Cambridge University Engineering Department

DESIGN PROJECT GF1: CONTROL SYSTEM

Week 1: Building a model in Simulink

1 An Evaporator Process

An evaporator is an industrial process used to evaporate solvent from a feed-stream — for example, in paper manufacture or sugar production. Figure 1 shows a diagram of a typical evaporator.

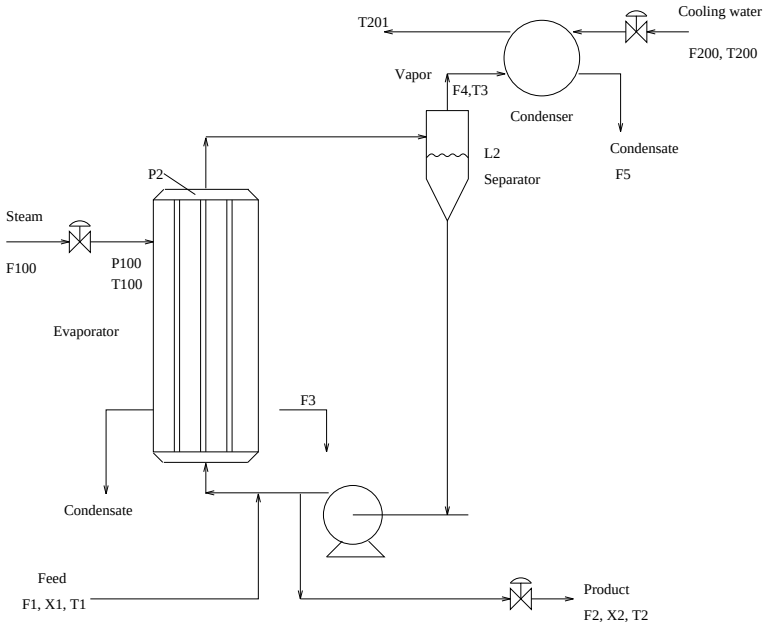

Figure 1: A typical evaporator

A liquid 'Feed' comes in at the bottom left-hand corner and is mixed with recirculating liquor. It is pumped into the evaporator itself, which is a heat exchanger, in which heat is exchanged with steam (which comes in at the top left-hand corner of the evaporator, and its condensate leaves at the bottom left). The recirculating liquor boils in the evaporator, and a two-phase mixture of liquid and vapour flows to the Separator, where the liquid and vapour are separated from each other. The liquid (now more concentrated than when it entered the evaporator) is pumped round again, and some of it is drawn off as 'Product' (bottom right of the figure).

The vapour flows to a Condenser, which is another heat exchanger, this time heat being exchanged with cooling water. The condensate, which may itself be a useful product in some cases, leaves the process at the top right of figure 1.

In this project we shall model and control the evaporator described in [1], on which figure 1 is based. The Library has 4 copies of [1] (shelfmark QL.30).

The main variable which needs to be controlled is the 'Product Composition', which we will call $X2$ (following the notation used in [1]). Keeping variations in the composition as small as possible maximises the profitability of the evaporator, because it minimises production of out-of-spec product (which cannot be sold, or has to be sold at a low price), while minimising production costs, since it is possible to aim for a composition only a little better than the minimum acceptable one. However, it is also necessary to operate the evaporator safely, and without damaging the installed equipment. This requires the pressure in the evaporator ($P2$), and the level of liquid in the separator ($L2$), to be controlled. (If the separator overflows the condenser will be damaged; if it runs dry the pump will be damaged.)

There are three variables which we assume we can adjust, in order to control the evaporator: the mass flow rate of the product being drawn off from the recirculating liquor ($F2$), the pressure of the steam entering the evaporator ($P100$), and the mass flow rate of the cooling water entering the condenser ($F200$). These three variables will be our 'control inputs'.

There are several other variables which affect the evaporator's performance, but we assume that these are not under our control. These will act as 'disturbances' which will keep pushing the controlled variables away from their target values. In particular, the Feed flow rate ($F1$), the circulating flow rate ($F3$), the Feed composition ($X1$), the Feed temperature ($T1$), and the cooling water inlet temperature ($T200$) all give the process significant disturbances.

2 Familiarisation with MATLAB

You should have some familiarity with MATLAB before going further. If you have not used MATLAB before, work through the handout *Getting Started With Matlab* or *Matlab by Example* — this takes about 1 hour. In-house MATLAB resources can be found at

`http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html`.

If you know what a MATLAB 'function' is, and how to create a simple one, then you know enough about MATLAB to go on. Some familiarity with basic graphical plotting commands will be useful later in the project.

If you want more information about MATLAB, there are several other documents produced at CUED (available on-line or hard copy), the on-line '`help`' facility in MATLAB is very extensive. The '`doc`' and '`helpdesk`' facilities give access to the full documentation in the form of web pages.

3 Familiarisaton with *Simulink*

Simulink is interactive software for simulating systems, which is run from MATLAB. To run it you can enter MATLAB and type `simulink`, which opens a *Simulink* window. The best way to get started is to work through the 'Getting Started' chapter of the *Simulink* documentation, especially the section 'Working with a Simple Model'. Spend about 1 hour going through this example, and exploring *Simulink* a little on your own. Make sure you know how to save a model to file, and how to get a printout of a block diagram.

During your exploration of *Simulink*, try running some of the other *Simulink* demonstrations available from the `demos` window. Some of these show clever things you can do with *Simulink*. We shall not attempt to be as ambitious in this project.

When you have done this you will know only a little about *Simulink*. You will get very detailed guidance about how to use it in the following instructions, but the detail will gradually disappear as you become more competent. You will probably need to consult the documentation from time to time.

Important Note: New versions of MATLAB and *Simulink* are installed on the DPO system from time to time. This handout has been modified to be consistent with recent versions, but it is possible that some remaining inconsistency has been missed. If you find one, please report it by email to `rs771@cam.ac.uk`. The figures were generated using an early version, so may not look exactly the same as the ones you will generate. In particular, Figs. 5–7 show the time axis labelled as 'seconds' (because it could not be changed in early versions) but in the model time is measured in minutes.

4 A *Simulink* model of the process

4.1 The Separator

The Separator Level ($L2$) is determined by equation (2.1) in [1]:

$$\rho A \frac{dL2}{dt} = F1 - F2 - F4 \tag{1}$$

where ρ is the liquid density and A is the cross-sectional area of the separator. To model this in *Simulink* first write this as

$$\frac{dL2}{dt} = \frac{1}{\rho A}(F1 - F2 - F4). \tag{2}$$

Open a new *Simulink* window (use `New...` on the `File` Menu), then `Model`. Then open the `Continuous` block in the *Simulink* Block Library, and from it drag an `Integrator` onto the new window. Next open the `Math Operations` block, and drag a `Sum` and a `Gain` from it onto the new window. Then open the `Sources` block, and drag a `Signal Generator`, a `Sine Wave`, and a `Step` to the new window. Finally open the `Sinks` block, and drag a `Scope` to the new window. (You can close or iconify the `Continuous`, `Math`, `Sources` and `Sinks` blocks if you want to reduce the clutter on the screen.) Now connect all the elements you have in the new window, so that they look like figure 2.

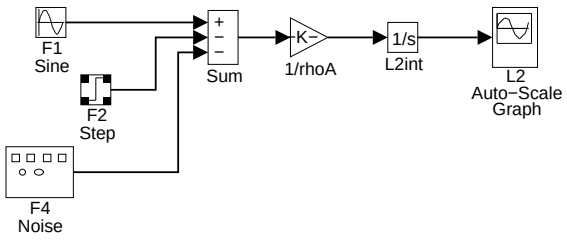

Figure 2: Separator ready for testing

To change the name of an object move the pointer to the name and click on the left mouse button; this should highlight the name, which can then be edited. The `Sum` block can be changed to

have the right number of inputs, and the right signs, by double-clicking on it and editing the `List of Signs` field. We shall assume initially that $\rho A = 20$ kg/m, so change the gain of the `Gain` block to `1/20` by double-clicking on the block and editing the `Gain` field.

When the diagram looks right, pull down the `Simulation` menu and select `Configuration Parameters`. Change the `Stop Time` to `100`. (Note that this will be in *minutes*, since the units of time used throughout the model definition are minutes rather than seconds.) Then start the simulation, using the default settings of the input signal blocks. Check that the output ($L2$) looks reasonable. The axis ranges on the `Scope` may not be suitable, in which case you need to 'open' the `Scope` and reset the ranges.

The point of doing this is just to check that the model assembled so far is defined properly. As with hardware or conventional programs, testing small modules is more effective than assembling a large model and then trying to find out why it doesn't work. Once you are happy that the Separator model is correct, replace the input and output devices by `In` and `Out` ports, respectively, (you will find these in the `Ports & Subsystems` block), so that the model looks like figure 3 — except that the ports now have oval icons, not square ones.

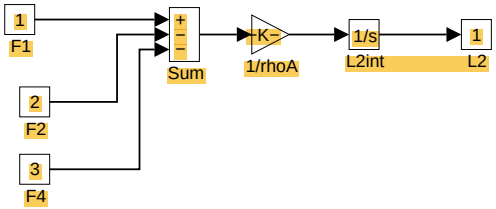

Figure 3: The Separator Module

Save this model to file, with the name `separator` (use `Save As...` in the `File` menu).

4.2 The Evaporator

The evaporator itself is modeled by 5 equations ((2.2)–(2.6) in [1]):

$$M\frac{dX2}{dt} = F1 \times X1 - F2 \times X2 \quad (3)$$

$$C\frac{dP2}{dt} = F4 - F5 \quad (4)$$

$$T2 = 0.5616P2 + 0.3126X2 + 48.43 \quad (5)$$

$$T3 = 0.507P2 + 55.0 \quad (6)$$

$$F4 = \frac{Q100 - F1 \times C_p(T2 - T1)}{\lambda} \quad (7)$$

where $F4$ is the vapour flowrate, $F5$ is the condensate flowrate, $T2$ and $T3$ are the product and the vapour temperatures, respectively, $Q100$ is the heater duty, and the coefficients have the following values:

M	20	kg
C	4	kg/kPa
C_p	0.07	kW/°K(kg/min)
λ	38.5	kW/(kg/min)

Open a new *Simulink* window and build a model of these 5 equations in it. Figure 4 shows what the finished model may look like. *Don't just copy figure 4, but work it out yourself from the*

equations. It is a good idea to sketch a solution out on paper before starting to use *Simulink*. Your solution need not look the same as the figure. The layout of course does not matter, but you may also have more significant differences.

One new feature of the evaporator is that there are products of signals in equations (3) and (7). You can implement these either by using the `Product` or the `Dot Product` blocks (which are in the `Math` block) — for scalar signals these are equivalent.

Note that the signal $T2$ will be needed as an output to the Steam Jacket model, $T3$ will be needed as an output to the Condenser, and $F4$ will be needed as an output to the Separator. Also note that, although $X2$ and $P2$ are not needed as inputs to other parts of the process, they are crucial outputs of the whole process, and will in fact be among the most important variables being controlled. So it is crucial to bring them out as outputs of the evaporator.

Since the evaporator model is quite complicated, save it to file from time to time as you build it (with name `evaporator`), so that you don't lose everything if the system crashes. It will probably take you about 2-3 hours to build the model, possibly more to test and debug it.

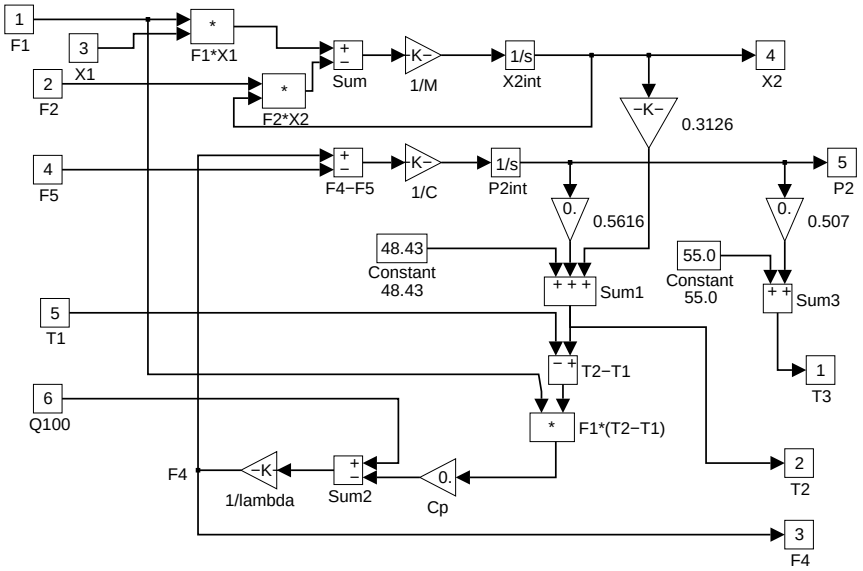

Figure 4: The Evaporator Model

Test your model by replacing each input port by a `Step Input`, and each output port by a `Scope`, and running a simulation. (In fact, since the same step function can be applied to each input, you can just use one `Step Input` and connect it to all 6 input signals.) Now, even if the simulation runs, you are faced with the considerable problem of deciding whether it is correct, and how long you should run the simulation for. In this case the model is simple enough for some analysis to be possible. If you examine the model, you will see that it contains two feedback loops, each one containing an integrator. Tracing the signals round each loop shows that they are both negative feedback loops, so the model should be stable. (For first-order loops this is a valid deduction.) How fast should the model respond? Although the model is nonlinear, if $F1$ and $F2$ are constant it will behave linearly, because the `Product` blocks then behave like constant gains. Suppose you apply unit steps at all the inputs, so that $F1 = F2 = 1$ (for $t > 0$). Then tracing the signal round the $P2$ loop shows that this loop contains a total gain of

$$\frac{0.5616 \times F1 \times C_p}{\lambda \times C} \quad = \quad 2.5527 \times 10^{-4}. \tag{8}$$

Work out what is the time constant of a negative feedback loop consisting of an integrator and

this gain. This will give you an idea of how long to run the simulation for. (The $X2$ loop is much faster, as you can check.)

This is a good opportunity to try the effect of changing the `Configuration Parameters` which appear on the `Simulation` menu. While the simulation is running, use `Pause` on the `Simulation` menu to suspend it, then select `Configuration Parameters`, and see what effect changing the simulation method has on speed. (Use `Continue` to resume the simulation.) All the methods should be equally accurate for this model, but some may be considerably slower than others. Also try the effects of changing the `Min step`, `Max step` `Relative tolerance` and `Absolute tolerance` parameters.

Figures 5, 6 and 7 show the responses I got.

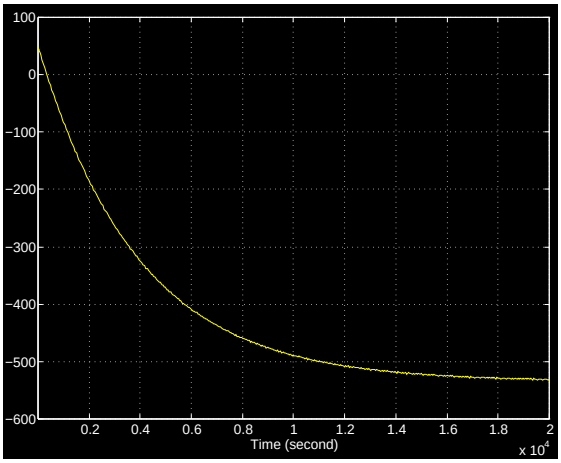

Figure 5: Response of $T2$

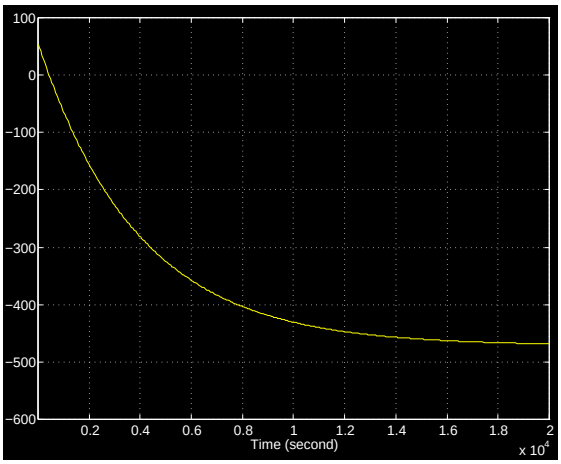

Figure 6: Response of $T3$

One more check you can do is to make the step on $F1$ of amplitude 10 rather than 1. According to the analysis above, this should make the $P2$ loop 10 times faster. Check whether this is the case in your model.

Linearising the evaporator

A facility provided within *Simulink* is linearisation of nonlinear models. We will use this later for designing controllers, but you can practice it now. It will also give a further check of the

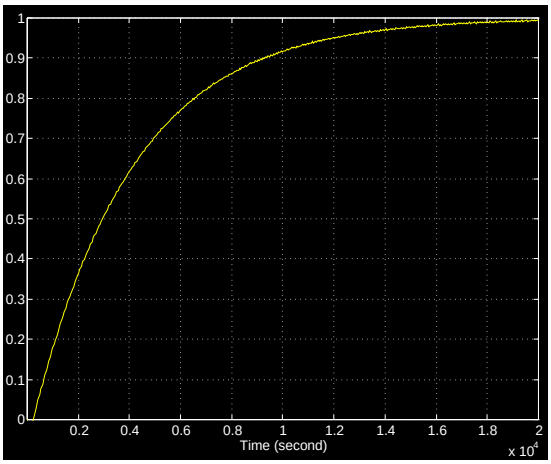

Figure 7: Response of $F4$

model. First you must put the input and output ports back on to all the inputs and outputs. (The easiest way to do this is to read your previously-saved `evaporator` system from file.) Now you call the linearisation function `linmod` *from the Matlab window*. You have to set the values of the states and inputs you want for the linearisation. In our case there will be 2 states (since there are 2 integrators and no other dynamics), and 6 inputs. We will linearise the evaporator about the point $P2 = 50.5$, $X2 = 25.0$, $F1 = F2 = X1 = F5 = T1 = Q100 = 1$. In order to see how *Simulink* has ordered the states (namely the integrators) type

```
[sizes,x0,xnames] = evaporator([],[],[],0)
```

On my model this shows that $P2int$ is the first integrator, and $X2int$ the second one. Consequently the appropriate linearisation command is:

```
[evapA,evapB,evapC,evapD]=linmod('evaporator',[50.5,25],[1,1,1,1,1,1])
```

This gives you the 'A, B, C, D' matrices of a linear state-space model which is the linearised evaporator. Check the eigenvalues of the 'A' matrix:

```
eigA = eig(evapA)
```

These should agree with the poles of the two feedback loops in the model, as estimated earlier.

For more details of the linearisation function `linmod` see the on-line documentation.

Can you get the linearised model for $F1 = 10$? Do the eigenvalues change as you expect?

4.3 The Heater Steam Jacket

The Heater Steam Jacket is described by equations (2.7)-(2.9) in [1]:

$$T100 = 0.1538 \times P100 + 90.0 \quad (9)$$

$$Q100 = 0.16(F1 + F3)(T100 - T2) \quad (10)$$

$$F100 = \frac{Q100}{\lambda_s} \quad (11)$$

where $T100$ is the steam temperature, $F100$ is the steam flowrate, and λ_s is a coefficient with value $\lambda_s = 36.6$ kW/(kg/min).

Note that $Q100$ is needed as an output to the Evaporator.

Build a *Simulink* model of the Heater Steam Jacket and save it to file with name `steamjacket`. Make sure that you test it.

4.4 The Condenser

The Condenser is described by equations (2.10)-(2.12) in [1]:

$$Q200 = \frac{UA2(T3 - T200)}{1 + \frac{UA2}{2C_p F200}} \quad (12)$$

$$T201 = T200 + \frac{Q200}{F200 \times C_P} \quad (13)$$

$$F5 = \frac{Q200}{\lambda} \quad (14)$$

where $Q200$ is the condenser duty, $T201$ is the cooling water outlet temperature, and $UA2$ is a coefficient with value $UA2 = 6.84$ kW/°K. The other coefficients have the same values as above.

Note that $F5$ is needed as an output to the Evaporator.

Build and test a *Simulink* model of the Condenser, and save it to file with name `condenser`. For this model you need to divide as well as multiply. A 'divider' block is obtained from a `Product` block by editing the parameters (replace the default value '2' by '/').

4.5 The Whole Process

Now it is time to bring all the sub-models together and build the model of the whole process. To do this open a new *Simulink* window. Then open the system `evaporator`, and drag it onto the new window. To keep the model tidy, convert the evaporator model into a subsystem. This is done as follows: put a 'rubber band' around the blocks to be grouped, using the left mouse button, then choose `Create subsystem` from the `Edit` menu. The lines connecting the `In` and `Out` ports to the evaporator subsystem may be horribly tangled at this stage, but you can simplify their routes by using the mouse buttons. Change the name from `Subsystem` to `Evaporator`. Save the result to file, with name `process`.

Next, repeat this process with each of the `separator`, `condenser`, and `steamjacket` models. Note that the name of each block in the window must be unique, so *Simulink* will change some of the `In` and `Out` port names, without warning you. If it finds a duplicate block name which ends with an integer, it will simply increase the integer until a unique name is obtained. This means that you may find names like `T9` suddenly appearing. If you are relying on the port names to keep track of which signal is which, you need to restore meaningful names straight away. This is done most easily *before* grouping a model into a subsystem. Proceed as follows, using `separator` as an example. When you drag `separator` into the *process* window, its `In` port labelled `F4` becomes changed to `F5`, because there is already an `Out` port on `evaporator` labelled `F4`. Change the label from `F5` to a name which is meaningful in the model, but still unique, such as `F4_`. If the names of several ports have been changed, you have to be careful to identify which is which correctly. This is best done by comparison with the submodel in its

original window. When all the ports have been given meaningful names, convert the `separator` model into a 'Subsystem'.

When you have done this for all the submodels, you should end up with a model which looks something like figure 8. At this stage the subsystems are not yet connected to each other. Finally, make the relevant connections between subsystems. This is where the use of meaningful

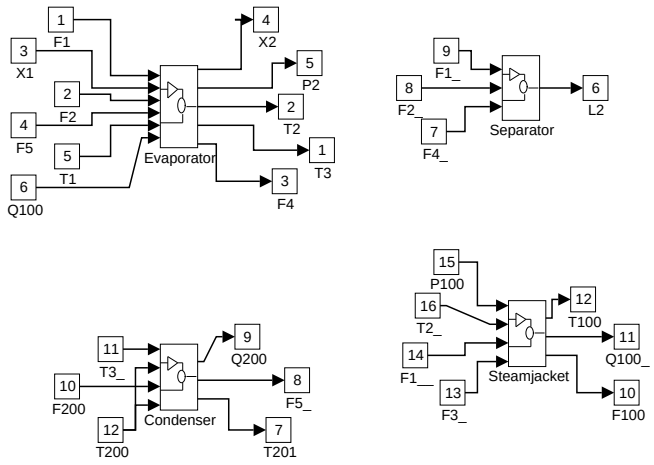

Figure 8: The process model before connecting subsystems

names for `In` and `Out` ports becomes very useful — if you have `F4` and `F4_` you know you should connect them together. The final model should look something like figure 9.

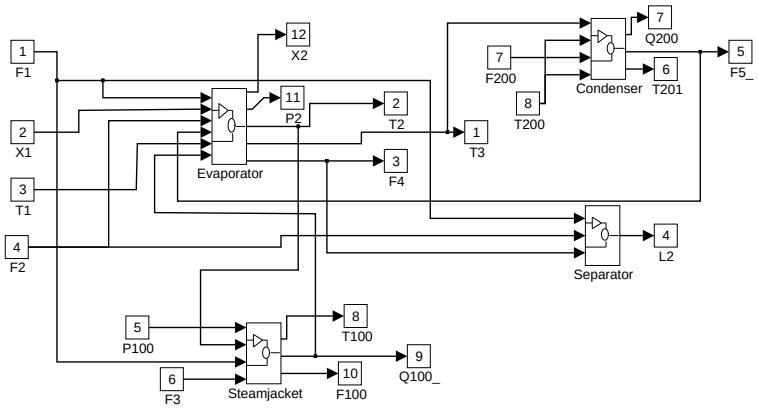

Figure 9: The process model with subsystems connected

Now of course you have to test the whole model.

4.6 Testing and Debugging

Table 1 lists all the variables in the complete process model, and their nominal steady-state values. (This is taken from Table 2.1 in [1].) The simplest test on the complete model is to check whether all the variables take on the correct steady-state values when the inputs are given the nominal steady-state values. To do this replace the `In` ports in the model by either `Constant` blocks or `Step Input` blocks, with the appropriate amplitudes. (You are advised to save the resulting system with a new name, such as `Process_ss`, since you are likely to need the system with `In` ports again.) Figure 10 shows the appearance of the model at this stage. Now

run a simulation and see whether each of the output variables reaches the correct value after transients have finished. Note that the dynamics will be much faster than in the tests you did earlier, because the 'operating point' of the process is very different — flow rates are higher, temperatures are higher, etc. There is no need to replace each `Out` port by a `Scope` or other device; it is enough to have a 'floating scope' and attach it to each output variable in turn.

Note that the value of 1.0 m given for the separator level $L2$ is misleading. Since this level is the result of an open-loop integration, it will settle down to an arbitrary level, depending on the initial conditions of all the integrators in the process, and on the dynamics of the transient. In fact, it will only settle to a constant level providing that the net flow rate into the separator is zero, namely if $F1 - F2 - F4 = 0$.

If all the variables converge to the correct values, congratulations! Otherwise, be prepared for a possibly lengthy debugging process. Most errors will be quite easy to track down, and will be due to things like incorrect `Gain` parameters and incorrect connections between blocks. (If you have tested the subsystems properly there should not be any errors within blocks at this stage.)

An alternative way of checking the steady-state values in the model is to use the function `trim` from the MATLAB command window. Type `help trim` in the MATLAB command window to see how this works, or read the appropriate sections of the *Simulink* documentation. Use `trim` to check the steady-state values in your model.

5 First Interim Report

At the end of the first week you must write a brief Interim Report recording what you have done so far. Further information on what this should contain will be given at the first session.

The report should be at most *2 sides* of A4 paper in length, excluding appendices. Appendices may be used for important supporting information such as plots of numerical data, computer listings, etc. Students must adhere to the page limit and keep the volume of appendices to a minimum.

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] Maciejowski, J. M., *Getting Started With Matlab*, DPO Handout (online).

[3] *Matlab User's Guide* (online).

[4] *Simulink User's Guide* (online).

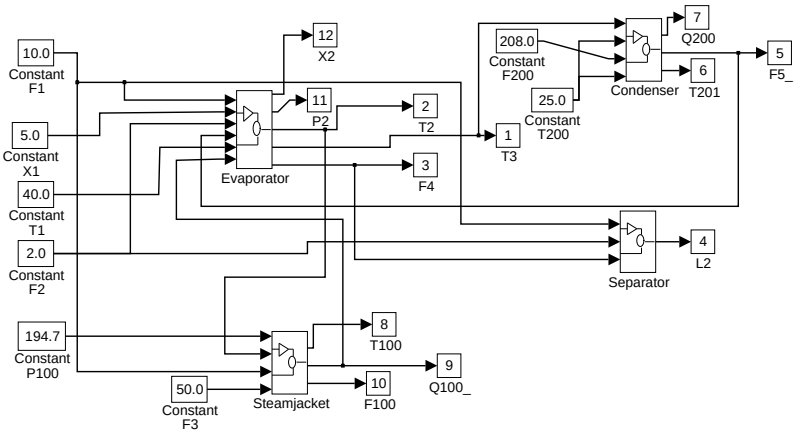

Figure 10: The process model with constant inputs

Variable	Description	Value	Units
F1	Feed flowrate	10.0	kg/min
F2	Product flowrate	2.0	kg/min
F3	Circulating flowrate	50.0	kg/min
F4	Vapour flowrate	8.0	kg/min
F5	Condensate flowrate	8.0	kg/min
X1	Feed composition	5.0	%
X2	Product composition	25.0	%
T1	Feed temperature	40.0	°C
T2	Product temperature	84.6	°C
T3	Vapour temperature	80.6	°C
L2	Separator level	1.0	m
P2	Operating pressure	50.5	kPa
F100	Steam flowrate	9.3	kg/min
T100	Steam temperature	119.9	°C
P100	Steam pressure	194.7	kPa
Q100	Heater duty	339.0	kW
F200	Coooling water flowrate	208.0	kg/min
T200	Cooling water inlet temp	25.0	°C
T201	Cooling water outlet temp	46.1	°C
Q200	Condenser duty	307.9	kW

Table 1: Process variables and their nominal values

J.M. Maciejowski 21 April 2016
Updated: R. Sepulchre, April 2019