Introduction

In this exercise the basics of non-parametric modelling techniques are introduced through an example where the heat load of a building is modelled. The aim is to give a hands-on experience with the techniques for modelling non-linear effects - using statistical models, where the non-linear functions are modelled without specifying the exact parametric function.

An example of a non-linear *parametric* function is

$$f(x;a) = x^a$$

which has the parameter a. If a is positive and not 1 then its non-linear and smooth. A *non-parametric* function is also non-linear and smooth, but its mathematical formula cannot be expressed explicitly and thus it doesn't have parameters. However, in the process of fitting a non-parametric function to data, several types of parameters are used - we could call them "smoothing parameters". They determine some characteristics of the function, mostly related to how much the function can "bend" and thus can much it can follow the data: If it doesn't follows data adequately, then the function (i.e. model) becomes "under-fitted" to the data, but if it follows data too much, then the model is said to be "over-fitted" to data. The model which is balanced and thus neither under- and over-fitted is called a "suitable model".

The following data is available for three houses:

- P_t (`P` in data) the heat load for the building (W)
- T_t^e (`Te` in data) the external temperature (°C)
- G_t (`G` in data) the global radiation (W/m^2)
- W_t^s (`Ws` in data) the wind speed (m/s)

The observations are 2 day average values. The internal air temperature was not measured, but during the winter period the heating was thermostatically controlled and thus the internal air temperature can be assumed constant.

In the questions below base spline and kernel functions are introduced. The exercise is then to apply them to build and fit non-parametric models to the data - first by "manually" tuning the smoothing parameters and then automizing this using cross-validation techniques. Finally, it will be shown how the techniques can be used realize a so-called semi-parametric model.

Q1: Plot the data and fit a linear regression model

Open the script "`q1_read_data_and_lm.R`". Get familiar with the data and fit the linear regression with heat load P_t as model output and external temperature T_t^e as input

$$P_t = \beta_0 + \beta_1 T_t^e + \varepsilon_t$$

where β_0 and β_1 are the model parameters and ε_t is the error. Until the last question, we do not consider G_t and W_t^s.

How does a linear regression model fit the data? When using the winter period only? When including the entire period?

So check if there are patterns left in the residuals, i.e. check the assumption that the residuals are independent and identically distributed (i.i.d.).

Q2: Base splines intro

Now open the script "`q2_base_splines.R`". The aim of this question is to give an idea of how base splines behave and what the effects of smoothing parameters are. Play around with the `bs()` function.

Try to vary the degrees of freedom (`df`). What happens to the base splines generated?

Try to vary the degree (`degree`). What happens to the base splines generated?

Give the knot points directly (using the `knots` argument). Give the knots as the quantiles of `x`, what happens when `degree = 1` and what happens when `degree = 3`? (exactly how the base splines are generated when `degree > 1` is left out of scope of the exercise, just think about that it is done using some "optimal" procedure).

Try with some non-equidistant `x` sequence, such that the quantiles are not equidistant. What happens with the base splines?

Q3: Base splines model

Go and open the script "`q3_spline_model.R`". Now we want to calculate the base splines as a function of the external temperature. And then use them as input to a linear regression model. In this way, it becomes a non-linear model

$$P_t = \beta_0 + f(T_t^e) + \varepsilon_t,$$

where β_0 is the intercept and $f()$ becomes a spline function. The characteristics of $f()$ depends on how the base splines are generated, hence it does not have any explicit parameters - only the smoothing parameters of the base splines.

Try to fit a model, is it linear or how is it shaped?

Try to change the degrees of freedom (`df`), what happens?

Q4: Kernel functions

Go and open the script "`q4_kernels.R`". Another way to make non-parametric models is to use locally weighted regression. To do this we need a kernel function. There exists many kernel functions, one of the simplest is the triangular kernel

$$k(x_1, x_2, h) = \begin{cases} 1 - \frac{|x - x_i|}{h} \text{ for } |x - x_i| < h, \\ 0 \text{ for } |x - x_i| \geq h, \end{cases}$$

where x_1 and x_2 are two points and h is the *bandwidth*.

Try to calculate and plot the triangular kernel and play around with the parameters. How do they affect the shape of the kernel?

Try to calculate and plot the Epanechnikov kernel and play around with the parameters. How do they affect the shape of the kernel?

Q5: Locally weighted model with a kernel function

Go and open the script "`q5_kernel_model.R`". Fit a locally weighted model for a single point and predict the heat load.

Fit for a sequence and make the plot of the function. Try to change the bandwidth h. How does that change the estimated function between T_e and P?

Q6: Tuning of the smoothing parameters

Go and open the script "`q6_model_tuning.R`". Now we face the challenge of finding the optimal values for the smoothing parameters, either the bandwidth h in the kernel or the degrees of freedom for the base splines. If the model is over-fitted it varies too much (the function is too adapted to the training data and bends around too much), and on the other hand if it is under-fitted, then it is not “bending” and adapting enough to the data.

One approach is to do a cross-validation optimization of a score function. In the case of estimating the (conditional) mean value, the score function should almost always be the Root Mean Squared Error (RMSE).

The most robust way is to do *leave-one-out*:

- Fit the model without observation i
- Predict observation i

Do this for all the observations and then calculate the score function using the predictions.

In this way we can find the right balance between under- and over-fitting (i.e. the suitable model).

Carry out leave-one-out cross validation. Try to change the bandwidth. What happens to the RMSE score?

Of course we cannot use our time doing manually optimization, so use an optimizer to optimize the bandwidth. Does the result look reasonable for the locally weighted model?

Use leave-one-out cross validation for the base spline model. Does the result look reasonable?

For the base spline models a model selection criteria, such as AIC or BIC can be used. Try that for the base spline model and compare. Do you get the same results?

Q7: Semi- and conditional parametric models (optional for reporting)

Until now, we have calculated the weights and base splines using the input to the model, namely T_t^e. What if we used another variable, but still fitted the same model. What will then happen?

Lets try to calculate the weights using the time t. By doing that we actually allow the coefficients in the model to change as a function of time. We would usually in this case add a t to the parameters

$$P_t = \beta_{0,t} + \beta_{1,t} T_t^e + \varepsilon_t,$$

indicating that they change over time.

This model can of course also be fitted using base splines.

Go and open the script "`q7_semi_parametric_models.R`".

Describe how the coefficients change over time.

What happens with the coefficients during the summer?

Can you explain the result in relation to how the heating system of the building is operating?

Q8: More aspects (optional for reporting)

In this question we will deal with two aspects:

- How to apply a 2. order local model and use another type of kernel function
- Investigate the effect of external temperature, conditional on the wind speed

First, go and open "`q8_more_aspects.R`".

In the first part: 2. order inputs are included into the model, hence it is now a local polynomial model. By including these, the curvature of the function is better estimated, hence this can, when the function is "bending" a lot, lead to a better fit.

Try changing the formula `frml` to find the model which minimizes the cross-validated score.

What is the best model you can find, does it have any second order inputs?

What happens to the bandwidth found with cross-validation when a 2. order term is included?

Compare the results of the `tri()` and `epanechnikov()` kernel function. Does is seem like one of them lead to slightly better fits?

In the second part: Investigate the effect of the external temperature conditional on the wind speed.

How does the coefficient for T_e change as a function of W_s?

Can you explain these results based on your knowledge from physics about building heat transfer?

Software for non- and semi-parametric modelling

It can be an advantage to use packages in R, since they often provide more advanced functionality and can make the workflow easier – most popular seems to be `np` for

kernel regression and `gam` for spline models. Of course packages are available in other languages, e.g. Python or C++, use some search engine to find the them.

Installation

If you did not install the package `ctsmr` in advance, then you need to do it now. See first the web page `ctsm.info` for OS specific instructions.

Introduction

This exercise is about grey-box modelling of the heat dynamics of a building using stochastic differential equations (SDEs). Further, the properties of the PRBS signal and the use of likelihood ratio tests for model selection are introduced.

The data consists of averaged values over five-minute intervals of:

- T_i (`yTi` in data) the average of all the indoor temperatures measured (one in each room in the building). The sensors were hanging approximately in the center of each room (°C)
- Φ_h (`Ph` in data) the total heat output for all electrical heaters in the building (kW)
- T_a (`Ta` in data) the ambient temperature (°C) (notice, that in other material T_e is used as the ambient (external) temperature. In this exercise T_e is used as envelope temperature)
- G (`Ps` in the data) the global radiation (kW/m^2)
- W_s (`Ws` in data) the wind speed (m/s)

The climate variables were measured with a climate station right next to the building. See Bacher and Madsen (2010) (it is in the file `Identifying_suitable_models_for_heat_dynamics.pdf`) for more details of the experiments in which the data was recorded (it is included in the .zip file).

Q1: Fit and validate a grey-box model

Open the script "`r/q1_fit_and_validate.R`". Remember to change the path with `setwd()` (in line 5) to set the working directory to the where the script file is located (in RStudio menu "Session->Set Working Directory->To Source File Location" can be used).

The simplest applicable SDE grey-box model has one state and consists of the system equation

$$dT_\mathrm{i} = \left(\frac{1}{R_\mathrm{ia}C_\mathrm{i}}(T_\mathrm{a} - T_\mathrm{i}) + \frac{1}{C_\mathrm{i}}g_\mathrm{A}\Phi_\mathrm{s} + \frac{1}{C_\mathrm{i}}\Phi_\mathrm{h}\right)dt + \sigma_\mathrm{i}d\omega_\mathrm{i} \tag{1}$$

and the measurement equation

$$Y_k = T_{\mathrm{i},k} + \epsilon_k \tag{2}$$

where k counts the measurements from 1 to N and where the measurement error is assumed to follow a normal distribution with $\epsilon_k \sim N(0, \sigma_\epsilon^2)$.

See Bacher and Madsen (2010) (it is in the file `Bacher2011.pdf`) for a description of the different parts and parameters in the model.

The RC-diagram in Figure 1 illustrates the deterministic part of the system equation.

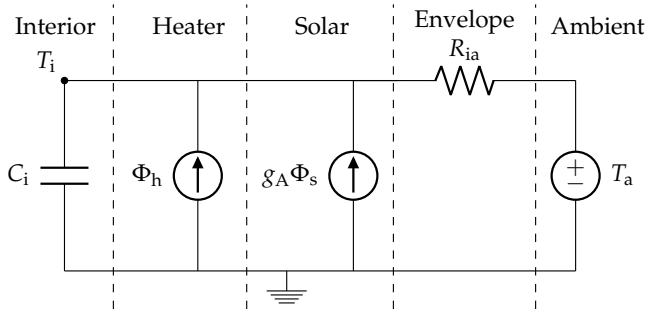

Figure 1 *RC-network of the most simple model Ti.*

Run the script line by line and read the comments in the code to see how a model and data is specified. Stop in the section where the parameter optimization is run and thereafter proceed here:

- Estimate the parameters in the simple model Ti in Equation (1) and see the estimated values with `summary(fit)`. Is the estimation successful (i.e. does the minimization of the negative log-likelihood converge)?
- Actually, the initial value and the boundary for one of the parameters are poorly set. You can see if the parameter estimates are close to one of their boundaries from the values of

  - $\frac{dF}{dPar}$ which is the partial derivative of the objective function (negative log-likelihood).
  - $\frac{dPen}{dPar}$ which is the partial derivative of the penalty function.

  If the value of $\frac{dPen}{dPar}$ is significant compared to the value of $\frac{dF}{dPar}$ for a particular parameter it indicates that a boundary should be expanded for the parameter. Correct one of the boundaries and re-estimate until the partial derivatives are all very small. Which boundary was not set appropriately?

- The one-step predictions (residuals) are estimates of the system noise (i.e. the realized values of the incremental $d\omega$ of the Wiener process) added together with the observation noise. The assumption is that the one-step predictions are white noise. Validate if this assumption is fulfilled by plotting the auto-correlation function and the accumulated periodogram for the residuals. Is the model suitable, i.e. does it describe the heat dynamics sufficiently?

- Next it is possible to gain some information about what is missing in the model, with time series plots of the residuals and the inputs. Generate the plots. Are there some systematic patterns in the residuals? If yes, do they seem to be related to the inputs? To any specific events in the inputs?

Q2: Model selection

Open "`r/q2_q3_model_selection.R`" and run the first part. As instructed in the script file go and open the file "`r/functions/Ti.R`" and see how the model Ti is defined there, it is simply wrapped in the function - simply to make the code more readable. Further, see that the model validation is also put into a function.

Now we will try to extend the simplest model, where a single part of the model is extended. For example a state variable representing the temperature in the building envelope T_e. Such a state is called "a hidden state" since it is not measured. The RC-diagram for this model is shown in Figure 2. The system equations are

$$dT_i = \left(\frac{1}{R_{ie}C_i}(T_e - T_i) + \frac{1}{C_i}g_A\Phi_s + \frac{1}{C_i}\Phi_h\right)dt + \sigma_i d\omega_i, \tag{3}$$

$$dT_e = \left(\frac{1}{R_{ie}C_e}(T_i - T_e) + \frac{1}{R_{ea}C_e}(T_a - T_e)\right)dt + \sigma_e d\omega_e \tag{4}$$

and the measurement equation is still

$$Y_k = T_{i,k} + \epsilon_k \tag{5}$$

Three other extensions of the simple model is also suggested. Their RC-diagrams are shown the figures 3, 4, 5.

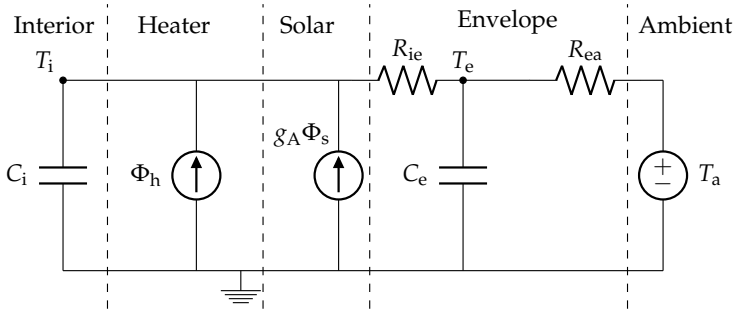

Figure 2 *RC-network of the most simple model extended with a state in the building envelope TiTe.*

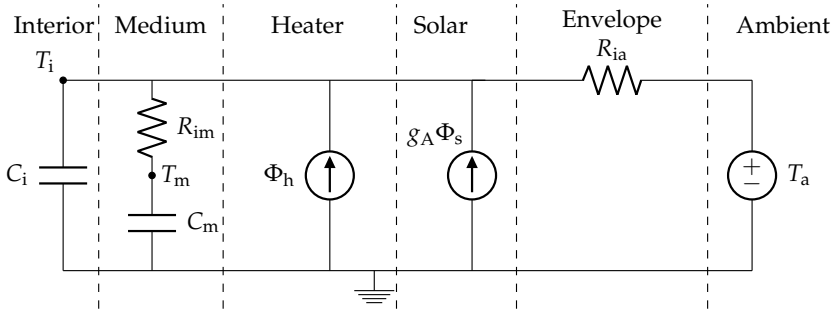

Figure 3 *RC-network of the most simple model extended with a state in which the solar radiation enters TiTm.*

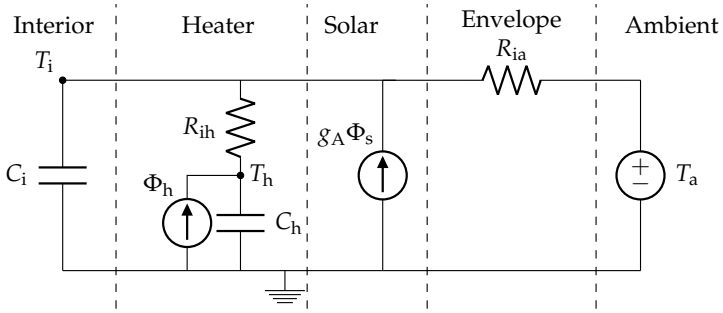

Figure 4 *RC-network of the most simple model extended with a state in the heater TiTh.*

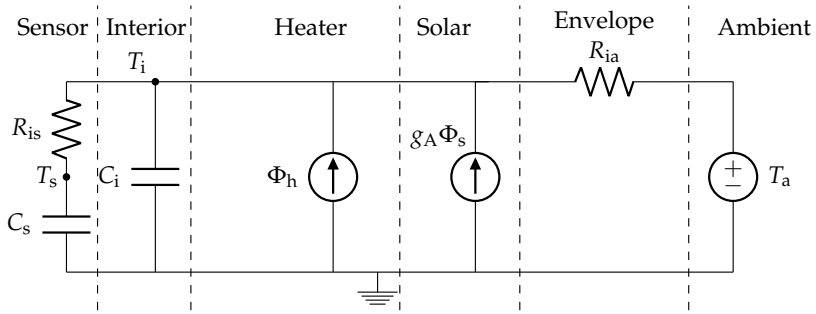

Figure 5 *RC-network of the most simple model extended with a state in the sensor TiTs.*

- Estimate the parameters in the four models (they are implemented in the functions with equivalent names).
- Are the extended models improved regarding the description of the dynamics (hint, analyse the residuals)?

Q3: Likelihood ratio test

Use the script to carry out a likelihood ratio test of the simple model to each of the extended models. If the p-values show that more than one of the extended models are a significant improvement, it is suggested to select the extended model with the highest maximum likelihood.

- Which of the four models should be selected for further extension?
- Take the selected model and extend it again once more, see the functions with 3-state models.

From this point the selection and extension procedure should be continued until no significant extension can be found, however this is beyond the scope of the exercise. In order to see what happens if you stay in that track read the article Bacher and Madsen (2011) (the .pdf is in the .zip file).

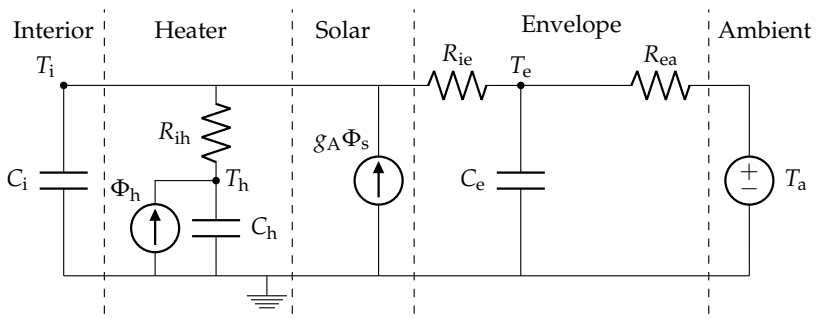

Figure 6 *RC-diagram of TiTeTh.*

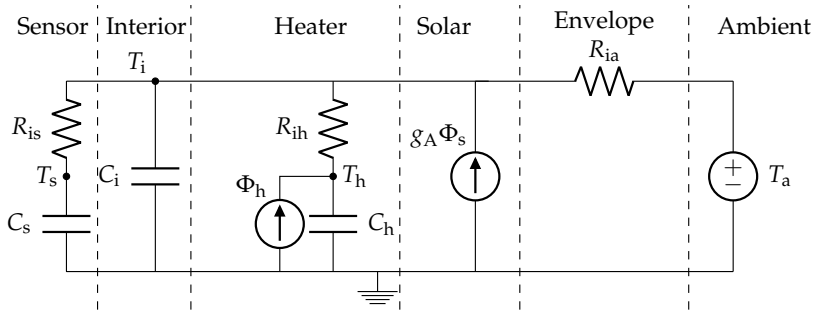

Figure 7 *RC-diagram of TiThTs.*

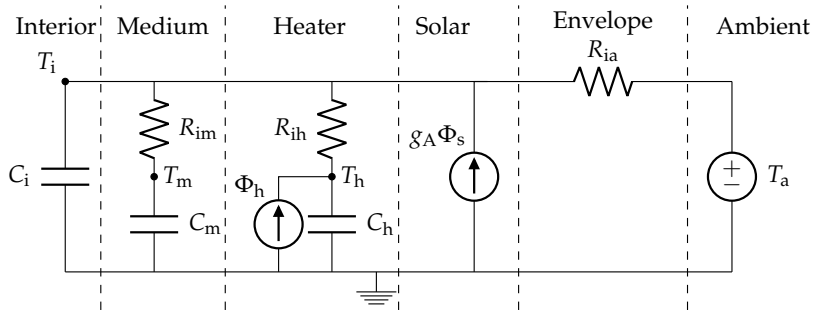

Figure 8 *RC-diagram of TiThTm.*

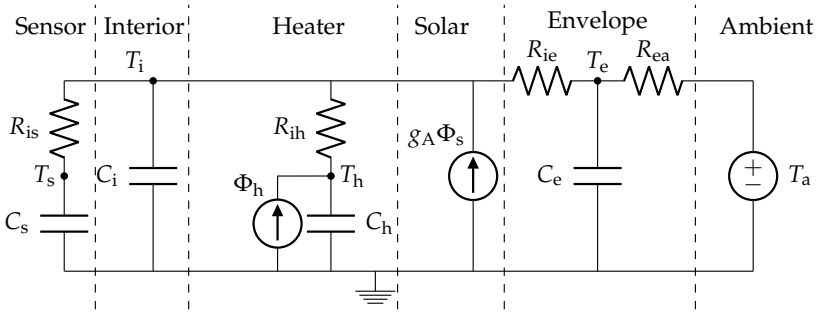

Figure 9 *RC-diagram of TiTeThTs.*

Question 4

This part deals with Pseudo Random Sequence Signals. See Godfrey (1980) for a detailed description of PRBS signals. Go and open "`r/q4_prbs.R`".

Answer the following questions:

- Do the plotting of the data. Which of the signals is a PRBS signal?
- Which of the other signals are highly dependent on the PRBS signal?
- The function `prbs()` is an implementation of the n-stage feedback registers in the paper (see the function definition in "`r/functions/prbs.R`). Generate a PRBS signal as in the script and investigate its properties with the ACF.

Optional:

- In order to generate multiple PRBS signals, they will be independent. Try generating two PRBS signals, lag one of them and plot the cross-correlation function.
- Again consider the plots of data. Can you already from this see a lot about the dynamics of the house (think about the step response)?

Finally, the R-code which was used to generate the PRBS signals used in the experiment is given, together with the code for generating three independent PRBS signals.

Questions and comments: `pbac@dtu.dk`

References

P. Bacher and H. Madsen. Experiments and data for building energy performance analysis : Financed by the danish electricity saving trust. Technical report, DTU Informatics, Building 321, Kgs. Lyngby, 2010.

P. Bacher and H. Madsen. Identifying suitable models for the heat dynamics of buildings. *Energy & Buildings*, 43(7):1511–1522, 2011. ISSN 03787788. doi: 10.1016/j.enbuild.2011.02.005.

K. Godfrey. Correlation methods. *Automatica*, 16(5):527–534, 1980. ISSN 00051098.

Installation

If you did not install the package `ctsmr` in advance, then you need to do it now. See first the web page `ctsm.info` for OS specific instructions.

Introduction

The exercise is about grey-box modelling of the heat dynamics of a (small) building using stochastic differential equations (SDEs). In addition to the first exercise on greybox modelling, we will in this exercise use different techniques to:

1. Alter the noise level (or system uncertainty) over time to take into account changing uncertainties.
2. Build a semi-parametric model to take into account that the solar penetration (i.e. relation between measured solar radiation and radiation entering into the building) as function of the position of the sun.
3. Balance heat gains to the air temperature and the temperature of the thermal mass.

The data consists of several measurement from a small test box with a single window. In this exercise the following signals are used:

- T_i (`yTi` in data) the observed indoor temperatures. (°C)
- Q_i (`Qi` in data) the heat emitted by the electrical heaters in the test box (W)
- T_e (`Te` in data) the external (or ambient) temperature (°C)
- G_v (`Gv` in the data) the vertical south total solar radiation (W/m^2)
- G_{vn} (`Gvn` in data) the vertical north total solar radiation (W/m^2)

A full description of the data and the test setup can be found in the document `ST3 CE4 Instruction document.pdf`.

Q1: Noise level

Open the script "`q1_system_noise_levels.R`". Remember to change the path with `setwd()` (in line 5) to set the working directory to the where the script file is located (in RStudio menu "Session->Set Working Directory->To Source File Location" can be used).

First we will work with the two-state model from the the exercise *Grey-box models and model selection*. The RC-diagram for the deterministic part of the model is shown in Figure 1. The system equations are

$$dT_\mathrm{i} = \left(\frac{1}{R_\mathrm{iw}C_\mathrm{i}}(T_\mathrm{w} - T_\mathrm{i}) + \frac{1}{C_\mathrm{i}}g_\mathrm{A}\Phi_\mathrm{s} + \frac{1}{C_\mathrm{i}}\Phi_\mathrm{h}\right)dt + \sigma_\mathrm{i}d\omega_\mathrm{i} \tag{1}$$

$$dT_\mathrm{w} = \left(\frac{1}{R_\mathrm{iw}C_\mathrm{w}}(T_\mathrm{i} - T_\mathrm{w}) + \frac{1}{R_\mathrm{we}C_\mathrm{w}}(T_\mathrm{e} - T_\mathrm{w})\right)dt + \sigma_\mathrm{w}d\omega_\mathrm{w} \tag{2}$$

and the measurement equation is

$$Y_k = T_{\mathrm{i},k} + \epsilon_k \tag{3}$$

where k counts the measurements from 1 to N and where the measurement error is assumed to be i.i.d. and follow a normal distribution $\epsilon_k \sim N(0, \sigma_\epsilon^2)$.

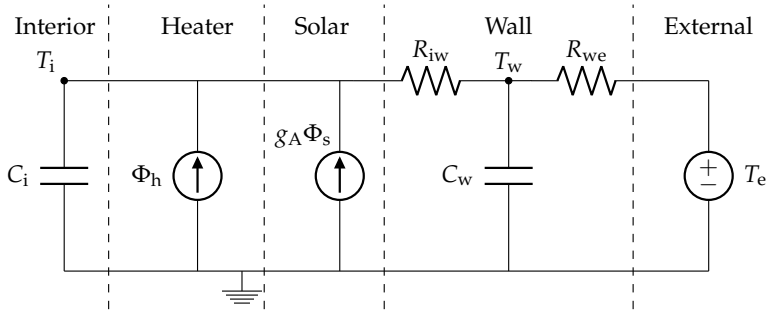

Figure 1 *RC-network of the most simple model extended with a state in the building envelope TiTw.*

Run the script line by line, stop right after plotting the data.

- The lower time series plot is of `stepQi`, which goes from 0 to 1. Try to change the argument `samples_after_Qi_step` above in the function preparing the data. How does it change `stepQi`?
- Now compare the two models implemented in `functions/sdeTiTw.R` and `functions/sdeTiTw_sigmalevels.R`. What is the difference?

Now go to the script and fit the two models. Compare the results:

- What is plotted in the upper two plots? (You maybe have to look into the `analyzeFit()` function).

- What is indicated by the blue lines in the upper plot?
- Step back in the plots and compare the results, and look at the summary output. Which of the two models will you prefer and why?

So it becomes clear that we have some (possible non-linear) dynamics when the heating turns on and off, which our models doesn't predict so well. But instead of adding a more detailed description to the deterministic part of the model, we simply vary the system noise, or in other words, change the uncertainty level of our states under under different conditions. This is a very useful thing, since there will be many phenomena in buildings, especially occupied buildings, which will lead different to levels of noise, e.g., solar radiation and occupants doing funny things.

Finally, go through the last part of the script under "Nice features of R and Rstudio" to learn some nice tricks for modelling in R and RStudio.

Q2: gA-curve modelled with spline

The "solar gain" is defined as the ratio of: heat absorbed inside the building to the measured solar power. So far we have assumed that the solar gain is simple always proportional to the radiation measured outside the test box. In reality, the solar gain depends highly on building geometry, surroundings, window properties, etc. In this part of the exercise, we will apply base splines to estimate the solar gain as a non-parametric function of the sun position.

The solar gain is more specifically referred to as the gA value (g is the percentage of solar heat that enters through the window, multiplied with the window area A).

We will start out using a trick to learn how the solar gain behaves. We make a hidden state for `gA`, such that we can investigate if it changes over time and further if it changes as the function of other variables - in particular the sun position.

The state equation which is added to the model is

$$dg_{\mathrm{A}} = \sigma_{\mathrm{a}} d\omega_{\mathrm{a}} \tag{4}$$

hence g_{A} is now not a parameter, but a state – which is modelled as a random walk and will be estimated when fitting the model.

Open the script "`q2_splined_ga_value.R`" and run it line by line. Stop after you have plotted the estimated state of `gA` as a function of time, and the state of `gA` as a function of the sun azimuth.

It is clear that the state of `gA` not is constant, but change. Furthermore, it seems like there could be a relation to the sun azimuth.

Now, answer the questions below as you progress in the script and apply base splines to model the gA curve:

- The spline function we want to estimate is the gA value as a function of the sun azimuth. First, plot the sun elevation as a function of the sun azimuth, as well as a horizontal line through 0 (notice that the angles is in radians). Find the azimuth angles (in radians) that corresponds to the sunrise and sunset, and assign them to `azumith_bound <- c(...  , ...)` below. These two angles will be used as the boundary knots. Outside the boundaries the gA value is 0, as the sun is below the horizon and the radiation is zero. Thus, we are only interested in the gA curve from sunrise to sunset.

- Generate the base splines and stop after you have merged the base splines to the dataframe with the command `X <- cbind(X,Xbs)`. Now play around with the four parameters in in the vector `gA` and plot the resulting spline function. Try different values to get an understanding of how the base splines and the resulting spline function behave. What happens if `gA` only consists of 1's?
(Tip: the package `lubridate` is very useful when working with dates and time. Which often is the case for when dealing with time series!)

- Fit the model and investigate the estimated parameters. Are the parameters `gA1`, `gA2`, `gA3` and `gA4` significant different from zero? and is the magnitude reasonable when the actual glazed area is 52 x 52 cm?

- Plot the estimated gA curve and the 95% confidence interval. The window in the test box is facing south towards an open area, and should therefore be rather unobstructed. If the estimated gA curve have a shape which is asymmetrical around south (180 degrees), what could be an explanaitionS?

- (Optional) If you think you know why the gA curve is asymmetrical, try to expand or modify the model to take this into account.

You can have a look into the report `CE4_Denmark_DTU_v2.pdf` Chapter 6, where the same things with the gA-value is carried out for a discrete model (an ARX model).

(Optional) Q3: Balancing heat gains

Now, there are other ways to improve the TiTw model that we started with. Open `q3_balanced_heat_gains`:

- Estimate the parameters in the model `fitTiTw_X` and in the model `fitTiTw_GinTw_X`. Which of the models has the highest (log) likelihood?

- The second model has included a parameter, `p`, which is used to control the proportions of the solar radiation entering the states `Ti` and `Tw`. Open the script for the function `sdeTiTw` and `sdeTiTw_GinTw` and compare them. How is the solar gain modelled, and for which value of `p` does the model `sdeTiTw_GinTw` correspond to in the first model `sdeTiTw`?

- What is the estimate of the parameter, `p`, and is it significantly different from 0?
- Explain in words what the meaning of a small and a large value of `p` means. Based on the estimated parameter, `p`, is it reasonable to assume that the solar radiation entering the building solely should be assigned directly to the air temperature?
- With the model `sdeTiTw_sigmalevels` as starting point, setup a model that includes two layers in the wall and a parameter, `p`, to divide the solar radiation between `Ti` and `Tw`.

As it is the case for the solar radiation, the heat input from the heating system can also enter into different states. Until now, we have assigned it directly to the indoor air temperature, `Ti`. To which state the heat should be assigned depends on the response time of the heating system. E.g. an electrical heat blower has a much faster response time than a built-in floor heating system, and should most likely not be assigned a state with very slow heat dynamics.

- Open the script of the function `sdeTiTw2_QinTw` and check how the model is defined. Compared to the previous model, we have introduced an additional layer in the wall and assigned the heat input from the heating system to the inner wall. Fit the model `fitTiTw2_QinTw_X` and assess – only from the log-likelihood – if it has improved compared to the previous model.
- Eyeball the residual plots (ACF, cummulated periodogram, and the residuals as function of time). Why does it not seem reasonable to conclude that the model has improved?
- What does it mean in physical terms when we include an additional state for the heating system, `Th`, as done in the model `sdeTiThTw2`?
- Look closely at the plots for the fit `fitTiThTw2_X`. What seems to drive the large fluctuations in the residuals, and what can the reason be?

Questions and comments: `pbac@dtu.dk` or `chrras@dtu.dk`

Intro

In this exercise you will work with setting up models, which are useful for many applications of forecasting in energy systems. The models are setup in a two-stage approach: first transformations of the inputs are applied and then an estimation method is applied.

The exercise first deals with load forecasting – and thereafter solar and wind forecasting. It starts by introducing a simple linear low-pass filter for input transformation (and base splines), which is then used together with linear regression for load forecasting. Then the recursive least squares (RLS) estimation method is introduced and use for load forecasting.

For solar and wind forecasting both base spline and kernel methods are used.

In the exercise numerical weather predictions (NWPs) are used as model input. The first step is to understand how they are set up. Each input is set up in a matrix, which for each time t holds *the latest available forecasts along the row* for the variable

$$\boldsymbol{u}_t^{\mathrm{nm}} = \begin{array}{c} \begin{array}{cccc} \mathbf{k1} & \mathbf{k2} & \dots & \mathbf{k}n_k \end{array} \quad \begin{array}{c} \leftarrow \mathbf{horizon} \\ \downarrow \mathbf{time} \end{array} \\ \begin{pmatrix} u^{\mathrm{nm}}_{2|1} & u^{\mathrm{nm}}_{3|1} & \dots & u^{\mathrm{nm}}_{1+n_k|1} \\ u^{\mathrm{nm}}_{3|2} & u^{\mathrm{nm}}_{4|2} & \dots & u^{\mathrm{nm}}_{2+n_k|2} \\ \vdots & \vdots & & \vdots \\ u^{\mathrm{nm}}_{t|t-1} & u^{\mathrm{nm}}_{t+1|t-1} & \dots & u^{\mathrm{nm}}_{t-1+n_k|t-1} \\ u^{\mathrm{nm}}_{t+1|t} & u^{\mathrm{nm}}_{t+2|t} & \dots & u^{\mathrm{nm}}_{t+n_k|t} \end{pmatrix} \begin{array}{c} 1 \\ 2 \\ \vdots \\ t-1 \\ t \end{array} \end{array} \tag{1}$$

where

- the notation e.g., $u^{\mathrm{nm}}_{t+1|t}$ means the forecast for time $t+1$ available at time t.
- t is the counter of time for equidistant time points. In this notation normalized, such that the sampling time is $t_1 - t_0 = 1$ (the time stamps are then just kept in another vector).
- t_0 is the first available time point.
- n_k is the length of the forecasting horizon.
- The column names are indicated above the matrix, they are simply a **k** concatenated with the value of k.

Hence, with a prediction horizon $n_k = 24$, having data from time $t_0 = 1$, then at time $t = 100$ we would have the following matrix

$$\boldsymbol{u}^{\text{nm}}_{100} = \begin{array}{c} \begin{array}{cccc} \textbf{k1} & \textbf{k2} & \dots & \textbf{k24} \end{array} \leftarrow \textbf{horizon} \\ \downarrow \textbf{time} \\ \begin{pmatrix} u^{\text{nm}}_{2|1} & u^{\text{nm}}_{3|1} & \dots & u^{\text{nm}}_{25|1} \\ u^{\text{nm}}_{3|2} & u^{\text{nm}}_{4|2} & \dots & u^{\text{nm}}_{26|2} \\ \vdots & \vdots & & \vdots \\ u^{\text{nm}}_{100|99} & u^{\text{nm}}_{101|99} & \dots & u^{\text{nm}}_{123|99} \\ u^{\text{nm}}_{101|100} & u^{\text{nm}}_{102|100} & \dots & u^{\text{nm}}_{124|100} \end{pmatrix} \end{array} \begin{array}{c} \\ \\ 1 \\ 2 \\ \vdots \\ 99 \\ 100 \end{array} \tag{2}$$

The model output is simply a vector, so for the above example it is

$$y_t = [y_0\ y_1\ \dots\ y_{100}]. \tag{3}$$

The time t can then just be thought of as an index (in the R code we use just i and the time stamps are kept in a vector).

Q1: Data setup and linear regression

Go and open "`q1_data_setup_and_lm.R`". First the data is loaded and in steps you try to see how it is setup.

The main point is, that in order to fit a model for the k'th horizon you will need to lag the forecast input, e.g. for $k = 1$, and setup the data like

$$\boldsymbol{X}_1 = \begin{pmatrix} y_1 & \text{NA} \\ y_2 & u^{\text{nm}}_{2|1} \\ y_3 & u^{\text{nm}}_{3|2} \\ \vdots & \vdots \\ y_{t-1} & u^{\text{nm}}_{t-1|t-2} \\ y_t & u^{\text{nm}}_{t|t-1} \end{pmatrix} \tag{4}$$

and for $k = 10$

$$\boldsymbol{X}_{10} = \begin{pmatrix} y_1 & \text{NA} \\ \vdots & \vdots \\ y_{10} & \text{NA} \\ y_{11} & u^{\text{nm}}_{11|1} \\ y_{12} & u^{\text{nm}}_{12|2} \\ \vdots & \vdots \\ y_{t-1} & u^{\text{nm}}_{t-1|t-11} \\ y_t & u^{\text{nm}}_{t|t-10} \end{pmatrix} \tag{5}$$

Now the point is, that if we want a forecast model for k steps ahead, then we can simply use `lm()` in R on this data.

Try to learn how the data is setup, divide into a training and a test set, fit a linear regression model for $k = 1$ step ahead. Does the forecast look reasonable?

Try to calculate a forecast for $k = 36$ steps ahead, try for a different house. Give an example and a summary of what you find.

Q2: Low-pass filtering

Now we do know that there are dynamical effects, such that the heating doesn't change immediately when the ambient temperature change, that's why we usually will use a time series model (discrete ARMAX or continuous GB), however here we introduce a slightly simplified way to do it.

Lets write up an ARX model

$$\phi(B)Y_t = \omega(B)u_t + \varepsilon_t$$

and rewrite it into transfer function form

$$Y_t = \frac{\omega(B)}{\phi(B)}u_t + \varepsilon_t$$

and let

$$Y_t = H(B)u_t + \varepsilon_t$$

where $H(B)$ is a transfer function.

We also know, that the response of a building can be modelled as an RC network, which lead to a low-pass filtering effect. Hence, we can apply a low-pass filter to the input and the use the filtered values in the linear regression.

Go and open "`q2_lowpass_filter.R`" and apply a low-pass filter to the generated on/off signal. It is the simplest first order low-pass filter with stationary gain of one

$$H(B) = \frac{1 - a_1}{1 - a_1 B} \tag{6}$$

What happens with the relation between the input and the low-pass filtered signal e.g. whats the relation between the time constant and a1?

What about the stationary gain? (i.e. the limit y approaches)

Q3: Load forecast model

Go and open "`q3_load_forecast.R`". Fit a model with low-pass filtering on the inputs

$$P^{\mathrm{h}}_{t+k|t} = \beta_0 + \beta_1 H(B) T^{\mathrm{a}}_{t+k|t} + \beta_2 H(B) G_{t+k|t} + \varepsilon_{t+k|t} \tag{7}$$

So note that *first* the low-pass filter is applied, such that e.g. $H(B)T^{\mathrm{a}}_{t+k|t}$ is the input used in `lm()`.

Is the model tuned for the particular building heat dynamics?

In order to tune the low-pass filter coefficients (one for the ambient temperature and one for the solar radiation) apply an optimizer to minimize the RMSE. In principle, we should apply a leave-one-out cross-validation on the test set, however that is quite time consuming, and it will make only a very small difference (only a few parameters are fitted), so we do fit "in-sample".

Are the forecasts improved in terms of RMSE?

Finally, include a diurnal curve using base splines

$$P^{\mathrm{h}}_{t+k|t} = f(t^{\mathrm{day}}_t) + \beta_1 H(B) T^{\mathrm{a}}_{t+k|t} + \beta_2 H(B) G_{t+k|t} + \varepsilon_{t+k|t} \tag{8}$$

where $f(t_{\mathrm{day}})$ is a spline function.

Make the base splines using $bs()$. Do we get better forecasts?

(Optional) How to choose the degrees of freedom `df`? maybe use AIC or BIC?

(Out of currect scope) Use Fourier series instead as basis functions for the diurnal curve.

Q4: Recursive least squares

Go and open "`q4_recursive_least_squares.R`". In the script the recursive least squares estimation is implemented in the function `rls()`.

Give it a look through and then try the part where data is generated from two periods – between which the parameter value are changed.

Does `lm()` estimate the parameters well on the combined data?

Does `rls()`?

Try to change the forgetting factor λ. What happens when it is set to 1 compared to the results obtained from `lm()`?

Is there a trade off between variance and bias (i.e. over- and under-fitting) related to λ?

Q5: Adaptive load forecast model with RLS

Now we will use RLS for fitting the coefficients, hence they can change over time. Go and open `q5_load_forecast_rls.R`, run first the first part:

- How about the forecasts, do they look fine?
- The tracked coefficients (the βs kept in θ), do they change?
- What was λ set to? it that optimal?

Run the next part plotting the first month of the training set:

- Are the forecasts good the first week?
- What about the coefficients?

Now, since the forecasts are poor until the coefficients are tracked, then make a "burn-in period", which simply means that a period in the beginning of the training set is left out in the score evaluation. Run next part and tune the parameters:

- Are the forecasts good the first week?
- What about the coefficients, did they change?

Run next part:

- Comment on the results: did the forecast improve? Does the coefficients change over time?

Final part is optional: Script for setting the forecasts for multiple horizons.

Q6: Solar forecasting

Now we can "easily" find a model which is useful for forecasting solar, e.g. the power generation on a PV panel. In the exercise, we will actually just use the observed global radiation as the solar power, hence this is somewhat a little bit simplified case. However, as presented in **?** these observations contain quite a few deviations: shadowing in the late morning hours from a chimney, some tilt of the sensor and also some saturation.

Open "`q6_solar_forecast.R`" and run firs part, where a very simple linear regression model is fitted:

- Do the forecasts seem to be good?
- Can you spot any systematic patterns?

Explore the residuals in the next part:

- Do you find any systematic patterns?
- What can cause the found differences in the relation between NWP global radiation G_{nwp} and the observed solar power (i.e. observed global radiation) P_{s}?

Now, define a model the relation between the solar power and the global radiation NWP is conditional on the time of day

$$P^{\text{s}}_{t+k|t} = f(t^{\text{day}}_{t+k}) G^{\text{nwp}}_{t+k|t} + \varepsilon_{t+k|t} \tag{9}$$

Fit it, using base splines and `lm()` and `rls()`, and finally with a kernel model.

- How can you decide which model is better?
- Do you find differences between the prediction performance of the models?

Q7: Wind forecasting

Go and open "`q7_wind_forecast.R`". Note: These are not real wind power measurements. The "wind power" P_w, which is used, is actually the observed wind speed put through a simple power curve function. Hence this is a "simplified" wind power, which have the basic properties of a "real" wind power – remember "real" wind power signals are potentially quite different depending on the wind turbine and wind farm properties, and surroundings etc. Hence, the signals used holds the basic properties of a wind power measurement.

Run first part and fit a linear model:

- Look at the scatter plot of the wind power P_w vs. wind speed NWP W_s (`plot(X$Ws, X$Pw)`) with the linear model fit (`abline(fit)`). Can you conclude that a linear model is suitable?

Run the next part:

- Are the forecast improving when you apply a base spline model taking into account a non-linear functional relationship between W_s and P_w?

Run next part: (`## Base spline model with rls`). So now we fit the base splines model with RLS:

- Do we achieve improvements with RLS over LM?
- If not, can you give some explanation? (i.e. somehow weird, since RLS with $\lambda = 1$ give the same result as LM, right!?)...what was the optimal λ found using the training set?

Finally, what about applying a locally weighted regression using a kernel? Do we achieve improvements?

So, this was a very "simplified" example, when forecasting "real" wind power of, say, a huge wind farm, then there are many things to take into account. First, surely a conditional dependence of wind direction is important, and further, a lot of information about the operational status is very important to take into account.

Q8: Probabilistic forecasting

Finally, an example of how a probabilistic forecasting model can be setup is presented. It is very simple: Replace linear regression `lm()` with quantile regression `rq()`, and replace the RMSE score function with CRPS!

Go and open "`q8_forecast_probabilistic.R`". Run both the linear and the base splines model:

- Go and find some information about the Continuous Ranked Probability Score (CRPS). When you got it, please fill out `https://en.wikipedia.org/wiki/Continuous_ranked_probability_score`!!
- Do we gain in predictive performance using the base spline model over the linear model?

Finally, an optional task is left to make a locally fitted quantile regression model (hence using a kernel). If this is of your interest, you basically have to replace `lm()` with `rq()`, and rmse() with `crpsDecomposition()$PRBS`...and yes, the same approach can be applied for load, solar, temperature, etc. forecasts!

Introduction

In this exercise we will illustrate the main concepts of Model Predictive Control (MPC) by using it to control the temperature of a building. As the name suggests MPC relies on models and predictions, and so, before designing a controller one needs to model the system and acquire relevant forecasts. In this exercise we will focus on the control part, assuming that we have perfect forecasts and models.

Questions

Question 1

Consider the model for a single room building developed during the previous exercises:

$$dT_i = \frac{1}{C_i}\left(\frac{1}{R_m}(T_m - T_i) + \frac{1}{R_a}(T_a - T_i) + \Phi + \eta * A_w G_v\right) dt + dw_i,$$
$$dT_m = \frac{1}{C_m}\left(\frac{1}{R_m}(T_i - T_m) + (1-\eta) A_w G_v\right) dt + dw_m,$$

where T_i and T_m are the temperatures of the indoor air and floor respectively, while T_a is the temperature of the air outside. C_i and C_m are heat capacities of the inside air and floor, and R_m and R_a are the thermal resistances between air-floor and air-outside. Φ is the effect of a radiator. G_v is global solar radiation with A_w being the effective window area. η describes the fraction of radiation going to the air and to the floor. w_i and w_m are both Wiener processes describing the stochasticity of the model. For this model it is assumed that the heating is through a radiator to the air of the building.

How would you change the model, if instead the building was heated by floor heating?

Question 2

Assume that you were to control the temperature of the air in this building. Would you expect it to be equally difficult for floor heating and air heating? If not, which one would be the most difficult and why?

Question 3

Open the script `r/ex_MPC.R`. The first part estimates a grey box model using CTSMR and data from a test facility at DTU (`http://www2.imm.dtu.dk/courses/02427/compEx3_E12.pdf`

Afterwards, the model is formulated in start state-space form, both in continuous time:

$$dX_t = AX_t + BU_t + dw_t,$$
$$Y_t = CX_t + DU_t + e_t,$$

and in discrete time (See appendix for how to do this):

$$X_{t+1} = A_d X_t + B_d U_t + \epsilon_{t+1},$$
$$Y_t = C_d X_t + D_d U_t + e_t,$$

where $X_t = \begin{bmatrix} T_t^i \\ T_t^m \end{bmatrix}$, namely the temperature of the inside air and the thermal mass. As is often the case $D_d = 0$ and $C_d = \begin{bmatrix} 1 & 0 \end{bmatrix}$ so that Y_t is the measured indoor air temperature. U_t is the amount of heating at time t.

Have a look at the values of the matrices A, B, A_d and B_d, and describe how they make sense.

Question 4

In this question we will formulate the equations of the MPC problem. The whole point of the control is to keep the temperature within comfort boundaries, so let us start with this. Assume that we do not care about the exact temperature of the building, as long as it is between 23 and 25 C°. Now look at the data concerning the heating of the building (data$Ph1). What is the maximum amount and minimum amount of heating? Finish the MPC formulation and explain each of the equations and inequalities in words:

$$\arg\min_{U_t,\, U_{t+1},\ldots,U_{t+N}} \sum_{k=0}^{N} U_{t+k}$$
$$s.t$$
$$X_{t+1} = A_d X_t + B_d U_t,$$
$$\leq C_d X_{t+1} \leq \quad ,$$
$$\leq U_t \leq \quad .$$

Question 5

Run the section `Perform Control` of the R-script, which simulates the controlled temperature using the same outside temperature and solar radiation as the data used for fitting the grey box model. The heating is optimized by solving the problem formulated in the previous question, with comfort boundaries and constraints on

the heating equipment specified initially. By default the noise of the simulation is turned off, so the controller is able to perfectly align the air temperature with the lower comfort boundary for maximum efficiency.

The control horizon is specified by `N` and is equal to 30 by default, meaning that the controller looks 30 time steps (300 min or 5 hours) ahead in time. What happens as you vary `N`?

Change the variable `Air` to `FALSE`, to simulate the building with floor heating instead of air heating. How does this affect the control?

Question 6

In reality unpredicted disturbances occur, which can be included in the model by increasing `Noise`. What happens when `Noise` is put equal to 0.5 or 1? What is the effect of the control horizon and heating medium (air or floor)? Since the noise is quite significant, especially for `Noise` equal to 1 you should try different combinations of the control parameters for a fixed seed, to see their effect, but also change the seed to see several realizations of the noise.

Question 7

In practice it is not possible to guarantee that the temperature stays within comfort boundaries, which is evident from the simulation results. However, it seems like the controller is doing a particularly bad job in this case, with the temperature going below the comfort limits many times. Why is this the case? How could we mend this issue?

Question 8

Sometimes one experiences varying prices or penalties, so that the objective is not to minimize energy consumption, but cost. This can easily be implemented by E-MPC (Economic Model Predictive Control). Mathmatically the problem changes to

$$\begin{aligned} \arg\min_{U_t,\, U_{t+1},\ldots,U_{t+N}} & \sum_{k=0}^{N} \lambda_{t+k} U_{t+k} \\ & s.t \\ X_{t+1} &= AX_t + BU_t, \\ &\leq CX_{t+1} \leq \ , \\ &\leq U_t \leq \ , \end{aligned}$$

where λ_t is the penalty or price at time t. Change the price in the R-script from a constant one to a varying one. This could for example be a PRBS signal which is

in the script as a comment. How does the optimization change when the prices are varying? What is the effect of the control horizon on the ability to minimize varying costs? What about the other parameters?

Question 9

In this example the process noise (ϵ_t) was estimated as part of the grey box model, and thus this can be used to estimate the distribution of the air temperature given the current temperature and the control action. If $\epsilon_t \sim N(0, \sigma^2)$, then e.g

$$(X_{t+1}|X_t, U_t) \sim N(AX_t + BU_t, \sigma^2),$$

and more generally

$$(X_{t+n}|\{X_s, U_s; t \leq s \leq t+n-1\}) \sim N(A^n X_t + \sum_{k=1}^{n} A^{k-1} B U_t, \sum_{k=1}^{n} A^{k-1} \Sigma).$$

We can use this to add an additional constraint that ensures that the probability of violating the comfort boundaries stays below some value, p:

$$\mathbb{P}(CX_t < T_{min}) < p$$
$$\Updownarrow$$
$$\int_{-\infty}^{T_{min}} \frac{1}{\sqrt{2\pi \sum_{k=1}^{n} A^{k-1}\sigma^2}} \mathrm{e}^{-\frac{\left(x - A^n X_t + \sum_{k=1}^{n} A^{k-1} B U_t\right)^2}{2\sum_{k=1}^{n} A^{k-1}\sigma^2}} dx < p.$$

Since the noise intensity (σ^2) is constant this formulation only depends on the difference between the expected temperature ($A^n X_t + \sum_{k=1}^{n} A^{k-1} B U_t$) and temperature limit (T_{min}). This means that we only have to compute the expression once, and the actual control implementation remains linear.

Change the variable `Stochastic` to `TRUE` and rerun the control simulation to see the effect. The variable `ViolationFraction` sets the value of p and can be used to adjust how often we are willing to accept comfort violations. Describe the difference compared to the previous case, where the stochasticity was not considered.

Questions and comments: `pbac@dtu.dk` or `rung@dtu.dk`

Appendix

The continuous time state space model

$$dX_t = AX_t + BU_t,$$
$$Y_t = CX_t + BU_t,$$

can be discretized with a time step of t_s to a model on the form

$$\begin{aligned} X_{t+1} &= A_d X_t + B_d U_t, \\ Y_t &= C_d X_t + D_d U_t, \end{aligned}$$

where

$$\begin{aligned} A_d &= e^{A t_s}, \\ B_d &= A^{-1}(A_d - I)B, \\ C_d &= C, \\ D_d &= D. \end{aligned}$$

References