Topics
My Posts
More
Categories
Lunar Scout (LS)
Guidelines
ChatbotDocs
All categories
Tags
task-0
task-1
task-2
other
task-4
All tags

Lunar Scout: Task 1B Practical

Krutarth e-Yantra Staff Sep 2023

Task Instructions

Download this **task1B.zip** (**UPDATED**) file and extract.
In the extracted folder you will find following files

- **`task1b.pyc`**
- **Task_1B.ttt**
- pendulumA.py
- pendulumB.py
- pendulumC.py
- linux / task1b
- windows / task1b.exe

> For linux : Use evaluator given in linux folder.
> Change permission by the command : `chmod a+x ./task1b`
> For windows : Use evaluator given in windows folder

CoppeliaSim Python Setup:

- **pyzmq** and **cbor** need to be installed in conda env.(Should be there already if Task 0A steps were followed properly)
- Conda environment's python interpreter path needs to be added in coppeliasim's installation directory inside **usrset.txt** .
  For example:
  - `..../CoppeliaSim_Edu_V4_5_1_rev4_Ubuntu22_04/system/usrset.txt`
  - A generic way to find location of usrset.txt is using this command in CoppeliaSim status bar console at the bottom of the window : `sim.getStringParam(sim.stringparam_usersettingsdir)`
  - Refer this thread to learn more : **Python Scripts not Working as Intended -**

Skip to main content

Coppelia Robotics forums

- Add your conda environment's python interpreter path here in `defaultPython` variable inside *usrset.txt*, for example :
  - `defaultPython = /home/e-yantra/miniconda3/envs/LS_9999/bin/python`
  - If you're not sure how to find the path, refer this: **Finding your Anaconda Python interpreter path — Anaconda documentation**

1. Problem Statement

In this task, teams have to write code to implement **LQR or PID** control strategy **to balance 3 rotary inverted pendulums at their unstable equilibrium point using python child scripts** in the given CoppeliaSim scene.

The 3 variants of rotary **pendulum should be balanced at upright position** and should **also parallelly maintain the angular position of arm in horizontal plane** - both together !!

This task should be accomplished using given scene environment without changing simulator or scene object properties. Team can change only the child scripts, following the mentioned guideline.

Two of these systems are pure rotary inverted pendulum with two cylindrical rods with uniform mass

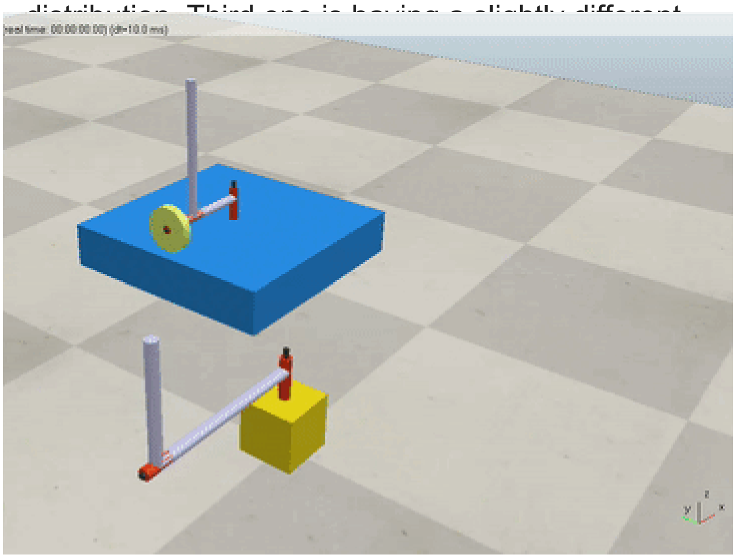


Figure 1: Rotary Pendulums Task Problem Statement

Skip to main content

2. Given

1. CoppeliaSim’s Scene File (Task_1B.ttt)

- As you open the scene file i.e. Task_1B.ttt in CoppeliaSim software as shown in Figure 1.
- You’ll find a hierarchical tree structure similar to the one shown in Figure 2.

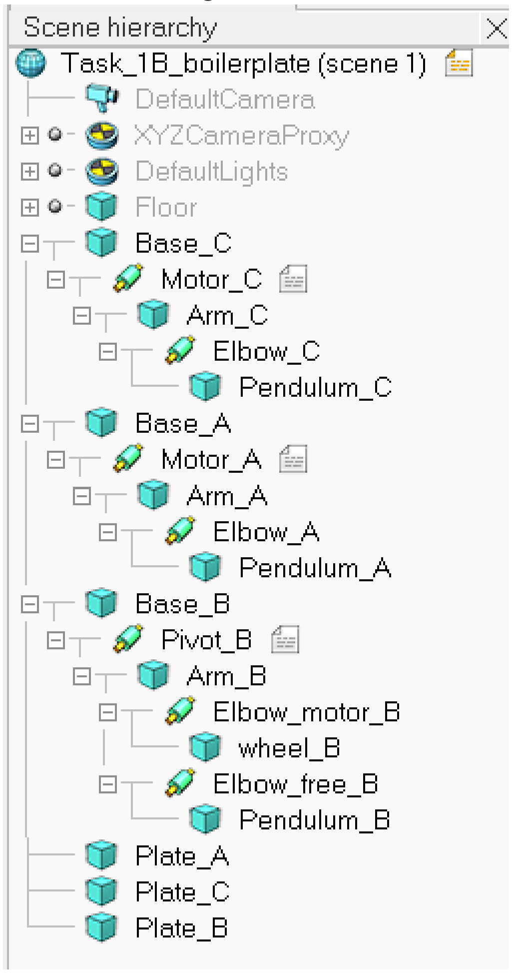


Figure 2: Example of Scene Hierarchy

- The **objects in the scene** along **with their names and uses** are given in Table 1.

Objects	Name in the scene	Use(s)
Base	*Base*	Acts like an immovable base for Rotary Inverted Pendulum.
		The names will be suffixed with object name. For ex: Base_A, Base_B, Base_C

Skip to main content

Objects	Name in the scene	Use(s)
Arm	*Arm*	Acts as the rotary arm in horizontal plane which moves the pendulum
Free Joint	*Pivot*	As a pivot for the rotary arm around which the arm may rotate
Motor	*Motor*	As an actuating joint behaving as DC geared motor for the rotary arm around which the arm may rotate
Pendulum	*Pendulum*	Acts as the pendulum
Motor	*Elbow_motor*	The joint acting as DC geared motor actuating wheel on rotary arm.
Free Joint	*Elbow_free*	The joint on rotary arm acting as free pivot for pendulum rod’s rotation in vertical plane
Free Joint	*Elbow*	The joint on rotary arm acting as free pivot for pendulum rod’s rotation in vertical plane

> **NOTE:**
>
> - In this task you are **NOT** allowed to **add or remove** objects in scene.
> - Any change in the **names of the objects, their parent child relationships, their properties, position, orientation** etc. will result in **poor evaluation** and hence low marks.

Skip to main content

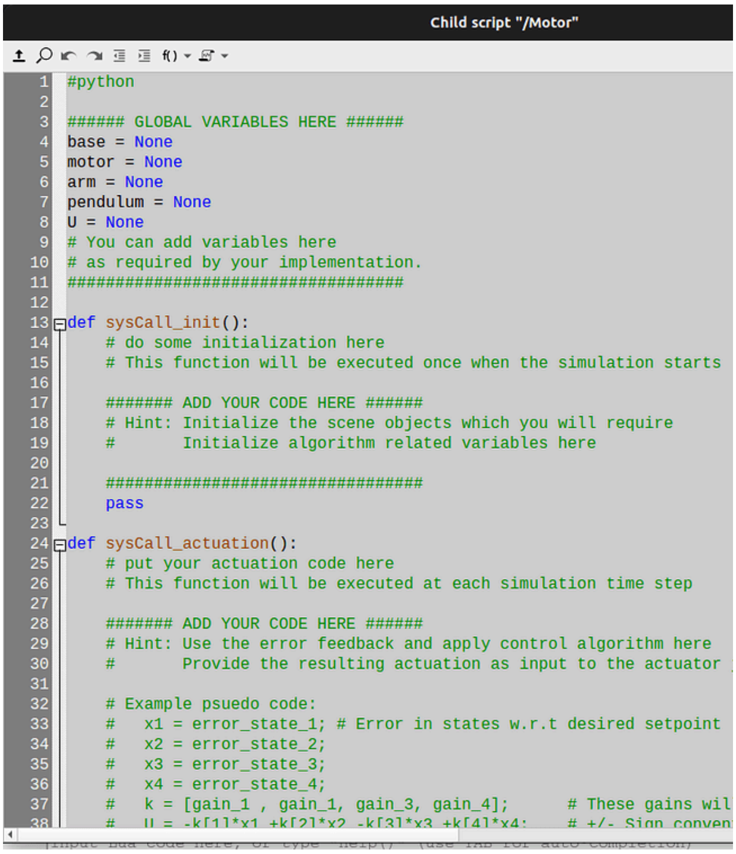
- **Pendulum** in each model is having a mass of 0.5kg.
- **Arm** in each model is having a mass of 0.25kg.
- **Pivot, Elbow, Elbow_free** are revolute joints in free mode.
- **Motor, Elbow_motor** are revolute joints in velocity control mode, with a max. torque rating.
- Similarly all other properties of all objects in scene can be known by double clicking the object name in scene hierarchy.

> If you’ve followed the introduction to coppeliasim tutorial, then try exploring other options in coppeliasim like the position and orientation of object in different frames, etc. You can set a good camera view in simulation for yourself 😉
> But yeah, make sure you use a fresh scene before starting implementation of the task - to make the evaluation smooth.

Understanding the Task:

- You’ll find a script icon near the Motor & Pivot as shown in Figure 2.
- Click on the script icon. You’ll have the script opened as shown in Figure 3. You have to edit this file by writing your code in appropriate function to execute the control loop to balance each rotary inverted pendulum.

> There are explainatory comments given in each function - read them through.




```
#python

###### GLOBAL VARIABLES HERE ######
base = None
motor = None
arm = None
pendulum = None
U = None
# You can add variables here
# as required by your implementation.
###################################

def sysCall_init():
    # do some initialization here
    # This function will be executed once when the simulation starts

    ####### ADD YOUR CODE HERE ######
    # Hint: Initialize the scene objects which you will require
    #       Initialize algorithm related variables here

    #################################
    pass

def sysCall_actuation():
    # put your actuation code here
    # This function will be executed at each simulation time step

    ####### ADD YOUR CODE HERE ######
    # Hint: Use the error feedback and apply control algorithm here
    #       Provide the resulting actuation as input to the actuator

    # Example psuedo code:
    #   x1 = error_state_1; # Error in states w.r.t desired setpoint
    #   x2 = error_state_2;
    #   x3 = error_state_3;
    #   x4 = error_state_4;
    #   k = [gain_1 , gain_1, gain_3, gain_4];      # These gains wil
    #   U = -k[1]*x1 +k[2]*x2 -k[3]*x3 +k[4]*x4:    # +/- Sign conven
```

Figure 3: Motor Child Script

Applying Control Technique (LQR/PID) :

> You are allowed to use either of LQR or PID control technique to balance the pendulums.

LQR : 💪

- Refer back to **Task 1A** where you have been provided with a function in octave named as `lqr_Rotary_Inverted_Pendulum(A,B)`. If you refer to the comments of the function, you can know that is uses the state space model of system and using Q & R matrices it generates optimal gains for each of the 4 states as per desired performance
- This desired performance is specified by Q & R matrices. **Q is the state cost weight matrix** & **R is the input cost weight matrix**. Both are ***diagonal matrices***.
    - For Q - each diagonal value corresponds to a state(as per *state sequence*) and it is of nxn size for n states. Whereas R is of uxu size for n states and u inputs. The diagonal entries of R correspond to each input.

Skip to main content

- Choosing Q & R is takes a thorough understanding and observation of system dynamics. Weights in Q matrix should be given in a sequence such that **higher priority is given to the most critical states** for the controller. Similary R matrix shares the energy cost among all the inputs. For single input system it's less significant as all energy is given to single input only.
- The relative values of weights matter for Q & R matrices. The absolute value doesn't make much change.(***Learn more about using lqr() here***)

- Refer this learn **more about LQR design**

> **WAIT HEY** … The Rotary Inverted Pendulum that you have modeled in Task 1A. Does it fit with all 3 of the pendulums given in this task ?
>
> - Prepare to make some changes in the state space model if you find it different here !!! 🙃

PID : 👊

- This technique can be applied without knowing state-space model. It requires lesser efforts in math as it's an observation based tuning method, but there are ways to mathematically tune it for our system for optimal performance.

> **IMPORTANT NOTE :** Learning about **PID is NOT compulsory** for this task. Teams can use either of PID or LQR for completing this task.

- There are good resources on PID tuning like
  - **Proportional–integral–derivative controller - Wikipedia**
  - **MATLAB Tech-Talks playlist on PID**
  - Refer this to learn **more about PID controller design**

> **SUGGESTION** : Controlling 2 angles and total 4 states may require more than a straightforward PID. So you can try out variations of it. For example, *cascade control* or *parallel PID* etc.

Skip to main content

Submission Instructions (**UPDATED**)

Once you have successfully performed this experiment:

1. Open the Task_1B.ttt in CoppeliaSim.
2. Open new Terminal (on Ubuntu OS or MacOS) or Anaconda Prompt (on Windows OS) and navigate to the **Task1B** folder.
3. Activate your conda environment with the command

   ```
   conda activate LS_<team_id>
   ```

   Example: conda activate LS_9999
4. Run the evaluator file **task1b** or **task1b.exe** by running following command
   For **linux**:

   ```
   ./task1b
   ```

   For **windows**:

   ```
   task1b.exe
   ```

5. When asked, you have to enter your Team ID, such as 9999.
6. It will trigger the simulation of Task_1B.ttt using the python API and calculate the settling time for all 3 pendulums to balance at the unstable equilibrium as shown below:

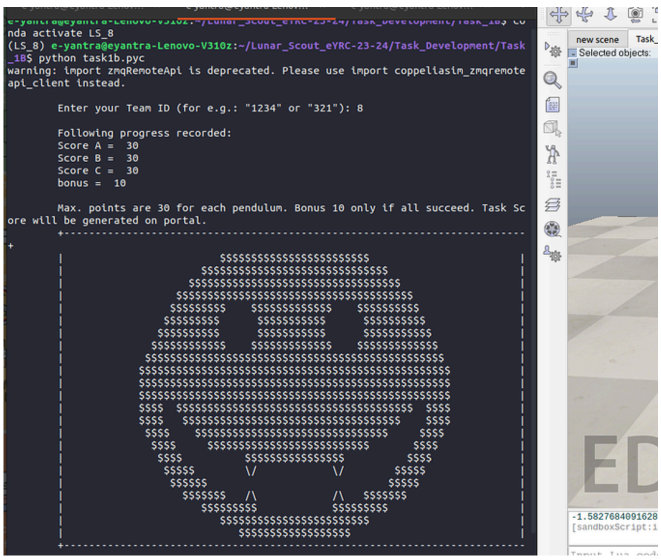


Figure 4: Final desired output

Skip to main content

8. It will generate `task1b_output.txt`. **Now copy the child script for all 3 pendulums and paste inside pendulumA.py, pendulumB.py, pendulumC.py given in Task1B folder.** Once you get your output, Task1B folder will contain the following:

- **Task_1B.ttt**
- pendulumA.py
- pendulumB.py
- pendulumC.py
- task1b_output.txt

9. Right click on the **Task1B** folder and compress as ***Task1B.zip***.
10. Click on this link: **https://portal.e-yantra.org/task_task1** . In the **Task 1 Upload** section, click on **Task 1B** bullet and select **Choose file** button to upload the `Task1B.zip` file. From the dialogue box, select the file and click **Open**.
11. You shall see the file name in text-box besides the **Choose file** button. Click on **Upload Task** button to submit the file.

Task 1B is complete!!

Congrats!! 🎉

🔗 **[Announcement] Grader Changed for Task-1B**

Unlisted on Sep 17, 2023

Closed on Sep 19, 2023

Related Topics

Topic	Replies	Views	Activity
🔒 **Lunar Scout: Task 1**	3	9.6k	Sep 2023

Skip to main content

Topic	Replies	Views	Activity
Lunar Scout: Task 2	3	3.8k	Sep 2023
Doubt in Task 2A LS_3083 task-2	1	76	Oct 2023
PID controller for task 1B task-1	2	107	Oct 2023
Lunar Scout: Task-4	1	969	Nov 2023