`*Patrick Star voice*` iBot, youBot, he-she-weBot (Capstone Final)

aka youBot a ferrari, but youDrive it like a fiat by *Nicole Baptist*

Overview

This ME449 Capstone Project code is used to manipulate a Kuka YouBot that picks up a block from one location and places it in another. The deliverables included in this package are as follows:

1. This `README` file! (also in pdf form)
2. A directory `code` containing:
    1. TrajectoryGenerator.py
    2. NextState.py
    3. FeedbackControl.py
    4. Mane.py*
3. A directory `results`** each containing:
    1. A brief background info `README` file noting the controller's type and feedback gains
    2. The robot data `.csv` file used to run the simulation
    3. A `video` of the simulation in action
    4. The corresponding X_err `.csv` file
    5. A `plot` of the X_err output over time (all 6 elements)
    6. A `log` file showing the program being called with the input*

**The results are organized as BEST, OVERSHOOT, and NEW_TASK

*The above deliverables can be achieved by running the `Mane.py` script in a code editor (such as VSCode by clicking the green `run` button in the top right corner).

This script incorporates a variety of Python libraries:

1. modern_robotics
2. numpy
3. logging
4. matplotlib
5. mrcapstone (custom made library containing the 3 milestone functions)

Please ensure that these libraries are installed before running `Mane.py`.

Ps: I can (mostly) spell don't worry

Instructions

---How to use this code--

This code can be run in any code editor (see the VSCode example above).

All variable required are built into the code and can be edited within the .py file.

Individual calls for each function can also be found in the respective function's docstring.

<!> Please remember to import the function before calling it! <!>

Background

Note: the inputs, outputs, etc. of each function can be found in their respective docstring.

TrajectoryGenerator.py:
Since using pandas was not my cup of tea, I rewrote it using the `CartesianTrajectory` functinon from `modern_robotics`. This calculates the transformation matrix of each step's end-effector.

NextState.py: Computes new configurations for the arms and wheels. Some examples:

- F = Hcross(0) = pseudo-inverse of H(0)
- new = old + (speeds * Δt) <--because of Euler step

FeedbackControl.py: Uses modern_robotics functions such as `JacobianBody` and `Adjoint` to calculate the commanded twist, speeds, and x_err of the simulation.

Mane.py Throws all the functions together. It also logs, saves csvs, and plots x_err data.

Surprises

- Could not get scene 6 running on coppeliaSim for the life of me. Wouldn't even be able to load a csv before an error popped up:

```
[CoppeliaSim:info] File was previously written with CoppeliaSim version 3.05.00
(rev 1) (CoppeliaSim Edu license) [CoppeliaSim:info] Scene opened.
[sandboxScript:info] Simulation started. [simExtCustomUI:warning] attribute name
'onclick' is deprecated. please use 'on-click' instead. [simExtCustomUI:warning]
unknown UI XML attribute 'value' in widget 'label' [youBot@childScript:error]
78: attempt to index local 'data' (a nil value) stack traceback: [string
"youBot@childScript"]:78: in function 'assignConfig' [string
"youBot@childScript"]:249: in main chunk
```

A bit of a bummer, but was able to use the V-REP player. Hopefully that won't cause any issues. (Could also copy the .csv directly into the scene directory.)

- YouBot dropped the block a bit. Adjusted by fiddling with variables to make the gripping period last longer.

Results

The best controller is the best, as it should be. It was chosen because it has no overshoot or oscillation. Although it is not particularly fast (as seen in the error plot), for this application, this slower, more robust controller is better than one that is fast with a lot of oscillation and overshoot. Because of this, it was also