Neuroscience 603

In this document you will find:

1. A Simulink Tutorial
2. Exercises

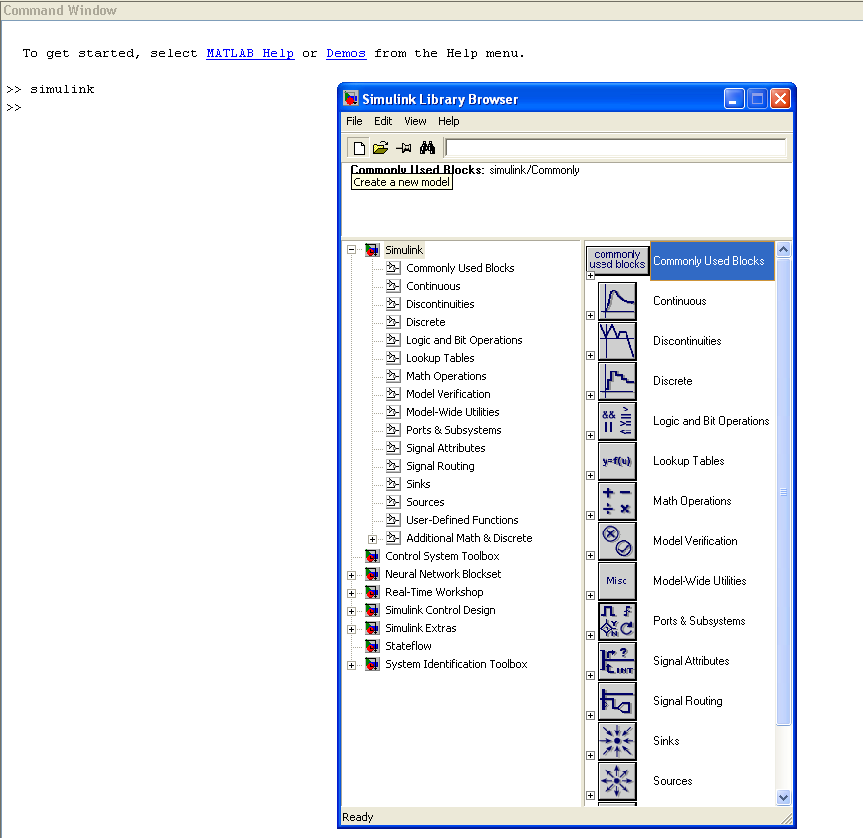
- Linear Systems Analysis of Semicircular canal mechanics vs. afferent responses.

- A control system for the VOR.

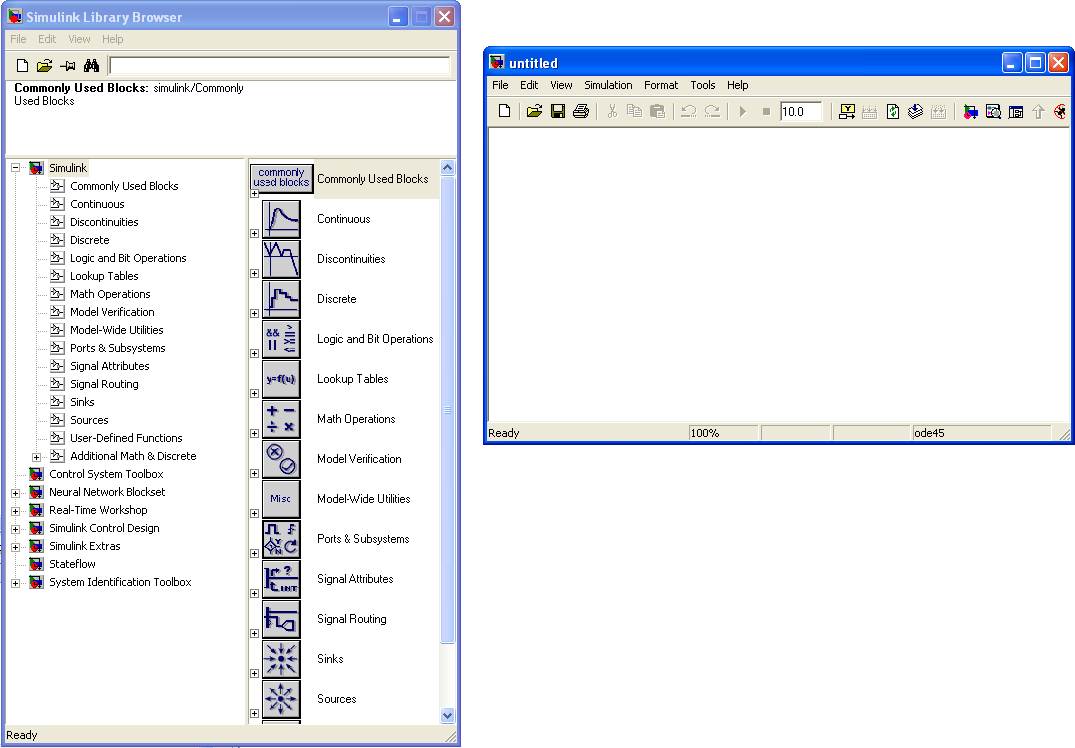
- Mutual information and temporal coding in vestibular pathways.

**Quick Simulink Tutorial:**

1. To start Simulink, open Matlab and type: >> **simulink** in the command window.

****

1. Click on the “new file” icon to create a new model.

****

1. To build a Simulink you simply need to drag and drop system/ function blocks from the Simulink Library browser.

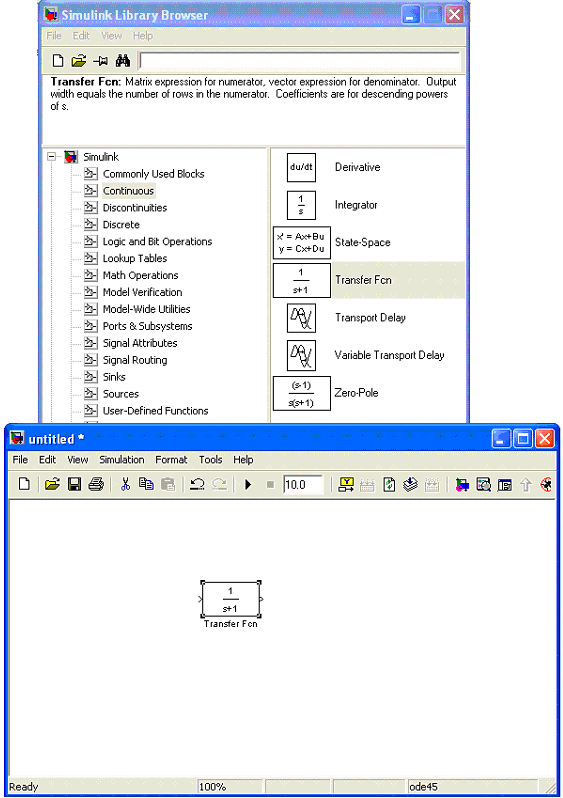
To illustrate let us build a simple first order low-pass filter of the form:

 (1)

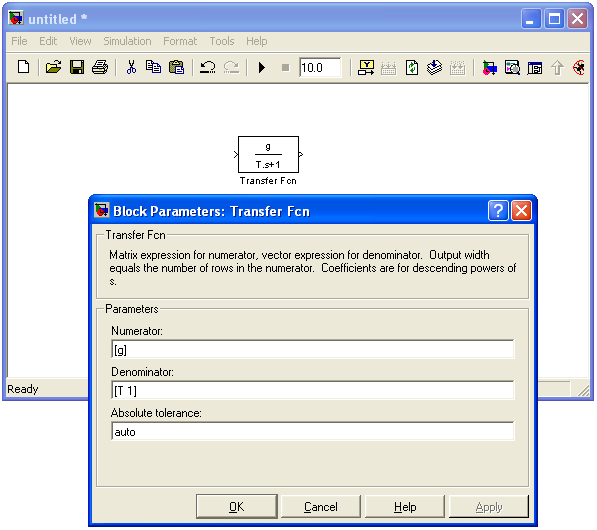
where *H(s)* is the transfer function (in the Laplace domain) of the first order low-pass filter, *T* is the time constant (), *fc* is the cutoff frequency and *g* is the gain of the filter.

In Simulink, you can use either the **“Transfer Fcn”** block or the **“Zero-pole”** block to design a transfer function. Here will illustrate with the “Transfer Fcn”, but as far as the assignment is concerned you might want to use “Zero-Pole”, but it is just a matter of notation and clarity.

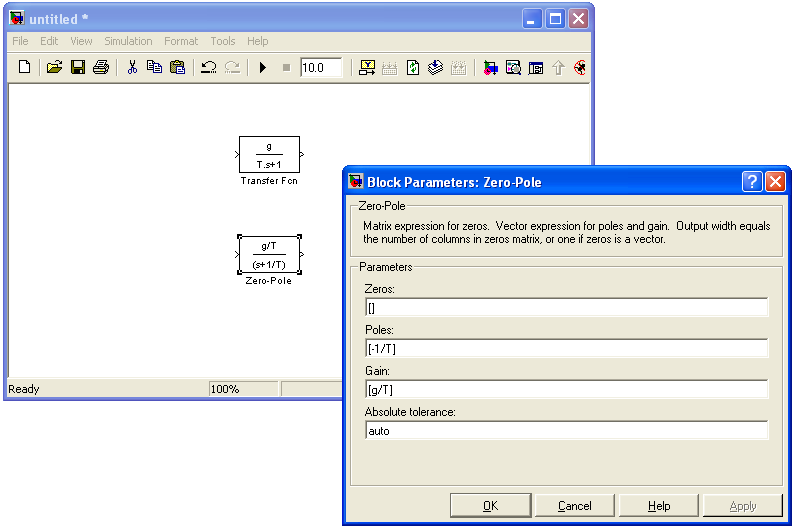
1. Drag and drop the “Transfer Fcn” block from the **Continuous** submenu.



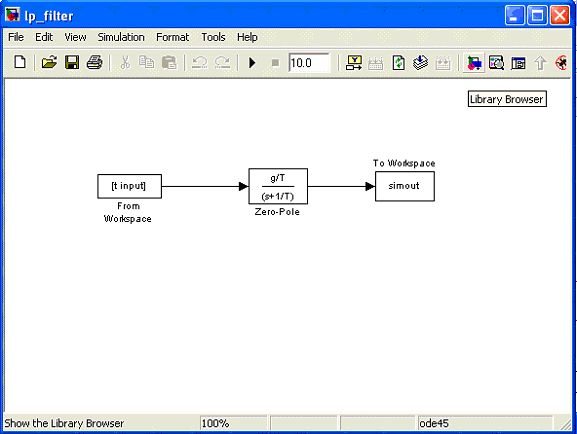
1. Double click on “Transfer Fcn” and enter numerator and denominator coefficients to correspond to equation (1):

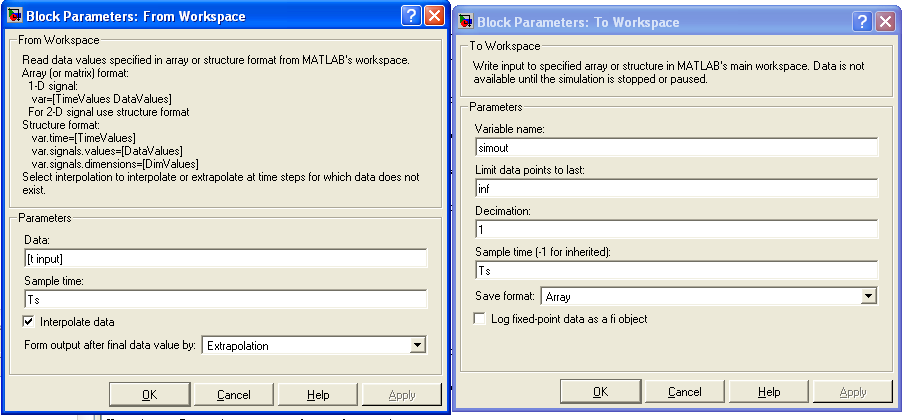


1. You can you use the **“Zero-pole”** block. Do this as well and enter the corresponding poles, zeros (i.e. the roots of the numerator and denominator) and gain:



1. Drag and drop the **“From Workspace”** block from the **Sources** submenu and the **“To Workspace”** block from the **Sinks** submenu. These blocks allow you to load inputs from/ return outputs to the workspace.





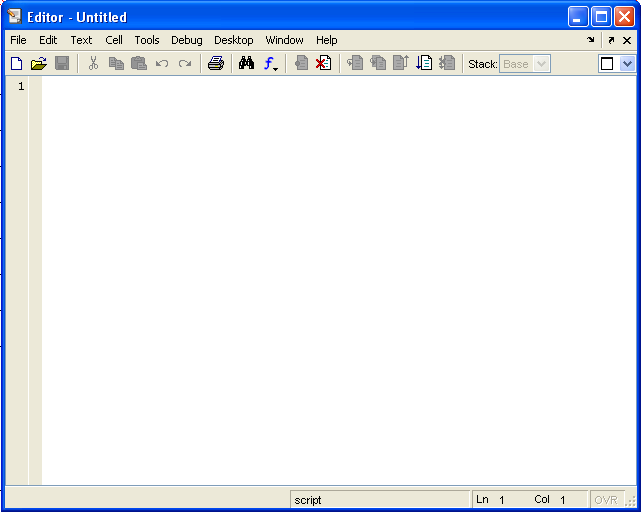
1. Connect the blocks as shown. Click on the first block, hold the CTRL button then click on the block you want to connect it to. Save the model as **“LP\_filter.mdl”**

Now our Simulink model of the first order low pass filter is ready!

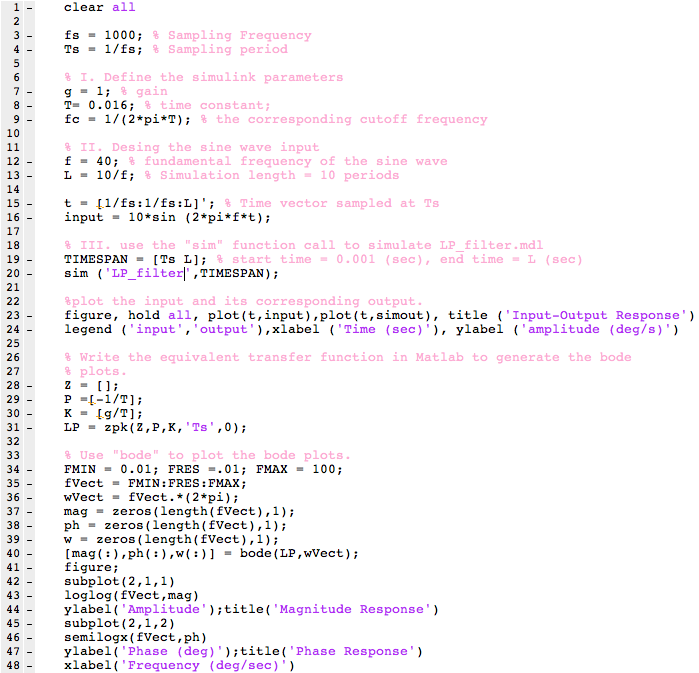
**Note:** In order for Simulink to interpret the variables (shown in the block diagram) they must be defined in the workspace. Alternatively, you can specify the numerical value of these variables directly in Simulink by double clicking on the respective blocks and manually assigning the values.

So let us define these variables. One elegant way to define the Simulink variables (especially when the model is more complex) is to do it in a Matlab “.m” file.

1. In Matlab, click on the “New M-file”  icon in the upper left corner.



1. Type your Matlab program (see tutorial.m):



The above (“**tutorial.m**”) code does the following:

1. Declares the Simulink parameters
2. Designs the sinewave input
3. Plots the output overlaid on the input
4. Also creates the same transfer function that we designed in Simulink, but this time in Matlab using function call “**zpk”** (type: >> **help zpk** in the workspace for more details)
5. Plots the Bode plots of the simple first order Low pass filter with a time constant of 16 ms or a cutoff frequency ~ 10 Hz.

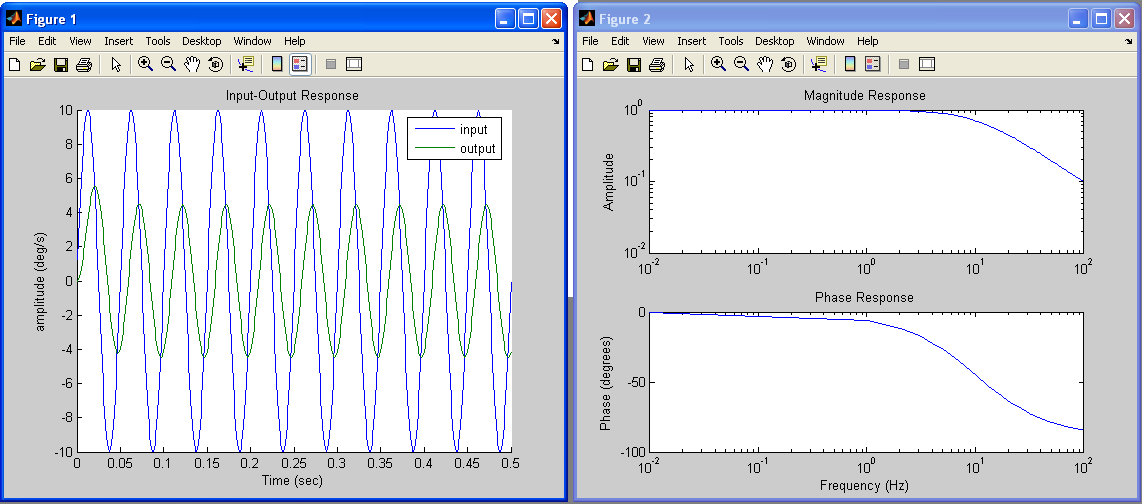


Figure A: Simulation result of “tutorial.m” with a sinewave input at 20 Hz.

**A) Semicircular canal mechanics versus afferent responses.**

1) Illustrate the response dynamics (Bode plots) of the canals versus the afferent response for frequencies spanning 0.01-100Hz. Comment on the main differences.

Steps:

1. In simulink load the provided **“canal\_dynamics.mdl”** circuit block.
2. In Matlab, write an M-file to manipulate the above block diagram as follow:
   1. Use the **sim** (For help type: >> **help sim**) command as shown in **tutorial.m**.
   2. Use the following parameters: T= 0.015 sec, T1= 5.7 sec, T2 = 0.003.
   3. Use 3 sinewave inputs at 0.5, 10 and 20 Hz, to generate 3 output sinewaves for *each* of the 2 systems (i.e. Torsion Pendulum and Canal afferents). For each system, manually compute the input-output gain and phase shift (in degrees), for the 3 frequencies chosen.
   4. Write the equivalent transfer functions in Matlab (using the command **zpk**) and plot their overlaid bode plots (using the command **bode**). Specify the frequency range from 0.01 to 100 Hz. Make sure the x-axis is in log scale with a reasonable resolution.
   5. Do the Bode plots predict well the gains and phases computed in #3?

What is the benefit of using Matlab’s **zpk** command?

**B) A control system for the VOR.**

Figure1 below shows a block diagram for the VOR:

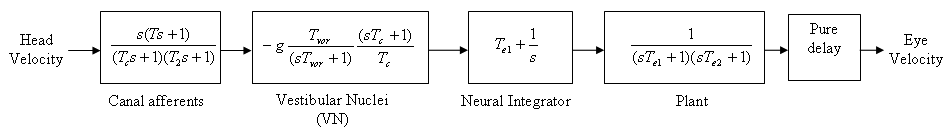


Figure 1: VOR block diagram

1) Use Simulink to load the module “**vor\_block\_diagram.mdl**” that you have been given. This should have the same form as that shown above in figure 1.

2) Run a simulation of the circuit using the following parameter set:

g = -5, Tc = 5.7 sec, T2 = 0.003 sec, T = 0.015 sec, Tvor = 20 sec, Te1 = 20 sec, Te2 = 0.016 sec and delay = 0.007 sec.

Create an M-file to manipulate the **“vor\_block\_diagram.mdl”** model as follows

1. Declare all the above parameters in the m-file and use the function call ***sim*** to run your **“vor\_block\_diagram.mdl”** from Matlab.
2. Use a sinusoidal head velocity as the input signal, with a frequency of 0.5 Hz, amplitude of 10 deg/s, and zero phase offset (ensure it is long enough to reach steady state). Do you find the expected VOR?

3)

1. Generate the Bode plots of the VOR circuit without the pure delay. This can be done by writing the effective transfer function for the system and using the **bode** command.
2. Now consider the effect of a constant physiological delay. This can either be done by manually running sinewave stimuli of different frequencies through the system, or qualitatively explaining the effect, and sketching the resultant Bode plot.
3. In reality, the eye response of the VOR has a reasonably constant phase difference with respect to the head velocity for all relevant frequencies (0-20Hz). Which block of the VOR circuit serves to compensate for the physiological delay at higher frequencies? (Hint: it may be useful to consider the transfer function for each block in the VOR module individually.)

4) Load the provided **“afferent\_step\_response.mdl”** Simulink file and run the provided **“canal\_afferent\_step\_response.m”** Matlab script.

Now modify the **“vor\_block\_diagram.mdl”** to use this same step input, and to output the step responses at the level of a) vestibular afferents, 2) vestibular nuclei, 3) eye movement output (simply add 2 extra outputs). Calculate the time constants manually for each of the 3 responses.

**C) Mutual information and temporal coding in vestibular pathways.**

For this part you have been given 2 data matrices representing one regular and one irregular vestibular afferent (Afferent\_model\_1.mat and Afferent\_model\_2.mat). The afferents have been stimulated with “frozen noise”, which is Gaussian white noise that has been low-pass filtered (8th order butterworth) at a cutoff frequency of 20 Hz. Each matrix has three columns: the first column is the resting discharge, the second is head velocity (i.e., the stimulus), and the third is the spike times (in msec) in response to the stimulus (i.e., the spike train). For all of the following sections, remember that the stimulus only has power up to 20 Hz (i.e., the normal range of frequencies that the vestibular afferents are exposed to). The data is sampled at 1 kHz.

**1) Variability of the resting discharge.**

To start, load the matrix for each afferent. Calculate CV for the resting discharge of each afferent and see if it is a regular or an irregular vestibular afferent (plotting a histogram of the isi distribution for each afferent may help).

**2) Compute the gain of the response.**

a) Load E.mat, which you will need for the ‘**multitaper\_cohere.m**’ function.

b) Use multitaper\_cohere.m to get the power spectrum of the stimulus (pss), the response (spike train) (prr), and the stimulus-response cross spectrum (prs):

[empty,prs, pss, prr, f]=multitaper\_cohere(s, r, 2048,1000,E,1024,'none',8);

c) Compute the gain of the response as the absolute value of the ratio of the input-output cross spectrum over the power spectrum of the stimulus. Normalize the gain by its value at f=1 Hz. Compare the gains for the regular and irregular afferents for low and high frequencies.

**3) Compute the mutual information density.**

To compute the MI density:

a) Compute the coherence C(f) between the stimulus and the spike train at a frequency f as C(f)=|Prs(f)^2|/[Pss(f).Prr(f)], where Prs is the stimulus-response cross spectrum, Pss is the power spectrum of the stimulus, and Prr is the power spectrum of the spike train (remember to get the absolute value of the Prs(f)^2).

b) Compute the mutual information (MI) as MI(f)=-log2[1-C(f)], where C is the coherence. Normalize MI for the mean firing rate of the afferent to get the MI density. What are the differences between the regular and irregular MI density for low and high frequencies? Also, compare the gain and the MI density curves for each afferent type.

**BONUS) Add jitter to the spike train and re-compute the gain and MI.**

Load jitter\_spike\_model\_1.mat and jitter\_spike\_model\_2.mat, separately. The loaded matrix has 4 columns. The first column is the original spike times, columns 2-4 have spike times jittered by adding a random number from a Gaussian distribution with zero mean and standard deviations (SD) of 1-3 msec to each spike time. The stimulus is the same as the one you used in sections C2 and C3.

Compute the gain and MI densities of the jittered spike trains as you did for sections C2 and C3, then averaging the 3 of each cell type. Compare these with the original MI density and normalized gain and relate your findings to the variability in the resting discharge of the afferents (e.g., Does the jittering affect the gain of the regular or irregular afferents? Does the jittering affect the MI density of the afferents? If so, over what range of frequencies?). Relate these findings to the transfer of information using spike times or firing rate for each afferent type.