SCE594: Special Topics in Intelligent Automation & Robotics

Assignment 5

Instructions:

- This is a **graded** assignment of 5%.
- You can do the assignment either **individually** or in a **group of 2**.
- You can write on the pdf directly with a tablet & electronic pen **or** print the pdf and answer with an ink-pen.
- Good handwriting and neatness makes your work easier to read and understand and is highly recommended.
- Upload your solution as a **single compressed folder** on Blackboard by **11:59pm Thursday 30 April 2026 (end of the day)**. Include your MATLAB code with the pdf.
- If you do the assignment in a group of two, both members should upload the pdf file.
- Plagiarism will **not be tolerated** at all.

Student 1 Name:	
Student 1 ID:	

Student 2 Name:	
Student 2 ID:	

1. Introduction

- In this assignment you will study how to control the orientation of a rigid body in 3D space using a PD (proportional-derivative) controller directly on the rotation group SO(3).
- The provided MATLAB script, rigid_body_PD_control.m, simulates the dynamics of a rigid body with a diagonal inertia matrix. The code applies different desired orientations over an 8-second window, switching to a new orientation every 2 seconds.

2. Given Code Explanation

Open rigid_body_PD_control.m in MATLAB. You will see these main sections:

1. **Simulation Setup**
   - Time-step *dt*, total simulation time *T*, and number of steps *steps*.
   - Inertia matrix *inertia*.
   - Initial orientation *R* and angular velocity *omega*.
   - Four different desired orientations *RdSet*, each selected for a 2-second interval.
2. **Integration & Visualization**
   - A loop runs from i = 1 to steps.
   - At each time step, the code checks which desired orientation Rd should be active.
   - It calls a **(placeholder) PD controller** function named pd_controller to compute the control torque tau.
   - The torque is then passed to an integrator *lie_poisson_midpt_solver* to update orientation and angular momentum.
   - Plots of angular velocity and a 3D rigid body are updated in real time.
3. **Utility Functions**
   - *lie_poisson_midpt_solver* for integrating orientation on SO(3).
   - *skew, skew_inv, and rodrigues* for matrix operations on rotations.
4. ***pd_controller* (Placeholder)**
   - In this empty function you will add your controller as part of the assignment tasks below.

3. Assignment Requirements

- **Implement Two Different Geometric PD Controllers**
  In the script, you will see a function:

```
function tau_k = pd_controller(R_k, w_k, Rd)
   % PD controller for orientation on SO(3)
   % R_k : Current orientation (3x3)
   % w_k : Current angular velocity in body frame (3x1)
   % Rd  : Desired orientation (3x3)

   % STUDENT TASK: Fill in TWO PD controllers
end
```

You must fill in the logic inside pd_controller to compute *tau_k* based on the orientation error and angular velocity.
Implement two PD controllers: 1) a geometric one that computes the proportional term from the gradient of a function $\Psi(R)$ as we did in class, 2) a standard PD controller using Euler angle parameterization.
You are free to try out other extra control approaches e.g. different $\Psi(R)$ functions on *SO(3),* feedback linearization or quaternion based … etc.

- **Experiment with Gains**
  - Choose **at least two** different sets of (Kp,Kd) (or your chosen gain matrices) for **each** of your two PD approaches.
  - For each set, run the simulation and collect:
    1. The angular velocity plots vs. time $(\omega 1, \omega 2, \omega 3)$.
    2. The control torque τ vs. time (plot *tauHistory* similarly or produce a separate figure to show how torque evolves).
- **Report in pdf format**
  - **Figures**: Submit plots for ω(t) and τ(t) for each of your two PD controller variants and for **at least two** gain settings each (total: 4 sets of plots).
  - **Discussion**: For each set of gains, comment on the comparison of the two (or more) approaches in terms of:
    i. How does the body converge to the desired orientation.
    ii. Whether there is overshoot or oscillation.
    iii. The magnitude of the control torque.

4. Assignment Deliverables:
You should turn in:

1. **Modified rigid_body_PD_control.m**
   Ensure it contains your final code for pd_controller with both PD approaches. You can handle this in separate "if/else" statements or as two separate functions.
2. **Plots & Explanation**
   - A PDF with the requested plots (ω vs. time, τ vs. time) for each approach and each gain set.
   - A short write-up describing your findings, especially how changes to the gains affected convergence and torque demands.

Hints:

- Remember that the code is already set to change orientations every 2 seconds. You should see how your controller transitions from one orientation to the next.
- If you see large transients or intense oscillations, consider tuning Kd and Kp.
- The desired rotation matrix R_d is not shown in the figure. You can modify the code to plot the axes of the desired frame if you wish.