CS 4/6545 Autonomous Robotics: Jacobians

Read all instructions carefully.

Submission instructions: Your submission will consist of all code and output from Problem #3 proving that you solved the assignment properly. The easiest way to submit the assignment to me is to "tar" or "zip" up your assignment:

```
Jacobians $ cd ..
AuRo $ tar czf Jacobians.tgz Jacobians
```

This will yield the file `Jacobians.tgz` in my `AuRo` directory. The `tar czf` command stores the entire `Jacobians` directory and compresses it.

Assignment overview: In this assignment, you are to practice with computing Jacobians using the planar arm. The code you need to modify will be in `Jacobians.cpp`.

1. *Gazebo* does not have much in the way of matrix/vector arithmetic or linear algebra. Install my library, *Ravelin*, which will give you access to these things. You can build *Ravelin* by downloading the source from `https://github.com/PositronicsLab/Ravelin` (easiest way is to click the "Download Zip" icon). You can then build and install by:

   ```
   Downloads $ unzip Ravelin.zip
   Downloads $ cd Ravelin
   Ravelin $ mkdir build
   Ravelin $ cd build
   build $ cmake ..
   build $ make
   build $ sudo make install
   ```

   We will continue to use this library through the remainder of the class.

2. Compute the 3×3 Jacobian matrix. Each column of the Jacobian will correspond to a joint (left-most column will correspond to joint 1, right-most to joint 3). The top row of the matrix will correspond to the movement of the end effector in the global $\hat{\boldsymbol{x}}$ direction; the middle row will correspond to the movement in the global $\hat{\boldsymbol{y}}$ direction; the bottom row will correspond to the *rotation around the* $\hat{\boldsymbol{z}}$-axis. The Jacobian for *spatial movement* will have this form:

   $$\mathbf{J} = \begin{bmatrix} \hat{\boldsymbol{z}}_1 \times (\boldsymbol{p} - \boldsymbol{p}_1) & \hat{\boldsymbol{z}}_2 \times (\boldsymbol{p} - \boldsymbol{p}_2) & \hat{\boldsymbol{z}}_3 \times (\boldsymbol{p} - \boldsymbol{p}_3) \\ \hat{\boldsymbol{z}}_1 & \hat{\boldsymbol{z}}_2 & \hat{\boldsymbol{z}}_3 \end{bmatrix} \tag{1}$$

   where $\boldsymbol{p}$ is the *current* location of the end point on the planar robot, $\boldsymbol{p}_1$ is the *current* location of the first joint, $\boldsymbol{p}_2$ is the current location of the second joint, and $\boldsymbol{p}_3$ is the current location of the third joint. Note that $\boldsymbol{p}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\mathsf{T}$ and $\hat{\boldsymbol{z}}_1 = \hat{\boldsymbol{z}}_2 = \hat{\boldsymbol{z}}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\mathsf{T}$.

   The matrix above is of dimension 6×3. *We do not care how the arm moves translationally in the z-direction or rotationally in the x- or y-directions*. This means that your matrix should only consist of rows 1, 2, and 6 of the matrix above.

Your mission: Set the values of, and return, the Jacobian matrix. $\boldsymbol{p}$ can be obtained through the multiplication:

$$\boldsymbol{p} = {}_0\mathbf{T}_3 \cdot \boldsymbol{o}_3 \tag{2}$$

where ${}_0T_3$ is computable using your submission from `coord_frame_planar2.cpp` (the one difference is that you need to compute ${}_0T_3$ instead of ${}_0T_c$) and $\boldsymbol{o}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T}$. Similarly, you will compute $\boldsymbol{p}_2$ using:

$$\boldsymbol{p}_2 = {}_0\mathbf{T}_1 \cdot \boldsymbol{o}_1 \tag{3}$$

and $\boldsymbol{p}_1$ using:

$$\boldsymbol{p}_3 = {}_0\mathbf{T}_2 \cdot \boldsymbol{o}_2 \tag{4}$$

Debugging check: *Temporarily* modify `OnUpdate(.)` to check that the Jacobian has the following values for the corresponding values of θ_1, θ_2, and θ_3:

```
Joint angles: [0, 0, 0]
Jacobian:
0 0 0
4.2012 2.2012 0.8712
1 1 1

joint angles: [2.1374625, -0.66361164, 1.7787649]
Jacobian:
-2.9146209 -1.2272302 0.096524703
-1.8107452 -0.73710061 -0.86583626
1 1 1

joint angles: [1.875154, 2.5864566, -1.9003408]
Jacobian:
-1.0973559 0.81072338 -0.47767365
-1.6579832 -1.0586223 -0.72857211
1 1 1

joint angles: [-1.035326, 1.6853362, -1.3962827]
Jacobian:
1.506612 -0.21344541 0.59146341
2.718931 1.6984397 0.63965653
1 1 1

joint angles: [0.33910323, -0.14201851, 0.8097199]
Jacobian:
-1.6619877 -0.9967046 -0.73627554
3.6560721 1.7699654 0.46571211
1 1 1

joint angles: [-0.84958421, 0.084200402, 2.8414432]
Jacobian:
1.6611131 0.15910127 -0.76234111
1.8579823 0.53739137 -0.42169357
1 1 1

joint angles: [2.6150307, 0.85270194, 1.3653169]
Jacobian:
0.28585515 1.2909831 0.8648658
-2.884106 -1.1550259 0.1048646
```

```
1 1 1

joint angles: [-2.2518776, 0.67210527, -3.0391731]
Jacobian:
2.016332 0.46254718 -0.86739924
-1.3524943 -0.093227469 -0.081289603
1 1 1

joint angles: [-1.6154901, -2.2793412, 1.9111989]
Jacobian:
1.8862839 -0.11171888 0.79800758
-1.4090958 -1.3197381 -0.34953304
1 1 1

joint angles: [-2.1571489, -0.62238473, -2.3260952]
Jacobian:
1.3323121 -0.33361674 -0.80470343
-2.0165995 -0.9099464 0.33382903
1 1 1

joint angles: [-2.4579268, 3.1348352, -1.7702441]
Jacobian:
1.2039541 -0.059324436 0.77376906
-0.11343416 1.4370907 0.40033845
1 1 1
```

3. Output the Jacobians required by the assignment by running planar.world *with the original values for* $\boldsymbol{t}heta$ (*i.e.*, the values $\in \{0, \pi/2\}$) and running the plugin:

```
Jacobians $ cd build
build $ make
build $ cd ..
Jacobians $ gzserver --verbose planar.world
```