CESPA Summer Course – working with QTM and implementing data analysis in Python

Exercise 1

In this first exercise, you are been provided with a file containing data of right ankle and right knee displacements during three strides of walking of an adult. The file contains seven columns: time, x, y, and z-axes of knee, and x, y, and z-axes of ankle. Here is adopted the convention of Jeff's lab and z-axis is parallel to gravity. Calculate:

1. Linear velocity of the ankle.
2. Linear acceleration of the ankle.
3. 2D Angle between the leg and the floor.
4. Angular velocity of the ankle. (Consider the knee marker as the rotation axis).
    a. Calculate it using only x and z-axes.
    b. Calculate it using x, y, and z-axes.
    c. Plot both results. Are they different? If yes, why?
5. Angular acceleration of the ankle relative to the ground using result of item 4.b.

Hints (each number refers to general suggestions – additional comments and guidelines are given on letters)

1. Open file in ipython notebook
    a. Import *os* library to change directory to where you saved the data file.
    b. Use the command *os.chdir* ("directory where the file is") to change the directory.
    c. Import *numpy* and use *numpy.loadtxt* to load the file in a variable. *Use delimiter = "\t", dtype = "f", and skiprows = 11*. For details, see documentation of the function.
2. Linear velocity from 3D vectors is given by: $\vec{v} = \dot{v_x} + \dot{y_y} + \dot{v_z}$, that is, the velocity in 3D space is given by the sum of the velocity of decomposed vectors.
    a. Pass ankles coordinates of the variable created in 1.c to a new variable "*ankle*".
    b. Apply *numpy.diff* to get velocity. Pay attention on the arguments of order and axis. Select first order (n = 1) and axis = 0.
    c. Do the same to acceleration, applying a second order differentiation on "*ankle*" or a first order differentiation on the velocity matrix.
    d. Get the final value summing coordinates with *sum*.
    e. Import *matplotlib.pyplot* and plot both, linear velocity and linear acceleration. Observe that to show the graphs it is necessary to use the command *matplotlib.pyplot.show()*.
    f. Save the results in a single txt.file. To do that you can use *numpy.savetxt*. Use *fmt="%f"* and certify that both vectors (velocity and acceleration have the same length).
3. Subtract the coordinates of one point by the other to have a vector "*leg*". Then, use trigonometric identities to get the angle.
    a. You can use *numpy.subtract* and then apply *numpy.arctan2*. Observe that the time series will be given in radians. Negative values are possible depending on how you setup previous vectors. To this exercise, it does not matter.
4. Angular velocity of a point is given by linear velocity divided by radius (difference between the point and its axis of rotation).
    a. You can use *numpy.subtract* again.
    b. Plot in the same graph the radius obtained using only x and z-axes and the radius obtained using x, y, and z-axes.
5. Differentiate angular velocity to obtain angular acceleration.

Exercise 2

In this exercise, you will use data referring to a countermovement jump.

1. Flight time, takeoff, instant of maximum height, landing, and the end of the jump.
2. Calculate the height of the jump.
3. Plot in the same graph the time series of the knee angle for each jump.

Hints (more general than in exercise 1)

1. Use the same methods applied in the previous exercise to load data file and create vectors. Observe that knee point is the connection between the two vectors that you need to creates, i.e. vector "*thigh*" and vector "*leg*".
2. Additionally, you will need to import *linealg* from *numpy* to calculate knee angle.
   a. Use *numpy.cross* to take the cross product of the two new vectors that you just created (i.e. *thigh* and *leg*).
   b. Use *linealg.norm* to take the norm of the same vectors.
   c. Take the dot product of these vectors as well. It is tricky. Given the type of data, you need use *numpy.multiply* (the second vector has to be transposed) and then apply *sum* in the result.
   d. Finally, calculate the angle using *numpy.arctan2(norm, dot product)*.
3. To create the time vector, you can apply *numpy.linspace* with the number of data points as the number of intervals between extremes (onset data set recording and finishing data set recording).
4. Event points of the countermovement jump.
   a. Jump onset is considered when the jumper started the preparatory phase, thus that is the time ZERO. In other words, it has to be subtracted from all other temporal measures to provide the right result. As an arbitrary criterion it is the first instant that the hip velocity becomes lower than -1.
   b. The highest velocity characterizes takeoff.
   c. Maximum z displacement is characterized by velocity zero during the aerial phase of the jump.
   d. Landing is characterized by the lower velocity (or larger negative value).
   e. The end of the jump is assumed as the instant that the jumper's hip crosses up the "-1 line" after landing.
5. To get time it is necessary to relate event points in hint 4 with time vector. I did that using the following piece of code:

```
cont = -1
ind_temp = []

for i in vel_vert_hip:
   cont = cont + 1
   while i < -1:
      ind_temp.append(cont)
      break
```

```
ind_onset = ind_temp[0]
ind_end = ind_temp[-1]
```

ind_temp is a temporary vector with index of all positions in the *vel_vert_hip* vector (i.e. vertical velocity of the hip) lower than -1. The *.append* stands for add those indices captured by *cont*. *ind_onset* receives the time referring to the beginning of the jump and *ind_end* referring to the end. This function works only if you consider one jump at once.

6. Flight height is given by $v_0^2/2g$. Where v_0^2stands for takeoff velocity and g stands for gravity acceleration (i.e. -9.81m/s^2). There are two ways to calculate takeoff velocity (given the limitations of our example they should provide different results).
    a. Method 1: use the value of velocity correspondent to the time index identified in 1.
    b. Method 2: use the formula $(g * t_{fligh})/2$. Where t$_{flight}$ is the time of the aerial phase of the jump.
7. Use the same plot functions used in the previous exercise.