

Matlab and Simulink for Modeling and Control – a Primer

Robert Babuška (r.babuska@its.tudelft.nl)

1 Introduction

For the use of MATLAB and Simulink in the modeling, analysis and control of dynamics systems, three sets of basic tools are available:

1. General MATLAB functions for matrix algebra (`*`, `inv`, `exp`, ...), functions (`sin`, `abs`, ...), data visualization (`plot`, `stairs`, ...) and data storage (`save`, `load`).
2. Control System Toolbox functions and tools for the definition and analysis of linear time-invariant (LTI) systems and synthesis of controllers (`tf`, `ss`, `pole`, `rltool`, `place`, ...).
3. Simulink for the implementation and simulation of nonlinear (and of course also linear) dynamic systems.

In this text we focus on the latter two groups, the reader is assumed to be familiar with general features and functions of MATLAB.

2 Entering and analyzing LTI models in Matlab

Linear time-invariant systems can be entered in the state-space form (`ss`) or as transfer functions, either by means of the numerator and denominator coefficients (`tf`) or by means of the zeros, poles and the gain (`zpk`). Coefficients of polynomials are entered in the descending order of the powers of s (continuous time) or z (discrete time). For instance, the polynomial $A(s) = 3s^3 + 2s + 10$ is defined as the vector: $A = [3 \ 0 \ 2 \ 10]$. Consider the transfer function

$$G(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \cdot \frac{a\omega}{s + a\omega} \quad (1)$$

First, define some values for the parameters in MATLAB:

```
w = 3; z = 0.8; a = 2;
```

Now, the transfer function (1) can be entered by using the function `tf` and the product operator:

```
G1 = tf(w^2, [1 2*w*z w^2])
G2 = tf(a*w, [1 a*w])
G = G2*G1
```

Note that complex systems can be defined in terms of simpler subsystems combined in parallel (operator `+`), series (operator `*`) or feedback (function `feedback`). The standard Matlab operators are overloaded (redefined) for the LTI class of the Control System Toolbox such that they compute the respective combination of the linear systems (instead of addition or multiplication of scalars or matrices).

Another possibility is to define first a system representing the Laplace operator s (a differentiator as the simplest possible LTI dynamic system) and then enter the transfer function as an algebraic expression (in a kind of ‘symbolic’ way):

```
s = tf([1 0], 1)
G1 = w^2 / (s^2 + 2*w*z*s + w^2)
```

Exercise 1: Use functions `pole`, `zero` and `damp` to get information about the poles, zeros of the system and the corresponding frequencies and the relative damping coefficients. Is the system stable? Convert G to the zero-pole-gain and state-space forms (using the functions `zpk` and `ss`).

3 Accessing model parameters

The parameters of an already entered LTI model can be conveniently accessed by using the dot operator. For instance, if $G_s = ss(G)$ is a state-space representation of our transfer function G , the A matrix is accessed by $G_s.A$. The other matrices can be obtained in the same way.

The same convention holds for transfer functions. For example, $G.den\{1\}$ is the denominator of G . The cell-array index $\{1\}$ is necessary, as for MIMO systems, transfer functions are defined by using arrays of polynomials.

Inputs and outputs can be labeled by the names of the physical variables they represent, for instance:

```
G.InputName = 'Control input';  
G.OutputName = 'Velocity';
```

4 Time-domain and frequency-domain plots

There is a variety of functions for plotting all possible responses, such as `step`, `impulse`, `bode`, etc. The most important ones are integrated in an 'LTI viewer' (`ltiview`). As a default plot, the step response is shown. By right-clicking on the plot area, a menu pops up that allows us to choose a different plot or to compute some important characteristics, such the settling time, rise time, gain and phase margins, etc.

Exercise 2: Use the LTI viewer to get the step response, impulse response and the bode plot for our system G . What is the settling time and the rise time?

5 Root locus and other analysis and design methods

The most important methods for system analysis and controller design are implemented in the Control Toolbox. They include the root locus method (functions `rlocus` and `rltool`), the Nyquist plot (`nyquist`), the Bode plot (`bode`), etc. Function `sisotool` integrates several of the design techniques in one Graphical User Interface.

Exercise 3: Use function `rlocus` to plot the root locus of our system G (assuming negative feedback with a varying gain K). Note that the default settings may not always produce a satisfactory plot. In this particular case, the branches may seem to end before reaching the real axis. From the theory, however, we know that they stretch to infinity, as this system has no zeros. For the proper interpretation of the results, it is therefore important to understand the theory well. The axes settings can be changed through the 'Properties...' option (right-click somewhere within the axes).

Function `rltool` is a bit more user-friendly and more flexible. You can drag the poles by means of the mouse to find out which gain corresponds to the current pole location. Find out, for which gain the pair of complex conjugate poles crosses the real axis. What is the corresponding location of the third pole?

Exercise 4: As a homework, plot the root loci for the three examples given in the corresponding lecture.

6 Discrete-time systems

Discrete-time systems can be entered directly in a way analogous to continuous-time systems. The sampling time is required as an additional parameter. For instance, for the sampling time $h = 0.1$, the a zero-order hold model of an integrator, $H(z) = h/(z - 1)$, is entered by:

```
H = tf(h, [1 -1], h)
```

Continuous-time systems can be converted to discrete-time systems by using the function `c2d`:

```
Gd = c2d(G, h)
```

Exercise 5: Convert the discrete-time integrator H to continuous time. What is the pole of this system?

7 Simulink basics

Using Simulink is very straightforward. Open the top-level block library by entering the command `simulink` in the MATLAB workspace. Create a new model and drag&drop the required blocks in the model. Connect them with lines using the right mouse button. Save the model and set up the simulation parameters (integration method, stop time, etc.).

Exercise 6: Implement a simulation model for a system described by the differential equation:

$$\ddot{y} + 0.5\dot{y} + y^2 = u$$

8 Exporting data from Simulink to Matlab

During simulations, signals can be displayed by using the 'Scope' block, located in the 'Sinks' library. To further process or save the data in MATLAB you need to export them to the workspace. This can be done either by using the 'To Workspace' block from the same library or directly from the scope. To enable data export from the scope, click on the 'Parameters' icon (second from the left), select 'Data history' and check the 'Save data to workspace' box. Choose 'Array' in the format field and a suitable name for the variable. In all subsequent simulations, the data will be exported to that matrix variable in the workspace. The first column is the time base, the remaining columns are the signals displayed in the scope.

9 Linearization of a Simulink model

There are two possibilities to linearize a nonlinear model:

- *Analytically*: by hand or using symbolic maths software such as Mathematica, Maple or the Symbolic Toolbox of MATLAB.
- *Numerically* by applying the `trim` and `linmod` functions of MATLAB.

The second possibility will be used here. First, the inputs and outputs of the model must be connected to the 'Inport' and 'Outport' blocks, respectively. In this way, we tell Simulink what the inputs and outputs of our system are. The `trim` function can then be used to search for an equilibrium of the nonlinear system numerically. A reasonable initial guess for x_0 , u_0 and y_0 should be provided. The additional parameters of this function are the indices of the inputs, states and outputs that are not free to vary during the optimization search. A typical example of such a variable is the state variable corresponding to the operating point. Let us linearize the nonlinear model around an operating point $x_0(1) = 2$:

```
x0 = [2;0];  
[x0,u0,y0]=trim(file,x0,[],[],1)
```

The `linmod` function extracts the matrices of a linear model by means of numerical linearization at the equilibrium. Once this model is available, it can be used for analysis or control design.

```
[A,B,C,D] = linmod(file,x0,u0,y0);  
sys = ss(A,B,C,D);
```

Exercise 7: Choose another operating point and extract a linear model at that point. Compare to the model obtained above in terms of its dynamics and the DC gain.

10 Implementing LTI models in Simulink

Linear time-invariant systems can be implemented in Simulink either by using the blocks from the 'Continuous' and 'Discrete' libraries, or by using the 'LTI System' block from the 'Control System Toolbox' library. The latter alternative is easier, as the 'LTI System' block accepts continuous and discrete-time systems in all three forms (`ss`, `tf` and `zpk`).

Exercise 8: Compare in simulations the linearized model obtained above with the original nonlinear model. Add to the operating-point input u_0 a square-wave signal (use the 'Signal Generator') with frequency of 1 Hz and amplitude 1.