Cambridge University Engineering Department

DESIGN PROJECT GF1: CONTROL SYSTEM

Week 2: Elaborating the model. Closing the $L2$ control loop.

1 Finish testing the process

The first essential activity this week is to finish the testing and debugging of the steady-state behaviour of the process model, if you did not finish it last week.

2 Lags on control inputs

In practice the control inputs $F2$, $P100$ and $F200$ are controlled by local servo-controllers (which adjust flow valves), and cannot be changed instantaneously. We shall model each of these controllers as a first-order lag with time constant 1.2 minutes. The input to each lag will be the set-point for the control input, and its output will be the actual value of that input.

Modify your process model to include these lags. You could use the `Transfer Fcn` block from the `Continuous` library, but it will be better to use a slightly more complicated construction. A simple first-order lag with transfer function $1/(1+sT)$ can be realised as an integrator with gain $1/T$, and a negative feedback around it containing gain 1. (*Check this.*) The advantage of doing it this way is that you can set the initial condition on an integrator in *Simulink*, and this will be very helpful — and save a lot of time — when running simulations and obtaining linearisations. An alternative would be to realise the lag as a state-space system — you can work out the required state-space matrices yourself; use $C = 1$ so that the state is the same as the output. Since you will need lags in several places it may be best to build one as a block, and re-use it later. From now on it will be assumed that you test changes to the model as you make them, so you will need to devise appropriate tests. (Keep them simple — for instance, attach a scope to $Q100$ or another suitable variable, and run the simulation.)

3 Controlling the separator level

Now comes the first bit of controller design. A controller is needed for the separator level $L2$, and the simplest controller is just a proportional one, which manipulates the 'Product Flowrate' $F2$. (In fact it manipulates the set-point for $F2$, as discussed above.) A Proportional and Integral (PI) controller could also be tried, but appears to be unnecessarily complicated, since there is already an integration in the separator dynamics. Later we shall see that in fact there is a need for integral action, but we begin by designing a proportional controller.

The set-point for $L2$ will be 1 m. Note that the proportional gain will in fact have to be negative, because increasing $F2$ reduces $L2$ since $F2$ is an outflow. (In other words, the separator has a negative gain between $F2$ and $L2$.)

The first design of the $L2$ controller will be based on a completely heuristic method. Add a proportional controller for the $L2 - F2$ loop, using a `Slider Gain` from the `Math` library to get an adjustable-gain controller. Increase the proportional gain until the response of $L2$ resembles that of a second-order system with damping factor 0.2 when the $L2$ set-point is changed from 1.0 to 1.4 m. (Note that the step response of a second-order system with damping factor of 0.2 has an overshoot of about 50%.) Of course you have to perform this exercise with the whole process at the operating point defined in Table 1 of the Week 1 lab sheet.

Investigate the performance of your proportional controller in response to ±10% set-point changes (namely when the separator level set-point is to be increased to 1.1 m or reduced to 0.9 m), and to ±10% changes in the Feed flow rate $F1$.

Now repeat the design, using a more theoretical approach:

1. Obtain a linearised model.
2. Compute the frequency response from $F2$ to $L2$.
3. Calculate the proportional gain which gives a phase margin of 45 degrees, and implement it.

For Step 1 use the function `linmod`, which you used last week for linearising the evaporator. You must first remove the existing proportional controller (which is easily done by setting its gain to zero). To use `linmod` you need the version of the model with Input and Output Ports (`In` and `Out`) attached. Furthermore they must be numbered consecutively; if there are numbers missing you will have to change some of the port numbers (the `Port number` parameters in the `In` and `Out` blocks). Alternatively, you can just attach a single `In` block to the $F2$ variable, as shown in figure 1 — note that this figure shows the F2 controller lag as a `Transfer Function` block, which is not the recommended way of implementing it. The resulting state-space model will be multivariable, and you will have to see, from the port numbers, which input and output corresponds to the $F2$ set-point and to $L2$, respectively. You can get the state-space model corresponding to just this input-output pair by retaining the appropriate column of the 'B' matrix, the appropriate row of the 'C' matrix, and the appropriate element of the 'D' matrix.

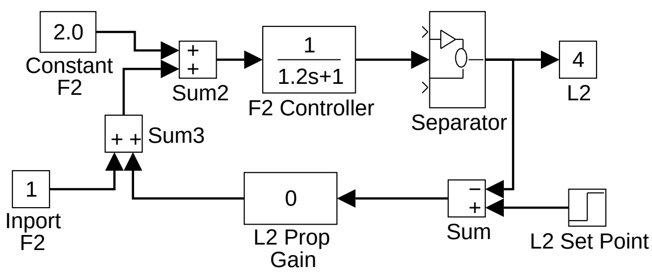

Figure 1: Part of the system ready for linearisation with input $F2$.

For Steps 2 and 3 you can use the MATLAB functions `bode`, `nyquist`, or `ltiview`. First assemble the state-space matrices of your linearised model into a 'Linear Time-Invariant' or `lti` system, using something like `sys1=ss(A,B,C,D);` (in the MATLAB command window). Then type `bode(sys1)`, or `ltiview(sys1)` etc. Type `help bode` etc to find out more about these functions. You may also find the function `margin` useful. Note that the Bode and/or Nyquist plots will look unfamiliar at first, because of the negative gain mentioned above. They will look more familiar if you change the sign, for example by changing the sign of the 'B' matrix.

Compare the performance of the controller designed in this way with that obtained by heuristic tuning. It is also interesting to see what phase margin results when you use the heuristically-obtained gain with the linearised model.

You should have noticed that when $F1$ changes to a new steady value, $L2$ does not return to the correct level, despite the fact that the Separator acts like an integrator. Explain why this should be so.

A PI controller is needed to force $L2$ back to the correct steady-state level. Design a PI controller for the $L2$ loop as follows. Keep the proportional gain which you have designed, and find the integral action time (namely T_i in the PI transfer function $K_p[1 + 1/(sT_i)]$) which gives a phase margin of 40 degrees. Implement the PI controller with these coefficients. (Note that the PI controller transfer function can be rewritten as $K_p(T_i s + 1)/sT_i$. This is more convenient for analysis, and can also be used for implementation, although *Simulink* has a 'PID' block in its `Blocksets_and_Toolboxes/Simulink_Extras/Additional_Linear` library.) An easy way of doing this is to find the frequency at which you previously calculated the 45 degree phase margin with the proportional controller, and choose T_i so that the PI controller has a phase lag of 5 degrees at this frequency. (Check that this procedure keeps the 0dB cross-over frequency nearly unchanged, and gives a phase margin of about 40 degrees. See [4, 5] for more details and/or alternative ways of tackling the design.)

Note that the functions `ss`, `nyquist`, `bode`, `ltiview` *and* `margin` *are in the Control Systems Toolbox of Matlab.*

4 Constraints on variables

The next step is to make your model more realistic by adding constraints on variables. Assume that the maximum value for each flow rate is double its nominal value, and the minimum is zero (so $0 \leq F1 \leq 20$, for example), that $0 \leq L2 \leq 2$, that $0 \leq P100 \leq 400$, and that $0 \leq P2 \leq 100$.

Using the `Saturation` block from either the `Discontinuities` or the `Commonly Used Blocks` library, implement these constraints on variables $F2, F4, F5, F200, L2, P2, P100$. To keep the complexity of the block diagram manageable, it is probably best to put the saturation elements inside the relevant subsystems. So the saturation of $F2$ should be implemented within the `F2 Controller` block, the saturation of $L2$ should be implemented within the `Separator` block, and so on. Be careful to implement the constraints on $F4$ in the right place, namely so that it affects $F4$ within the `Evaporator` block, and not just the signal path which appears outside the block.

With the constraints in place, investigate whether the $L2$ controller still works satisfactorily. What are the largest step disturbances on $F1$ and on $X1$, respectively, which the controller can cope with?

Note that a big perturbation can induce nonlinear oscillations, and that these may persist even after the perturbation ceases.

5 Second interim report

At the end of this week you must produce the second interim report.

References

[1] Newell, R. B, and Lee, P. L, *Applied Process Control, A Case Study*, New York: Prentice-Hall, 1989. (CUED Shelfmark: QL 30)

[2] *Matlab User's Guide.*

[3] *Simulink User's Guide.*

[4] Dorf, R. C, *Modern Control Systems, 9th edition*, Reading MA: Addison-Wesley, 2001. (CUED Shelfmark: QC 250)

[5] Franklin, G. F., Powell, J. D, and Emami-Naeini, A, *Feedback Control of Dynamic Systems, 4th edition*, Reading MA: Prentice-Hall, 2002. (CUED Shelfmark: QC 253)

R. Sepulchre May 2022