TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

LUND UNIVERSITY

Diplomarbeit

Calorimetric measurement of indoor sunshields

in Partial Fulfillment of the Requirements for the Degree of Diplom-Ingenieur under the direction of

Univ.Prof. Dr. Björn Karlsson

Bengt Hellström
and
Håkan Håkansson

Institute for Energy and Building Design

submitted to the Vienna University of Technology
by

Markus Heimberger, BSc.

Matr. Nr.: E0425837
Obernondorf 22, 3914 Waldhausen, Austria

Vienna, April 2010
Date

Signature

Acknowledgments

First of all, I would like to express my sincere gratitude to Prof. Dr. Björn Karlsson, who provided me with this opportunity to write the Master Thesis at the Department of Energy and Building Design. He not only provided me with the information I required for my work, but also gave me invaluable support and advice.

I wish to thank Bengt Helström and Håkan Håkansson who provided me with technical information on the modeling process, hardware solutions and the inspiration to solve problems as they cropped up.

I would also like to say thank you to my parents Maria and Silvester, for their great support during the years of study and during my year abroad; to Xue Yuen Lim, for proofreading my thesis; and to my sister Sandra, without whose support I would never have gone abroad and had the chance to write this thesis. I wish her all the best in her studies

Finally, I would like to thank everyone else who contributed to this thesis.

Abstract

The conservation of energy is one of the most important ways of ensuring future demand for energy can be met. The usage of air conditioning and other power-consuming luxuries are increasingly common as the average standard of living rises worldwide. Rather than wasting energy on just cooling, a better solution is available. The effective utilization of a combination of sun shields and air conditioning can save a lot of energy and money.
At the Department for Energy and Building Design (EBD), a measurement system has been developed for outdoor sunshields (e.g.: Italian and Venetian blinds); however this system is incapable of measuring indoor sunshields. Therefore the measurement system has to have real room conditions, if a curtain is installed in a room, the measurement system should not disturb the natural air circulation. In the 1990's the PASYS system was developed to measure indoor shields in a comparable way. The system described in this thesis is a further development to the PASYS system, in which only one wall facing the outside instead of five. Measurements were made and a model for the behavior of the room was developed. There are two identical rooms; one is used as a reference room without any installed sunshades, and the other is used for measurements. With this system, the disadvantage of using only one room and having to wait for the same weather conditions to measure the room with and without sunshields is eliminated, as both rooms can be measured at the same time.
Characteristics and parameter identification mechanisms are shown for the PASSYS and EBD lab rooms. Various sophisticated methods were used for the PASSYS room and a large amount of effort was spend on error correction.
A modeling process was required for parameter identification, starting with the most basic model. Two methods were evaluated - the capacity model and the time-shift model. Physical aspects that influence energy balance were added to each model step-by-step. It will be illustrated that the capacity model reached its limits and there is no longer any way of improving it further.
The pros and cons of each model are discussed, with the time-shift model left at the end to be developed further. The goal of this system is to measure the U- and g-values for different types of indoor sunshields and windows. In this part of the project, only the g-value calculation was included, then the total energy balance between the rooms were compared and the Newton-Raphson method was used to calculate the g-value by equating the difference of the energy to zero. Beside this g-vale, a few others are calculated to ensure that the calculations are reasonable and accurate.

Contents

1 Introduction

1.1 Background

Solar radiation is a reliable source of heat all year round as sunlight is always available. Because of the irradiation the incorrect dimensioning of a house can lead to excessive internal temperatures, especially when the external temperature is high. Cooling is a common solution to reducing room temperature to a moderate level, but this consumes a large amount of energy. Preventing the entry of solar radiation into the building in the first place is a better solution; while the initial investment is higher, energy consumption is greatly reduced, to the point where it is close to not having a sunshield at all. Other benefits include:

- Lower running costs
- Reduced dependence on electric power
- Less use of Freon
- Lower depreciation costs
- Better work environment due to less noise as a result of the absence of active ventilation (e.g.: air conditioning)

[8]
Since well insulated windows were invented, the construction of buildings with wholly glass facades have become increasingly common. Such designs present no problems with heating during winter but can get excessively hot during summer which requires a large amount of energy for cooling. There are two ways of preventing heat from entering the building; using indoor or outdoor sunshields. The system described in this work is designed to measure indoor sunshields under realistic conditions.
Very rarely is solar protection a consideration in the design phase of a building; it is usually only considered after excessively high temperatures have been identified as an issue. In the past there was only a big interest for office and industrial buildings in using proper sunshields, since low energy or "Passivhaus" for single family houses are more common and a lot of effort was used to reduce the energy costs for heating during winter. There should not be losing sight of the amount of cooling energy during summer. The houses are well insulated and the risk of overheating during summer is high. For such buildings an indoor solution is probably more convenient as a outdoor sunshield. A proper sun protection is unavoidable, otherwise the total annual energy per m^2 would not be below the limit for the "Passivhaus"-standard. This fast growing market makes proper indoor sunshields measurements necessary, there are a lot of different building designs, for all this it is no longer only a question of design which curtain or blind is used, also the energy aspect has to be considered.
Many manufacturers and retailers of sunshields only have rough estimations of their products' characteristics, which makes it difficult for architects to design buildings with sunshields because

they are unable to exactly determine their effect. The goal of this project is to overcome the lack of information by figuring out how well sunshields work under realistic conditions.

1.2 Energy use in buildings

Annual energy use in the building and service sector totalled 150 TWh in 1999, approximately 40 % of total energy used in Sweden. For a standard house, energy consumption totalled 4800 kWh, which is split into several areas (see Figure 1.1).

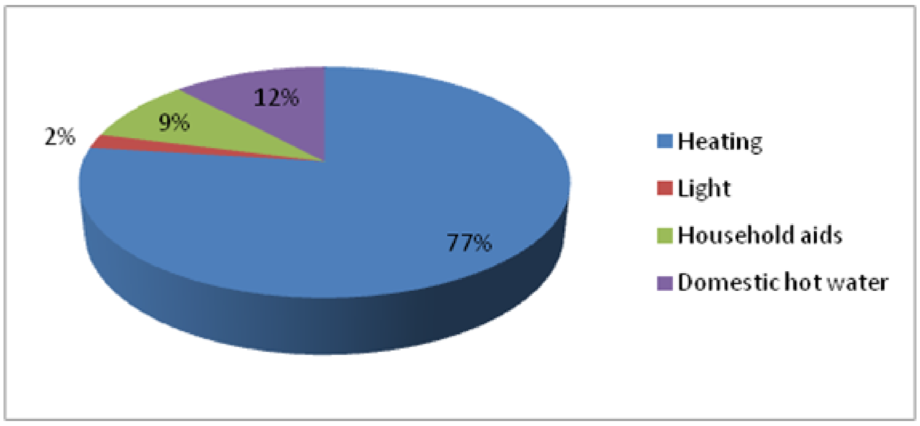


Figure 1.1: Energy consumption in private households

There has been significant progress in energy conservation since the 1970's, demonstrated by the fact that energy consumption in buildings has remained practically constant even though the heated area has increased by more than 50 %. The specific gross energy for single- or twin-dwelling buildings was about 330 − 340 kWh/(yr, m^2) in 1970; 210 − 220 kWh/(yr, m^2) in 1994; and less than 120 kWh/(yr, m^2), inclusive of all electricity consuming devices such as ventilation, heat pump etc for a "Passivhaus" today, which is the newest standard for low energy buildings. To reduce the effects of the climate change it is necessary to change the energy supply compared to the whole energy production, as much as possible must be renewable therefore a constant growing energy consumption is not adequate any more.
Approximately 77 % of energy is used for heating and there have been large efforts made in the last century to reduce this. Windows with a lower U-value were developed, home insulation was improved and the "Passivhaus" standard was established in Austria, Germany and is now increasingly common in Sweden. A "Passivhaus" uses energy-saving equipment to reduce the amount of energy used to heat the building during winter, and can achieve an annual heating power usage of approximately 15 kWh/(yr, m^2). In general, countries that experience more extreme temperatures in summer and winter are more likely to have overheating issues. This is because the airtight construction and air-air heat exchanger system used to keep a building warm during winter does not work well at dissipating heat during the summer. It is not possible to open a window or increase the air ventilation over the dimension limit to keep the annual energy consumption low, a large cooling system cannot be used and blinds are the only solution to keeping the indoor temperature at a cozy level ($\approx$ 22 °C).
While saving energy is always important, the type of energy being saved (e.g.: hydroelectricity, biogas, oil or gas) should be considered as each one would have a varying impact on the

environment. The distribution of the energy will be one of the biggest problems, in countries like Sweden where electricity is cheap, there is less sense for saving and house heating is realize as electric heating. The blackout's in the last years in Italy and USA only highlighted the problem, some grids work most of the time at the 100 % limit or even above, if a small error occur, like a grid is cut and there is no energy transport possible any more, another grid is overloaded and the security system shut the whole area down. In the past those errors mostly occurred during wintertime when a lot of energy for heating was needed, if the energy for cooling during summer is increasing and reach the same amount as the heating energy, these problems can happen in summer too. Therefore a cooling way without using energy should be used like blinds or curtains. While the increasing demand for electricity is also due to other factors such as the increased use of electrical equipment in the household and office buildings (e.g.: TV's, photocopiers, computers etc.), they only make up a minor part of total energy demand; therefore a reduction in energy usage in this area would only have a minor effect on total energy demand. As a result, energy-reduction efforts should be targeted at activities responsible for a large chunk of total energy demand, such as cooling. The split of energy usage among different activities in Figure 1.1 is valid for countries such as Sweden and Austria.[1]

1.3 Goals and limitations

This system has been designed and built to measure indoor sunshields such as curtains and blinds in conditions as realistic as possible. A lot of effort during the design stage was used to provide a natural air circulation, this should not be disturbed by the installed ventilation system for cooling or heating of the room, depending on the actual room temperature. With a system for outdoor shields it is not possible to measure the behavior in realistic conditions, it would be possible to measure the g- and U-value, it wouldn't be possible to consider that the shields heat up and change the "natural" air circulation and this change the behavior of the room. Even a blind-window or curtain-window combination has a different g- and U-value as the product of both systems at ones. Such parameters are hard to calculate and simulate, therefore a blind temperature must be estimated, the blind work as a long wave radiator, with this value and the spectral range of the window the total U-value could be calculated, a regressive calculation would be necessary. That's a big advantage of the measurement system, it take care of all this effects.
As mentioned in the beginning of this chapter, a lot of effort was put in to ensuring that the natural circulation of air was not disturbed, and that the room temperature was constant in all weather conditions. This is difficult, as realistically the temperature cannot be constant under all weather conditions across all levels of the room, due to the following reasons. If there is solar irradiation and the installed cooling system works, cold air immediately goes down and cools the lower levels of the room by a few degrees; if the cooling system is not designed this way, the general room temperature will get too hot. On the other hand, if it is cold outside and heating is required to keep the room temperature within an acceptable range, warm air rises and the top levels of the room will reach above average temperatures. The stratification phenomenon is unavoidable especially if there are two different conditions - (e.g.: cooling and heating). This occurs in all rooms, even though it is not noticed in most cases due to temperatures only being measured at one level in each room and the variance in temperature not being obvious as long

[1] `http://www.ebd.lth.se/fileadmin/energi_byggnadsdesign/images/Personal/Johan_S/Energy_20efficient_20household_20appliances_01.pdf`

as they are not too large.
The temperature in the test room is measured at three different levels (ground, middle, top) and the arithmetic average is used as room temperature. As acceptable setup, values of the controlling system and positions of the temperature sensors (it is a different system as the measurement system, see chapter 3) where used at which the arithmetic room temperature was almost constant under all conditions. If the suns heats up the air, which results in it rising to the top of the room (with the top temperature sensor reading this higher value as the average), the cooling system should cool the air at the lower levels in the same manner (with the lower temperature sensor reading this lower value as the average), resulting in the same arithmetic value under all conditions, and vice versa. This control technique has an ideal side effect, with the sun heating up the dark floor hence storing a large amount of energy, while the cooling reduces the energy from the floor.

2 State of the art (PASSYS)

2.1 Introduction

The European commission started the PASSYS project (Passive Solar Systems and Component Testing) in 1985 with the aim of testing passive solar components and developing calculation methods. The PASLINK Network (formed in 1986) emerged from PASSYS, with the fundamental aim of developing and improving methods for obtaining the thermal and solar properties of building/sunshade components. The PASSYS standardized test cells are used for measurements under realistic conditions with a controllable test environment, but it was determined that the measurements should also be obtainable under different and especially dynamic outdoor conditions.
As a result, a special room was developed (Figure 2.1). 35 of these rooms were built in 10 countries (Denmark, Germany, Belgium, France, Italy, Greece, Netherlands, Spain, Portugal and the UK) across Europe. Various building facades were utilised, and the thermal properties determined by outdoor tests at different locations across Europe. While the evaluation of the solar gain factor (g-value) of passive solar components was the main purpose, but it was also possible to measure the U-value and thermal capacity. The test cells provided a well controlled environment in a realistic room size without occupancy effects. At the beginning of PASSYS the test methodology were based on steady state evaluations, but as the project progressed it became clear, that dynamic testing and analysis methods were required for high quality performance in real climates. [1],[11]

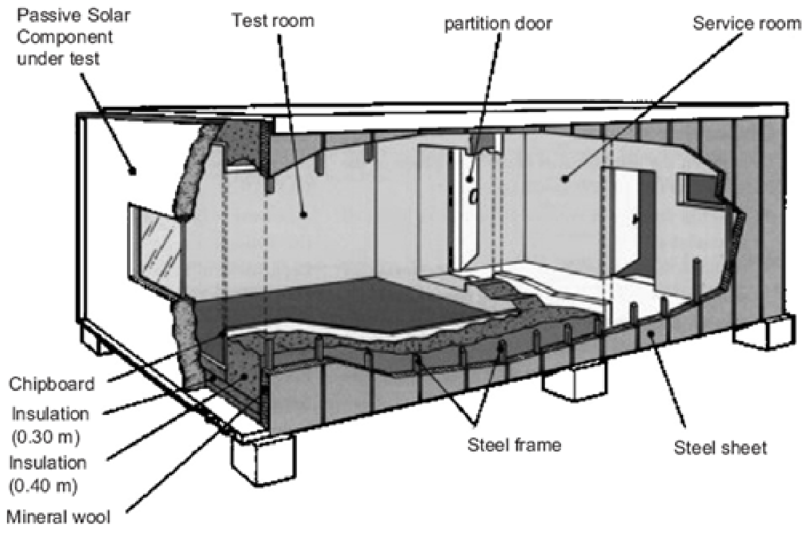


Figure 2.1: Room construction of PASSYS system [1]

In February 1994, at the end of the PASSYS project, a European Economic Interest Grouping (E.E.I.G.) PASLINK was founded. Its goal was quality assurance and improvement of the test methods. To reduce the test duration and improve the measurement accuracy, several equipments were developed, e.g. the pseudo adiabatic shell (PAS) developed by Belgian Building Research Institutes (BBRI), heat flux sensitive tiles suggested by TN0 Building and Construction Research, and several arrangements of mobile cold boxes applied at ITW, TN0 and BBRI. With these tools, test periods of around 2 weeks are feasible.
[6]

2.2 Detailed room description

The main features of this room design are:

- controllable climate within the room
- realistic room size
- the room is located in a real outdoor climate
- two rooms (not at each place)
- not occupied
- well-insulated, exchangeable south wall

Each PASSYS test cell has a test room of 13.8 m^2 ground area and 38 m^3 air volume, with an adjoining room to the north. The measuring and air-conditioning equipment are contained in this service room (see Figure 2.2).

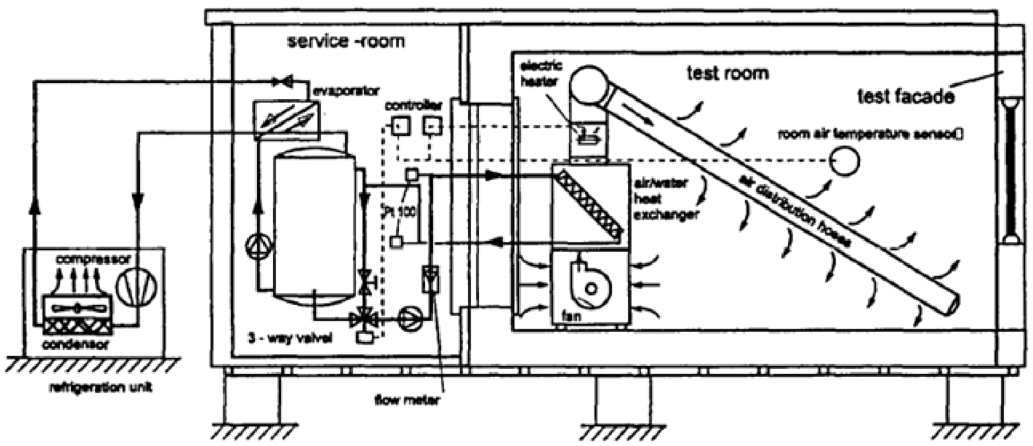


Figure 2.2: Details of room construction [6]

On the outer part of the envelope, a rigid steel frame is suited, which supports the form of the structure. Thick insulation layers of 400 mm extruded polystyrene foam and mineral wool are located between the steel beams on the inner part of the envelope. Because of this well insulation the U-value is less than 0.1 W/(m^{2}K) and the thermal conductance (U·A) is less then 8 W/K. When testing components with a U-value of 0.3 W/(m^{2}K), the ratio of the heat

flux through the test components to that through the test room envelope is about 0.3. As a result of the good insulation the time constant of the test room is about 4 days. Consequently, each part of the test procedure lasts at least 1 week. This gives a total test duration of 8 weeks according to the new calibration and component test procedures.
Inside, the walls are covered with chipboard panels, 2 mm thick metal plates, and are painted blue to ensure stable optical properties. The outer surfaces are covered with weatherproof stainless steel plates.
The test room is sealed against air infiltration (air-change rate $< 0.01\,\mathrm{l/h}$). The south aperture area, with a size of $7.6\,\mathrm{m}^2$, contains the test specimen, fixed in a removable and insulated frame. For calibration purposes, a well insulated panel of polystyrene (400 mm) instead of the measuring device is incorporated into the frame.

Experiments showed that the air-conditioning of the test room is a difficult task. The accuracy of the test results is directly affected by the precision of heat flow measurements. For heating, an electric heater was proposed to be installed in the room. The cooling task is much more difficult. For accuracy reasons, water was chosen as heat transfer medium, as the temperature and flow rate of water can be measured more accurately than those of gas (e.g.: air) and the specific heat is well known. In the test room, heating and cooling power is distributed mainly through convection; because of the low temperatures radiation only plays a minor part. Each PASSYS test cell has its own heating and cooling system, shown in Figure 2.2. This system can be divided into three subsystems with the following specifications:

- cooling system: max. cooling power at 40 °C ambient
  and 22 °C test room temperature, 2.3 kW
- heating system: max. heating power of 1.9 kW
- air distribution system: adjustable air flow rate up to $1600\,\mathrm{m}^3/\mathrm{h}$ [12]

The central data acquisition system controls the heating and cooling system. Two separate power transducers, one for the fan and the other for the electric resistances, measure the incoming heat, while the outgoing heat is determined by the measurement of inlet and outlet temperature and volume flow rate of the cooling fluid $Q = c_p \cdot \Delta T \cdot flow$. Different control strategies such as constant heating power or constant test room temperature can be used, and will be discussed in chapter 2.6.2. Temperature measurements are performed with either Pt100 or thermocouples. A standardized calibration procedure is performed by heating or cooling the test room at different power levels for the heating procedure a reference heater is used to calibrate the system.

Each test site has a standardized meteorological station, which can measure wind direction at a 10 m height, wind velocity, and the global solar irradiances on a horizontal as well as on a vertical, south facing plane. Long wave radiation is measured by a pyrgeometer, diffuse solar radiation by a pyranometer, and outdoor ambient air temperature by both a shielded and a ventilated air temperature sensor, and the relative humidity. Sensors measure the temperatures of the surfaces at 23 different positions inside as well as outside. The indoor climate in the test room is recorded using specially developed double shielded air temperature sensors at 7 precise locations within the room. Heat flux sensors are used to quantify the heat transmission flows through the test component. A hot air anemometer is used to measure the air velocity. For continuous measurement of the air infiltration rate, specific equipment for injection of tracer gas and sampling of the air at 4 locations was developed. All data are measured, computed and stored, on a common piece of hardware, which is also used for the regulation of the heating and cooling system. Up to 300 sensors can be measured with a high accuracy voltmeter at a 1 minute sample rate, with a high frequency sampling of selected sensors using a sample rate of 3 seconds being possible. All test rooms in Europe have standardized equipment, which guarantees comparable and exchangeable measurement results. [6],[12]

2.3 Further Improvements

2.3.1 The pseudo adiabatic shell (PAS)

There are two problems, the low heat flux ratio and the large time constant, both of which can be overcome by inserting a pseudo adiabatic shell (PAS). In Figure 2.3 a vertical section of the test room with the installed PAS is shown.

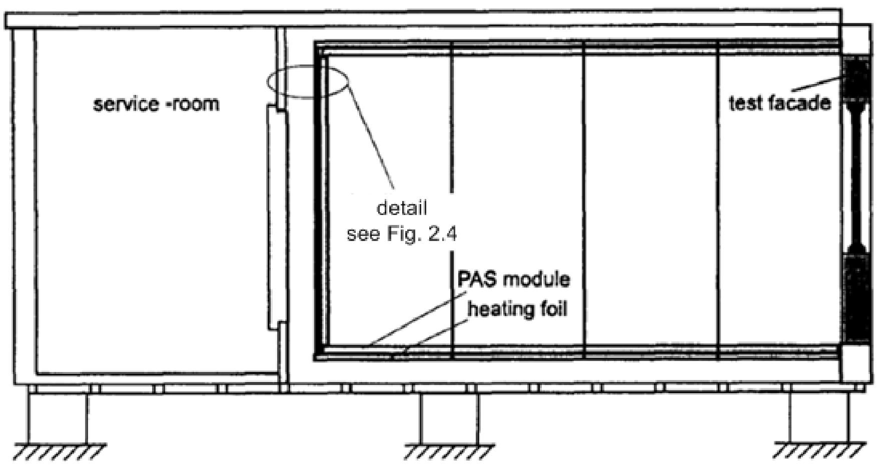


Figure 2.3: PAS module [6]

The PAS is an electric heating foil that compensates the heat loss through the test room envelope. Figure 2.4 illustrates how the heat foil is located between the outside original envelope and a 10 mm polystyrene layer.

The mean temperature difference between the aluminum plates is measured by thermopile sensors. These sensors use two unequal metals which are soldered together at one point (electrical and thermal contact) and only a thermal contact at a second point, and the measured voltage is assumed to be proportional to the temperature difference. The heating foil is controlled with the thermopile voltage a way that the resulting heat flux is almost zero. If the temperature of the inner aluminum wall is higher, the heating foil is switched on.
Calibration is still necessary before the first measurement, in order to identify thermo physical properties such as the U·A-value or thermal capacities of the test room. There is no difference between the calibration process with PAS or without, but the number of unknown parameters is decreased from about 10 to about 5. From calibration experiments, it was determined that the PAS has a U·A-value of 24.3 W/K, with an uncertainty of approximately 0.3 W/K. With

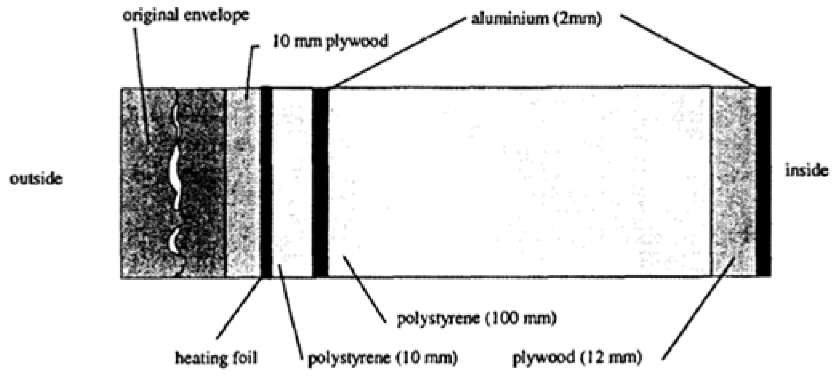


Figure 2.4: Detail of PAS module [6]

PAS, higher accuracy is obtained and the duration of the test sequence able to be reduced significantly. Instead of approximately four weeks, all the important data necessary to determine the main thermal parameters can be collected within one week. This reduction of measurement time has a big influence on costs. During the design phase of the PAS panels, several improvements were discovered, as well as within the heating foil control and the arrangement of the thermocouples.
The panels were split into four separate parts per wall, with each part having its own controlled heating foils. The temperature difference between the aluminum plates (each panel) was measured with one thermopile, which consisted of 16 thermocouples connected in series. Moreover, two thermocouples were used to measure the internal surface temperature per panel. Each PAS element was removable, in case an error occurred within a thermo element or the heating foil. If the whole PAS-system is installed as an add-on on the inner wall, a lot of space is lost; this can be overcome by dismantling parts of the existing frame/envelope before installation of the PAS-system.

2.3.2 The use of heat flux sensitive tiles (HFS)

In principle, the PASSYS cell is a calorimeter and the PAS module can be considered as a heat flux sensor. TN0 (Building and Construction Research, Delft, Netherlands), on the other hand, had the idea of substituting the PAS with a number of heat flux sensors in the form of tiles. These elements are only a few millimeters thick. The construction of the heat flux sensitive tiles is shown in Figure 2.5.

Rubber foam is applied inside the original envelope layer to embed flat cables and connectors, and fill gaps. The layer on which the tiles are mounted are made up of a plastic material with low thermal conductivity (pertinax) which is bonded to an aluminum plate as a substrate; this aluminum plate equalizes the surface temperature. A spirally wound thermopile inside each tile is placed near a corner of the tile where the heat flux is supposed to equal the average heat flux through the tile. Thermocouples measure this temperature. The small thermal inertia of the tiles allows direct measurements of rapid changes. There is a uneven flux distribution because of corner effects, thermal bridges and solar radiation effect on floor and inner walls when windows are tested. If a grid of heat flux sensors is set up, then the approximation of the flux through the test cell is more accurate. This solution has the disadvantage of the tiles being costly o manufacture and install.

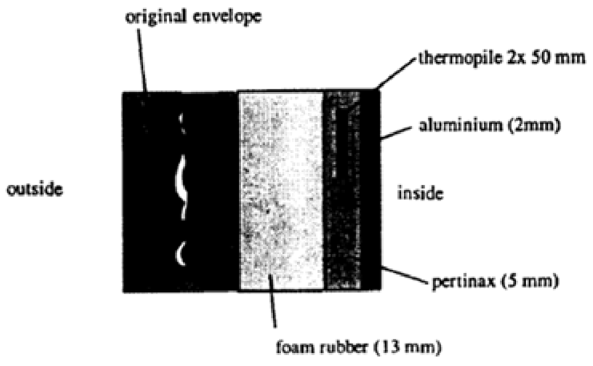


Figure 2.5: Arrangement with HFS inside of the wall [6]

2.3.3 Development of a large area heat flux sensor (LAH)

The "Institut für Thermodynamik und Wärmetechnik" (ITW) developed a large area heat flux sensor (LAH), with the main aim of finding a sensor with a good spatial resolution and relatively low construction costs. Different prototypes were analyzed and tested in the laboratory. This type of sensor differs from a conventional heat flux sensor in that the plate is passed by the heat flux vertically instead of across. For homogenous materials, isotherms are parallel to the surface, if any material with another thermal conductivity then the substrate-conductivity is inserted, a perturbation of the flux causes. The LAH use this effect of no longer parallel isotherms and the temperature gradients in any section parallel to the surface can be used as a measurement point.

The physical construction is shown in Figure 2.6 a.). Small aluminum bars with a diameter of 5 mm are inserted in rigid polystyrene plates (thickness of 10 mm), and a 2 mm aluminum plate covers the surface of the LAH. A thermopile below the two bars measure the temperature difference, within one of the thermocouple junctions placed directly under a bar (Junction 2), the other junction is located in the polystyrene between the bars (Junction 1). Figure 2.6 b.) shows the isothermal lines calculated using the finite differences calculation method.

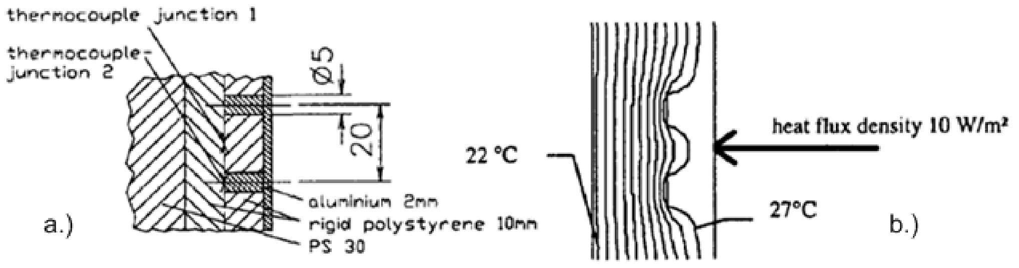


Figure 2.6: a.) Physical realization of LAH; b.) Isotherm curves of LAH [6]

The temperature profile between the two bars is shown in Figure 2.7. The temperature difference between T_1 and T_2 is the result of this construction and responsible for the output voltage. In this case, a heat flux density of 10 W/m^2 was used, with 1.1 K being the calculated temperature difference between the thermocouple junctions. This value is higher than in reality due to the thermal contact-resistances between the aluminum surface and the aluminum bars (adhesive tape).

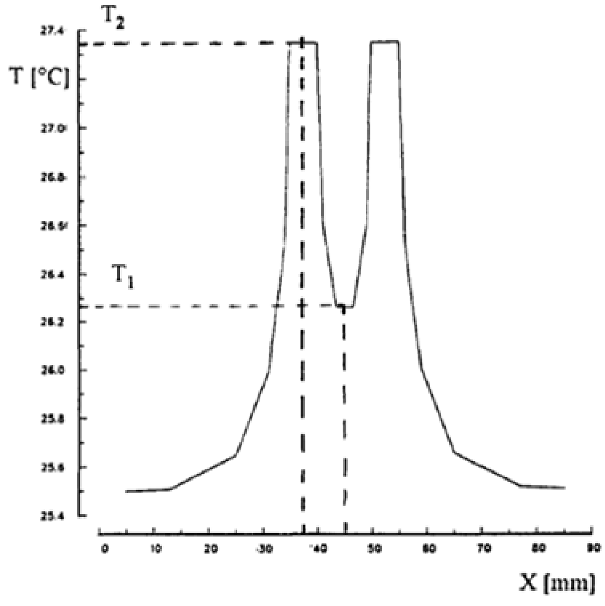


Figure 2.7: Temperature profile on the plane of the LAH [6]

For the LAH, a thermopile made from constantan wire, partly copper-coated via a galvanizing process, was used. The junctions are alternatively located below the aluminum bars and the polystyrene. The arrangement of the thermopile at one LAH element is shown in Figure 2.8.

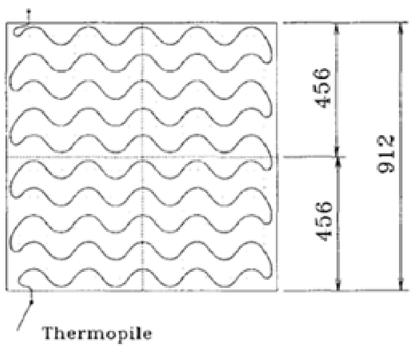


Figure 2.8: Prototype of LAH [6]

The sensitivity of the whole arrangement ($7.6\,\mathrm{m}^2$) is $2\,\mathrm{mV}$ per W/m^2, therefore all the thermopiles are connected in series. The calibration result at a mean temperature of $17.2\,°\mathrm{C}$ is shown in Figure 2.9. The LAH can be used to cover the whole inner surface of the PASSYS cells in the same way as the heat flux sensitive tiles (HFS).

The following chapters describe the room measurement system, the kind of modeling used for the PASSYS cell and the software/analysis tools.

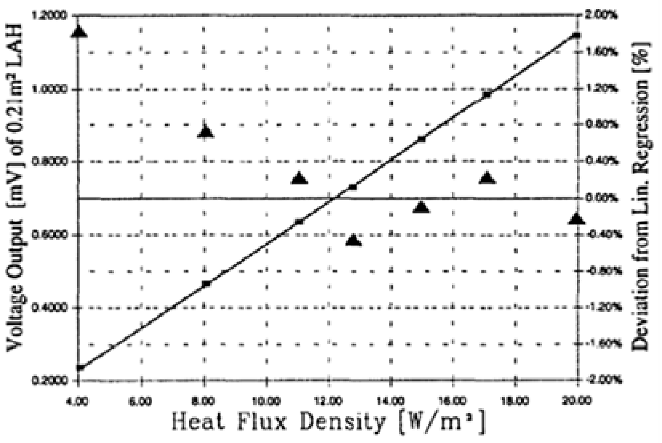


Figure 2.9: Calibration result of LAH (17.2 °C mean temperature) [6]

2.4 Measurement Methods

During the first phase (1986-1990), different test evaluation methods were tried. Depending on the simulations and results of the test, these methods were then compared. The methods ranged from simple steady state methods to more sophisticated transient parameter identification techniques. The categories are as follows:

2.4.1 Integrated absolute method

The heat loss coefficient or thermal transmission coefficient (U·A-value) and the total solar heat gain factor or solar transmittance (g·A-value) of a component are the main concerns of a building designer as these two factors have a key influence on the energy balance in rooms and buildings.
These parameters are defined as follows:

- $U \cdot A\,(\mathrm{W/K})$ gives the heat flow rate in Watts (W) in the steady state divided by the temperature difference (K) between the ambiance's on each side of the system or component.
- $g \cdot A\,(\mathrm{m}^2)$ is the heat flow rate (W) transmitted through the component to the internal environment under steady state conditions, caused by solar radiation incident on the outside surface, divided by the intensity of incident solar radiation (W/m^2) on the plane of the component. [1],[7]

It must be considered that g·A includes both effects, direct solar transmission, (e.g.: through glazings) and those due to solar radiation on opaque parts of the component structure. It can be considered as an equivalent open area that provides the same amount of solar transmitted energy as through the component (equ.(2.1)).

$$g_{open} \cdot A_{open} = g_{component} \cdot A_{component} \tag{2.1}$$

The ability of the component to accumulate heat, expressed as thermal capacity (C) and a thermal coupling to the test room, is not considered in equ.(2.1).

While many passive solar system concepts are quite simple, the involved thermal processes are very complex and the experimental determination of these properties is not straight forward. The performance depends on numerous diverse factors, such as relative area, the building's thermal mass, orientation and position of the component, to mention a few examples. The isolation of these factors and the understanding of how the component interacts with its surrounding is the primary task of experiments.
Because of the simultaneous operation of a mixture of heat transfer mechanisms, such as thermal radiation and free convection, it is not possible to measure the heat loss factor (U·A) or the solar gain (g·A) directly. Based on the measurement of the net heat flow through the building component, these quantities can be determined indirectly. The test cells are well equipped to measure this quantity.
After a long integration time the measured values are obtained to solve the steady state heat balance equ.(2.2). A physical illustration of this equation is shown in Figure 2.10.

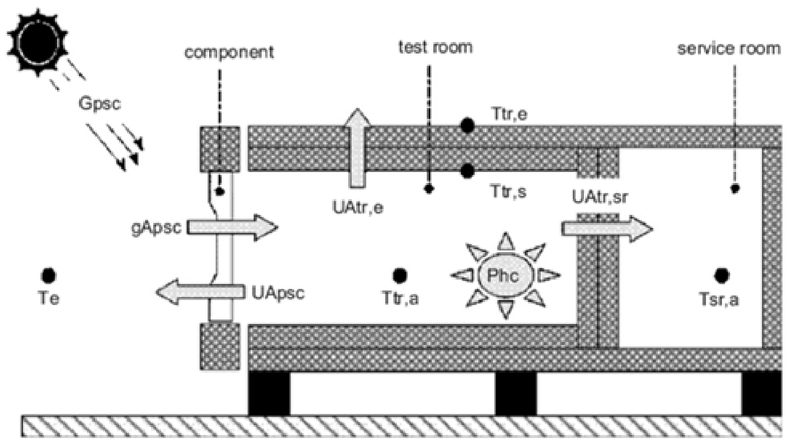


Figure 2.10: Schematic view of PASSYS cell [1]

The first term on the left side of equ.(2.2) represent the losses through the measuring object, while the second shows the irradiation energy going through. On the right side, the first term represents the losses from the test room to the outside, while the second the losses from the test room-service room. The last two parts are the heating and cooling energies. It is possible to assume a constant room temperature because it is controlled via cooling and heating, then both sides must be equal.

$$\begin{aligned} -(UA)_{psc} \cdot (T_{tr} - T_e) + (gA)_{psc} \cdot G_{psc} = \\ (UA)_{tr,e} \cdot (T_{tr} - T_{s,e}) + (UA)_{tr,sr} \cdot (T_{tr} - T_{sr}) - P_{he} + P_{co} \end{aligned} \tag{2.2}$$

All the different temperatures (e.g.: T_{tr},$T_{s,e}$, the heating (P_{he}) and cooling-energy (P_{co})) are measured. If $(UA)_{tr,e}$ and $(UA)_{tr,sr}$, are obtained via calibration and it is assumed that two or more measurements with different conditions are available, equ.(2.2) yields the values for $(UA)_{obj}$ and $(gA)_{obj}$. In steady state measurements, the temperatures of the test room and service room are kept at the same level $(T_{tr}-T_{sr})\approx 0$ and this part of equ.(2.2) becomes negligible. The advantage of this operation is that it is straightforward. The disadvantage is that there needs to be time between each measurement in order, to avoid the influence of transient effects, therefore a sufficiently long period is necessary. It has to be considered that the test cells have a time constant to the order of two days due to the high insulation level, but other heavy and heavily insulated components may also have as high inertia. Another obvious disadvantage is, that this method only yields the steady state characteristics. [1],[2],[5]

2.4.2 Integrated absolute method, with first order correction

In this variant a first order thermal capacity is added to the steady state equation as a first order correction for transient effects, e.g.:

$$\frac{C \cdot (T(t) - T(t-dt))}{dt} \tag{2.3}$$

This part is added to equ.(2.2) with C as a first order approximation of the heat capacity. The quantity dt stands for the time period of integration (time step) between two successive measuring points. From this point onwards, new terms can be added to the equation in order to further reduce the necessary duration of integration, and this brings it closer to a full dynamic parameter identification method- see further on. [5]

2.4.3 Transient comparative method

This method requires two test cells, which have to be in identical conditions, with the second cell with adiabatic components being the reference room. Both rooms are run in parallel, while one cell contains the actual test component. The indoor temperature is kept equal and constant via controlled cooling and heating in both rooms. This mean that the heat fluxes through the test cell envelopes are the same under all conditions. The difference between the cells in heating or cooling power is now the transient heat flow through the test component. It is an elegant principle, but there are still some drawbacks:

- the need of a second cell
- the question of which indoor temperature should be kept, a somehow equal weighted mean of air and surface temperature
- the possibility of inaccurate power measurements, in the case of combined heating and cooling

The direct result of this method is still only a transient heat flow through the component, which still has to be processed by a parameter identification in order to obtain the required transient and steady state properties. If equ.(2.4) is integrated over a long enough period, the parameters can be found. The right side describes the energy difference between the rooms; if T_{tr}, T_e and G_{psc} are measured, the transmission coefficient and the solar transmittance can be calculated.

$$-\left(UA\right)_{psc}\cdot\left(T_{tr}-T_{e}\right)+\left(gA\right)_{psc}\cdot G_{psc}=-\left(P_{he}-P_{he,ref}\right)+\left(P_{co}-P_{co,ref}\right) \tag{2.4}$$

2.4.4 Transient absolute method: parameter identification

A transient mathematical model of the cell and test component is assumed with the parameter identification approach. The parameters of the model, e.g. resistances, capacitances, define the dynamic, and steady-state, thermal and solar properties of the test room. First, an initial guess of the parameter values is made. The output of the actual test (for instance, the test room temperature as a function of time) is compared with the model output for the same input conditions.

In the case of the test cell, the main input variables (e.g.: outdoor temperature, solar radiation, cooling and heating power, solar radiation) are taken to be measured as functions of time. Via statistical deviation analysis of the values between the model and the measured outputs, the parameter values are progressively adjusted in order to improve the agreement. By iterating this process, the parameters for the model are found, which gives the best agreement between model calculations and measurement (Figure 2.11). This iterative process is carried out with specialized software tools. The identification technique has common characteristics with linear regression analysis.

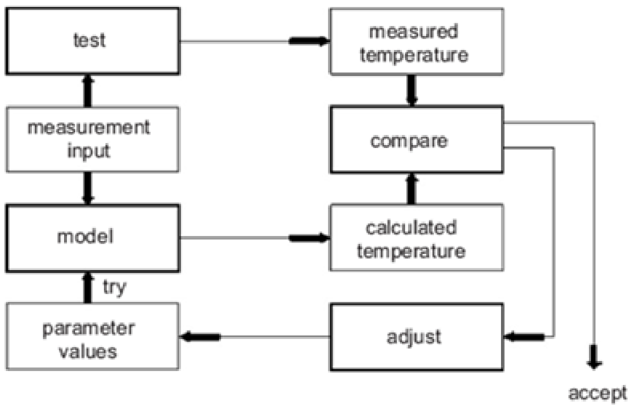


Figure 2.11: Parameter identification flow chart [1]

In this analysis, the set of parameters (coefficients) of the model (equation) is found using analytical calculations. If the model is non-linear or complex, the coefficients can only be obtained via an iterative process. This method is mainly suited to operate, under dynamic test conditions, due to the varying outdoor conditions and the large inertia of the test cell, hence giving it a great advantage. [5]

Figure 2.12 shows an example of the iteration process. This figure shows the measured output variable as a function of time, and the following calculated output behavior:

1. In the first iteration step which is based on guessed parameter values, is a strong deviation from the measured output.

2. After n iterations with improved parameter values, there is less, but still significant deviation from the measured output.
3. At the end of convergence, with best fit parameter values, there is minimum deviation from the measured output.

Figure 2.12 shows that the adjusted model, while not exact, produces the best possible fit and a satisfactory representation of the observed behavior of the tested system. With this scheme it is not necessary to wait until dynamic effects have been canceled out, unlike using the steady-state method with long integration periods. In fact dynamic effects are explicitly considered in the analysis. The consequence that, test durations can be much shorter. Parameter identification requires the right choice of software tools to obtain statistical information on the reliabilities of the identified parameter values. Measurement errors or model errors, and correlation between parameters may affect the reliability.

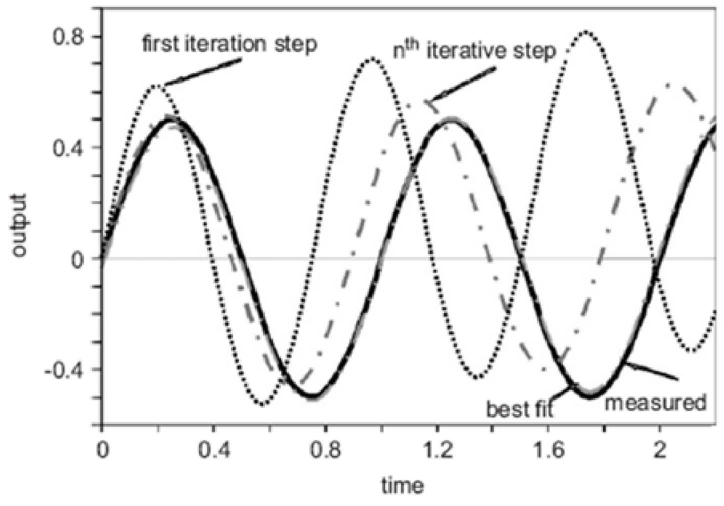


Figure 2.12: An example of the iterative parameter identification process [1]

Choosing an adequate dynamic mathematical model for the system is another majorly significant. Extensive attention has been paid to choosing a suitable model for the whole system (both the test cell and the components). [1]

2.5 Test room model

The choice of an adequate dynamic mathematical model is one of the major points of attention. Commonly, lumped parameter models have been chosen for the evaluations. Such a model can be represented as a network of thermal resistances (R) or conductance's (H=1/R) and thermal capacitances (C), with heat flows from sources (e.g.: heating power, solar radiation) connected to specific points, represented as nodes. The complexity of the model can vary between a very simple first order RC-model, to a multi-node model with a large number of resistances, capacitances and connections with external heat flow sources.
The main requirements for an appropriate model for the test room and the test component are as follows:

- The model should be able to accurately reproduce the steady-state and dynamic process, to identify parameters like U-value, g-value and time constants; and the result must be sufficiently accurate and precise.
- It should be possible to separate physical properties, i.e.: the heat loss rate through the test cell envelope should be distinct from the heat loss rate and the solar energy transmission rate through the test component. It should be possible to relate the identified physical properties to definitions per international standards.
- It should not be excessively detailed, in the sense that it leads to over-parameterization. This means that some of the parameters cannot be identified because of strong correlation with other free parameters in the model.
- Prior knowledge should be able to be utilised.
- It should allow the option of adding non-linearity such as a specific thermal resistance that varies according to temperature or with wind velocity, or solar transmittance varying according to solar and sky conditions.
- The model should be capable of reproducing the dynamic behavior of the system with sufficient accuracy.

Sufficient accuracy means that the inconsistency between measurement and calculation, should only consist of noise originating from measurement errors. The contribution by “noise” resulting from an inadequate model should be negligible.

By adopting these requirements the created model should be transparent, which means that the main elements of the heat balance in the test room and the test component should be recognized. Figure 2.13 shows the three different branches of the model between test room, service room and component used within the test cell. The branches are connected at the test room air node.

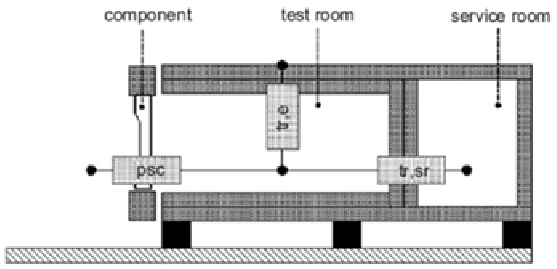


Figure 2.13: Three branches of the test cell model [1]

Figure 2.14, show the currently used RC-model which corresponds to Figure 2.13. The upper string represents the test room envelope; the middle string, the partition with the service room; and the bottom string, the test-component. It has to be considered that the identification of the large number of parameters shown in the model does not mean that each individual parameter necessarily has a physical meaning.
It is assumed in this model that part of the solar radiation (G_{psc}) penetrates via transmission through a window in the test component directly into the test room and is absorbed with solar

heat gain factor at the indoor surface of the test room envelope (node number 2). At the same time, part of the solar radiation is absorbed with a solar heat gain factor at the external surface of the test component between conductance H_{psc3} and H_{psc4}, leading to an indirect contribution to the heat balance in the test room. Of course, the practical behaviors will usually be more complicated. There will be direct thermal radiative coupling between the indoor facing surfaces of the test room envelope and the test component. The RC-network between the measured outdoor and indoor environments are actually "black box" models. The individual values are only needed to obtain a satisfactory dynamic behavior.

Also being considered are other models in the category of more general, less tailor-made mathematical descriptions. These may have advantages with respect to flexibility, being capable of dealing with a wide variety of components and/or with respect to efficiency, fewer amounts of parameters or nodes in the model. The models can be reduced in complexity, e.g.: to suppress apparent high correlation between specific individual parameters. This can be simply done by sticking specific parameters to their initial values, i.e.: removing these from the list of optimizing-parameters which are supposed to be fitted in the iteration process, for instance in the case of two resistances in series without significant thermal capacity between them. Any available a prior knowledge of the physical and structural properties of the tested system may also be used to decrease the number of free parameters.
On the other hand, it will be more difficult to anticipate in the model the expected occurrence of non-linear effects. [5]

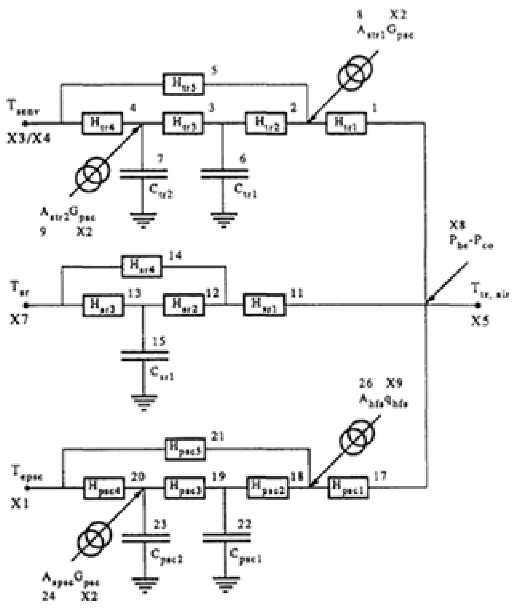


Figure 2.14: Lumped parameter model of test cell and component [5]

2.6 Control Strategy and evaluation

Now that the test can be carried out, parameter identification is needed. Sufficient temperature variation in the lowest frequency range is a key requirement for steady state properties.

“Lowest frequency range” means changes slower than a few times the largest time constant of the test room and component. “Sufficient” means that a temperature difference should be much larger than the uncertainty in indoor and outdoor temperatures. This uncertainty consists of not only the measurement error which is around 0.1 K, but also the model error. Uniform indoor and outdoor environmental temperatures are assumed while in reality the real indoor and outdoor temperatures are usually more complex. The outdoor temperature is a mixture of sky, ground and local air temperature; the indoor temperature is a mixture of surface and air temperatures. The model error can easily be in the order of 1 K. The indoor temperature is usually highly correlated to external environmental factors such as solar radiation, therefore it is not possible to get a clear separation between the heat loss, governed by the temperature difference, and the solar gain, governed by the radiation intensity. De-coupling can be achieved by providing significantly different levels of internal temperature, a sufficiently long period (a few days) with high power in the test room and a period of similar length with low power, and chosen power levels to yield a temperature variation in the order of at least 10 K (20 K is preferred).
For the evaluation of dynamic characteristics, it is necessary that variations cover the range of frequencies corresponding with the range of characteristic times within the system. In the case of the test cell and component, the corresponding range is from 20 minutes up to 50 hours.
Finally, in order to isolate the heat balance characteristics from components in the test cell, it is necessary for the heat flow through the test room envelope to be determined. To achieve this, prior calibration tests are carried out.

2.6.1 Calibration

The actual tests are preceded by a calibration of the test cell using a homogeneous, well insulated opaque calibration panel (typically 400 mm of expanded polystyrene sandwiched between two layers of plywood) mounted in the test cell aperture. With this wall, a full heating or cooling test sequence is carried out while the heat flow through the calibration panel with a heat flux meter on its inside surface is measured. Since the heating power to the test room is known, the thermal characteristics of the test cell envelope can be determined. For parameter identification, the model shown in Figure 2.14 can be used, which yields the parameters of the test room and partition of the service room. To exclude the calibration wall from the model, this part of the model is replaced by the output of the heat flux sensor on the inner surface of the calibration wall. Figure 2.15 shows the corresponding model for the calibration of the test room parameters.

During the evaluation of the actual test, the parameters in the model representing the test room envelope and the partition to the service room are fixed.
It has to be considered when analyzing test results that the interaction between component and test room may complicate the evaluation.

2.6.2 The test sequence

To obtain the intended effect on the test room temperature, different power levels are needed. There is a choice between temperature or power levels. In the case of power levels, changes in the levels lead to a less rapid response by the system than in the case of temperature set points. On the other hand, due to interaction of the dynamics of the heating or cooling system,

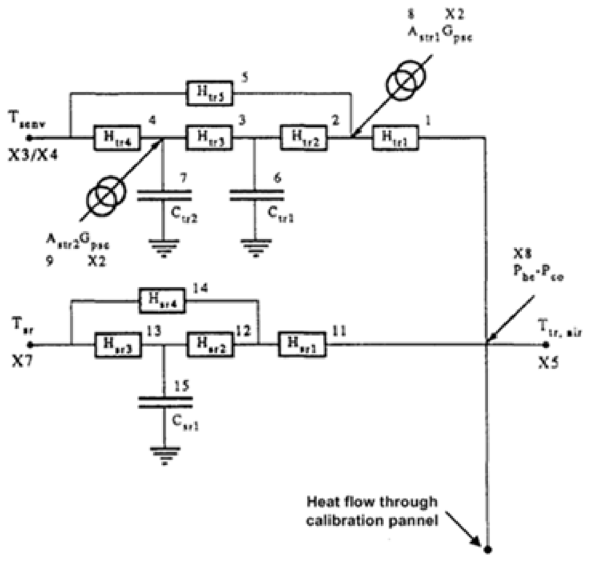


Figure 2.15: Lumped parameter model with calibration wall [5]

a temperature control may lead to a biased identification of the dynamic characteristics. For this reason, power control is chosen whenever possible, despite a number of control disadvantages.

To identify the dynamic characteristics of the test room, a dynamic power control is used. The dynamic on/off power sequence is organized such that the sequence covers the whole band of relevant frequencies; therefore the "on" and "off" periods are chosen at logarithmically equal intervals and shuffled in a quasi-random order (ROLBS sequence: randomly ordered log. distributed binary sequence, see Figure 2.16).

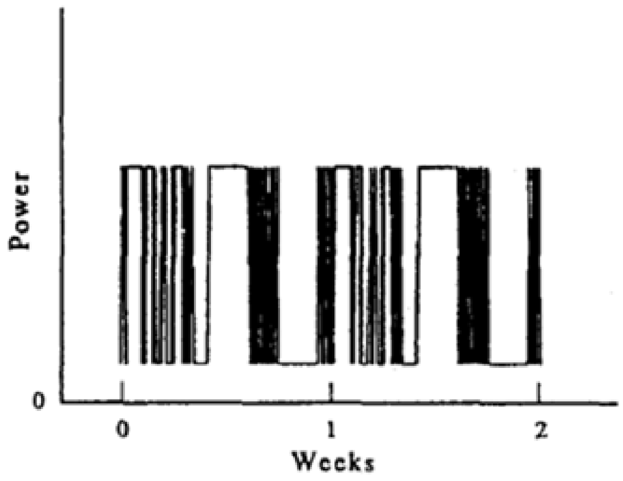


Figure 2.16: ROLBS control strategy [5]

All previously mentioned elements have been combined into the test sequence. The test takes nine weeks, which consists of one initialization week followed by 4 parts with a total length of 8 weeks.

Figure 2.17 shows a schematic view of the procedure, which is split into the following parts:

- part 0: 1 week, initialization at constant test room temperature
- part 1: 2 weeks, minimum power
- part 2: 2 weeks, high power
- part 3: 2 weeks, moderate power
- part 4: 2 weeks, dynamic power

[5],[1]

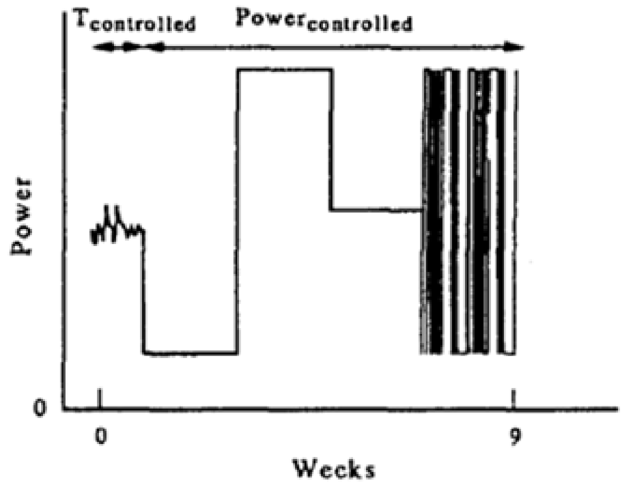


Figure 2.17: Schematic view of the test strategy [5]

In moderate weather conditions and/or components with low solar gains, heating power is used. In the case of high outdoor temperatures and/or high solar gains, the cooling system is required. As already mentioned, the aim is to maximize the temperature difference between the high and low power parts of the sequence by at least 20 K.
For a steady-state situation, the heat balance equation can be written like equ.(2.2) or in a slightly different form:

$$Q_{psc} = P_{hc} - (UA)_{tr,e} \cdot \Delta T_{tr,e} - (UA)_{tr,sr} \cdot \Delta T_{tr,sr} \tag{2.5}$$

where Q_{psc}=$(UA)_{psc}$·T_{psc}-$(gA)_{psc}$· G_{psc} is the net heat loss in Watts from the test room through the component to the exterior, P_{hc} is the heating/cooling power (W) supplied to the test room, $(UA)_{psc}$ is the UA-value (W/K) of the component, T_{psc} the temperature difference between the air node in the test room ($T_{tr,a}$) and the external air temperature (T_e), $(gA)_{psc}$ the gA-value (m^2) of the component and G_{psc} the intensity of solar radiation (W/m^2) in the plane of the component.

Dividing both the left hand and right hand sides in equ.(2.5) by ΔT_{psc} gives

$$\frac{Q_{psc}}{\Delta T_{psc}} = (UA)_{psc} - (gA)_{psc} \frac{G_{psc}}{\Delta T_{psc}} \tag{2.6}$$

A graphical (X,Y) plot, with $Y=Q_{psc}/\Delta T_{psc}$ and $X=G_{psc}/\Delta T_{psc}$, gives the linear relationship

$$Y = (UA)_{psc} - (gA)_{psc} \cdot X \tag{2.7}$$

where $(UA)_{psc}$ is the intercept of the curve with the Y-axis, and $(gA)_{psc}$ is the slope of the curve shown in Figure 2.18. In principle, only two different measurements points are needed to obtain the characteristics. With more points, the UA- and g-values are obtained by linear regression analysis, as illustrated below. [1],[10]

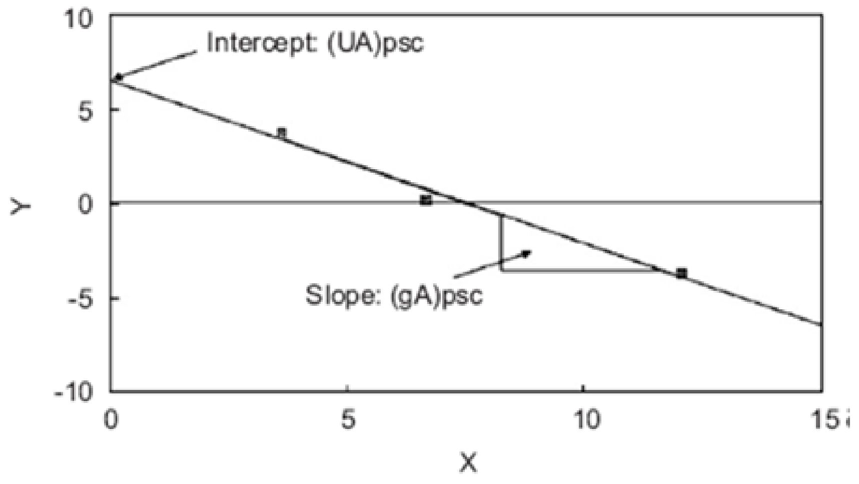


Figure 2.18: Steady state analysis [1]

3 The system at LTH (Lund tekniska högskola)

3.1 Introduction

During fall 2004, construction of a new full scale laboratory at Lund University began and was completed by summer 2005. The building was financed through a grant to the Division of Energy and Building Design by the Delegation of Energy Supply in South of Sweden (DESS). The goal of the laboratory is to find solutions to reduce the electricity used in buildings, which would reduce their running costs. The complex is designed for testing building components e.g.: glazing and shading systems. The laboratory contains four test rooms (numbering 106-109, see Figure 3.1 & Figure 3.2) and is a $220\,\mathrm{m}^2$ free-standing one-storey building, with a roof that has space to test solar components and systems, solar heating, photovoltaic systems and hybrid systems. There are four by a climate controlled chamber (105) surrounded rooms, if it is possible the temperature of these rooms (106-109) is kept at the same level as the chamber-temperature. The rooms are constructed using insulated sandwich panels which are homogeneous, with low thermal mass and no thermal bridges. The south wall is removable for the placement of different measurement objects. The four rooms consist of two identical pairs - 106 & 107 and 108 and 109. One pair is equipped with a modern office ventilation and lighting system for the study of visual and thermal comfort, while the other is designed for the advanced study of the impact of curtains or other forms of solar shading systems on energy use. A combination of a heat pump and solar thermal collectors is used to heat the laboratory, with high and low temperature solar collectors able to be connected to the system. The thermal energy can be used to produce hot water or to charge the ground source borehole for the heat pump.
Separate collector test loops are also installed to analyze solar thermal and hybrid collectors. Using this system, it is possible to simulate operating temperatures from below the dew point of air to that as high as 90 degrees Celsius.
Among the evaluable measures are:

- Modern window systems with variable sunshade and daylight performances
- Solar thermal systems for hot water production
- Energy-efficient appliances
- Heat pumps (both ground source and air-to-air)
- New solar collector designs
- Hybrid photovoltaic - thermal collectors
- Grid-connected PV systems
- Improved storage and system technologies for active solar systems [3]

The building is also equipped to house two smaller test boxes placed on top of each other. These will replace the earlier twin-boxes used for to determine g-values of shading devices in the Solar Shading Project. The roof holds a $120\,\mathrm{m}^2$ platform used to test solar components and solar systems, for both solar heating and photovoltaic (PV) systems. A small combi-system installed in the building for student laboratory work and demonstration purposes has four permanently installed solar panels from Borö on the solar roof connected to a 750 l hot water storage tank. The tank is used to produce domestic hot water (DHW) and low-temperature water for space heating. A 1300 l storage tank is used for component testing of solar panels. [17]

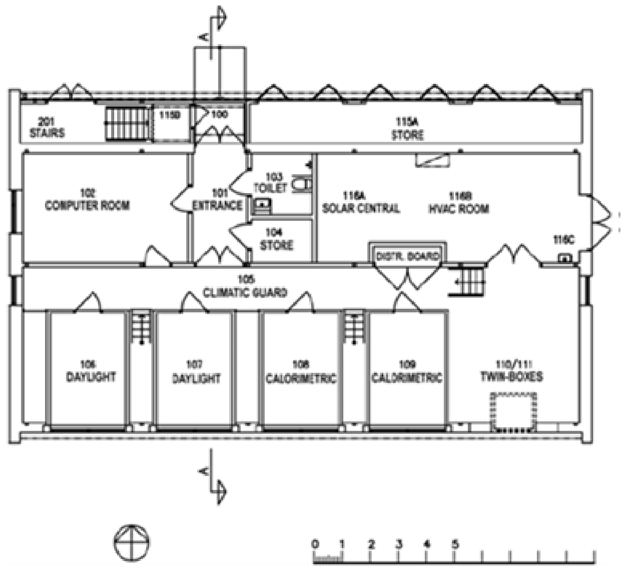


Figure 3.1: Laboratory plan [3]

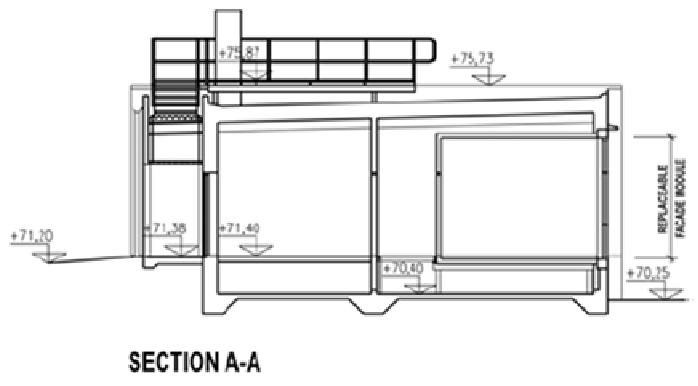


Figure 3.2: Laboratory section plan [3]

3.2 Laboratory description (Test rooms)

The two pairs of test rooms are surrounded by a temperature-controlled climate chamber. As mentioned in the previous chapter, one pair is designed for calorimetric measurements, which enables it to measure the effect of shading devices. The other pair is constructed in the style of an average office room, where it is possible to evaluate certain aspects of visual and/or thermal effects. The rooms measure 2.7 m x 4.0 m x 3.1 m (w x l x h), which is representative of typical single person offices. The four south walls are removable, for the attachment of different types of window or facade materials, see Figure 3.3.

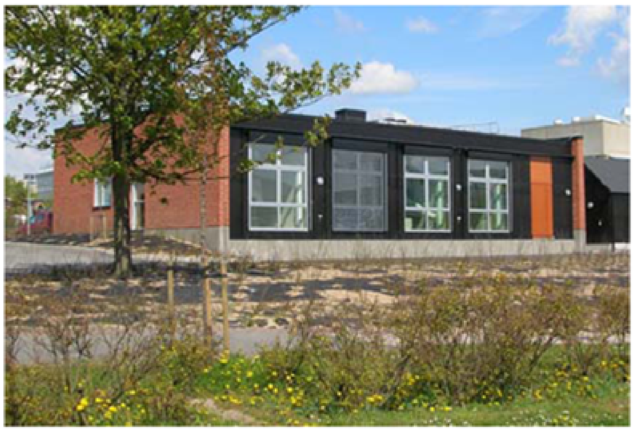


Figure 3.3: The south facade [3]

The room walls are constructed using a sandwich method, on a core of 120 mm extruded polystyrene and a surface of 0.5 mm vacuum-glued steel plates. This method of construction guarantees extremely low thermal mass and avoids the creation of thermal bridges. To determine the heat flow through the wall, thermocouple piles measure the temperature difference between the two surfaces (inside and outside) of the sandwich panels in equally distributed points. This construction principle is shown in Figure 3.4.

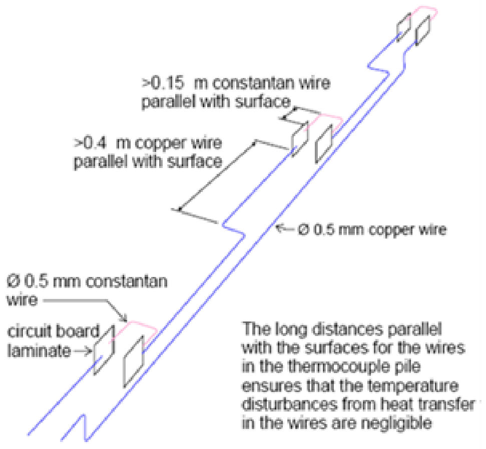

>0.15 m constantan wire parallel with surface
>0.4 m copper wire parallel with surface
Ø 0.5 mm copper wire
Ø 0.5 mm constantan wire
circuit board laminate
The long distances parallel with the surfaces for the wires in the thermocouple pile ensures that the temperature disturbances from heat transfer in the wires are negligible

Figure 3.4: Thermocouples are mounted inside the wall to measure the heat flow through the wall [3]

The rooms were pre-fabricated and mounted directly into the facade opening of a factory, like a chest of drawers (see Figure 3.5) compared to other test cells, like the PASSYS cell, is that the rooms are surrounded by a large climate guard (room 105) to minimize heat flows through all

walls except the south wall. Because of this, all thermal losses should be through the replaceable south facade.
With the “adiabatic” walls and the homogenous wall construction it is possible to assure a very high quality calorimetric measurement. It is therefore expected that the cells are better than the PASSYS cells.
The initial removable facade design is fully glazed with a thermally isolated aluminum frame and six windows. Four of the windows can be opened, the two in the middle are side-hung and the top ones are bottom-hung. All four rooms are initially equipped with 80 mm exterior motorized venetian blinds with grey lamellas, see Figure 3.3. [17]

Figure 3.5: Test room mounting [3]

In the calorimetric room-pair, steel surface plates are glued directly on to the polystyrene on the walls and ceiling. The floor surface is made of 9 mm anti-slip phenolic-coated plywood. The south surface is made of solar-controlled double glazing windows from Pilkington (6 mm HP Brilliant 66 - 15 argon - 4 clear glass), with a centre of glass U-value of $1.1\,\mathrm{W}/\left(\mathrm{m}^2\mathrm{K}\right)$, resulting in total solar energy transmittance (g-value) of 0.34 and a visual transmittance of approximately 67%.
The air in the rooms is circulated in a closed loop. A fan and duct unit is placed inside the room and the air is diffused via a diagonally across the room textile hose, see Figure 3.6. To keep the temperature at a constant level, there is an electric heater (rectangular box above fan, Figure 3.7) or an air-water heat exchanger (cooling, rectangular box at bottom). Both systems are placed close to the circulation fan (blue device). The specifications for the calorimetric rooms were:

- Air temperature range: 16 °C — 27 °C
- Precision of temperature control: ±1 °C
- Heating power: 0.6 kW
- Cooling power: 2.7 kW

- Air tightness: 0.2 ach at 50 Pa and airtight facade module [3]

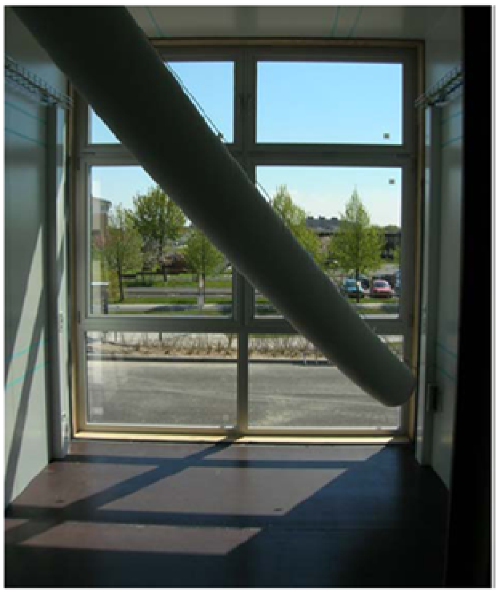


Figure 3.6: Air distribution hose [3]

For each room the indoor air temperature, water flow, inlet and outlet water temperature, and the amount of electricity is measured. Using the water flow and temperature difference of the water, the cooling energy can be calculated, the measured electricity is corresponding to the heating energy in the room, it includes each electric consumer inside. Both are measured continuously using the building's own Siemens (Desigo Insight) monitoring system. This system is only used for controlling purpose, with the evaluation done by a LOGGER (Campbell Scientific CR1000), which is described later. The flow diagram of the Siemens system is shown in Figure 3.8.

3.2.1 The controlling system

There are three different systems which work together in some part, the Siemens system, MBUS system and the LOGGER, a schematic model is shown in Figure 3.9. The MBUS system measures the water flow, inlet and outlet water temperature, with which it is possible to calculate the cooling energy $Q_{cooling} = c_p \cdot \Delta T \cdot flow$.

The Siemens system measures the electricity used in the room and the room temperature. PT100 are used as sensors for temperature measurement, with four sensors in different positions in each room- two at a top level, one close to the duct for the air circulation and one at the bottom. This arrangement guarantees a fast reaction to different impact behaviors. The two top level sensors take care of increasing temperatures (stratification, rising warm air), and as a result the room starts cooling quickly if there is sun irradiation. The sensor close to the duct acts as a sort of negative feedback; if the cooling system starts working, this sensor realizes this immediately and damp out oscillations, while the bottom sensor is for averaging purposes, without which the mean room temperature would be too high.
For the controlling of temperature, the SIEMENS system uses two PID (proportional, integral,

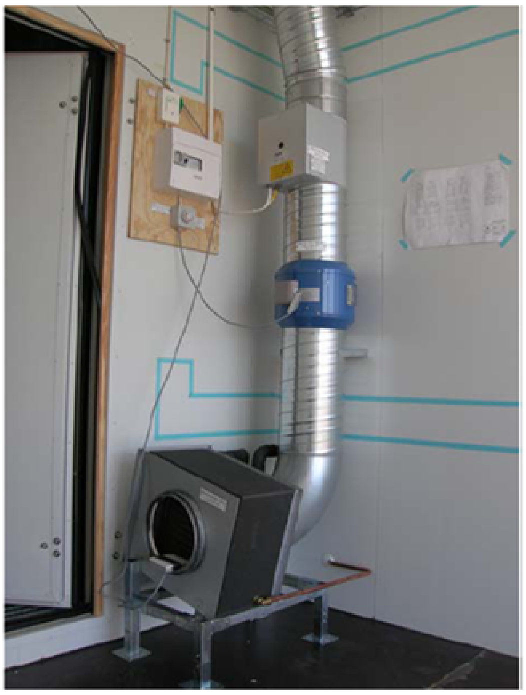


Figure 3.7: Cooling-, heating-system in room 108 or 109 [3]

derivative-controller), one for the heating system and one for the cooling system. There are two room temperature set points, one which denotes the point at which the cooling should start (higher level) and the other for the heating.
While the SIEMENS system is able to monitor all parameters which are measured by this system, there is the problem that these parameters are not sufficient for modeling room behaviors, which can lead to important parameters e.g.: solar irradiation missing. This is the reason a separate system which specialises in data collection is used (LOGGER); this system doesn't take part in any controlling.

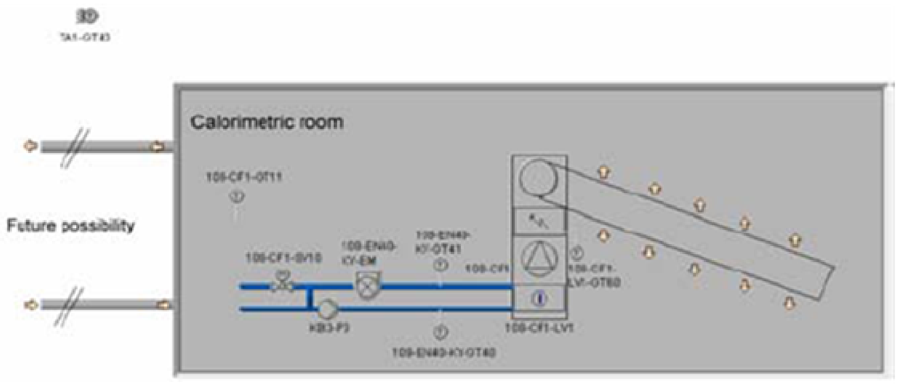


Figure 3.8: User interface of the Siemens system for cooling and heating [3]

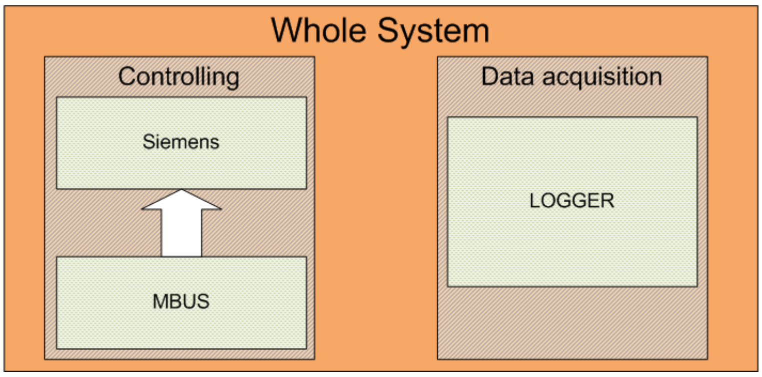


Figure 3.9: Schematic model of controlling and acquisition system

3.2.2 The acquisition system

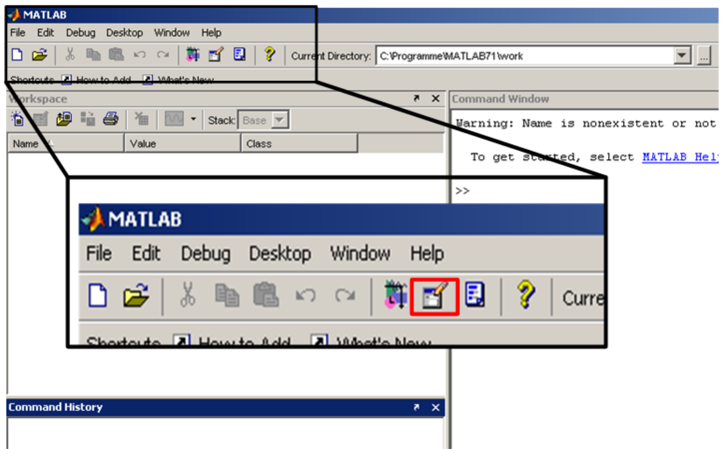
The system which takes care of logging all important data, needed to model room behavior and extract the relevant data (g- and U-value), is the Campbell Scientific CR1000 LOGGER (for more information, e.g.: interfaces of the logger, programming, etc. refer to: `http://www.campbellsci.com/`)[2]. This system measures numerous different parameters and while not all of them are used in the model right now, they may be required in a more complicated model and some of them can be used to check if the room is working correctly. All measured values are listed in Tab. A.1 along with a description. There is one case of measurement which is not an error but should be discussed to ascertain the reason behind it, which is as follows. The LOGGER system use three different sensors in each room to measure room temperature. The sensors are of a thermocouple type and they are placed at three different levels (57 cm above the floor, in the middle of the room and 34 cm below the ceiling); as discussed in the chapter before, the controlling system uses four PT100 at four different places. There being two different systems this is the reason the LOGGER system measures small ($\approx 0.5\,°C$) variations in room temperature, even though the SIEMENS system keeps the average room temperature stable. Due to different room working situations, cooling or heating, the stratification is different, therefore it is a balancing act between the two systems to keep the LOGGER average temperatures at a constant level.[4]

The LOGGER measures all parameters in 30 sec. intervals, which are averaged in 6 min. or 1h steps (average of 12 or 120 values). The 6 min. average is stored in a Datatable called CR1000_2_#6min_YYYY_MM_DD_hh_mm_ss.dat, the 1 h average is stored in CR1000_2_#oview_YYYY_MM_DD_hh_mm_ss.dat. The convention is as follows Y...Year, M...Month, D...Day, h...hour, m...min, s...sec. These files can be read with a self written MATBLAB program which is used for analyzing and modeling. The program is described in the next section and the program code is shown in the APPENDIX.

[2] 5'th of June 2009

4 The MATLAB program

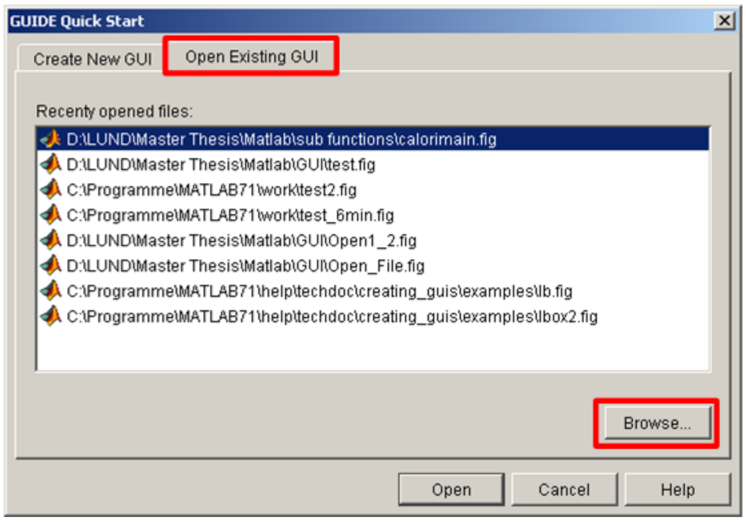
MATLAB[3] is a program for computing, calculation and analyzing data in every field of engineering. The purpose of this chapter is not to explain how to program or calculate using MATLAB, but to describe the code which was written to analyze and model room behaviors. To use the program, the files need to be copied or the program copied from the Appendix and stored under the right names. The program consists of six files. “calorimain.fig” and “calorimain.m” are the main files, with the “*.fig” telling MATLAB how the GUI should look and the “*.m” file being the corresponding code driving the functions by the pressing of individual buttons. The files “Modeling.m” and “Calculation.m” contain the code for the different room models e.g.: capacity and time-shift-model (described in Chapter 5). “strsplit.m” is the code for splitting a string in different pieces and is used to separate the header of the LOGGER-file into different parts. The last m-file “xticklabel_rotate.m” is for the formatting of the x-axes, especially for the 45 ° tilt of the date and time stamps. [9]
The first section describes the GUI (general user interface), which, under stable working conditions, is the only point of interaction between the user and the program. The second part describes how the modeling and calculation of the g-value is implemented so that a user can modify this part of the program, if necessary, to create a more sophisticated and accurate model.
If once MATLAB is started up, the following window appears on the screen (Figure 4.1). By pressing on the GUIDE button marked in red the window shown in Figure 4.2 pops up. Click on the tab “Open Existing GUI”. If the list of “Recently opened files:” is empty or the path to the file “calorimain.fig” is not listed, click ”Browse. . . ” and select the path where the file is stored.


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\Programme\MATLAB71\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
Command Window
Command History
MATLAB
File Edit Debug Desktop Window Help
Curre
Shortcuts How to Add What's New

```
Warning: Name is nonexistent or not

  To get started, select MATLAB Hel

>>
```

Figure 4.1: MATLAB start window

[3]http://www.mathworks.com/ (5’th of June 2009)



Recenty opened files:
D:\LUND\Master Thesis\Matlab\sub functions\calorimain.fig
D:\LUND\Master Thesis\Matlab\GUI\test.fig
C:\Programme\MATLAB71\work\test2.fig
C:\Programme\MATLAB71\work\test_6min.fig
D:\LUND\Master Thesis\Matlab\GUI\Open1_2.fig
D:\LUND\Master Thesis\Matlab\GUI\Open_File.fig
C:\Programme\MATLAB71\help\techdoc\creating_guis\examples\lb.fig
C:\Programme\MATLAB71\help\techdoc\creating_guis\examples\lbox2.fig

Figure 4.2: GUIDE quick start

The GUIDE editor then opens (Figure 4.3). In this window, the layout of the GUI can be changed or new buttons, slides and axes added. By clicking on the “Run”button (green arrow, red rectangle on the right) the program will start running. If something in the program has to be altered, it is then necessary to open the “M-file EDITOR” (red rectangle on the left). Up to this point, only the programming part has been used; now that program is working, it can be used to analyze data.

After clicking the run-button the program is now running and the following window will be shown (Figure 4.4). First the dat-file which needs to be read has to be selected by clicking the “open” button (*1.* in Figure 4.4). A browser window will then open and the path and file can be selected. In the next step, the program lists all the available parameters in the listbox (*2a.*) e.g.: average room temperature, cooling energy etc.
It can read both types of LOGGER-files the 6 min. as well as the 1 h overview, for displaying some logged values (e.g.: average room temp, water flow) there is no difference from the program side. If the option(s) “Model’s” and/or “Calculation” is chosen (Figure 4.4), the program will only work for 1 h overview data as the modeling process is optimized for 1 h steps- for example, the time shift would be 3 mins. instead of 30 mins. and this does not work. If nothing appears in this box and a “ping” sound is emitted there are maybe not a number entries (“NAN”) in the LOGGER-file and the program cannot function. When this occurs, the error-message displayed in Figure 4.5, will appear in the MATLAB command window. It is then necessary to replace this value with a real number; if it is a value which is not used in the model, the “NAN’s” can simply be replaced by -1, otherwise a real value if the modeling/calculation is used to create real parameters.
If parameters are shown in listbox *2a.* as expected, then there are two different options. One is to have a look at an unlimited number of parameters by selecting them in the listbox (multiple selections can be done by pressing the SHIFT or CTRL keys) then clicking “Read Listbox”. The listbox for “Chose Start and Stop Date” (*4.*) will then list all 6 minute or hourly time steps, and the different selected values for the whole period stored in the dat-file will appear in the plot (*6.*). By selecting two of them (by pressing the CTRL key)- one for the start, and one for the stop value, a separate figure with only the specific range can be plotted. If the button

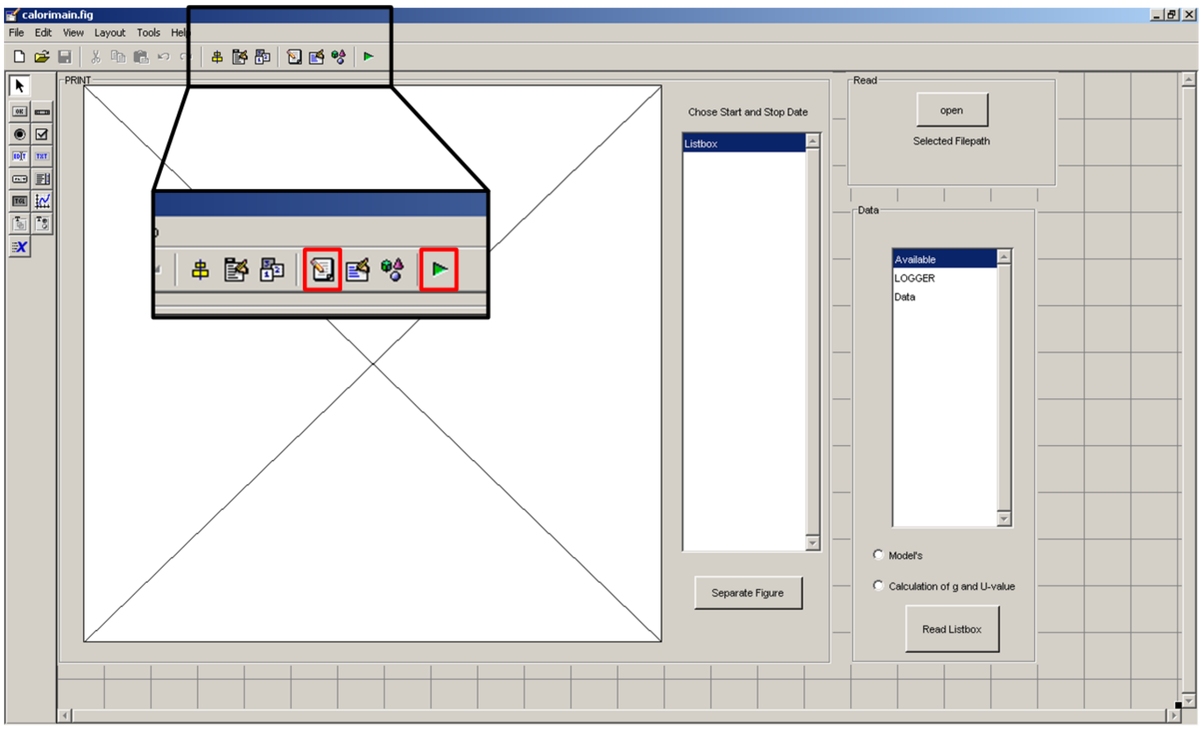


Figure 4.3: GUIDE editor

“Separate Figure” is clicked, the start and stop time is marked with two vertical dashed red lines.

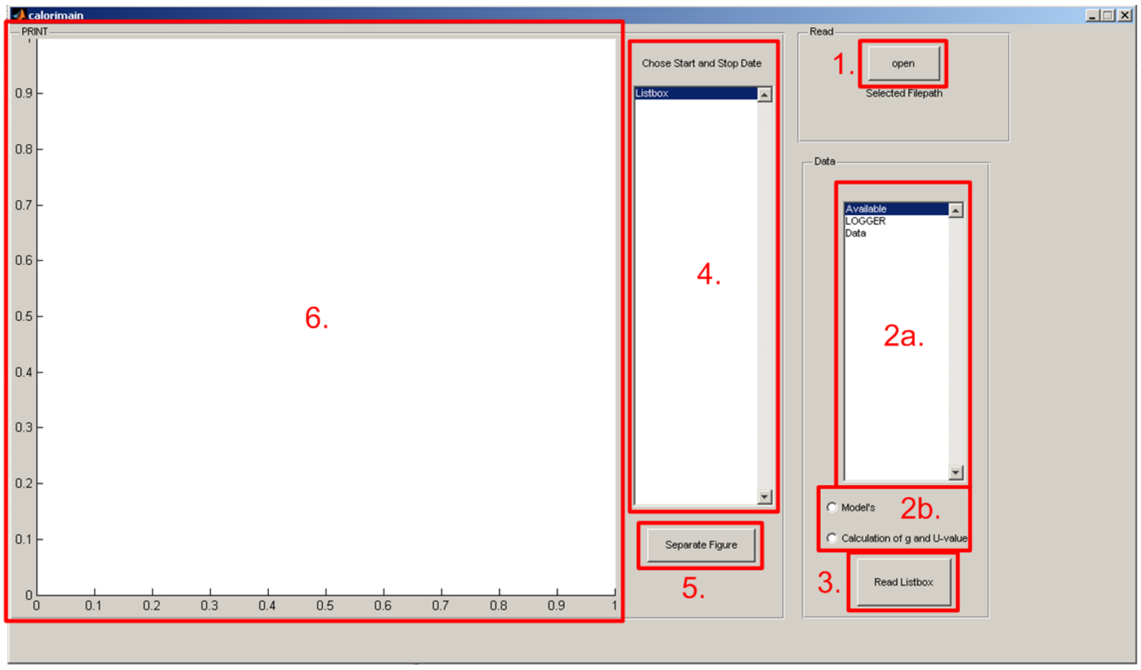
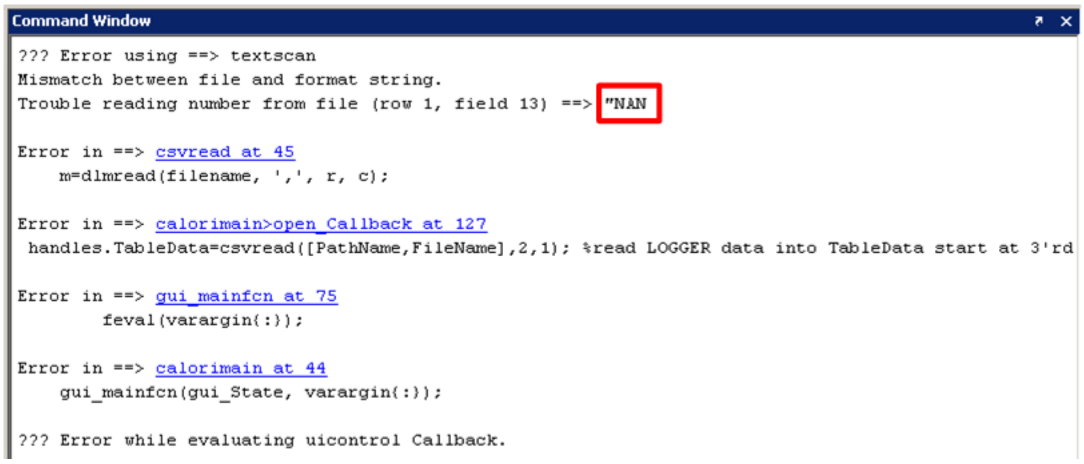


Figure 4.4: Blank user interface

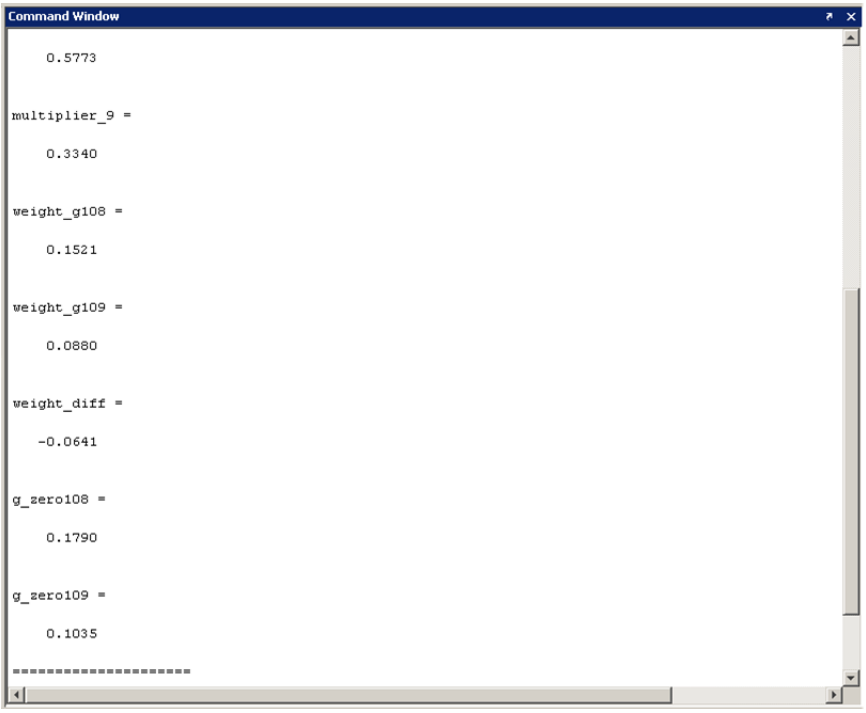
The other option is to only select the options "Model's", or both the "Model's" and "Calculation of g and U-value" options at the same time (*2b.*), because the calculation only works for the model and therefore both buttons have to be marked. The order is as follow: after marking one or both buttons, click "Read Listbox" and the whole time period of the time-shift and capacity-model for which the necessary parameters are stored in the dat-file will appear in the plot (*6.*). The next step is to select the time stamps for which the calculation should be performed (*4.*; *Choose Start and Stop Date*) and/or for which range the model should be printed. Next, click "Read Listbox", because the calculation is performed by clicking this button and it will account for the selected time scale. Finally, the button "Separate Figure" should be clicked to display the model in a new window and to see the calculations in the command window which will be as follows (Figure 4.6). There will be several g-values calculated and what they mean is described in Chapter 6.



```
??? Error using ==> textscan
Mismatch between file and format string.
Trouble reading number from file (row 1, field 13) ==> "NAN

Error in ==> csvread at 45
    m=dlmread(filename, ',', r, c);

Error in ==> calorimain>open_Callback at 127
 handles.TableData=csvread([PathName,FileName],2,1); %read LOGGER data into TableData start at 3'rd

Error in ==> gui_mainfcn at 75
        feval(varargin{:});

Error in ==> calorimain at 44
    gui_mainfcn(gui_State, varargin{:});

??? Error while evaluating uicontrol Callback.
```

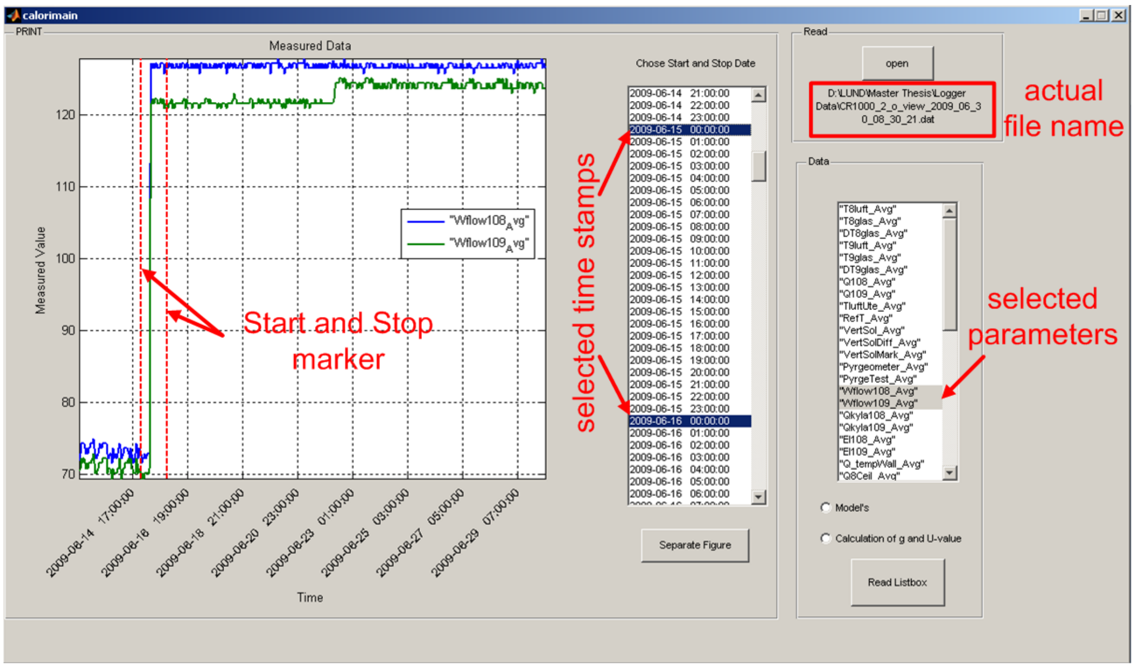
Figure 4.5: Error message during the reading process, if there is not a number ("NAN") present in the LOGGER-file

The following two pictures (Figure 4.7; Figure 4.8) are examples of how the first case can look. There is only one thing which needs to be considered when a time range is selected- the period should be more than 5 steps (30 min. for the 6 min. model, and 5 h for the 1 h model), otherwise MATLAB will encounter problems labeling the x-axis. Figure 4.8 illustrates the detailed plot between the two vertical red lines ("Start and Stop marker") in Figure 4.7.



```
    0.5773

multiplier_9 =

    0.3340

weight_g108 =

    0.1521

weight_g109 =

    0.0880

weight_diff =

   -0.0641

g_zero108 =

    0.1790

g_zero109 =

    0.1035

=====================
```

Figure 4.6: Results of different g-value calculations



D:\LUND\Master Thesis\Logger Data\CR1000_2_o_view_2009_06_30_08_30_21.dat

Figure 4.7: Description of case one, print of logged parameters, e.g.: water flow room 108 & 109

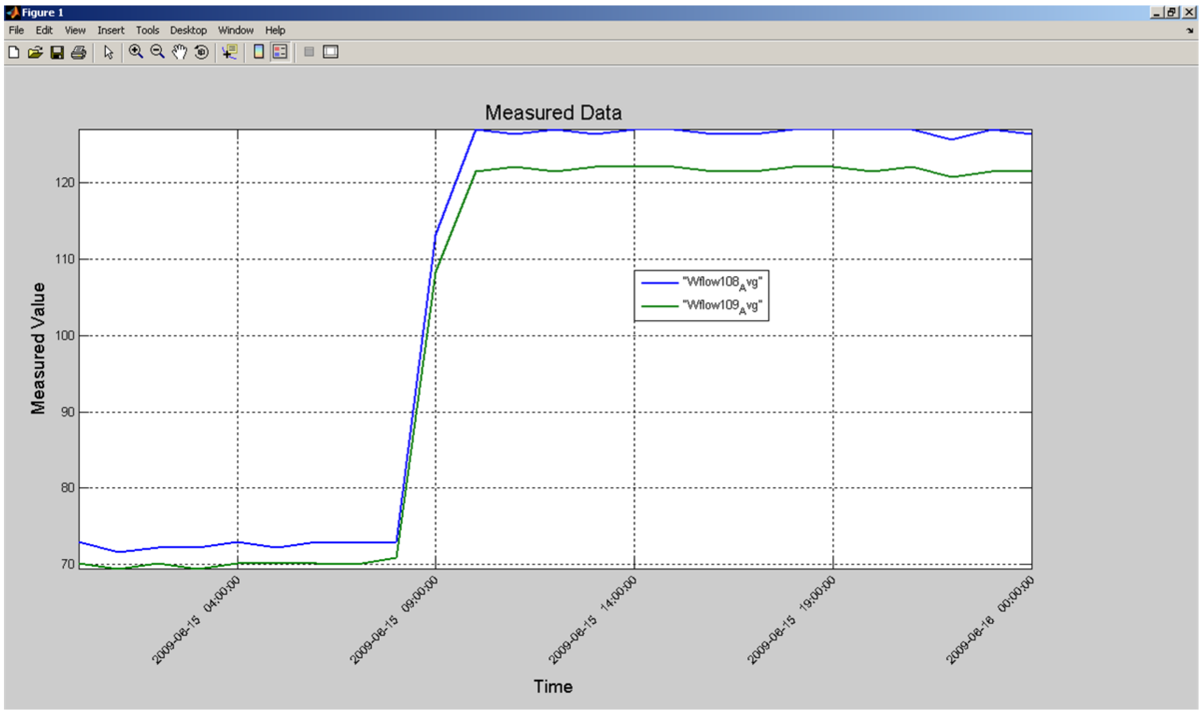


Figure 4.8: Detailed print of the selection in Fig.4.7 (between red the vertical dashed lines)

5 Modeling

In the previous chapter the MATLAB program was used to analyze data. The next step is to use the program. This chapter discusses the entire modeling process, starting with the analysis of a few basic forms of thermal room behaviors. By keeping these behaviors in mind, it is easier to understand the modeling and why each step is performed. The modeling will be outlined in small steps from the basic model to more sophisticated versions. Two different models will be examined, the capacity model and the time-shift-model, and the advantages and disadvantages of each discussed. At the end it will be shown which model yields better results.

During the modeling process, it was found that a angle-dependent g-value and calibration constants for the cooling energy were required, the need for which were discovered during the modeling process and included from the beginning on, to avoid too many modeling steps. Otherwise at the point where the need of one of those parameters came up, the whole modeling would have been described form the start of the thesis, that's why they were included from the basic model on and at certain sections 5.2.4, 5.2.5 the need for this is described.

5.1 Thermal room behaviors

Two forms of thermal behaviors are discussed. The first one is generally stable because the same origin value is reached in the long run even if it changes during the day. The other is dynamic and the same point is not reached daily, (nor is it possible to run this kind of test every day as the preparation and evaluation of measurements requires more than a day).
Every attempt is made to use the same time range for the whole modeling process in order to make proper comparison, therefore the vertical solar irradiation (red line in Figure 5.1) is plotted in all figure in which it is needed, or is useful for understanding.
Figure 5.1 shows average room temperature against solar radiation, being the mean value of the three measured temperature levels (top, middle and bottom). The red line shows the vertical solar irradiation; while the weather conditions during the first two days were perfect, the third day was slightly cloudy but it is good to evaluate the rooms under different conditions. As shown, the temperature over a long run is stable, which is important for further modeling steps, as it means that there is no energy stored in the air. Even on a day where the temperature is almost stable, it can fluctuate by $\pm 0.5\,^{\circ}\mathrm{C}$. This is due to the sun heating up the air in the room, the average temperature is increased. The cooling system then starts working, cooling the air and this causes a change in stratification, especially compared to a more moderate situation such as night time.

Figure 5.2 shows the heat losses through all surfaces except the south front (test object), and also shows the influence of the stratification process. At noon, when the cooling system works the most, heat losses reach negative values ($-10\,\mathrm{W}$), which is why the floor is cooled down

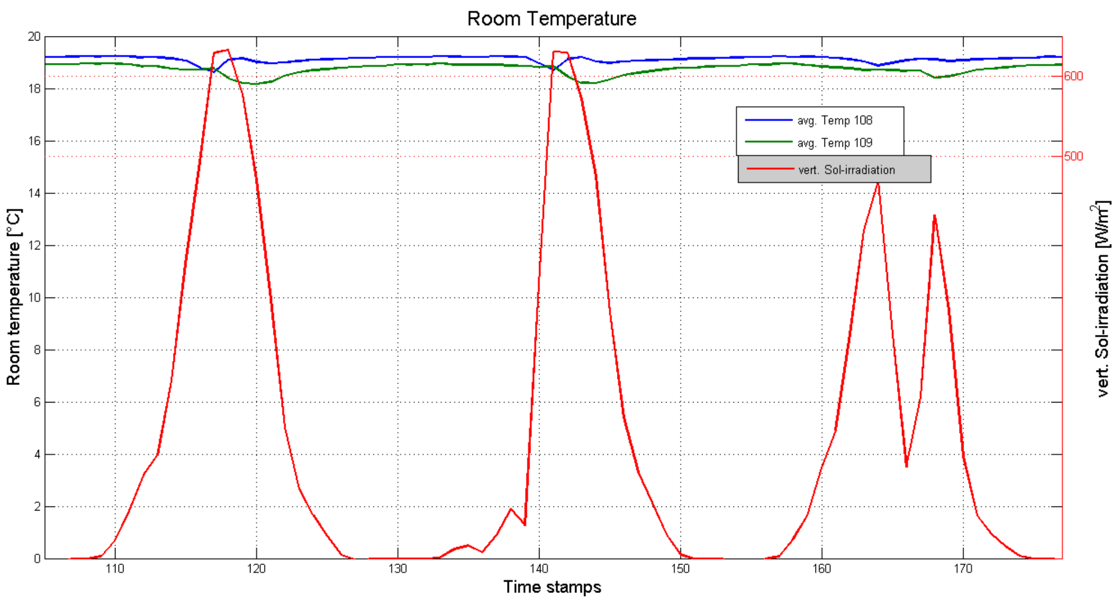


Figure 5.1: Room temperature against vertical solar radiation

dramatically compared to the surrounding temperature outside of the rooms. The surrounding room temperature is 20 °C that means that the average room temperature (see Figure 5.1) is always a little too low and energy is coming into the room. This effect increases during the day because of the cooling. In the evening, when the cooling works as a lower level, the heat at the top of the room has a bigger influence and energy is going out. This effect decreases through the night and the cycle repeats the next day.

Next a closer look at the “dynamic” behaviors is taken and it is shown how quickly the rooms react when the temperature leaves the set point. Two different systems are working in each room: the heating system with a maximum power of 600 W and the cooling system with a maximum power of 2700 W. The next two figures illustrate how the different levels of the systems affect the thermal characteristics of the rooms Figure 5.3 shows the room behavior when the cooling is operating. For this measurement, the internal cooling and heating were switched to manual mode and turned off. To heat the room up, electric heaters were placed in the room, and when the temperature reached approximately 42 °C (there is no specific reason for this particular temperature) the heaters were removed and the cooling and heating switched to automatic mode. As illustrated, the end temperature is approximately 19 °C which corresponds to the controlled temperature in the first figure in this section. The time constant is roughly 5 h. An important point is that both rooms behave identically, as this is important for the measurement of test objects.

The opposite procedure is shown in Figure 5.4. The cooling and heating was switched to manual and turned off, and the windows were opened to cool down the room as it was February and the outside temperature was sufficiently cold to cool the room. When the temperature reached 8 °C (no specific reason for this point) the windows were closed but the internal system was not switched to automatic mode; instead, the cooling was left on manual and off , and the heater was physically separated from the controlling. The heater was directly plugged into a plug socket (working at 100%, this is equal to 600 W) without the thyristor-switch, which controls

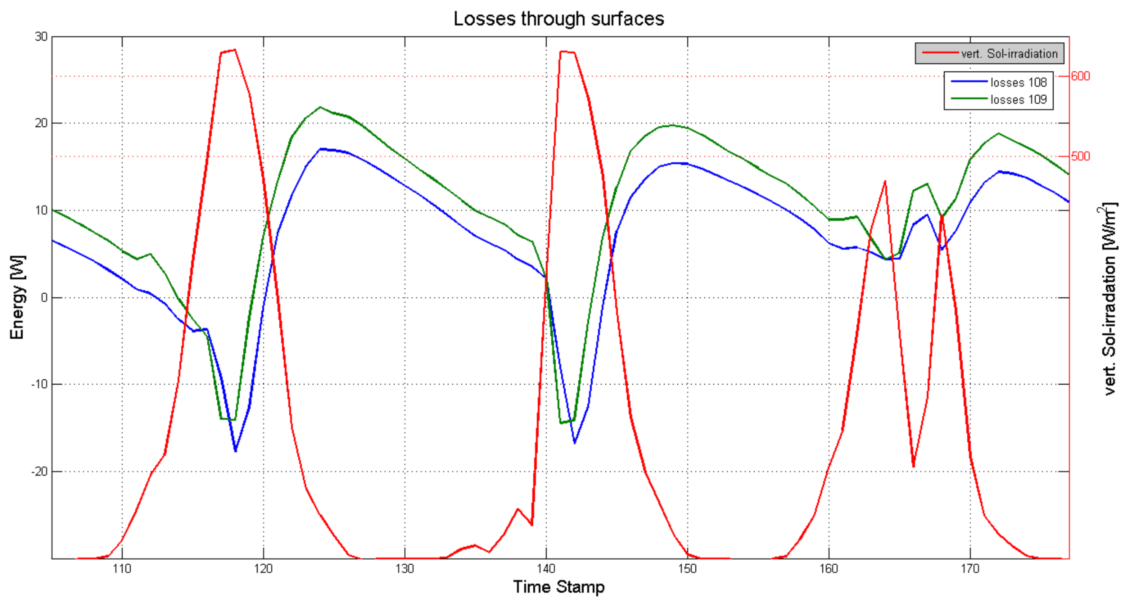


Figure 5.2: Heat losses through surfaces against vertical solar irradiation

the amount of heating used to reach the set temperature in a range of 0%—100%. The reason the thyristor switch was bypassed is because the purpose of heating was not to heat only to a set temperature, and the SIEMENS program does not allow switching the heating to manual and 100%. The portion of the graph within the gray area from 2009-02-15 09:00:00 onwards was not used for the time constant measurement as sun irradiation caused the room to heat up faster than before. As a result, an end temperature of 38 °C was estimated instead of 42 °C, which gives a total temperature difference of 30 °C and a time constant of approximately 5.5 h. As mentioned at the beginning of this chapter there are two different system values for cooling and heating, which explains the different time constants. The heating with 600 W provides a similar effect to that of a nice sunny day; as shown later in this chapter, a g_0-value of 0.31 is used and the glass area is around $2.7\,\mathrm{m}^2$. If there is sun irradiation of around $650\,\mathrm{W/m}^2$ with g_0 and a glass size of $\approx 2.7\,\mathrm{m}^2$, the energy coming into the room is 550 W and the room will behave in a similar manner for this irradiation condition. Using the following reaction calculation, if the temperature range should be within 1 °C the room has to react in:

$$\begin{aligned} \Delta T_{1\,^\circ\mathrm{C}} &= \Delta T_{30\,^\circ\mathrm{C}} \cdot \left(1 - e^{-\frac{t}{\tau}}\right) \\ \rightarrow t &= ln\left(1 - \frac{1\,^\circ\mathrm{C}}{30\,^\circ\mathrm{C}}\right) \cdot 330\,\mathrm{min.} \approx 11\,\mathrm{min.} \end{aligned} \tag{5.1}$$

If the room temperature does not exceed the ±1 °C range, then the cooling has 11 min. to get rid of the sun's effects.

Having looked at thermal behaviors and knowing that the rooms are able to keep the room temperature stable under high sun irradiation conditions and dynamic conditions, we move on

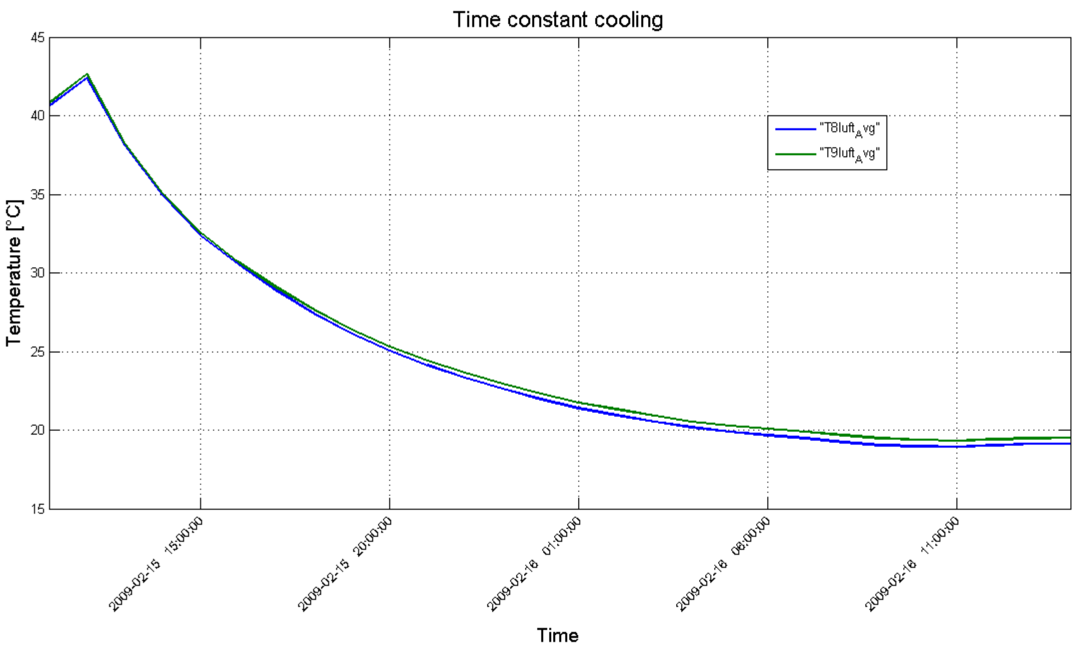


Figure 5.3: Time constant measurement for the cooling process

to the most important part of this work. In the next chapter the entire modeling process is explained.

5.2 Modeling process

The modeling process is split into three steps. First, we discuss the basic model and the improvements made to it. Two different approaches were evaluated, the first being the capacity model and the second the time-shift model. We take a closer look at the advantages and disadvantages of each model and discuss why each model could or could not be used to obtain the parameters (g- and U-values) in the final calculation.
The entire modeling process was carried out in a manner similar to a recursive calculation process. From the basic model onwards, if there were improvements or solutions to an issue at any particular step that were not part of the existing model, it was implemented in the model and the whole modeling process restarted. This is the reason for the influence of an angle dependent g-value being shown at the end of this chapter, as this principle is shown in all three modelling steps. For ease of understanding, only the final model will be explained, and a comparison of what the results would look like if an angle-dependent g-value was not used displayed, to show why it was needed.

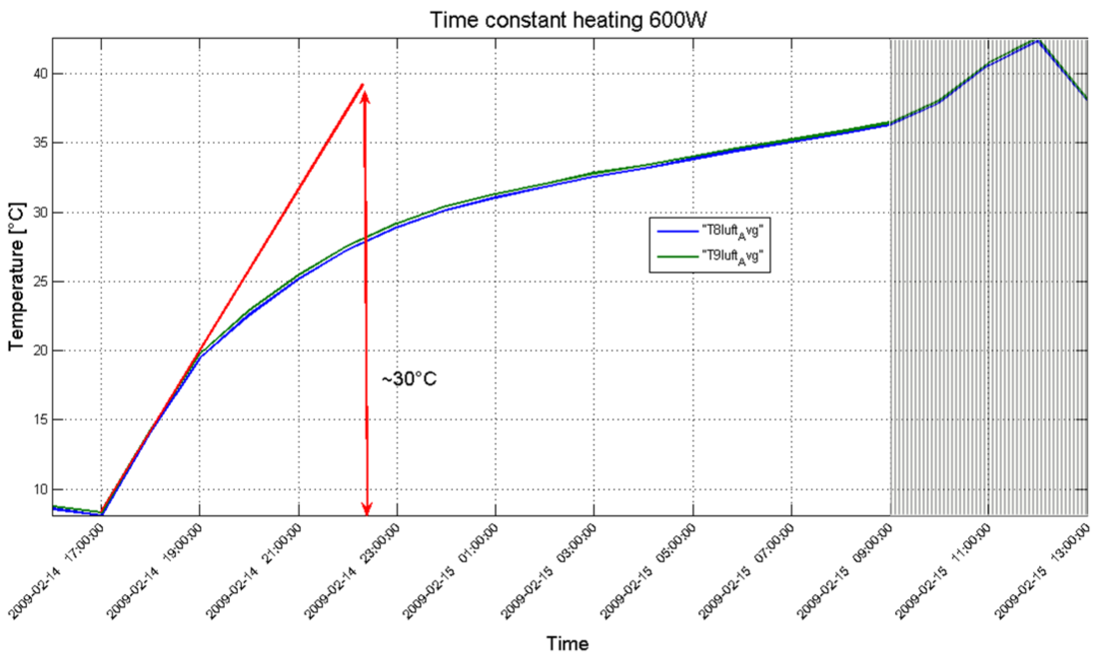


Figure 5.4: Time constant measurement for heating with 600 W

5.2.1 The basic model

Figure 5.5 shows the vertical solar irradiation $[W/m^2]$ and cooling energy $[W]$. The fact that cooling has a time shift is important for understanding why the basic model behaves as it does, and why the time-shift model was used. Due to the sensor arrangement (PT100) there is no way of avoiding this time shift. This is because it takes time (up to when the minimum level is reached) for the temperature control to realize the room has heated up from the sun. Furthermore, not all energy from the sun gets transferred to the air and instead heats up the surfaces in the room, which results in energy stored in these surfaces and later radiated back into the room, which also causes a time-shift effect.

The basic model (equ.(5.2); Figure 5.6) consists of five terms:

1. Energy coming through the window: $g \cdot G \cdot A_{glass}$; g. . . g-value, G. . . vert. sun irradiation, A_{glass}. . . glass size of the window (2.667 m^2)
2. Energy going out through the window and frame: $U \cdot \Delta T \cdot A_{total}$; U. . . U-value, ΔT. . . temperature difference between inside and outside, A_{total} . . . total size of glass and frame
3. Cooling energy: $Q_{cooling}$
4. Heating energy (The circulation fan, at 200 W—240 W, heats the room): $Q_{heating}$
5. Losses through surfaces: Q_{losses}

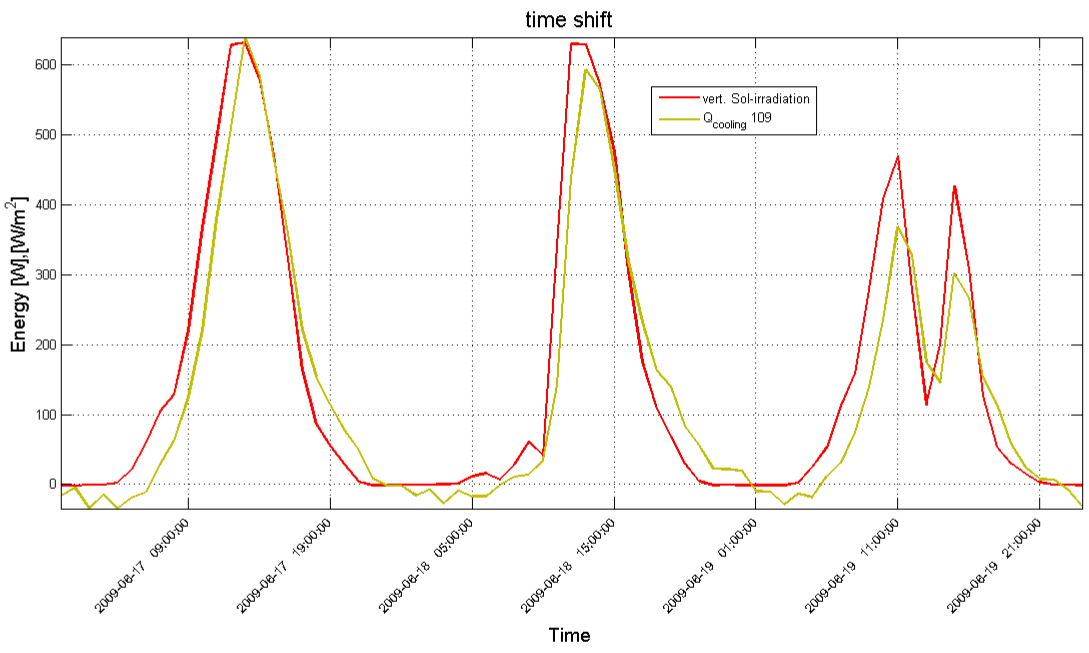


Figure 5.5: Time shift between solar irradiation and cooling energy

$$Q_{tot} = g \cdot G \cdot A_{glass} - U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses} \tag{5.2}$$

The losses through the window and walls (2. and 5. term) have a lesser influence compared to the amount of energy coming in through the window and taken care of by the cooling (1. and 3. term). The heating energy is a constant factor, which has no influence on dynamic characteristics. The brown line shows a combination of the first two parts in the Q$_{tot}$ equation. The perfect equation for describing the room's behaviors should net to zero (green line), but this solution is not sophisticated enough to account for all factors to give exactly zero. However, because sun irradiation and the cooling are major parts in this balance equation, it is clear why the green curve in the figure has a positive peak in the morning and a negative peak in the evening. Furthermore, Figure 5.5 illustrates a positive energy balance in the morning when the cooling is not powerful enough, and a negative energy balance in the evening when the cooling is stronger than sun irradiation. This is illustrated by the green line (Figure 5.6) and is representative of what would happen under perfect conditions. While analysis would be much more complex if there are disturbances (e.g.: clouds on the third day), the perfect conditions during the first two days were chosen to see how the system/model reacts under perfect conditions.

5.2.2 The capacity model

As discussed the basic model, two factors are responsible for Q$_{tot}$ in the basic model not equating to zero. One was the delayed reaction of the cooling and the other the stored energy in surfaces.

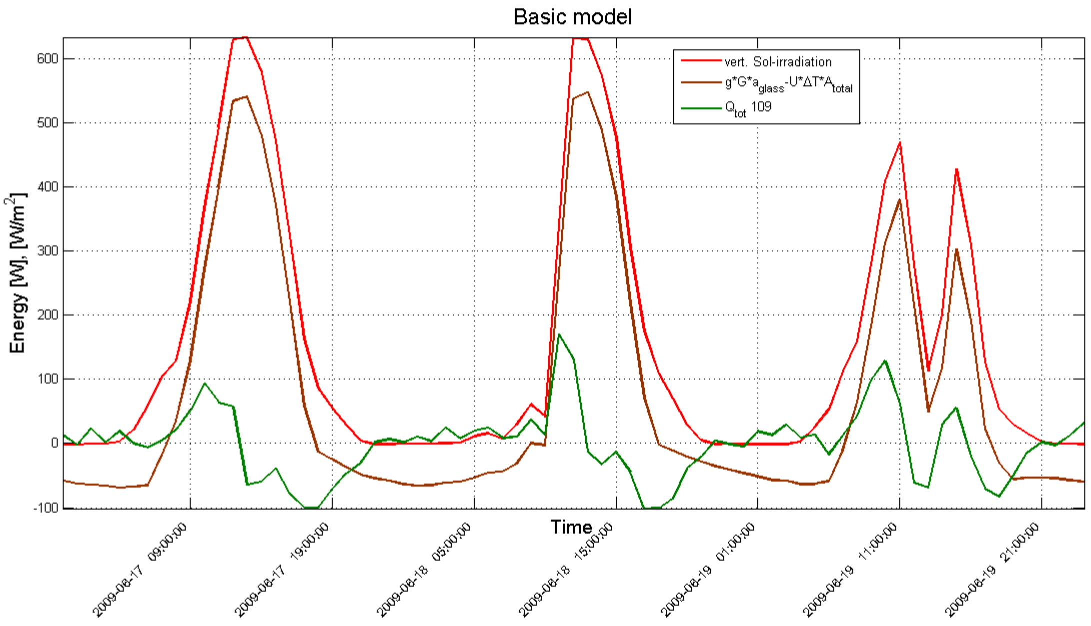


Figure 5.6: Basic model $Q_{tot} = g \cdot G \cdot A_{glass} - U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses}$

It therefore makes sense to include a term in the model which takes care of the stored energy, which is done by extending the basic model using the following term:

6. With regards to the energy stored in surfaces, the temperature difference between the old timestamp (index i-1) and the new (index i) is taken and multiplied by a specific heat constant: $c_p \cdot (T_i - T_{i-1})$; c_p ... specific heat constant, $T_{i;i-1}$ ... temperature for different timestamps.

The whole model then looks as follows:

$$Q_{tot} = g \cdot G \cdot A_{glass} - U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses} - c_p \cdot (T_i - T_{i-1}) \quad (5.3)$$

The way this extension to the model influences the total energy balance is shown Figure 5.7, which shows vertical solar irradiation total energy of the basic model (green), the total energy of the capacity model (dark blue) and the negative value of the capacitance. The negative value makes it easier to compare the overlapping parts with the basic model, to see which parts rid of by with this extension.
As shown, the model only works around noon and the basic peaks were removed assuming c_p 40 was used; this was not a calculated number but found using trial an error to see if the model works in principle. If it is smaller in real life, the effect would not be large enough to make a significant impact; conversely, if the number is much larger, then there would be a huge negative peak during noon. Some may argue that using only one capacity term for all different surfaces is an oversimplification because different surfaces are made from different materials,

and hence have different specific constants; this has been accounted for in the model.
During the initial steps in creating this model, all surfaces where represented by different terms in the equation; the floor and the other areas of the room are still separated in the MATLAB program and only use the same c_p and there is no difference between using them separately or combined. The justification for this simplification is that all the capacity process are correlated hence one surface heating up would result in all the others following in more or less the same heating shape, therefore the lack of a need for separating them. Instead of using four terms with different c_p, one can be used and the c_p adjusted correspondingly. Because of the correlation and similarity of all the shapes of the capacity effect it isn't possible to spread this effect out; the capacity peaks as shown in Figure 5.7 are high enough but not broad enough. The only takes care of the unbalanced energy during noon.

Because of all these effects there was no way of getting a better fit with this kind of model and no further steps were taken.

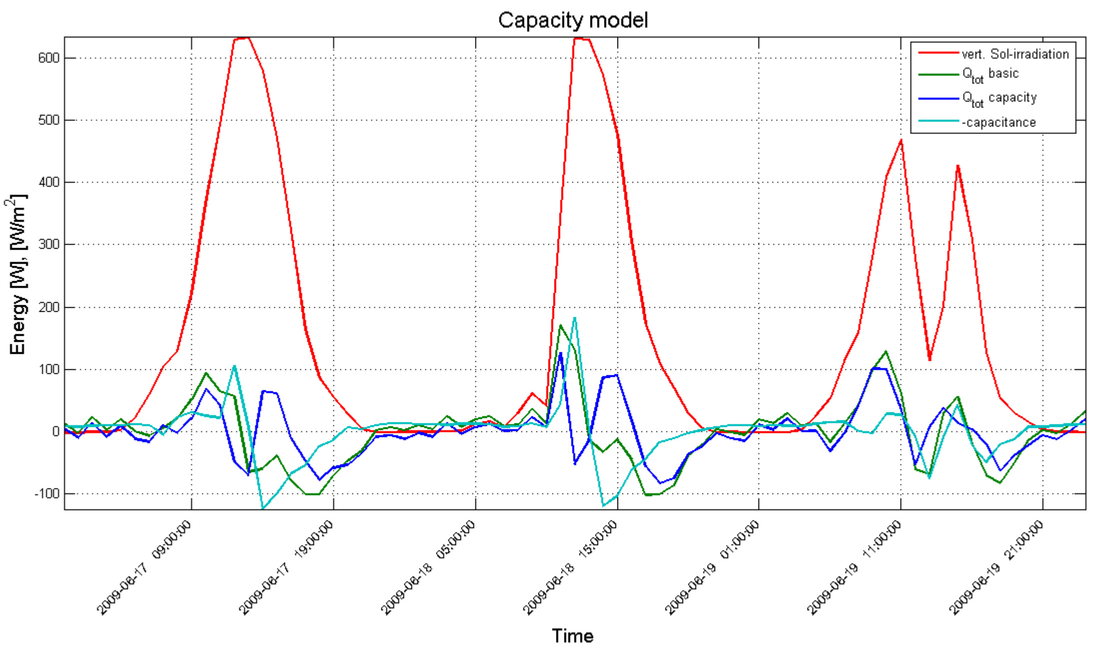


Figure 5.7: Capacity model $Q_{tot} = g \cdot G \cdot A_{glass} - U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses} - c_p \cdot (T_i - T_{i-1})$

Another model has to be found and tested to see if there is a better way to fit and zero out the energy. This was done using the next part, the time-shift model.

5.2.3 The time-shift-model

As mentioned in the basic model, there were two reasons for the effects not zeroing out. One was the stored energy in the surfaces, which the last model attempted to account for, but was unsuccessful. The other was the cooling system started too late; even if it is physically incorrect, there is no rule that disallows the sun irradiation from being shifted forwards and backwards. A forward shift would not make sense as it would increase the gap between cooling and room heating by irradiation. This model shifts the irradiation back by 1 h and allows either no shift or a 1 h shift. For example the sun irradiation from 10 a.m. was used and shifted in the model to 11 a.m. The influence of this shift is drawn in Figure 5.8.
The equation for this model is as follows:

$$Q_{tot} = g_{i-1} \cdot G_{i-1} \cdot A_{glass} - U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses}, \tag{5.4}$$

The changes to the basic model are the i-1 indices for g and G. This allows shifts in full time step periods (meaning only hourly or 6 min. steps), but for which the modeling is not optimized. The red and light green lines are known from the time-shift explanation at the beginning of the modeling process. The brown line is the solution to the first two parts of equ.(5.4). The difference between this model and the basic model is that if the vertical sun irradiation is at maximum, this part of the equation should be maximum as well, but this has been shifted 1 h back. The cooling energy has been printed for a better comparison. If cooling and irradiation are compared, the cooling starts too late; if the brown line and the cooling is compared then it is recognized that the cooling starts too early; which is exactly what the total energy balance (Q_{tot}) shows. Instead of a positive disbalance in the morning and negative in the evening, the situation is now the opposite and the cooling starts too early compared to the irradiation. This means that a full 1 h shift is too much, but as has been mentioned the LOGGER only logs in 1 h steps, so it is not possible to shift half an hour by setting the index to 0.5. A trick is thus is necessary.

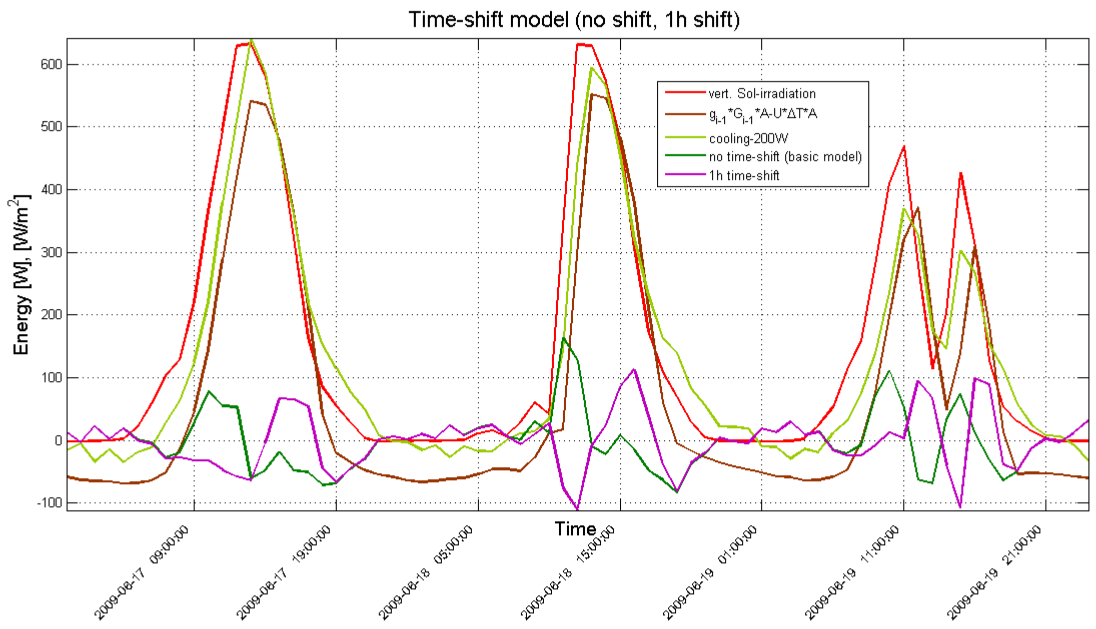


Figure 5.8: Time-shift-model (no shift, 1h shift)

This trick is explained in the following step of the modeling process, where the time steps were weighted: 50% of the old values and 50% of the actual values were used as the shift rate.

$$Q_{tot} = g_{i-1} \cdot (0.5 \cdot G_i + 0.5 \cdot G_{i-1}) \cdot A_{glass} - U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses} \quad (5.5)$$

This equation explains how the shift in the program was solved, Figure 5.9 shows a really good result, even the phase between the cooling and the shifted energy coming into the room (brown) is low. As they are in phase right now, there is no better way to fit the two curves with this model. Compared to the total energy (brown), the deviation of Q_{tot} from zero is really low and are two positive and two negative peaks, which will zero out over the span of a few days. This step of modeling was approved as being sufficiently accurate and it was decided that it was time to proceed to the next stage of this work, which is to compare the two rooms to see if they work equally and the compared measurements are acceptable. Before this step is explained, the result of the angle-dependent g-value is shown to clarify why it was used in all the previous steps of modeling, from the basic model to this stage of the time-shift model.

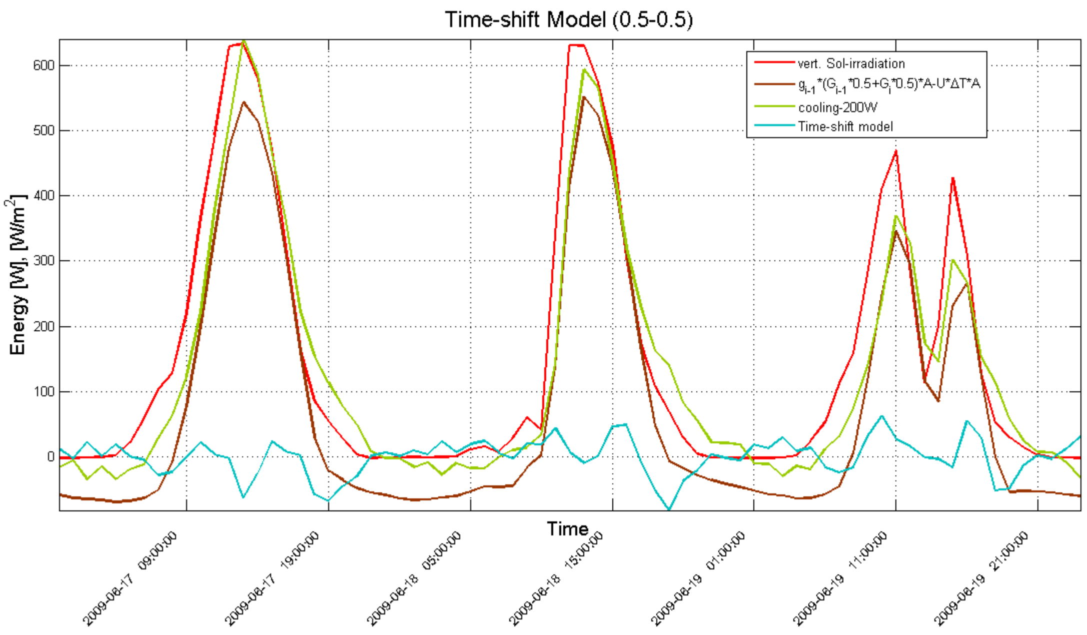


Figure 5.9: Weighted time-shift-model

5.2.4 The angle-dependent g-value influence g(Θ)

As been mentioned a few times, the angle-dependent g-value was included during the whole modeling process. The reasoning behind this is now discussed. This dependency actually exists in the real world and was not including as a modeling step to remove undesirable effects. The angle-dependency was included as follows:

$$g\left(\Theta\right) = g_0 \cdot \left(1 - b_0 \cdot \left(\frac{1}{cos\left(\Theta\right)} - 1\right)\right) \tag{5.6}$$

g_0=0.31 and b_0=0.2. Θ is the angle between the normal projection of the window and the sun irradiation "azimuthally" angle, Θ is deduced from the actual time, when at 6 a.m. and p.m. the angle is 90°, and at 12 a.m. the angle is 0°, with 15° steps between each hour. For time-steps before 6 h a.m. or p.m., g(Θ) would be negative therefore the values are set to zero. In Figure 5.10, the shapes of the angle-dependent g-value and the constant g-value are show, with the zeroing outside of the 6 a.m., p.m. range. In each model the product of g·G was obtained; this dependency results in a dampening of the sun irradiance in the morning and evening. This makes the sun irradiation during summer, where the sun rises before 6 a.m. and sets after 6 p.m., effectless in the model. More sophisticated models exists which use equ.(5.6) only between a angle of ±60°, and between 60° to 90° switch to another equation which is smoother in the change to zero.

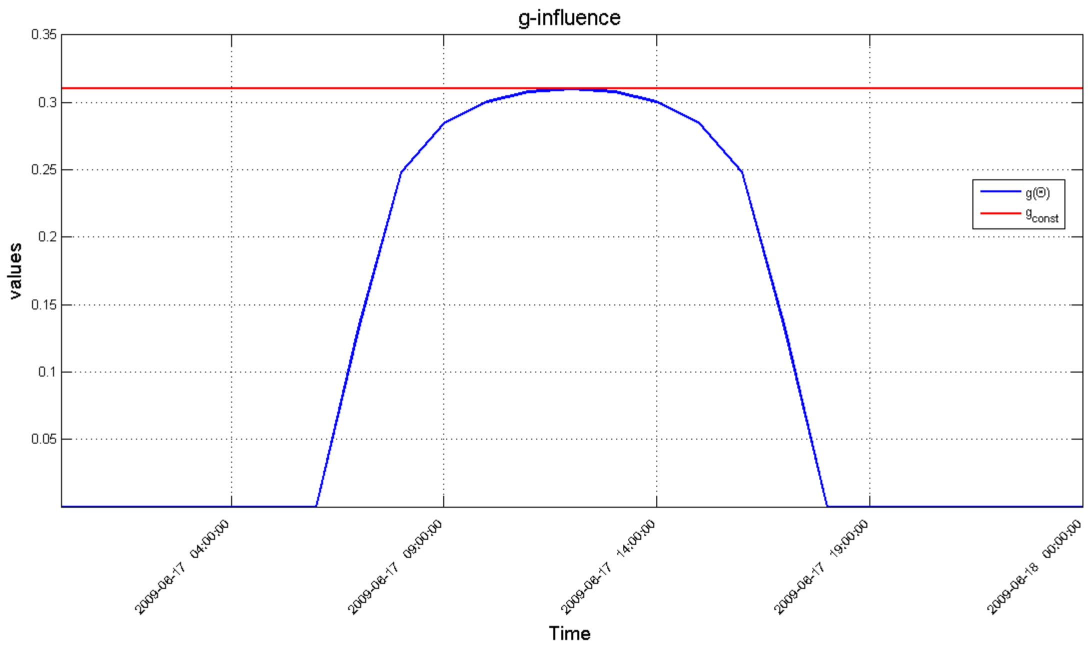


Figure 5.10: g(Θ)- and g_{const}-curve for 24h

By looking at Figure 5.9 it can be seen that there is still some parts of this effect left; during the morning and evening there are some small positive peaks, which would be worse with a constant g-value. The exact impact of this effect can be seen in Figure 5.11, where the light blue line is from the weighted time-shift with included g(Θ) model, and the green is with a constant g-value. To remove the morning and evening peak could possibly be achieved by increasing b_0 but as a result g_0 also has to be increased, otherwise the whole curve would be too low. Even so, there is still the problem of the second negative peak during the day which would increase a lot with an increasing b_0, which is why this was a good compromise.

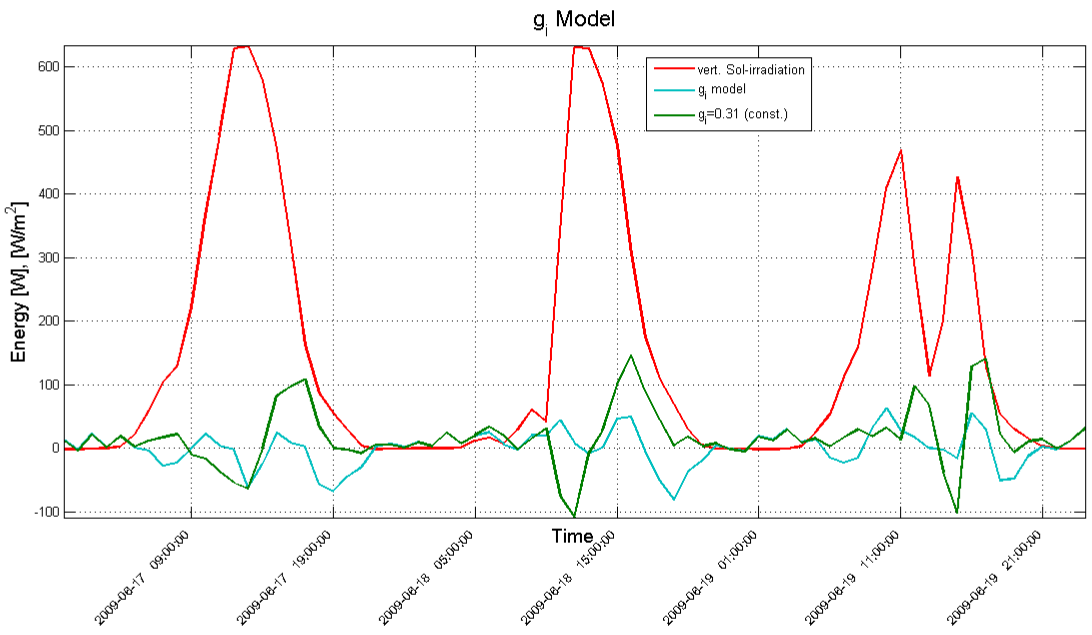


Figure 5.11: Influence of angle dependent g-value

5.2.5 Room comparison

Now that an acceptable model had been created, both rooms had to be compared. For this measurement in one room one and two 300 W electric heaters were placed. The point of this measurement was to determine the difference in cooling energy, if there is sun irradiation plus 300 W of heating in one room and only sun irradiation in the other room, the difference in cooling energy should measure 300 W. As illustrated Figure 5.12 there are a few steps: one 300 W heater, no heater, one 300 W again and two 300 W heaters (in that order). The objective of this was to figure out why there is a difference between the heating and cooling energy and what kind of error it is- a proportional or a constant one. If the cooling is used 1.1 times then it almost matches the heating energy. This pattern repeated on the measurements taken up to 1200 W. With this method of measurement it was not possible to figure out why the measurement is wrong and where the error is- if room 108, 109 or both have measurement errors.
To figure out where the error was, a 100 mm thick wall with a Styrofoam material (calibration wall) inside the south wall of room 109 was placed and the same done in 108, then the same kind of measurement carried out, but instead of using the difference in values between the rooms, only the absolute values of one room were used.

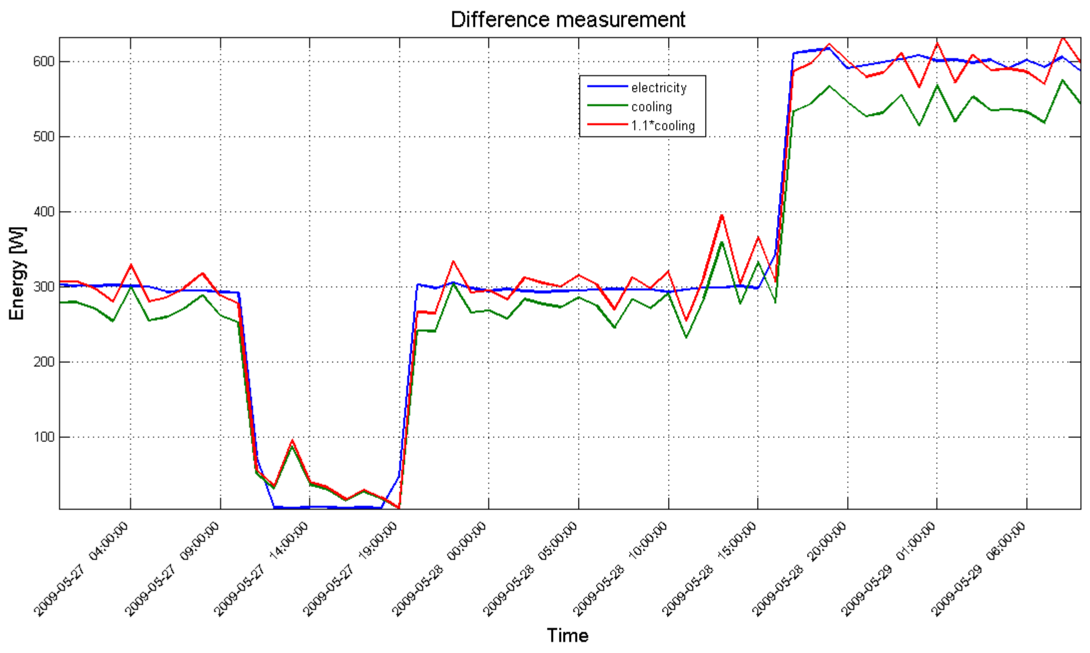


Figure 5.12: Difference measurement without calibration factors

The result of this measurement was, that room 108 required a calibration constant of 1.15 and room 109 1.1. With these two values, the measurement for the same period as that in Figure 5.12 was re-analyzed and the result shown in Figure 5.13. Both figures look almost the same with the difference being that even if the heater is removed, the heating and cooling energy fits better with the different calibration constants (red line at second step in both Figures 5.12 & 5.13).

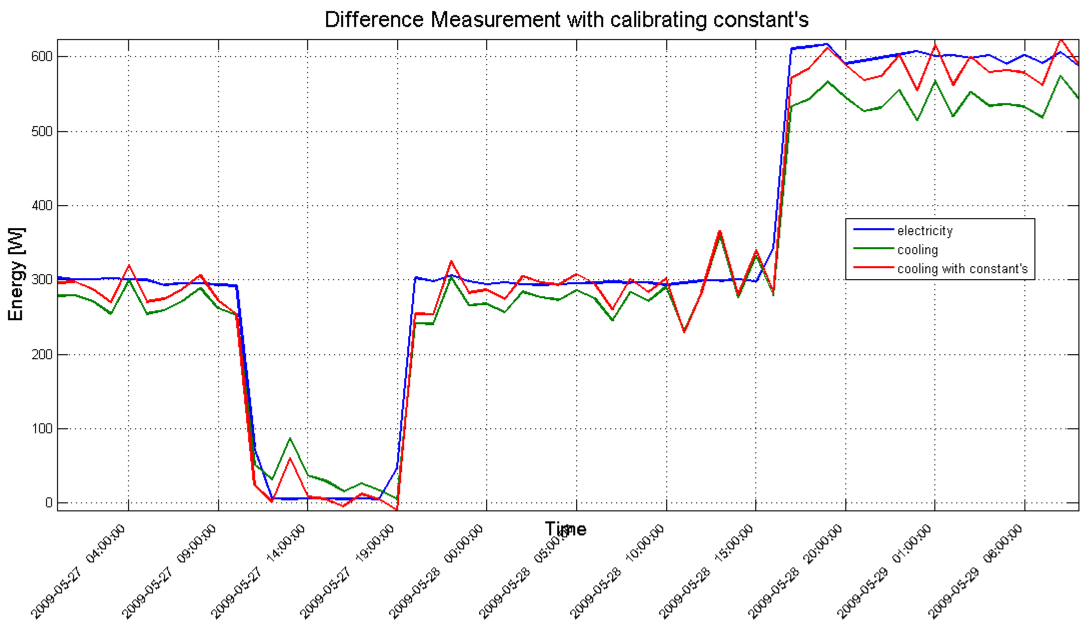


Figure 5.13: Difference measurement with calibration constants

It would of course be incorrect to say that the measurement was wrong and instead of finding the error, take the easy way out and include some parameters in the model, then tick this off as solved.
For the measurement of the cooling energy, two different types of sensors were used. One was a flow meter for water flow and the other a thermocouple for the inlet and outlet water temperature. To check if one of those sensors measured wrongly, the water flow was reduced by switching the circulation pump to a lower power mode. To get the same amount of cooling energy with reduced water flow, the temperature difference between the inlet and outlet must increase. If the temperature measurement measure wrong and because of the increased temperatures the influence of the error should decrease and this should decrease the % of cooling error, as in Figure 5.14 shown this wasn't the case. In the marked area which contained no changes other than water flow (less flow), this corresponded to an increasing difference in water temperature but the error was still the same.

There is also the MBUS system which measures the water flow with the same sensor as the LOGGER, but it use PT100 sensors for the measurement of inlet and outlet temperature. Comparing these two systems is a way of figuring out if only the LOGGER measured wrongly. The result is shown in Figure 5.15. Both systems measured exactly the same; in the first part there was around 1100 W of electric heaters plus 240 W for the circulation fan in the room, while in the second period there was only the fan. A problem of the water temperature measurement is that both systems (PT100, Thermocouple) measure dry, which means they do not directly measure the water temperature, but the copper temperature of the tube. Even if the sensors are well shielded from the surroundings, to avoid being affected by ambient air temperature, this cannot rule out that effect.

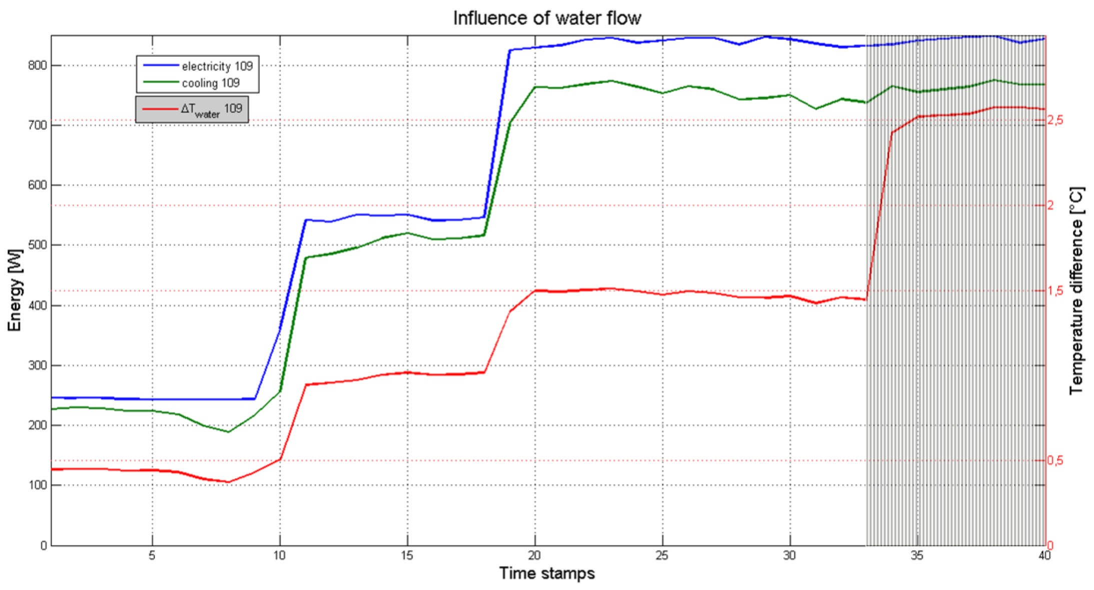


Figure 5.14: Influence of the water flow change on the measurement error

At this stage of the project, there was no other way rule out that effect the measurement error and use the calibration constants for the correction. Other steps would have been too time consuming, which is also why constants were included in the whole modeling process.

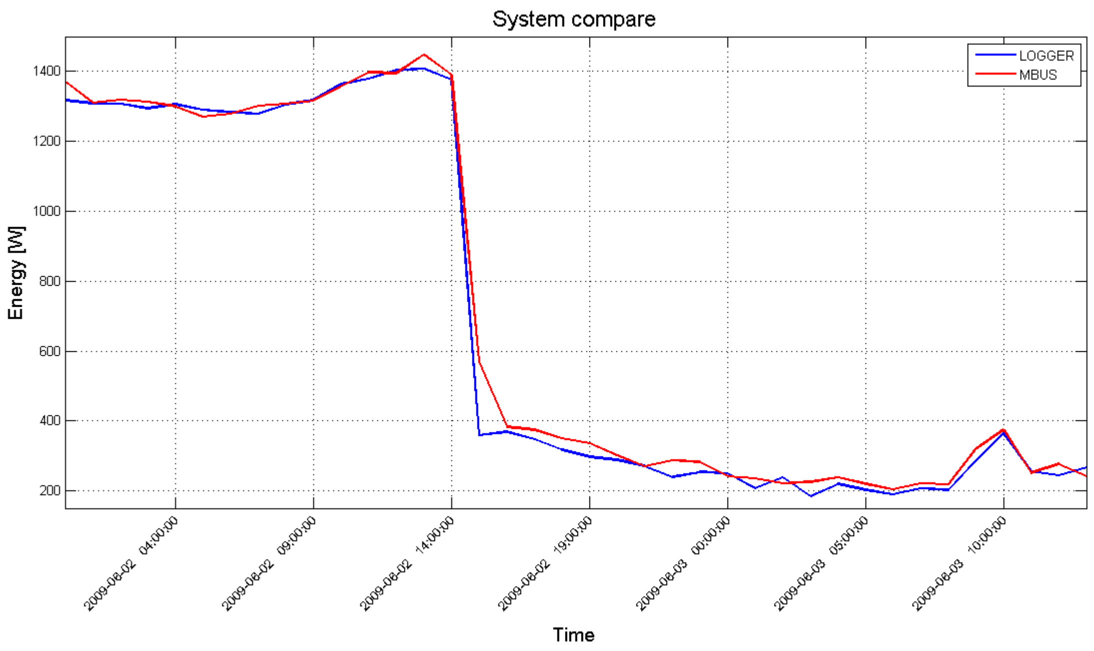


Figure 5.15: Comparison of cooling energy measurements between the MBUS and LOGGER systems

6 Parameter identification

Following the entire modeling process is the final step in this work, and the main purpose of the measurement system - to obtain the g-value and U-value. The comparison of measurements between the two rooms was the first step in this direction. At this stage of the project, only the process of identifying the g-value had been carried out via a comparison of the measurements of total energy between both rooms. There are currently eight different g-value calculations, the following three and three weighted version of this $\left(\frac{\sum g \cdot G}{\sum G}\right)$ and two for control purpose:

1. needed g-value only for room 108 that Q_{tot108} zero out
2. needed g-value only for room 109 that Q_{tot109} zero out
3. needed g-value for room 108 that the Q_{tot} difference between the rooms zero out

The third calculation is correlated to the first two because there is no difference if each room itself zero out, this would mean the difference is zero as well. Or if only the difference is calculated without taking care if Q_{tot} is zero. The equation for cases 1. and 2. is as follows:

$$\begin{aligned} Q_{tot} = 0 = multiple \cdot g_{i-1} \cdot (0.5 \cdot G_i + 0.5 \cdot G_{i-1}) \cdot A_{glass} \\ -U \cdot \Delta T \cdot A_{total} - Q_{cooling} + Q_{heating} - Q_{losses} \end{aligned} \tag{6.1}$$

and for case 3.:

$$\begin{aligned} Q_{diff} = 0 = g_{i-1} \cdot (0.5 \cdot G_i + 0.5 \cdot G_{i-1}) \cdot A_{glass} \\ -U \cdot \Delta T_{109} \cdot A_{total} - Q_{cooling109} + Q_{heating109} - Q_{losses109} \\ -(multiple \cdot g_{i-1} \cdot (0.5 \cdot G_i + 0.5 \cdot G_{i-1}) \cdot A_{glass} \\ -U \cdot \Delta T_{108} \cdot A_{total} - Q_{cooling108} + Q_{heating108} - Q_{losses108}) \end{aligned} \tag{6.2}$$

The solution with the multiple was chosen because it is easier to implement in terms of programming. Instead of calculating a new g-value for each hour, only one parameter needs to be calculated and the new g-value is the product of the original one and the multiple.
The Newton-Raphson algorithm was used to solve the equation and zero out the absolute value or difference as it is a quick and robust method as long as no local singularities. Figure 6.1 illustrates the schematic function of this algorithm. [4]

As an attempt to create a well defined scenario for g-value measurement, $\frac{1}{2}$ of the window size in one room was covered, as shown in Figure 6.2 a.). Each room contained six windows- two small ones at the top and bottom respectively which were covered with styrofoam to reduce the effect of sun irradiation; and two large windows in the middle. While the small windows at the top and bottom of both rooms were identical and permanent, the large windows were not.

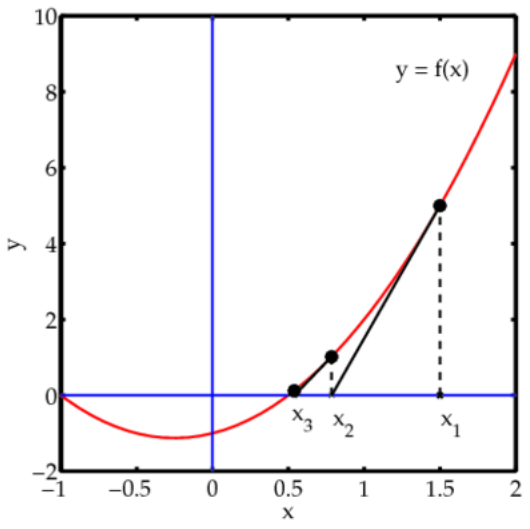


Figure 6.1: Schematic view of the Newton Raphson method

The large windows in one room were covered with aluminum foil on the outside, as aluminum foil has a high reflectance and the tight placement on the outside of the window should not affect the U-value; while the other had uncovered windows.

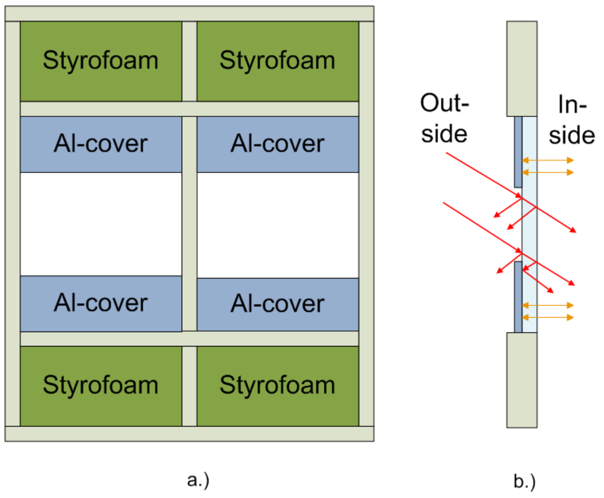


Figure 6.2: a.) Window layout, b.) Effect of aluminum covers

At the first view, it was mentioned that only half of the g-value should be measured if half of the window is transparent/uncovered. However, the measurement results shown in Figure 6.3 gave a multiple factor of ≈0.7 instead of 0.5. The blue line shows the reference room 109 with no covered windows. The brown line is the adjusted g-value of room 108, which is way too high during noon as the cooling only had a reduced amount of sun irradiation to take care of. If we take into account the fact that the model still considers that the entire window is uncovered with full sun irradiation, and make corresponding adjustments, we get the corrected Q_{tot} for room 108, illustrated by the green line. This gives a difference of zero between the blue and green lines. Before the adjustment, the deviation was 2.3 kWh for all 3 days; it was 0.62 Wh after.

Further analysis uncovered the reasons for this difference. Both panes of glass reflect some sunlight back outside, shown in Figure 6.2 b.), which illustrates two red arrows being reflected back out, such as in the case of a window not covered by aluminum foil. In the lower case the reflection from the inner glass is reflected back to the room. Furthermore, light reflected off the walls of the room is also reflected back into the rooms, represented by the orange arrows. This measurement only accounts for half of the glass but does not consider the aluminum frame of the windows. If the ratio of glass to frame is 70% to 30% and 50% of the glass is covered, this would correspond to 35% less sun irradiation. All these effects combined may result in the measured 30% less irradiation and explain 0.7 as a multiple.
As of this moment eight different g-values have been calculated, with three having been explained- the g-value zeroing out the total energy for each room (cases 1 and 2), and the third zeroing out the difference of Q_{tot}. Three of the remaining five are weighted g-values calculated using $\frac{\sum g \cdot G}{\sum G}$, which gives the g values around noon more weight. This is calculated for the difference as well as each room individually.

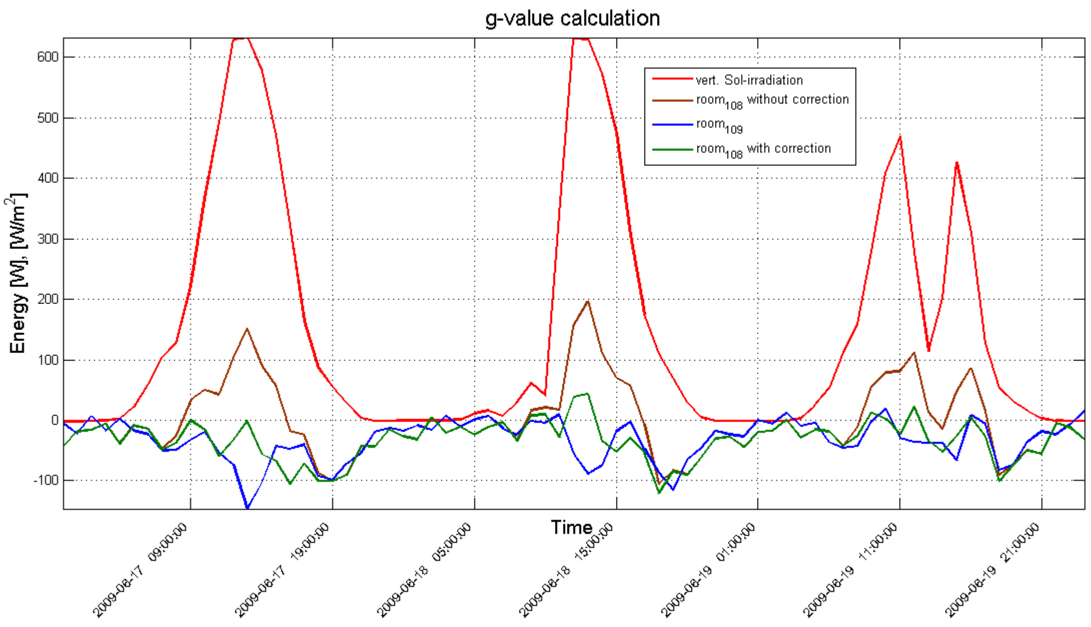


Figure 6.3: Results of the g-value calculation

7 Conclusion

The entire system consisted of many different parts, and it was a gradual process with many distinct steps/stages of getting each of them to work, and of eliminating the errors in each stage.
The whole project can be split into three steps. The first was the measurement system involved the measurement of various temperatures as well as the water flow and electricity consumption in the rooms. Going further would have been pointless if this part had not worked correctly.
The next and biggest step was the modeling, in which two variants to the basic model were evaluated: the capacity model and the time-shift model. The capacity model did not fulfill the necessary requirements and it was not feasible to develop it into a more sophisticated model that would give reliable results, hence no further work was done on it. Instead, the time-shift model was chosen for further development around the principle of time-shifts between sun irradiation and the cooling energy. Further development into the weighted shift version resulted in a good and acceptable model which zeroed out most, except a few remaining peaks.
Besides these two models there were two other findings during the modeling process. One was the angle-dependent g-value, g(Θ) and the other was an error in the measurement of cooling energy, found towards the end of the modeling process during the comparison of both rooms. The systems had measured inaccurately by 10% (room 108) and 15% (room 109). The assumption that room temperature is stable in the long run means that only the measurement could have been wrong, otherwise the temperature would have slowly increased. significant amount of effort was spent on finding the error and even a comparison between the MBUS and LOGGER systems could not unearth where the error lay. This is also why constants were used in the entire modeling process.
The third and last step was the parameterization of the g-value, where the Newthon-Raphson method was used. To date, eight different g-values have been calculated: one to zero out the difference in the total energy between the rooms, two values to zero out each individual room, three were weighted version of the previous g-values and the last two were only for control purposes. How the weighted version differs is that the sum of g·G is divided by the sum of G for the period analyzed, which gives more weight to the g-value around noon.
A possible future development should be around the measurement of the inlet and outlet water temperature. Currently, both the MBUS and LOGGER systems measure with different sensors (PT100 and thermocouples) but both measure dry, which means that the water temperature is not measured directly, but instead taken using the temperature of the copper tube. Even if the tubes are well insulated, there is no way of being sure that the surrounding environment has no effect on it. Therefore a wet sensor should be trailed at some stage, by exposing it directly to the water flow and hence directly measuring the water temperature. If a difference is found between the wet sensor reading and that of the dry sensor, then the origin of the error has been identified; if there is no difference, the only other possible location of the error lies in the water flow sensor.
To even further develop the model, a specialised version which takes into account more physical aspects of a specific room could be a possible option. Combining the capacity and time-shift

models could be a way of removing two effects at the same time - the stored energy in surfaces and the time-shift between irradiation and cooling. The most important part is the U-value, which has not been included thus far and the parameterization of the measuring object is the purpose of the rooms. Therefore a way of obtaining this value was devised. The inside and outside glass temperature is measured the whole time, during the night when there is no sunshine, the difference of the glass temperature between the rooms can give information of the U-value. This has to be incorporated process so that the U-value is calculated first, then followed by calculation of the g-value using the Newton-Raphson method.

Bibliography

[1] P.H. Baker and H.A.L. VAN DIJK. Paslink and dynamic outdoor testing of building components. *Building and Environment*, 43:143–151, 2008.

[2] Elena Palomo Del Barrio and Gilles Guyon. Theoretical basis for empirical model validation using parameters space analysis tools. *Energy and Buildings*, 35:985–996, 2003.

[3] Helena Bülow-Hübe, Håkan Håkansson, and Bengt Perers. New full-scale solar and building energy laboratory at lund university.

[4] Inc. Campbell Scientific. *CR1000 Measurement and Control System.* http://www.campbellsci.com/19_2_103.

[5] H.A.L. VAN DIJK and G.P. VAN DER LINDEN. The passys method for testing passive solar components. *Building and Environment*, 28:115–126, 1993.

[6] Erich Hahne and Rainer Pfluger. Improvements on passys test cells. *Solar Energy*, 58:239–246, 1996.

[7] M. J. Jimenez and M.R. Heras. Application of multi-output arx models for estimation of the u and g values of buildings components in outdoor testing. *Solar Energy*, 79:302–310, 2005.

[8] Maria Wall Helena and Bülow-Hübe. *Solar Protection in Buildings.* Lund University, Lund Institute of Technology, Division of Energy and Building Design, 2001.

[9] MathWorks. *MATLAB 7 Getting Started Guide.* http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf.

[10] M.E. McEvooy, R.G. Southall, and P.H. Baker. Test cell evaluation of supply air windows to characterise their optimum performance and its verification by the use of modelling techniques. *Energy and Buildings*, 35:1009–1020, 2003.

[11] P.A. Strachen and L. Vandaele. Case studies of outdoor testing and analysis of building components. *Building and Environment*, 43:129–142, 2008.

[12] P. Wouters, L. Vandale, P. Voit, and N. Fisch. The use of outdoor test cells for thermal and solar building research within the passys project. *Energy and Buildings*, 28:107–113, 1993.

List of Figures

Appendix A LOGGER table $6\,\mathrm{min.}$ and $1\,\mathrm{h}$

Table position	LOGGER name	Description
1	RECNBR	number of data set
2	T8luft_Avg	arithmetic average room temperature room 108
3	T8glas_Avg	arithmetic average of inside glass temperature room 108
4	DT8glas_Avg	glass temp. diff. between inside & outside middle level 108
5	T9luft_Avg	arithmetic average room temperature room 109
6	T9glas_Avg	arithmetic average of inside glass temperature room 108
7	DT9glas_Avg	glass temp. diff. between inside & outside middle level 109
8	Q108_Avg	heat losses through all surfaces except south wall room 108
9	Q109_Avg	heat losses through all surfaces except south wall room 108
10	TluftUte_Avg	outside air temperature
11	RefT_Avg	reference temperature for thermocouples
12	VertSol_Avg	vertical sun irradiation
13	VertSolDiff_Avg	diffuse sun irradiation
14	VertSolMark_Avg	infrared ground irradiation
15	Pyrgeometer_Avg	vertical infrared radiation
16	PyrgeTest_Avg	measured value without const. multipl. for vert. infr. rad. [mV]
17	Wflow108_Avg	water flow for coolingwater room 108
18	Wflow109_Avg	water flow for coolingwater room 109
19	Qkyla108_Avg	cooling energy room 108
20	Qkyla109_Avg	cooling energy room 109
21	El108_Avg	total electricity consumption room 108
22	El109_Avg	total electricity consumption room 109
23	Q_tempWall_Avg	
24	Q8Ceil_Avg	heatlosses through ceiling room 108
25	Q8Floor_Avg	heatlosses through floor room 108
26	Q8WWall_Avg	heatlosses through west wall room 108
27	Q8EWall_Avg	heatlosses through east wall room 108
28	Q9Ceil_Avg	heatlosses through ceiling room 109
29	Q9Floor_Avg	heatlosses through floor room 109
30	Q9WWall_Avg	heatlosses through west wall room 109
31	Q9EWall_Avg	heatlosses through east wall room 109
32	LoggerT_Avg	LOGGER temperature
33	T8luftUpp_Avg	air temperature 34 cm below the ceiling room 108
34	T8luftMitt_Avg	air temperature in the middle of the high room 108
35	T8luftNert_Avg	air temperature 57 cm above the floor room 108

$\vdots$ symbolises that the range between 51-60, 61-70, 71-80 and 83-90 is not used;
the first range is reserved for more sensor values in the rooms; and
the other spaces are for values in a more sophisticated model.

Table position	LOGGER name	Description
36	T8glasInMitt_Avg	glass temperature at the inside middle window room 108
37	T8glasInNert_Avg	glass temperature at the inside botom window room 108
38	T8glasUtMitt_Avg	glass temperature at the outside middle window room 108
39	T9luftUpp_Avg	air temperature 34 cm below the ceiling room 109
40	T9luftMitt_Avg	air temperature in the middle of the high room 108
41	T9luftNer_Avg	air temperature 57 cm above the floor room 109
42	T9glasInUpp_Avg	temperature at the top window inside at the glas room 109
43	T9glasInMitt_Avg	glass temperature at the inside middle window room 109
44	T9glasInNert_Avg	glass temperature at the inside botom window room 109
45	T9glasUtmitt_Avg	glass temperature at the outside middle window room 109
46	T_w108ut_Avg	outlet cooling water temperature room 108
47	T_w108in_Avg	inlet cooling water temperature room 108
48	T_w109ut_Avg	outlet cooling water temperature room 109
49	T_w109in_Avg	inlet cooling water temperature room 109
50	DeltaTW8_Avg	ΔT between in and outlet of cooling water room 108
51	DeltaTW9_Avg	ΔT between in and outlet of cooling water room 109
$\vdots$	$\vdots$	$\vdots$
60	$\cdots$	$g_{(i-1)} \cdot (0.5 \cdot G_i + 0.5 \cdot G_{i-1}) \cdot A_{glass} - U \cdot \Delta T \cdot A_{total}$ room 108
61	$\cdots$	Q_{tot} weighted time-shift model room 108
$\vdots$	$\vdots$	$\vdots$
70	$\cdots$	$g_{(i-1)} \cdot (0.5 \cdot G_i + 0.5 \cdot G_{i-1}) \cdot A_{glass} - U \cdot \Delta T \cdot A_{total}$ room 109
71	$\cdots$	Q_{tot} weighted time-shift model room 109
$\vdots$	$\vdots$	$\vdots$
80	$\cdots$	$Q_{tot} = g \cdot G \cdot A_{glass} - U \cdot \Delta T \cdot A_{total}$ room 108
81	$\cdots$	Q_{tot} capacity model room 108
82	$\cdots$	stored energy in surfaces, except south wall room 108
83	$\cdots$	Q_{tot}+stored energy room 108
$\vdots$	$\vdots$	$\vdots$
90	$\cdots$	$Q_{tot} = g \cdot G \cdot A_{glass} - U \cdot \Delta T \cdot A_{total}$ room 109
91	$\cdots$	Q_{tot} capacity model room 109
92	$\cdots$	stored energy in surfaces, except south wall room 109
93	$\cdots$	Q_{tot}+stored energy room 109

Table A.1: LOGGER data table (6 min. and 1 h)

Appendix B MATLAB Code

B.1 Main program ("calorimain")

```
%Author: Markus Heimberger
%June 2009
function varargout = calorimain_export(varargin)
% CALORIMAIN_EXPORT M-file for calorimain_export.fig
%      CALORIMAIN_EXPORT, by itself, creates a new CALORIMAIN_EXPORT or raises
    the existing
%      singleton*.
%
%      H = CALORIMAIN_EXPORT returns the handle to a new CALORIMAIN_EXPORT or the
     handle to
%      the existing singleton*.
%
%      CALORIMAIN_EXPORT('CALLBACK',hObject,eventData,handles,...) calls the
    local
%      function named CALLBACK in CALORIMAIN_EXPORT.M with the given input
    arguments.
%
%      CALORIMAIN_EXPORT('Property','Value',...) creates a new CALORIMAIN_EXPORT
    or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before calorimain_export_OpeningFunction gets called.
    An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to calorimain_export_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help calorimain_export

% Last Modified by GUIDE v2.5 27-Jul-2009 10:12:55

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calorimain_export_OpeningFcn, ...
                   'gui_OutputFcn',  @calorimain_export_OutputFcn, ...
                   'gui_LayoutFcn',  @calorimain_export_LayoutFcn, ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before calorimain_export is made visible.
function calorimain_export_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to calorimain_export (see VARARGIN)

% Choose default command line output for calorimain_export
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes calorimain_export wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = calorimain_export_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%Debug switch hidden fields for debuging on
handles.Debug= 'Off';
guidata(hObject, handles);


% --- Executes on button press in open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%==================================================================
%  reading file and store LOGGER data in different variables
%  Date and time is stored in handles.Date
%  all the sensor datas are stored in handles.DataTable

 [FileName,PathName,FilterIndex] =uigetfile('D:/Lund/Master_Thesis/Logger_Data/*.
     txt','Open_Logger-file');%default file path and file type

 %if file selection is cancelled, pathname should be zero
 %and nothing should happen
 if PathName == 0
```

```
        return
    end

    set(handles.selectedFile,'String',[PathName,FileName]);%set path and file-name
        on GUI (selected filepath)

    FID = fopen([PathName,FileName]);
    headerString1 = fscanf(FID,'%s/n'); % reads header data into a string first
        header line
    headerString2 = fscanf(FID,'%s/n'); % reads header data into a string second
        header line
    size(findstr(headerString2,'"'));

    Header_1 =strsplit( ',',headerString1,'omit'); %split header string into
        separate parts
    handles.Header_2 =strsplit( ',',headerString2,'omit');

    i=1;
    run=1;
    while run
        Date_Time=fscanf(FID, '%s/n'); %format of Date_Time e.g.: "2009-07-02
        if strcmp(Date_Time,'')          %break if Date_Time is empty
          break;
        else
            handles.Day(i,:)= Date_Time(length(Date_Time)-9:length(Date_Time)); % e.g
                .: 2009-07-02
        end
      Date_Time=fscanf(FID, '%s/n'); %format of Date_Time now
          06:00:00",12.23,124.123,....
      handles.Time(i,:)=Date_Time(1:8); %take only the first 8 charcters of
          Date_Time e.g.:06:00:00
      i=i+1;
    end

    handles.Date=strcat(handles.Day,{'␣␣␣'},handles.Time()); %put Day and Time
        together to one string
    handles.TableData=csvread([PathName,FileName],2,1); %read LOGGER data into
        TableData start at 3'rd row and 2'nd column
    guidata(hObject, handles); %store all handles at hObject that other functions
        have access
    fclose(FID);
    % end of read file
    %===================================================================

    set(handles.listb,'String',handles.Header_2(3:length(handles.Header_2)));%show
        avaible LOGGER data in listbox (Available LOGGER Data)


    % --- Executes on selection change in listbox1.
    function listbox1_Callback(handleToListbox, eventdata, handles)
    % hObject    handle to listbox1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
    %        contents{get(hObject,'Value')} returns selected item from listbox1
    vals = get(handles.handleToListbox,'Value');

```

```
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(handleToListbox, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in ReadListBox.
function ReadListBox_Callback(hObject, eventdata, handles)
 %liste=get(handleToListbox,'String');
 %[msg{1:numel(vals),1}]=liste{vals};
 %msgbox(msg))
% hObject    handle to ReadListBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.vals = get(handles.listb,'Value'); %read selected parameters from
    Available LOGGER Data listbox
plot(handles.TableData(:,handles.vals+1),'LineWidth', 2); %plot selcted Data in
    left plot


guidata(hObject,handles);%store all new handles in hObject

 %function call for modeling calculation
 %time-shift and capacity model
 %##########################################################################
 %##########################################################################
 %##########################################################################
 if (get(handles.Energy,'Value') == get(handles.Energy,'Max'))
    Modeling(hObject);
 end
 %##########################################################################
 %##########################################################################
 %##########################################################################

 handles=guidata(hObject); %read the new calculated handles from hObject
                            %e.g.60 or 61 and store them in handles


 %======================================================================
 % plot formatting for smal plot on left side
 set(handles.axes1,'XGrid','on','YGrid','on');
 axis tight; %plot range to max and min value of Data
 tick1=get(handles.axes1, 'XTick');
 set(handles.axes1,'XTickLabel',handles.Date(tick1,:)); %set Date as X lable
 xticklabel_rotate([],[45,80,150,500,450],[],'Fontsize',10) %rotate X lable 45°
 legendObject = legend( handles.Header_2(handles.vals+2),'Location','Best');
 title('Measured Data');
 xlabel('Time');
 ylabel('Measured Value');
 % END plot formatting for smal plot on left side
 %==========================================================================

 set(handles.listbDate,'String',handles.Date);  %show date & time in listbox to
     select 2 for separate print
 guidata(hObject, handles);


% --- Executes on selection change in listb.
function listb_Callback(hObject, eventdata, handles)
% hObject    handle to listb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listb contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listb


% --- Executes during object creation, after setting all properties.
function listb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1
%set(hObject,'String',vals,'Visible',handles.Debug);


% --- Executes on selection change in listbDate.
function listbDate_Callback(hObject, eventdata, handles)
% hObject    handle to listbDate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbDate contents as cell
    array
%        contents{get(hObject,'Value')} returns selected item from listbDate

%==========================================================================
%drawing of red vertical lines for time periode selction

VLines = findobj('type','line','Marker','none','LineStyle','--'); %find red
    vertical lines in smal plot
delete(VLines);   %delete this lines
Range=get(hObject,'Value'); %read selected value of Date listbox
```

```
YRange=get(handles.axes1,'YLim');
Rsize=size(Range);

for i=1:Rsize(1,2)
 line([Range(1,i), Range(1,i)],YRange,'Color', 'r', 'Linewidth', 2, 'Linestyle','
    --'); %plot new red lines
end

%END drawing of red vertical lines for time periode selction
%==========================================================================


% --- Executes during object creation, after setting all properties.
function listbDate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbDate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



%==========================================================================
%print figure in new window

% --- Executes on button press in SepFigure.
function SepFigure_Callback(hObject, eventdata, handles)
% hObject    handle to SepFigure (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ChosenDate=get(handles.listbDate,'Value'); %read selected Date in listbox
A=size(ChosenDate);
if A(1,2) ~= 2   %control if exactly 2 Dates are selected
 msgboxText{1} =  'You have not chosen 2 values';
 msgbox(msgboxText,'Input not allowed', 'error');
else             %if 2 are selected print in separate plot
 scrsz = get(0,'ScreenSize');
 SepPrint=figure('Position',[1 scrsz(2) scrsz(3) scrsz(4)-70]);
 SepAxes=axes('Units','pixels','Position',[1+80 scrsz(2)+80 scrsz(3)-150 scrsz(4)
    -70-150]);

 guidata(hObject,handles);%store all new handles in hObject

 %function call for g and U Value calculation
 %for time shift model only
 %##########################################################################
 %##########################################################################
 %##########################################################################
 if (get(handles.Energy,'Value') == get(handles.Energy,'Max')) %if radiobutton
    Model's & Calculation is selected call function Calculation and calculate
    different Model's
    if (get(handles.Calc,'Value') == get(handles.Calc,'Max'))
     Calculation(hObject,ChosenDate);
    end
```

```
%#########################################################################
%#########################################################################
%#########################################################################
  handles=guidata(hObject); %read the new calculated handles from hObject
                            %e.g.g or U value

  %======================================================================
  % DIFFERENT PLOT (separate window)

    plot([handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12)...
          handles.TableData(ChosenDate(1,1):ChosenDate(1,2),61)...
          handles.TableData(ChosenDate(1,1):ChosenDate(1,2),71),...
          ],'LineWidth', 2);
    legend('vert.␣Sol-irradiation','Q_{tot}␣108','Q_{tot}␣108','Location','Best');
        %default legend entries
  else
     SepPlot=plot('v6',handles.TableData(ChosenDate(1,1):ChosenDate(1,2),handles.
         vals+1),'LineWidth', 2);
     legendObject = legend( handles.Header_2(handles.vals+2),'Location','Best');
  end

  set(SepAxes,'XGrid','on','YGrid','on');
  axis tight
  tick1=get(SepAxes, 'XTick');

  set(SepAxes,'XTickLabel',handles.Date(tick1+ChosenDate(1,1)-1,:));
  xticklabel_rotate([],45,[],'Fontsize',10)

  title('Measured␣Data','Fontsize', 16);
  xlabel('Time','Fontsize', 14);
  ylabel('Energy␣[W],␣[W/m^2]','Fontsize', 14);

end
%END DIFFERENT PLOT'S
%======================================================================


% --- Executes on button press in Calc.
function Calc_Callback(hObject, eventdata, handles)
% hObject    handle to Calc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Calc


% --- Executes on button press in Energy.
function Energy_Callback(hObject, eventdata, handles)
% hObject    handle to Energy (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Energy



% --- Creates and returns a handle to the GUI figure.
function h1 = calorimain_export_LayoutFcn(policy)
% policy - create a new figure or use a singleton. 'new' or 'reuse'.
```

```

persistent hsingleton;
if strcmpi(policy, 'reuse') & ishandle(hsingleton)
    h1 = hsingleton;
    return;
end

appdata = [];
appdata.GUIDEOptions = struct(...
    'active_h', [], ...
    'taginfo', struct(...
    'figure', 2, ...
    'pushbutton', 4, ...
    'listbox', 5, ...
    'text', 6, ...
    'axes', 2, ...
    'uipanel', 7, ...
    'radiobutton', 3, ...
    'edit', 2), ...
    'override', 0, ...
    'release', 13, ...
    'resize', 'none', ...
    'accessibility', 'callback', ...
    'mfile', 1, ...
    'callbacks', 1, ...
    'singleton', 1, ...
    'syscolorfig', 1, ...
    'blocking', 0, ...
    'lastSavedFile', 'C:\Dokumente_und_Einstellungen\heimbe\Desktop\Thesis_Latex\
        Markus\appendix\calorimain_export.m');
appdata.lastValidTag = 'figure1';
appdata.GUIDELayoutEditor = [];

h1 = figure(...
'Units','characters',...
'Color',[0.831372549019608 0.815686274509804 0.784313725490196],...
'Colormap',[0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;0 0 0.8125;0 0 0.875;0 0
    0.9375;0 0 1;0 0.0625 1;0 0.125 1;0 0.1875 1;0 0.25 1;0 0.3125 1;0 0.375 1;0
    0.4375 1;0 0.5 1;0 0.5625 1;0 0.625 1;0 0.6875 1;0 0.75 1;0 0.8125 1;0 0.875
    1;0 0.9375 1;0 1 1;0.0625 1 1;0.125 1 0.9375;0.1875 1 0.875;0.25 1
    0.8125;0.3125 1 0.75;0.375 1 0.6875;0.4375 1 0.625;0.5 1 0.5625;0.5625 1
    0.5;0.625 1 0.4375;0.6875 1 0.375;0.75 1 0.3125;0.8125 1 0.25;0.875 1
    0.1875;0.9375 1 0.125;1 1 0.0625;1 1 0;1 0.9375 0;1 0.875 0;1 0.8125 0;1 0.75
     0;1 0.6875 0;1 0.625 0;1 0.5625 0;1 0.5 0;1 0.4375 0;1 0.375 0;1 0.3125 0;1
    0.25 0;1 0.1875 0;1 0.125 0;1 0.0625 0;1 0 0;0.9375 0 0;0.875 0 0;0.8125 0
    0;0.75 0 0;0.6875 0 0;0.625 0 0;0.5625 0 0],...
'IntegerHandle','off',...
'InvertHardcopy',get(0,'defaultfigureInvertHardcopy'),...
'MenuBar','none',...
'Name','calorimain',...
'NumberTitle','off',...
'PaperPosition',get(0,'defaultfigurePaperPosition'),...
'Position',[103.8 9.00000000000002 241 52.4615384615385],...
'Resize','off',...
'HandleVisibility','callback',...
'Tag','figure1',...
'UserData',[],...
'Behavior',get(0,'defaultfigureBehavior'),...
'Visible','on',...
```

```
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'open';

h2 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','calorimain_export(''open_Callback'',gcbo,[],guidata(gcbo))',...
'Position',[184 47.8461538461538 15.4 2.84615384615385],...
'String','open',...
'Tag','open',...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'selectedFile';

h3 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[170.2 43.1538461538462 43 4.15384615384615],...
'String','Selected Filepath',...
'Style','text',...
'Tag','selectedFile',...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'uipanel2';

h4 = uipanel(...
'Parent',h1,...
'Units','characters',...
'Title','Read',...
'Tag','uipanel2',...
'UserData',[],...
'Behavior',get(0,'defaultuipanelBehavior'),...
'Clipping','on',...
'Position',[168.8 42.7692307692308 45.6 9.61538461538462],...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'listbDate';

h5 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback','calorimain_export(''listbDate_Callback'',gcbo,[],guidata(gcbo))',...
'Max',3,...
'Min',1,...
'Position',[133.6 12.7692307692308 30.2 34.6923076923077],...
'String',{ 'Listbox' },...
'Style','listbox',...
'Value',1,...
'CreateFcn', {@local_CreateFcn, 'calorimain_export(''listbDate_CreateFcn'',gcbo
    ,[],guidata(gcbo))', appdata} ,...
'Tag','listbDate',...
```

```
'Behavior',get(0,'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'uipanel4';

h6 = uipanel(...
'Parent',h1,...
'Units','characters',...
'Title','PRINT',...
'Tag','uipanel4',...
'UserData',[],...
'Behavior',get(0,'defaultuipanelBehavior'),...
'Clipping','on',...
'Position',[-0.2 3.46153846153846 166.2 48.9230769230769],...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'axes1';

h7 = axes(...
'Parent',h6,...
'Units','characters',...
'Position',[5.4 1.84615384615385 124 45.9230769230769],...
'CameraPosition',[0.5 0.5 9.16025403784439],...
'CameraPositionMode',get(0,'defaultaxesCameraPositionMode'),...
'Color',get(0,'defaultaxesColor'),...
'ColorOrder',get(0,'defaultaxesColorOrder'),...
'LooseInset',[30.966 5.74538461538462 22.629 3.91730769230769],...
'XColor',get(0,'defaultaxesXColor'),...
'YColor',get(0,'defaultaxesYColor'),...
'ZColor',get(0,'defaultaxesZColor'),...
'CreateFcn', {@local_CreateFcn, 'calorimain_export(''axes1_CreateFcn'',gcbo,[],
    guidata(gcbo))', appdata} ,...
'Tag','axes1',...
'Behavior',get(0,'defaultaxesBehavior'));

h8 = get(h7,'title');

set(h8,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.499193548387097 1.0108877721943 1.00005459937205],...
'Rotation',0,...
'String','',...
'Interpreter','tex',...
```

```
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey','',...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag','',...
'UserData',[],...
'Behavior',struct(),...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'Clipping','off');

h9 = get(h7,'xlabel');

set(h9,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.499193548387097 -0.0393634840871022 1.00005459937205],...
'Rotation',0,...
'String','',...
'Interpreter','tex',...
'VerticalAlignment','cap',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey','',...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag','',...
'UserData',[],...
'Behavior',struct(),...
```

```
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'Clipping','off');

h10 = get(h7,'ylabel');

set(h10,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0459677419354839 0.498324958123953 1.00005459937205],...
'Rotation',90,...
'String','',...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey','',...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag','',...
'UserData',[],...
'Behavior',struct(),...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'Clipping','off');

h11 = get(h7,'zlabel');

set(h11,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
```

```
'BackgroundColor','none',...
'Color',[0 0 0],...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','right',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0459677419354839 1.02093802345059 1.00005459937205],...
'Rotation',0,...
'String','',...
'Interpreter','tex',...
'VerticalAlignment','middle',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey','',...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag','',...
'UserData',[],...
'Behavior',struct(),...
'Visible','off',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'Clipping','off');

appdata = [];
appdata.lastValidTag = 'text4';

h12 = uicontrol(...
'Parent',h6,...
'Units','characters',...
'Position',[134.8 44 26.4 2.15384615384615],...
'String','Chose Start and Stop Date',...
'Style','text',...
'Tag','text4',...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'SepFigure';

h13 = uicontrol(...
'Parent',h6,...
'Units','characters',...
'Callback','calorimain_export(''SepFigure_Callback'',gcbo,[],guidata(gcbo))',...
```

```
'Position',[136.4 4.53846153846154 23.2 2.76923076923077],...
'String','Separate_Figure',...
'Tag','SepFigure',...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'listb';

h14 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback','calorimain_export(''listb_Callback'',gcbo,[],guidata(gcbo))',...
'CData',[],...
'Max',5,...
'Min',1,...
'Position',[178.6 14.7692307692308 26.2 23.1538461538462],...
'String',{ 'Available'; 'LOGGER'; 'Data' },...
'Style','listbox',...
'Value',1,...
'CreateFcn', {@local_CreateFcn, 'calorimain_export(''listb_CreateFcn'',gcbo,[],
    guidata(gcbo))', appdata} ,...
'Tag','listb',...
'UserData',[],...
'Behavior',get(0,'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'ReadListBox';

h15 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback','calorimain_export(''ReadListBox_Callback'',gcbo,[],guidata(gcbo))'
    ,...
'CData',[],...
'Position',[181.6 4.53846153846154 20.2 3.92307692307692],...
'String','Read_Listbox',...
'Tag','ReadListBox',...
'UserData',[],...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'uipanel6';

h16 = uipanel(...
'Parent',h1,...
'Units','characters',...
'Title','Data',...
'Tag','uipanel6',...
'Behavior',get(0,'defaultuipanelBehavior'),...
'Clipping','on',...
'Position',[169.8 3.53846153846154 40.2 38.1538461538462],...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'Energy';

```

```
h17 = uicontrol(...
'Parent',h16,...
'Units','characters',...
'Callback','calorimain_export(''Energy_Callback'',gcbo,[],guidata(gcbo))',...
'Position',[4.6 7.61538461538462 24.6 2.69230769230769],...
'String','Model''s',...
'Style','radiobutton',...
'Tag','Energy',...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );

appdata = [];
appdata.lastValidTag = 'Calc';

h18 = uicontrol(...
'Parent',h16,...
'Units','characters',...
'Callback','calorimain_export(''Calc_Callback'',gcbo,[],guidata(gcbo))',...
'Position',[4.6 5.23076923076923 32.8 2.38461538461538],...
'String','Calculation of g and U-value',...
'Style','radiobutton',...
'Tag','Calc',...
'Behavior',get(0,'defaultuicontrolBehavior'),...
'CreateFcn', {@local_CreateFcn, '', appdata} );


hsingleton = h1;


% --- Set application data first then calling the CreateFcn.
function local_CreateFcn(hObject, eventdata, createfcn, appdata)

if ~isempty(appdata)
   names = fieldnames(appdata);
   for i=1:length(names)
       name = char(names(i));
       setappdata(hObject, name, getfield(appdata,name));
   end
end

if ~isempty(createfcn)
   eval(createfcn);
end


% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)


%   GUI_MAINFCN provides these command line APIs for dealing with GUIs
%
%      CALORIMAIN_EXPORT, by itself, creates a new CALORIMAIN_EXPORT or raises
    the existing
%      singleton*.
%
%      H = CALORIMAIN_EXPORT returns the handle to a new CALORIMAIN_EXPORT or the
     handle to
%      the existing singleton*.
%
```

```
%      CALORIMAIN_EXPORT('CALLBACK',hObject,eventData,handles,...) calls the
    local
%      function named CALLBACK in CALORIMAIN_EXPORT.M with the given input
    arguments.
%
%      CALORIMAIN_EXPORT('Property','Value',...) creates a new CALORIMAIN_EXPORT
    or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".

%   Copyright 1984-2005 The MathWorks, Inc.
%   $Revision: 1.4.6.12 $ $Date: 2005/06/21 19:41:16 $

gui_StateFields =  {'gui_Name'
                    'gui_Singleton'
                    'gui_OpeningFcn'
                    'gui_OutputFcn'
                    'gui_LayoutFcn'
                    'gui_Callback'};
gui_Mfile = '';
for i=1:length(gui_StateFields)
    if ~isfield(gui_State, gui_StateFields{i})
        error('Could not find field %s in the gui_State struct in GUI M-file %s',
             gui_StateFields{i}, gui_Mfile);
    elseif isequal(gui_StateFields{i}, 'gui_Name')
        gui_Mfile = [gui_State.(gui_StateFields{i}), '.m'];
    end
end

numargin = length(varargin);

if numargin == 0
    % CALORIMAIN_EXPORT
    % create the GUI
    gui_Create = 1;
elseif isequal(ishandle(varargin{1}), 1) && ispc && iscom(varargin{1}) && isequal
    (varargin{1},gcbo)
    % CALORIMAIN_EXPORT(ACTIVEX,...)
    vin{1} = gui_State.gui_Name;
    vin{2} = [get(varargin{1}.Peer, 'Tag'), '_', varargin{end}];
    vin{3} = varargin{1};
    vin{4} = varargin{end-1};
    vin{5} = guidata(varargin{1}.Peer);
    feval(vin{:});
    return;
elseif ischar(varargin{1}) && numargin>1 && isequal(ishandle(varargin{2}), 1)
    % CALORIMAIN_EXPORT('CALLBACK',hObject,eventData,handles,...)
    gui_Create = 0;
else
    % CALORIMAIN_EXPORT(...)
    % create the GUI and hand varargin to the openingfcn
    gui_Create = 1;
end
```

```
if gui_Create == 0
    varargin{1} = gui_State.gui_Callback;
    if nargout
        [varargout{1:nargout}] = feval(varargin{:});
    else
        feval(varargin{:});
    end
else
    if gui_State.gui_Singleton
        gui_SingletonOpt = 'reuse';
    else
        gui_SingletonOpt = 'new';
    end

    % Open fig file with stored settings.  Note: This executes all component
    % specific CreateFunctions with an empty HANDLES structure.

    % Do feval on layout code in m-file if it exists
    if ~isempty(gui_State.gui_LayoutFcn)
        gui_hFigure = feval(gui_State.gui_LayoutFcn, gui_SingletonOpt);
        % openfig (called by local_openfig below) does this for guis without
        % the LayoutFcn. Be sure to do it here so guis show up on screen.
        movegui(gui_hFigure,'onscreen')
    else
        gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
        % If the figure has InGUIInitialization it was not completely created
        % on the last pass.  Delete this handle and try again.
        if isappdata(gui_hFigure, 'InGUIInitialization')
            delete(gui_hFigure);
            gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
        end
    end

    % Set flag to indicate starting GUI initialization
    setappdata(gui_hFigure,'InGUIInitialization',1);

    % Fetch GUIDE Application options
    gui_Options = getappdata(gui_hFigure,'GUIDEOptions');

    if ~isappdata(gui_hFigure,'GUIOnScreen')
        % Adjust background color
        if gui_Options.syscolorfig
            set(gui_hFigure,'Color', get(0,'DefaultUicontrolBackgroundColor'));
        end

        % Generate HANDLES structure and store with GUIDATA. If there is
        % user set GUI data already, keep that also.
        data = guidata(gui_hFigure);
        handles = guihandles(gui_hFigure);
        if ~isempty(handles)
            if isempty(data)
                data = handles;
            else
                names = fieldnames(handles);
                for k=1:length(names)
                    data.(char(names(k)))=handles.(char(names(k)));
                end
            end
        end
        guidata(gui_hFigure, data);
    end

    % If user specified 'Visible','off' in p/v pairs, don't make the figure
    % visible.
    gui_MakeVisible = 1;
    for ind=1:2:length(varargin)
        if length(varargin) == ind
            break;
        end
        len1 = min(length('visible'),length(varargin{ind}));
        len2 = min(length('off'),length(varargin{ind+1}));
        if ischar(varargin{ind}) && ischar(varargin{ind+1}) && ...
                strncmpi(varargin{ind},'visible',len1) && len2 > 1
            if strncmpi(varargin{ind+1},'off',len2)
                gui_MakeVisible = 0;
            elseif strncmpi(varargin{ind+1},'on',len2)
                gui_MakeVisible = 1;
            end
        end
    end

    % Check for figure param value pairs
    for index=1:2:length(varargin)
        if length(varargin) == index || ~ischar(varargin{index})
            break;
        end
        try set(gui_hFigure, varargin{index}, varargin{index+1}), catch break,
           end
    end

    % If handle visibility is set to 'callback', turn it on until finished
    % with OpeningFcn
    gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
    if strcmp(gui_HandleVisibility, 'callback')
        set(gui_hFigure,'HandleVisibility', 'on');
    end

    feval(gui_State.gui_OpeningFcn, gui_hFigure, [], guidata(gui_hFigure),
        varargin{:});

    if ishandle(gui_hFigure)
        % Update handle visibility
        set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);

        % Make figure visible
        if gui_MakeVisible
            set(gui_hFigure, 'Visible', 'on')
            if gui_Options.singleton
                setappdata(gui_hFigure,'GUIOnScreen', 1);
            end
        end

        % Done with GUI initialization
        rmappdata(gui_hFigure,'InGUIInitialization');
    end

    % If handle visibility is set to 'callback', turn it on until finished with
    % OutputFcn
    if ishandle(gui_hFigure)
        gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
        if strcmp(gui_HandleVisibility, 'callback')
            set(gui_hFigure,'HandleVisibility', 'on');
        end
        gui_Handles = guidata(gui_hFigure);
    else
        gui_Handles = [];
    end

    if nargout
        [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn, gui_hFigure, [],
            gui_Handles);
    else
        feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
    end

    if ishandle(gui_hFigure)
        set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);
    end
end

function gui_hFigure = local_openfig(name, singleton)

% this application data is used to indicate the running mode of a GUIDE
% GUI to distinguish it from the design mode of the GUI in GUIDE.
setappdata(0,'OpenGuiWhenRunning',1);

% openfig with three arguments was new from R13. Try to call that first, if
% failed, try the old openfig.
try
    gui_hFigure = openfig(name, singleton, 'auto');
catch
    % OPENFIG did not accept 3rd input argument until R13,
    % toggle default figure visible to prevent the figure
    % from showing up too soon.
    gui_OldDefaultVisible = get(0,'defaultFigureVisible');
    set(0,'defaultFigureVisible','off');
    gui_hFigure = openfig(name, singleton);
    set(0,'defaultFigureVisible',gui_OldDefaultVisible);
end
rmappdata(0,'OpenGuiWhenRunning');
```

B.2 Modeling

```
function Modeling(hObject)

 handles=guidata(hObject);% read data from hObject and store it in handles

 %angle dependent g-value calculation
 %angle depending on time during day
 Hour=handles.Time(:,1:2);   %separate hour
 Minute=handles.Time(:,4:5);%separate minutes
 Hour=str2num(Hour);         %change hour string to number
 Minute=str2num(Minute)/60;
 Time=Hour+Minute;           %combine hour and minute
 deta=(12-Time)*15/180*pi;   %calculate delta depending on time
 gc=cos(deta);
 zero=find(gc < 0.001);      %replace 0 in gc, bc of 1/gc would give inf.
 gc(zero)=0.16667;           %0 in gc replaced by 0.167 bc for this equiation
     below gives almost zero
 handles.g=0.31*(1-0.2*(1./(gc)-1));    %g0=0.31; b0=0.2


      %Temperatures in wall, ceiling, floor #########################

      handles.T8_ceil= handles.TableData(:,24)/(6*15.13*40e-3); %6 bc of 6
          thermopiles in serie , 15.13 Håkans const in logger program (horizontal
          ), 40uV/K const of one thermopile
      handles.T8_floor=handles.TableData(:,25)/(6*15.13*40e-3);
      handles.T8_WWall=handles.TableData(:,26)/(8*13.15*40e-3); %in walls 8
          thermopiles in serie, and 13.15 as const in logger program
      handles.T8_EWall=handles.TableData(:,27)/(8*13.15*40e-3);

      handles.T9_ceil= handles.TableData(:,28)/(6*15.13*40e-3); %6 bc of 6
          thermopiles in serie , 15.13 Håkans const in logger program (horizontal
          ), 40uV/K const of one thermopile
      handles.T9_floor=handles.TableData(:,29)/(6*15.13*40e-3);
      handles.T9_WWall=handles.TableData(:,30)/(8*13.15*40e-3); %in walls 8
          thermopiles in serie, and 13.15 as const in logger program
      handles.T9_EWall=handles.TableData(:,31)/(8*13.15*40e-3);


     A=size(handles.TableData);
     %Temperature differents in wall, ceiling, floor #################
     for i=2:A(1)
         handles.DT8_ceil(i) = handles.T8_ceil(i)-handles.T8_ceil(i-1);
         handles.DT8_floor(i)= handles.T8_floor(i)-handles.T8_floor(i-1);
         handles.DT8_WWall(i)= handles.T8_WWall(i)-handles.T8_WWall(i-1);
         handles.DT8_EWall(i)= handles.T8_EWall(i)-handles.T8_EWall(i-1);
         handles.Glas_in8(i) = handles.TableData(i,36)-handles.TableData(i-1,36);
         handles.Glas_out8(i)= handles.TableData(i,38)-handles.TableData(i-1,38);

         handles.DT9_ceil(i) = handles.T9_ceil(i)-handles.T9_ceil(i-1);
         handles.DT9_floor(i)= handles.T9_floor(i)-handles.T9_floor(i-1);
         handles.DT9_WWall(i)= handles.T9_WWall(i)-handles.T9_WWall(i-1);
         handles.DT9_EWall(i)= handles.T9_EWall(i)-handles.T9_EWall(i-1);
         handles.Glas_in9(i) = handles.TableData(i,43)-handles.TableData(i-1,43);
         handles.Glas_out9(i)= handles.TableData(i,45)-handles.TableData(i-1,45);
     end
```

```
        %time shift modell #######################################
        for i=2:A(1)
            handles.TableData(i,60)= (handles.g(i-1)*2.667)*((handles.TableData(i
                -1,12))*0.5+(handles.TableData(i,12))*0.5)-1*(handles.TableData(i,2)
                -handles.TableData(i,10))*8.37; %A*g*G-U*deltaT*A timeshift included
                 0.5 old value 0.5 new value
            handles.TableData(i,70)=          (handles.g(i-1)*2.667)*((handles.
                TableData(i-1,12))*0.5+(handles.TableData(i,12))*0.5)-1*(handles.
                TableData(i,5)-handles.TableData(i,10))*8.37; %A*g*G-U*deltaT*A
                timeshift included 0.5 old value 0.5 new value
        end

        handles.TableData(:,61)=handles.TableData(:,60)+handles.TableData(:,21)-
            handles.TableData(:,19)*1.15+handles.TableData(:,8); %61... total energy
            108 1.15 calibration constant for cooling
        handles.TableData(:,71)=handles.TableData(:,70)+handles.TableData(:,22)-
            handles.TableData(:,20)*1.1+handles.TableData(:,9); %71... total energy
            109   1.1 calibration constant for cooling

        %end time shift modell ####################################

        %capacity modell #########################################
        handles.TableData(:,80)=(handles.g*2.667).*(handles.TableData(:,12))-1*(
            handles.TableData(:,2)-handles.TableData(:,10))*8.37; %A*g*G-U*deltaT*A
        handles.TableData(:,90)=(handles.g*2.667).*(handles.TableData(:,12))-1*(
            handles.TableData(:,5)-handles.TableData(:,10))*8.37; %A*g*G-U*deltaT*A

        handles.TableData(:,81)=handles.TableData(:,80)+handles.TableData(:,21)-
            handles.TableData(:,19)+handles.TableData(:,8); %81... total energy 108
        handles.TableData(:,91)=handles.TableData(:,90)+handles.TableData(:,22)-
            handles.TableData(:,20)+handles.TableData(:,9); %91... total energy 109

        handles.TableData(:,82)=((handles.DT8_ceil+handles.DT8_WWall+handles.
            DT8_EWall)'*40+ handles.DT8_floor'*40); %losses trough wall,floor,
            ceiling 108 times constant
        handles.TableData(:,92)=((handles.DT9_ceil+handles.DT9_WWall+handles.
            DT9_EWall)'*40+ handles.DT9_floor'*40); %losses trough wall,floor,
            ceiling 109 times constant

        handles.TableData(:,83)=handles.TableData(:,81)+handles.TableData(:,82); %
            total energy + losses trough the surfaces
        handles.TableData(:,93)=handles.TableData(:,91)+handles.TableData(:,92); %
            total energy + losses trough the surfaces

        %end time shift modell ####################################
        plot([handles.TableData(:,61),handles.TableData(:,71),handles.TableData
            (:,83),handles.TableData(:,93)], 'LineWidth', 2);

guidata(hObject,handles);
```

B.3 Calculations

```
function Calculation(hObject,ChosenDate)

handles=guidata(hObject);

%==========================================================================
    %calculation of g-value with Newton Raphson method
    %x_{i+1}=x_i-f(x)/f'(x)
    %avarage Qtot108-Qtot109 out to zero

        multiplier=1;
        old=0;
        A=size(handles.TableData);
        while abs(multiplier-old)>0.001
            old=multiplier;
             for i=2:A(1)
                handles.TableData(i,60)=multiplier*(handles.g(i-1)*2.667)*((handles.TableData(i-1,12))*0.5+(handles.TableData(i,12))*0.5)-1*(handles.TableData(i,2)-handles.TableData(i,10))*8.37; %A*g*G-U*deltaT*A timeshift included 0.5 old value 0.5 new value
                handles.TableData(i,70)=           (handles.g(i-1)*2.667)*((handles.TableData(i-1,12))*0.5+(handles.TableData(i,12))*0.5)-1*(handles.TableData(i,5)-handles.TableData(i,10))*8.37; %A*g*G-U*deltaT*A timeshift included 0.5 old value 0.5 new value
            end
            dif=sum((handles.TableData(ChosenDate(1,1):ChosenDate(1,2),60)+handles.TableData(ChosenDate(1,1):ChosenDate(1,2),21)-handles.TableData(ChosenDate(1,1):ChosenDate(1,2),19)*1.15+handles.TableData(ChosenDate(1,1):ChosenDate(1,2),8))...
                    -(handles.TableData(ChosenDate(1,1):ChosenDate(1,2),70)+handles.TableData(ChosenDate(1,1):ChosenDate(1,2),22)-handles.TableData(ChosenDate(1,1):ChosenDate(1,2),20)*1.1+handles.TableData(ChosenDate(1,1):ChosenDate(1,2),9))); %difference of Q_tot of time shift model between the two rooms

            prime=sum((handles.g(ChosenDate(1,1):ChosenDate(1,2))*2.667).*((handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12))*0.5+(handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12))*0.5)); %slope of the curve
            multiplier=old-dif/prime; %new multiplier
        end
        multiplier_diff=1/multiplier
        %==========================================================================
        % room 108 only Newton Raphson, avarage Q_tot109 out to zero
        multiplier_8=1;
        old_8=0;
        A=size(handles.TableData);
        while abs(multiplier_8-old_8)>0.001
            old_8=multiplier_8;
             for i=2:A(1)
                handles.TableData(i,60)=(handles.g(i-1)*2.667)*multiplier_8*((handles.TableData(i-1,12))*0.5+(handles.TableData(i,12))*0.5)-1*(handles.TableData(i,2)-handles.TableData(i,10))*8.37; %A*g*G-U*deltaT*A timeshift included 0.5 old value 0.5 new value
             end
             dif_8=sum((handles.TableData(ChosenDate(1,1):ChosenDate(1,2),60)+
                 handles.TableData(ChosenDate(1,1):ChosenDate(1,2),21)-handles.
                 TableData(ChosenDate(1,1):ChosenDate(1,2),19)*1.15+handles.
                 TableData(ChosenDate(1,1):ChosenDate(1,2),8)));


            prime_8=sum((handles.g(ChosenDate(1,1):ChosenDate(1,2))*2.667).*((
                handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12))*0.5+(handles
                .TableData(ChosenDate(1,1):ChosenDate(1,2),12))*0.5));
            multiplier_8=old_8-dif_8/prime_8;
        end
        multiplier_8

    % end room 108 only
    %==================================================================

    %==================================================================
    % room 109 only Newton Raphson, avarage Q_tot109 out to zero
        multiplier_9=1;
        old_9=0;
        A=size(handles.TableData);
        while abs(multiplier_9-old_9)>0.001
            old_9=multiplier_9;
             for i=2:A(1)
                handles.TableData(i,75)=(handles.g(i-1)*2.667)*multiplier_9*((
                    handles.TableData(i-1,12))*0.5+(handles.TableData(i,12))*0.5)
                    -1*(handles.TableData(i,5)-handles.TableData(i,10))*8.37; %A*g*G
                    -U*deltaT*A timeshift included 0.5 old value 0.5 new value
             end
             dif_9=sum((handles.TableData(ChosenDate(1,1):ChosenDate(1,2),75)+
                 handles.TableData(ChosenDate(1,1):ChosenDate(1,2),22)-handles.
                 TableData(ChosenDate(1,1):ChosenDate(1,2),20)*1.1+handles.
                 TableData(ChosenDate(1,1):ChosenDate(1,2),9)));


            prime_9=sum((handles.g(ChosenDate(1,1):ChosenDate(1,2))*2.667).*((
                handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12))*0.5+(handles
                .TableData(ChosenDate(1,1):ChosenDate(1,2),12))*0.5));
            multiplier_9=old_9-dif_9/prime_9;
        end
        multiplier_9

    % end room 109 only
    %==================================================================
    %end of g-value calculation
    %==================================================================
    %average g-Value calculation
    %weighted g-value=(g_i*Soll_irradiation)/Sol_irradiation
    irrad=sum(handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12));
    sum_g=sum(handles.TableData(ChosenDate(1,1):ChosenDate(1,2),12).*handles.g(
        ChosenDate(1,1):ChosenDate(1,2)));
    %room 108
    sum_g108=sum_g*multiplier_8;
    weight_g108=sum_g108/irrad
    %room 109
    sum_g109=sum_g*multiplier_9;
    weight_g109=sum_g109/irrad
    %difference & g_0 corrected
    weight_diff=weight_g109-weight_g108 %refernce room - measure room
```

```
  g_zero108=0.31*multiplier_8   %corrected g_0 for 108
  g_zero109=0.31*multiplier_9   %corrected g_0 for 109

  fprintf('======================\n')

guidata(hObject,handles);       %store new handles for main program (calorimain)
```

B.4 String Split

```
function parts = strsplit(splitstr, str, option)
%STRSPLIT Split string into pieces.
%
%   STRSPLIT(SPLITSTR, STR, OPTION) splits the string STR at every occurrence
%   of SPLITSTR and returns the result as a cell array of strings. By default,
%   SPLITSTR is not included in the output.
%
%   STRSPLIT(SPLITSTR, STR, OPTION) can be used to control how SPLITSTR is
%   included in the output. If OPTION is 'include', SPLITSTR will be included
%   as a separate string. If OPTION is 'append', SPLITSTR will be appended to
%   each output string, as if the input string was split at the position right
%   after the occurrence SPLITSTR. If OPTION is 'omit', SPLITSTR will not be
%   included in the output.

%   Author:       Peter J. Acklam
%   Time-stamp:   2004-09-22 08:48:01 +0200
%   E-mail:       pjacklam@online.no
%   URL:          http://home.online.no/~pjacklam

   nargsin = nargin;
   error(nargchk(2, 3, nargsin));
   if nargsin < 3
      option = 'omit';
   else
      option = lower(option);
   end

   splitlen = length(splitstr);
   parts = {};

   while 1

      k = strfind(str, splitstr);
      if isempty(k)
         parts{end+1} = str;
         break
      end

      switch option
         case 'include'
            parts(end+1:end+2) = {str(1:k(1)-1), splitstr};
         case 'append'
            parts{end+1} = str(1 : k(1)+splitlen-1);
         case 'omit'
            parts{end+1} = str(1 : k(1)-1);
         otherwise
            error(['Invalid option string -- ', option]);
      end


      str = str(k(1)+splitlen : end);

   end
```

B.5 X-tick rotate ("45°")

```
function hText = xticklabel_rotate(XTick,rot,varargin)
%hText = xticklabel_rotate(XTick,rot,XTickLabel,varargin)     Rotate XTickLabel
%
% Syntax: xticklabel_rotate
%
% Input:
% {opt}     XTick       - vector array of XTick positions & values (numeric)
%                           uses current XTick values or XTickLabel cell array by
%                           default (if empty)
% {opt}     rot         - angle of rotation in degrees, 90° by default
% {opt}     XTickLabel  - cell array of label strings
% {opt}     [var]       - "Property-value" pairs passed to text generator
%                           ex: 'interpreter','none'
%                               'Color','m','Fontweight','bold'
%
% Output:   hText       - handle vector to text labels
%
% Example 1:  Rotate existing XTickLabels at their current position by 90°
%    xticklabel_rotate
%
% Example 2:  Rotate existing XTickLabels at their current position by 45° and
    change
% font size
%    xticklabel_rotate([],45,[],'Fontsize',14)
%
% Example 3:  Set the positions of the XTicks and rotate them 90°
%    figure;  plot([1960:2004],randn(45,1)); xlim([1960 2004]);
%    xticklabel_rotate([1960:2:2004]);
%
% Example 4:  Use text labels at XTick positions rotated 45° without tex
    interpreter
%    xticklabel_rotate(XTick,45,NameFields,'interpreter','none');
%
% Example 5:  Use text labels rotated 90° at current positions
%    xticklabel_rotate([],90,NameFields);
%
% Note : you can not re-run xticklabel_rotate on the same graph.
%



% This is a modified version of xticklabel_rotate90 by Denis Gilbert
% Modifications include Text labels (in the form of cell array)
%                       Arbitrary angle rotation
%                       Output of text handles
%                       Resizing of axes and title/xlabel/ylabel positions to
    maintain same overall size
%                           and keep text on plot
%                           (handles small window resizing after, but not well
    due to proportional placement with
%                           fixed font size. To fix this would require a serious
    resize function)
%                       Uses current XTick by default
%                       Uses current XTickLabel is different from XTick values (
    meaning has been already defined)

```

```
% Brian FG Katz
% bfgkatz@hotmail.com
% 23-05-03
% Modified 03-11-06 after user comment
%       Allow for exisiting XTickLabel cell array
% Modified 03-03-2006
%   Allow for labels top located (after user comment)
%   Allow case for single XTickLabelName (after user comment)
%   Reduced the degree of resizing

% Other m-files required: cell2mat
% Subfunctions: none
% MAT-files required: none
%
% See also: xticklabel_rotate90, TEXT, SET

% Based on xticklabel_rotate90
%   Author: Denis Gilbert, Ph.D., physical oceanography
%   Maurice Lamontagne Institute, Dept. of Fisheries and Oceans Canada
%   email: gilbertd@dfo-mpo.gc.ca  Web: http://www.qc.dfo-mpo.gc.ca/iml/
%   February 1998; Last revision: 24-Mar-2003

% check to see if xticklabel_rotate has already been here (no other reason for
    this to happen)
if isempty(get(gca,'XTickLabel')),
    error('xticklabel_rotate : can not process, either xticklabel_rotate has
        already been run or XTickLabel field has been erased')  ;
end

% if no XTickLabel AND no XTick are defined use the current XTickLabel
%if nargin < 3 & (~exist('XTick') | isempty(XTick)),
% Modified with forum comment by "Nathan Pust" allow the current text labels to
    be used and property value pairs to be changed for those labels
if (nargin < 3 || isempty(varargin{1})) & (~exist('XTick') | isempty(XTick)),
        xTickLabels = get(gca,'XTickLabel')  ; % use current XTickLabel
        if ~iscell(xTickLabels)
                % remove trailing spaces if exist (typical with auto generated
                    XTickLabel)
                temp1 = num2cell(xTickLabels,2)         ;
                for loop = 1:length(temp1),
                        temp1{loop} = deblank(temp1{loop})  ;
                end
                xTickLabels = temp1                     ;
        end
varargin = varargin(2:length(varargin));
end

% if no XTick is defined use the current XTick
if (~exist('XTick') | isempty(XTick)),
    XTick = get(gca,'XTick')        ; % use current XTick
end

%Make XTick a column vector
XTick = XTick(:);

if ~exist('xTickLabels'),
        % Define the xtickLabels
        % If XtickLabel is passed as a cell array then use the text
        if (length(varargin)>0) & (iscell(varargin{1})),
            xTickLabels = varargin{1};
            varargin = varargin(2:length(varargin));
            else
            xTickLabels = num2str(XTick);
            end
end

if length(XTick) ~= length(xTickLabels),
    error('xticklabel_rotate␣:␣must␣have␣same␣number␣of␣elements␣in␣"XTick"␣and␣"XTickLabel"')    ;
end

%Set the Xtick locations and set XTicklabel to an empty string
set(gca,'XTick',XTick,'XTickLabel','')

if nargin < 2,
    rot(1,1) = 90  ;
end

% Determine the location of the labels based on the position
% of the xlabel
hxLabel = get(gca,'XLabel');  % Handle to xlabel
xLabelString = get(hxLabel,'String');

% if ~isempty(xLabelString)
%    warning('You may need to manually reset the XLABEL vertical position')
% end

set(hxLabel,'Units','data');
xLabelPosition = get(hxLabel,'Position');
y = xLabelPosition(2);

%CODE below was modified following suggestions from Urs Schwarz
y=repmat(y,size(XTick,1),1);
% retrieve current axis' fontsize
fs = get(gca,'fontsize');

% Place the new xTickLabels by creating TEXT objects
hText = text(XTick, y, xTickLabels,'fontsize',fs);

% Rotate the text objects by ROT degrees
%set(hText,'Rotation',rot,'HorizontalAlignment','right',varargin{:})
% Modified with modified forum comment by "Korey Y" to deal with labels at top
% Further edits added for axis position
xAxisLocation = get(gca, 'XAxisLocation');
if strcmp(xAxisLocation,'bottom')
    set(hText,'Rotation',rot(1,1),'HorizontalAlignment','right',varargin{:})
else
    set(hText,'Rotation',rot(1,1),'HorizontalAlignment','left',varargin{:})
end

% Adjust the size of the axis to accomodate for longest label (like if they are text ones)
% This approach keeps the top of the graph at the same place and tries to keep xlabel at the same place
% This approach keeps the right side of the graph at the same place

set(get(gca,'xlabel'),'units','data')           ;
    labxorigpos_data = get(get(gca,'xlabel'),'position')   ;
```

```
set(get(gca,'ylabel'),'units','data')            ;
    labyorigpos_data = get(get(gca,'ylabel'),'position')   ;
set(get(gca,'title'),'units','data')            ;
    labtorigpos_data = get(get(gca,'title'),'position')   ;

set(gca,'units','pixel')                         ;
set(hText,'units','pixel')                       ;
set(get(gca,'xlabel'),'units','pixel')           ;
set(get(gca,'ylabel'),'units','pixel')           ;

origpos = get(gca,'position')                    ;

% textsizes = cell2mat(get(hText,'extent'))       ;
% Modified with forum comment from "Peter Pan" to deal with case when only one
    XTickLabelName is given.
x = get( hText, 'extent' );
if iscell( x ) == true
    textsizes = cell2mat( x ) ;
else
    textsizes = x;
end

largest =  max(textsizes(:,3))                   ;
longest =  max(textsizes(:,4))                   ;

laborigext = get(get(gca,'xlabel'),'extent')     ;
laborigpos = get(get(gca,'xlabel'),'position')   ;

labyorigext = get(get(gca,'ylabel'),'extent')    ;
labyorigpos = get(get(gca,'ylabel'),'position')  ;
leftlabdist = labyorigpos(1) + labyorigext(1)    ;

% assume first entry is the farthest left
leftpos = get(hText(1),'position')               ;
leftext = get(hText(1),'extent')                 ;
leftdist = leftpos(1) + leftext(1)               ;
if leftdist > 0,    leftdist = 0 ; end           % only correct for off screen
    problems

% botdist = origpos(2) + laborigpos(2)           ;
% newpos = [origpos(1)-leftdist longest+botdist origpos(3)+leftdist origpos(4)-
    longest+origpos(2)-botdist]
%
% Modified to allow for top axis labels and to minimize axis resizing
if strcmp(xAxisLocation,'bottom')
    newpos = [origpos(1)-(min(leftdist,labyorigpos(1)))+labyorigpos(1) ...
            origpos(2)+((longest+laborigpos(2))-get(gca,'FontSize')) ...
            origpos(3)-(min(leftdist,labyorigpos(1)))+labyorigpos(1)-largest ...
            origpos(4)-((longest+laborigpos(2))-get(gca,'FontSize'))] ;
else
    newpos = [origpos(1)-(min(leftdist,labyorigpos(1)))+labyorigpos(1) ...
            origpos(2) ...
            origpos(3)-(min(leftdist,labyorigpos(1)))+labyorigpos(1)-largest ...
            origpos(4)-(longest)+get(gca,'FontSize')] ;
end
A=size(rot);
if A(1,2) ==5;
 newpos= [rot(1,2:5)];
end
```

```
set(gca,'position',newpos)                              ;

% readjust position of text labels after resize of plot
set(hText,'units','data')                               ;
for loop= 1:length(hText),
    set(hText(loop),'position',[XTick(loop), y(loop)])   ;
end

% adjust position of xlabel and ylabel
laborigpos = get(get(gca,'xlabel'),'position')   ;
set(get(gca,'xlabel'),'position',[laborigpos(1) laborigpos(2)-longest 0])    ;

% switch to data coord and fix it all
set(get(gca,'ylabel'),'units','data')                    ;
set(get(gca,'ylabel'),'position',labyorigpos_data)       ;
set(get(gca,'title'),'position',labtorigpos_data)        ;

set(get(gca,'xlabel'),'units','data')                    ;
    labxorigpos_data_new = get(get(gca,'xlabel'),'position')   ;
set(get(gca,'xlabel'),'position',[labxorigpos_data(1) labxorigpos_data_new(2)])
     ;


% Reset all units to normalized to allow future resizing
set(get(gca,'xlabel'),'units','normalized')             ;
set(get(gca,'ylabel'),'units','normalized')             ;
set(get(gca,'title'),'units','normalized')             ;
set(hText,'units','normalized')                         ;
set(gca,'units','normalized')                           ;

if nargout < 1,
    clear hText
end
```

Appendix C LOGGER code

```
'CR1000 Series Datalogger Markus1.2.CR1
'date: 2009 06 05
'program author:  Håkan Håkansson & Markus Heimberger

CONST MM = 22  'T [°C]
CONST NN = 8  'Q  [W]
CONST PP = 2    'Sol [W/m2]
CONST CalConstVert=13.15  'Heat flow in floor and ceiling
CONST CalConstHor=15.13  'Heat flow in walls
CONST Pyrconst1=219.9  'constant for pyranometer which one measure beam
CONST Pyrconst2=105.6  'constant for pyranometer which one measure diffuse
CONST Pyrconst3= 1 ' ,Licor, dubbelavläst
CONST PyrgConst=238.663  '4.19uV/(W/m^2)
Public LoggerT, RefT  ' PTemp reftemperatur i logger, Multiplexer
Public Q8Ceil, Q8Floor, Q8WWall, Q8EWall
Public Q9Ceil, Q9Floor, Q9WWall, Q9EWall
Public Q108,Q109
Public Q_tempWall
Public VertSolMark
Public T8luft,T8glas,  T9luft,T9solSk,T9glas,DT8glas,DT9glas
Public Wflow108, Wflow109,WAvgRun8, WAvgRun9 '(Running averages)
Public Qkyla108, Qkyla109, DeltaTW8, DeltaTW9
Public El108, El109 ' New          elctric Pulscount
Public Pyrgeometer,PyrgeTest,Pyr_Ref, Pyr_RefTest
Public PyrMux(PP),Pyran(PP)

Dim T_Mux(MM)  'Temperatures
Dim V_ThPile(NN)  ' V_tempThPile  'Voltage from termopiles
Dim m     ' uppräknings variabel (MM)
Dim n     ' (NN)
Dim p     ' (PP)

Alias T_Mux(1)=T8luftUpp
Alias T_Mux(2)=T8luftMitt
Alias T_Mux(3)=T8luftNert
Alias T_Mux(4)=T8glasInUpp
Alias T_Mux(5)=T8glasInMitt
Alias T_Mux(6)=T8glasInNert
Alias T_Mux(7)=T8glasUtmitt
Alias T_Mux(8)=T9luftUpp
Alias T_Mux(9)=T9luftMitt
Alias T_Mux(10)=T9luftNert
Alias T_Mux(11)=T9solSkUpp
Alias T_Mux(12)=T9solSkMitt
Alias T_Mux(13)=T9solSkNert
Alias T_Mux(14)=T9glasInUpp
Alias T_Mux(15)=T9glasInMitt
Alias T_Mux(16)=T9glasInNert
Alias T_Mux(17)=T9glasUtmitt
Alias T_Mux(18)=TluftUte
Alias T_Mux(19)=T_w108ut
Alias T_Mux(20)=T_w108in
Alias T_Mux(21)=T_w109ut
Alias T_Mux(22)=T_w109in
Alias Pyran(1)=VertSol
Alias Pyran(2)=VertSolDiff


'Define Data Tables— Data Tables –– DataTables— Data Tables
DataTable(#6min,1,-1) '#6MIN tabell
        Datainterval(0,6,Min,-1)  'data intervall 6min. average
                Average(1,T8luft,FP2,False)
                Average(1,T8glas,FP2,False)
                Average(1,DT8glas,FP2,False)
                Average(1,T9luft,FP2,False)
                Average(1,T9glas,FP2,False)
                Average(1,DT9glas,FP2,False)
                Average(1,Q108,FP2,False)
                Average(1,Q109,FP2,False)
                Average(1,TluftUte,FP2,False)
                Average(1,RefT,FP2,False)
                Average(1,VertSol,FP2,False)
                Average(1,VertSolDiff,FP2,False)
                Average(1,VertSolMark,FP2,False)
                Average(1,Pyrgeometer,FP2,False)
                Average(1,PyrgeTest,FP2,False)
                Average(1,Wflow108,FP2,False)
                Average(1,Wflow109,FP2,False)
                Average(1,Qkyla108,FP2,False)
                Average(1,Qkyla109,FP2,False)
                Average(1,El108,FP2,False)
                Average(1,El109,FP2,False)
                Average(1,Q_tempWall,FP2,False)
                Average(1,Q8Ceil,FP2,False)
                Average(1,Q8Floor,FP2,False)
                Average(1,Q8WWall,FP2,False)
                Average(1,Q8EWall,FP2,False)
                Average(1,Q9Ceil,FP2,False)
                Average(1,Q9Floor,FP2,False)
                Average(1,Q9WWall,FP2,False)
                Average(1,LoggerT,FP2,False)

         'control values
        Average(1,T8luftUpp,FP2,False)
        Average(1,T8luftMitt,FP2,False)
        Average(1,T8luftNert,FP2,False)
        Average(1,T8glasInMitt,FP2,False)
        Average(1,T8glasInNert,FP2,False)
        Average(1,T8glasUtmitt,FP2,False)
        Average(1,T9luftUpp,FP2,False)
        Average(1,T9luftMitt,FP2,False)
        Average(1,T9luftNert,FP2,False)
        Average(1,T9glasInUpp,FP2,False)
        Average(1,T9glasInMitt,FP2,False)
        Average(1,T9glasInNert,FP2,False)
        Average(1,T9glasUtmitt,FP2,False)
        Average(1,T_w108ut,FP2,False)
        Average(1,T_w108in,FP2,False)
```

```
            Average(1,T_w109ut,FP2,False)
            Average(1,T_w109in,FP2,False)
            Average(1,DeltaTW8,FP2,False)
            Average(1,DeltaTW9,FP2,False)
EndTable ' End    #6MIN ==============================

DataTable(o_view,1,-1)    ' O_VIEW
           DataInterval (0,60,Min,-1) ' hourly mean for key variables
                      Average(1,T8luft,FP2,False)
                      Average(1,T8glas,FP2,False)
                      Average(1,DT8glas,FP2,False)
                      Average(1,T9luft,FP2,False)
                      Average(1,T9glas,FP2,False)
                      Average(1,DT9glas,FP2,False)
                      Average(1,Q108,FP2,False)
                      Average(1,Q109,FP2,False)
                      Average(1,TluftUte,FP2,False)
                      Average(1,RefT,FP2,False)
                      Average(1,VertSol,FP2,False)
                      Average(1,VertSolDiff,FP2,False)
                      Average(1,VertSolMark,FP2,False)
                      Average(1,Pyrgeometer,FP2,False)
                      Average(1,PyrgeTest,FP2,False)
                      Average(1,Wflow108,FP2,False)
                      Average(1,Wflow109,FP2,False)
                      Average(1,Qkyla108,FP2,False)
                      Average(1,Qkyla109,FP2,False)
                      Average(1,El108,FP2,False)
                      Average(1,El109,FP2,False)
                      Average(1,Q_tempWall,FP2,False)
                      Average(1,Q8Ceil,FP2,False)
                      Average(1,Q8Floor,FP2,False)
                      Average(1,Q8WWall,FP2,False)
                      Average(1,Q8EWall,FP2,False)
                      Average(1,Q9Ceil,FP2,False)
                      Average(1,Q9Floor,FP2,False)
                      Average(1,Q9WWall,FP2,False)
                      Average(1,LoggerT,FP2,False)

            'control values
            Average(1,T8luftUpp,FP2,False)
            Average(1,T8luftMitt,FP2,False)
            Average(1,T8luftNert,FP2,False)
            Average(1,T8glasInMitt,FP2,False)
            Average(1,T8glasInNert,FP2,False)
            Average(1,T8glasUtmitt,FP2,False)
            Average(1,T9luftUpp,FP2,False)
            Average(1,T9luftMitt,FP2,False)
            Average(1,T9luftNert,FP2,False)
            Average(1,T9glasInUpp,FP2,False)
            Average(1,T9glasInMitt,FP2,False)
            Average(1,T9glasInNert,FP2,False)
            Average(1,T9glasUtmitt,FP2,False)
            Average(1,T_w108ut,FP2,False)
            Average(1,T_w108in,FP2,False)
            Average(1,T_w109ut,FP2,False)
            Average(1,T_w109in,FP2,False)
            Average(1,DeltaTW8,FP2,False)
            Average(1,DeltaTW9,FP2,False)
EndTable ' End   O_VIEW  ==================

'Define Subroutines   'Sub   'EndSub
'Main Program_________________________________
BeginProg
            Scan(30,Sec,3,0) 'scannar logger o multiplex
                        VoltDiff (VertSolMark,1,mV25C,3,True ,0,250,10,0)
                        VoltDiff (Pyrgeometer,1,mV2_5C,5,True ,0,250,PyrgConst,0) '
                            Pyrgeometer
                        VoltDiff (PyrgeTest,1,mV2_5C,5,True ,0,250,1,0) ' measurement of
                            Pyrgeometer
                        VoltDiff (Pyr_Ref,1,mV2_5C,6,True ,0,250,238.095,0)
                        VoltDiff (Pyr_RefTest,1,mV2_5C,6,True ,0,250,1,0)
                        PanelTemp(LoggerT,250)  'panel temperature in logger (for check)
                        Therm107 (RefT,1,16,Vx2,0,250,1.0,0) '<— SE16 and Exitation Vx2

                        '30sec scan interval 2500ml per pulse/30s = 83.3333 ml per pulse/
                            s
                        PulseCount (Wflow108,1,1 ,2,0,83.3333,0) 'reps,P1,switch closure,
                            ,multiplier
                        PulseCount (Wflow109,1,2 ,2,0,83.3333,0) 'reps,P2,switch closure,
                            ,multiplier
                        AvgRun (WAvgRun8,1,Wflow108,12) 'Running average 12*30s = 6 min
                        AvgRun (WAvgRun9,1,Wflow109,12)
                        PulseCount (El108,1,17,0,0,120,0) 'Electric power in W  Control
                            port <—C7
                        PulseCount (El109,1,18,0,0,120,0) 'Electric power in W  Control
                            port <—C8
                        PortSet(1,1)    'port C1 high for MUX scan counter reset <— C1
                        For m=1 to MM '22 Thermocouples on MUX
                              Pulseport(2,15000) 'first M MUX channels
                              TCDiff(T_Mux(m),1,mV2_5,1,TypeT,RefT,True,0,250,1,0) '
                                  termoelement
                        Next m
                        ' 2 Extra readings of wheather data sensors on south wall
                        Pulseport(2,15000)
                        'no variable on which one port 23 is saved unknown connection
                        Pulseport(2,15000)
                        VoltDiff(PyrMux(2),1,mV7_5,1,0,0,250,1,0)
                        Pyran(2)=PyrMux(2)*Pyrconst2
                        For n=1 to NN                'ThermoPILE channels on
                            Multiplexer (8)
                              Pulseport(2,15000)       'scan, and time delay <—C2
                              VoltDiff(V_ThPile(n),1,mV7_5,1,0,0,250,1,0)
                        Next n
                        Portset(1,0)     'set port 1

                        VoltDiff(PyrMux(1),1,mV7_5,2,0,0,250,1,0)
                        Pyran(1)=PyrMux(1)*Pyrconst1

                        'Measuring of test calibration wall if needed
                        VoltDiff(Q_tempWall,1,mV25,4,True,0,250,1,0) ' <==multiplier!,<--
                            Ch4
                        'Q8Ceil, Q8Floor, Q8WWall, Q8EWall, Q9Ceil, Q9Floor, Q9WWall,
                            Q9EWall
                        Q8Ceil=V_ThPile(1)*CalConstHor  '108
                        Q8Floor=V_ThPile(2)*CalConstHor  '108
```

```
                Q8WWall=V_ThPile(3)*CalConstVert '108
                Q8EWall=V_ThPile(4)*CalConstVert '108
                Q9Ceil=V_ThPile(5)*CalConstHor '109
                Q9Floor=V_ThPile(6)*CalConstHor '109
                Q9WWall=V_ThPile(7)*CalConstVert '109
                Q9EWall=V_ThPile(8)*CalConstVert '109
                Q108= Q8EWall+Q8WWall+Q8Floor+Q8Ceil
                Q109= Q9EWall+Q9WWall+Q9Floor+Q9Ceil
                T8luft=(T8luftUpp+T8luftMitt+T8luftNert)/3 ' mean value for 3
                    levels
                T8glas=(T8glasInUpp+T8glasInMitt+T8glasInNert)/3 ' mean value for
                    3 levels
                T9luft=(T9luftUpp+T9luftMitt+T9luftNert)/3 ' mean value for 3
                    levels
                T9solSk=(T9solSkUpp+T9solSkMitt+T9solSkNert)/3 ' mean value for 3
                    levels
                T9glas=(T9glasInUpp+T9glasInMitt+T9glasInNert)/3 ' mean value for
                    3 levels
                DT8glas=T8glasUtMitt-T8glasInMitt 'Temperature diff between panes
                     at mid level 108
                DT9glas=T9glasUtMitt-T9glasInMitt 'Temperature diff between panes
                     at mid level 109
                DeltaTW8=(T_w108ut-T_w108in)
                DeltaTW9=(T_w109ut-T_w109in)
                Qkyla108=DeltaTW8*WAvgRun8*4.18
                Qkyla109=DeltaTW9*WAvgRun9*4.18

                CallTable #6min    'call 6min data table
                CallTable o_view   'call 1h data table


        NextScan   'next scan
EndProg
```

Appendix D Pulse measurements

D.1 Circuit for level adaption

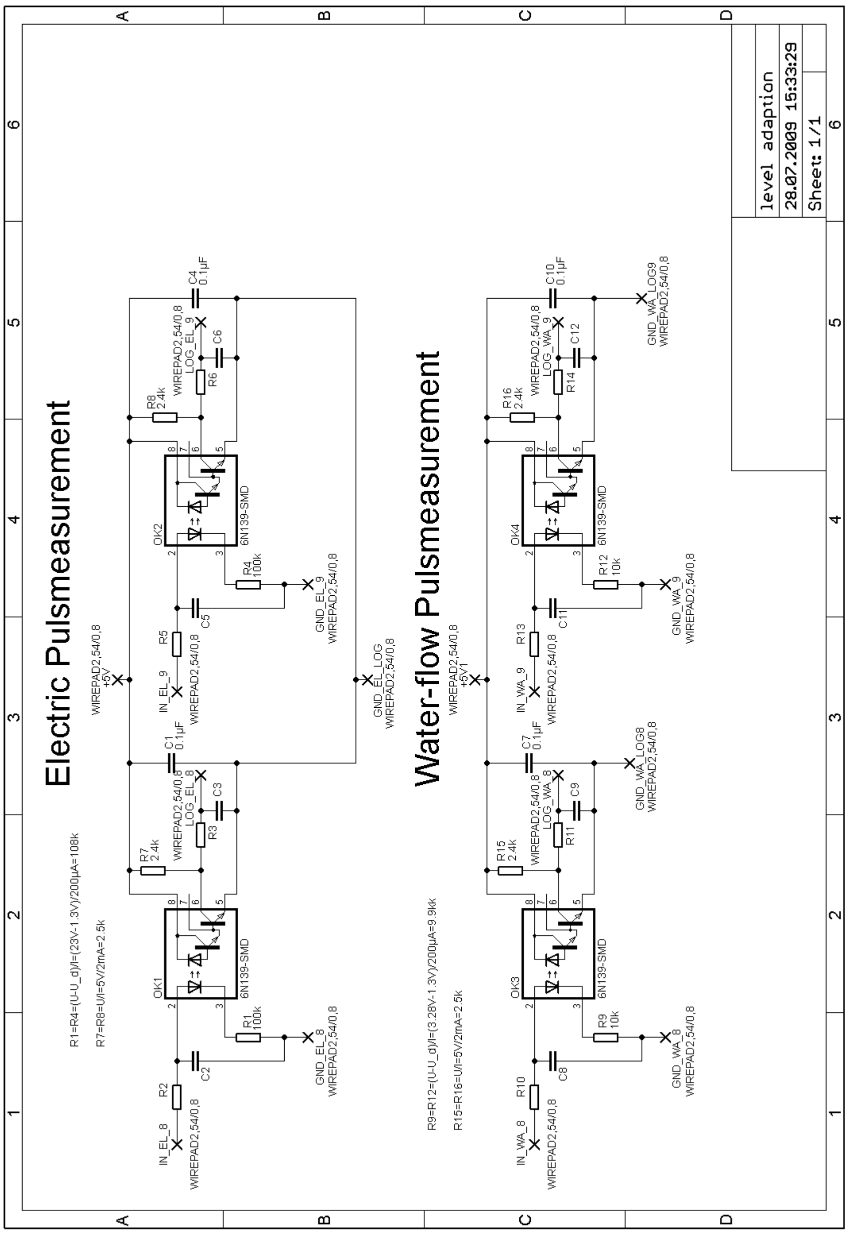


Figure D.1: Circuit plan for electric and water-flow pulse measurement

D.2 Circuit board layout

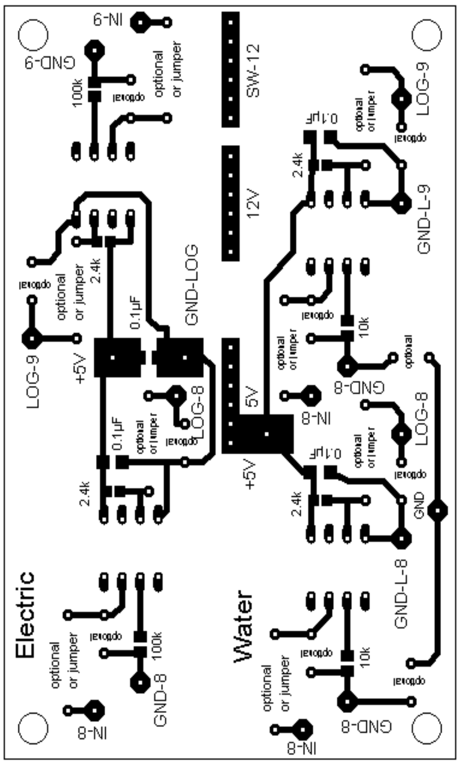


Figure D.2: Circuit board layout

D.3 Device plan

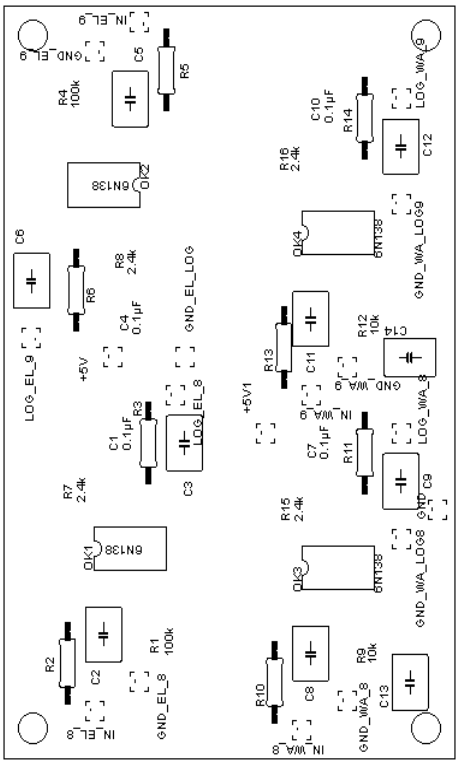


Figure D.3: Device plan