Empirical modelling of Some Processes

Group 1

Jamel Hudson och Andreas Särnblad

Introduction

Models are used in order to describe the workings of a system - to describe what happens to that system when something is changed within it, or outside of it, without actually performing the experiments on the true system. There are various kinds of ways to model a system but what is common for all the different ways is that the models are useful when they accurately enough describe the true dynamics of the true system.

Empirical modelling is an approach to modelling which uses known observations from a system to identify the dynamics of the system.

A black box model is a model of a system based only on the inputs and outputs from that system without any knowledge of the internal workings of the system. Black box modelling is often “easier” to do because you do not need to know, or care of, any of the internal physical workings of the system; purely modelling a system based on its physical properties might require a lot of mathematical and physical knowledge which may be very hard to acquire.[1]

Grey box models are tailor made models combines knowledge of the internal workings of the system with black box modelling based on data. The models are then tailor made to the problem and are part mathematical, theoretical and part black box. The parameters thus have physical interpretations which is not the case in black box models.[2]

Basic Description of the applications

Wind power

Wind power is the term used for the transformation of wind energy to electrical power in a wind turbine. The amount of energy that can be absorbed by the turbine blades is about half the total energy in the wind. A wind turbine can operate in wind speeds between 4 and 25 meters per second. Therefore its important to place the wind turbine in a place with as constant winds as possible, which in Sweden is at the coastline.[3]

[1] Glad & Ljung, Modellbygge och Simulering, 2004, s. 38
[2] Glad & Ljung, Modellbygge och Simulering, 2004, s. 275
[3] Energikunskap, *Så här fungerar ett vindkraftverk*, retrieved 2018-10-19

Solar heating systems

Solar thermal heating systems do not transform the solar energy but merely moves the energy from a less desirable location to a desirable one. The systems use the solar insolation to heat a solar receiver which in turn heats a medium, usually water or air, which is pumped into a heat storage. Such a system thus consists of a solar receiver, a pump and a heat storage and it is the mass transfer within the system that transports the energy within the system. (Ljung, Glad, s.221)

Task 1: Identification of wind speed dynamics

Description of task

The objective of the task was to estimate the spectrum of the wind speed using nonparametric as well as parametric methods.
The data given in the assignments consisted of a vector containing the measured wind speeds and a corresponding matrix with information on the time for the wind speed data sample measurements.

Theory

The frequency content of signals are described through their spectrums.

The spectrum of a signal, $\omega(t)$, is given by its magnitude of the fourier transform squared

$$\Phi_y(\omega) = |Y(\omega)|^2 \tag{1}$$

and is a measure of the amount of energy contained by the signal in that frequency, and the fourier transform $Y(\omega)$ is[4]

$$Y(\omega) = \int_{-\infty}^{\infty} y(t) \cdot e^{-i\omega t}\, dt \tag{2}$$

The spectral density of a signal, from frequencies ω_1 to ω_2, describes the energies between the two frequencies and is given as[5]

$$\text{Spectral density} = \int_{y_1}^{y_2} \Phi_y(\omega)\, d\omega = \int_{y_1}^{y_2} |Y(\omega)|^2\, d\omega \tag{3}$$

[4] Glad & Ljung, Modellbygge och Simulering, 2004, s. 409
[5] Glad & Ljung, Modellbygge och Simulering, 2004, s. 231

Non-parametric identification

Non-parametric ways of identification may be used to estimate a systems transient response or frequency response. One way of estimating the power (energy) spectrum of a signal is by using the Blackman-Tukey procedure. First the autocorrelation function, R_u, (a.k.a the covariance function), which measures how previous data points of the data varies with current data points, is calculated as[6]

$$R_u(k) = \frac{1}{N} \cdot \Sigma_{t=1}^{N} u(t+k)u(t) \tag{4}$$

Secondly a windows function, γ, is chosen and applied. And thirdly $\widehat{R}_u(k)$ is computed for k=0,..., γ. The power spectrum $\Phi(\omega)$ is then formed as[7]

$$\Phi(\omega) = \Sigma_{k=-\gamma}^{\gamma} \omega_\gamma(k) \cdot \widehat{R}_u(k) \cdot e^{-i\omega k} \tag{5}$$

Parametric identification

The spectrum can also be estimated using parametric models. The ARX-model is on the form[8]

$$y(t) = -a_1 y(t-1) - \ldots - a_{n_a} y(t-n_a) + b_1 u(t-n_k) + \ldots + b_{n_b} u(t-n_k-n_b+1) + e(t) \tag{6}$$

and the prediction of y(t) is then[9]

$$\widehat{y}(t|\theta) = -a_1 y(t-1) - \ldots - a_{n_a} y(t-n_a) + b_1 u(t-n_k) + \ldots + b_{n_b} u(t-n_k-n_b+1). \tag{7}$$

By forming the linear regression matrix[10]

$$\varphi^T(t) = \left[-y(t-1) \; -y(t-2) \ldots \; -y(t-n_a) \; u(t-n_k) \ldots u(t-n_k-n_b+1\right] \tag{8}$$

and the parameter matrix[11]

$$\theta^T = [a_1 \ldots a_{n_a} \; b_1 \ldots b_{n_b}] \tag{9}$$

The prediction of y(t) can then be written as[12]

$$\widehat{y}(t|\theta) = \theta^T \varphi(t). \tag{10}$$

and the loss function is[13]

$$V(\theta) = \frac{1}{N} \cdot \Sigma_{t=1}^{N} \{y(t) - \widehat{y}(t|\theta)\}^2. \tag{11}$$

The parameters minimizing the loss function, in a least square sense, can then be estimated as[14]

$$\widehat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T Y \tag{12}$$

---

[6] Glad & Ljung, Modellbygge och Simulering, 2004, s. 261
[7] ibid.
[8] Glad & Ljung, Modellbygge och Simulering, 2004, s. 283
[9] ibid.
[10] Glad & Ljung, Modellbygge och Simulering, 2004, s. 284
[11] ibid.
[12] Glad & Ljung, Modellbygge och Simulering, 2004, s. 285
[13] Glad & Ljung, Modellbygge och Simulering, 2004, s. 287
[14] Glad & Ljung, Modellbygge och Simulering, 2004, s. 289

where

$$\Phi^T = [\varphi^T(1) \dots \varphi^T(N)], \quad Y^T = [y(1) \dots y(N)]. \tag{13}$$

Method

To identify the dynamics of the measure wind speeds a subset of 2000 samples with constant mean and variance was selected. Then the mean of the data was removed to be able to compare parametric and non-parametric models. The spectral density was then obtained by using the Matlab *systemIdentification* GUI with different values of window size. This plot was compared to the graph in example 10.3 in Ljung & Glad to find a reasonable window size.[15]

The spectrum was then estimated using AR-models with increasing order which were compared to the spectral density obtained with the non-parametric method. The order number which gave a similar result as the non-parametric spectrum was then selected.

The AR-models was then used to predict the wind speed for different prediction horizons.

Results

Figure 1 shows how the spectral density change with different values of window size. The window size that yielded the spectrum density most alike the one in Ljung & Glad was 4.

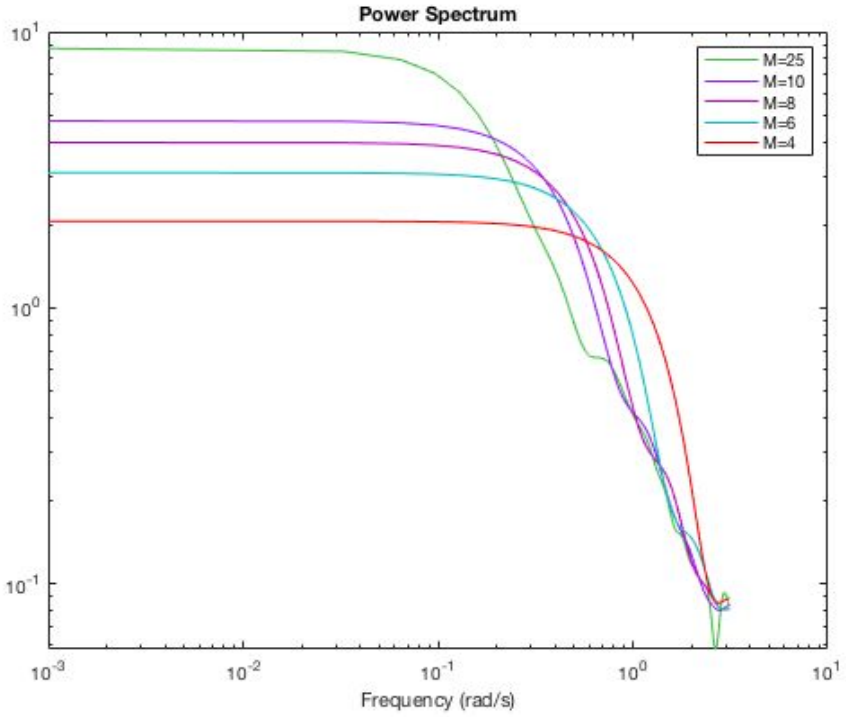

Figure 1. Spectrum analysis for different window sizes.

[15] Glad & Ljung, Modellbygge och Simulering, 2004, s. 237

The non-parametric method of obtaining the wind spectrum with window size 4 in comparison to parametric methods is shown in *figure 2*. The two methods does not show the same results but as seen in the figure the low order models captures the behavior of the non-parametric model the most.

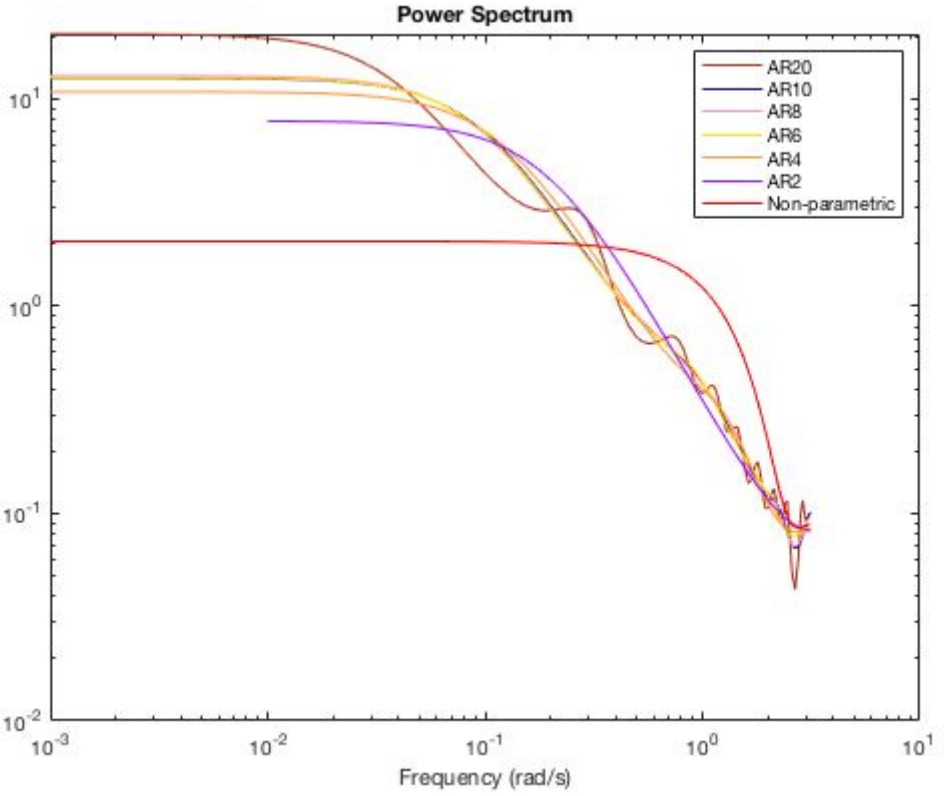

Figure 2. The spectrum of the different AR-models and the non-parametric model.

The AR-models with increasing order predictions of the wind speed is shown in *table 1*. The fit of the AR-models increase with the order but the fit is still quite poor, especially for long time horizons.

Table 1. The fit of different AR-models with different time horizons.

Model	**1 second horizon**	**3 second horizon**	**10 second horizon**
AR2	32.53%	7.53%	-7.53%
AR4	32.92%	8.77%	-5.86%
AR6	33.42%	9.51%	-5.20%
AR8	33.60%	9.69%	-5.06%
AR10	33.63%	9.68%	-5.09%
AR20	32.93%	9.87%	-2.99%

Discussion

The results shows that the spectrum can be very different depending on how window size for non parametric models. The most alike spectrum had a higher value for lower frequencies than the example in Ljung & Glad but was the best fit of the tested window sizes.

For the parametric method the spectrum becomes more unlike the example in Ljung & Glad for all model orders. When the models were used for prediction of the wind speed the results were very bad. This shows that this way of modeling wind speed may not be the best and that the wind behaves very arbitrary, and perhaps it is not possible to accurately model the wind at all many time-steps ahead.

Task 2: A Solar Heating System

Description of task

A solar heating system consisting of a solar receiver, a pump and a heat storage is modeled. The inputs are the solar insolation and the pump speed and the output is the heat storage temperature. The first objective of the task was to find the least square estimates of the coefficients a_i, b_i and c_i, i=1,2 for a linear ARX model of the system with model structure

$$y(t) + a_i y(t-1) + a_2 y(t-2) = b_1 u(t-1) + b_2 u(t-2) + c_1 I(t-1) + c_2 I(t-2) + e(t)$$

using calibration data and then to simulate the model on a second data subset.

The second objective was to calibrate a grey-box model for the same system and to compare it to the black-box model with respect to mean square error.
The third objective was to reduce the order of the grey-box model one parameter at a time and to find which order that resulted in the lower mean square error.

Theory

A solar house as described above in the section *Solar heating systems* with solar insolation I(t), pump speed u(t) and heat storage temperature y(t) can be shown to the grey box modeled as:

$$y(t) = \theta_1 \varphi_1 + ... + \theta_6 \varphi_6 \qquad (14)$$

The regressors of the grey-box model could according to Ljung and Glad be described as:

$$\begin{aligned} \theta_1 &= (1 + d_1) & \varphi_1(t) &= y(t-1) \\ \theta_2 &= -(d_3 - 1) & \varphi_2(t) &= (y(t-1)u(t-1))/u(t-2) \\ \theta_3 &= (d_3 - 1)(1 + d_1) & \varphi_3(t) &= (y(t-2)u(t-1))/u(t-2) \end{aligned} \qquad (15)$$

$$\theta_4 = d_0 d_2 \qquad \varphi_4(t) = u(t-1)I(t-2)$$
$$\theta_5 = -d_0 \qquad \varphi_5 = y(t-1)u(t-1)$$
$$\theta_6 = d_0(1+d_1) \qquad \varphi_6(t) = u(t-1)y(t-2)$$

where $d_0,\ d_1,\ d_2\ and\ d_3$ represents unknown constants which we want to determine.[16]

The mean square error (MSE) measure used to test the models with respect to the validation data is the value of the loss function given in equation (11).

Method

The system was first modelled with an linear ARX-model with known order. This was done by first removing the mean and dividing the data into two subsets for estimation and validation. The ARX-model was created with the Matlab command “arx” and simulated with “sim” with the estimation and valuation data. The model was evaluated by calculating the MSE.

When estimating the Grey-box model one of the inputs in form of the pump is a binary signal which is why a small number had to be added to it to avoid dividing by zero.
The regressors for the Grey-box model were obtained by an mass balance calculation of the system according to the theory. These regressors were used to calculate the parameters of the grey-box model. The grey-box model was then simulated in a for loop in Matlab for every time step. This was repeated for the model but with all the different combinations of regressors and the MSE was calculated for every model.

Results

The MSE of the simulated black-box model was calculated to 18.54 and its fit to the validation data is shown in *figure 3*. The model captures most of the output of the solar heating system. The corresponding Matlab code is found in Appendix A1.
The least squares estimates of the coefficients a_i, b_i and c_i, for i=1,2, were found to be a_1=-0.697, a_2=-0.0978, b_1=-0.3605, b_2=0.2978, c_1=-0.263 and c_2=0.6229.

[16] Glad & Ljung, Modellbygge och Simulering, 2004, s. 373

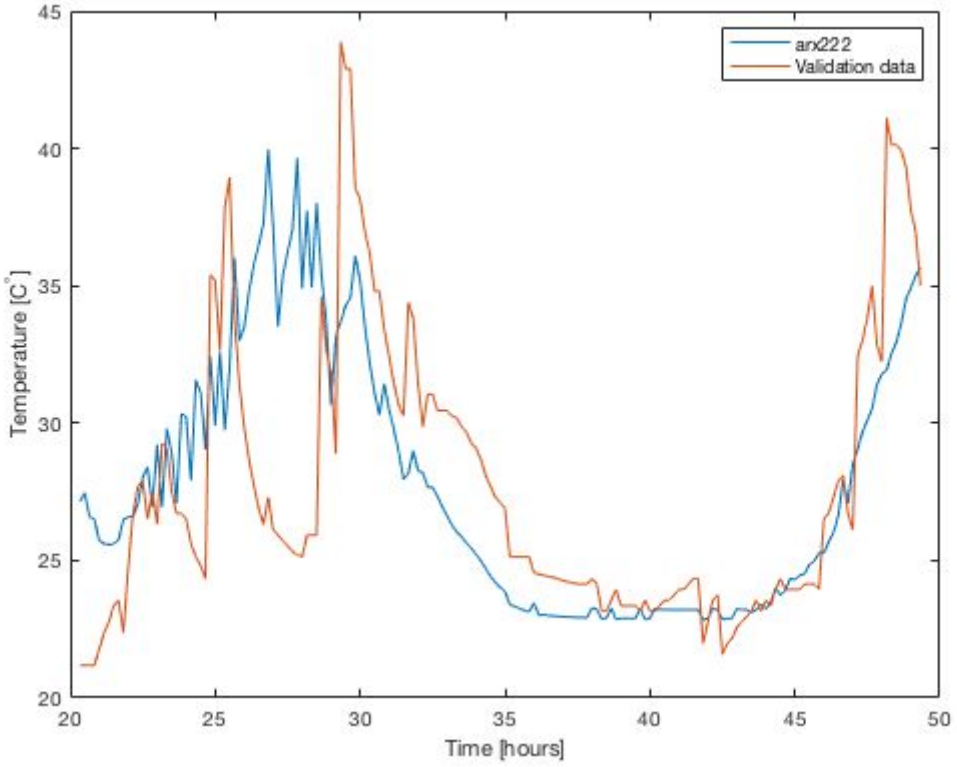

Figure 3. The simulated black-box model valuated with the validation data.

The MSE of the five best grey-box models with different number of parameters is shown in table 2. The MSE indicates that regressor 1,4 and 6 gives the best result. The best model simulated and compared to validation data is shown is *figure 4.*

Table 2. The MSE of the five best Grey-box models.

Included regressors (Numbered as in the theory)	***MSE***
1,4 and 6	*9.92*
1,2,3,4 and 6	*10.08*
1,3,4 and 6	*10.28*
1,2,4 and 6	*10.93*
1,4,5 and 6	*11.93*

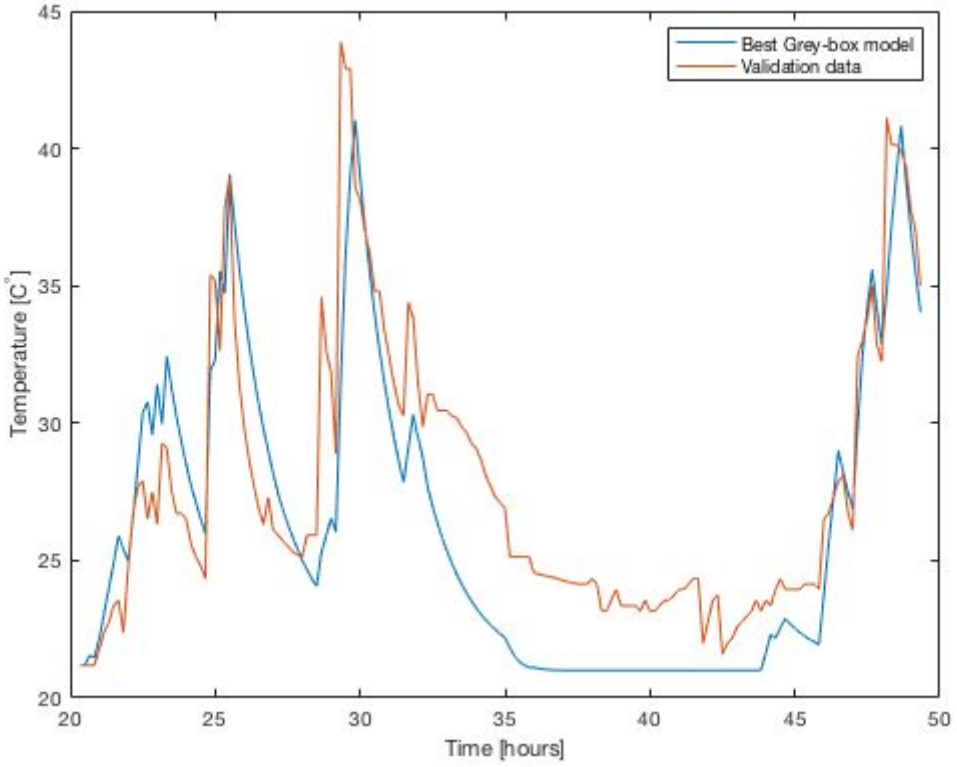

Figure 4. The best grey-box model simulated valuated with the true data.

Discussion

The simulated grey-box proved better fit the validation data than the black-box model for a certain combination of regressors. This was expected due to that it took into account some of the known properties of the system. When analysing the optimal combinations of regressors in the grey-box model, regressor one, four and six was in all of the best five models, which implies that they are the most important for the model.

Task 3: Study of a Neural Network model

Abstract

Artificial neural network are computing systems which can learn from experience, similarly to the animal brain, and thus very popular and interesting because of that property.
In this study of a neural network model a linear ARX model is first developed to simulate an engine, and is then demonstrated to be improved with a nonlinear neural network ARX model. The nonlinear model was obtained with an iterative process that tested 500 different combinations for best fit to validation data.

With the linear ARX model alone a best fit of 77.56% is attained, but with a nonlinear ARX model the fit was improved to 82.29%.

Introduction

Artificial Neural Network models have become very popular in recent years partly due to the increase of cheap and readily accessible computational power. Such models are mildly inspired and similar to the working of the brain in the way that they learn with experience.

The concept of neural networks has been around for decades, some of the most important methods was discovered in the 1960s and 1970s. Neural networks became a explicit research subject in the 1990s and was developed even more with the use of unsupervised learning. Even more complex neural networks have recently become more relevant due to its advanced capability in visual pattern recognition. [17]

Neural networks has the advantage over other technologies in pattern detection and complex modeling. The applications of pattern detection is mainly different types of identification of time-varying signals for example speech or sonar as well as image recognition. In cases when a system is described by a nonlinear process, neural networks can be used due to the nonlinear transforms it uses. Neural networks has for example been used in models for collision avoidance in aircrafts. [18]

Neural network models consists of an input layer, weighted connections, neurons, activation functions and the output itself. The layers of neurons constitute the “hidden layers”. Artificial neural networks with many layers are said to be Deep. The inputs are multiplied with weights which are summed in each neuron. The output of these neurons are then fed into activation functions whose outputs are then multiplied with different parameters (weights) and summed in other neurons. This repeats for each hidden layer, and the outputs of the final hidden layer forms the outputs which are estimation of the true model outputs.
The general form of the One hidden layer feedforward neural network model is[19]

$$\hat{y}(t|\theta) = \sum_{k=1}^{d} \alpha_k k(\beta_k(\varphi - \gamma_k)) \qquad (16)$$

where a_k, β_k, φ and γ_k are the activation functions, scaling parameters, regressors and weights.

[17] Jurgen Schmidhuber, Neural Networks volume 61, 2015
[18] Maren et al, Handbook of neural computing, 1990.
[19] Glad & Ljung, Modellbygge och Simulering, 2004, s.330

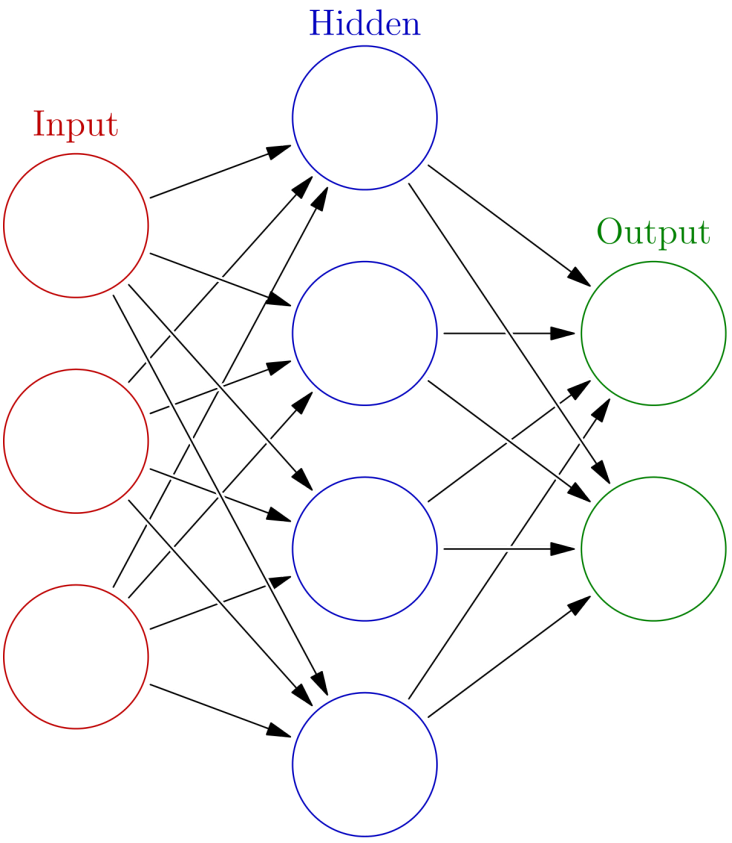

Figure 5. An illustration of a neural network with one hidden layer and four neurons.

The learning rule (or learning process) is the rule, or algorithm, by which the network achieves a better result (e.g. predicting the output better in a least squares sense) by modifying some parameters of the model, typically the weights.[20]

The activation functions (also known as transfer functions) in the neural networks define the transformation of the input data to output data. Some commonly used activation functions are:

- Linear function $y(t) = t$
- Rectified linear unit $y(t) = 0 \, for \, t < 0, \; 1 \, for \, t \geq 0$
- Sigmoid function $y(t) = (1 + e^{-t})^{-1}$
- Gaussian (radial basis) function $y(t) = e^{-t^2}$

There are different kinds of network types but the first and simplest one is the feedforward network wherein the connections between the nodes only go forward and not backwards, hence the structure involves no loops or cycles and is not recurrent. The feedforward network is on the form as illustrated in *figure 5*.

[20] Zell, *Simulating Neural Networks,* 1994.

In this task a dataset consisting of measurements of an engine is used to train a neural network (nonlinear ARX) model and is compared to the linear ARX model. The input signal is the voltage (V) and the output is the rotational speed of the motor (RPM/100) and the sampling time (Ts) which is 0.04 seconds.

The method

The mean value of the data sets was first removed then it was split up in training and validation data, 1000 and 500 data points respectively.
The System Identification toolbox was used to find the best linear ARX-models, with respect to best fit, given the constraint that na, nb, and nb be less or equal to 10.
The *Matlab* function *feedforwardnet* was used to create a feedforward neural network with number of hidden layers and the number of neurons in each layer as input. To train the network and to obtain the non linear model the function *nlarx* was used. The function *compare* was used to estimate the fit to validation data .
Then different neural network structures of the non-linear ARX model was created in a for loop with different combinations of neurons in the different hidden layers. The number of hidden layers was restricted to two and the number of possible neuron was between 1 and 10 due to simulation time restrictions. The common logarithmic sigmoid transfer function was used for both layers.
The number of combinations for the non-linear ARX-model with these inputs was 100. The models were compared to the validation data and the best fits were sorted out. The regressors of the five best linear ARX models were then used to test the 100 different networks.

Illustration

The input and output signals of the engine are illustrated in *figure 6*.

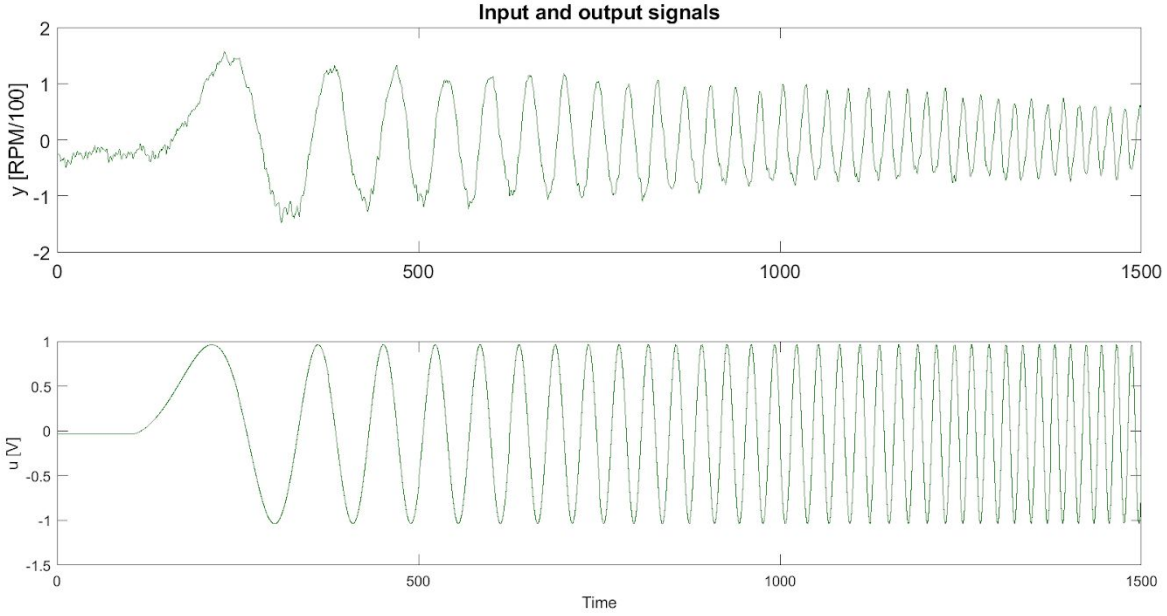

Figure 6. The input and output signals of the engine.

The linear ARX model with the best fit was the ARX(10 2 5) with a fit of 77.56% as is displayed in *Table 3*.

Table 3. The tested linear models and their fit.

Model	**Fit**
ARX(7 2 5)	76.62%
ARX(7 1 8)	75.39%
ARX(5 2 5)	76.09%
ARX(9 2 5)	76.63%
ARX(10 2 5)	77.56%

The non-linear ARX model with the best fit was the ARX(7 1 8) with a fit of 82.29% as is displayed in *Table 4*.

Table 4. The tested non-linear models and their fit and net structure.

Model	**Fit**	**Neurons in hidden layer 1**	**Neurons in hidden layer 2**
nlARX(7 2 5)	79.08%	2	5
nlARX(7 1 8)	82.29%	4	3
nlARX(5 2 5)	78.53%	1	3
nlARX(9 2 5)	81.26%	4	4
nlARX(10 2 5)	79.59%	4	5

The nonlinear ARX-models of *Table 4* were simulated and plotted against the validation data which is shown in *figure 7.*

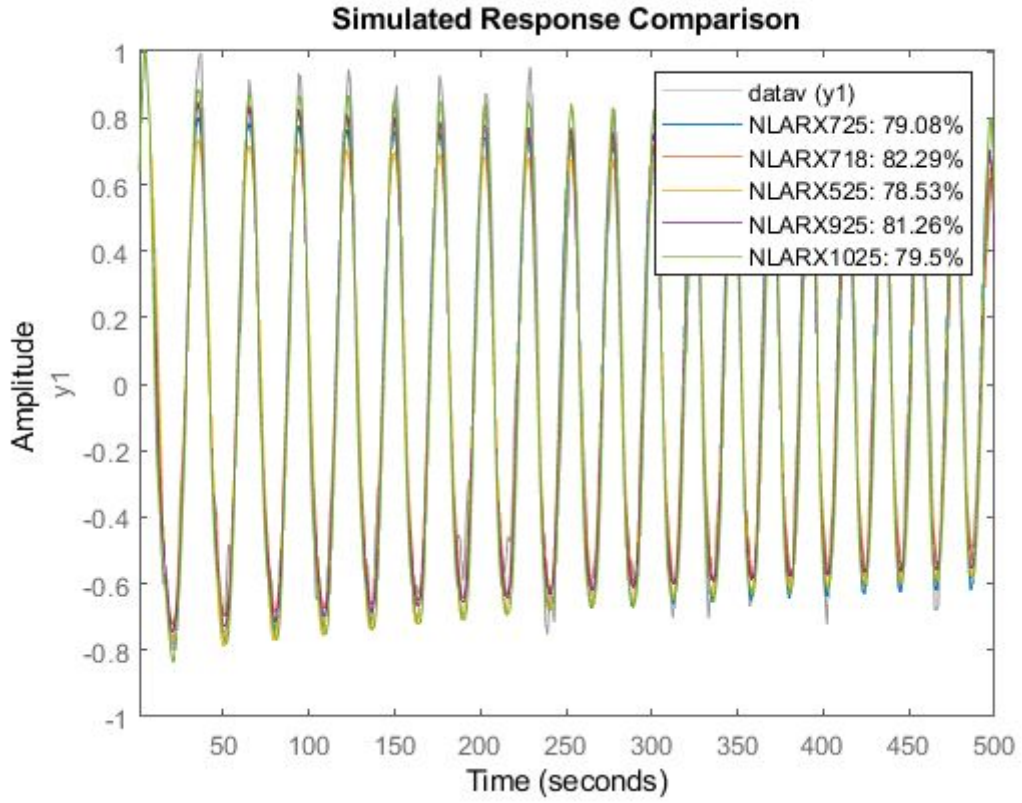

Figure 7. The simulated nonlinear models compared to validation data.

Conclusion

The best fit that could be obtained with an linear ARX-model was 77.56%. With the use of a neural network with 4 neurons in its first and 3 in its second hidden layer the ARX-model (7 1 8) could reach a fit of 82.29 %. This shows that linear ARX-models can be improved with the use of a non-linear part that is derived from a neural net.

The iterative process of finding the best linear part of the nonlinear model and the neural nets structure demonstrates that obtaining this kind of model is not always trivial.

Optional task 8.6: Modelling of the solar heating system using artificial neural networks

Description of task

In this task the solar heating system in Task 2, which consisted of a solar receiver, a pump and a heat storage, is modelled using artificial neural networks with the the aim of getting an even better result, in a means square error sense, than that of the greybox model in Task 2 which had a MSE of 9.92. The previous linear ARX221 black box model had a MSE of 18.54.

Theory

Neural networks, as described in *Task 3: Study of a Neural Network*, consists of an input layer, weighted connections, neurons, activation functions and the output itself. The weights are the parameters being estimated and are the parameters being adjusted in the *learning process* in order to minimize loss functions. In the process of designing a neural network the depth of the network, i.e. how many hidden layers it consists of, the number of neurons per hidden layer and the activation functions are the parameters to be considered when designing the network. The network will then through the *learning process*, by experience, adjust weights to improve its result. It is not simply so that a deeper network or a network with more neurons is always better, so therefore when designing such a network one needs to consider many options which can be done through an iterative process, e.g. a *for* loop.

Method

To simulate the solar heating system with an nonlinear model, the same method as in task 3 was used. The regressors used for the nonlinear model was the same as for the black box model in the first part of task 2. The number of hidden layers was increased to three and possible neurons in each layer for the neural network was kept the same as in task 3 for simplicity.
The obtained nonlinear model was then simulated with the same data as the models in task 2 and its MSE was calculated and compared to the other models.

Results

The MSE of the nonlinear model using neural networks was 8.91 corresponds to a 10% reduction of the previous Grey-box MSE of 9.92 which is significant. The simulated results of the non linear model and the grey-box model are shown in *figure 8*.

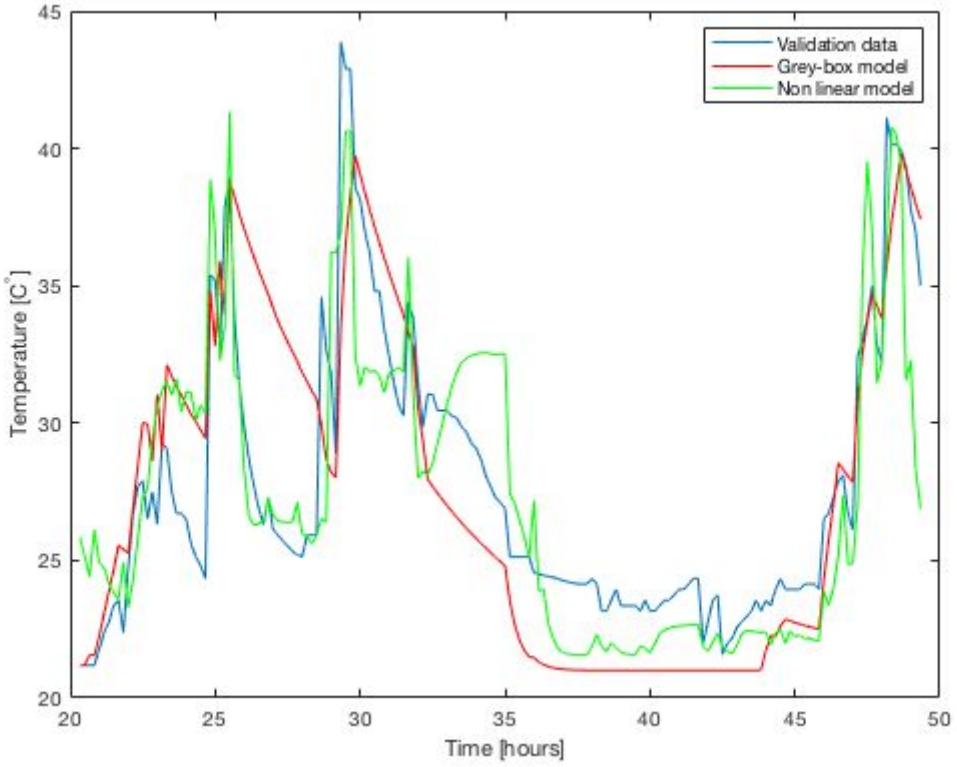

Figure 8. The simulated results of the non linear model and the grey-box model.

Discussion

In this task of improving upon a model of a nonlinear solar heating system a better results was achieved than that of the greybox model which may be due to the nonlinearity of the activation functions, in combination with the powerful iterative learning process of the network. We did not expect the neural network to be so powerful in reducing the MSE so much so fast, that we did not need to try different activation functions other than the common sigmoid function or different regressors, but it would be interesting to do so. Moreover, increasing the number of combination of neurons and layers to say 10000 from 1000 might also yield much better results, or not; we simply don’t know but it would also be interesting to study.

References

Energikunskap, 2011,retrived 09/10/2018 from:
http://www.energikunskap.se/sv/FAKTABASEN/Vad-ar-energi/Energibarare/Fornybar-energi/Vind/Sa-har-fungerar-ett-vindkraftverk/

Jurgen Schmidhuber, Neural Networks volume 61, 2015

Maren et al, Handbook of neural computing, 1990, Academic press, San Diego

Glad & Ljung, *Modellbygge och Simulering*, 2004, Studentlitteratur AB, Lund

Zell, *Simulating Neural Networks,* Chapter 5.2 1994

Appendix

A2.1 Linear Black-Box Modelling - Matlab Code

```
clear all;
load("solardat.mat");
time = (122:1:296)*10/60';

u1_m = flakt-mean(flakt);
u1_Est = u1_m(1:121);
u1_Val = u1_m(122:296);

u2_m = sol-mean(sol);
u2_Est = u2_m(1:121);
u2_Val = u2_m(122:296);

y_m = temp-mean(temp);
y_Est = y_m(1:121);
y_Val = y_m(122:296);

model = arx([y_Est, u1_Est,u2_Est],[2,2,2,1,1]);
y_Hut=sim(model,[u1_Val,u2_Val]);

for i = 1:length(y_Val)
   d(i) = (y_Val(i)-y_Hut(i))^2;
end

MSE = sum((y_Val-y_Hut).^2)/length(y_Val);

disp("MSE: " + MSE)
plot(time,y_Hut,time,y_Val)
legend("arx222","Validation data");
```

```
xlabel("Time [hours]");
ylabel("Temperature [C^{\circ}]")
```

A2.2 Grey Box modelling matlab code

```
clear all;
load("solardat.mat");
time = (121:1:296)*10/60';
temp = temp - 21; %Ta bort utomhustemperatur
y_Est = temp(1:120);
y_Val = temp(121:296);
x = 0.01; %korrigeringsfaktor
u1_Est = flakt(1:120)+x;
u1_Val = flakt(121:296)+x;
u2_Est = sol(1:120);
u2_Val = sol(121:296);
N = length(y_Est);
M = length(y_Val);

y_Hut = zeros(length(y_Val),63);
y_Hut(1,:) = y_Val(1);
y_Hut(2,:) = y_Val(2);
comb = combvec([0:1],[0:1],[0:1],[0:1],[0:1],[0:1]);
comb(:,1) = [];
for i = (1:length(comb))
   phi_est=[];
   if (comb(1,i)==1)
   phi_est(:,end+1) = [y_Est(2:(N-1))];
   end
   if (comb(2,i)==1)
   phi_est(:,end+1) = [y_Est(2:(N-1)).*u1_Est(2:(N-1))./u1_Est(1:(N-2))];
   end
   if (comb(3,i)==1)
   phi_est(:,end+1) = [y_Est(1:(N-2)).*u1_Est(2:(N-1))./u1_Est(1:(N-2))];
   end
   if (comb(4,i)==1)
   phi_est(:,end+1) = [u1_Est(2:(N-1)).*u2_Est(1:(N-2))];
   end
   if (comb(5,i)==1)
   phi_est(:,end+1) = [y_Est(2:(N-1)).*u1_Est(2:(N-1))];
   end
   if (comb(6,i)==1)
   phi_est(:,end+1) = [u1_Est(2:(N-1)).*y_Est(1:(N-2))];
   end
   theta = phi_est\y_Est(3:120);

   for j = 1:M-2
```

```
    phi=[];
  if (comb(1,i)==1)
  phi(end+1) = y_Hut(j+1,i);
  end
  if (comb(2,i)==1)
  phi(end+1) = y_Hut(j+1,i).*u1_Val(j+1)./u1_Val(j);
  end
  if (comb(3,i)==1)
  phi(end+1) = y_Hut(j,i).*u1_Val(j+1)./u1_Val(j);
  end
  if (comb(4,i)==1)
  phi(end+1) = u1_Val(j+1).*u2_Val(j);
  end
  if (comb(5,i)==1)
  phi(end+1) = y_Hut(j+1,i).*u1_Val(j+1);
  end
  if (comb(6,i)==1)
  phi(end+1) = u1_Val(j+1).*y_Hut(j,i);
  end
  y_Hut(j+2,i) = phi*theta;
  end
MSE(i) =  sum((y_Val-y_Hut(:,i)).^2)/(M-2);
end

plot(time,y_Hut(:,41)+21,time,y_Val+21)
legend("Best Grey-box model","Validation data");
xlabel("Time [hours]");
ylabel("Temperature [C^{\circ}]")
```

A3 Neural Network - Matlab Code

```
clf
load icEngine.mat
%detrend
y=detrend(y);
u=detrend(u);

%delar upp i est. och val. data
ye=y(1:1000);
yv=y(1001:1500);
ue=u(1:1000);
uv=u(1001:1500);

%skapar iddata objekt
datae=iddata(ye,ue);
datav=iddata(yv, uv);
```

```
%skapar olika arx-modeller
m2=arx(datae,[7 2 5]);

a=1:10;
b=1:10;
cv=combvec(a,b);

for i=1:length(cv)
ff = feedforwardnet(cv(:,i)');
ff.layers{1}.transferFcn = 'logsig';
ff.layers{2}.transferFcn = 'logsig';
A{i} = nlarx(datae,[7 2 5],neuralnet(ff));
end

[y,fit]=compare(data2,A{1:length(cv)});
FIT=cell2mat(fit);
Max=max(FIT)
```

A4. Extra Task Matlab code

```
clear all;
load("solardat.mat");
time = (122:1:296)*10/60';

u1_m = flakt-mean(flakt);
u1_Est = u1_m(1:121);
u1_Val = u1_m(122:296);

u2_m = sol-mean(sol);
u2_Est = u2_m(1:121);
u2_Val = u2_m(122:296);

y_m = temp-mean(temp);
y_Est = y_m(1:121);
y_Val = y_m(122:296);

model = arx([y_Est, u1_Est,u2_Est],[2,2,2,1,1]);

a=1:10;
b=1:10;
cv=combvec(a,b);

for i=1:length(cv)
ff = feedforwardnet(cv(:,i)');
```

```
ff.layers{1}.transferFcn = 'logsig';
ff.layers{2}.transferFcn = 'logsig';
A{i} = nlarx([y_Est, u1_Est,u2_Est],[2,2,2,1,1],neuralnet(ff));
end

[y,fit]=compare([y_Val, u1_Val,u2_Val],A{1:length(cv)});
FIT=cell2mat(fit);
max=max(FIT)

y_Hut=sim(A{59},[u1_Val,u2_Val]);
MSE = sum((y_Val-y_Hut).^2)/length(y_Val)
```