DAT116 (Mixed-signal system design) Lab 0: MATLAB and Simulink primer/refresher

Lars Svensson
`lars.svensson@chalmers.se`

Version 1.8, November 5, 2018

1 Introduction

The lab series in this course will be based in part on the simulation package Simulink, which is developed and marketed by The Mathworks[1]. Simulink, in turn, is implemented on top of the software package MATLAB, which you will also use during the labs. Lab 0 is intended as an introduction to this software system. You may already be familiar with MATLAB and Simulink; but even then, a refresher may be in order before the real work begins.

MATLAB may be described as a scripting language for numerical work, which also includes numerous library functions for almost any computation. It allows rapid interactive development of data processing tasks and experiments and also offers an abundance of plotting and visualization functions.

Simulink is more specialized: it presents the user with a graphical area where function blocks can be placed and connected in a drag-and-drop fashion, and then allows detailed control of the simulation of the system represented by the interconnected blocks. In order to stay focused on the modeled systems rather than on the modeling environment, we will use a very small subset of the MATLAB/Simulink system.

It will not be necessary to develop MATLAB code to complete the labs. In case you decide to take advantage of MATLAB's scripting features anyway (a very good idea, as you may then easily run variants of your experiments during debugging), make sure that you save your scripts in a file that can be loaded and run from the MATLAB command line (it is also possible to type commands into

[1] `http://www.mathworks.com`

the interpreter line by line).

The MATLAB system includes a code editor, which is launched when you open a MATLAB code file from within the system. (In case you prefer to use Emacs, an elisp file containing a major editing mode for MATLAB can be downloaded from the course PingPong pages. If you have yet another preference, let us know what we might do to facilitate its use with MATLAB.)

Please note that labs 1 and upwards require written lab reports to be prepared, *including Simulink and MATLAB plots;* you need to make sure that you can easily transfer text and diagrams from the simulation system to the document preparation application of your choice (LaTeX, Word, LibreOffice, Google Docs, Pages, etc). *Also, make sure that you can generate PDF files from your document preparation application,* as this is the only accepted format for all submissions.

The 2018 installment of the lab PMs assumes release R2017b of the software, which is based on MATLAB version 9.3.0.713579 . None of the tasks depends on a certain version of MATLAB or Simulink; but if you use other versions, you may find that window panes, menus, libraries, and documentation are arranged somewhat differently.

2 Preparation

This PM points to several parts of the MATLAB documentation, which you will need to go through to prepare for the lab. *Your lab assistant will assume that you are familiar with the contents.* Look for sections marked "**Preparation**" in the text below.

3 Launch MATLAB

- Launch MATLAB. A main application window with several panes will pop up. In addition to the main command pane (center), you may find a command history pane at bottom right, a browsing pane for the current directory at top left, a file preview pane at bottom left, and a workspace window at top right. (The window layout may be modified via the **Layout** menu, which is available at the top of the MATLAB window.)

- A documentation browser window may also pop up immediately; if it does not, call it up with **Help→Documentation** .

The documentation window allows easy browsing and searching though the very extensive documentation of the system. Use this documentation to investigate the capabilities of MATLAB and Simulink! *Note:* the same name may denote a MATLAB function as well as a Simulink block, and overloading is sometimes used to extend function definitions to new argument types. Make sure that you are referring to the right part of the documentation!

Unless you are already a proficient MATLAB user, you will need an introduction to the system and the notation used.

Preparation task 1.

> Find the `MATLAB` section in the documentation browser, and select `Getting Started with MATLAB`. Read at least the tutorials labeled `Desktop Basics`, `Matrices and Arrays`, and `2-D and 3-D Plots`.

Many of the plots we will use in the labs will be created by pre-defined commands. You will, however, also need to generate other simple plots of MATLAB array values.

- In MATLAB, create a vector `a` with ten elements, containing the integers 1 through 10. Make sure you know how to do this without using an explicit `for` loop.
- Next, create another vector `b`, also with ten elements, with each element containing the square of the element in `a` with the same index. Again, do not use a `for` loop; it is doable with one assignment.
- Create a plot of `b` with respect to `a`. Add a title, labels on the axes, and a legend. (How to accomplish this task should be clear once you have read the introductory material indicated above; additional information you might need may be found in the documentation system.)

The MATLAB plot window offers a tool bar with several tools for reformatting, zooming, exporting, etc.

- Find out how to export a plot into a format accepted by the document preparation system of your choice, and verify that you can import them into a document.

- Verify that you can output a PDF file from your document preparation system, and that the plot appears therein without artifacts.

One more step of preparation is needed.

- If you didn't do so already, designate a directory in your file hierarchy where your lab files will be kept. Change to this directory in MATLAB (either by browsing in the top left pane, or by a `cd` command in the command pane). If you launch MATLAB from the command line, the initial working directory will be the one from which MATLAB was started.

4 Launch Simulink

- Launch Simulink from within MATLAB by clicking on the Simulink icon in the toolbar. A "wizard"-style "start page" appears, letting you choose among several project templates.

Here and in the rest of this course, we will use a fixed time step in our simulations.

- Locate the icon titled **Fixed-step** in the category **Simulink** (you may have to expand the "Show more" tag to find it) and click the icon. A simulation model window named **untitled** will appear. This window shows an empty graphical area, where you will construct your model.
- In the simulation model window, select **View→Library Browser**. A Simulink Library Browser window will (eventually) appear.

The libraries include all simulation blocks installed with Simulink. Each icon in the library window represents a collection of Simulink blocks and of other collections. Double-clicking on an icon will display the selected collection in place of the top-level one.

- Open some Simulink library collection windows. Also, double-click on some block icons (with an appearance distinct from the collection icons); a dialog box with block parameters appears, although the parameter fields are greyed out. Close these dialogs after inspecting them.

Preparation task 2.

> Just as you did for MATLAB, you need to read and digest some Simulink introductory material. Find the `Getting Started with Simulink` section under the `Simulink` heading in the documentation browser. Under `Tutorials`, read the material under `Create Simple Model`.

---

5 Create a model

In Simulink, you create a simulation model by instantiating and connecting system blocks from the libraries[2]. You will use the empty model window which is still named **untitled**. As a first step, save your empty model with a suitable name (such as `lab0`).

- In the model window, select

  File→Save As...

  and give the name `lab0` in the dialog. Observe how a file `lab0.slx` appears in the directory browser pane of the MATLAB window, and how the window title changes to **lab0**.

You will now add a *signal source* and a *signal sink* to your model.

- In the Simulink library window, double-click on the `Sources` icon to open the **Sources** collection. Locate the `Sine Wave` block and drag it into the **lab0** window. A copy of the block icon appears in the **lab0** window[3].
- Double-click on the `Sine Wave` block in **lab0**. As before, a parameter dialog opens, but this time the parameter fields are *not* greyed out (as your copy is a local instance of the block, it is possible to modify the parameters). For the top two parameters, choose **Time based** and **Use simulation time**,

[2]It is also possible to build up models hierarchically, and to code models in MATLAB, ADA, FORTRAN, C, or C++. We will stay simple in this course.

[3]Also, an inline shortcut dialog offers an opportunity to give a value to one block parameter. In this case, we will want to set more than one block parameter before running a simulation; but you may find this feature useful later.

and make sure that the checkbox marked **Interpret vector parameters as 1-D** at the bottom of the page is checked. Ignore the other parameters for now. Click **OK** to close the parameter dialog.

- Next, add a `Scope` from the **Sinks** collection to the **lab0** model window.
- Connect the two blocks by clicking close to the visible terminal on one of the blocks and dragging to the terminal on the other block. A wire with a direction arrow should appear, connecting the two blocks.
- Save your model again, by **File→Save**.

6 Run simulation

It should now be possible to simulate the small model you just defined.

- In the model window, do:

  Simulation→Run

  A beep will indicate that simulation has completed.
- Next, double-click on the `Scope` icon. If the simulation was successful, an oscilloscope-like **Scope** window opens, in which the sine wave is shown.

The sine wave parameters in the display should correspond to those of the `Sine Wave` block. You will now modify these parameters to verify the correspondence.

- Open the parameter dialog for the sinewave generator. Modify the **Amplitude** and **Bias** parameters, click **OK**, and re-run the simulation.
- Verify that the sinewave displayed in the **Scope** window changes in correspondence with the parameter changes.

It is now time to modify the frequency of the sine wave.

- Open the `Sine Wave` parameter dialog again and set the **Frequency**[4] parameter to 10. Save your model and re-run the simulation. Does the sinewave display change in the way you expected it to?

[4]Really the angular frequency, ω.

- Zoom in on the scope signal, using the magnifier symbols in the menubar of the **Scope** window. What is the timestep used in the plotting of the sine wave?

In order to minimize runtimes for large simulations, Simulink will always use the longest timestep it believes it can get away with. To avoid artifacts such as the one presently seen in the **Scope** window, Simulink must be forced to use a smaller timestep.

- In the **lab0** window, do:

  Simulation→Model Configuration Parameters

  A new dialog window opens, allowing you to control myriad aspects of the simulation (including the start and stop times).

- Select the **Solver** panel and locate the field **Solver options**. Set the **Type** to **Fixed-step**. Expand the field `Additional options` if necessary. Locate the field `Fixed-step size` and set the value to `0.01`, click **Apply**, and re-run your simulation.

- Zoom in on the scope signal and verify that the new timestep accords with your specification.

For most of the labs experiments, you will want to generate a precise number of sine wave periods. We will now attempt to generate exactly 10 periods of a sine wave in 10 seconds.

- Recall that $y = \sin(\omega \cdot t)$. Calculate the value ω should have to yield exactly 10 periods in 10 seconds.

- Set the `Frequency` parameter for the sinewave to the ω value calculated above. Rerun the simulation and verify that you got the intended number of cycles.

- Note that MATLAB and Simulink understand many common constants such as π. If you used a numeric approximation in the previous step, try again, but use an expression including the literal `pi`; verify that the simulation still runs and produces the right results.

Clearly, constant expressions can be used as parameter values in simulations. Moreover, it is possible to use MATLAB *variables* in parameter value expressions.

- In the command pane in the MATLAB window, define the variable `a` with a given value:

  ```
  >> a = pi / 5
  ```

- Next, use the variable name `a` as the **Frequency** parameter for the sine wave, and re-run the simulation. Does the scope waveform correspond to your expectations?

7 Post-processing

The `Scope` block is a handy tool for a quick look at a signal. In the lab series, however, we will often need to save the result of a simulation for later post-processing in MATLAB.

There are several ways of saving a Simulink signal trace; circumstances will determine which one to use. Here, we will use the `To Workspace` block from the **Sinks** library collection (where we previously found the `Scope` block).

- Drag the `To Workspace` block from the **Sinks** window into your simulation model window. Connect its input to the wire which already connects the `Sine Wave` block with the `Scope` block.
- Open the parameter dialog for the new block. Replace the default name with a recognizable one, such as `foobar`. Also, at the bottom of the dialog, select the save format `Structure With Time`. (This option will ensure that the time values are saved with the signal; the post-processing routines used in the lab series rely on this format.) Click **OK** to close the dialog (note that the variable name in the block icon changes!) and re-run the simulation.
- When the simulation is complete, note that the `Workspace` pane in the MATLAB window now lists a variable `foobar`.

We provide some MATLAB functions which simplify viewing and analysis of the Simulink results. The functions are provided in `.m` files which can be found on the course PingPong page, together with a brief documentation sheet. These files must be available in a known place for MATLAB to find them, and the directory in question must be on MATLAB's search path. The MATLAB function `path` is available to inspect and set the search path.

- Download *all* the provided `.m` files, place them in a common directory and prepend the directory name to the MATLAB search path. (Information on how to use the `path` function is available in the MATLAB documentation system.)
- Use the function `sigview` to display the signal which was saved to the MATLAB workspace.

8 Spectra

So far, we have viewed and manipulated signals in the time domain only. Much specification and verification of mixed-signal systems, however, is performed in the frequency domain instead, using amplitude and especially power spectra. A MATLAB function, `sigspectrum`, is provided to calculate and plot signal power spectra[5].

- Apply `sigspectrum` to the signal previously displayed by `sigview` above. Does the plot conform to expectation?

The `sigspectrum` function should have warned you that it used only 512 data points for the FFT (the signal length was less than 1024, and the FFT needs a number of points that is a power of two; in case of a mismatch, `sigspectrum` uses the trailing end of the signal, for reasons that will become clear later). This means that the FFT was calculated on a signal which contained *partial* cycles of the fundamental. A simple remedy is to let the value for the maximum simulation time step size be 10/1024 rather than 0.01.

- Make this change[6], save your model, re-run the simulation, and re-run `sigspectrum`. It should now use 1024 points for the FFT (verify!). How does the spectrum compare with the previous version? Why?

[5]There are many ways to estimate a signal spectrum; what is used here is a one-sided periodogram. Refer to the MATLAB Signal Processing Toolbox or to a signal processing reference book such as Proakis + Manolakis for *much* more information about spectrum estimation and calculation.

[6]Use the actual expression 10/1024 rather than an approximation!

9 Wrap-up

After completing this lab session, you are expected to be able to carry out the following tasks:

- Launch MATLAB and Simulink.
- Assign values to array elements in MATLAB and generate plots based on these arrays.
- Set the MATLAB search path to include a directory where your user-defined MATLAB functions are stored as `.m`-files.
- Create a new simulation model in Simulink and set its simulation parameters and the parameters of the constituent Simulink blocks.
- Save results from a Simulink simulation to the MATLAB workspace for further analysis.
- Examine and plot the spectra of the signals from a Simulink simulation.
- Transfer plots to the document production system of your choice (Word, LibreOffice, Google Docs, LaTeX, Pages, etc) and generate PDF reports including said plots.