Matlab and Simulink for Modeling and Control

Robert Babuska
Delft University of Technology
482 PUBLICATIONS 19,261 CITATIONS
SEE PROFILE

Stefano Stramigioli
University of Twente
435 PUBLICATIONS 10,073 CITATIONS
SEE PROFILE

Matlab and Simulink for Modeling and Control

Robert Babuška and Stefano Stramigioli

November 1999

Delft University of Technology

Control Laboratory
Faculty of Information Technology and Systems
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, The Netherlands

1 Introduction

With the help of two examples, a DC motor and a magnetic levitation system, the use of MATLAB and Simulink for modeling, analysis and control design is demonstrated. It is assumed that the reader already has basic knowledge of MATLAB and Simulink. The main focus is on the use of the Control System Toolbox functions. We recommend the reader to try the commands out directly in MATLAB while reading this text. The examples have been implemented by the authors and can be downloaded from http://lcewww.et.tudelft.nl/˜et4092. The implementation is done in MATLAB version 5.3 and has also been tested in version 5.2.

2 Modeling a DC Motor

In this example we will learn how to develop a linear model for a DC motor, how to analyze the model under MATLAB (poles and zeros, frequency response, time-domain response, etc.), how to design a controller, and how to simulate the open-loop and closed-loop systems under SIMULINK.

2.1 Physical System

Consider a DC motor, whose electric circuit of the armature and the free body diagram of the rotor are shown in Figure 1.

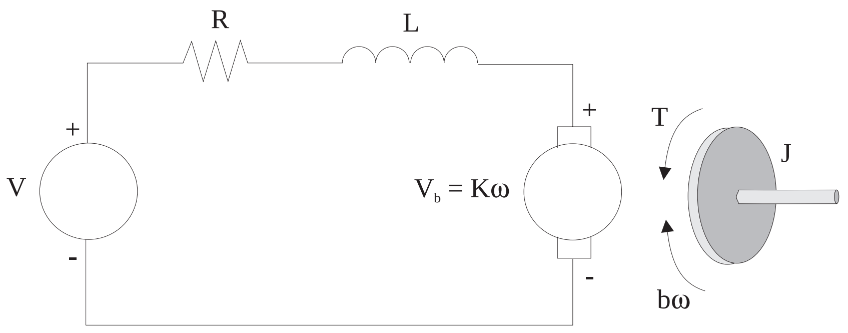

Figure 1: Schematic representation of the considered DC motor.

The rotor and the shaft are assumed to be rigid. Consider the following values for the physical parameters:

moment of inertia of the rotor	$J =$0.01 kg$\cdot$ m^2
damping (friction) of the mechanical system	$b =$0.1 Nms
(back-)electromotive force constant	$K =$0.01 Nm/A
electric resistance	$R = 1\ \Omega$
electric inductance	$L = 0.5$ H

The *input* is the armature voltage V in Volts (driven by a voltage source). Measured variables are the angular velocity of the shaft ω in radians per second, and the shaft angle θ in radians.

2.2 System Equations

The motor torque, T, is related to the armature current, i, by a constant factor K:

$$T = Ki\,. \tag{1}$$

The back electromotive force (emf), V_b, is related to the angular velocity by:

$$V_b = K\omega = K\frac{d\theta}{dt}\,. \tag{2}$$

From Figure 1 we can write the following equations based on the Newton's law combined with the Kirchhoff's law:

$$\begin{aligned} J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} &= Ki, && (3)\\ L\frac{di}{dt} + Ri &= V - K\frac{d\theta}{dt}\,. && (4)\end{aligned}$$

2.3 Transfer Function

Using the Laplace transform, equations (3) and (4) can be written as:

$$\begin{aligned} Js^2\theta(s) + bs\theta(s) &= KI(s), && (5)\\ LsI(s) + RI(s) &= V(s) - Ks\theta(s), && (6)\end{aligned}$$

where s denotes the Laplace operator. From (6) we can express $I(s)$:

$$I(s) = \frac{V(s) - Ks\theta(s)}{R + Ls}, \tag{7}$$

and substitute it in (5) to obtain:

$$Js^2\theta(s) + bs\theta(s) = K\frac{V(s) - Ks\theta(s)}{R + Ls}\,. \tag{8}$$

This equation for the DC motor is shown in the block diagram in Figure 2.

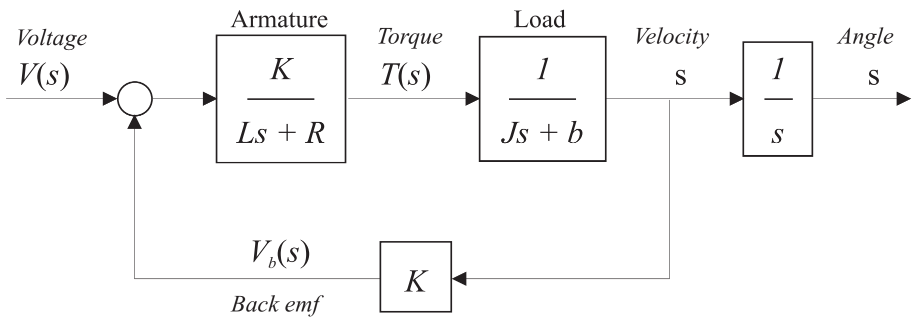

Figure 2: A block diagram of the DC motor.

From equation (8), the transfer function from the input voltage, $V(s)$, to the output angle, θ, directly follows:

$$G_a(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s[(R + Ls)(Js + b) + K^2]}\,. \tag{9}$$

From the block diagram in Figure 2, it is easy to see that the transfer function from the input voltage, $V(s)$, to the angular velocity, ω, is:

$$G_v(s) = \frac{\omega(s)}{V(s)} = \frac{K}{(R + Ls)(Js + b) + K^2}\,. \tag{10}$$

3 MATLAB Representation

The above transfer function can be entered into Matlab by defining the numerator and denominator polynomials, using the conventions of the MATLAB's Control Toolbox. The coefficients of a polynomial in s are entered in a descending order of the powers of s.

Example: The polynomial $A = 3s^3 + 2s + 10$ is in MATLAB entered as: `A = [3 0 2 10]`.

Furthermore, we will make use of the function conv(A,B), which computes the product (convolution) of the polynomials A and B. Open the M-file motor.m. It already contains the definition of the motor constants:

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
```

The transfer function (9) can be entered in MATLAB in a number of different ways.

1. As $G_a(s)$ can be expressed as $G_v(s) \cdot \frac{1}{s}$, we can enter these two transfer functions separately and combine them in series:

   ```
   aux  = tf(K,conv([L R],[J b]))
   Gv  = feedback(aux,K);
   Ga  = tf(1,[1 0])*Gv;
   ```

   Here, we made use of the function feedback to create a feedback connection of two transfer functions and the multiplication operator *, which is *overloaded* by the LTI class of the Control System Toolbox such that is computes the product of two transfer functions.

2. Instead of using convolution, the first of the above three commands can be replaced by the product of two transfer functions:

   ```
   aux = tf(K,[L R])*tf(1,[J b]);
   ```

3. Another possibility (perhaps the most convenient one) is to define the transfer function in a symbolic way. First introduce a system representing the Laplace operator s (differentiator) and then enter the transfer function as an algebraic expression:

   ```
   s = tf([1 0],1);
   Gv = K/((L*s + R)*(J*s + b) + Kˆ2);
   Ga = Gv/s;
   ```

It is convenient to label the inputs and outputs by the names of the physical variables they represent:

```
Gv.InputName  = ’Voltage’;
Gv.OutputName = ’Velocity’;
Ga.InputName  = ’Voltage’;
Ga.OutputName = ’Angle’;
```

Now by calling motor from the workspace, we have both the velocity (Gv) and the position (Ga) transfer functions defined in the workspace.

3.1 Exercises

1. Convert Gv and Ga into their respective state-space (function ss) and zero-pole-gain (function zpk) representations.

2. What are the poles and zeros of the system? Is the system stable? Why?

3. How can you use MATLAB to find out whether the system is observable and controllable?

4 Analysis

The Control System Toolbox offers a variety of functions that allow us to examine the system's characteristics.

4.1 Time-Domain and Frequency Responses

As we may want plot the responses for the velocity and angle in one figure, it convenient to group the two transfer functions into a single system with one input, the voltage, and two outputs, the velocity and the angle:

```
    G = [Gv; Ga];
```

Another way is to first convert G_a into its state-space representation and then add one extra output being equal to the second state (the velocity):

```
    G = ss(Ga);
    set(G,’c’,[0 1 0; 0 0 1],’d’,[0;0],’OutputName’,{’Velocity’;’Angle’});
```

Note that this extension of the state-space model with an extra output has to be done in one set command in order to keep the dimensions consistent.
Now, we can plot the step, impulse and frequency responses of the motor model:

```
    figure(1); step(G);
    figure(2); impulse(G);
    figure(3); bode(G);
```

You should get the plots given in Figure 3 and Figure 4.

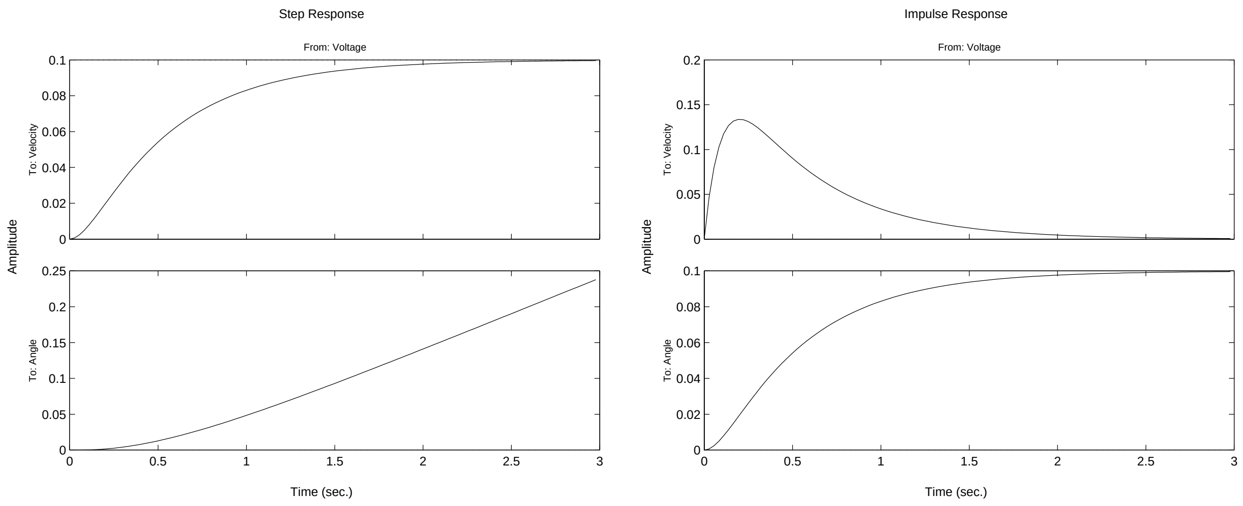

Figure 3: Step and impulse response.

4.2 Exercise

1. Simulate and plot in MATLAB the time response of the velocity and of the angle for an input signal $\cos 2\pi t$, where t goes from 0 to 5 seconds.

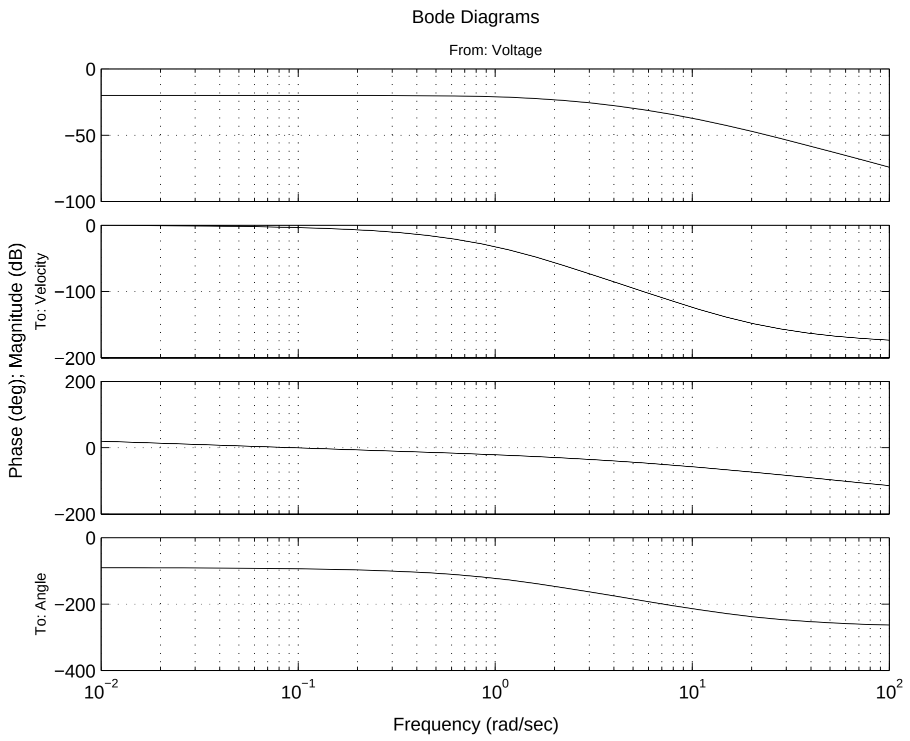

Figure 4: Bode diagram.

5 Control Design

Let us design a PID feedback controller to control the velocity of the DC motor. Recall that the transfer function of a PID controller is:

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}, \tag{11}$$

where u is the controller output (in our case the voltage V), $e = u_c - y$ is the controller input (the control error), and K_p, K_d, K_i are the proportional, derivative and integral gains, respectively. A block diagram of the closed-loop system is given in Figure 5.

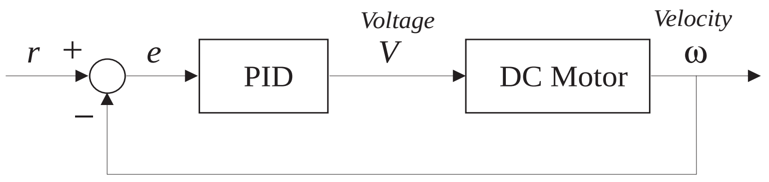

Figure 5: Closed-loop system with a PID controller.

5.1 Proportional Control

First, try a simple proportional controller with some estimated gain, say, 100. To compute the closed-loop transfer function, use the feedback command. Add the following lines to your m-file:

```
Kp = 100;
Gc = feedback(Gv*Kp,1);
Gc.InputName = 'Desired velocity';
```

Here Gc is the closed-loop transfer function. To see the step response of the closed-loop system, enter:

```
figure(4); step(Gc,0:0.01:2);
```

You should get the plot given in Figure 6:

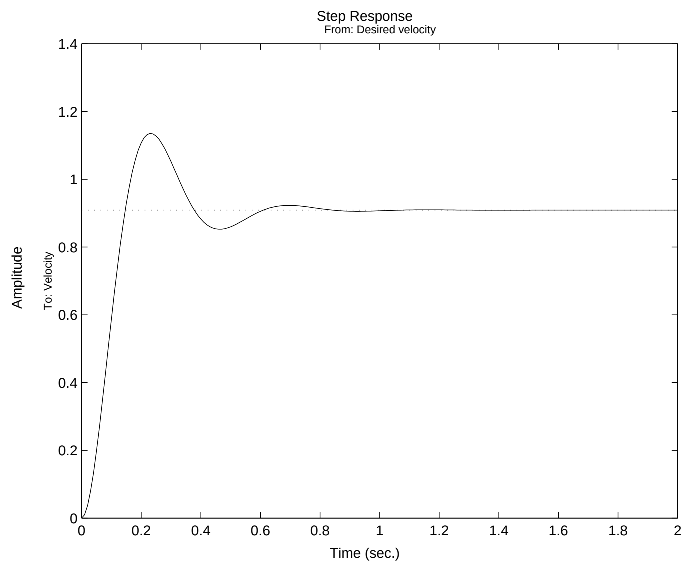

Figure 6: Closed-loop step response with a P controller.

To eliminate the steady-state error, an integral action must be used. To reduce the overshoot, a derivative action can be employed. In the following section, a complete PID controller is designed.

5.2 PID Control

Let us try a PID controller. Edit your M-file so that it contains the following commands:

```
Kp = 1;
Ki = 0.8;
Kd = 0.3;
C = tf([Kd Kp Ki],[1 0]);
rlocus(Ga*C);
Kp = rlocfind(Ga*C);
Gc = feedback(Ga*C*Kp,1);
figure(9); step(Gc,0:0.01:5)
```

The rlocus and rlocfind functions are used to select the overall gain of the PID controller, such that the controller is stable and has the desired location of the poles (within the defined ratio among the K_p, K_i and K_d constants). If the design is not satisfactory, this ratio can be changed, of course. We should obtain a plot similar to the one in Figure 7:

5.3 Exercise

1. Use the root locus and the Nyquist criterion to find out for what value of the gain K_p the proportional controller for the angle $G_a(s)$ becomes unstable.

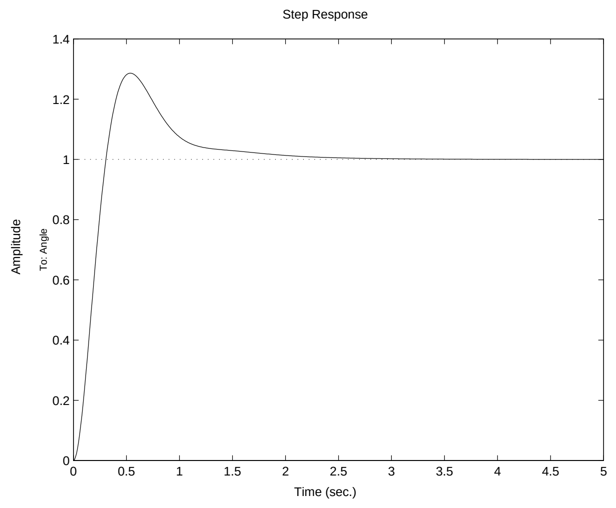

Figure 7: Closed-loop step response with a PID controller.

6 SIMULINK Model

The block diagram from Figure 2 can be directly implemented in SIMULINK, as shown in the figure Figure 8:

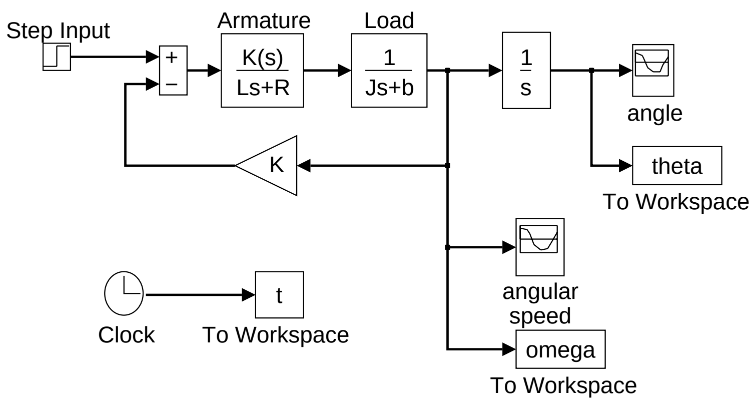

Figure 8: SIMULINK block diagram of the DC motor.

Set the simulation parameters and run the simulation to see the step response. Compare with the response in Figure 3. Save the file under a new name and remove the position integrator along with the 'Graph' and 'To Workspace' blocks. Group the block describing the DC motor into a single block and add a PID controller according to Figure 5. The corresponding SIMULINK diagram is given in Figure 9. Experiment with the controller. Compare the responses with those obtained previously in MATLAB.

7 Obtaining MATLAB Representation from a SIMULINK Model

From a SIMULINK diagram, a MATLAB representation (state space, transfer function, etc.) can be obtained. The 'Inport' and 'Outport' blocks must be added to the SIMULINK diagram, as shown in Figure 10.
Then we can use the linmod command to obtain a state-space representation of the diagram:

```
[A,B,C,D] = linmod('filename');
```

where filename is the name of the SIMULINK file.

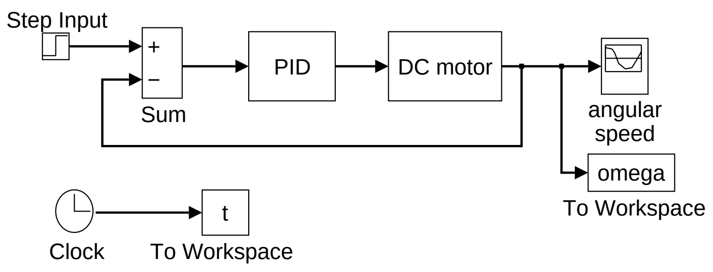

Figure 9: SIMULINK block diagram of the DC motor with a PID controller.

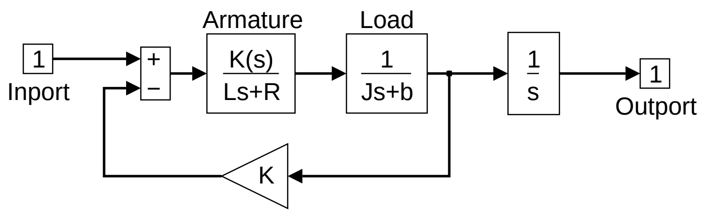

Figure 10: SIMULINK block diagram of the DC motor with 'Inport' and 'Outport' blocks.

7.1 Exercise

1. Convert the four matrices A, B C and D into a corresponding state space LTI object. Convert this one into a transfer function and compare the result with the transfer function entered previously in MATLAB.

8 Linearization of Nonlinear Simulink Model

In this section, we will use an example of a highly nonlinear system to show how to linearize a nonlinear Simulink model and extract the linearized model to MATLAB for control design purposes.

8.1 Magnetic Levitation System

Magnetic levitation as a friction-less support for high-speed trains, in bearings of low-energy motors, etc. It consists of an electromagnet which is attracted to an object made of a magnetic material (such as a rail). The control goal is to keep the air gap between this material and the electromagnet constant by controlling the current in the coil. A schematic drawing is given in Figure 11.

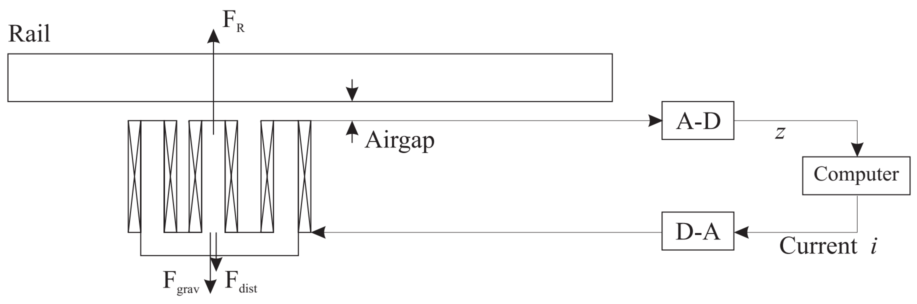

Figure 11: *Schematic drawing of the magnetic levitation system.*

The position and the motion of the object in the magnetic field are dependent on the forces that act on it. These forces are: (i) the gravitational force, (ii) the electromagnetic force, and (iii) a disturbance force. The dynamic equation of the system is derived from the basic law $F = ma$,

$$\frac{d^2}{dt^2}y(t) = \frac{1}{m}(F_{grav} + F_{dist} - F_R), \tag{12}$$

where $F_{grav} = mg$, F_{dist} is an unknown disturbance, and the electromagnetic force is

$$F_R = \frac{\mu_0 N^2 A i^2(t)}{2y^2(t)} = K_{mag}\frac{i^2(t)}{y^2(t)}. \tag{13}$$

In our example, we use $K_{mag} = 17.8$ μH and $m = 8$ kg.

8.2 Nonlinear Simulink Model

The nonlinear equation (12) is implemented in a Simulink model given in Figure 12.

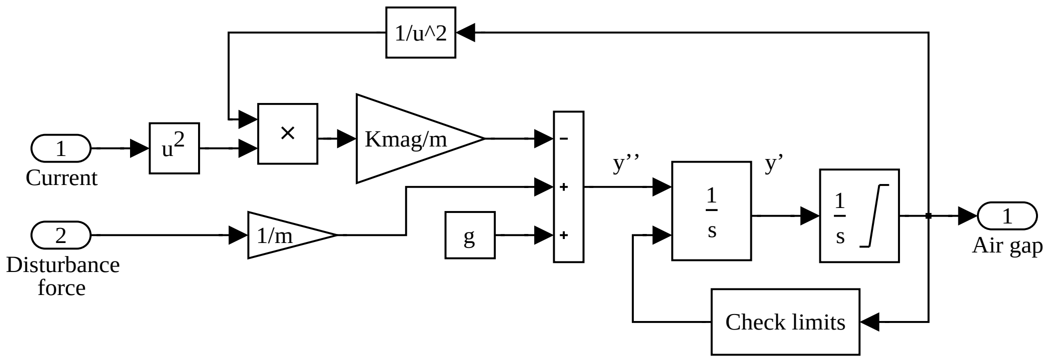

Figure 12: Nonlinear Simulink model (bearing.mdl) of the magnetic levitation system.

8.3 Linearization

Let us linearize the nonlinear model around an operating point $y_0 = 2$ mm. 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 explored here (you can do the first one as an exercise). Let us use the following script (lin.m):

```
params;                 % a script with definition of system’s parameters
file = ’bearing’;       % nonlinear Simulink model to be linearized
u0 = [10; 0];           % initial input guess   [input; disturbance]
y0 = 0.002;             % initial output guess
x0 = [y0 0]’;           % initial state guess
[x0,u0]=trim(file,x0,u0,y0,[],[2],[]);
[A,B,C,D] = linmod(file,x0,u0);
sys = ss(A,B,C,D);      % make an LTI object
```

The trim function numerically searches for an equilibrium of the nonlinear system. A reasonable initial guess (x0, u0 and y0) must be provided. The additional parameters of this function are indices of the inputs, states and outputs that are not free to vary during the search. A typical example of such a variable is the state variable corresponding to the operating point.
The linmod function extracts the matrices of a linear model obtained by numerical linearization at the equilibrium. Once this model is available, it can be used for analysis or control design.

8.4 Exercise

1. Choose another operating point and extract a linear model at that point. Compare to the model obtained above in terms their gains, poles and zeros.

9 Concluding Remarks

The authors hope that this text has been useful and would appreciate receiving feedback from you. Let us know if you have found any errors and omissions or if you have suggestions for improvements. Send them preferable by e-mail to: R.Babuska@ITS.TUDelft.NL.