Technische Universität Berlin
Institute for Process an Chemical Engineering
Chair of Control Engineering

IESS – Laboratory 2

Simulating the dynamic model of a self-balancing robot

In this laboratory, we will explore the use of SIMULINK, a powerful simulation and modeling tool, to design and simulate a self-balancing robot model. SIMULINK is an extension of MATLAB that provides a graphical environment for the construction of a block diagram representation of a system. The aim of this laboratory is to introduce the SIMULINK environment and give an understanding of the dynamics of our self-balancing robot model. Throughout this laboratory, we will use SIMULINK's graphical interface for simulating the dynamic model you derived in Laboratory 1. By the end of this laboratory, we will have a working simulation model of a self-balancing robot and will have gained valuable insights into the power of SIMULINK as a simulation and modeling tool.

In this laboratory, we will use SIMULINK to model a first principles model of a free-standing robot. The requirements for the laboratory include:

- Completion of Laboratory 1 – Modeling a self-balancing robot (some important information from this Laboratory is summarized in the Self-balancing robot model from Laboratory 1 section below)
- Familiarity with the MATLAB and SIMULINK environment

If you have never used SIMULINK, it is recommended that you complete the SIMULINK Onramp course before continuing with this lab.

Task 1 – Initializing parameters and the Euler-Lagrange model in MATLAB

1. Write a MATLAB script to initialize the parameters of the model from Table 1 as global parameters and run the script so they are available in your MATLAB workspace.

2. The standard Euler-Lagrange structure is given by

$$\mathcal{M}(q)\ddot{q} + \mathcal{C}(q,\dot{q})\dot{q} + \mathcal{G}(q) = \bar{\tau}\,. \tag{1}$$

   Use the Euler-Lagrange model from Laboratory 1 to implement the $\mathcal{M}(q)$ matrix in a MATLAB function like that shown in Listing 1.

Listing 1: Example MATLAB function for the $\mathcal{M}(q)$ matrix

```
function MM = M_matrix(q)

    phi = q(1);
    theta = q(2);

    global r m M J I l Mlr;
```

```
    MM(1,1) = J + (M+m)*r^2;
    MM(1,2) = J + (M+m)*r^2 + Mlr*cos(theta);
    MM(2,1) = J + (M+m)*r^2 + Mlr*cos(theta);
    MM(2,2) = J + (M+m)*r^2 + 2*Mlr*cos(theta) + I + M*l^2;

end
```

3. Use the Euler-Lagrange model from Laboratory 1 to implement the $\mathcal{C}(q,\dot{q})$ matrix in a MATLAB function.

4. Use the Euler-Lagrange model from Laboratory 1 to implement the $\mathcal{G}(q)$ matrix in a MATLAB function.

Task 2 – Building the non-linear self-balancing robot model in Simulink

From the SIMULINK library browser, perform the following:

1. Choose the *In1* block from the *Sources* section and label it as your input τ.

2. Choose four *Out1* blocks from the *Sinks* section and label them as your states φ, θ, $\dot{\varphi}$, $\dot{\theta}$.

3. Add two *Integrator* blocks from the *Continuous* section. These will be used to integrate the states $\ddot{q}$ to $\dot{q}$ and q, for the output.

4. Use the *Interpreted MATLAB Function* blocks in the *User-Defined Functions* section to use the matrices implemented in Task 1 to compute $\ddot{q}$ from (2).

5. Connect the blocks to solve $\ddot{q}$.

6. Highlight the system, right-click on the selected area, and choose *Create Subsystem from Selection*.

Task 3 – Building the DC motor model in Simulink

1. Choose two *In1* blocks from the *Sources* section and label them as your inputs E_a and $\dot{\varphi}$.

2. Choose two *Out1* blocks from the *Sinks* section and label them as your motor outputs i_a and τ.

3. Add an *Integrator* block from the *Continuous* section. This will be used to integrate the current $\dot{i}_a$ to i_a.

4. Add a *Dead Zone* block from the *Discontinuities* section.

5. Complete the block diagram for the motor model in (3).

Task 4 – Determining better parameters for the MinSeg™ robot

Table 1 contains some placeholder values for a self-balancing robot. However, you should notice that these parameters are substantially different for your MinSeg™ robot.

1. Use some measurement tools (e.g., ruler, scales) to determine better parameter values for your robot and comment on the parameters that are more difficult to determine.
2. Use information on the N20 micro gear motor to determine the parameters of the MinSeg™ motors.

Extension 1 – Comparing the linear and non-linear models

1. Using a MATLAB function or Simulink block, implement the linear model from Laboratory 1.
2. Choose an appropriate input from the *Sources* section and connect it to both the non-linear and linear systems.
3. Connect *Scopes* from the *Sinks* section to compare the linear and non-linear models.

Summary from Laboratory 1

Table 1: Parameters used to describe the free-standing robot and DC motor

Robot Parameter	Values [unit]	Definition
r	0.045 [m]	Radius of the wheels
m	0.4 [kg]	Combined mass of both wheels and motors
J	5×10^{-6} [kg·m^2]	Mass moment of inertia for m about the axle
M	1 [kg]	Mass of the robot chassis
I	1×10^{-3} [kg·m^2]	Mass moment of inertia for M about the COM
l	0.2 [m]	Distance to the center of gravity from the wheel axle
g	9.81 [m·s^{-2}]	Acceleration due to gravity
Motor Parameter	**Values [unit]**	**Definition**
E_a	12 [V]	Motor input voltage
N	50 [-]	Gear ratio
R_a	0.5 [Ω]	Armature resistance
L_a	1.5×10^{-3} [H]	Armature inductance
K_m	0.05 [V/rad/s]	Motor speed constant
η	60 [%]	Gearbox efficiency
τ_m	6 [N·m]	Friction torque

$$\begin{bmatrix} J+(M+m)r^2 & J+(M+m)r^2+Mlr\cos\theta \\ J+(M+m)r^2+Mlr\cos\theta & J+(M+m)r^2+2Mlr\cos\theta+I+Ml^2 \end{bmatrix} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\theta} \end{bmatrix} - Mlr\dot{\theta}\sin\theta \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \end{bmatrix} - \begin{bmatrix} 0 \\ Mgl\sin\theta \end{bmatrix} = \begin{bmatrix} \tau \\ 0 \end{bmatrix}. \quad (2)$$

$$L_a \dot{i}_a + R_a i_a + K_m \dot{\varphi}_{\mathrm{in}} = E_a\,. \quad (3)$$

Acknowledgments

This laboratory was developed by Roxanne R. Jackson, Magnus Axelson-Fisk, and Tom J. Jüstel. ChatGPT (AI-generated text) was used to assist in writing the content of this laboratory.