AMME5520 Assignment 1

Nic Barbara & Ian Manchester

Due date: Sunday 16th of April, 11:59pm, 2023

The following questions explore motion planning via the material from weeks 1-6 (Question 1 on weeks 1-3, Question 2 on weeks 4-6). You will submit a report through TurnItIn justifying the design decisions you have made and exploring their consequences. All matlab code must also be submitted through TurnItIn.

While you are encouraged to discuss approaches with tutors and your fellow students, all code, analysis and submitted writing must be entirely your own work.

1 Question 1: Path Planning [45 marks]

The task is to design a path for a space probe to inspect various sites along an abandoned space station in deep space. The space station can be modelled as a cylinder with two rectangular solar arrays attached. The probe can be modelled as a cube. Dimensions and details are provided in Table 1. A spherical bounding surface for the probe is used to check for collisions between the two spacecraft. We have provided two Matlab functions to help with this:

1. `check_collision()`: a function to check if the bounding sphere and space station intersect.

2. `plot_scene()`: a function to make a plot like Figure 1.

1.1 Your task

Your task is to design a path planning algorithm that constructs a path for the robot to visit a series of locations while avoiding collisions at all times. The robot must inspect: top and bottom of both solar panels and the cylinder, as well as the ends of the cylinder.

The robot must come within 5m of the space station to take images with sufficient resolution. Assume that it can capture a 5m x 5m region with each image.

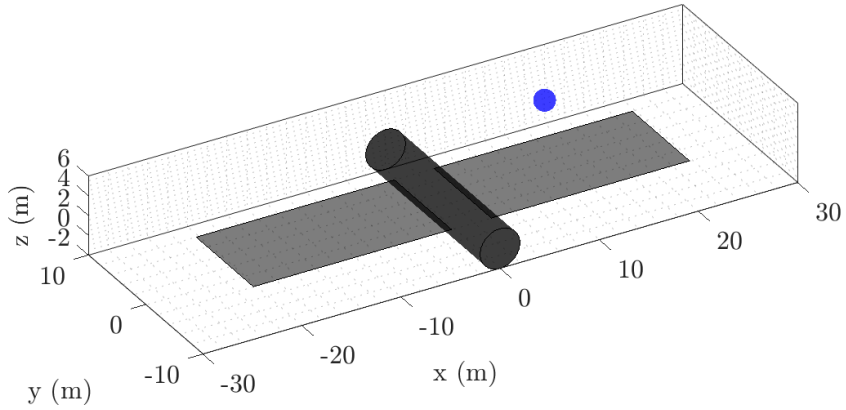

Figure 1: Spacecraft (blue) inspecting abandoned space station. The spacecraft turns red if a collision is detected.

We suggest investigating probabilistic roadmaps (PRMs) and Dijkstra's algorithm or $A^\star$. If you use $A^\star$, justify your choice of heuristic and compare its performance to Dijkstra's algorithm.

You are free to use basic Matlab functions and your code from Matlab Grader quizzes. You may not use any packages that implement Dijkstra, $A^\star$, PRMs, or similar graph-based motion planning algorithms. Basic Matlab functions are considered to be those found in the documentation (type `doc` in the command window) under the following headings: Language fundamentals, Graphics, Programming Scripts and Functions, and under Mathematics the following subheadings: Elementary Math, and Linear Algebra.

2 Question 2: Control Design [45 marks]

Define a "world" frame as the frame of reference with the space station centred at the origin. Let $\mathbf{x} = (x, y, z)^\top$ be the position of the probe in the world frame, and let $\mathbf{v} = (v_x, v_y, v_z)^\top$ be its velocity in the body frame. Write $\bar{\mathbf{q}} = (q_0, \mathbf{q}^\top)^\top$ for the quaternion describing the orientation of the probe's body frame, where $\mathbf{q} = (q_1, q_2, q_y)^\top$ is the vector component of the quaternion. Its angular rates are $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^\top$, also in the body frame. The control inputs to the system are $\mathbf{F} = (F_x, F_y, F_z)^\top$, the thrust vector from the probe's three main thruster aligned with each axis of the body frame, and $\mathbf{T} = (T_x, T_y, T_z)^\top$, the applied torque from three pairs of de-coupled attitude thrusters which are arranged to provide no net force.

The equations of motion for a 6 degree of freedom rigid body in zero-g are as follows:

$$\dot{\mathbf{x}} = C_{\text{wb}}\mathbf{v} \tag{1}$$
$$\dot{\mathbf{v}} = -\boldsymbol{\omega} \times \mathbf{v} + \mathbf{F}/m \tag{2}$$
$$\dot{q_0} = -\frac{1}{2}\boldsymbol{\omega}^\top \mathbf{q} \tag{3}$$
$$\dot{\mathbf{q}} = -\frac{1}{2}\boldsymbol{\omega} \times \mathbf{q} + \frac{1}{2}q_0\boldsymbol{\omega} \tag{4}$$
$$I\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times (I\boldsymbol{\omega}) + \mathbf{T} \tag{5}$$

where m is the probe's mass (assumed constant) and I is its inertia matrix. For those interested, the attitude kinematics can be written very neatly as a quaternion product

$$\dot{\bar{\mathbf{q}}} = \frac{1}{2}\bar{\mathbf{q}} \circ \bar{\boldsymbol{\omega}}$$

where $\bar{\boldsymbol{\omega}} = (0, \boldsymbol{\omega}^\top)^\top$. The coordinate transform C_{wb} from the body frame to the world frame is a function of the attitude quaternion, and is given by

$$C_{\text{wb}} = \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{pmatrix}^\top \tag{6}$$

where

$$l_1 = q_0^2 + q_1^2 - q_2^2 - q_3^2 \qquad m_1 = 2(q_1q_2 - q_0q_3) \qquad n_1 = 2(q_0q_2 + q_1q_3) \tag{7}$$
$$l_2 = 2(q_1q_2 + q_0q_3) \qquad m_2 = q_0^2 - q_1^2 + q_2^2 - q_3^2 \qquad n_2 = 2(q_2q_3 - q_0q_1) \tag{8}$$
$$l_3 = 2(q_1q_3 - q_0q_2) \qquad m_3 = 2(q_2q_3 + q_0q_1) \qquad n_3 = q_0^2 - q_1^2 - q_2^2 + q_3^2. \tag{9}$$

Probe mass	50 kg
Probe size	1 m×1 m×1 m
Thrust limit F_z	100N
Thrust limit F_x, F_y	20N
Torque limits T_x, T_y, T_z	2Nm

Table 1: Properties of the space probe.

These equations may look scary, but Matlab is your friend. You can linearise the equations of motion using symbolic variables and the `jacobian()` function. We have provided the following set of functions:

1. `spacecraft_dynamics()`: an implementation of the equations of motion

2. `nonlinear_dynamics()`: also an implementation of the equations of motion, but treating $q_0 = \sqrt{1 - \mathbf{q}^\top \mathbf{q}}$ separately. You should use this function alongside `jacobian()` to linearise the system. It avoids issues with controllability of the quaternion kinematics.

3. `performance_objective()`: a function to compute $z = c(x)$, where z is the performance objective to minimise in your cost function. This converts the quaternion components of the state vector to Euler angles so they are more meaningful in control design.

4. `my_eul2quat(), my_quat2eul(), my_quat2dcm()`: helper functions to convert between Euler angles, quaternions, and rotation matrices.

All of these functions work with both numeric and symbolic variables, which you may use to your advantage.

2.1 Your task

You are designing a controller to stabilize a particular pointing direction.

1. For a selection of different operating conditions (equilibrium states), construct linearized equations of motion in the form

$$\dot{x} = Ax + Bu.$$

2. Design state-feedback controllers via LQR to stabilize the desired position and pointing direction. Justify your choice of weighting matrices and examine the effect of different choices on the closed-loop system response.

3. Examine the range of initial conditions that can be stabilized, the effect of external disturbances, and the effect of noisy state measurements.

Report Quality [10 marks]

10 marks will be allocated for the presentation and structure of your report and any associated code. There is a **strict page limit of 10 pages**, i.e. we will not read or mark anything past page 10. We recommend your report is closer to 5 pages if possible. Your Matlab code will be submitted separately, but you can include short snippets in your report if you like. Do not attach all code as an appendix.

It is expected that your report is well-written, clearly formatted, design decisions are well justified, and the results are concisely but thoroughly analysed, and that your code is well-structured, legible, and properly commented. Figures must be legible and clear.