```
# Lee Bradley, Martha Gizaw, Nate Winneg
# Engineering Physics Capstone Project
# Unstable Seniors: Data Processing
# May 2020

# Import the following libraries.
import matplotlib.pyplot as plt
import numpy as np
import csv
import math

class PlotThigh:
    # PURPOSE: Plot the total angle change about the axis of the thigh.

    # Initialize the Euler and quaternion CSV input variables as empty arrays.
    def __init__(self, xThighRoll = [], yThighRoll = [], xThighPitch = [],
                 yThighPitch = [], xThighYaw = [], yThighYaw = [],
                 xThighW = [], yThighW = [], xThighX = [], yThighX = [],
                 xThighY = [], yThighY = [], xThighZ = [], yThighZ = [],
                 changeThigh = []):
        # Euler angles
        self.xThighRoll = xThighRoll
        self.yThighRoll = yThighRoll
        self.xThighPitch = xThighPitch
        self.yThighPitch = yThighPitch
        self.xThighYaw = xThighYaw
        self.yThighYaw = yThighYaw

        # W, X, Y, and Z coordinates in a quaternion
        self.xThighW = xThighW
        self.yThighW = yThighW
        self.xThighX = xThighX
        self.yThighX = yThighX
        self.xThighY = xThighY
        self.yThighY = yThighY
        self.xThighZ = xThighZ
        self.yThighZ = yThighZ

        # For finding the net angles changes about the limb's axis
        self.changeThigh = changeThigh

    # Execute the CSV readers, and append the data to the appropriate arrays
    # for each Euler angle to be plotted.
    # For presentation purposes, set the Euler angles
    # to zero at the initial time of the user selected interval, where we can describe
    # the events of a single cycle of leg motion (eg, walking, sitting, etc.)
    def euler_angle_thigh(self, xThighRoll, yThighRoll, xThighPitch, yThighPitch,
    xThighYaw, yThighYaw):
        figThigh, axsThigh = plt.subplots(3, sharex = True, sharey = False)
        figThigh.suptitle('Euler Axis Rotations for THIGH')

        with open('angles_thigh_roll2.csv', 'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            for row in plots:
                xThighRoll.append(float(row[0]))
                yThighRoll.append(float(row[1]))
        setRoll2Zero = []
        for t in range(0, len(xThighRoll)):
            setRoll2Zero.append(yThighRoll[t]-yThighRoll[500])
        axsThigh[0].plot(xThighRoll, setRoll2Zero, linewidth = 2, color='teal')
        axsThigh[0].set(xlabel='', ylabel='Roll')
        axsThigh[0].set_xlim(500, 750)
        axsThigh[0].set_ylim(20, -20)

        with open('angles_thigh_pitch2.csv', 'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            for row in plots:
```

```
                xThighPitch.append(float(row[0]))
                yThighPitch.append(float(row[1]))
        setPitch2Zero = []
        for t in range(0, len(xThighPitch)):
            setPitch2Zero.append(yThighPitch[t]-yThighPitch[500])
        axsThigh[1].plot(xThighPitch, setPitch2Zero, linewidth = 2, color='magenta')
        axsThigh[1].set(xlabel='', ylabel='Pitch')
        axsThigh[1].set_xlim(500, 750)
        axsThigh[1].set_ylim(50, -50)

        with open('angles_thigh_yaw2.csv', 'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            for row in plots:
                xThighYaw.append(float(row[0]))
                yThighYaw.append(float(row[1]))
        setYaw2Zero = []
        for t in range(0, len(xThighYaw)):
            setYaw2Zero.append(yThighYaw[t]-yThighYaw[500])
        axsThigh[2].plot(xThighYaw, setYaw2Zero, linewidth = 2, color='black')
        axsThigh[2].set(xlabel='', ylabel='Yaw')
        axsThigh[2].set_xlim(500, 750)
        axsThigh[2].set_ylim(20, -20)

        # Optional!
        # figThigh.show()

    # Execute the CSV readers, and append the data to the appropriate arrays
    # for each quaternion to be plotted.
    def quaternion_thigh(self, xThighW, yThighW, xThighX, yThighX, xThighY,
                         yThighY, xThighZ, yThighZ):
        figQuats, axsQuats = plt.subplots(4, sharex = True, sharey = False)
        figQuats.suptitle('Quaternion Axis Rotations for THIGH')

        with open('angles_thigh_W2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xThighW.append(float(row[0]))
                yThighW.append(float(row[1]))
        axsQuats[0].plot(xThighW, yThighW, color='blue')
        axsQuats[0].set(xlabel='', ylabel='W')
        axsQuats[0].set_xlim(500, 750)

        with open('angles_thigh_X2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xThighX.append(float(row[0]))
                yThighX.append(float(row[1]))
        axsQuats[1].plot(xThighX, yThighX, color='red')
        axsQuats[1].set(xlabel='', ylabel='X')
        axsQuats[1].set_xlim(500, 750)

        with open('angles_thigh_Y2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xThighY.append(float(row[0]))
                yThighY.append(float(row[1]))
        axsQuats[2].plot(xThighY, yThighY, color='green')
        axsQuats[2].set(xlabel='', ylabel='Y')
        axsQuats[2].set_xlim(500, 750)

        with open('angles_thigh_Z2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xThighZ.append(float(row[0]))
                yThighZ.append(float(row[1]))
        axsQuats[3].plot(xThighZ, yThighZ, color='orange')
        axsQuats[3].set(xlabel='Time (count)', ylabel='Z')
```

```
134             axsQuats[3].set_xlim(500, 750)
135
136             # Optional!
137             # figQuats.show()
138
139         # Calculate 2 times the inverse cosine
140         # of the dot product between two quaternions, and convert the net angle
141         # change to degrees. Show the plots!
142         def dot_product_thigh(self, xThighW, yThighW, xThighX, yThighX, xThighY,
143                               yThighY, xThighZ, yThighZ, changeThigh):
144             oneRad2Degrees = 57.296
145             changeThighFix = []
146             for t1 in range(0, len(xThighW)):
147                 changeThigh.append(np.arccos(np.minimum(1, yThighW[0]*yThighW[t1] +
148                                                 yThighX[0]*yThighX[t1] +
149                                                 yThighY[0]*yThighY[t1] +
150
                                                 yThighZ[0]*yThighZ[t1]))*(180/np.pi)-(oneRad2D
                                                 egrees/2))
151
152             for t2 in range(0, len(xThighW)):
153                 changeThighFix.append(changeThigh[t2]-changeThigh[500])
154
155             fig, axs = plt.subplots()
156             axs.set_title('Quaternion-Based Net Angle Changes for THIGH')
157             axs.plot(changeThighFix, color='blue')
158             axs.set_xlim(500, 750)
159             axs.set_ylim(-20, 20)
160             axs.set(xlabel='Time (seconds)', ylabel='Total Angle Change (degrees)')
161             axs.invert_yaxis()
162             positions = (500, 550, 600, 650, 700, 750)
163             labels = (14.42, 15.86, 17.30, 18.75, 20.19, 21.63)
164             plt.xticks(positions, labels)
165             fig.show()
166
167         # Combine the Euler angles when more than one are changing significantly.
168         def euler_combo_thigh(self, xThighRoll, yThighRoll, yThighPitch, yThighYaw,
169                               yThighY, yThighZ):
170
171             # Initialize the following variables for Euler-based net angle changes.
172             theta_array = []
173             R = []
174
175             combinedEulerX = []
176             combinedEulerY = []
177             combinedEulerZ = []
178
179             combinedNetAngle = []
180             undoCombinedCos = []
181             undoCombinedSin = []
182
183             combinedQuatW = []
184             combinedQuatX = []
185             combinedQuatY = []
186             combinedQuatZ = []
187
188             rebuildCombined = []
189             rebuildCombinedFix = []
190
191             # Convert the original Euler angles into rotation matrices to be all
192             # multiplied.
193             for t3 in range(0, len(xThighRoll)):
194                 theta = [yThighRoll[t3] * (np.pi/180), yThighPitch[t3]* (np.pi/180),
                    yThighYaw[t3]* (np.pi/180)]
195                 theta_array.append(theta)
196
197                 R_x = np.array([[1,         0,                  0                   ],
```

```
                            [0,         math.cos(theta[0]), -math.sin(theta[0]) ],
                            [0,         math.sin(theta[0]), math.cos(theta[0])  ]
                            ])



            R_y = np.array([[math.cos(theta[1]),    0,      math.sin(theta[1])  ],
                            [0,                     1,      0                   ],
                            [-math.sin(theta[1]),   0,      math.cos(theta[1])  ]
                            ])

            R_z = np.array([[math.cos(theta[2]),    -math.sin(theta[2]),    0],
                            [math.sin(theta[2]),    math.cos(theta[2]),     0],
                            [0,                     0,                      1]
                            ])

            R.append(np.dot(R_x, np.dot(R_y, R_z)))

            # Report the new Euler rotations about their axes from the resultant
            # rotation matrix.
            combinedEulerX.append(math.atan2(R[t3][2,1], R[t3][2,2]))
            combinedEulerY.append(math.asin(R[t3][0,2]))
            combinedEulerZ.append(math.atan2(R[t3][1,0], R[t3][0,0]))

            # Obtain the cosine and sin of half of one of the new Euler rotations.
            combinedNetAngle.append(combinedEulerY[t3] * (180/np.pi))
            undoCombinedCos.append(np.cos(combinedNetAngle[t3] * (np.pi/360)))
            undoCombinedSin.append(np.sin(combinedNetAngle[t3] * (np.pi/360)))

            # Compute all 4 quaternion coordinates.
            combinedQuatW.append(undoCombinedCos[t3])
            combinedQuatX.append(undoCombinedSin[t3] * np.sin(0.5*np.pi) * np.cos(np.pi))
            combinedQuatY.append(0.01 * (yThighY[t3] / 2) *
            np.cos(yThighY[t3]/(undoCombinedSin[t3])))
            combinedQuatZ.append(0.01 * (yThighZ[t3] / 2) *
            np.cos(yThighZ[t3]/(undoCombinedSin[t3])))

            # Use the coordinates above to find the combined-Euler based net angle
            change.
            rebuildCombined.append(np.arccos(np.minimum(1,
                                combinedQuatW[0]*combinedQuatW[t3] +
                                combinedQuatX[0]*combinedQuatX[t3] +
                                combinedQuatY[0]*combinedQuatY[t3] +
                                combinedQuatZ[0]*combinedQuatZ[t3])) *(180/np.pi))

        # Set the net angle change to zero at the beginning of the plot interval.
        for t4 in range(0, len(xThighRoll)):
            rebuildCombinedFix.append(rebuildCombined[t4]-rebuildCombined[500])

        # Show the plots!
        fig, axs = plt.subplots()
        axs.set_title('Euler-Based Net Angle Changes for THIGH')
        axs.plot(rebuildCombinedFix, color='violet')
        axs.set_xlim(500, 750)
        axs.set_ylim(-20, 20)
        axs.set(xlabel='Time (seconds)', ylabel='Total Angle Change (degrees)')
        axs.invert_yaxis()
        positions = (500, 550, 600, 650, 700, 750)
        labels = (14.42, 15.86, 17.30, 18.75, 20.19, 21.63)
        plt.xticks(positions, labels)
        fig.show()

class PlotCalf:
    # PURPOSE: Plot the total angle change about the axis of the calf.

    def __init__(self, xCalfRoll = [], yCalfRoll = [], xCalfPitch = [],
                 yCalfPitch = [], xCalfYaw = [], yCalfYaw = [],
```

```
262                     xCalfW = [], yCalfW = [], xCalfX = [], yCalfX = [],
263                     xCalfY = [], yCalfY = [], xCalfZ = [], yCalfZ = [],
264                     changeCalf = []):
265             self.xCalfRoll = xCalfRoll
266             self.yCalfRoll = yCalfRoll
267             self.xCalfPitch = xCalfPitch
268             self.yCalfPitch = yCalfPitch
269             self.xCalfYaw = xCalfYaw
270             self.yCalfYaw = yCalfYaw
271 
272             self.xCalfW = xCalfW
273             self.yCalfW = yCalfW
274             self.xCalfX = xCalfX
275             self.yCalfX = yCalfX
276             self.xCalfY = xCalfY
277             self.yCalfY = yCalfY
278             self.xCalfZ = xCalfZ
279             self.yCalfZ = yCalfZ
280 
281             self.changeCalf = changeCalf
282 
283         def euler_angle_calf(self, xCalfRoll, yCalfRoll, xCalfPitch, yCalfPitch, xCalfYaw,
        yCalfYaw):
284             figCalf, axsCalf = plt.subplots(3, sharex = True, sharey = False)
285             figCalf.suptitle('Euler Axis Rotations for CALF')
286 
287             with open('angles_calf_roll2.csv', 'r') as csvfile:
288                 plots = csv.reader(csvfile, delimiter=',')
289                 for row in plots:
290                     xCalfRoll.append(float(row[0]))
291                     yCalfRoll.append(float(row[1]))
292             setRoll2Zero = []
293             for t in range(0, len(xCalfRoll)):
294                 setRoll2Zero.append(yCalfRoll[t]-yCalfRoll[500])
295             axsCalf[0].plot(xCalfRoll, setRoll2Zero, linewidth = 2, color='teal')
296             axsCalf[0].set(xlabel='', ylabel='Roll')
297             axsCalf[0].set_xlim(500, 750)
298             axsCalf[0].set_ylim(20, -20)
299 
300             with open('angles_calf_pitch2.csv', 'r') as csvfile:
301                 plots = csv.reader(csvfile, delimiter=',')
302                 for row in plots:
303                     xCalfPitch.append(float(row[0]))
304                     yCalfPitch.append(float(row[1]))
305             setPitch2Zero = []
306             for t in range(0, len(xCalfPitch)):
307                 setPitch2Zero.append(yCalfPitch[t]-yCalfPitch[500])
308             axsCalf[1].plot(xCalfPitch, setPitch2Zero, linewidth = 2, color='magenta')
309             axsCalf[1].set(xlabel='', ylabel='Pitch')
310             axsCalf[1].set_xlim(500, 750)
311             axsCalf[1].set_ylim(50, -50)
312 
313             with open('angles_calf_yaw2.csv', 'r') as csvfile:
314                 plots = csv.reader(csvfile, delimiter=',')
315                 for row in plots:
316                     xCalfYaw.append(float(row[0]))
317                     yCalfYaw.append(float(row[1]))
318             setYaw2Zero = []
319             for t in range(0, len(xCalfYaw)):
320                 setYaw2Zero.append(yCalfYaw[t]-yCalfYaw[500])
321             axsCalf[2].plot(xCalfYaw, setYaw2Zero, linewidth = 2, color='black')
322             axsCalf[2].set(xlabel='', ylabel='Yaw')
323             axsCalf[2].set_xlim(500, 750)
324             axsCalf[2].set_ylim(20, -20)
325 
326             # Optional!
327             # figCalf.show()
```

```
328
329     def quaternion_calf(self, xCalfW, yCalfW, xCalfX, yCalfX, xCalfY,
330                         yCalfY, xCalfZ, yCalfZ):
331         figQuats, axsQuats = plt.subplots(4, sharex = True, sharey = False)
332         figQuats.suptitle('Quaternion Axis Rotations for CALF')
333 
334         with open('angles_calf_W2.csv', 'r') as csvfile:
335             plots= csv.reader(csvfile, delimiter=',')
336             for row in plots:
337                 xCalfW.append(float(row[0]))
338                 yCalfW.append(float(row[1]))
339         axsQuats[0].plot(xCalfW, yCalfW, color='blue')
340         axsQuats[0].set(xlabel='', ylabel='W')
341         axsQuats[0].set_xlim(500, 750)
342 
343         with open('angles_calf_X2.csv', 'r') as csvfile:
344             plots= csv.reader(csvfile, delimiter=',')
345             for row in plots:
346                 xCalfX.append(float(row[0]))
347                 yCalfX.append(float(row[1]))
348         axsQuats[1].plot(xCalfX, yCalfX, color='red')
349         axsQuats[1].set(xlabel='', ylabel='X')
350         axsQuats[1].set_xlim(500, 750)
351 
352         with open('angles_calf_Y2.csv', 'r') as csvfile:
353             plots= csv.reader(csvfile, delimiter=',')
354             for row in plots:
355                 xCalfY.append(float(row[0]))
356                 yCalfY.append(float(row[1]))
357         axsQuats[2].plot(xCalfY, yCalfY, color='green')
358         axsQuats[2].set(xlabel='', ylabel='Y')
359         axsQuats[2].set_xlim(500, 750)
360 
361         with open('angles_calf_Z2.csv', 'r') as csvfile:
362             plots= csv.reader(csvfile, delimiter=',')
363             for row in plots:
364                 xCalfZ.append(float(row[0]))
365                 yCalfZ.append(float(row[1]))
366         axsQuats[3].plot(xCalfZ, yCalfZ, color='orange')
367         axsQuats[3].set(xlabel='Time (count)', ylabel='Z')
368         axsQuats[3].set_xlim(500, 750)
369 
370         # Optional!
371         # figQuats.show()
372 
373     def dot_product_calf(self, xCalfW, yCalfW, xCalfX, yCalfX, xCalfY,
374                          yCalfY, xCalfZ, yCalfZ, changeCalf):
375         oneRad2Degrees = 57.296
376         changeCalfFix = []
377         for t1 in range(0, len(xCalfW)):
378             changeCalf.append(np.arccos(np.minimum(1, yCalfW[0]*yCalfW[t1] +
379                                           yCalfX[0]*yCalfX[t1] +
380                                           yCalfY[0]*yCalfY[t1] +
381 
                                          yCalfZ[0]*yCalfZ[t1]))*(180/np.pi)-(oneRad2Deg
                                          rees/2))
382 
383         for t2 in range(0, len(xCalfW)):
384             changeCalfFix.append(changeCalf[t2]-changeCalf[500])
385 
386         fig, axs = plt.subplots()
387         axs.set_title('Quaternion-Based Net Angle Changes for CALF')
388         axs.plot(changeCalfFix, color='blue')
389         axs.set_xlim(500, 750)
390         axs.set_ylim(-20, 20)
391         axs.set(xlabel='Time (seconds)', ylabel='Total Angle Change (degrees)')
392         axs.invert_yaxis()
```

```
        positions = (500, 550, 600, 650, 700, 750)
        labels = (14.42, 15.86, 17.30, 18.75, 20.19, 21.63)
        plt.xticks(positions, labels)
        fig.show()

    def euler_combo_calf(self, xCalfRoll, yCalfRoll, yCalfPitch, yCalfYaw,
                         yCalfY, yCalfZ):
        theta_array = []
        R = []

        combinedEulerX = []
        combinedEulerY = []
        combinedEulerZ = []

        combinedNetAngle = []
        undoCombinedCos = []
        undoCombinedSin = []

        combinedQuatW = []
        combinedQuatX = []
        combinedQuatY = []
        combinedQuatZ = []

        rebuildCombined = []
        rebuildCombinedFix = []

        for t3 in range(0, len(xCalfRoll)):
            theta = [yCalfRoll[t3] * (np.pi/180), yCalfPitch[t3]* (np.pi/180),
            yCalfYaw[t3]* (np.pi/180)]
            theta_array.append(theta)

            R_x = np.array([[1,         0,                  0                   ],
                            [0,         math.cos(theta[0]), -math.sin(theta[0]) ],
                            [0,         math.sin(theta[0]), math.cos(theta[0])  ]
                            ])



            R_y = np.array([[math.cos(theta[1]),    0,      math.sin(theta[1])  ],
                            [0,                     1,      0                   ],
                            [-math.sin(theta[1]),   0,      math.cos(theta[1])  ]
                            ])

            R_z = np.array([[math.cos(theta[2]),    -math.sin(theta[2]),    0],
                            [math.sin(theta[2]),    math.cos(theta[2]),     0],
                            [0,                     0,                      1]
                            ])

            R.append(np.dot(R_x, np.dot(R_y, R_z)))

            combinedEulerX.append(math.atan2(R[t3][2,1], R[t3][2,2]))
            combinedEulerY.append(math.asin(R[t3][0,2]))
            combinedEulerZ.append(math.atan2(R[t3][1,0], R[t3][0,0]))

            combinedNetAngle.append(combinedEulerY[t3] * (180/np.pi))
            undoCombinedCos.append(np.cos(combinedNetAngle[t3] * (np.pi/360)))
            undoCombinedSin.append(np.sin(combinedNetAngle[t3] * (np.pi/360)))

            combinedQuatW.append(undoCombinedCos[t3])
            combinedQuatX.append(undoCombinedSin[t3] * np.sin(0.5*np.pi) * np.cos(np.pi))
            combinedQuatY.append(0.01 * (yCalfY[t3] / 2) *
            np.cos(yCalfY[t3]/(undoCombinedSin[t3])))
            combinedQuatZ.append(0.01 * (yCalfZ[t3] / 2) *
            np.cos(yCalfZ[t3]/(undoCombinedSin[t3])))

            rebuildCombined.append(np.arccos(np.minimum(1,
                                   combinedQuatW[0]*combinedQuatW[t3] +
```

```
                                    combinedQuatX[0]*combinedQuatX[t3] +
                                    combinedQuatY[0]*combinedQuatY[t3] +
                                    combinedQuatZ[0]*combinedQuatZ[t3])) *(180/np.pi))

        for t4 in range(0, len(xCalfRoll)):
            rebuildCombinedFix.append(rebuildCombined[t4]-rebuildCombined[500])

        fig, axs = plt.subplots()
        axs.set_title('Euler-Based Net Angle Changes for CALF')
        axs.plot(rebuildCombinedFix, color='violet')
        axs.set_xlim(500, 750)
        axs.set_ylim(-20, 20)
        axs.set(xlabel='Time (seconds)', ylabel='Total Angle Change (degrees)')
        axs.invert_yaxis()
        positions = (500, 550, 600, 650, 700, 750)
        labels = (14.42, 15.86, 17.30, 18.75, 20.19, 21.63)
        plt.xticks(positions, labels)
        fig.show()

class PlotFoot:
    # PURPOSE: Plot the total angle change about the axis of the foot.

    def __init__(self, xFootRoll = [], yFootRoll = [], xFootPitch = [],
                 yFootPitch = [], xFootYaw = [], yFootYaw = [],
                 xFootW = [], yFootW = [], xFootX = [], yFootX = [],
                 xFootY = [], yFootY = [], xFootZ = [], yFootZ = [],
                 changeFoot = []):
        self.xFootRoll = xFootRoll
        self.yFootRoll = yFootRoll
        self.xFootPitch = xFootPitch
        self.yFootPitch = yFootPitch
        self.xFootYaw = xFootYaw
        self.yFootYaw = yFootYaw

        self.xFootW = xFootW
        self.yFootW = yFootW
        self.xFootX = xFootX
        self.yFootX = yFootX
        self.xFootY = xFootY
        self.yFootY = yFootY
        self.xFootZ = xFootZ
        self.yFootZ = yFootZ

        self.changeFoot = changeFoot

    def euler_angle_foot(self, xFootRoll, yFootRoll, xFootPitch, yFootPitch, xFootYaw,
    yFootYaw):
        figFoot, axsFoot = plt.subplots(3, sharex = True, sharey = False)
        figFoot.suptitle('Euler Axis Rotations for FOOT')

        with open('angles_foot_roll2.csv', 'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootRoll.append(float(row[0]))
                yFootRoll.append(float(row[1]))
        setRoll2Zero = []
        for t in range(0, len(xFootRoll)):
            setRoll2Zero.append(yFootRoll[t]-yFootRoll[500])
        axsFoot[0].plot(xFootRoll, setRoll2Zero, linewidth = 2, color='teal')
        axsFoot[0].set(xlabel='', ylabel='Roll')
        axsFoot[0].set_xlim(500, 750)
        axsFoot[0].set_ylim(20, -20)

        with open('angles_foot_pitch2.csv', 'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootPitch.append(float(row[0]))
```

```
                yFootPitch.append(float(row[1]))
        setPitch2Zero = []
        for t in range(0, len(xFootPitch)):
            setPitch2Zero.append(yFootPitch[t]-yFootPitch[500])
        axsFoot[1].plot(xFootPitch, setPitch2Zero, linewidth = 2, color='magenta')
        axsFoot[1].set(xlabel='', ylabel='Pitch')
        axsFoot[1].set_xlim(500, 750)
        axsFoot[1].set_ylim(50, -50)

        with open('angles_foot_yaw2.csv', 'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootYaw.append(float(row[0]))
                yFootYaw.append(float(row[1]))
        setYaw2Zero = []
        for t in range(0, len(xFootYaw)):
            setYaw2Zero.append(yFootYaw[t]-yFootYaw[500])
        axsFoot[2].plot(xFootYaw, setYaw2Zero, linewidth = 2, color='black')
        axsFoot[2].set(xlabel='', ylabel='Yaw')
        axsFoot[2].set_xlim(500, 750)
        axsFoot[2].set_ylim(20, -20)

        # Optional!
        # figFoot.show()

    def quaternion_foot(self, xFootW, yFootW, xFootX, yFootX, xFootY,
                        yFootY, xFootZ, yFootZ):
        figQuats, axsQuats = plt.subplots(4, sharex = True, sharey = False)
        figQuats.suptitle('Quaternion Axis Rotations for FOOT')

        with open('angles_foot_W2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootW.append(float(row[0]))
                yFootW.append(float(row[1]))
        axsQuats[0].plot(xFootW, yFootW, color='blue')
        axsQuats[0].set(xlabel='', ylabel='W')
        axsQuats[0].set_xlim(500, 750)

        with open('angles_foot_X2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootX.append(float(row[0]))
                yFootX.append(float(row[1]))
        axsQuats[1].plot(xFootX, yFootX, color='red')
        axsQuats[1].set(xlabel='', ylabel='X')
        axsQuats[1].set_xlim(500, 750)

        with open('angles_foot_Y2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootY.append(float(row[0]))
                yFootY.append(float(row[1]))
        axsQuats[2].plot(xFootY, yFootY, color='green')
        axsQuats[2].set(xlabel='', ylabel='Y')
        axsQuats[2].set_xlim(500, 750)

        with open('angles_foot_Z2.csv', 'r') as csvfile:
            plots= csv.reader(csvfile, delimiter=',')
            for row in plots:
                xFootZ.append(float(row[0]))
                yFootZ.append(float(row[1]))
        axsQuats[3].plot(xFootZ, yFootZ, color='orange')
        axsQuats[3].set(xlabel='Time (count)', ylabel='Z')
        axsQuats[3].set_xlim(500, 750)

        # Optional!
```

```
590         # figQuats.show()
591
592     def dot_product_foot(self, xFootW, yFootW, xFootX, yFootX, xFootY,
593                          yFootY, xFootZ, yFootZ, changeFoot):
594         oneRad2Degrees = 57.296
595         changeFootFix = []
596         for t1 in range(0, len(xFootW)):
597             changeFoot.append(np.arccos(np.minimum(1, yFootW[0]*yFootW[t1] +
598                                         yFootX[0]*yFootX[t1] +
599                                         yFootY[0]*yFootY[t1] +
600
                                        yFootZ[0]*yFootZ[t1]))*(180/np.pi)-(oneRad2Deg
                                        rees/2))
601
602         for t2 in range(0, len(xFootW)):
603             changeFootFix.append(changeFoot[t2]-changeFoot[500])
604
605         fig, axs = plt.subplots()
606         axs.set_title('Quaternion-Based Net Angle Changes for FOOT')
607         axs.plot(changeFootFix, color='blue')
608         axs.set_xlim(500, 750)
609         axs.set_ylim(-20, 20)
610         axs.set(xlabel='Time (seconds)', ylabel='Total Angle Change (degrees)')
611         axs.invert_yaxis()
612         positions = (500, 550, 600, 650, 700, 750)
613         labels = (14.42, 15.86, 17.30, 18.75, 20.19, 21.63)
614         plt.xticks(positions, labels)
615         fig.show()
616
617     def euler_combo_foot(self, xFootRoll, yFootRoll, yFootPitch, yFootYaw,
618                          yFootY, yFootZ):
619         theta_array = []
620         R = []
621
622         combinedEulerX = []
623         combinedEulerY = []
624         combinedEulerZ = []
625
626         combinedNetAngle = []
627         undoCombinedCos = []
628         undoCombinedSin = []
629
630         combinedQuatW = []
631         combinedQuatX = []
632         combinedQuatY = []
633         combinedQuatZ = []
634
635         rebuildCombined = []
636         rebuildCombinedFix = []
637
638         for t3 in range(0, len(xFootRoll)):
639             theta = [yFootRoll[t3] * (np.pi/180), yFootPitch[t3]* (np.pi/180),
                    yFootYaw[t3]* (np.pi/180)]
640             theta_array.append(theta)
641
642             R_x = np.array([[1,         0,                  0                   ],
643                             [0,         math.cos(theta[0]), -math.sin(theta[0]) ],
644                             [0,         math.sin(theta[0]), math.cos(theta[0])  ]
645                             ])
646
647
648
649             R_y = np.array([[math.cos(theta[1]),    0,      math.sin(theta[1])  ],
650                             [0,                     1,      0                   ],
651                             [-math.sin(theta[1]),   0,      math.cos(theta[1])  ]
652                             ])
653
```

```
654             R_z = np.array([[math.cos(theta[2]),    -math.sin(theta[2]),    0],
655                             [math.sin(theta[2]),    math.cos(theta[2]),     0],
656                             [0,                     0,                      1]
657                             ])
658 
659             R.append(np.dot(R_x, np.dot(R_y, R_z)))
660 
661             combinedEulerX.append(math.atan2(R[t3][2,1], R[t3][2,2]))
662             combinedEulerY.append(math.asin(R[t3][0,2]))
663             combinedEulerZ.append(math.atan2(R[t3][1,0], R[t3][0,0]))
664 
665             combinedNetAngle.append(combinedEulerY[t3] * (180/np.pi))
666             undoCombinedCos.append(np.cos(combinedNetAngle[t3] * (np.pi/360)))
667             undoCombinedSin.append(np.sin(combinedNetAngle[t3] * (np.pi/360)))
668 
669             combinedQuatW.append(undoCombinedCos[t3])
670             combinedQuatX.append(undoCombinedSin[t3] * np.sin(0.5*np.pi) * np.cos(np.pi))
671             combinedQuatY.append(0.01 * (yFootY[t3] / 2) *
                np.cos(yFootY[t3]/(undoCombinedSin[t3])))
672             combinedQuatZ.append(0.01 * (yFootZ[t3] / 2) *
                np.cos(yFootZ[t3]/(undoCombinedSin[t3])))
673 
674             rebuildCombined.append(np.arccos(np.minimum(1,
675                                 combinedQuatW[0]*combinedQuatW[t3] +
676                                 combinedQuatX[0]*combinedQuatX[t3] +
677                                 combinedQuatY[0]*combinedQuatY[t3] +
678                                 combinedQuatZ[0]*combinedQuatZ[t3])) *(180/np.pi))
679 
680         for t4 in range(0, len(xFootRoll)):
681             rebuildCombinedFix.append(rebuildCombined[t4]-rebuildCombined[500])
682 
683         fig, axs = plt.subplots()
684         axs.set_title('Euler-Based Net Angle Changes for FOOT')
685         axs.plot(rebuildCombinedFix, color='violet')
686         axs.set_xlim(500, 750)
687         axs.set_ylim(-20, 20)
688         axs.set(xlabel='Time (seconds)', ylabel='Total Angle Change (degrees)')
689         axs.invert_yaxis()
690         positions = (500, 550, 600, 650, 700, 750)
691         labels = (14.42, 15.86, 17.30, 18.75, 20.19, 21.63)
692         plt.xticks(positions, labels)
693         fig.show()
694 
695 class PlotLegRaising:
696     # PURPOSE: Plot the total angle change for when a person is sitting and
697     # raising a leg by up to 90 degrees.
698 
699     def __init__(self, xLegW = [], yLegW = [], xLegX = [], yLegX = [],
700                  xLegY = [], yLegY = [], xLegZ = [], yLegZ = [],
701                  changeLeg = [], pointsMinMax = []):
702         self.xLegW = xLegW
703         self.yLegW = yLegW
704         self.xLegX = xLegX
705         self.yLegX = yLegX
706         self.xLegY = xLegY
707         self.yLegY = yLegY
708         self.xLegZ = xLegZ
709         self.yLegZ = yLegZ
710 
711         self.changeLeg = changeLeg
712         self.pointsMinMax = pointsMinMax
713 
714     def leg_quat_analysis(self, xLegW, yLegW, xLegX, yLegX, xLegY,
715                           yLegY, xLegZ, yLegZ):
716         figLeg, axsLeg = plt.subplots(4, sharex = True, sharey = False)
717         figLeg.suptitle('Sitting/Leg Raising Quaternions')
718 
```

```
719             with open('test_quat_wholeleg_w.csv', 'r') as csvfile:
720                 plots= csv.reader(csvfile, delimiter=',')
721                 for row in plots:
722                     xLegW.append(float(row[0]))
723                     yLegW.append(float(row[1]))
724             axsLeg[0].plot(xLegW,yLegW,linewidth=2, color='teal')
725             axsLeg[0].set(xlabel='', ylabel="Quat'n (W)")
726             axsLeg[0].set_xlim(50, 100)
727 
728             with open('test_quat_wholeleg_x.csv', 'r') as csvfile:
729                 plots= csv.reader(csvfile, delimiter=',')
730                 for row in plots:
731                     xLegX.append(float(row[0]))
732                     yLegX.append(float(row[1]))
733             axsLeg[1].plot(xLegX,yLegX,linewidth=2, color='red')
734             axsLeg[1].set(xlabel='', ylabel="Quat'n (X)")
735             axsLeg[1].set_xlim(50, 100)
736 
737             with open('test_quat_wholeleg_y.csv', 'r') as csvfile:
738                 plots= csv.reader(csvfile, delimiter=',')
739                 for row in plots:
740                     xLegY.append(float(row[0]))
741                     yLegY.append(float(row[1]))
742             axsLeg[2].plot(xLegY,yLegY,linewidth=2, color='green')
743             axsLeg[2].set(xlabel='', ylabel="Quat'n (Y)")
744             axsLeg[2].set_xlim(50, 100)
745 
746             with open('test_quat_wholeleg_z.csv', 'r') as csvfile:
747                 plots= csv.reader(csvfile, delimiter=',')
748                 for row in plots:
749                     xLegZ.append(float(row[0]))
750                     yLegZ.append(float(row[1]))
751             axsLeg[3].plot(xLegZ,yLegZ,linewidth=2, color='orange')
752             axsLeg[3].set(xlabel='Time (Count)', ylabel="Quat'n (Z)")
753             axsLeg[3].set_xlim(50, 100)
754             # Display the matplotlab figure showing quaternion behaviors in the
755             # raising leg (optional).
756             # figLeg.show()
757 
758         def leg_net_angles(self, xLegW, yLegW, xLegX, yLegX, xLegY,
759                                yLegY, xLegZ, yLegZ, changeLeg, pointsMinMax):
760             # Append changeLeg with the total angle change, which equates to
761             # the inverse cosine of the dot product for each quaternion at the
762             # initial and final time points, all multiplied by 360 degrees over
763             # pi (for converting from radians to degrees).
764             for t1 in range(0, len(xLegW)):
765                 changeLeg.append(np.arccos(np.minimum(1, yLegW[0] * yLegW[t1] +
766                                 yLegX[0] * yLegX[t1] +
767                                 yLegY[0] * yLegY[t1] +
768                                 yLegZ[0] * yLegZ[t1]))*(360/np.pi))
769 
770             # Plot the total angle change for the raising leg based on the
771             # quaternions using the changeLeg array. Limit the x-axis to
772             # 50-100, and invert and limit the y-axis to 90-0.
773             figAngleLeg, axsAngleLeg = plt.subplots()
774             axsAngleLeg.set_title('Sitting/Leg-Raising')
775             axsAngleLeg.set_ylabel("Total Angle Change (Degrees)")
776             axsAngleLeg.set_xlabel('Time (Count)')
777             axsAngleLeg.plot(changeLeg, color='blue')
778             axsAngleLeg.set_xlim(50, 100)
779             axsAngleLeg.set_ylim(90, 0)
780 
781             # Narrow down the time interval to 50-100, and append pointMinMax with
782             # the y-values occuring within that interval.
783             for t2 in range(50, 101):
784                 pointsMinMax.append(changeLeg[t2])
785 
```

```
786             # Determine the highest and lowest values of pointMinMax, and find their
787             # locations within the x-axis.
788             xmax = pointsMinMax.index(max(pointsMinMax))+50
789             ymax = max(pointsMinMax)
790             xmin = pointsMinMax.index(min(pointsMinMax))+50
791             ymin = min(pointsMinMax)
792
793             # Annotate the highest point in the plot within the selected interval.
794             text1= "Max Angle: {:.3f}° \nTime: {:.3f}".format(ymax, xmax)
795             bbox_props1 = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
796             arrowprops1=dict(arrowstyle="->", lw=1.5)
797             kw1 = dict(xycoords='data',textcoords="axes fraction",
798                       arrowprops=arrowprops1, bbox=bbox_props1, ha="left", va="top")
799             axsAngleLeg.annotate(text1, xytext=(0.4, 0.0925), xy=(xmax, ymax), **kw1)
800
801             # Annotate the lowest point in the plot within the selected interval.
802             text2= "Min Angle: {:.3f}° \nTime: {:.3f}".format(ymin, xmin)
803             bbox_props2 = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
804             arrowprops2=dict(arrowstyle="->", lw=1.5)
805             kw2 = dict(xycoords='data',textcoords="axes fraction",
806                       arrowprops=arrowprops2, bbox=bbox_props2, ha="left", va="bottom")
807             axsAngleLeg.annotate(text2, xytext=(0.18, 0.85), xy=(xmin, ymin), **kw2)
808
809             # Display the matplotlab figure showing the total angle change in the
810             # raising leg.
811             figAngleLeg.show()
812
813
```