Process Control Project

Document preparation and some tips

May 18, 2015

The intention of this document is to provide with some guidance to develop the project for the Process Control course (TKP4140), which is a 4^{th} year course in the Chemical Engineering program at NTNU (first year of the master program). The main description of the project and the problem statements are found in It's learning.

This document was first developed for the 2015 Fall semester. If you find any mistakes or have any comments, please let us know.

This project is relevant for the course because you will apply many of the concepts you have learned or will learn. It is also thought to be developed in teams. The workload is for a team of up to 3 students and every team member should participate equally. Start working on your project with time so that you have enough time in case you have any unforeseen problems.

If you have any further questions or specific questions regarding your project, do not hesitate to come during "drop-in hours" to K4, 2^{nd} floor. Outside these hours you can get help using the forum in It's learning.

1 General Tips

In this section you will find some general tips that might be useful for each part of the project. There are more tips in section 2.

1.1 Part 1

- You may use the example files as a starting point.
- When writing your nominal values in the code, try to not truncate the values. It is better to use the exact values or, when applicable, the equations. This can avoid some round-up errors. Be careful with the units.

- You may declare more symbolic variables, and then simplify your equations.
- Remember that the example files are for a 1×1 system and yours is not; so at some points you will have to use vectors instead of scalars [1].
- When simulating, you should do one step change at a time.
- After applying a step change, you should wait for steady state; except for integrating plants (e.g. level).
- Check that when you are not disturbing your process nor changing the set point, your outputs are actually zero when you simulate your process. This can help you to detect issues with your model or with your nominal values.

1.2 Part 2

- Remember that after you linearize you are working with deviation variables. This is important when setting up the Simulink block diagram; you should be careful when adding the nominal values.
- By this point you should already know how to use the half-rule approximation. However, this could be a good reference for some special cases:
  http://www.nt.ntnu.no/users/skoge/publications/2012/skogestad-improved-simc-pid/

  You can also find information about the SIMC rule in the same reference; you will use this tuning rule in part 3.

1.3 Part 3

- As for the previous parts, you can use the example files as starting point.
- For the Bode diagram:
  - Remember to change the unit for magnitude from dB to absolute, also in logarithmic scale. You can do this in the Property Editor of the Bode figure. You can open the Property Editor by double clicking on the diagram or by right clicking and then selecting "properties". You can also change the units directly from the code (see the documentation for *bodeplot*).
  - You can draw the asymptotes using Insert → line.
  - If you right click on the lines in the figure, a small window with the frequency and the magnitude (or phase) at that exact point will open. You can use this feature to read all the required points in the plot.

[1] Remember that in Matlab [2,2] is a horizontal vector and [2;2] is a vertical vector. You can transpose vectors with an apostrophe (’), so that [2;2]’=[2,2]

2 Using Matlab/Simulink

The project is done in Matlab. All NTNU students can install Matlab on their own computer; you should already have done it to solve some of the assignments. If you still do not have Matlab installed, refer to: https://innsida.ntnu.no/wiki/-/wiki/English/Matlab+for+students.

Remember to include Simulink and the Control Toolbox in the installation. If you have any problems with your computer, the computers in the computer room in K5 also have Matlab installed.

Previous knowledge is of course useful, but it is not essential; you can learn along the way. In this section you can find some tips for each part of the project. These might be useful especially for those who have not used Matlab previously. These tips are not *absolute*, and you might find better ways to solve your problem or to code.

If you require specific information about functions or would like to have code examples, it is **highly recommended** that you visit the Matlab Documentation website: http://se.mathworks.com/help/matlab/.

2.1 Simulink

- You can start Simulink by typing `simulink` in the command window in Matlab.
- Remember to set an appropriate simulation time. The default 10 seconds may not be appropriate for all systems. You can change this directly in Simulink or in your script: http://se.mathworks.com/help/simevents/ug/regulating-the-simulation-length.html
- Use the *To Workspace* block to send variables to the workspace for plotting, for example. The default save format is *Timeseries*. If you prefer to treat your variables as arrays in the workspace, change the format to *Array*; this way you can plot as usual.
- In "Sinks", you can also find the block *scope*, which can help you to see your results while simulating.
- You can combine several signals into one vector using *mux*, and then separate them using *demux*; both are blocks in Simulink. You find these blocks in "Signal routing".
- You can find the *S-function* under "user-defined functions".
- When you save your Simulink file, check whether you saved it as *.slx or *.mdl; it has to be consistent if you use the extension when running the model using `sim`('model') in the code.

2.2 Figures and plots

Probably the most important recommendation is to include axis titles and legends.

- In Matlab preferences, set the figure copy options to *transparent background*.
- You can format the figures directly from the code.
- You can save your figures directly from the code, using *saveas*.
- You can use *subplot* to plot together trends that are related.
- When defining the axis for your plots, it is good to have relevant axis limits. If you are comparing two comparable plots, use the same limits for both plots.
- In your report, remember to describe your figures with a caption.
- For the axis titles and legends you can use a caret (^) for superscript and underscore (_) for subscripts. For example: 'y_s' for y_s, 'm^3' for m^3. You can also use \ to display Greek letters: for example, '\tau' to display τ. See the documentation: http://se.mathworks.com/help/matlab/creating_plots/greek-letters-and-special-characters-in-graph-text.html

Matlab help for plotting:
http://se.mathworks.com/help/matlab/ref/plot.html

2.3 Matlab script

Not all the students have the same experience with Matlab, but everybody should be capable of doing the project. Especially if you are not very experienced, try to code as simple as possible and include all the comments you need [2], so that you can understand your own code later; remember that the project is cumulative.

If you have more experience with Matlab, you can use everything you know when coding.

2.3.1 Matlab functions

As mentioned previously, Matlab documentation is very complete; so, if you are not sure of how to use a function, you should visit the website.

An alternative option to get documentation is to type `doc name` + enter, in the command window. Then you will get the documentation for that functionality. For example, if you type: `doc minreal`, you will get the documentation for `minreal`.

Other ways to get documentation are explained here:
http://se.mathworks.com/help/matlab/matlab_env/help-resources.html

Some functions that you might need or want to use are listed below. The list is not exhaustive.

[2] Remember that everything written after a percentage sign (%) is a comment.

- *saveas*: saves figures in your preferred format.
- *sim*: simulate dynamic (Simulink) system.
- *syms*: creates symbolic variables that you can use for your equations.
- *subs*: symbolic substitution, replacing the symbolic variables with their values.
- *jacobian*: calculates the Jacobian of a matrix.
- *inv*: inverse of a matrix
- *minreal*: removes unobservable states for transfer functions.
- *set*: in general, it sets properties; for example, you can set plot properties.

  Another use of `set` is for transfer functions. By default, transfer functions are not displayed in a *time constant* format. To change this, you need to specify the display format. For example, if you have a transfer function called g_{11}:

  ```
  g11 = set(g11,'DisplayFormat','time constant');
  ```

  This feature can help you to get the transfer function in the format you are used to, so that you see the time constants easily. However, remember that for the exam you should be able to do it by hand.

3 Document preparation

Remember that the second and third reports are cumulative. This means that the second report should contain what was done in the first part and that the third delivery is a final report covering the whole project.

Your report should be "readable", meaning that there should be a sequence and it should be simple to follow and to understand what you did.

Simulink diagrams should be included in the report or in the appendix, where you find it better. Please include the code that you wrote or modified as an appendix; you can also include code snippets in the main report.

Even if you do not include a front page, please include the names of every team member in the first page of the document.

3.1 LaTex or Word

You may use either LaTex or Word. In any case, please upload only the *.pdf file.

LaTex is a free document preparation software that is widely used in academia. It is convenient when typing mathematical formulas or including code in your document (`listings` package).

If you choose to use Word, you may use an equation editor such as Math Type to type your equations. You can get it from *progdist* (Innsida).

3.2 Other useful software

If you need to draw some additional diagrams or generate some images you may use:

- Dia Diagram Editor: open source software to draw diagrams (such as PFDs).
- Microsoft Visio: Microsoft software to draw diagrams (such as PFDs). It is included in the Microsoft Office package in Sofware Farm.
- Ipe: open source drawing editor to create vector graphics that can also to be included in LaTex documents.
- Inkscape: open source software to create vector graphics.

These are just some examples. You may use any software that you find most convenient for your purposes, if you require any.

3.3 Report contents

Each report should cover what is required in the description. This is just a summary.

The first report should include:

- **Introduction**: description of the process and statement of the control objectives.
- **Derivation** of the nonlinear model and all the assumptions that you made.
- **Simulation results**: step change (10% increase) all inputs and disturbances, as in the example.
- **S-function** script (.m file), as an appendix; so we can check your equations.
- **Discussion**: are your simulation results reasonable? Why?

The second report should include:

- **First report.** If there were corrections, these should be included.
- **Linearization**
- **Laplace transform** of linearized models and transfer function matrix for u's and d's.

- **Comparison** of original non-linear models with linearized models (plots).
- **Zero, pole, gain** of transfer functions.
- **Half-rule** approximation of transfer functions to obtain a first-order with time delay of the transfer functions relating MVs to the CVs.
- **Discussion** of the zeros, poles, gain. Is it consistent with the simulations?

The third and final report should include:

- **Part 1 and 2.** If there were corrections, these should be included.
- **Pairing**: selection of the best control structure (pairing), using your engineering insight. Include a block diagram.
  - For single MVs and single CVs projects, instead of pairing, you should implement either a cascade controller or a feedforward (from the disturbances) controller.
- **Tuning** of PI controllers using the SIMC rules.
- **Bode plot**:
  - Choose one CV and the corresponding controller.
  - Do the bode plot for that loop: $L(s) = C_1(s) * g_{11}(s)$
  - Compute phase and gain margin.
  - Identify poles and zeros in the plot.
  - Draw the asymptotes.
- **Closed loop behavior**
  - Add the controller(s) to your Simulink simulation.
  - Apply step changes of of 10% from nominal for set points.
  - Apply step changes of of 10% from nominal for disturbances.
- **Performance**: calculate IAE (integral absolute error) to evaluate the controllers
- **Discussion**: discuss the pairing, quality of control and performance.
- **Conclusion of the project**: overall conclusion of the project, summarizing what has been done.
  - The conclusion may include a feedback about the project itself - how to improve it for the next year.

4 Uploading your report

Only one person per team should upload the files to It's Learning. Remember to add every team member in It's Learning. Please upload:

- Report (*.pdf)
  Only the *.pdf file is required.

- Matlab files (zipped)
  This is not mandatory but could be helpful when reviewing your project, specially if there are any mistakes or areas of improvement in parts 1 and 2.

5 Final Comments

Part 1 and part 2 will not receive a grade and will be either "approved" or "not approved". If your report is approved you might receive some comments that should be included/corrected in the next delivery. If it is not approved, you should correct your report and upload it again to have it reviewed again and continue with your project.

The final report (third delivery) is graded and will count for your final grade, as stated in the course description.