Sensors and Signals: Simulink Coursework 2019

1 Coursework Introduction

1.1 What is Simulink?

Simulink is a numerical simulation software package for modelling dynamic systems. It is part of the Matlab matrix analysis package. You build a model of your system using the available **blocks** connected together by **signals**. The simulation engine breaks down the time duration to be simulated into small **steps** and then works through each block, calculating how its outputs will change over the step given the current inputs. Note that this is an **approximate** method as in truth the values of the inputs will change over the step. The **accuracy** of the simulation depends strongly on the step sizes used and the method of predicting the changes to the outputs.

1.2 Coursework format

The coursework activity involves designing a heading reference system for an aircraft that fuses input from two sensors. There are a series of support tasks described in this handout and the main sensor fusion task; however only activities associated with the main task are assessed.

The background activities are described in Section 3 and identified as follows:

- *Bx.x This is a support activity providing background that might help you design your model. You can work through if needed, but it doesn't have to be written up in the report.*

The subsystems you need to construct to complete the coursework are described in Section 4 and identified as follows:

- *Sx.x In these activities you will create the sub-systems for your model. You don't need to write them up in the report, but you should follow these closely so your overall system works and matches what is expected by the mark scheme!*

The activities you will be assessed on are contained in Section 5 and identified as follows:

- *Ax.x This is an assessed activity: you need to write it up in the report and it will be marked. There are 5 activities, and each is worth 10 marks; the coursework has a raw mark out of 50.*

1.3 Coursework assessment

The coursework is assessed via an individual report that is submitted on Blackboard; this can be of 'technical note' style and only needs to contain answers to the activities A5.1 – A5.5.

This coursework should be carried out individually and an individual report handed in

1.4 Objectives

After completing this coursework, you should be able to:

- use Simulink to simulate dynamic systems numerically
- explain qualitatively the limitations of numerical simulation
- compare results from theoretical and numerical simulation
- determine sensor system characteristics from simulation results
- explain the concept of complementary filtering and its use

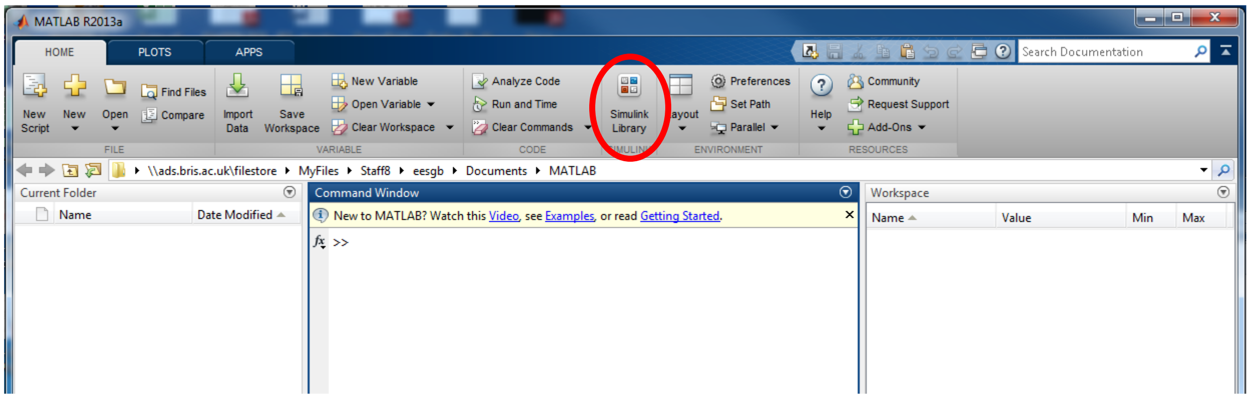
1.5 More Information

See the FAQs section on Blackboard for clarifications to some points.

2 Getting Started with Simulink

This section gives a basic guide to Simulink modelling.

Begin by starting Matlab. The command window should appear similar to figure 1 (version R2013a shown, but it is almost identical in vR018). The Simulink 'button' is highlighted. Click on it to open the Simulink library browser



New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>>
```

Figure 1. Matlab command window

The Simulink library browser is shown in figure 2. Individual blocks that are used to create a model are grouped in 'block-sets'. Clicking on the various block-sets in the menu on the left hand side will let you see what blocks are available to build your model. Items at the top are typically the often-used blocks and more specialist blocks are towards the bottom.

Click on the highlighted button in figure 2 to open a new model, or use 'file>New'

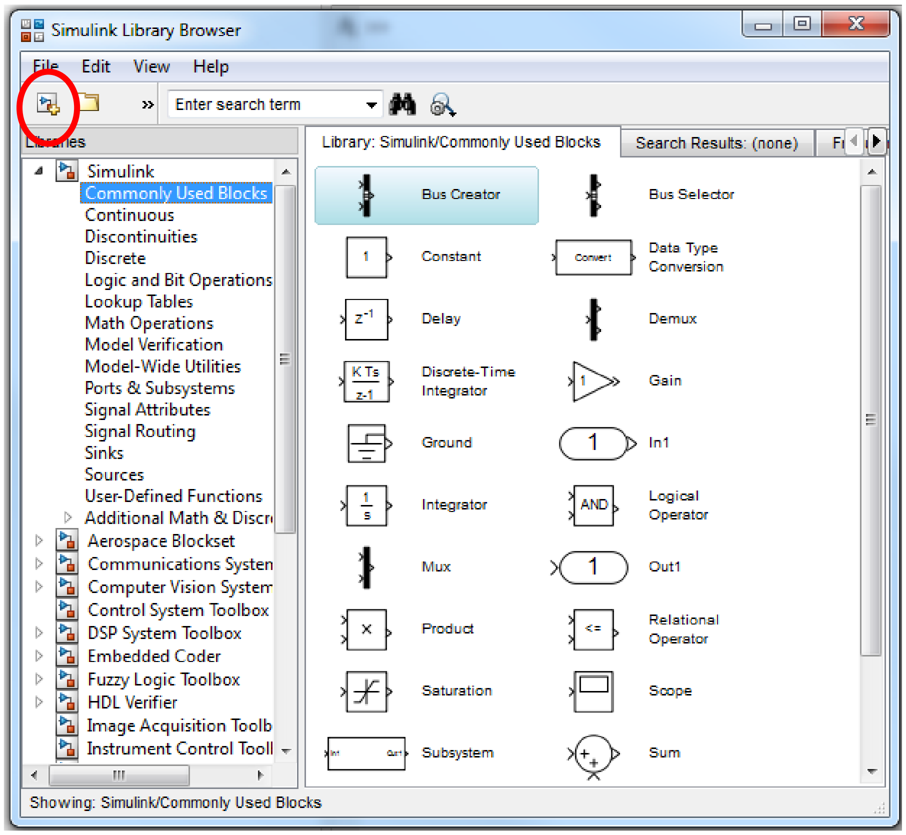


Figure 2. Simulink library browser

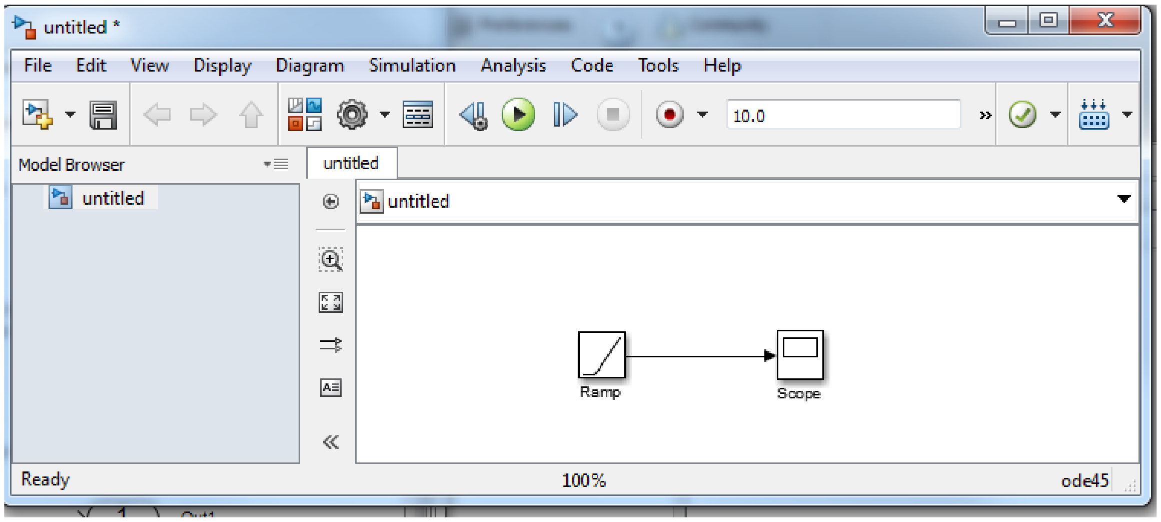


Figure 3. Simulink model browser

The model of your system is built in the model browser (fig. 3). To create a simple model follow these steps;

1. In the library, click on the "ramp" block in the 'sources' block-set and drag it to the model browser window.
2. Do the same with a "scope" block from the "sinks" block-set.
3. To connect the blocks as shown, click on the output (right hand) terminal of the ramp block and drag the cursor to the input (left hand) terminal of the scope block and release.

The ramp block generates a signal that increases constantly. The scope block stores its input signal for viewing.

The model is now complete – you can save it in the familiar Windows ways. You may find it helpful to save the models you build as you go along: future exercises will re-use parts of them.

To **run** the simulation, click the button that looks like a play symbol, ringed in figure 4. The number next to it is the time duration to be simulated, in this case 10 seconds

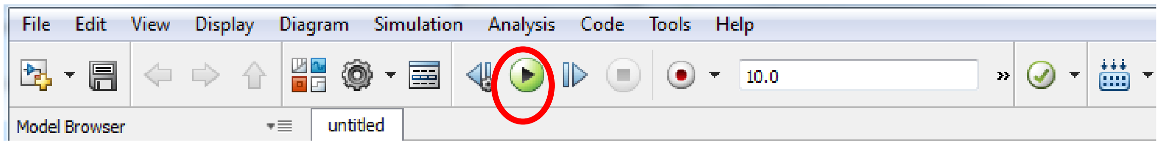
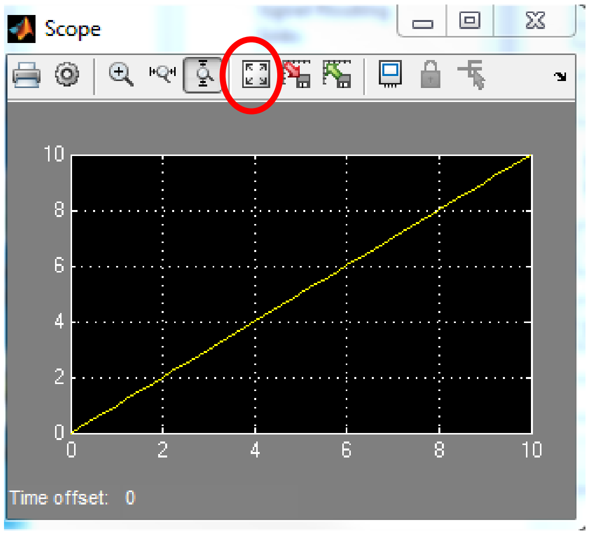
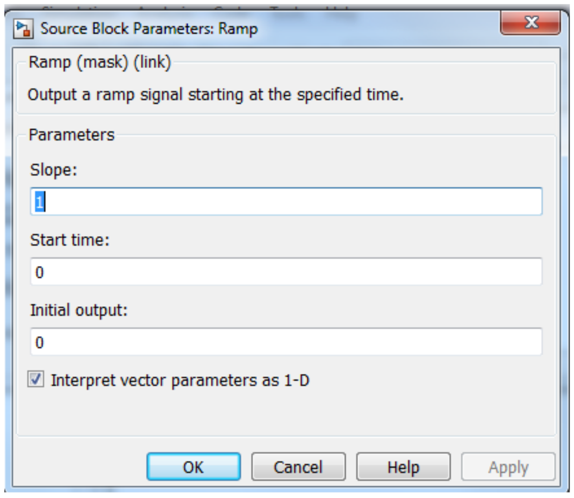


Figure 4. simulation control toolbar

The "scope" block records a signal for viewing. To see the signal it recorded, double click on the scope block. The scope window, fig. 5a, opens. You may need to click the 'autoscale' to scale the axes correctly.





Source Block Parameters: Ramp

Ramp (mask) (link)

Output a ramp signal starting at the specified time.

Parameters

Slope:
1

Start time:
0

Initial output:
0

☑ Interpret vector parameters as 1-D

a) Scope plot

b) Ramp parameters

Figure 5.

To change the parameters of the ramp signal, double click on the ramp block. This opens the parameter window (fig. 5b). Nearly all blocks can be edited this way. Experiment with different settings and make sure you get the signal you expect.

3 Guidance Exercises

3.1 Designing a second order system from the step response

The step input, *i.e.* from 0 to 1 at t=0, is a useful signal to characterise systems. The characteristics ***rise time***, ***settling time*** and ***sensitivity*** describe how a second order system responds to a unit step input.

The ***sensitivity*** is the level to which the output eventually settles; the ***rise time*** of a system is the time taken to go from 10% to 90% of that level; and the ***settling time*** is the time after which the output remains within ±5% of the steady state level. See fig. 6 for graphical explanations.

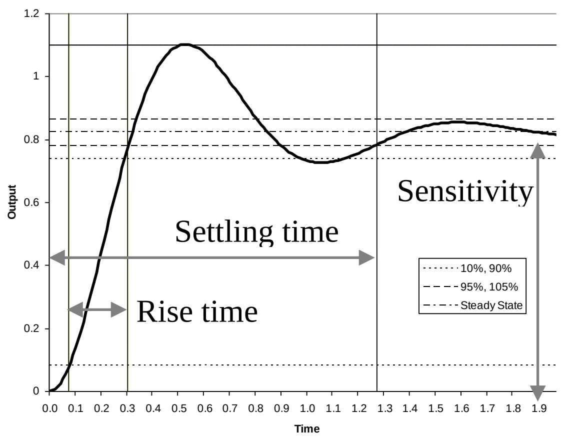


Figure 6. Second order step response

The standard form of a ***second order system*** (*e.g.* a spring-mass-damper combination) has the differential equation $\frac{\ddot{y}}{\omega_n^2} + \frac{2\varsigma\dot{y}}{\omega_n} + y = ku$ where *u* is the input as a function of time and *y* is the output. If the damping is low, *i.e.*$\varsigma << 1$, then we can approximate the time domain characteristics as: rise time $t_R \approx \frac{1.02}{\omega_n}$, settling time $t_S \approx \frac{3}{\varsigma\omega_n}$ and the sensitivity is *k*.

To build a model of this in Simulink choose the “step” block in the “sources” library and the “transfer function” block in the “continuous” library – this represents an LTI system.

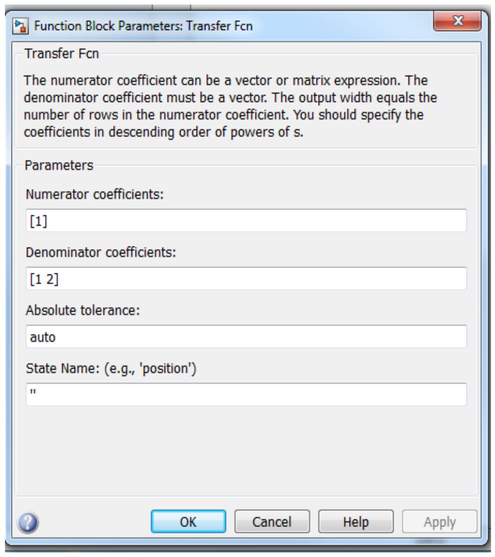
The parameter window for the transfer function block are shown in fig. 7. For a system;

$$c\ddot{y}(t) + b\dot{y}(t) + ay(t) = e\dot{u}(t) + du(t),$$

i.e. with transfer function

$$\frac{Y(s)}{U(s)} = \frac{es+d}{cs^2+bs+a},$$

the denominator would be [c b a] and the numerator would be [e d]. You can have as high an order as you need, but the right-most element is always the coefficient of the "zero'th" derivative, i.e. the signal itself. So the system for the parameters shown in fig. 7 is $\dot{y} + 2y = u$.



Function Block Parameters: Transfer Fcn

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

Parameters

Numerator coefficients: [1]

Denominator coefficients: [1 2]

Absolute tolerance: auto

State Name: (e.g., 'position') ''

Figure 7. Transfer function parameters

- B3.1 First determine the transfer function of a second order system with rise time 0.4s, settling time 18s, and sensitivity 2, using the approximate expressions above.
- B3.2 Simulate the step response of this system, see if it matches your predictions

Useful Simulink features

You can use variables in the block parameters, *e.g.* you can write [N1 N2] in the numerator box. The block takes the value of those parameters from the Matlab workspace. So, you could have a Matlab '.m' file script that calculates the required block parameters and then references those values in the block. The "to workspace" block in the "sinks" library sends the values of a signal to a variable in the Matlab workspace, so that you can save, plot, and analyse data from a simulation using the Matlab command line (or from within a '.m' file).

Signals can be combined using the 'MUX' block which allows scopes or the 'to workspace' block to deal with multiple signals.

You can also run a simulation from the command line (or from within a '.m' file) using the 'sim(*filename*)' command. Type 'help sim' into the MATLAB command window to learn how to send parameters to a simulation called from an .m file.

- B3.3 You can write an '.m' file script to calculate your TF block parameters, call the simulation and then plot the response.

3.2 Investigating the Accuracy

Numerical simulation is an approximate method. In this section, you will check the accuracy of your simulation and investigate the effects of different settings.

Simulink can use a range different solvers and step sizes. Certain solvers are good for certain problems and the step size can be used to trade-off simulation time against accuracy.

- B3.4 Leaving all the simulation parameters as standard, run a long (>100 sec) duration simulation of the step response of the system from Task 1. Take data from near the end of the simulation. Zoom in and look very closely at the signal, does the output signal settle to a steady value?
- B3.5 Read the MATLAB support for configuration parameters – it gives a guide to how you would choose a solver for a particular problem.
- B3.6 Investigate the effect of the integration settings in the Simulation>Configuration Parameters (or Control+E) menu on the simulated output. Change the simulation parameters to give a more accurate result.

3.3 Ramp Response

You can predict the ***steady state*** response of a second order system to a ramp by assuming that the output will also be a ramp signal, but scaled and offset from the input. *i.e.* if the input is $u(t) = t$ then the output would be $y(t) = \alpha t + \beta$ for some values of α and β. Substitute these two expressions into the differential equation representing the system, and you get an equation involving α, β and *t*. Finally, since this equation must hold for all time *t*, solve for α and β by equating coefficients. Note that this doesn't give you the whole signal, just the steady-state solution. You should see the difference when you do the simulation experiment.

- B3.7 Simulate the ramp response of the system model from activity A3.1 and plot the ideal ramp response from your equations above. Plot the error between these signals, what do you notice?

Simulink hints

The "sum" block in the "math" library can be used to add or subtract signals. The "gain" block in the "math" library can be used to scale signals.

3.4 Frequency Response of a System

In this section you will derive a dynamic model of a first order system and determine its frequency response.

The differential equation for a ***first order system*** (*e.g.* a system with one energy storage element) in standard form is $\tau\dot{y} + y = ku$ where τ is known as the ***time constant*** and k is the ***sensitivity.***

Considering the response of the system to a sinusoidal input, the ***steady state gain*** is the ratio of output magnitude over input magnitude (after any transients have died out). The ***bandwidth*** is determined from a graph of steady state gain against frequency and is defined as the frequency at which the steady state gain drops to $1/\sqrt{2}$ of its 'passband' level, i.e. the gain in the region where the graph is flat. The phase response of the system is defined as the phase shift between the input and output waveforms.

- B3.8 Determine the transfer function of a first order dynamic system with time constant 0.01s and unity static sensitivity (i.e. sensitivity of 1).
- B3.9 Simulate the response of this system model to a sinusoidal input. Obtain results for several input frequencies in the range 10rad/s to 1000rad/s. Use can use your data to construct an approximate plot of steady state gain verses frequency, and of phase response verses frequency. Can you identify the bandwidth of the system?

Simulink hints

The "sinusoid" block is in the "sources" library. Represent the system with a "transfer function" block from the "continuous" library. Remember that gain vs. frequency plots typically employ logarithmic scales on both axes. You can use Matlab's loglog command to generate plots of this type. Also, the grid command adds a grid to your plot, making it easier to identify parameters.

MATLAB can also derive gain and phase plots directly from the transfer function. Use the 'tf' function in MATLAB to create a system and the 'bode' function to plot the gain and phase response.

- B3.10 use MATLAB to create a theoretical gain and phase response for the 1st order system and compare with your results from T4.2.

4 Modelling a Sensor System

In this section you will use ideas from Section 3 to construct and analyze a model of an aircraft heading sensing system.

The system comprises a magnetic compass and a spinning-wheel gyro-compass; the data from each is ‘fused’ into a single heading estimation that is better than either sensor could provide alone.

Magnetic compass

A magnetic compass has a mass (compass needle) that is sprung (magnetic attraction to north) and has some damping (in the needle bearing or with fluid around the needle). Its behaviour is modelled as a second order system. The input to our system is true heading and the output will be the indicated heading. Remember, in a real system we don’t know what the true heading is, we only see the heading indicated by the sensor: one advantage of simulations over experiment is that we can often observe variables that are unavailable to us in the real world. *You should note that this model of the compass only describes one degree of freedom (rotation in 1 axis) and is highly simplified for this coursework!*

- S4.1 Construct a model of a magnetic compass. Represent the compass as a second order system with unity sensitivity, rise time 0.4s and settling time 18s. The input to your compass model will be true heading and the output will be the heading indicated by the compass needle.
- S4.2 Produce a Bode plot (gain and phase frequency response) of the compass and use this to identify the useable frequency range of this sensor.

Gyro compass

The gyro-compass is more complex to model, so we will build up a representative model from what we know of its behaviour.

An ideal gyro would remain spinning in the axis in which it was initialised, irrespective of the movement of the support gimbal. However since the support bearing will suffer some friction, changes in the heading of the aircraft will tend to introduce errors gyro .

The gyro law $T = h\dot{\theta}_m$ implies that due to the large value of the angular momentum, h, of the spinning mass, the disturbance torque T (caused by gimbal bearing friction as the gyro outer frame moves with the aircraft), leads only to small rotation rates of the gyro mass spin axis, $\dot{\theta}_m$. Ideally the gyro axis would not rotate at all, thereby providing a constant directional reference, so the angle θ_m represents the gyro measurement error. To model this error, we integrate the equation, giving $\theta_m = {}^1/_h \int T dt$ (assuming that the gyro was initialized without error).

If we assume the disturbance torque is caused by aircraft heading changes and by buffeting of the aircraft, and these are random in nature, we can model them with the 'random number' Simulink block.

In addition, the gyro will also drift slowly over time, which can be due to the spinning of the earth and the aircrafts velocity, and manufacturing/construction limitations. This is generally a slower effect in real systems but to capture it here we can add a mean value (offset) to our random number.

Therefore, modelling the disturbance torque *T* as a Gaussian random signal, a ***Simulink model of a gyro*** would look like that shown in fig. 8. ***Important:*** you should set the standard deviation of the random number to 0.1, which gives a reasonable value for *h,* and set the mean to ~0.02 which should give you around 1.5^o per minute drift

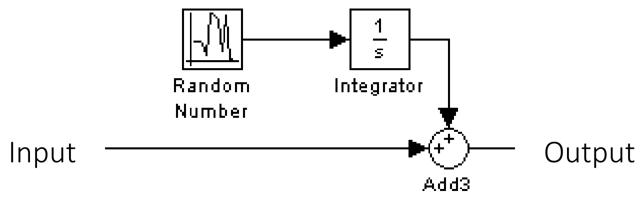


Figure 8. Gyro model

- S4.3 Construct a model of a gyro compass. Represent the gyro-compass as an ideal sensor with error modelled as integrated white noise with an offset, e.g. Fig. 8.
- S4.4 **Sketch** the frequency spectrum of the output of the random number block, and the gain frequency response of an integrator, hence **sketch** the frequency spectrum of the error signal added to the true heading.
- S4.5 Suggest the useable frequency range of the gyro compass – note it is not possible to quantify in the same way as the magnetic compass.

Complimentary filters

You have now described the two sensors and have determined some characteristics of each, *hopefully* you will have found that their negative characteristics do not completely overlap.

We now need to combine the output of these sensors in a way that produces a better estimate of heading than either one used alone – this is called ***sensor fusion***

Since we have seen the sensors differ in frequency domain, we are going to weight the output of each individual sensor according to frequency, thus combining the low frequency information from one sensor with the high frequency information of the other. This approach is one of the simplest in sensor fusion and involves *complimentary filters*

Two filters, $H_1(s)$ and $H_2(s)$ are **complementary** *if* $\mathbf{H_1(s) + H_2(s) = 1}$ *for all s.*

➢ S4.6 Determine the transfer functions for a pair of complementary first order filters with gain of '1' in the pass band. Choose a suitable cut-off frequency for your filters based on the results of T4.2/T4.5.

5 Building and testing your system

In this section you will construct a complete model of your system and investigate its performance.

System model

The overall system diagram is shown in fig. 9.

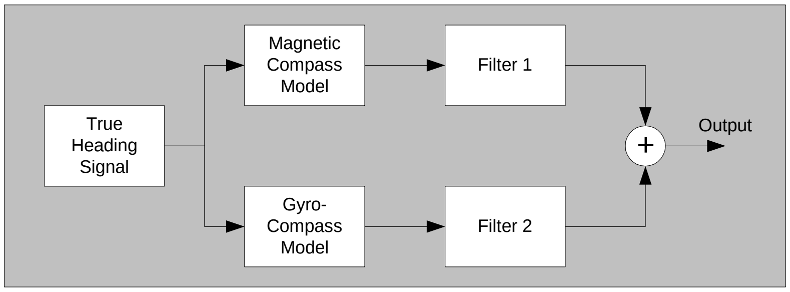


Figure 9. Complete heading system model

➢ A5.1 Construct the complete heading system model in SIMULINK and include the diagram in your report. Also give details of the parameters used in the blocks representing the various subsystems. *(10 marks)*

➢ A5.2 Determine a True Heading Signal to test your system – this is the bearing the aircraft is travelling on at any moment in time. To illustrate the behaviour of the system best, consider a small or medium sized UAV – Simulate a few minutes of flight that might include typical features: straight flight, high and low rate turns, buffeting from gusts etc. (Tip: try and ensure you can see the non-ideal response of both of the individual sensors when their outputs are plotted on a graph. Depending on your heading signal, your system may be well behaved and not demonstrate all sensor behaviour. If this is the case introduce more extreme features in the heading signals). ***(10 marks)***

- T5.3 Produce a graph that plots the True Heading Signal, the output of each sensor individually, and the output of the fused system. Discuss your results. ***(10 marks)***
- T5.4 Propose a metric and quantify the error for each sensor output and for the fused output over the whole simulated flight. Hint – you may want to review the time domain statistical measures lecture. ***(10 marks)***
- T5.5 Explore how variations in the True Heading Signal or in the filter cut-off frequency affect the measured error. ***(10 marks)***