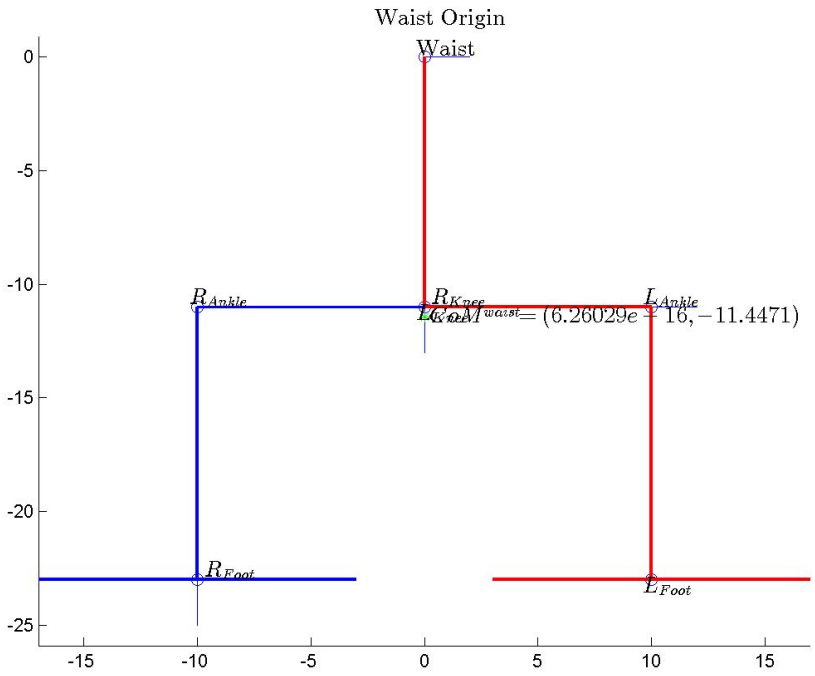
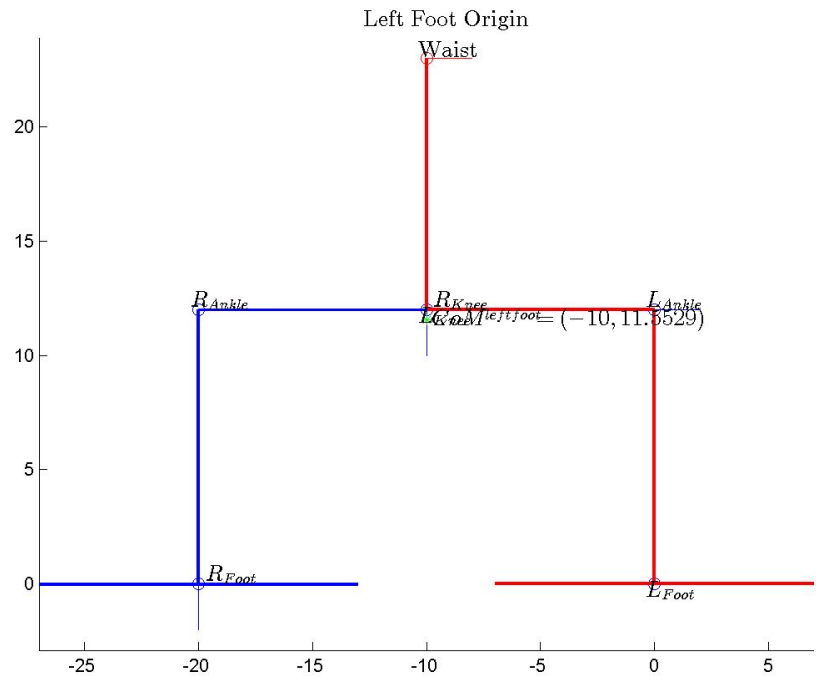
Week 1

1. Challenge Task
    a. Create a MechBot class in Matlab.
    b. Identify between three different frames (Left Foot/Right Foot/Waist)
    c. Write a function for forward kinematics code for each frame
    d. Create a planar version of the mechbot and visualize it using lines.
2. Solution Strategy
    a. Created a class definition called MechBot
    b. Created a constructor that takes in 3 angles for both the left and the right leg
        i. These angles are used to calculate an SE2 configuration
        ii. The lengths were hard coded in the constructor because they do not change
    c. Created a forward kinematics function
        i. First decide which frame the origin is located in
            - 0,1,2 = waist, left, right
        ii. Output is the forward kinematics at the two non-origin frames
    d. Created a plot function
        i. The function takes in the mechbot argument and the origin frame
            - plot(mechbot, frame)
        ii. Kinematics are used to calculate the location of each joint
        iii. Each SE2 joint configuration is plotted with the SE2 plot function
        iv. Lines are plotted between each of the configurations to visualize the limbs

Waist Origin



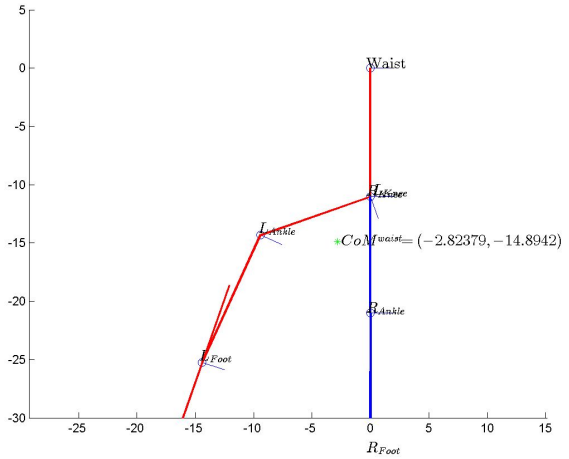
Left Foot Origin



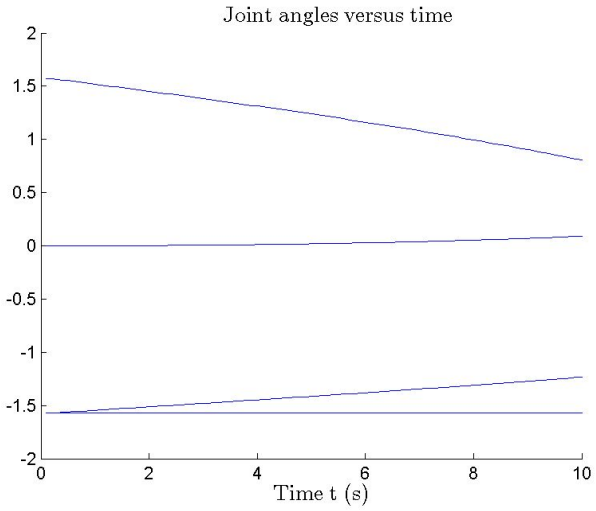
Week 2

1. Challenge Task
    a. Create and populate a function, jacobian(), in the MechBot class
    b. Perform resolved rate kinematics to move this leg in a prescribed trajectory
        i. Plots of joint angles vs. time
        ii. Plot foot position vs. time
        iii. For each time instant in your prescribed trajectory, update the plot with the robot's new joint configuration. This should create an animation/video of your biped executing your prescribed trajectory.
2. Solution Strategy
    a. Mechbot class takes in initial angle for each of the legs
    b. The angles are then calculated through resolved rate kinematics
        i. An array of changes in position from initial to final value is determined
        ii. Computed a change in position (change in position divided by time)
        iii. Computed the manipulator jacobian with the mechbot function by passing in mechbot arguments
        iv. Computed the pseudo inverse
        v. Computed the change in joint angles by multiplying the pseudo inverse and the position vector
        vi. Computed the next alpha value and stored it in an output array
        vii. Updated the bot configuration

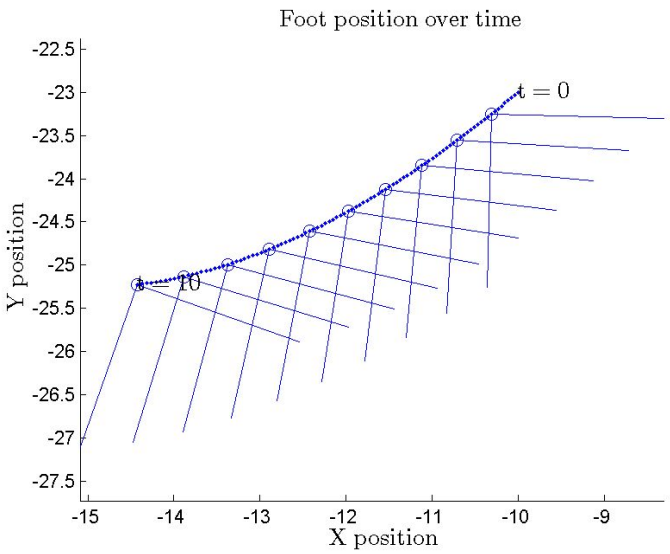
c. The final position of the mechbot by using the last set of angles from RRK
d. The joint angles vs time was saved into an array from the RRK
e. The foot configuration vs time was saved by substituting each joint angle into the SE2 configuration
f. The mechbot moving along it's trajectory is shown by using a loop that cycles through each of the joint angles generated by RRK and substitutes them into a mechbot argument
g. Each plot shows:
    i. Figure 1: Mechbot at the end of trajectory



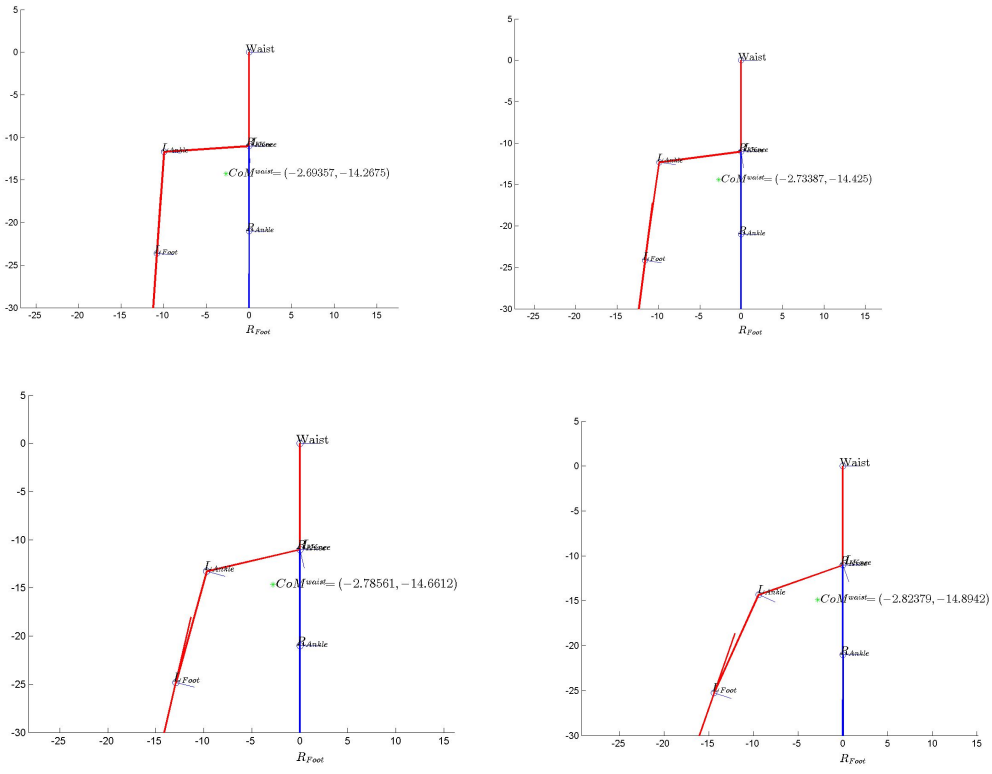
    ii. Figure 2: Joint angles vs time



    iii. Figure 3: Foot configuration over time



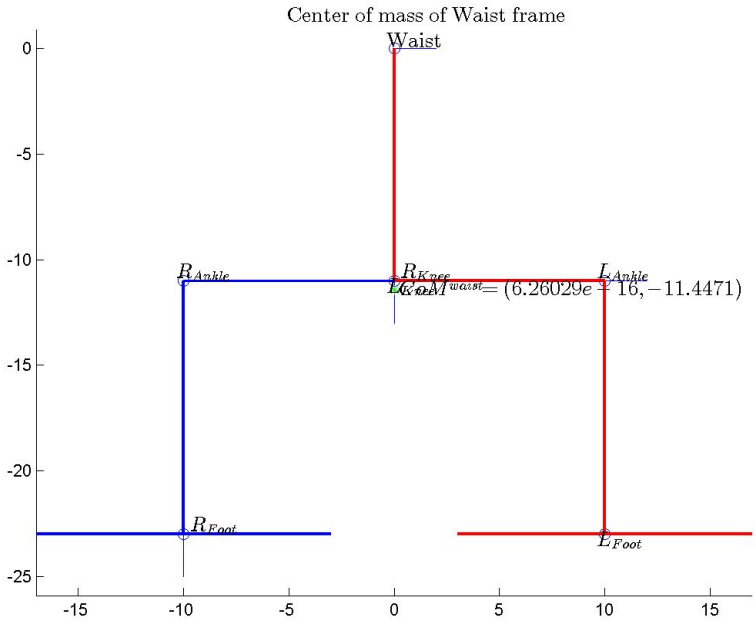
iv. Figure 4: Mechbot movement along trajectory

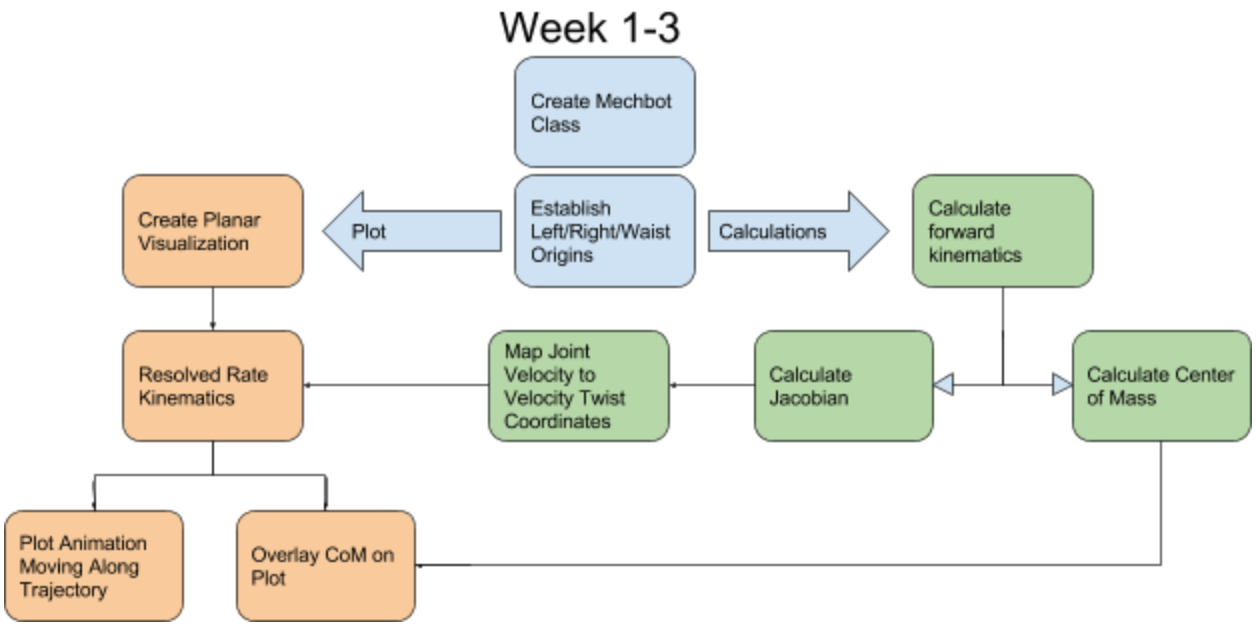


Week 3

1. Challenge Task
    a. Create a center of mass function, CoM(), for your robot.
    b. Update the 2-D stick figure visualization of the robot to also overlay the CoM location.
2. Solution Strategy

a. The center of mass function starts by computing the midpoints of each limb T
b. he center of mass location of each joint is calculated by multiplying the reference frame and the midpoints
c. Then the center of mass for the entire robot is calculated by multiplying the mass with the location and adding up each of the joints for both the x and y coordinate
d. The default reference frame is the waist and CoM is therefore calculated based off that, but a switch statement is used based on the stance argument and then kinematics are used to convert between frames
e. Plotting the center of mass was done by calling the CoM function inside the plot function and, assigning the correct stance to each origin frame





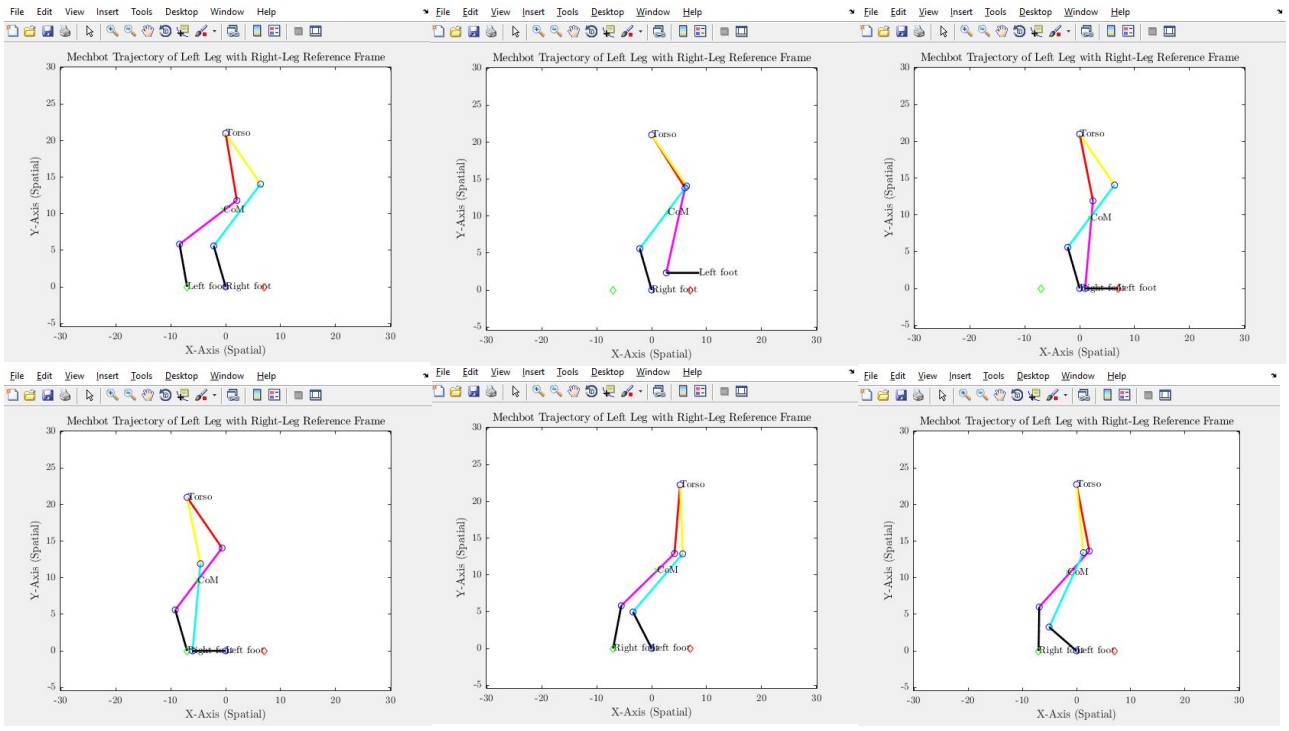
Week 4/5

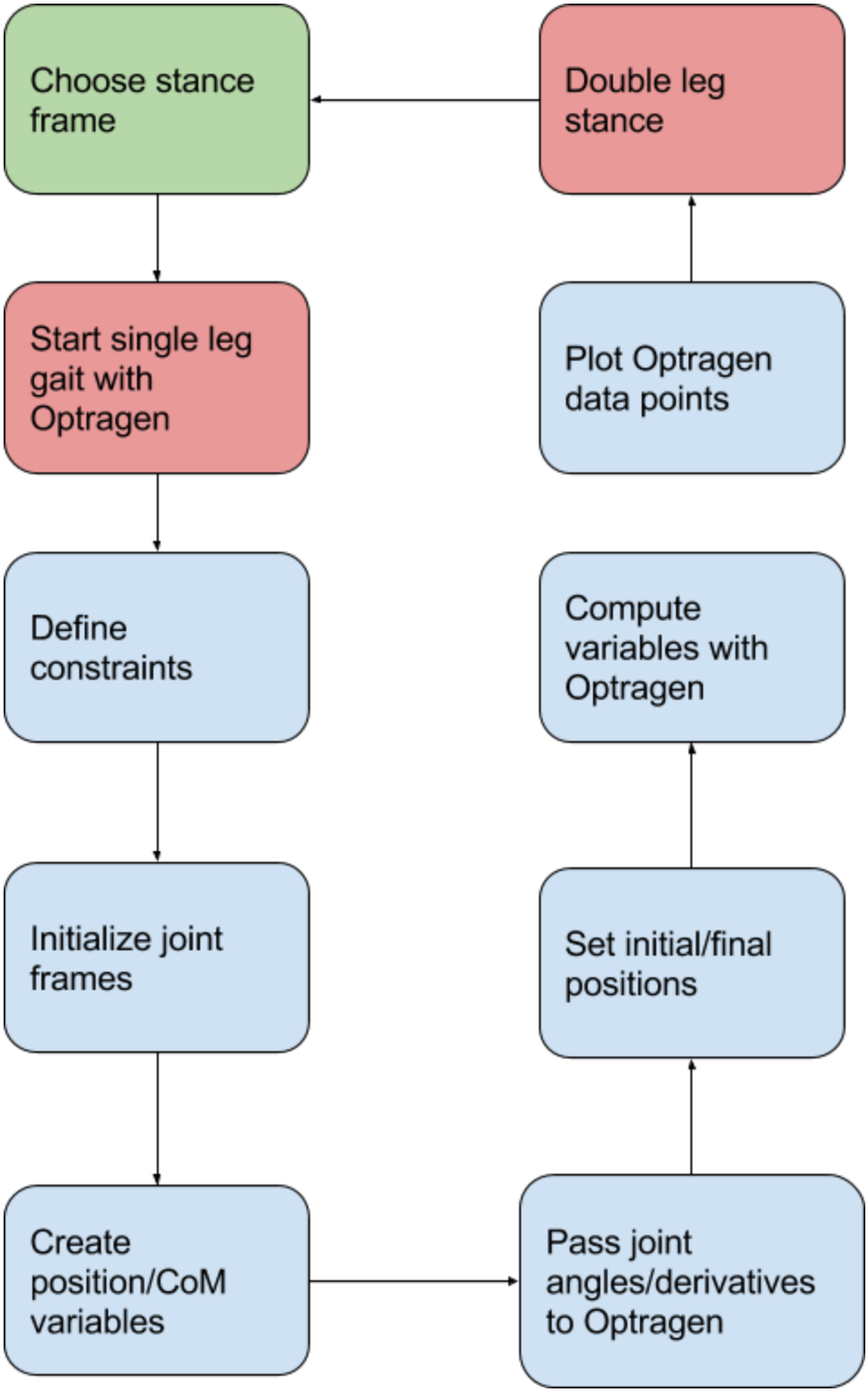
1. Challenge Task
    a. Use Optragen to generate the corresponding joint trajectory from the past assignments.
    b. Formulate the constraint that restricts the projection of the CoM (of the entire robot) to fall within a connected region on the ground
    c. Generate a single legged walking gait using Optragen
    d. After completing the single-legged gait, the biped will enter into a double-leg stance phase.
2. Solution Strategy
    a. Pick a stance frame and have the other foot walk forward
    b. The end effector frame begins on the ground and moves forward until it settles back on the ground
    c. At this point the mechbot needs to go into a double leg stance phase so that the CoM projection can transfer from one foot to the other
    d. The Optragen code follows this pattern:

i. Set up joint frames and create variables defining the walking foot, hip, knee etc
ii. Create variables defining the x and y positions of those variables, and for the center of mass. These variables are strings that are passed to the Optragen library
iii. Set the variables that should be passed to Optragen (joint angles and their derivatives)
iv. Set the desired initial and final positions
v. Create constraints for joint angles, initial and final positions/angles etc
vi. Pass everything to the Optragen library and let it compute everything
vii. Plot all of the data points that Optragen returns

e. The first section of the script computes and plots the trajectory of the left foot, with the right foot used as a reference frame.
i. The torso position is constrained to a fixed point above the right foot.
ii. Only one waypoint is used (the end position), and the leg naturally lifts itself into the air and moves to the destination.
iii. The CoM is constrained to within +/- 3 inches of the center of the right foot on the x-axis; this is the “projection” of the CoM onto the foot that was required by the assignment.

f. The second section of the script shifts the CoM from the right leg to the left leg to allow the robot to continue its gait.
i. The final joint angles that were computed previously are used as the initial joint angles, and the torso is constrained to move to a position directly above the left leg.
Additionally, the CoM’s constraint is changed to within +/- 3 inches of the center of the *left* foot, i.e., the new reference frame.



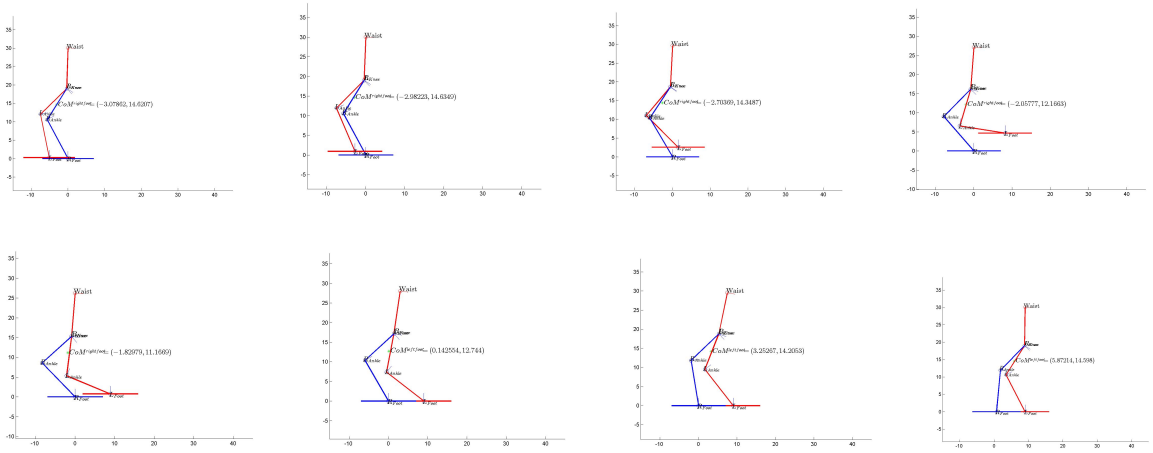


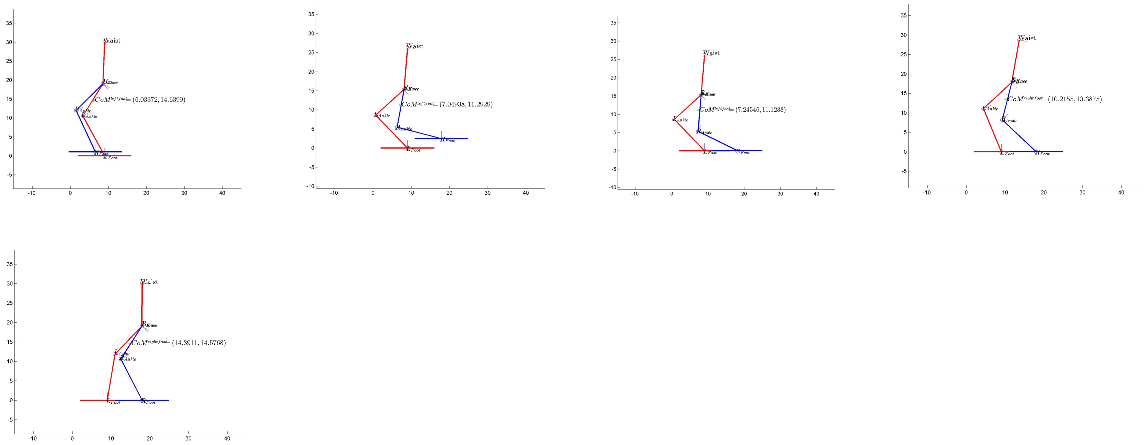
Week 6

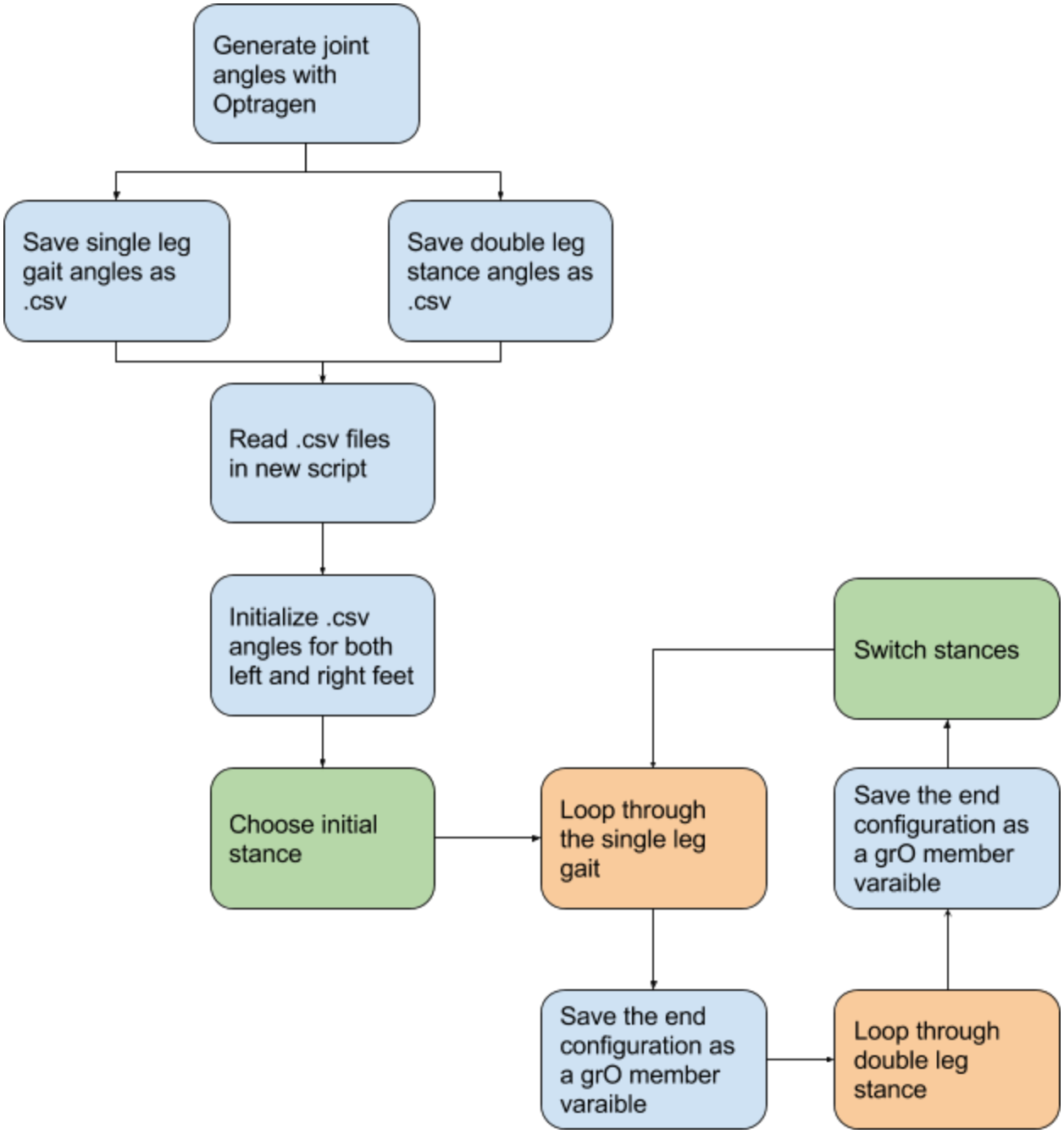
1. Challenge Task
    a. Create a grO member variable that keeps track of the spatial pose of the stance frame over multiple steps

b. Start in LEFT stance and right foot executes a single leg gait. The the right foot position relative to the stance frame is saved as an SE2 element
c. Forward kinematics are used with grO to save the frame location
d. RIGHT stance is now the frame and the left leg completes a single leg gait and is then saved into the grO member variable
e. Execute the single leg phase on the physical robot

2. Solution Strategy
    a. The Optragen from the previous week generates an array of angles along a trajectory.
    b. These angles are then saved into a .csv file
    c. In a new script this .csv file is read and the angles are saved into arrays for the left and right foot
    d. The for both the single gait and the rebalance are read and then saved into arrays for both left and right feet.
    e. Once the angles are saved, 4 loops run through each of the arrays
        i. LEFT stance, Double gait, RIGHT stance, Double gait
    f. The “MechWalk_animation.m” script animates a full walking cycle of the robot (movement of the left foot, rebalancing of the robot, movement of the right foot, and final rebalancing of the robot).
        i. To run the script, first execute “MechWalk_r.m” to generate the trajectory .csv files.
        ii. Then run the MechWalk_animation.m script to view the animation.
        iii. This contains frames for the left and right feet and the waist, as well as a plot of the center of mass of the robot.
    g. The single-legged stance of the robot can be executed by running “MechWalk_real.m”.
        i. The trajectory must first be generated by running MechWalk_r.m, as the motor control script uses the .csv files generated by the Optragen script to set the joint angles of the legs.
    h. The double-legged stance phase is also included in MechWalk_re




Waist
$CoM^{leftfoot} = (6.03372, 14.6399)$
Waist
$CoM^{leftfoot} = (7.04938, 11.2929)$
Waist
$CoM^{leftfoot} = (7.24546, 11.1238)$
Waist
$CoM^{rightfoot} = (10.2155, 13.3875)$
Waist
$CoM^{rightfoot} = (14.8911, 14.5768)$



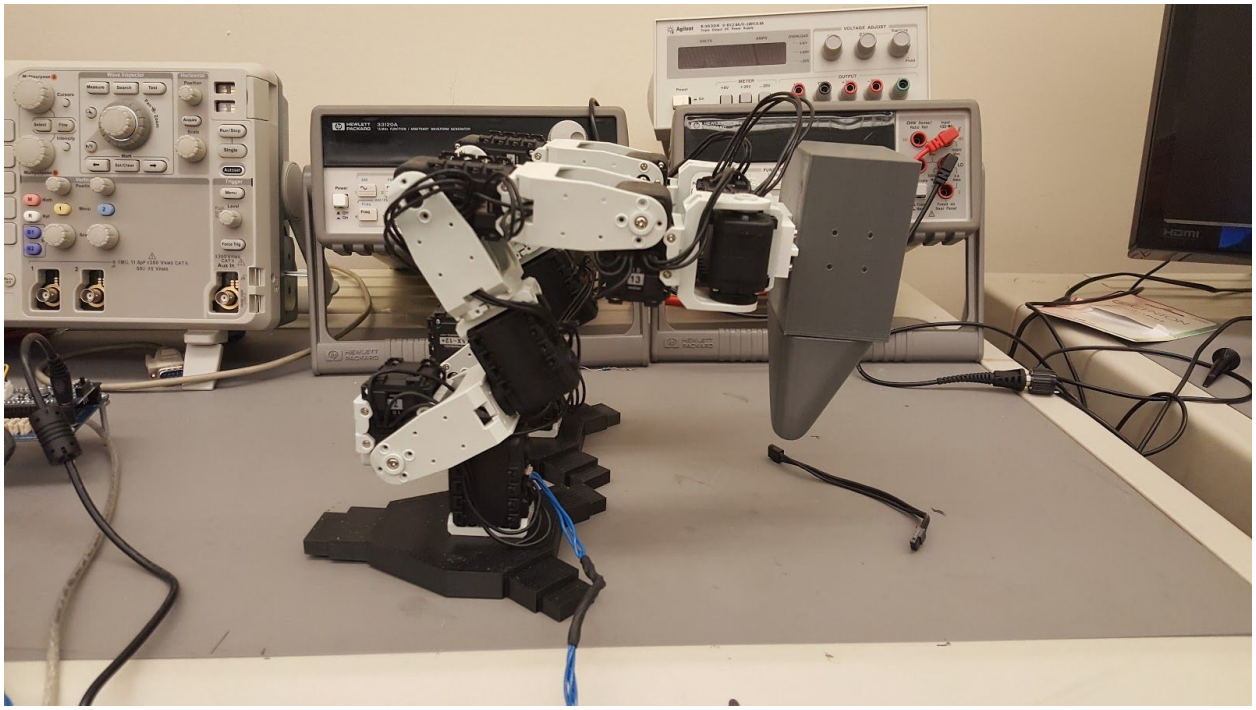
Week 7

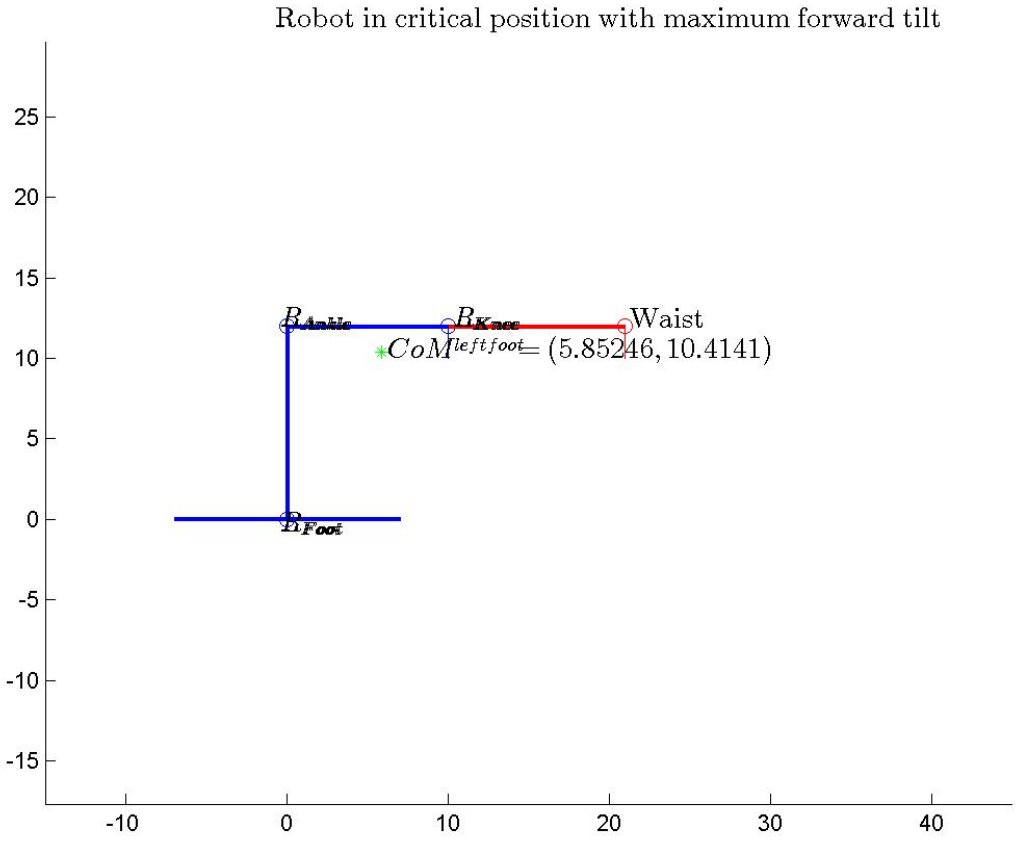
1. Challenge Task
    a. Execute the entire walking gait on your biped robot. Run it for 3 complete gait cycles (ie. left and right step constitutes one gait cycle) and (video) record the results. When demonstrating, hold the STANCE foot to the ground by hand.
    b. Establish some default/resting configuration of your robot (eg. standing 'straight').
    c. Bend the waist (via the hip) or knees forward until the CoM projection is at the forward edge of the support region in your animation/display. Save that joint

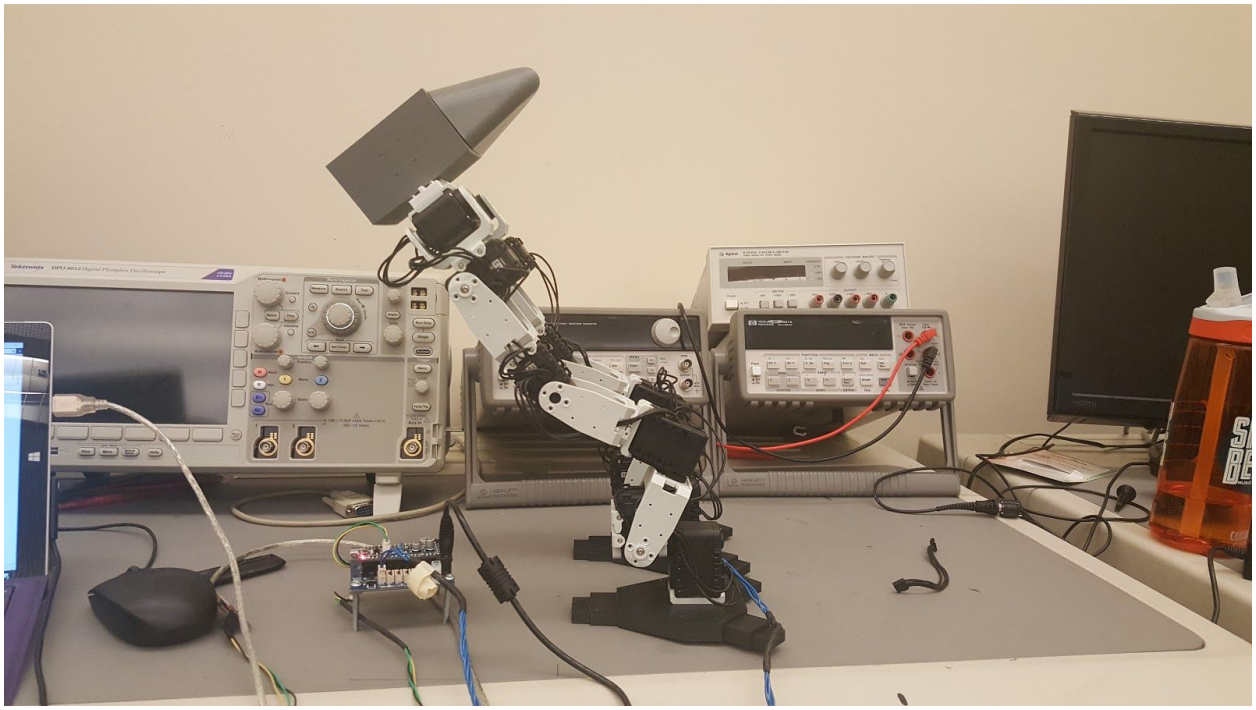
configuration, execute it on the robot and check that the physical robot is mechanically stable.

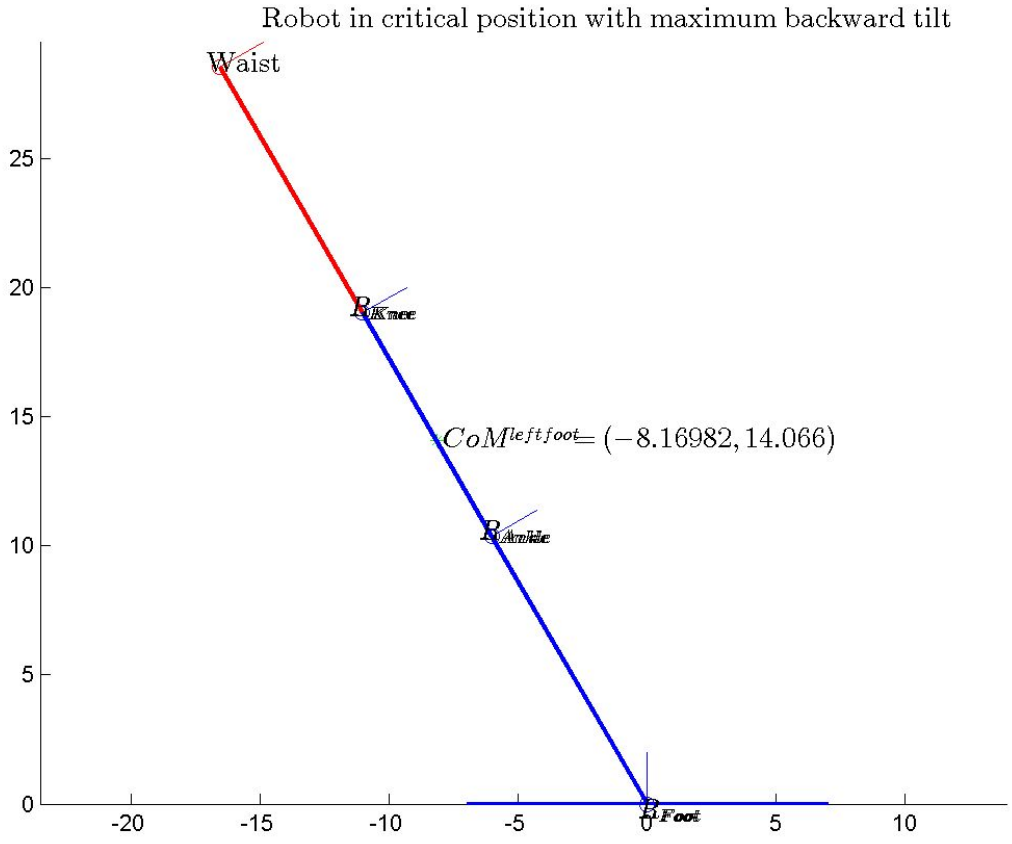
d. Starting from the resting configuration, bend the waist or knees backward until the CoM projection is at the back edge of the support region in your animation/display. Save that joint configuration, execute it on the robot and check that the physical robot is mechanically stable.
e. Bend the physical robot forward (or backward, respectively) a little more and make sure that it DOES tip over .

2. Solution Strategy
    a. Walking
        i. To execute the walking gait, the angles from the Optragen are saved and then exported into a Dynamixel code
        ii. For the robot to walk, an array of motor IDs for each of the robot legs needs to be addressed
        iii. Each joint angle is passed to it's respective motor joint and a loop is used to cycle through the full array
    b. Critical Angles
        i. Used Optragen to determine critical angles and saved angles as .csv files
        ii. Read in .csv files of angles generated from Optragen
        iii. Set the MechBot variables at the critical angle locations



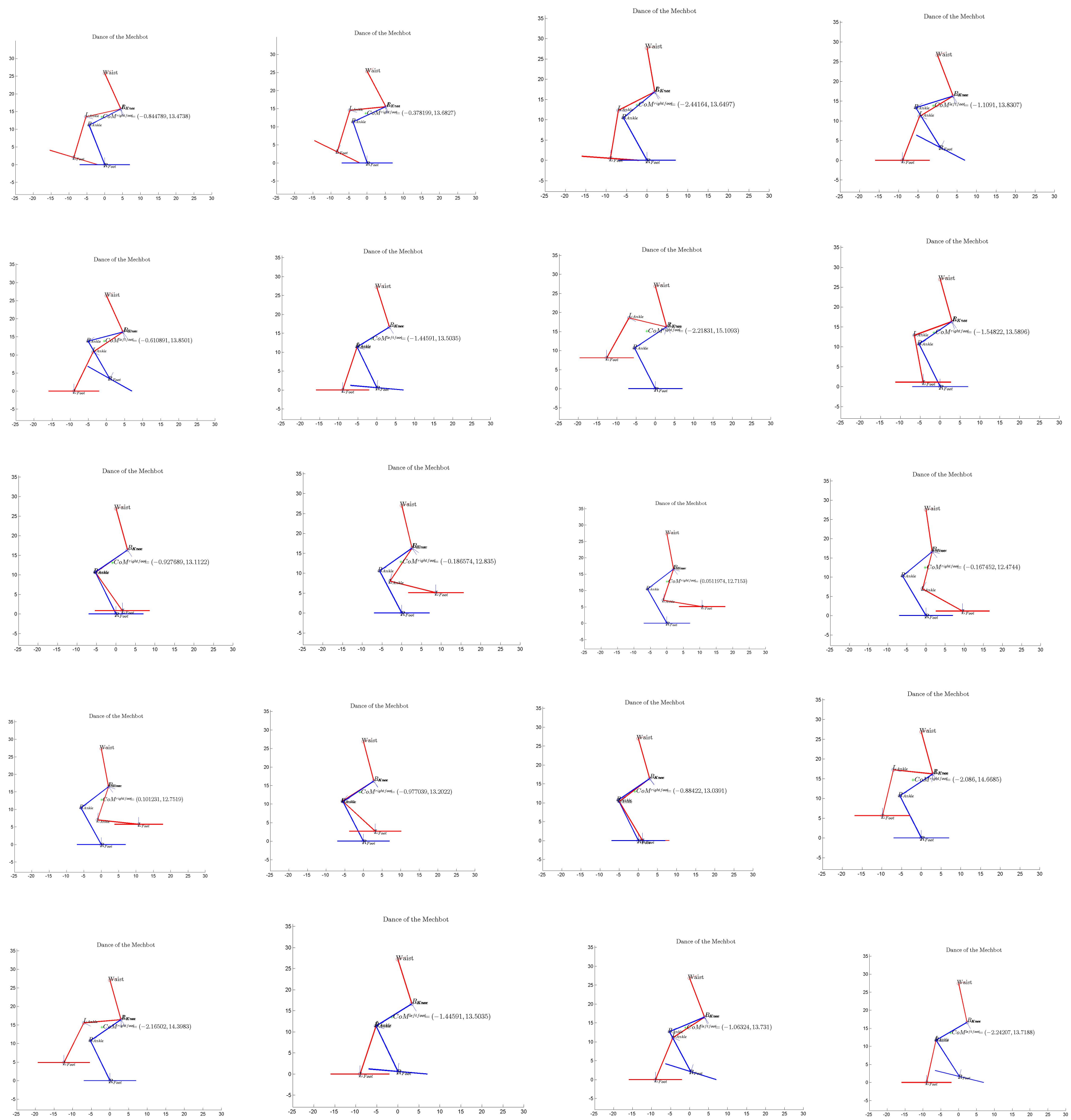

Robot in critical position with maximum forward tilt
25
20
15
10
5
0
-5
-10
-15
-10
0
10
20
30
40
B_{Ankle}
B_{Knee}
Waist
$CoM^{leftfoot} = (5.85246, 10.4141)$
B_{Foot}

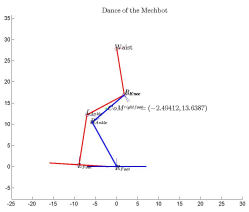


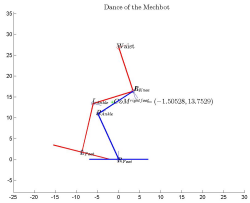


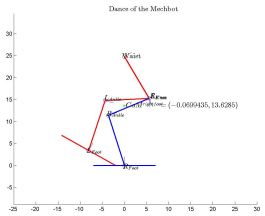
Week 8

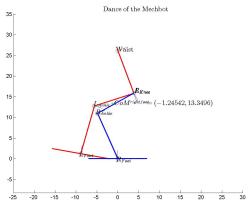
1. Challenge Task
    a. Create an animation of the robot dancing
2. Solution Strategy
    a. A series of trajectories were created in Optragen and then pieced together
    b. The robot first goes *en pointe* on the left foot, then on the right foot, and then steps forward with the left foot.
    c. This series of trajectories is then reversed, for the robot to return to its initial position.

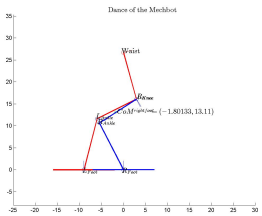

Dance of the Mechbot











Week 9

1. Challenge Task
    a. Attach the robot to the boom in the TSRB Automation Lab and demonstrate it executing a walking gait over ten cycles.
    b. Demonstrate the robot executing the dancing cycle designed in Week 8 while attached to the boom.
2. Solution
    a. Walking gait demonstration
        i. Previous trajectory modeling of the robot did not constrain the waist angle with respect to the origin frame, as we did not realize the boom would not allow the waist to rotate. Additionally, the end-effector trajectory had extreme curves and was difficult to predict.
        ii. The waist was constrained to a rotation of -pi/2 radians with respect to the origin frame, so that it was locked in a vertical position.
        iii. The end-effector trajectory was constrained to linear movements in a single dimension. To accomplish this, a second waypoint was added and the trajectory was constrained such that the end effector moved from point (-9, 0) to (-9, 6) to (9, 6) to (9, 0).This allowed the foot's movement to be easily predictable.
    b. Dance demonstration
        i. Originally, the dance's planar modeling in Optragen suffered from the same issues as the walking gait did. The Optragen scripts that generated the dance's trajectory was updated with the constraints determined in the walking gait.

Video of the walking gait and dance:
https://drive.google.com/open?id=0B0GBKyVUjWfqQW9PX1ZwWk9yd2s

Conclusion

As a group we have learned a lot about both modeling in matlab and understanding how robots work. We have all become much more accustomed to developing code with other people (Mechbot classdef) as well as understanding someone else's code and implementing it (Optragen). This project was an application of information that we learned from class and it went beyond the lectures because we were able to deal with a physical robot. The challenges involved with a physical robot included integrating theoretical concepts with code that made the motors run accordingly. One of the big challenges was figuring out how to integrate our matlab code with the Dynamixel code so that the actual motors were addressed correctly and worked as planned.

Some of our successes for this project were understanding the Optragen code, being able to plot the robot and show it move along it's trajectory, as well as developing a functioning class definition. One of our main challenges that was more difficult than anticipated was understanding the Optragen repository. When we were first introduced to Optragen we were given a link to its github and had to recreate the previous weeks challenges using the provided code. This proved very difficult because we had to spend a large amount of time sifting through the various code examples and applications to get a sense of what all of the different functions were used for. There was not a lot of documentation explaining what various functions did so a lot of trial and error took place to make it operational.

After finishing the project and having our robot able to walk with the assistance of the boon, the group as a whole has learned a lot about robot manipulation. This project's main goal, having the robot walk alone, seems simple enough but the groundwork to make this happen is very comprehensive. Just seeing the robot walk demonstrates all the key concepts that we as a group have learned throughout the semester.