Advanced Control Systems (ELEN90064)
Project Description: Single-Link Flexible-Joint Robot

Project overview

Throughout this project, you will select sensors and actuators and design controllers for a so-called *single-link flexible-joint robot* with the goal to make the robot reliably meet the performance requirements of various tasks. Figure 1 shows a schematic of the flexible-joint robot, which may at first sight seem to be "too simple" to be a robot, but take a second to think about how many different systems this can represent:

1. The carrier of the reading head of a hard drive;
2. The boom of a horizontal crane;
3. The impeller or paddle of a mixing pot (industrial or domestic scale);
4. The carrier for the end effector of a pick-and-place machine or a painting/cutting machine;
5. A hover board.

The project is broadly split into the following three phases:

Phase 1. Familiarisation with the flexible-joint robotic system and controller design using classical continuous-time methods; with emulation used to achieve a digital control implementation. Phase 1 addresses the content from Weeks 1 to 4 and it is due at the end of Week 5.

Phase 2. Using continuous and discrete-time methods for control systems analysis and design in state space, i.e., state-feedback controller and observer designs. Robust tracking and disturbance rejection strategies, such as integral action and internal model principle, will also be considered. Phase 2 addresses the content from Weeks 5 to 8 and it is due at the end of Week 9.

Phase 3. Optimal control methods, i.e., LQR, LQG and MPC. Phase 3 addresses the content from Weeks 8 to 12 and it is due at the end of Week 14.

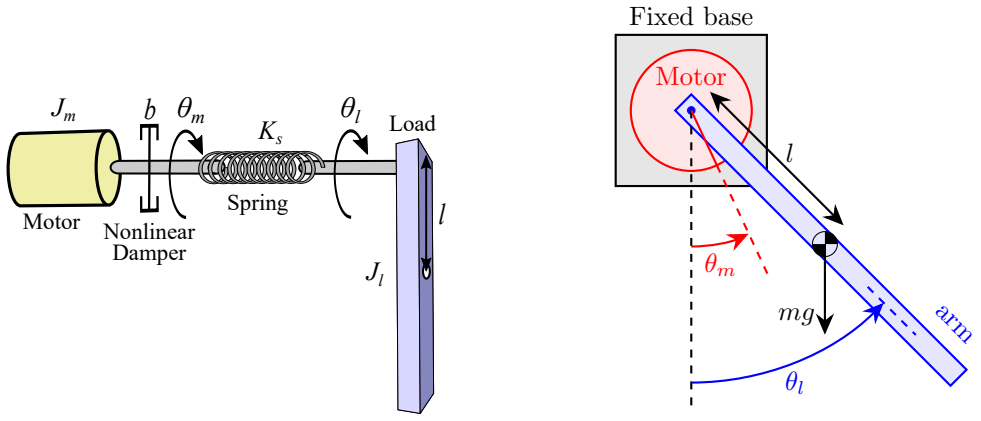


Figure 1: Schematic of the flexible-joint system. The left schematic highlights the spring and damper elements of the flexible connection from the motor to the arm, and the right schematic shows the formal definitions of the system parameters.

Assessment

The assessment for the project is as follows:

1. Report 1: associated with Phase 1. 12 pages limit. 10% of overall mark for the subject.
2. Report 2: associated with Phase 2. 12 pages limit. 10% of overall mark for the subject.
3. Report 3: associated with Phase 3. 12 pages limit. 10% of overall mark for the subject.
4. Oral exam: addresses content from all three phases, i.e., any questions about the project as well as any theory related to it. 15 minutes per student. 20% of overall mark for the subject.

This project manual guides you through numerous tasks for each phase of the project, and all together these tasks follow the steps of the *Control System Design Principles* described below (Section 1).

1 Control System Design Principles

The following steps match the design principles described in [1, Chapter 10]].

Step 1) Understand the process and translate dynamic performance requirements into time, frequency, or pole–zero specifications.

Step 2) Select sensors.

Step 3) Select actuators.

Step 4) Construct a linear model.

Step 5) Try a simple proportional–integral–derivative (PID) or lead–lag design.

Step 6) Evaluate/modify plant.

Step 7) Try state-space and optimal design.

Step 8) Build a (high fidelity) computer model, simulate the performance of the design, and verify whether the performance requirements in **Step 1)** are met.

Step 9) Build a prototype.

Notes:

- These steps do **not** prescribe a sequence that is carried out once; instead, these steps provide a framework that includes iterating back to a previous step as well as skipping a step when justified. The exact sequence of steps followed when carrying out an iterative design cycle is the prerogative of the control engineers. They must decide and justify the sequence of steps based on the results and analyses that are available as the design cycle progresses.
- **Step 8)** needs to be carried out in each phase of the project since this corresponds to simulating the performance of the proposed controllers in the provided (high fidelity) nonlinear model including sensor and actuator.
- **Step 9)** will not be considered in this project.
- For a further discussion on control design principles, refer to Video "Control Design Principles" on Canvas.

2 System Overview, Definitions, and Simulation

A schematic for the single-link flexible-joint robot considered in this project is shown in Figure 1. The key parameters which define the setting and the task specifications are as follows:

θ_m	angular position of the motor relative to the fixed base (rad).
θ_l	angular position of the load (arm) relative to the fixed base (rad).
ω_m	angular speed of the motor, i.e., $\dot{\theta}_m$ (rad/s).
ω_l	angular speed of the load (arm), i.e., $\dot{\theta}_l$ (rad/s).
m	mass of the load.
l	arm length from the center of gravity to the center of rotation.
J_m	moment of inertia of the motor's rotor.
J_l	moment of inertia of the load.
$\overline{K}_s(\cdot)$	spring stiffness as a function of the angular displacement between the arm and the motor.
$\overline{b}(\cdot)$	torsional friction as a function of the motor's angular velocity.
V_a	input voltage of the motor.

Further details for the components of this system are provided in the appendices, namely:

- Appendix A provides details for the plant and motors, and provides the specifications for the various motors that are available to drive the robot.
- Appendix B provides details for the various sensors that are available to obtain feedback of the arm for feedback control and performance assessment.

As the robotic system contains many non-negligible nonlinearities, we provide a Simulink model for simulating the full system as any combination of the plant, a motor, and one (or more) sensor(s). Some parameters of the system **are not revealed** to provide an experience that is analogous to working with a real system. When working with a real system you generally cannot identify all parameters precisely, and manufacturing/calibration tolerances require that your controller performs robustly for all systems produced with the same nominal parameters.

The key plant parameters, already identified through measurement and testing, are the following:

- $m = 0.1$ kg
- $l = 0.1$ m
- $J_l = 0.001$ kg·m^2
- $\overline{K}_s(\theta_m - \theta_l) \approxeq K_s(\theta_m - \theta_l)$ with $K_s \in [1.55, 1.80]$ N·m/rad.
- $\overline{b}(\omega_m) \approxeq \text{sgn}(\omega_m) \cdot b\omega_m^2$ with $b \in [0.013, 0.019]$ N·m/(rad/s)^2, where sgn is the sign function (also called signum function).

2.1 Plant Model

Fig. 1 shows the schematic of the single-link flexible joint robot. The spring models the elasticity of the shaft, belt and gear teeth. We model this spring using a linear model:

$$T_s = K_s(\theta_l - \theta_m) \tag{1}$$

where T_s is the spring torque, K_s is the spring stiffness, and the spring is assumed under tension.

The damper models the friction in the bearings and the friction between the teeth of the gears. We model this damper using a nonlinear viscous friction model:

$$T_b = \text{sgn}(\omega_m) \cdot b\omega_m^2 \tag{2}$$

where T_b is the friction torque, b is the torsional friction coefficient and $\omega_m = \dot{\theta}_m$ is the angular velocity of the motor.

We choose the states as

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} = \begin{bmatrix} \theta_m & \theta_l & \dot{\theta}_m & \dot{\theta}_l \end{bmatrix} \tag{3}$$

Then, the state-space model for the plant dynamics is

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -\frac{K_s}{J_m} & \frac{K_s}{J_m} & 0 & 0\\ \frac{K_s}{J_l} & -\frac{K_s}{J_l} & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} + \begin{bmatrix}0\\ 0\\ -\frac{b\cdot\mathrm{sgn}(x_3)x_3^2}{J_m}\\ -\frac{mgl\sin(x_2)}{J_l}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{1}{J_m}\\ 0\end{bmatrix} T_m \tag{4}$$

where T_m is the output torque of the motor.

2.2 Motor Model

A permanent magnet DC motor is described by the following equations:

$$T_m = K_m I_a \tag{5}$$

$$e = K_m \omega_m \tag{6}$$

where T_m is the output torque of the motor, K_m is the motor torque constant, I_a is the armature (rotor) current, e is the back electromotive force (emf) induced in the armature windings, and ω_m is the angular velocity of the motor.

The electrical model for a permanent magnet DC motor is shown in Fig. 2. Choosing current as the fifth state and the input voltage V_a as the control variable, i.e., $x_5 = I_a$ and $u = V_a$, the equation for the motor dynamics can be written as

$$\dot{x}_5 = -\frac{R_a}{L_a}x_5 - \frac{K_m}{L_a}x_3 + \frac{1}{L_a}u \tag{7}$$

3 Phase 1: Familiarisation and continuous-time control

3.1 Introduction

This phase of the project works through **Step 1)** to **Step 5)** of the control system design principles (see Section 1). The primary goal of this phase is that you take a rigourous design approach to addressing the tasks below. An additional goal of Phase 1 is that you become familiar with the plant, motor, and sensor models, as well as becoming proficient at using the simulation framework provided.

3.2 Performance requirements

The behaviour that the single-link flexible joint robot is required to perform for Phase 1 is motivated by a pick-and-place application and described as the following:

- Design a controller that moves the position of the arm (i.e., θ_l) repeatedly between the 30 degree and 60 degree angular positions.

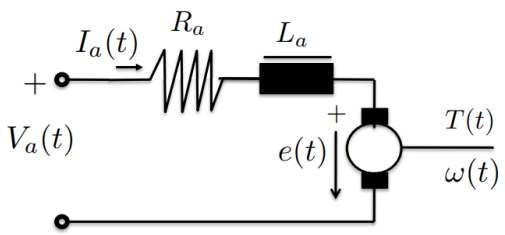


Figure 2: Electrical model for permanent magnet DC motor.

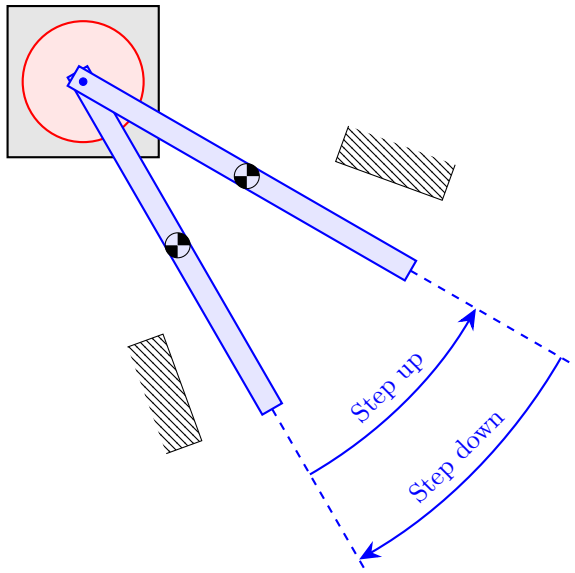


Figure 3: Schematic showing the pick-and-place task for Phase 1, which is to repeatedly step the position of the arm between the two positions shown, i.e., step up from 30 to 60 degrees and step down from 60 to 30 degrees. The hatched sections shown the constraining objects that physically motivate the overshoot specification for this task.

- For both the step up from 30 to 60 degrees and for the step down from 60 to 30 degrees, the controller must achieve the following specifications:
  - Maximum overshoot of at most 5 degrees.
  - Settling time of at most 2.0 seconds (for a less than 2% settling criterion).
  - Steady state error of at most 2 degrees.
  - Maximum controller sampling frequency of 50 Hz.
- **Specifications need to be satisfied on the "real" system provided, i.e., nonlinear model including sensor and actuator, and discrete controller.**

Hint:

- To efficiently test and display both the step up and step down performances, you can use a reference square wave with a 30 degrees offset and an amplitude of 30 degrees.

The combination of settling time and steady state error are critical for the pick-and-place operation because together they specify the accuracy of the end effector that still ensures a fast and reliable pick or place action can be executed.

3.3 Project tasks

Task A: Sensor selection
Select and justify one or more sensors that are relevant for meeting the performance requirements.

Task B: Actuator selection
1. Show that the plant dynamics satisfies the state-space model (4), as described in Section 2.1.

2. Show that the motor dynamics satisfy equation (7), as described in Section 2.2.

Using the above model, determine and justify which of the available motors you expect will be able achieve the performance requirements.

Hints:
For justifying your actuator selection, you might consider:

- Equilibrium analysis of the equations of motions, i.e., required torque for the given steady-state specifications.
- Maximum torque for each of the motors, i.e., considering $\dot{\theta}_m = 0$.
- Simulink models of the motors to identify a steady-state mapping from voltage to torque for each of the motors provided.

Task C: Linear model
Derive a linear system model for the plant and motor combination, which is to be used for controller design in the subsequent tasks. As part of this, provide a justification for your choice of linearisation point. State and justify the value that you use for K_s and b.

Task D: Performance requirements for linear design
Translate the high-level performance requirements given above (Section 3.2) to frequency domain and/or pole-zero specifications. As part of this, show the requirements on a Bode-diagram and/or imaginary plane relative to the open-loop properties of the linear model.

Hints:

- As the high-level performance requirements (overshoot, settling time, steady state error) are given for the nonlinear system with actuator and sensor, you should aim for tighter requirements when designing your controller based on a linear model approximation. Make this difference clear as part of answering this question.
- You may need to revisit your choice of performance requirements once you simulate the controller design on the nonlinear system, which is the final task of this phase.

Task E: Continuous-time controller design
Use frequency domain design techniques to design "simple" controller(s) for the linear system model derived in the previous task. Describe and justify the design process that you follow and design decisions that you make.

You need to make your own simulation for the linear model, e.g., using Simulink or using a Matlab function such as `lsim`. Then, validate your design by simulating your controller on the linear model.

Hints:

- Examples of possible "simple" controller forms include: PID; lead; lag; lead-lag; or a polynomial transfer function.
- Examples of possible design processes include: loop shaping; root locus; pole assignment for the polynomial transfer function controller using the Sylvester matrix.

Task F: Digital emulation of the controller on the linear model
Emulate the continuous-time controller(s) from the previous task using: (a) the zero-order hold method, (b) the Euler method, and (c) the bilinear transform.

Use multiple simulation results to show how the step performance on the linear model changes when the emulation sample time varies, and provide the approximate sample time where the closed-loop behaviour becomes unstable.

For the required sampling frequency of 50 Hz, show the step performance for the combination of spring and damper coefficients that provides the worst performance on the linear model.

Hints:

- To efficiently display the results for multiple sampling times, use one figure to plot multiple results with a legend indicating the sample-time of each results. For example, one figure with multiple step responses, or one figure with multiple Bode diagrams, or one figure with multiple pole locations.

Task G: Evaluate the controller on the nonlinear plant
Use the simulation framework provided, and use the required sampling frequency of 50 Hz for emulation of the controller from the previous task, to evaluate the performance of the controller(s) on the nonlinear plant with motor and sensor.

Use the simulation results to show if and how the step performance differs for the step up (30 to 60 degrees) and step down (60 to 30 degrees).

Hint:

- The motor, plant, and sensor blocks provided include the nonlinearities described in Appendices 2.2 and B. As a result, performance is likely to degrade in comparison to the linear models. Hence, as mentioned in Task D, you may need to revisit your choice of performance requirements once you simulate the controller design on the nonlinear system.

Task H: Evaluate controller performance without motor dynamics
In this task, exclude the motor dynamics from the linear model derived in Task C above and design "simple" controllers for the given specifications. Discuss the changes in the controller parameters and in the tracking performance of closed-loop system.

4 Phase 2: Continuous and Discrete-Time Analysis and Design Methods in State Space

4.1 Introduction

This phase of the project focuses on **Step 7)** of the control system design principles (see Section 1). The primary goal of this phase is that you take a rigorous design approach to applying and contrasting the state-space controller and observer design techniques.

4.2 Performance requirements

To facilitate comparison with the controller performance from Phase 1, the required system behaviour is the same as Phase 1, i.e., the single-link flexible joint robot is required to satisfy the performance requirements given in Phase 1 for overshoot, settling time, steady state, and sampling frequency.

In Task D, an alternative application of tracking a sinusoidal reference is considered, with the performance requirement provided as part of the description of that task.

4.3 Project tasks

Task A: Controllability and observability analysis.
Consider the linear system model for the plant and motor derived in Phase 1 Task C. Evaluate and tabulate the controllability and observability of the system for different selections of motors and sensors. Please note that, if using the *rank* command in MATLAB, you may be required to specify a small tolerance to obtain correct results.

Here, you are required to analyse the observability of the system using each individual sensor and, minimally, five other sensor combinations, e.g., (a) vision and encoder, or (b) potentiometer, tachometer and gyroscope. We also recommend indicating the price ranges for these combinations.

Task B: State-feedback control assuming all states measured
In this task, you need to design controllers using the following two strategies. Note that the design specifications must be satisfied on the nonlinear system including the motor but not including the sensor, i.e., assume there exist sensors (not given in Appendix B) which give perfect measurement of states. Additionally, the sampling frequency of 50 Hz must be used both for emulation and discrete-time design.

B1. Continuous-time controller design
Using the linear system model as per Task A, design a continuous-time full-state feedback controller using the pole placement method.

Discuss and justify the choice of poles for the controller.

In this task, you are asked to demonstrate the controller performance using each of the following: (a) linear model, (b) linear model with emulation (zero-order hold method), (c) nonlinear model, and (d) nonlinear model with emulation (zero-order hold method).

B2. Discrete-time controller design
Using the linear system model as per Task A, design a discrete-time full-state feedback controller using the pole placement method.

In this task, you are asked to demonstrate the controller performance using each of the following: (a) linear model, and (b) nonlinear model.

Continuous to discrete-time poles equivalence

To convert between the s-plane (continuous-time) and the z-plane (discrete-time), the following equation can be used

$$z = e^{sT}$$

where z are the discrete-time poles, s are the continuous-time poles and T is the sampling time. For more information, refer to Section 8.2.3 ("Relationship between s and z") in [1].

Task C: Observer-based feedback control

In this task, based on the observability and cost analysis in Task A, select one or more sensors and design an observer-based feedback controller. In the design process, you are allowed to use one or multiple ideal measurements, i.e., direct output from plant model. However, the specifications must be satisfied using the output of one or more provided sensors.

C1. Continuous-time observer design

Using the linear system model as per Task A, design a continuous-time Luenberger observer for the full state, and combine it with the continuous-time controller from Task B.

Discuss and justify the choice of poles for the observer.

In this task, you are asked to demonstrate the observer-based controller performance using each of the following: (a) linear model, (b) linear model with emulation (zero-order hold method), (c) nonlinear model, and (d) nonlinear model with emulation (zero-order hold method).

For emulation with the observer-based controller(s), you are expected to employ two zero-order hold blocks: one before the observer and one after the controller.

C2. Discrete-time observer design

Using the linear system model as per Task A, design a discrete-time *reduced-order* observer, and combine it with the discrete-time controller from Task B.

In this task, you are asked to demonstrate the observer-based controller performance using each of the following: (a) linear model, and (b) nonlinear model.

Task D: Disturbance rejection and sinusoidal tracking

In this task, you need to use the internal model principle (IMP) to design controllers to satisfy the following requirements:

D1. Constant disturbance rejection

Design and demonstrate a controller that regulates the load angle to 60 degrees and, after reaching steady-state, rejects a constant torque disturbance of 0.05 Nm. Please refer to Appendix A.1 for instructions on how to set up the torque disturbance.

D2. Sinusoidal reference tracking

Design and demonstrate a controller that tracks a sinusoidal reference of 30 degrees amplitude, 30 degrees offset and period of 4 s with zero steady-state error.

In Tasks D1 and D2, the closed-loop performance should be demonstrated on the nonlinear system including the provided blocks for motor(s) and sensor(s). The performance can be demonstrated for either a continuous-time design with emulation or discrete-time design.

As a point of comparison for your IMP designs, you should contrast with the performance of a controller from a previous task (from Phase 1 or Phase 2) for the requirements of Task D1 and D2.

Hint:

- You may use either of the methods discussed in the lectures, i.e., IMP via disturbance estimation or IMP via additional dynamics.

Task E: Robustness analysis
Empirically show how the step up and down performance changes when the parameters of the chosen sensors are varied, for example, the frame rate of the vision sensor, the quantisation resolution of the encoder, or the noise level of the tachometer or gyroscope.

Task F: Alternative sensor selection
In this task, you are required to implement an observer-based controller using one or more different sensors from the previous tasks. Note that any design methods from the previous tasks can be used here.

Discuss the differences in performance and the trade-offs (e.g, cost of the sensor and complexity of installation) associated with alternative sensor selections.

Challenge: as a motivation for sensor and actuator selection, attempt to select a set of sensor(s) and actuator which meet the requirements at the lowest budget. Note that the designs with the lowest costs will be awarded additional marks.

5 Phase 3: Optimal control methods

5.1 Introduction

This phase of the project focuses on **Step 7)** of the control system design principles (see Section 1) using optimal control design techniques.

5.2 Performance requirements

In this phase, you are required to demonstrate tracking performance for the pick-and-place behaviour specified in Phase 1 (step up and step down between 30 and 60 degrees). The settling time, overshoot, steady-state error and sampling frequency requirements remain as defined in Phase 1.

The performance specifications can be satisfied using:

- Continuous-time control with emulation and nonlinear plant and actuator models.
- Discrete-time control with nonlinear plant and actuator models.

5.3 Project tasks

Task A: Linear Quadratic Regulator (LQR) Control
In this task, assume full-state feedback to design LQR controllers using the following two strategies. Note that design specifications must be satisfied by the nonlinear system including the motor but not including the sensor.

A1. Symmetric root locus (SRL) controller design
Using the linear system model for plant plus actuator in the design, i.e, as per Task A, use the symmetric root locus approach to design an LQR controller.

Discuss how the choice of pole locations via the root locus affects the tracking performance of the closed-loop system.

A2. Riccati equation controller design
Using the linear system model for plant plus actuator in the design, i.e, as per Phase 2 Task A, use the Riccati equation approach to design an LQR controller.

Discuss how the SRL design from Task A1 can be used to inform an initial choice for the Q and R matrices.

Discuss also how adjustments to the Q and R matrices affects the tracking performance of the closed-loop system.

A3. Sinusoidal reference tracking
Using an LQR controller and the internal model principle, design and demonstrate a controller that tracks a sinusoidal reference of 30 degrees amplitude, 30 degrees offset and period of 4 s with zero steady-state error.

Task B: Linear Quadratic Gaussian (LQG) Control
In this task, you are given a **maximum budget of $150 for the sensors**.

Using at least two of the available sensors, design an LQG controller to satisfy the required specifications.

Discuss how the choice of covariance matrices for the process and measurement noises, i.e., W and V, affect the tracking performance of the closed-loop system.

Task C: Model Predictive Control (MPC) Control
Using a sampling frequency of 50 Hz, and the Q and R matrices selected in Task A2 as a starting point, implement a reference tracking MPC controller.

Demonstrate and discuss the effect of prediction and control horizons, terminal cost and weighting matrices Q and R on the closed-loop performance.

Add state constraints to the MPC controller to require zero overshoot on the angle of the link θ_l and demonstrate the performance. Observe the status of the quadratic programming (QP) solver and discuss how the solve time and exit flag behave with and without constraints.

Hint:

- Use the guide provided to assist with implementation of the MPC block in Simulink.
- Terminal costs to consider are: the solution of the Riccati equation; the Q matrix; zero.
- Performance must be demonstrated using an observer to provide the state estimate feedback to the MPC Controller block in Simulink. You may find it useful while preparing your MPC Controller block to use perfect state feedback in place of an observer estimate.

References

[1] Gene F Franklin, J David Powell, and Abbas F Emami-Naeini. *Feedback control of dynamic systems*. Pearson London, 2015.

A Plant and Actuator: Model and Specifications

A.1 Simulink Blocks for Motor and Motor/Plant System

You are provided with three simulink blocks that model each of the three motors described in Section A.2, as shown in Fig. 4.

For each motor block, the input/output ports are:

- One input port for input voltage of the motor V_a.
- One input port for the motor angular velocity ω_m.
- One output port for the motor torque T_m.

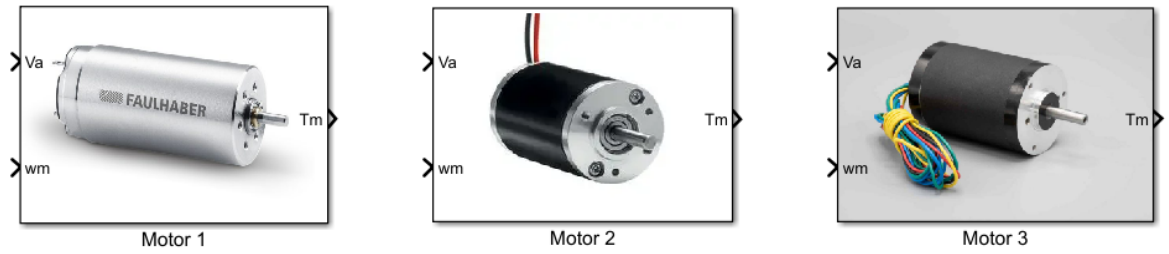


Figure 4: Simulink blocks for each of the three motors described in Section A.2.

Each of the motors correspond to a respective full system model for the robot which includes the dynamics of both the plant and actuator, i.e., Motor i corresponds to Flexible-joint robot i. You are provided with three Simulink blocks that model each of the corresponding robots, as shown in Fig. 5.

For each robot block, the input/output ports are:

- One input port for input voltage of the motor V_a.
- One output port for the state vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T = \begin{bmatrix} \theta_m & \theta_l & \dot{\theta}_m & \dot{\theta}_l & I_a \end{bmatrix}^T$.

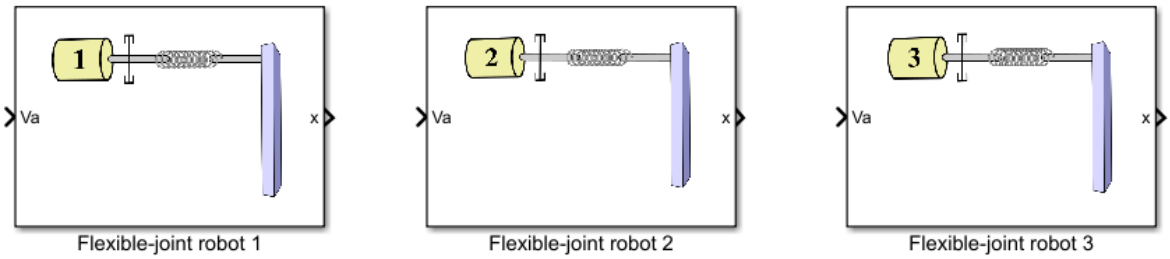


Figure 5: Simulink blocks for the single-link, flexible-joint robots. These blocks include the nonlinear plant and motor models for each of the motors described in Section A.2.

Catalogues: To get you started with the motor and robot blocks, we provide you with the Simulink files:

- `Motors_Catalogue.slx` - this file contains all three motor blocks, as shown in Fig. 4
- `FlexibleJointRobots_Catalogue.slx` - this file contains all three robot blocks (including both the motor and plant dynamics), as shown in Fig. 5

Protected model (`*.slxp`): In order to actually run the motor and robot models, you need to have the following files on the Matlab path:

- `Motor_i.slxp` - models for each of the three motors.
- `Flexible_joint_robot_i.slxp` - models for all three robot blocks (including both the motor and plant dynamics).

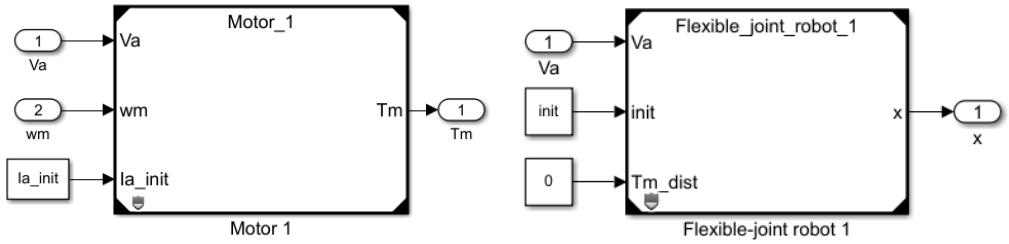
Adding to the Matlab path: You can add these motor and robot models to the Matlab path by either:

- Having the respective `slxp` files in the directory you are currently working in in Matlab.
- Right-clicking on a folder that contains the `slxp` files and selecting "Add to Path > Selected Folders"

Adjusting parameters: Before using the Simulink blocks described above, you need to set up initial conditions for the motor and plant. You adjust these parameters by editing the Matlab script named:
`ELEN90064_Project_Parameters.m`
This script should be run before the simulation is run.

Adding to your own Simulink model: Simply copy the motor or robot blocks from the Catalogue files into your own Simulink model, i.e., copy any block with an image on it in Figs. 4 or 5. Each of these blocks is a subsystem, and when you double-click these blocks you see internals similar to Fig. 6:

- The initial conditions for the motor (`Ia_init`) and for the full model of the robot (`init`) are defined in the parameter script described above (i.e., in `ELEN90064_Project_Parameters.m`).
- The large blocks are model references to the appropriate protected models (e.g., `Motor_1.slxp` and `Flexible_joint_robot_1.slxp`).
- For Phase 2, Task D1 (Constant disturbance rejection), you will also need to set up the constant torque disturbance of 0.05 N·m in your Simulink files. This is achieved by replacing the constant (zero) at the input `Tm_dist` with an appropriate block generating the disturbance after reaching steady-state (e.g., step or pulse generator blocks). Please ensure `Tm_dist` is set to zero (as shown in the figure) for all other tasks.



(a) Internal view of motor block in Fig. 4. (b) Internal view of robot block in Fig. 5.

Figure 6: Internal view of Simulink blocks indicating the initial conditions edited via `ELEN90064_Project_Parameters.m` and input torque disturbance for the robot.

A.2 Motor Specifications

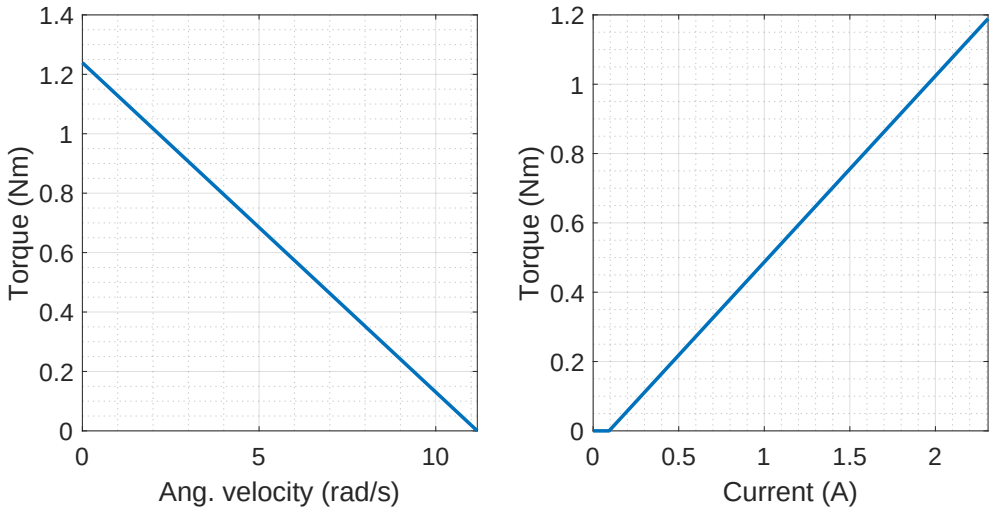
Three DC motors are available for actuator selection in this project. The specifications for these motors are provided in Tables 1, 2 and 3. The respective torque-velocity and torque-current characteristics are shown in Figs. 7, 8 and 9.

Motor 1:

- Manufacturer

Table 1: DC Motor 1

Characteristic	Value	Unit
Nominal voltage	6	V
Starting torque (T_{max})	1.239	N·m
Permissible peak current (I_{max})	2.5	A
Speed at no-load	11.175	rad/s
Current at no-load	0.1	A
Motor inertia	0.0021	kg·m^2
Motor torque constant	0.5369	V/(rad/s)
Resistance	2.6	Ω
Inductance	0.18	mH
Gear play	0.03	deg
Service life	10,000	h
Cost	197.45	AUD



(a) Torque vs velocity characteristics.

(b) Torque vs current characteristics.

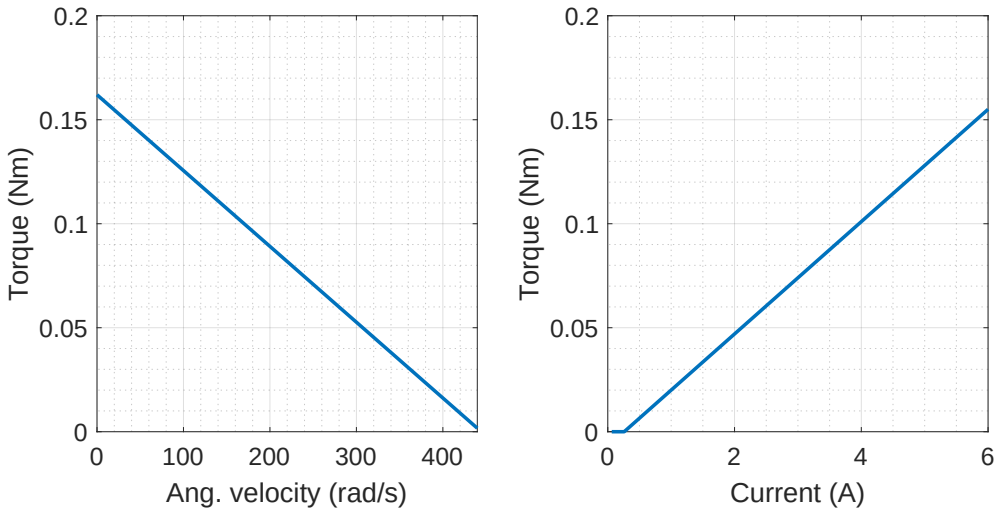
Figure 7: DC Motor 1.

Motor 2:

- Manufacturer

Table 2: DC Motor 2

Characteristic	Value	Unit
Nominal voltage	12	V
Starting torque (T_{max})	0.162	N·m
Permissible peak current (I_{max})	6.1	A
Speed at no-load	439.823	rad/s
Current at no-load	0.26	A
Motor inertia	7.5e-6	kg·m^2
Motor torque constant	27	mV/(rad/s)
Resistance	2	Ω
Inductance	1.3	mH
Gear play	0.01	deg
Service life	4,000	h
Cost	244.61	AUD



(a) Torque vs velocity characteristics.

(b) Torque vs current characteristics.

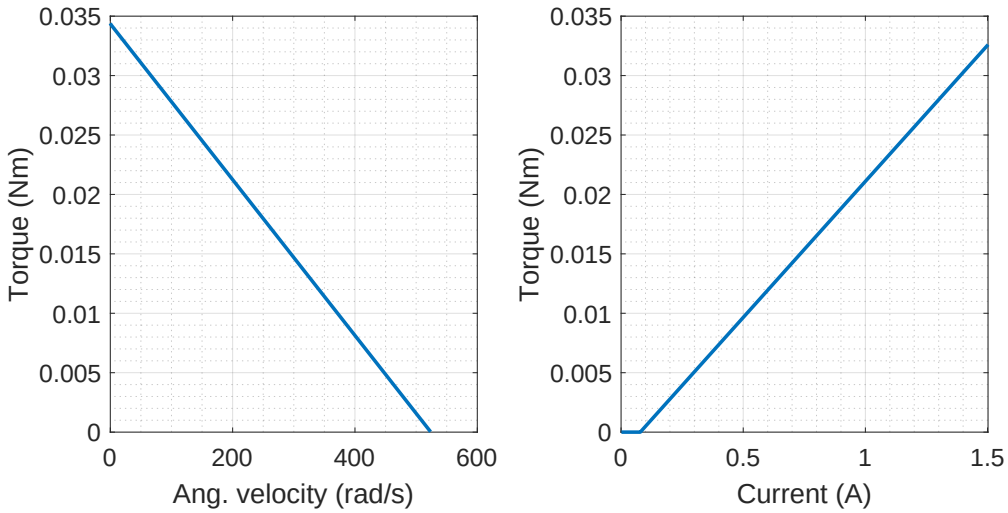
Figure 8: DC Motor 2.

Motor 3:

- Manufacturer

Table 3: DC Motor 3

Characteristic	Value	Unit
Nominal voltage	12	V
Starting torque (T_{max})	34.4	mN·m
Permissible peak current (I_{max})	1.5	A
Speed at no-load	523.599	rad/s
Current at no-load	0.08	A
Motor inertia	54e-6	kg·m^2
Motor torque constant	0.0229	V/(rad/s)
Resistance	8	Ω
Inductance	10	mH
Gear play	0.08	deg
Service life	2,000	h
Cost	46.76	AUD



(a) Torque vs velocity characteristics.

(b) Torque vs current characteristics.

Figure 9: DC Motor 3.

B Sensor Specifications

B.1 Overview

The following 5 sensors are available for selection in this project:

- **Vision:** This sensor provides a direct measurement of the link's angular displacement θ_l based on a camera that tracks a point on the moving link relative to the fixed base.
- **Absolute encoder:** This sensor provides a direct measurement of the motor's angular displacement θ_m in quantised increments.
- **Potentiometer:** This sensor provides a direct measurement of the motor's angular displacement θ_m in based on a potentiometer that changes resistance as a function of angle.
- **Tachometer:** This sensor provides a direct measurement of the motor's angular velocity $\dot{\theta}_m$ based on a quantised measurements of angular displacement from an incremental encoder; the sensor output is the angular displacement measured divided by the time between measurements.
- **Gyroscope:** This sensor provides a direct measurement of the link's angular velocity $\dot{\theta}_l$ based on a MEMS (micro electrical mechanical system) rate gyroscope attached to the link.

Section B.2 describes the Simulink blocks provided for modelling each sensor. Sections B.3-B.7 provide information about the cost, function, parameters, and indicative behaviour of each sensor. All sensors are subject to calibration errors, i.e., a constant disturbance that is the difference between the true zero measurement of the system and the zero measurement of the sensor. The following table summarises some of the strengths and weaknesses of each sensor.

Sensor	Strengths	Weaknesses
Vision	- Gives a direct measurement of the link's angular displacement.	- A low cost variant generally has a long sample time and high computational load relative to other sensors. - Must be mounted external to the flexible-joint robot device and hence increases the overall volume required for operation and can be subject to vibration and misalignment disturbances.
Absolute encoder	- Gives a direct measurement of the motor's output shaft. - Low latency and can be sampled at essentially any rate the microcontroller it is connected to can handle. - A calibration to the zero position should remain valid for a long service duration.	- Provides no information about the actual angular position within its quantisation resolution. - Provides no information about the angle of the link relative to the motor.
Potentiometer	- Gives a direct measurement of the motor's output shaft. - Low cost.	- A low cost variant generally has lower tolerance. - Needs to be re-calibrated semi-frequently due to changing resistance characteristics.
Tachometer	- Can be more accurate than estimating velocity based on the absolute encoder or potentiometer measurements.	- Dividing angular displacement by the time between measurements has a trade-off of between latency and noise level.
Gyroscope	- Sensor is designed and developed for angular velocity measurements - High sampling rates possible without loss of accuracy.	- Must be mounted on the link with wires connected back to the fixed base. - Measurements are subject to drift over a relatively short time frame, i.e., the measurement from a stationary gyroscope increases (or decreases) at a constant rate over a relatively short time frame.

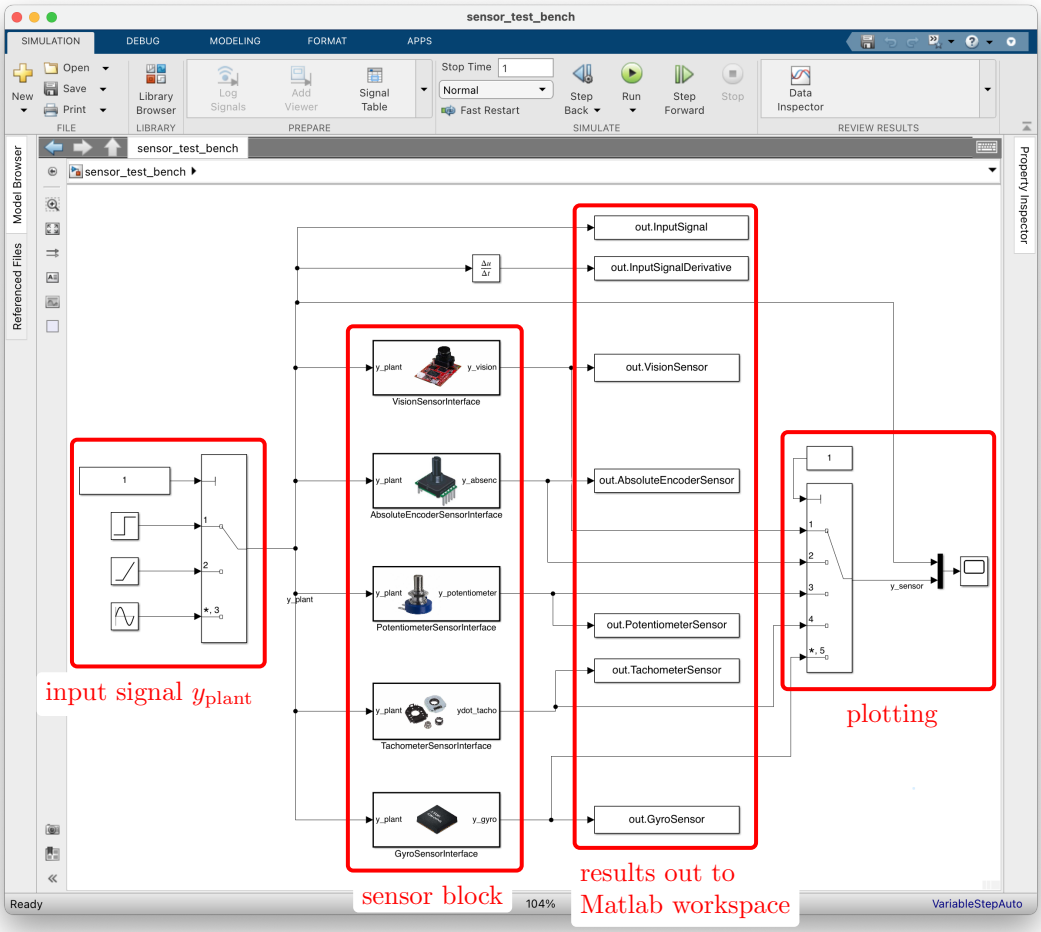


Figure 10: Screenshot of the "sensor test bench" Simulink model that is provided. The model includes: the 5 sensor block provided (in the center with icons on the blocks), an example of one option for quickly switching between different input signal (left), an example of one option for quickly switching between which sensor output is displayed on the scope (right), and an example of writing the simulation results out to the Matlab workspace (the `out.*` blocks).

B.2 Simulink Blocks and Sensor Test Bench

Based on the models described in Sections B.3-B.7 you are provided with 5 Simulink blocks that model each of the sensors.

Inputs/outputs: Each sensor block has:

- One input port that accepts a scalar input signal, which should be connected to the true value from the plant (i.e., θ_m, θ_l, $\dot{\theta}_m$, $\dot{\theta}_l$).
- One output port, which is a scalar signal that represents the modelled sensor output.

Test bench: To get you started with the sensor blocks, we provide you with the Simulink file:

- `sensor_test_bench.slx` - this file contains all 5 sensor blocks, already connected to an source signal, displaying the sensor output on a scope, and saving the data out to the Matlab workspace. Figure 10 shows a screenshot of the sensor test bench model.

Protected model (`*.slxp`): In order to actually run the sensor models, you need to have the following 5 files on the Matlab path:

- `VisionSensor.slxp` - models the vision sensor, see Section B.3.
- `EncoderSensor.slxp` - models the absolute encoder sensor, see Section B.4.
- `PotentiometerSensor.slxp` - models the potentiometer sensor, see Section B.5.
- `TachoSensor.slxp` - models the tachometer, see Section B.6.
- `GyroSensor.slxp` - models the gyroscope sensor, see Section B.7.

Adding to the Matlab path: You can add these sensor models to the Matlab path by either:

- Having the 5 `slxp` sensor files in the directory you are currently working in in Matlab.
- Right-clicking on a folder that contains the `slxp` files and selecting "Add to Path > Selected Folders"
- Using a Matlab command equivalent of the previous point: `addpath /path/to/sensor/blocks/`

Adjusting parameters: As per the tables in Sections B.3-B.7, each sensor has multiple options that trade-off cost against performance in some fashion. You adjust these parameters by editing the Matlab script named:
`ELEN90064_Project_Parameters.m`
This script should be run before the simulation is run. Sections Sections B.3-B.7 describe each parameter in the script.

Adding to your own Simulink model: Simply copy the sensor block from the `sensor_test_bench.slx` file into your own Simulink model (i.e., copy any block with an image on it in Figure 10). Each of these blocks is an atomic subsystem, and when you double-click on a block you see internals similar to Figure 11:

- The constants on the left (i.e., `vision_fps` and `theta_l_init` in the Figure 11 example) are defined in the parameter script described above (i.e., in `ELEN90064_Project_Parameters.m`).
- The large block is a model reference to the appropriate protected model (i.e., a reference to `VisionSensor.slxp` in the Figure 11 example).

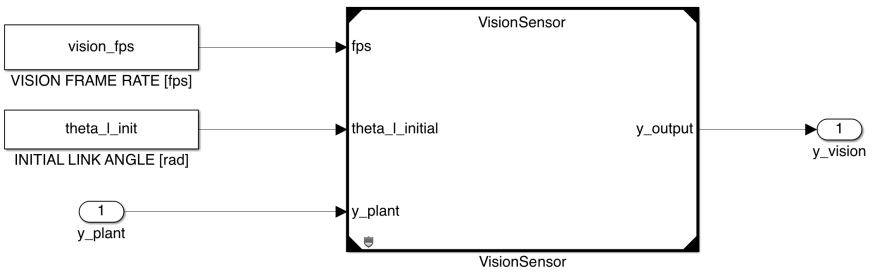
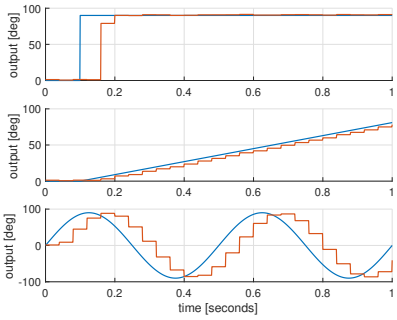


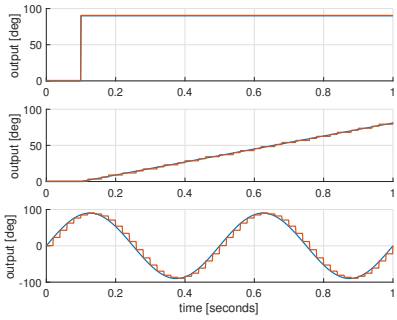
Figure 11: Screenshot of the internals of the "Vision Sensor Interface" shown in Figure 10.

Usage and interpretation of the test bench:

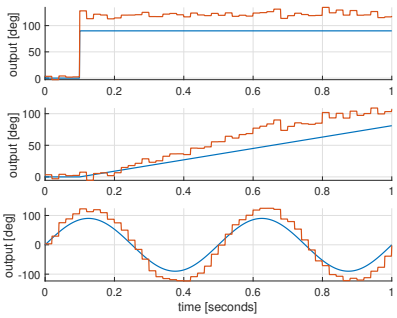
- The "sensor test bench" Simulink model mimics the case that you test the sensor separate of it being installed on the plant of interest (i.e., separate of the single-link flexible-joint robot). Such "test bench" experiments are typically performed to determine the performance characteristics of the sensors so that the influence of these characteristics on the overall system can be modelled and accounted for during design.
- You are encouraged to adjust and utilise the sensor test bench in any way that you see fit to gain the data you need for understanding the sensor characteristics.
- You can assume that you can generate and inject any precisely know signal into the Simulink sensor blocks. Keep in mind that such testing would typically require a costly test bench to physically realise such experiments. Test bench costs are not considered in this project.



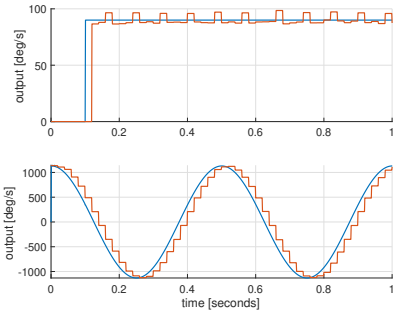
(a) Response of the **vision sensor** output from the sensor test bench, with $f = 25$ [fps] and $k_p = 0.0042$ [m/pixel].



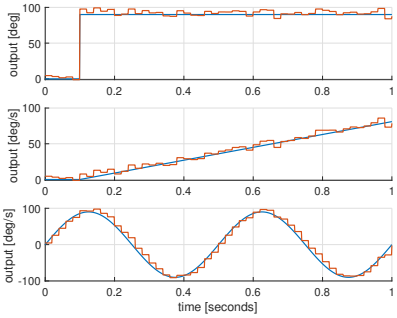
(b) Response of the **absolute encoder sensor** output from the sensor test bench, with $l_{\text{absenc}} = 128$ and 50 [Hz] sampling frequency.



(c) Response of the **potentiometer sensor** output from the sensor test bench, with $l_{\text{pot}} = 20$ [percent] and 50 [Hz] sampling frequency.



(d) Response of the **tachometer sensor** output from the sensor test bench, with $l_{\text{tacho}} = 2048$ [counts per shaft revolution], $k_{\text{tacho}} = 1\%$, and 50 [Hz] sampling frequency.



(e) Response of the **gyroscope sensor** output from the sensor test bench, with $\sigma_{\text{additive}} = 0.5$ [degrees/sec], $\sigma_{\text{drift}} = 5$ [degrees/sec^2], and 50 [Hz] sampling frequency.

Figure 12: Response produced by the sensor labelled in the caption. In all figures, blue is the input y_{plant} and red is the output from the respective sensor (y_{vision}, y_{absenc}, y_{pot}, y_{gyro}). As the tachometer sensor outputs the derivative of its input (i.e., y_{plant} in, $\dot{y}_{\text{tacho}}$ out), the blue line for Figure 12d is the derivative of the input y_{plant}. The notation used in the captions is defined in the section for that sensor.

Example test bench experiments: Figure 12 shows an example contrasting the behaviour of each of the 5 sensor blocks, where the results were produced using the "sensor test bench" Simulink model. Note that a Matlab script was used to iteratively run the sensor test bench for various configurations, storing the data of each run, and then plot the results show in Figure 12. The notation used in the figure captions is defined in Sections B.3-B.7 for each sensor respectively.

B.3 Vision sensor details

Description

The following are the key parameters for specifying the vision sensor:

- f frame rate, in units of [fps, framer per second], which is the time between image being available for processing. This parameter influences the elongation on a moving object in each image captured due to pixel accumulation occurring over the full sample time of the frame rate. Elongation means that the centre of the object as determined by image processing is delayed relative to the true position.
- k_p pixel resolution on the 2D plane of interest, in units of [meters/pixel]. This parameter is directly related to the resolution of the camera's sensor (e.g., image size of 320x240 pixel) and the distance of the camera from the 2D plane of interest. This parameter introduces the noise into the centre of the object as determined by the image processing algorithm.

Figure 13 shows how these aspects can be combined to form an idealised model of a vision sensor.

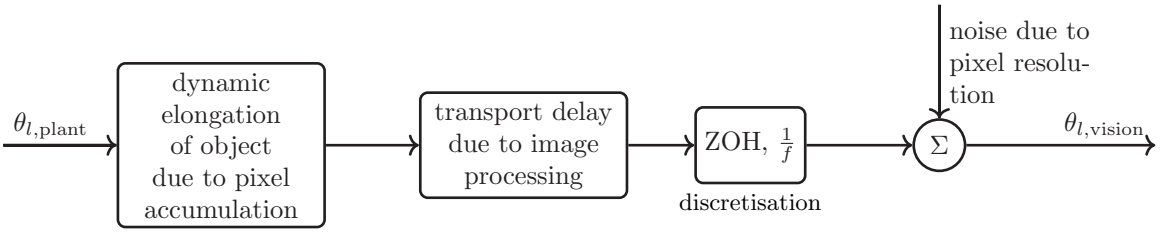


Figure 13: Block diagram showing an idealised model of a vision sensor

Cost

The following table summarises the vision sensors available for selection in this project. These cameras have the benefit of supporting object detection algorithm being run on the camera's microcontroller, hence the image processing computational load does not burden the other microcontroller in the overall flexible-joint robot.

Model	Cost (AUD)	Frame Rate	Resolution	Website
Raspberry Pi Camera Module 2 (IMX219)	$39	25 FPS	320x240	Manufacture Reseller
OpenMV Cam H7 R2 (MT9M114)	$135	50 FPS	320x240	Manufacture Reseller
Allied Vision Alvium 1800 U-050	$416	100 FPS	320x240	Manufacture Reseller

Simulink block

The input and output signal of the vision sensor block are:

- **Input:** scalar signal that is the true angular position of the link.
- **Output:** scalar signal that is the vision sensor measurement of the angular position of the link.

The parameters that need to be defined via the `sensor_parameters.m` script are:

- `vision_fps` – This is the value of f as described above. **Note: the allowed values for fps are: 25, 50, 100. Any other value used defaults to 25 fps.**
- `theta_l_init` – This is the initial position of the link θ_l in units of radians. This parameter ensures that the initial output of the sensor agrees with the initial condition of the plant.

Note that $k_p = 0.0042$ [m/pixel] is fixed for the Simulink block provided. Figure 12a shows an example of the input-output behaviour as produced using the "sensor test bench" Simulink model.

B.4 Absolute encoder sensor details

Description

The following is the key parameter for specifying the absolute encoder sensor:

- l_{absenc} the quantisation interval of the absolute encoder, in units of [radians/count] as determined by the counts per revolution specification. As the precise angular position of the link within the quantisation interval is not known, calibration of the "zero" position can have a constant offset from the true zero position.

Figure 14 shows how this parameter can be combined to form an idealised model of a absolute encoder sensor.

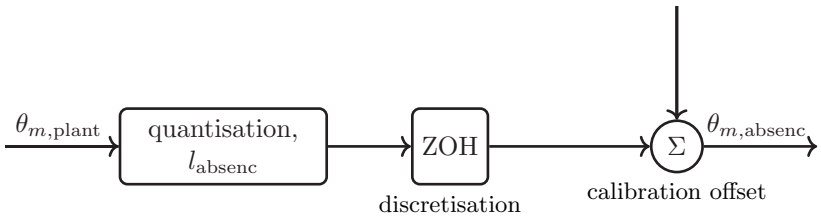


Figure 14: Block diagram showing an idealised model of an absolute encoder sensor

Cost

The following table summarises the absolute encoder sensors available for selection in this project.

Model	Cost (AUD)	Counts per revolution	Website
Panasonic (EVQ-V9C00116E)	$4	16	Reseller
Bourns (EAW0J-B24-AE0128L)	$15	128	Reseller
CUI Devices (AMT203-V)	$70	4096	Reseller

Simulink block

The input and output signal of the absolute encoder sensor block are:

- **Input:** scalar signal that is the true angular position of the motor.
- **Output:** scalar signal that is the absolute encoder sensor measurement of the angular position of the motor.

The parameters that need to be defined via the `sensor_parameters.m` script are:

- `absolute_encoder_counts_per_revolution` – This is the counts per revolution as given in the table above. Any value greater than zero may be used.

Figure 12b shows an example of the input-output behaviour as produced using the "sensor test bench" Simulink model.

B.5 Potentiometer sensor details

Description

The following is the key parameter for specifying the potentiometer sensor:

- l_{pot} the linearity bounds of the potentiometer [percent] as per the datasheet. This describes how far the static nonlinear mapping from input to output can differ from the an ideal linear mapping. The percentage is the relative to the value at the mid-point of the potentiometer's range.

Figure 15 shows how these aspects can be combined to form an idealised model of a Potentiometer sensor.

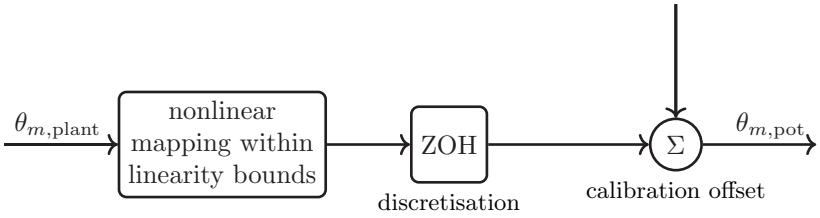


Figure 15: Block diagram of the potentiometer model

Cost

The following table summarises the Potentiometer sensors available for selection in this project.

Model	Cost (AUD)	Linearity	Website
Sparkfun	$3	20%	Reseller
Bourns (3852A-282-103AL)	$19	10%	Reseller
Bourns (3590S-2-202L)	$25	5%	Reseller
Bourns (3549S-1AA-202A)	$34	3%	Reseller
Nidec Copal (M-22S10 1K)	$34	0.2%	Reseller

Simulink block

The input and output signal of the Potentiometer sensor block are:

- **Input:** scalar signal that is the true angular position of the motor.
- **Output:** scalar signal that is the Potentiometer sensor measurement of the angular position of the motor.

The parameters that need to be defined via the `sensor_parameters.m` script are:

- `potentiometer_linearity_percent` – This is the value of l_{pot} in units of percent, as described above and as given in the table. Any value greater than or equal to zero may be entered.

Figure 12c shows an example of the input-output behaviour as produced using the "sensor test bench" Simulink model.

B.6 Tachometer sensor details

Description

The following are the key parameters for specifying the tachometer sensor:

- l_{tacho} the quantisation interval of the tachometer encoder [rad/count] as determined by the counts per revolution specification and the gearbox ratio.
- ΔT_{tacho} - sampling time
- k_{tacho} the standard deviation of the multiplicative uncertainty.

Figure 16 shows how these aspects can be combined to form an idealised model of a tachometer sensor.

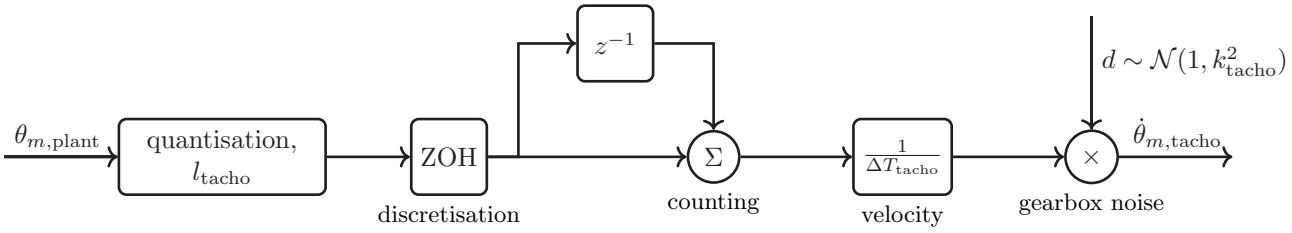


Figure 16: Block diagram of the tachomater encoder model

Cost

The following table summarises the tachometer sensors available for selection in this project.

Model	Cost (AUD)	Counts per revolution	Website
Pololu (2590)	\$17	20	Reseller
DFRobot (SEN0230)	\$40	400	Reseller
CUI Devices (AMT103-2048-N4000-S)	\$32	2048	Reseller

Simulink block

The input and output signal of the tachometer sensor block are:

- **Input:** scalar signal that is the true angle of the motor.
- **Output:** scalar signal that is the tachometer sensor measurement of the angular velocity of the motor.

NOTE: The input units differ from the output units of the tachometer sensor. The input is angle in [radians] while the output is angular velocity in [radians/second].

The parameters that need to be defined via the `sensor_parameters.m` script are:

- `tachometer_counts_per_revolution` – This is the counts per revolution as given in the table above. Any value greater than zero may be used.
- `tachometer_output_frequency` – This is the inverse of ΔT_{tacho} from above and hence it is the frequency at which velocity estimates are computed and output from the sensor block. **Note: the allowed values for frequency are: 10, 25, 50, 100. Any other value used defaults to 50 Hz.**
- `theta_m_init` – This is the initial position of the motor θ_m in units of radians.
- `thetadot_m_init` – This is the initial angular velocity of the motor $\dot{\theta}_m$ in units of radians/sec. Together with the previous parameter, these ensure that the initial output of the sensor agrees with the initial condition of the plant.

Figure 12d shows an example of the input-output behaviour as produced using the "sensor test bench" Simulink model.

B.7 Gyroscope sensor details

Description

This MEMS rate gyroscope sensors would be mounted on the link and hence measures the angular speed of of the link. A widely-used continuous-time rate gyroscope model is the following with additive noise and drift:

$$\omega_{\text{gyro}} = \omega_{\text{plant}} + \beta + \eta_{\text{additive}} \tag{8a}$$

$$\dot{\beta} = \eta_{\text{drift}} \tag{8b}$$

where ω_{gyro} is the rate gyroscope measurement, ω_{plant} is the true angular velocity of the plant, β is the drift term, and η_{additive}, η_{drift} are independent zero-mean Gaussian white-noise processes with respective variances $\sigma^2_{\text{additive}}$ and σ^2_{drift}. Hence, the following are the key parameters for specifying the gyroscope sensor:

- σ_{additive} the standard deviation of the additive noise on the output of the rate gyroscope.
- σ_{drift} the standard deviation of the noise driving the drift equation.

Figure 17 shows how these aspects can be combined to form an idealised model of a gyroscope sensor.

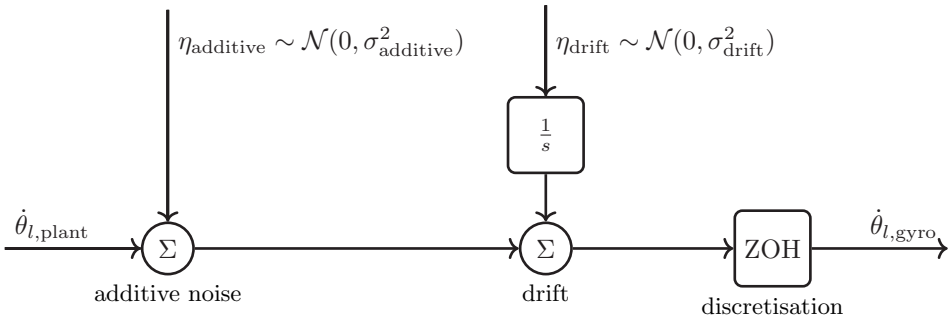


Figure 17: Block diagram of the gyroscope model

Cost

The following table summarises the gyroscope sensors available for selection in this project.

Model	Cost (AUD)	σ_{additive} [degree/s]	σ_{drift} [degrees/s^2]	Website
Invensense (ICM-20948)	\$30	0.5	5.0	Breakout Board Reseller
STM (LSM9DS1)	\$28	0.5	10.0	Breakout Board Reseller

Simulink block

The input and output signal of the gyroscope sensor block are:

- **Input:** scalar signal that is the true angular velocity of the link.
- **Output:** scalar signal that is the gyroscope sensor measurement of the angular velocity of the link.

The parameters that need to be defined via the `sensor_parameters.m` script are:

- `gyro_sigma_drift` – This is the value of σ_{drift} as described above and given in the table, i.e., it is the standard deviation of the drift noise in units of radians/sec^2. Any value greater than or equal to zero may be used.

Figure 12e shows an example of the input-output behaviour as produced using the "sensor test bench" Simulink model.

Version Control

July 17, 2022: Paul Beuchat, Matheus Xavier, Dragan Nesic