Animating a Robot Arm Using Forward Kinematics: Additional Instructions

Overview

The following are additional requirements to be implemented in the solution of the assignment on animating a robot arm using forward kinematics. The added requirement consists of three parts:

1. Implement a version of the forward-kinematic function specifically for testing purposes.
2. Run the test program provided to ensure the function produces the correct results. Include a screenshot of the test results to the `README.md` file in the assignment repository.
3. Update the testing workflow files in the repository so the automated testing runs every time you push new code to the repository.

This implementation is solely aimed at automated testing of your solution. It is not necessarily the same robot that you will implement for the rendering component of the assignment or animation generation.

1. The function to be implemented

Add a function named `forward_kinematics()` to your program. Implement the function so it represents the robot arm shown in Figures 1, 2, and 3 below:

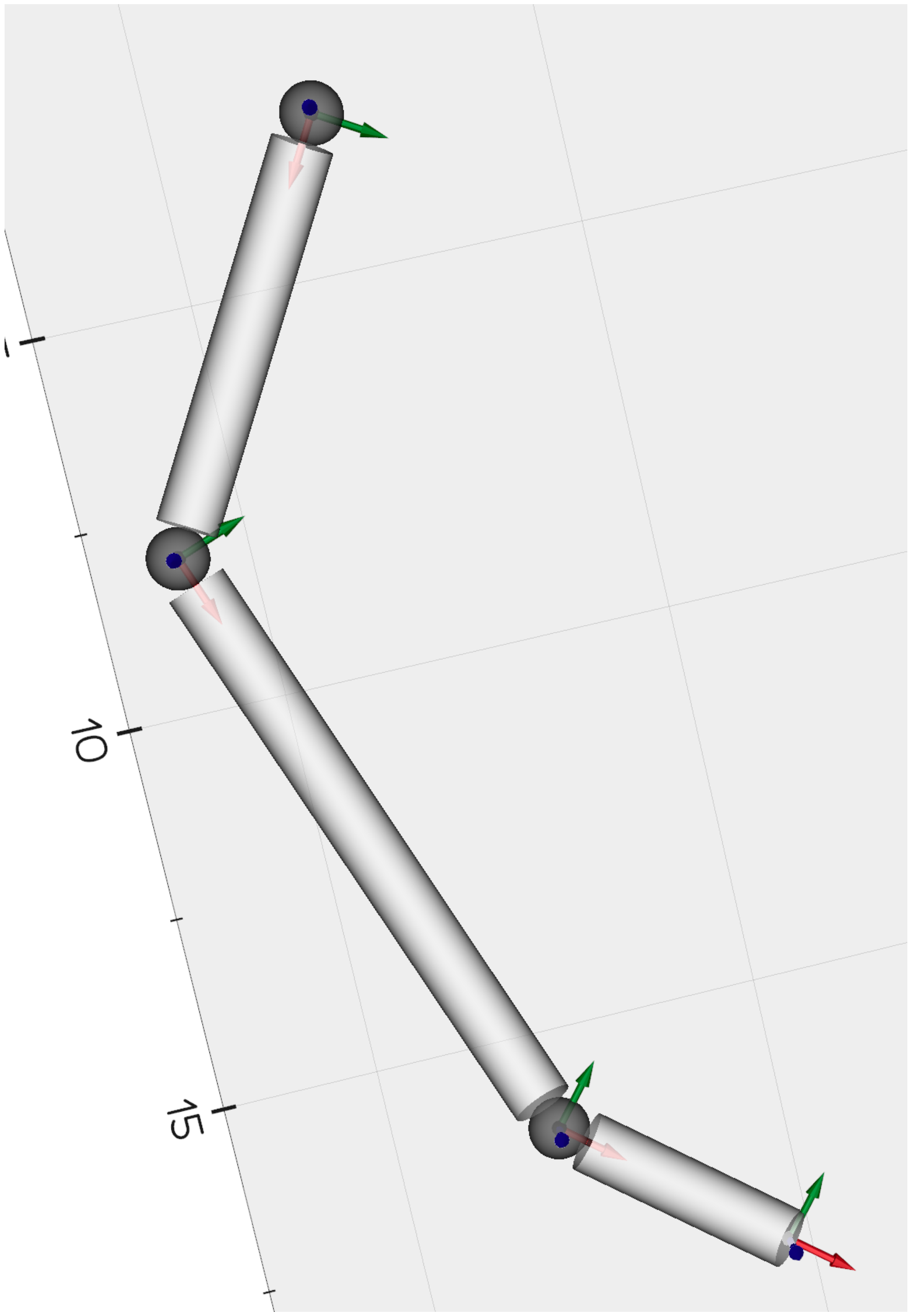


Figure 1: Robot arm with 4 parts. The 4th part is the end effector. Joints are represented by spheres of radius = 0.4. The location of the origin of the first frame of the arm is at $(3, 2, 0)^{\mathsf{T}}$.

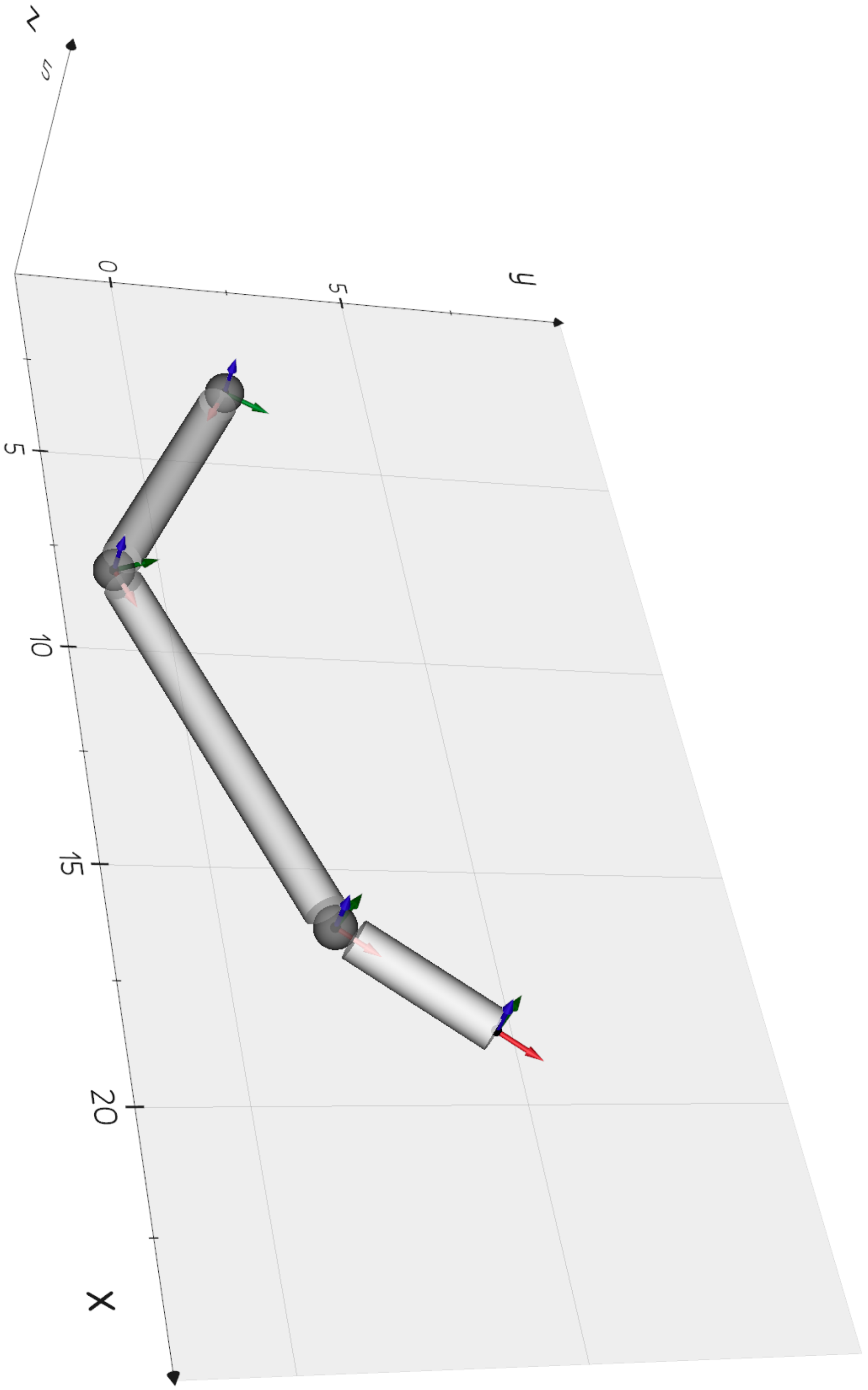


Figure 2: Robot arm with 4 parts. The 4th part is the end effector. Joints are represented by spheres of radius = 0.4. The location of the origin of the first frame of the arm is at $(3, 2, 0)^\top$

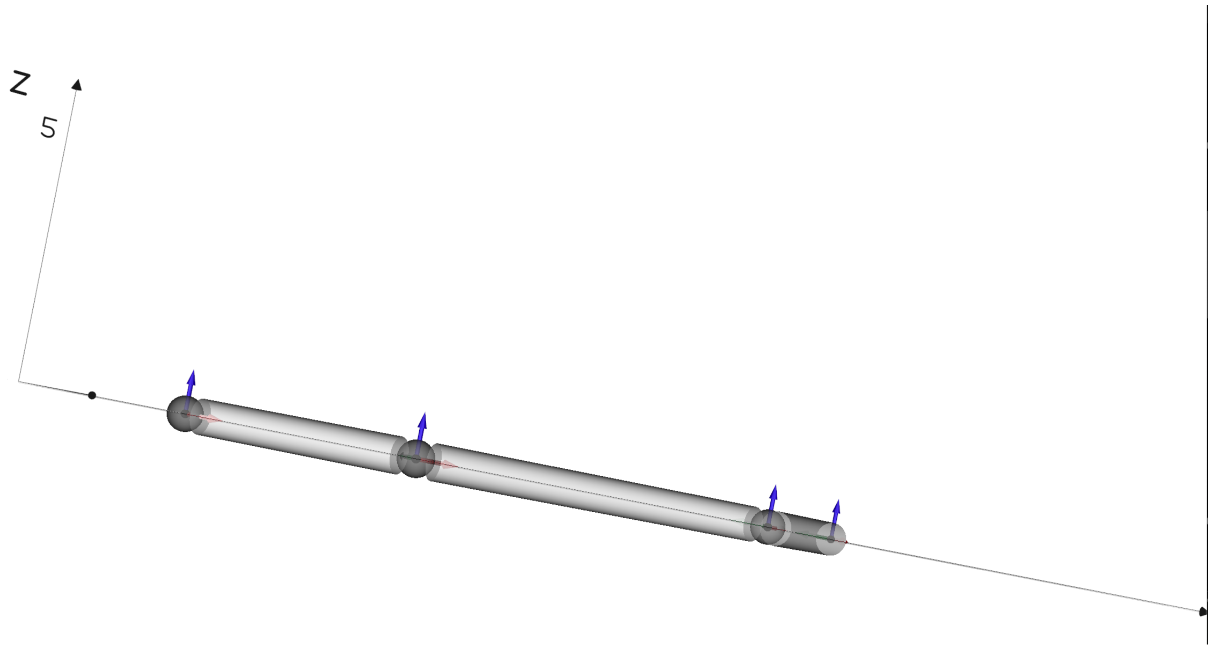


Figure 3: The arm lies entirely on the $xy-$plane. See now the $xy-$plane bi-sects the cylinders and spheres. Also, the origins of the coordinate frames lie on the $xy-$plane, i.e., their z-coordinate is zero.

Specification of the test robot arm

- Radius of the spheres representing the joints = 0.4.
- Arm has 4 local coordinate frames. Three coordinate frames for parts, and one coordinate frame for the end effector.
- The location of the origin of the first frame of the arm is at $(3, 2, 0)^\mathsf{T}$. This is the location of the arm in space.

The `forward_kinematics()` function should have the following signature:

```
def forward_kinematics(Phi, L1, L2, L3, L4):
    """Calculate the local-to-global frame matrices,
    and the location of the end-effector.

    Args:
        Phi (4x1 nd.array):       Array containing the four joint angles
        L1, L2, L3, L4 (float):   lengths of the parts of the robot arm.
                                 e.g., Phi = np.array([0, -10, 20, 0])

    Returns:
        T_01, T_02, T_03, T_04:   4x4 nd.arrays of local-to-global matrices
                                 for each frame.

        e:                       3x1 nd.array of 3-D coordinates, the
                                 location of the end-effector in space.

    """


  # Function implementation goes here



  return T_01, T_02, T_03, T_04, e
```

Finally, this testing function should not be part of a class in your program. It is fine to have your own version of the function inside a class for using in your program for rendering. But, for this testing version of the forward-kinematics function, it must be written as a standard function outside a class.

2. Testing the function

The following is the example of calling this function for testing its output:

```
# Lentghs of the parts
L1, L2, L3, L4 = [5, 8, 3, 0]

# Retrieve pose matrices and end-effector coordinates
# for a given configuration of Phi = [ phi1, phi2, phi3, phi4 ]
Phi = np.array([-30, 50, 30, 0])
T_01, T_02, T_03, T_04, e = forward_kinematics(Phi, L1, L2, L3, L4)
```

which should then result in the following expected value for `e`:

```
e = [ 1.85e+01, 4.71e+00, 0.00e+00 ]
```

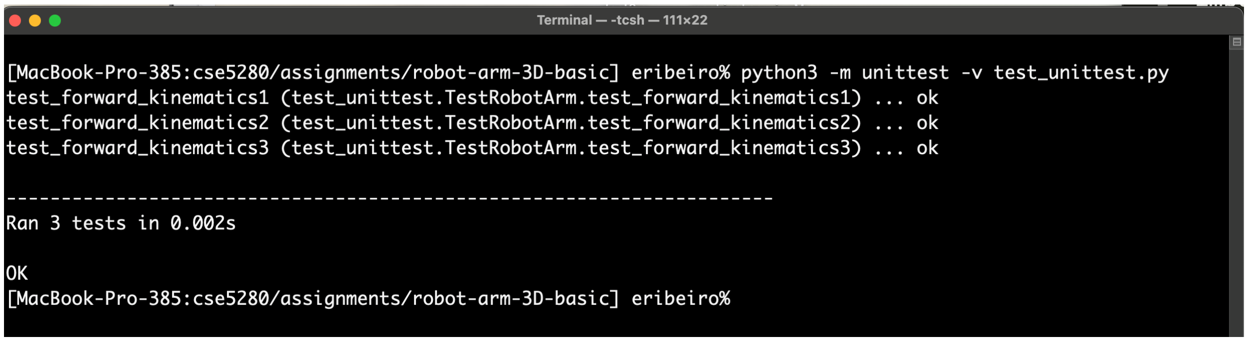
while the expected value of transformation T_{04} is:

```
T_04 = [ 6.43e-01 -7.66e-01  0.00e+00  1.85e+01]
       [ 7.66e-01  6.43e-01  0.00e+00  4.71e+00]
       [ 0.00e+00  0.00e+00  1.00e+00  0.00e+00]
       [ 0.00e+00  0.00e+00  0.00e+00  1.00e+00]
```

To run the unit test on your computer, call the test program from the command line as follows:

```
python3 -m unittest -v test_unittest.py
```

which should produce the following output once the function passes the tests.



```
[MacBook-Pro-385:cse5280/assignments/robot-arm-3D-basic] eribeiro% python3 -m unittest -v test_unittest.py
test_forward_kinematics1 (test_unittest.TestRobotArm.test_forward_kinematics1) ... ok
test_forward_kinematics2 (test_unittest.TestRobotArm.test_forward_kinematics2) ... ok
test_forward_kinematics3 (test_unittest.TestRobotArm.test_forward_kinematics3) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
[MacBook-Pro-385:cse5280/assignments/robot-arm-3D-basic] eribeiro%
```

Add the screenshot of the testing result to the `README.md` file of the repository.

3. Update the GitHub testing workflow

To update the test workflow, you need make the following changes directly to your repository:

1. Add the file `requirements.txt` to the repository.
2. Add the file `test_unittest.py` to the repository.
3. Create a directory named `.github`
4. Inside the directory `.github/`, create the directory named `classroom/`
5. Copy the file `autograding.json` into `.github/classroom/`
6. Inside the directory `.github/`, create the directory named `workflows/`
7. Copy the file `classroom.yml` into `.github/workflows/`
8. Commit and push the changes into the GitHub repository.

After these steps are completed, your test workflow should work as expected, and you will be able to see the *green checkmark* ✅ once your notebook passes all tests.

The files you need are in attachment to the email that was sent with these notes.

`.github/workflows/classroom.yml`

```yaml
name: GitHub Classroom Workflow

on:
  - push
  - workflow_dispatch

permissions:
  checks: write
  actions: read
  contents: read

jobs:
  build:
    name: Autograding
    runs-on: ubuntu-latest
    if: github.actor != 'github-classroom[bot]'
    steps:
      - uses: actions/checkout@v4
      - uses: education/autograding@v1
```

`.github/classroom/autograding.json`

```json
{
  "tests": [
    {
      "name": "test_forward_kinematics_assignment",
      "setup": "pip install -r requirements.txt",
      "run": "python3 -m unittest -v test_unittest.py",
      "input": "",
      "output": "",
      "comparison": "exact",
      "timeout": 10,
      "points": null
    }
  ]
}
```

requirements.txt

```
jupyterlab
nbclient
pandas
importnb
cvxopt
matplotlib
vedo
```