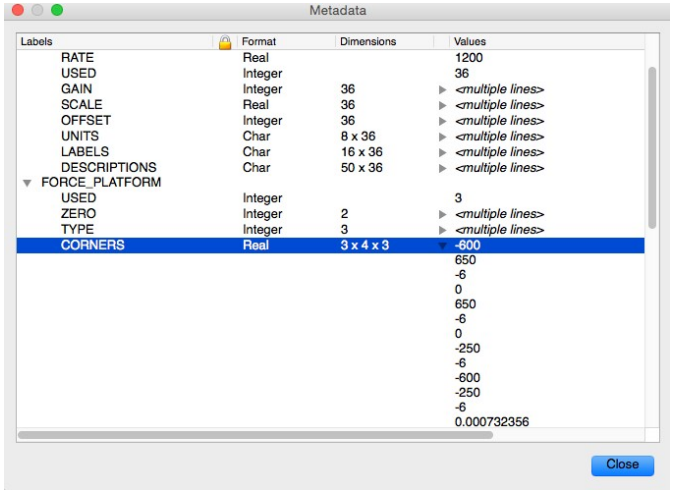
Lab Tutorial 4: Adjusting OpenSim models

Copy the Lab Material-Tutorial 4 folder to your workspace. Within this folder you will see some Jupyter notebooks and Python functions, an opensim file called *gait2354_3DOF.osim* and some data.

Task 1: Exporting force plate data from a c3d file using Mokka

[Use Mokka to visualise the *jw_ngait_og1.c3d* file]

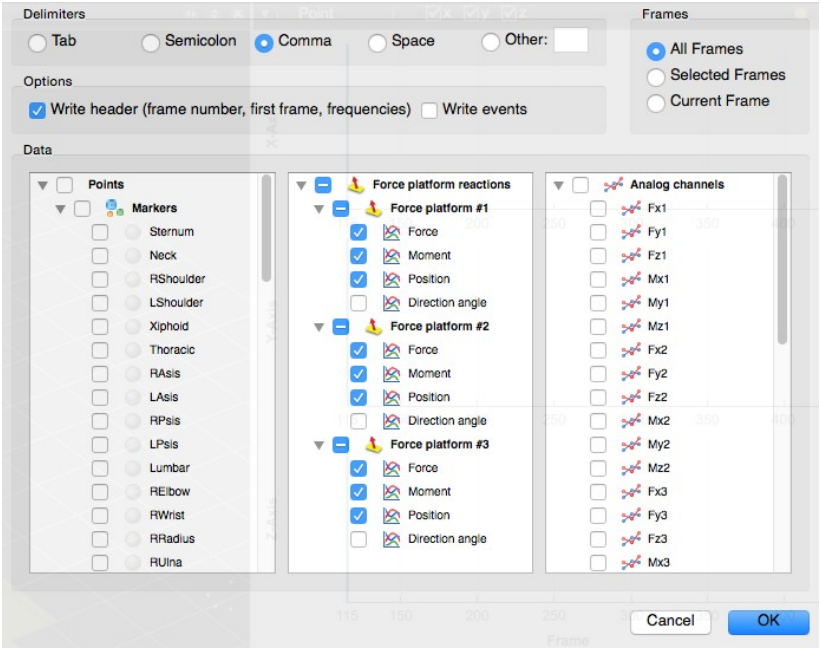
Notice the numbering and orientation of the three force plates. These force plates have their own coordinate system and the calculated centre of pressure (CoP) is calculated with respect to these. Note also that their origin is at the centre of each plate. In order to perform inverse dynamics we need to make sure that these forces and CoP are in the global CS, which is what our marker data are recorded in. To do this you need to know the orientation and position of each plate with respect to the global CS. This information is actually stored within the C3D file and you can see this in Mokka by clicking on *View > Metadata*, which brings up a window showing all of the additional information stored in the C3D file. If you expand the FORCE_PLATFORM section you can get access to the position of the force plate corners wrt the global CS.



Mokka will do this transformation for you when you export the force plate data as a text/ascii file.

[Select *File > Export > ASCII/Text file …*]

The following window appears. Deselect the Points (markers) and Analog channels.



If you save your file as comma delimited .csv file, we can use it with our Jupyter notebook and it will automatically open in Excel, so you can easily plot the forces and check that everything is in the correct coordinate frame.

Task 2: Filtering ground reaction forces using Python

We are going to filter the GRF data using a low-pass Butterworth filter; similar to the way we filtered the motion data in the previous tutorial.

[open up the *FilterForcePlateData.ipynb* Jupyter notebook]

The key steps to this code are:

1. read in ground reaction forces from csv file
2. define cut off frequency and the capture rate of the force plate data
3. tease apart the force array into separate force plates
4. filter just the first force plate data (by way of example)
5. set any forces below a threshold to zero (this can reduce artifact from filtering and noisy force plates)
6. rotate the force plate data to a coordinate system consistent with opensim (in this example a 90deg positive rotation about global-x
7. export data as a motion file (calling write_motion_file.py)

[run the *FilterForcePlateData.ipynb* Jupyter notebook and select the jw_ngait_og1_grf.csv trial]

Take a close inspection at the forces in the plot and you will see the influence of the filtering on these force data. At heel strike and toe-off there is some 'overshoot' from the filter. These result in small forces before and after the real heel strike and toe off events. For this reason, we reset zero values based on a threshold of raw vertical ground reaction force data (step 5 in the code above).

This code will output a .mot file giving it the same name as the input .csv file into the same folder. This file should now be good to import into OpenSim.

Task 3: Altering joint parameters in OpenSim – increasing DOFs at the knee

The standard *gait2354_simbody.osim* model has only one degree of freedom (DOF) at the knee. If you calculate IK and ID, you will only get joint coordinates and moments for this single DOF. We would like OpenSim to calculate knee moments about internal-external rotation and adduction-abduction (varus-valgus). To do this we need to hack the .osim file and change a few parameters.

[Open up gait2354_simbody.osim using Notepad++ and resave it as gait2354_3dofknee.osim]

Let's change the right knee joint as an example. The easiest way to find this joint is to search for "knee_r". You will see that the right knee joint is nested within the Body of the right tibia. In OpenSim the joint describes the relationship between the 'child' body with respect to the 'parent' body. In this case the tibia is the child body and the parent body is the femur. In the <Joint> section you will see a *CustomJoint* called "knee_r". You can define various joints within OpenSim. A custom joint is a 6dof joint requiring that you describe all 3 rotations and translations. These rotations and translations are described using <SpatialTransform>. The first rotation for the right knee is "rotation1" and the <coordinates> associated with this DOF is called "knee_angle_r". The <axis> of this rotation is given as 0 0 1, corresponding to the z-axis of the segment. This of course is the flexion-extension axis for the knee. Notice that the <coordinates> section for "rotation2" is empty. We will include our own definition here for this rotation. Since the <axis> associated with this rotation is 0 1 0, which corresponds to the y-axis of the segment, we will call this generalized coordinate internal-external rotation.

[Add the name "knee_ie_r" to the <coordinates> section of the osim file]

```
<Body name="tibia_r">
    <mass>3.7075</mass>
    <mass_center> 0 -0.1867 0</mass_center>
    <inertia_xx>0.0504</inertia_xx>
    <inertia_yy>0.0051</inertia_yy>
    <inertia_zz>0.0511</inertia_zz>
    <inertia_xy>0</inertia_xy>
    <inertia_xz>0</inertia_xz>
    <inertia_yz>0</inertia_yz>
    <!--Joint that connects this body with the parent body.-->
    <Joint>
        <CustomJoint name="knee_r">
            <!--Defines how the child body moves with respect to the parent as a functior
            <SpatialTransform>
                <!--3 Axes for rotations are listed first.-->
                <TransformAxis name="rotation1">
                    <!--Names of the coordinates that serve as the independent variables
                    <coordinates>knee_angle_r</coordinates>
                    <!--Rotation or translation axis for the transform.-->
                    <axis>0 0 1</axis>
                    <!--Transform function of the generalized coordinates used to
                    <function>
                        <LinearFunction>
                            <coefficients> 1 0</coefficients>
                        </LinearFunction>
                    </function>
                </TransformAxis>
                <TransformAxis name="rotation2">
                    <!--Names of the coordinates that serve as the independent variables
                    <coordinates></coordinates>
                    <!--Rotation or translation axis for the transform.-->
                    <axis>0 1 0</axis>
                    <!--Transform function of the generalized coordinates used to
                    <function>
                        <Constant>
                            <value>0</value>
                        </Constant>
                    </function>
                </TransformAxis>
                <TransformAxis name="rotation3">
                    <!--Names of the coordinates that serve as the independent variables
                    <coordinates></coordinates>
```

Copy the <LinearFunction> from here to here

Add a coordinate name here

Next we have to ensure that we have OpenSim will know how to move this generalized coordinate. For this we will have a one-to-one relationship or linear function between the angle and the generalized coordinate. To do this we need to change the <function> from being a <constant> to a <LinearFunction>.

[copy and paste the <function> section from the "rotation1" section to the "rotation2" section, replacing the existing <Constant> variable with a <LinearFunction>]

[Repeat this for rotation3, creating a generalized coordinate called "knee_vv_r"]

The final change we need to make is to include these new generalized coordinates (*knee_ie_r* and *knee_vv_r*) in the <CoordinateSet>. Search for <Coordinate name="knee_angle_r"> in the osim file and you will see the default settings for this coordinate. We need to add our two new coordinates underneath this section and provide some default values.

```
    <location> 0 0 0</location>
    <orientation> 0 0 0</orientation>
    <!--Generalized coordinates parameterizing this joint.-->
    <CoordinateSet>
        <objects>
            <Coordinate name="knee_angle_r">
                <!--Coordinate can describe rotational, translational, or coupled mot
                <motion_type>rotational</motion_type>
                <!--The value of this coordinate before any value has been set. Rotat
                <default_value>0</default_value>
                <!--The speed value of this coordinate before any value has been set
                <default_speed_value>0</default_speed_value>
                <!--The minimum and maximum values that the coordinate can range betw
                <range>-2.0943951 0.17453293</range>
                <!--Flag indicating whether or not the values of the coordinates shou
                <clamped>false</clamped>
                <!--Flag indicating whether or not the values of the coordinates shou
                <locked>false</locked>
                <!--If specified, the coordinate can be prescribed by a function of t
                <prescribed_function />
            </Coordinate>
        </objects>
        <groups />
    </CoordinateSet>
    <reverse>false</reverse>
</CustomJoint>
int>
```

Copy the entire <Coordinate> section and paste a copy underneath

[copy the entire <Coordinate> section of code and paste it underneath the </Coordinate> line]

The values associated with this coordinate include: *name*, *default value* (in radians), *range* (in radians).

[enter the new coordinate name “knee_ie_r”, a default value of 0 and a range of -1.0 to 1.0]

[repeat this process for “knee_vv_r”]

There are also some options here for locking or clamping the coordinate value. Another way to constrain the motion (such as for knee varus valgus) is to set the default range from 0 to 0. By doing so you will not permit any motion of this coordinate during IK, but the forces and moments will still be calculated by OpenSim during ID.

[save the adjusted .osim file and then reopen the model in OpenSim]

You will now see your new coordinates for the right knee joint in the main Coordinates tab of OpenSim and you will be able to adjust the values and see the corresponding rotations.