VTT-R-04946-16

Solar heat forecasting in heat system optimal control

Kirjoittajat: Riku Pasonen, Atte Löf, Göran Koreneff

Luottamuksellisuus: Julkinen

Name of the report Solar heating forecast in heat system optimal control	
For **Tekes and Clic Innovation**	
Project name Future Flexible Energy Systems	**Project acronym** FLEXe
Authors Riku Pasonen, Atte Löf, Göran Koreneff	**pages** 16
Keywords Solar heat, forecasting, optimization	**Identification code** VTT-R-04946-16

Abstract

Use of solar radiation forecasting in heating system optimization simulation was carried out in this study. Open data from Finnish Meteorological Institute (FMI) data base was used to generate solar and temperature forecast. Temperature forecast was used to estimate the coefficient of performance (COP) for heat pump and generate heat load forecast with heat model. Temperature data from previous year was used to build heat model of case building (district heating connected building in Espoo). The solar forecast calculation was built using Python language, the heat forecast model with APL and the optimization code using Julia language. The aim of the optimization was to minimize operational costs and investigate results using different storage and collector sizes.

It was possible to get savings in heating costs with solar collectors and heat pump. Storage can also yield savings both with heat pump and solar collectors. Excess energy from solar collectors can be stored to storage and used later and also heat pump usage shifted to different hours to take advantage on hourly electricity prices. The results indicate that with low electricity price that we have now in Nordic markets it is possible have cheaper heating energy than district heating. However due to the limited scope of the work here, maintenance fees are not investigated. The operation cost can however be up to 50% less without taking account the maintenance. For total profitability analysis, investment cost and longer time period performance simulation would be needed.

Target of the study was to investigate integration of solar radiation and heat load forecast models to optimization code. Two models were combined via a bit non-optimal way because both had been previously built on different platforms. It was discussed during the project that combined system could be rewritten with Python for more elegant approach.

Confidentiality	Public

Espoo 17.11.2016

Author	Reviewer	Approver
Riku Pasonen	Seppo Hänninen	Tuula Mäkinen

Contact information
Riku.Pasonen@vtt.fi

Distribution
Project consortium and VTT

Preface

This work was carried out in the research program Flexible Energy Systems (FLEXe) and supported by Tekes – the Finnish Funding Agency for Innovation. The aim of FLEXe is to create novel technological and business concepts enhancing the radical transition from the current energy systems towards sustainable systems. FLEXe consortium consists of 17 industrial partners and 10 research organisations. The programme is coordinated by CLIC Innovation Ltd. www.clicinnovation.fi

Authors

Espoo 17.11.2016

Contents

Preface 2

Contents 3

1. Introduction 4

2. Solar forecasting module 5

   2.1 Open data weather forecast 5
   2.2 Calculation of solar radiation on tilted surface 5
   2.3 Database solution 6

3. Heat pump and storage model 6

4. Heat load forecasting module EME Forecast 8

   4.1 EME Forecast working principle and time series model 8

5. Optimization tool 9

   5.1 Clp solver 9
   5.2 Program code explained 9

6. Results 12

   6.1 Accuracy of forecasts 14

7. Conclusions 16

References 16

1. Introduction

Use and creation of new information from vast sources of open data has been one of the driving forces behind large number of new start-up companies. The study here is about combining open weather data with solar forecast for to create a simple optimization system.

Solar forecasting calculation is a carryover from the national research program Smart Grids and Energy Markets (SGEM), which was a large collaboration project in Finland between research institutions and key industry players. Some refinements to solar calculation have been made since the previous project but most of the work presented here has been done in the FLEXe project with emphasis on interconnection of optimization with the Julia language to Solar production estimation running in Python to web interface to present results. Also a heat demand forecast model made with APL is used in the study.

2. Solar forecasting module

2.1 Open data weather forecast

Finnish Meteorological Institute's (FMI) open data was utilized in the solar heat collector production forecasting algorithm. HIRLAM weather forecast model is run four times a day (00, 06, 12 and 18 UTC) and it is a limited-range model, which covers Europe, the North Atlantic and part of the Arctic regions. (Finnish Meteorological Institute, 2015) The data is available in XML format and therefore some parsing of the data is required. The Python programming language was used for data retrieval because also the calculation of solar equations was done using python.

2.2 Calculation of solar radiation on tilted surface

To estimate the solar collector production, the azimuth and elevation angle of the sun should be accurately calculated. Production estimation algorithm calculates these angles for selected time steps during the day from coordinates and altitude of selected location. Modelling of the sun position angles is explained in (Bratu, 2008). These angles are utilized to calculate the amount of solar radiation on a tilted surface as well as the panel temperature. Solar radiation is divided into three components which are direct, diffuse and reflected solar radiation. These components are illustrated in the **Figure 2-1**.

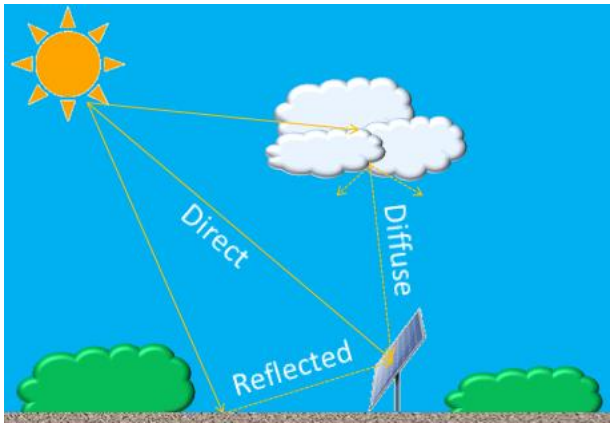

Figure 2-1. Radiation components on tilted panel surface.

Of the three components, direct and diffuse parts are the most important for estimation of the PV system energy production. Beam radiation is the perpendicular component of direct radiation to the surface.(see **Figure 2-1**) The reflected component depends on the place where the panels are installed(type of ground/sea). Reflected part of the radiation is usually an insignificant compared to direct and diffuse radiation. Calculation of the radiation on tilted panel surface was done with the HDKR model (the Hay, Davies, Klucher, Reindl model). The total radiation on tilted panel surface is (Duffie & Beckman, 2013):

$$I_T = (I_b - I_d A_i) R_b + I_d (1 - A_i) \left(\frac{1 + \cos\beta}{2}\right) \left[1 + f \sin^3\left(\frac{\beta}{2}\right)\right] + I_{\rho_g} \left(\frac{1 - \cos\beta}{2}\right)$$

Where I_T is the total radiation on the tilted surface, I_b is the beam radiation, I_d is the diffuse radiation, I_{ρ_g} is the ground reflectance (also called the albedo), R_b is the ratio of beam radiation on the tilted surface to beam radiation on the horizontal surface, A_i is the anisotropy index, β is the slope angle of the surface and f is the final factor. The anisotropy index determines a portion of the horizontal diffuse and it is given by the following equation.

$$A_i = \frac{I_b}{I_0} \tag{2}$$

Where I_0 is the extraterrestrial horizontal radiation. The final factor is related to the cloudiness of the location and it is given by the following equation (Duffie & Beckman, 2013):

$$f = \sqrt{\frac{I_b}{I}} \tag{3}$$

Where I is the global horizontal radiation on the earth's surface. Solar collector efficiency model was not managed to be created in smaller than intended time frame of the project. So solar production capacity values are assumed to be losses already subtracted.

2.3 Database solution

Database is used for storage of data on local server. Postgre SQL database solution was selected because good experiences and performance on previous work. (PostgreSQL: The world's most advanced open source database, 2015). Command library for connecting to database is available for python so integration was fairly smooth.

3. Heat pump and storage model

Heat pump is modelled with equation approximation using following measurement data in **Figure 3-1**.

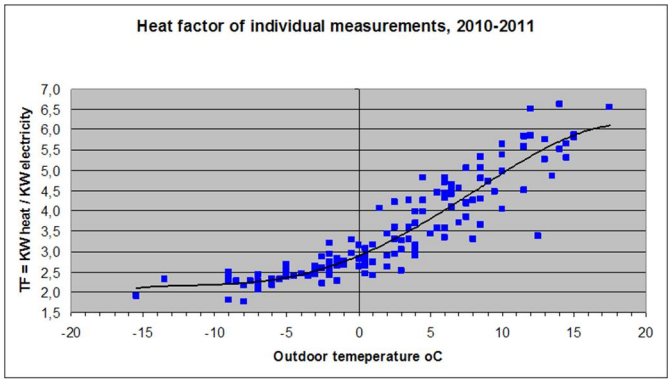

Figure 3-1. *Heat pump measurements (Zdeněk, 2011)*

Curve fitting from Matlab was used to do fit displayed in Figure 3-2

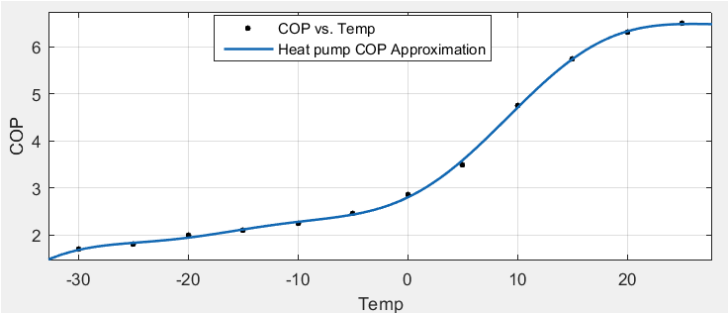

Figure 3-2. *Curve fit of COP vs temperature using selected points from measurements*

Equation for the fit is done with three sine functions.

$$COP(temp) = a1 * \sin(b1 * temp + c1) + a2 * \sin(b2 * temp + c2) + a3 * \sin(b3 * temp + c3) \quad (4)$$

Parameters for the fit are available in **Table 1**.

Table 1. *Parameters for polynomial COP approximation*

Parameter	Value
a1	9.662
c1	0.39
b2	0.1208
a3	0.1778
c3	3.998
b1	0.01016
a2	0.8445
c2	-1.064
b3	0.2412

COP for each hour can be calculated as constants from temperature forecast and supplied as input for the optimization code.

The storage model is more or less ideal simplification but there is option to use different efficiency ratios. Default capacity is 30 kWh. See more from chapter 5.2 which explains the code.

4. Heat load forecasting module EME Forecast

EME Forecast is a computer software for hourly short term load forecasting. It is a heuristic time series algorithm and it was especially designed for electricity load forecasting in the de-regulated market. Nonetheless, it could be used for all kind of energy sector forecasting, including district heat load and NordPool spot price, assuming the target has an appropriate day, week and season cycle. EME Forecast is used interactively or automated. It is easy to take into use, because no target specific parametrization by the user is required. EME Forecast is developed by Göran Koreneff at VTT.

4.1 EME Forecast working principle and time series model

The forecasting method of EME Forecast is based on the usage of comparison days from history. For each day to forecast a dynamic set of comparison day selection criteria is created. Comparison days are primarily searched from days with similar behavior. The criteria take into consideration day-of-week, day type including special holidays, week and several user definable parameters.

EME Forecast allows for the inclusion of one user chosen explaining factor influence of which is automatically determined through a detached regression analysis process. The explaining factor must be an hour time series like the target. It is up to the user to decide if the data input will be pure measurements or, for example, a 12 hours moving average. Outside temperature is a strong explaining factor. The operational environment of EME forecast tool is shown in **Figure 4-1**.

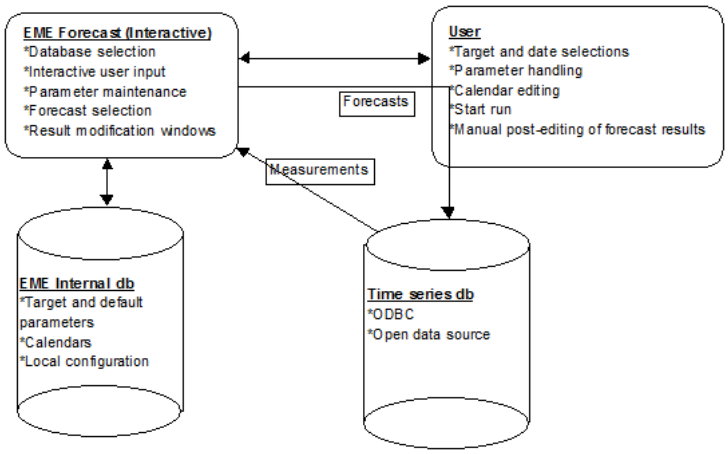

Figure 4-1. *EME Forecast operational environment.*

EME Forecast has a flexible calendar system, enabling multiple calendars. A calendar has information about which days are special days and how they are to be forecasted. All special days' definitions are in a common and editable table, which is available to all calendars. Each forecast target is connected to either the default calendar or a named calendar. The forecast of each hour is the average of corresponding comparison day(s) values. The average can also be calculated as a median area average.

5. Optimization tool

The optimization tool for the study is built using the Julia language. Julia was selected because it is an interesting new high performance programming language with easy syntax and good amount of libraries for linear optimization and data processing. Some things are underdeveloped compared to Python but Julia is improving fast. The main developer of Julia is MIT. (MIT, 2015)

Linear programming is a method for finding optimal solution to problems which can be described with linear a group of equations. Linear equations are described with inequality operators. The task is to maximize or minimize equation given constraints for variables. The equation to be maximized or minimized is called the objective.

Depending on the form of the actual problem, the linear optimization method suits to the case better or worse, but is unquestionably the most popular method for general optimization needs. Linear equations can be solved very fast with modern computers or even small handheld devices like smart phones. And this is one of the reasons for the popularity of method.

Following presents example of linear problem,

$$f(x_1, x_2 \ldots .) = C_1 x_1 + C_2 x_2 \ldots \quad (5)$$

with equations like:

$$A_{11}\, x_1 \; + \; A_{12} x_1 \ldots \; \leq \; b1 \quad (6)$$

$$A_{21} x_1 \; + \; A_{22} x_1 \ldots \; \leq \; b2 \quad (7)$$

And group of equations in matrix form,

$$Ax \; \leq \; b \quad (4)$$

And variable constraints like,

$$0 \; \leq \; b \leq c_n \quad (9)$$

5.1 Clp solver

The solver selected for the optimization is Clp (Coin-OR Linear Programming solver), which is an linear optimization solver written in C-code and uses Simplex algorithm. The Clp is used with the Julia using a package called “Clp”. This package provides low level interface to the solver but also high level Mathprog interface which enables to solve sequences of linear programs. Mathprog interface also enables easy way to use indexed variables which makes it possible to write constraints in indexed loops for compact code. (Coin-or linear programming Solver, 2015)

5.2 Program code explained

First part of the code involves reading input data from file.

```
using DataFrames
df= readtable("C:\\data\\tyohakemisto\\Flex\\Julia_optimization\\input.csv",separator = ';')
df=convert(DataArray,df)

#Model includes:,Solar heat,heat consumption,storage capacity,Air to air heat pump,electricity price
nordPool=df[:,1]
solar_forecast=df[:,2]

price(nordPool/1000+0.05+ 2.79372/100)/10000
#print(price)
#W is the unit for power and Wh for energy
heatload=df[:,3]

#Maximum heap pump power drawn
elpower=df[1,4]
#COP of heat pump for given hour
cop=df[:,5]
#heat storage kapasiteetti Wh
heatstorage_cap=df[1,6]
heatstorage_start=df[1,7]
storagepowermax=df[1,8]
```

Figure 5-1. *Reading input data from csv-file*

Input data includes following components:

- Nordpool area price for spot(arbitrary data in this case)
- Solar production estimate
- Heat load estimate
- COP(coefficient of performance) for heat pump
- Heat pump maximum power drawn from the grid
- Heat storage capacity
- Heat storage initial capacity
- Maximum heat storage power

Next part is to initialize the solver and the library.

```
using JuMP
using Clp
m = Model(solver = ClpSolver())
```

Figure 5-2. *Initialization of solver*

The base structure of the optimization problem is also built and the solver selected. Variables and limits are named and limits set. Variables can be set with index number. The index number is the same as the number of the time step. In this case the time step is one hour.

```
#defining variables and limits

@defVar(m, 0<=storagedischarge[1:pituus] <= storagepowermax)
@defVar(m, 0<=storagecharge[1:pituus] <= storagepowermax)
#defining maximum range for heatpump heat output
@defVar(m, 0<=heatpump[1:pituus] <= elpower*10)
@defVar(m, 0<=DH[1:pituus] <= 500)


#Local storage SOC variable
@defVar(m, 0<=storage_SOC[1:pituus] <= heatstorage_cap)

for i in 1:pituus
 @addConstraint(m,heatpump[i]+0<=elpower*cop[i])
end
```

Figure 5-3. *Defining variables and their limits*

The variables in the code are heat storage charge rate, heat storage discharge rate, district heating power and heat pump heat power. Heat power maximum is defined with COP value of ten. Heat balance equation for each hour is defined with for loop using the previously described index.

```
#Heat balance
for i in 1:pituus
 @addConstraint(m,0.95*storagedischarge[i]+DH[i]+heatpump[i]-storagecharge[i] >= heatload[i]-solar_forecast[i])
end
```

Figure 5-4. *Heat balance equation as a constraint*

The state of charge is calculated as a cumulative value of charge rates.

```
#stationary storage SOC constraints
@addConstraint(m,0<=heatstorage_start+storagecharge[1]-0.98*storagedischarge[1]<=heatstorage_cap)
@addConstraint(m,storage_SOC[1]==heatstorage_start+storagecharge[1]-0.98*storagedischarge[1])

for i in 2:pituus
  #if loop is in the final hour, SOC is demanded to be same as initial state
  if i==pituus
    @addConstraint(m,storage_SOC[i-1]+storagecharge[i]-0.98*storagedischarge[i]==heatstorage_start)
    @addConstraint(m,storage_SOC[i]==storage_SOC[i-1]+storagecharge[i]-0.98*storagedischarge[i])
  else
    @addConstraint(m,0<=storage_SOC[i-1]+storagecharge[i]-0.98*storagedischarge[i]<=heatstorage_cap)
    @addConstraint(m,storage_SOC[i]==storage_SOC[i-1]+storagecharge[i]-0.98*storagedischarge[i])
  end
end
```

Figure 5-5. *Storage stage of charge kWh unit*

The efficiency for the storage can also be given here. As an example it has a value of 98%. Self-discharge each hour can be added to equation also to count as time related storage loss. Finally, the objective function for the optimization is defined. The objective function is the electricity cost of the heat pump.

```
costs=(1/(1000*cop[1]))*price[1]*heatpump[1]
for i in 2:38
 costs=costs+(1/(1000*cop[i]))*price[i]*heatpump[i]
end
costs
```

Figure 5-6. *Cost function*

The additional costs could be the cost of direct electric heating cost or the district heating cost. Lastly, the optimization is executed with the following code:

```
@setObjective(m, Min, costs)

status = solve(m)

println("Objective value: ", getObjectiveValue(m))
println("heat = ", getValue(heatpump))
```

Figure 5-7. *Code to run calculation and print results*

The only things to specify for the solver at this point is the optimization task name (m) and to minimize the objective function. The result of the objective value and variable(s) are printed after the calculation is ready.

6. Results

As an example simulation case we have the housing co-operative Aristoteles from Espoo. They have three separate buildings. Heating is provided mainly by district heating but also heat recovery system with booster electric heaters are used.

In this case we focus on the possibilities to have 100 to 300 kW of solar collectors with heat storage and air-to-air heat pump as optional sources for district heating.

Figure 6-1. *A view from street to the buildings*

The main task is to investigate the savings in respect to heat storage size in the example week between 4th to 9th of February. Using only district heating for the period, total heat cost would be 1027€. Air-to-air heat pump has a maximum electricity consumption of 20 kW. Actual maximum heat output depends on outside temperature as simulated in chapter 3.

Here are the results from calculations with different collector and storage capacities.

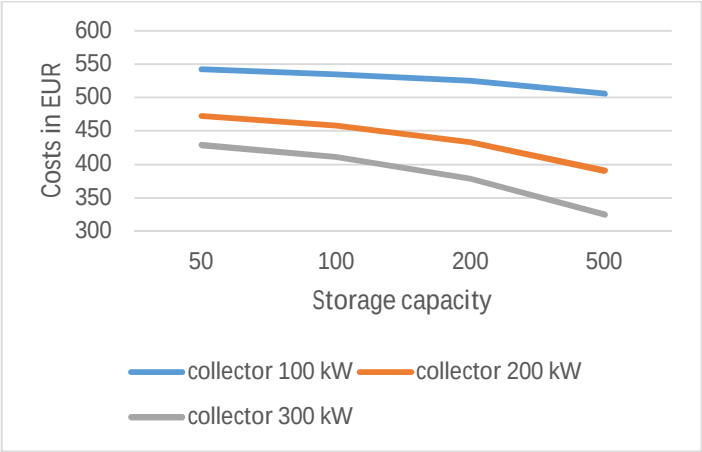

Figure 6-2. *Operation cost results with different collector and storage capacity*

The benefits from a storage are much larger with collector capacities of 200 and 300 kW than with 100 kW.

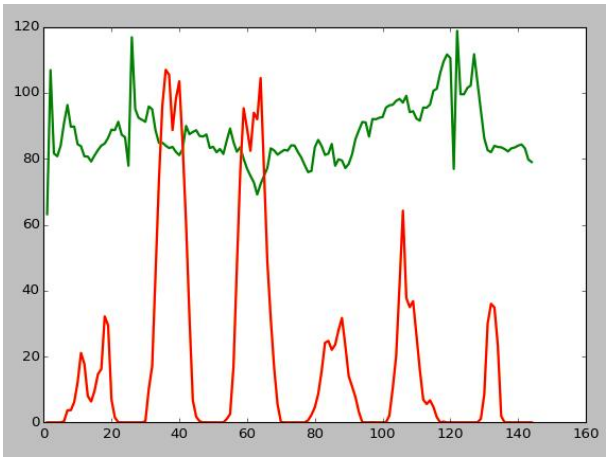

Figure 6-3. *Heat load (green) with a 100 kW solar collector simulated output (red)*

As the previous figure shows, 100 kW solar heat capacity fits almost entirely under the heat load curve, so there is no much benefit from storage. But that is not the only benefit from heat storage. Because there is also a heat pump available in the case, shifting load from an hour to another yields savings in electricity costs of the pump.

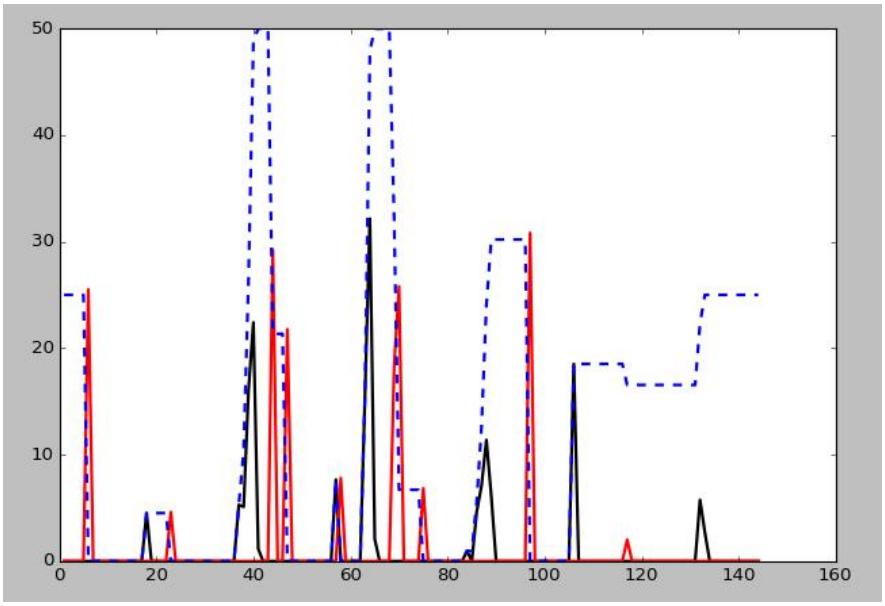

Figure 6-4. *The optimum heat storage usage. State of charge blue(kWh), recharging heat storage in black(kW) and discharging heat storage in red(kW).*

The figure shows that storage is charged to full capacity(50 kWh in this case) when there is a very sunny day. Figure also shows that load is shifted with the storage also in day when the sunshine is weaker.

6.1 Accuracy of forecasts

The optimization system of course needs reasonably good forecasts. Here are the previously described solar and load forecasts.

The heat load is forecasted using forecasted temperatures and compared to measurement data. The heat load forecast model succeeds in capturing the heat demand level, see Figure 6-5, although hourly variations can be seen. The three short-term valley-spikes found in the forecast stem from corresponding irregularities in the past measurements. First two of them can be avoided by increasing the robustness parameter in the model for this forecasting target. The third spike is more difficult, as it also involves a spike in the temperature measurement. Overall, a data management module could easily be added to identify and take actions in case of abnormal data.

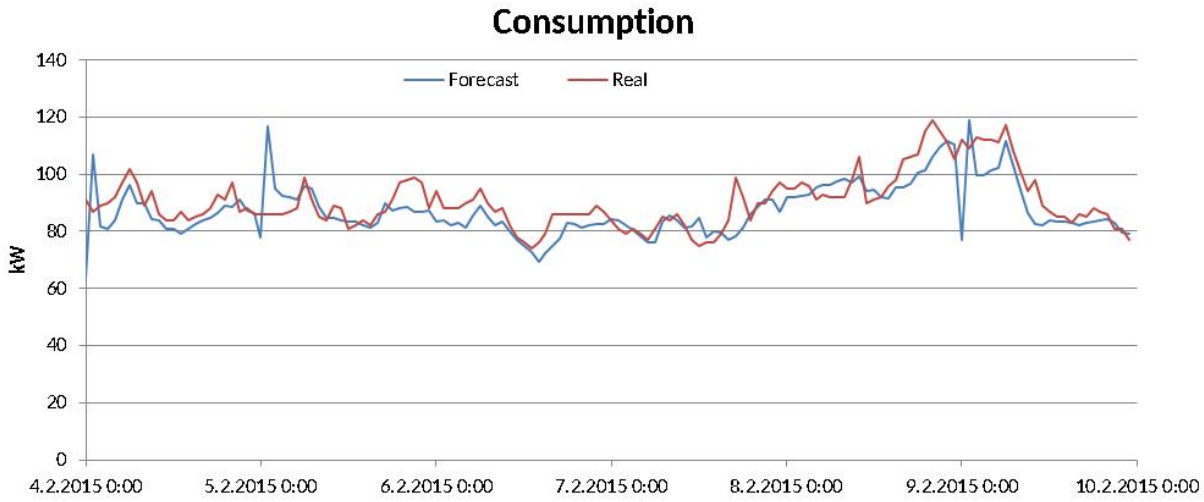

Figure 6-5. *Heat load forecast accuracy*

Solar production is forecasted using weather forecasts, and then compared to production estimates using measured weather parameters. The weather forecasts and measurements are for the given nearest weather locality, not the building itself. Solar production forecast errors eventually stemming from the collectors themselves and the surroundings of the building and other local aspects are thus not assessed.

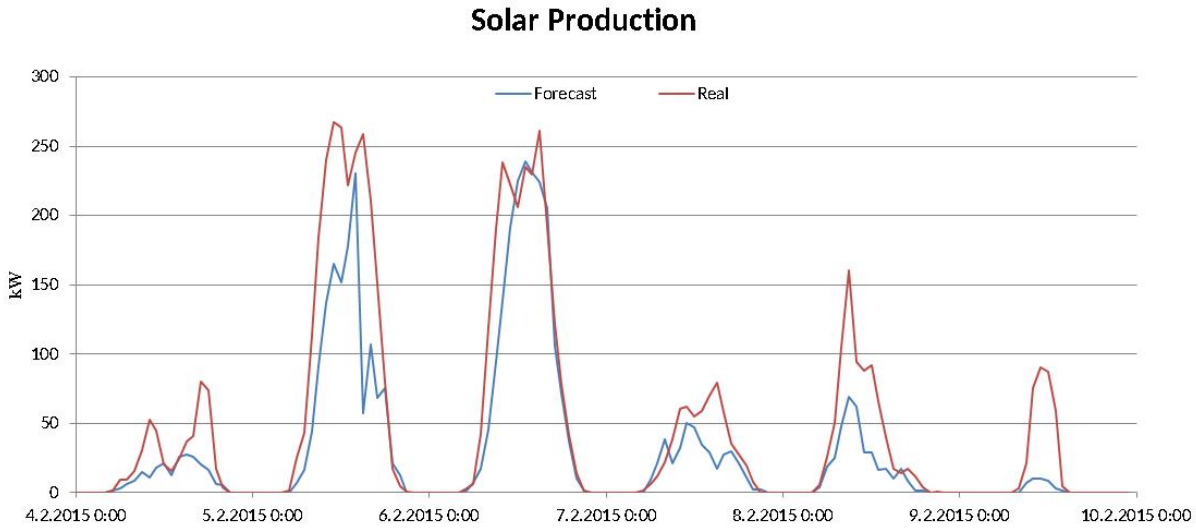

Figure 6-6. *Solar forecast accuracy*

There were no large inaccuracies that would hurt the optimization result. Usually it is enough for heat that system can recognize really good days and really bad days in terms of sunshine and prepare the right actions for the day to come.

7. Conclusions

It was possible to get savings in variable heating costs using solar collectors and a heat pump. A heat storage can also result to both with a heat pump and solar collectors. Excess energy from solar collectors can be stored and used later and also load shifted to different hours as the heat pump to takes advantage of cheaper hourly electricity prices.

The results indicate that with the low prices of electricity that we have now in Nordic market, it is possible to have cheaper energy than district heating. However due to the limited scope of the work here, the maintenance fees are not simulated. The cost difference can however be up to 50% without taking account the maintenance costs. For a total profitability analysis, investment costs and performance study would be needed to be executed over longer time period.

The main point of the study was to investigate integration of solar radiation and load forecast models to optimization code. Two models were combined via a bit non-optimal way because both had been previously built on different platforms. It was discussed during the project that combined system could be rewritten with python for more elegant approach.

References

Coin-or (2015). Retrieved from Coin-or linear programming Solver: https://projects.coin-or.org/Clp

Bratu, C. (2008). Evaluation of solar irradiance to a flat surface arbitrary oriented. Electrical Engineering series, No.32, 2008, 1842-4805.

Duffie, J., & Beckman, W. A. (2013). Solar Engineering of Thermal Processes 4th edition. Wiley.

Finnish Meterological institute . (2013, December). Open Data Manual. Retrieved from http://en.ilmatieteenlaitos.fi/open-data-manual

Finnish Meterological Institute. (2015, December). Weather Forecast Data HIRLAM. Retrieved from http://ilmatieteenlaitos.fi/avoin-data-saaennustedata-hirlam

Honsberg, C., & Bowden, S. (2013, December). Photovoltaic Education Network. Retrieved from PVeducation: http://pveducation.org/

MIT. (2015). Retrieved from The Julia Language: http://julialang.org/

NASA. (2014). NASA Surface meorology and Solar Energy. Retrieved from https://eosweb.larc.nasa.gov/sse/

PostgreSQL: The world's most advanced open source database. (2015). Retrieved from http://www.postgresql.org/

Zdeněk, B. (2011). My hobby - Heat pump. Retrieved from zbroz: www.zbroz.cz