Assignment of (AE4324)
Physical Interaction for Aerial and Space Robots

Please create a single PDF report with your group containing your answers and results. Please explain your way of working, include derivations, and explain how your code works. Read the tasks very well and make sure that you fully answer each question.
Include with your report all the code used to produce the results you present in the report. This can be via a GitHub repository (set to public!), or simply by adding the code files. Do not forget to include a readme with clear instructions on how to use your code. Your results should be easily reproducible using your code, the robot, and the instructions provided in your readme.
The last task contains a small competition; however full credit is achievable regardless of competition performance.
Please upload the assignment archive on Brightspace **by 1st April 2026, 17:00 o'clock**.

Grading rubric

Task	Points	
Reporting quality	2.5	5
Code quality	2.5	
Task 1.1 Measuring & mobility	2	5
Task 1.2 FK & workspace	3	
Task 2.1 IK & feasibility	3	8
Task 2.2 Elbow up/down	2	
Task 2.3 Shape trajectory	3	
Task 3.1 Jacobian	2	6
Task 3.2 Singularities	2	
Task 3.3 Velocity control	2	
Task 4 Pick & place	2.5	6
Task 5 Dynamic stacking of cubes	3.5	
Competition Top 3 Groups	5	1st: 5pts, 2nd: 3pts, 3rd: 2pts
Total	**30 + 5**	

Reporting guidelines

To receive full reporting quality credit, the report must be clearly structured and easy to read. A solution that is easy to follow and evaluate is appreciated. Include:

- A title page with title, group number, names, and student numbers.
- Assumptions and conventions that form the basis of your approach.
- Brief conceptual explanation of the chosen approach.
- Clear and intuitively understandable visualizations with descriptive captions.
- Concise but complete explanation of intermediate steps and derivation steps.
- Full derivations may be given in well-commented or explained code .e.g. using a Jupyter Notebook with Sympy.
- Conclusive answers to the questions or results showing completion of the task.

Coding guidelines

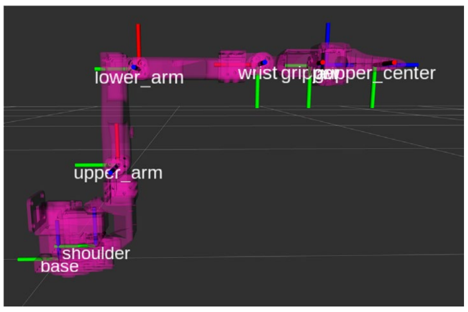
To receive full coding quality credit, the code should allow for easy reproducibility of the results you report.

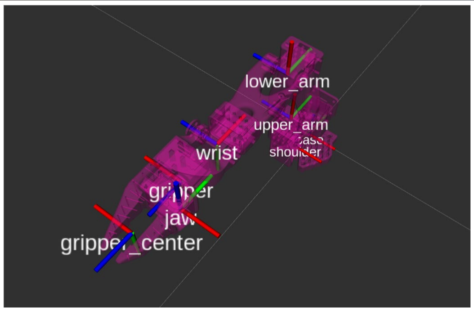
- Include a readme with instructions on how to use the code.
- Put comments in your code explaining what happens.
- Structure your code (i.e. your functions, classes, files etc.) in a logical way
- Agree with your group on common style conventions (e.g. camelCase or under_scored function names, descriptive variable names).
- Clearly indicate which scripts or entry points correspond to presented results, where needed with an explicit command.

1. Mobility and Workspaces

Task 1.1:

Consider a generic 5-DoF robotic manipulator and its skeleton assembled from rigid links and revolute joints (RRRRR) as pictured below.





Parent Frame	Child Frame	Translation (X Y Z)	Rotation (Roll Pitch Yaw)
world	base	0.0 0.0 0.0	0 0 3.14159
base	shoulder	0 -0.0452 0.0165	0 0 0
shoulder	upper_arm	0 -0.0306 0.1025	0.0 -1.57079 0.0
upper_arm	lower_arm	0.11257 -0.028 0	0 0 0
lower_arm	wrist	0.0052 -0.1349 0	0 0 1.57079
wrist	gripper	-0.0601 0 0	0 -1.57079 0
gripper	gripper_center	0.0 0.0 0.075	0 0 0
gripper	jaw	-0.0202 0 0.0244	1.57079 3.14158 0.0

1. Find the robot and measure each of the joint limits.
2. Mobility:
    a. What is the mobility of the robot?
    b. Which DoFs in joint space and which DoFs in Euclidean space are controllable?

Task 1.2:

Consider the same robot as in Task 1.1.

1. Find the mathematical expression for the forward kinematics of the robot.

Using your visualization software of choice (Python, Matlab, ROS2+RVIZ, etc.), make a 3D visualization of the robot's reachable workspace if

2. None of the robot joints would be constrained.
3. The actual joint limits apply.

Hint: The visualization should show the outline of the space that is reachable by the end-effector.

4. Describe qualitatively what configurations are most likely to break the robot i.e. what configurations are most taxing on the robot hardware.

2. Inverse Kinematics

Task 2.1:

Again, consider the robot as in Task 1.1.

1. Using your programming language of choice, program the inverse kinematics (IK) function:

   $$q = IK(p)$$

   I.e. a function the returns the required joint angles **q** to achieve a given end-effector position **p**.
2. Consider the following set of end-effector poses [x, y, z, pitch, roll]:

   I. [0.2 0.2 0.2 1.57 0.0]
   II. [0.2 0.1 0.4 0.0 1.57]
   III. [0.0 0.0 0.45 0.785 0.785]
   IV. [0.0 0.0 0.07 3.141 0.0]
   IV. [0.0 0.0452 0.45 0.785 3.141]

   a. Which positions have solutions through your IK and which ones do not?
   b. For those positions you answered with "no", why is this the case?
3. Showcase your IK-function by commanding the robot to the feasible positions
   a. In simulation.
   b. On the real system.

Task 2.2:

The robot's IK with respect to the end-effector position and two rotational DoFs can have multiple solutions. Visualize some of the multiple solutions for the feasible positions from task 2.1. On the physical robot, which of the solutions are practically feasible? For the ones that aren't, why not?

Task 2.3:

Pick a shape of your choice (e.g. a circle, a square, a rectangle, or why not Prometheus' flame from the TU Delft logo). In this task you need to command the robot to follow this shape with its end-effector.

1. Pick a plane to visualize the shape. Does it fit better into the workspace vertically or horizontally?
2. Parametrize the shape along time such that you obtain a set of end-effector setpoints that trace out your shape. We call this set a *trajectory.*
3. Using your IK function to find the set of joint state values that result in the end-effector following your trajectory.
4. Demonstrate your trajectory on the real robot. (Why not add a light to the end-effector and take a long exposure image?).

3. The Jacobian

The Jacobian is the matrix that maps the joint velocities to the end-effector velocities as follows:

$$\dot{x} = J(q)\,\dot{q}$$

It is derived by forming the total time derivative of the forward kinematics. In this task, we will concern us with how to compute the Jacobian, its properties, as well as its applications.

Task 3.1:

Again, consider the robot introduced in task 1.1. Find the symbolic expression for its linear and rotational Jacobian. Furthermore, implement a function that computes your Jacobian and evaluate it on the configurations you found in task 2.1.2.

Task 3.2:

What happens to the mathematical expression of the Jacobian in the configurations above? Why is this the case?
On the real robot, what are the implications when considering the equation above?

Task 3.3:

Using the Jacobian matrix, we can compute the joint velocities required to achieve given end-effector velocities. Choose a constant velocity trajectory for the end-effector and execute it on the real robot.

Hints:

- The robot provides direct joint-velocity control; change the control mode to velocity and test your approach. Make sure to test your approach in simulation first, before you move to the robot.
- Make sure to avoid any singularities while following the constant velocity profile.

4. Pick and Place

One of the most common tasks in robotics is *Pick and Place*: picking an object at one location and placing it at a different one. This task can be arbitrarily hard by just varying the nature of the object or the desired location. E.g. grasping something on an open, flat surface is easier to reach than a narrow gap with obstacles all around, or soft objects are harder to grasp without damaging them compared to rigid objects that can be grasped easily.
In this task you will demonstrate pick & place with a fragile object such as an egg or a piece of fruit (grape, raspberry, strawberry etc.).

Setup: Define two locations within the robot workspace that are 10 cm apart. You can use a printout of a grid from Brightspace to easily align the object. Place the object of your choice on top of a platform in one location and a platform in the other location. The robot is at a "home" position, in the proximity of the object.
Pick and place the object from its initial position to the platform. Regulate the grasping force to ensure that the robot doesn't squeeze the object too hard, and it arrives intact. Afterwards, command the robot back into its home position. Finally, pick and place the object from the second platform back to its initial position.

- What challenges do you observe while performing this task?
- What modifications of the robot would improve the performance?

5. Dynamic Stacking of Cubes: COMPETITION!

Pick-and-place manipulation inherently requires a trade-off between speed and accuracy. This trade-off becomes particularly critical in consecutive placement tasks such as stacking, where small positioning errors can compromise the stability of the entire structure. In this task, you will program a robotic system to autonomously pick up and stack cubes in a designated target area.

Baseline Task (full credit if completed)
To achieve full credit for this task, your robot must successfully stack **three cubes** on top of each other within the **target area** defined by the provided coordinate system.

- The **initial positions** of the cubes may be chosen arbitrarily.
- The **final stacked configuration** must satisfy the following conditions:
    - All cubes are stacked vertically (one cube placed on top of another).
    - The entire stack is **fully contained within the target area**.
    - The stack must be **stable** (i.e., the cubes remain stacked without external support after placement).

Competition: Biomorphic Intelligence Lab Robotics Cup
In the final tutorial session, all teams are invited to participate in an optional competition.

- Each team will have **three attempts**.
- In each attempt, the robot has 3**0 seconds** to stack the highest possible tower from cubes in the target area.
- Only stacks that are fully inside the target area and remain stable for > 10 seconds at the end of the attempt will be counted.

- The team that achieves the **highest number of stacked cubes in a single attempt** wins the **official Biomorphic Intelligence Lab Robotics Cup** 🏆

This challenge rewards not only precision but also efficiency and robustness, encouraging teams to explore the balance between speed, accuracy, and reliable manipulation.