Lunar Scout: Task 1A - Practical

Krutarth e-Yantra Staff Sep 2023

Task Instructions

In this task, we will be deriving the state space model of the Rotary Inverted Pendulum system and implementing the LQR control scheme on the model. You will be required to find out the equations of motion for the physical system and modify code as instructed to find the correct mathematical model & find optimal controller gains.
The Mathematical Modeling process is already demonstrated in the previous pages of this doc. Additionally a similar process for a two wheeled self balancing robot was shown during e-Yantra's technical sessions. So those who missed that, can refer it here:

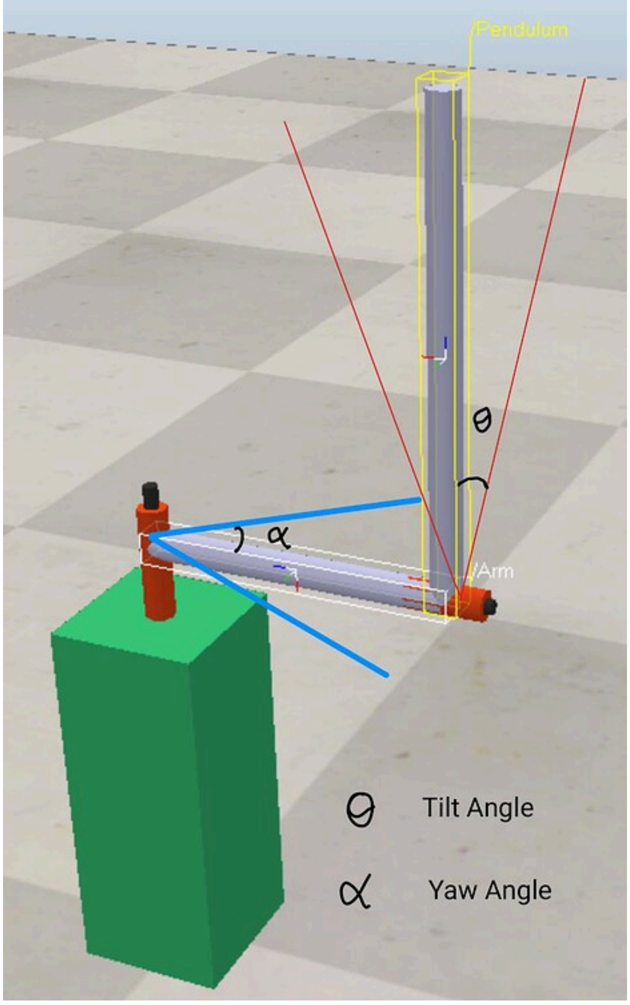
- https://www.youtube.com/watch?v=nOHky2f58ME
- https://www.youtube.com/watch?v=NPuD1dLu0lo

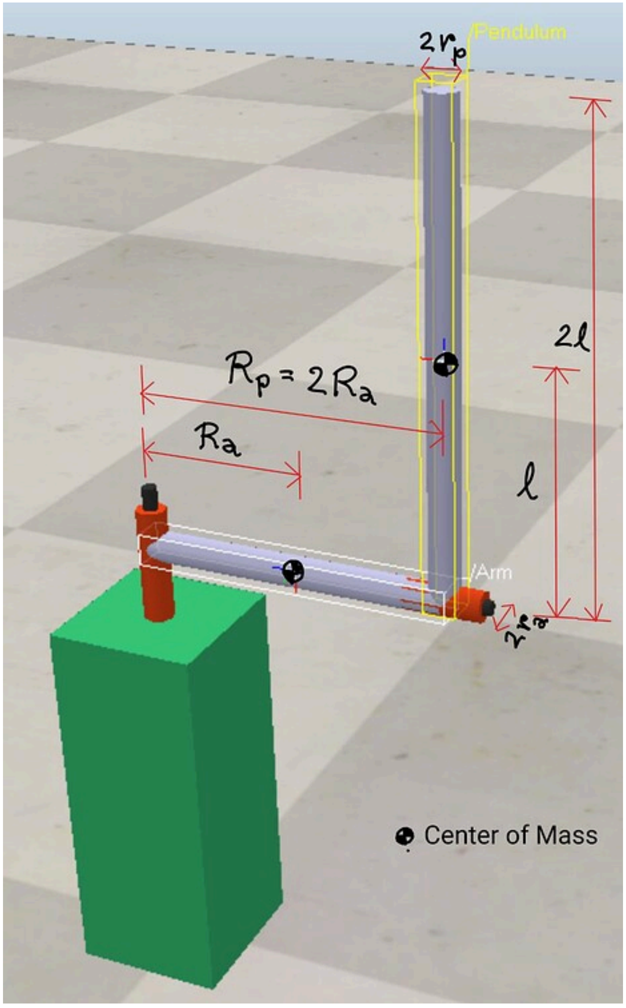
Download this **Task1A.zip** (**UPDATED**) file and extract. For every Physical System that we will discuss here, you will find following files in the folder:

1. Task_1A.m
2. task1a.pyc

Physical System - Rotary Inverted Pendulum

Skip to main content





Skip to main content

Open the file **Task_1A.m** using Octave.

In this file you will find the following functions defined:

- Jacobian _A_B()
- lqr_Rotary_Inverted_Pendulum()
- Rotary_Inverted_Pendulum_main()

---

1. Jacobian _A_B():
This function is used to define the dynamics of the system by using the equations of motion you have derived for this system, finally to get the **State Space Model** of the system.

```
Jacobian _A_B()

function [A,B] = Jacobian_A_B(Mp,l,g,Ma

 alpha = sym('alpha');

 theta = sym('theta');

 theta_dot = sym('theta_dot');

 alpha_dot = sym('alpha_dot');

 u         = sym('u');

 cos_theta = cos(theta);

 sin_theta = sin(theta);

 cos_alpha = cos(alpha);

 sin_alpha = sin(alpha);

  ########## ADD YOUR CODE HERE ##########
  #{
  Steps :
    1. Define equations of motion (4-state
    2. Partial Differentiation of equation
    3. Linearization by substituting equil

  ### NOTE ### : Sequence of states should

  SEQUENCE OF STATES : [alpha_dot; alpha;
        Example:
                A = [x x x x;   # correspo
```

```
                    x x x x;   # ...alpha
                    x x x x;   # ...theta
                    x x x x]   # ...theta
               B = [x;   # ...alpha_dot
                    x;   # ...alpha
                    x;   # ...theta_dot
                    x]   # ...theta
  #}
  ########################################

11.  A = ; # A should be (double) datatype

12.  B = ; # B should be (double) datatype

13. endfunction
```

Lets now understand this function:

- **Line 2 to Line 6** are the representation of states in symbols
- **Line 7 to Line 10** are the symbolic representation of **cos** and **sin** angle required in forming dynamic equations of the system
- There are HINTS given as comments in the function to help with the steps.
- Use the same sequence of states in the model as given in comments. Use **“0”** degree angle as the reference equilibrium point for tilt angle.
- Notice the Shape of **B matrix** given in comments. It says that you have to derive model considering it as **single input** system. (**Underactuated !!** 🤔)
- This function is intended to work as a helper to deal with long mathematical expressions in a simplified manner. **You can choose to do the math partially by-hand/on-paper and feed in the values here. It will be considered valid as long as it passes the evaluator’s tests without modifying anything else.**

---

2. lqr_Rotary_Inverted_Pendulum():

You do not have to modify anything in this function. This function demonstrates the use of Octave to apply LQR control and find optimal gains using performance specified by Q & R matrices.

> It will be used in **Task 1B**.

```
lqr_Rotary_Inverted_Pendulum()

1. function K = lqr_Rotary_Inverted_Pendul

2. C = [1 0 0 0;
        0 1 0 0;
        0 0 1 0;
        0 0 0 1];             ## Initialise

3. D     = [0;0;0;0];           ## Initiali

4. Q     = eye(4);              ## Initiali

5. R     = 1;                   ## Initiali

6. sys   = ss(A,B,C,D);         ## State Sp

7. K     = lqr(sys,Q,R);        ## Calculat

8. endfunction
```

> Note- **You are not allowed to make any changes to this function.**

---

3. Rotary_Inverted_Pendulum_main():

This function defines the physical parameters of system. It uses state space model generated and finds optimal gain using respective functions.

```
Rotary_Inverted_Pendulum_main()

1. function Rotary_Inverted_Pendulum_main(

2. Mp = 0.5 ;                   # mass of t

3. l = 0.15 ;                   # length fr

4. g = 9.81 ;                   # Accelerti

5. Ma = 0.25 ;                  # mass of t

6. r_a = 0.01;                  # radius of
```

Skip to main content

```
7. r_p = 0.01;                  # radius of
8. Rp = 0.2 ;                   # length fr
9. Ra = 0.1 ;                    # length f
10. I_arm = ;                    # Moment o
11. I_pendulum_theta = ;         # Moment o
12. I_pendulum_alpha = ;         # Moment o
13. [A,B] = Jacobian_A_B(Mp,l,g,Ma,Rp,Ra,I
14. K = lqr_Rotary_Inverted_Pendulum(A,B)
endfunction
```

- You have to use the dimensions and other physical parameters of the model to compute respective **moments of inertia** for the system. You can feed it to use while deriving dynamics equations.
- **Do not change the arguments or return values of any function.**

Testing your solution:

- Your solution can be tested using `task1a.pyc` provided in the Task1A.zip

environment

```
python task1a.pyc
```

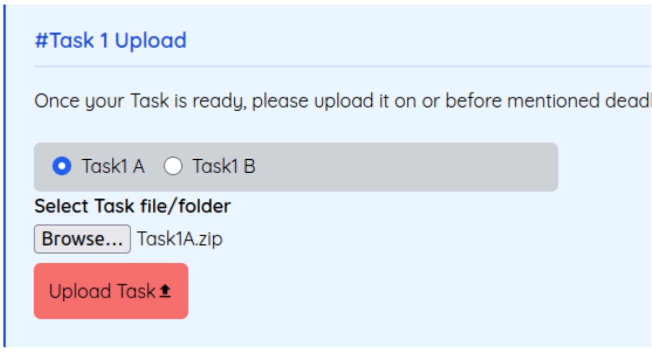
- Feed respective TEAM ID and Octave executable path when prompted by it.
- Results will be printed at the last and an encrypted `task1a_output.txt` file will be generated in same folder.

> For your reference, the evaluator checks format of your state space model, A, B matrices, open loop and closed loop eigen values/poles and stability, controlability, open loop response and LQR gains, etc.

Skip to main content

Submission Instructions (UPDATED):

- Once you get your output, **Task1A** folder will contain the following.:
  - Task_1A.m
  - task1a.pyc
  - task1a_output.txt
- Right click on the **Task1A** folder and compress as ***Task1A.zip***.
- Upload the `Task1A.zip` file on the portal.
- In the **Task 1 Upload** section, click on **Task 1A** bullet and select **Choose file** button to upload the `Task1A.zip` file. From the dialogue box, select the file and click **Open**.
- You shall see the file name `Task1A.zip` in text-box besides the **Choose file** button. Click on **Upload Task** button to submit the file.



Task 1A is complete!!

Congrats!!

Next >>

[Announcement] Task 1 Released!

Unlisted on Sep 17, 2023

Closed on Sep 19, 2023

Skip to main content

Related Topics

Topic	Replies	Views	Activity
Lunar Scout: Task 1	3	9.6k	Sep 2023
Lunar Scout: Task 2	3	3.8k	Sep 2023
TEAM ID:3280 Task1A practical task-1	14	276	Sep 2023
Doubt in Task 2A LS_3083 task-2	1	76	Oct 2023
Questionarie question 2	5	86	Dec 2023