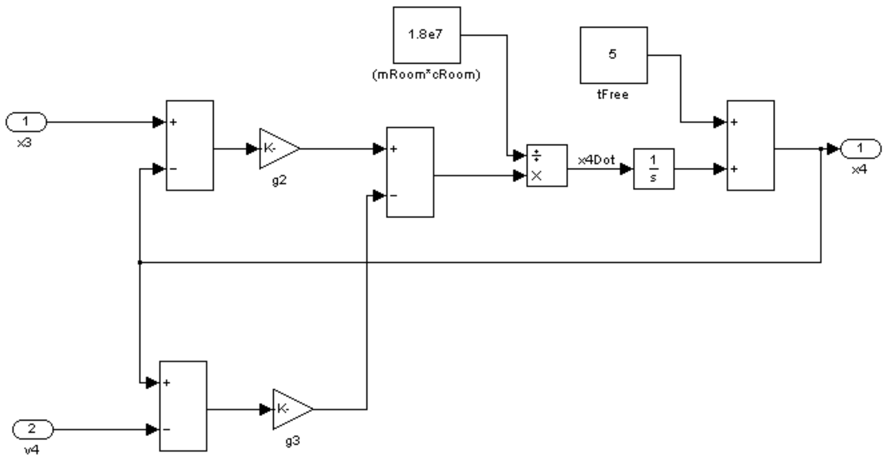

Figure B.6: Application temperature x_4

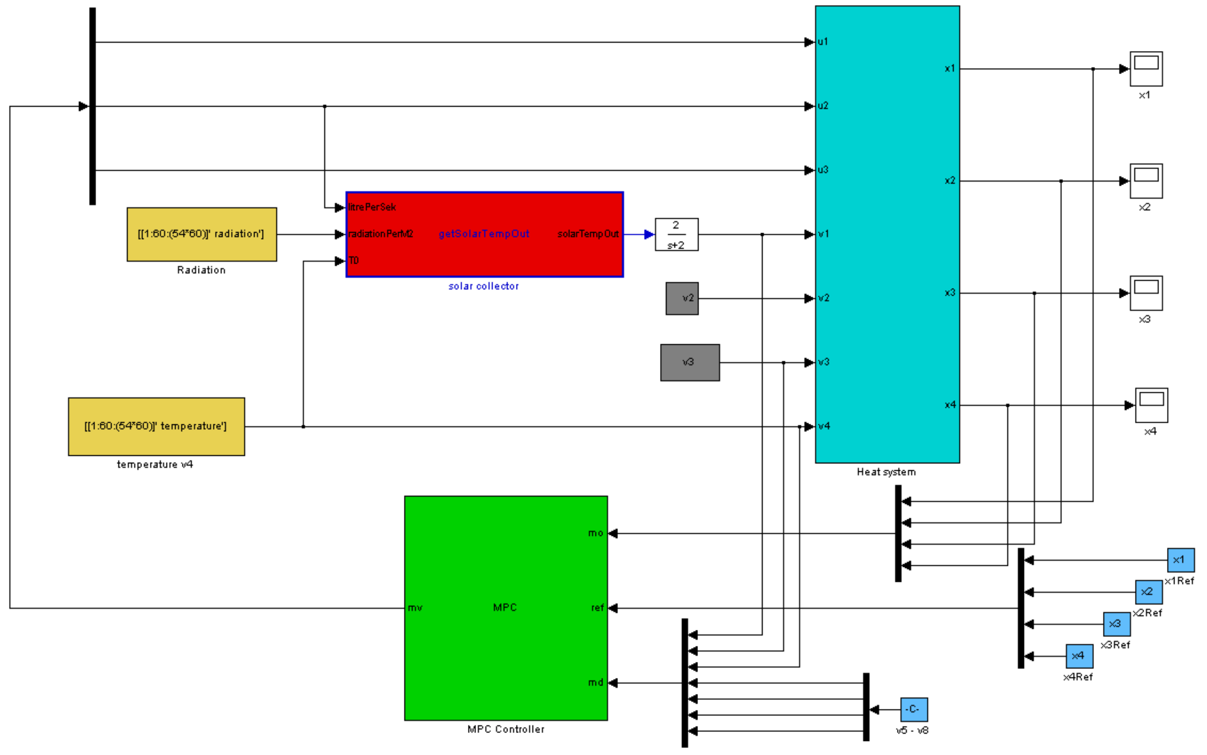

Figure B.7: Heat model with mpc control (Notice that a filter with gain 1 and time constant 0.5 seconds is added on the outport on the solar collector to avoid problems with algebraic loops when simulating.)

Bibliography

[1] Jens G. Balchen, Trond Andresen, and Bjarne A. Foss. *Reguleringsteknikk*. Institutt for teknisk kybernetikk, fifth edition, 2004.

[2] E.F Camacho and C. Bordons. *Model Predictive Control*. Springer, 2005.

[3] S.J. Frank L. Pedrotti, Leno M. Pedrotti, and Leno S. Pedrotti. *INTRODUCTION TO OPTICS*. Pearson Prentice Hall, third edition, 2007.

[4] Vaillant Group. Vaillant, 2009. http://www.vaillant.com/.

[5] Mathworks. Matlab, 2009. http://www.mathworks.com/access/helpdesk/help/helpdesk.html.

[6] Norwegian Institute of Metrology. Weather API, 2009. http://api.met.no/weatherapi/locationforecast/1.6/documentation.

[7] John Rekstad. Aventa, 2009. http://www.aventa.no.

[8] John Rekstad and Michaela Meir. *ENERGY AND PHYSICS*. Department of Physics, University of Oslo, Norway, 2008.

[9] Sintef. niprox, 2002. http://www.niprox.com/.

[10] Solelec. Solelec, 2009. http://www.solelec.lu/.