OpenSim 3.3 Vs. 4.1 Scaling Investigation

Fouad Matari

August 31, 2020

1 Background

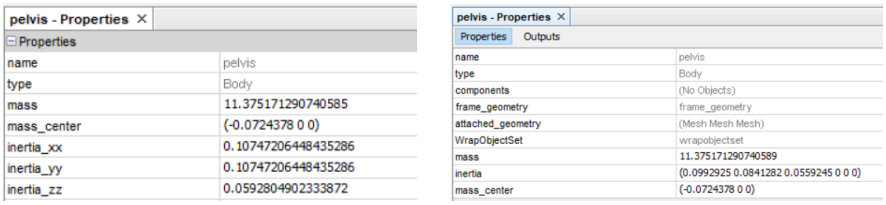
OpenSim has a tool for altering the anthropometry and adjusting the mass properties of a model through manual scaling factors or marker measurements. This is beneficial in order to match a particular subject as close as possible. Manual scaling provides two options, uniform and non-uniform scaling. Uniform scaling will take in one scale factor for all three axes while non uniform scaling will take an independent scaling factor for each axis. After the scale factors are set, mass and inertial properties are scaled followed by the muscles and other model components that depend on length. Finally the scale tool will generate new marker locations for the scaled model.

2 Problem

There is a difference in inertia results after performing non-uniform manual scaling in versions 3.3 and 4.1, an investigation sprouted from this difference. There is a need for identifying the version that provides the correct inertia results. Then determining the cause and solution for the version which provides the incorrect results.

3 Solution Approach

An examination of the source codes to determine the handling of the scaling routine was necessary in ensuring that the calculations performed are in fact the same. Visually, routines looked practically identical with minimal differences. Scaling routine is found in the body.cpp file within the source code. A spreadsheet was then created in order to determine the expected inertia results through manual calculations and then compare with the outputs of two versions.



pelvis - Properties

Properties	
name	pelvis
type	Body
mass	11.375171290740585
mass_center	(-0.0724378 0 0)
inertia_xx	0.10747206448435286
inertia_yy	0.10747206448435286
inertia_zz	0.0592804902333872

(a) OpenSim 3.3

pelvis - Properties

Properties | Outputs

name	pelvis
type	Body
components	(No Objects)
frame_geometry	frame_geometry
attached_geometry	(Mesh Mesh Mesh)
WrapObjectSet	wrapobjectset
mass	11.375171290740589
inertia	(0.0992925 0.0841282 0.0559245 0 0 0)
mass_center	(-0.0724378 0 0)

(b) OpenSim 4.1

Figure 1: Non-Uniform Scaling

The gait2354 model is used in both versions. Comparison was performed for all segments and the pelvis will be used as the example. Scaled results were retrieved from the OpenSim GUI. Mass and center of mass for each segment were a match between the two versions. Only difference in scaling was seen in inertia during non-uniform scaling.

3.1 Spreadsheet

Performing the manual calculations on the spreadsheet required retrieving the original inertia and mass of the model. Following procedure was utilized for calculating the scaled inertia,

1. Determine the smallest diagonal inertia component
2. Calculate radius of cylinder
3. Calculate length of cylinder
4. Scale the radius and length
5. Recalculate inertia

Gathering the OpenSim outputs into the same spreadsheet as the calculated inertia, the comparisons were ready to be made.

			Scaled (U)	Calculated(U)	Scaled (NU)	Calculated(NU)
			4.1			
			0.10423	0.10423	0.09929	0.10747
Non-Uniform Scaling factors		Original Inertia	0.08831	0.08831	0.08413	0.10747
Pelvis_X (U)	1.02458	0.10280	0.05871	0.05871	0.05592	0.05928
Pelvis_Y	1.03458	0.08710	3.3			
Pelvis_Z	1.04458	0.05790	0.10423	0.10423	0.10747	0.10747
			0.08831	0.08831	0.10747	0.10747
			0.05871	0.05871	0.05928	0.05928

Figure 2: Spreadsheet Results

Calculated results are from the manual calculations performed on the spreadsheet and scaled are from the outputs of OpenSim. Original inertia of the model in both versions is the same and scaling factors were set to the same values. Uniform is denoted by an upper case U and non-uniform by NU. OpenSim 3.3 calculates the inertia properly while performing non-uniform scaling and OpenSim 4.1 provides an incorrect result.

3.2 Source Code

Looking further into OpenSim 4.1 source code, mainly body.cpp, the cause of this issue was hunted. Utilizing the command line for executing the scaling tool allowed for print statements to be used as a debugging measure. The function scaleInertialProperties() contains the routine for scaling inertia.

There are a couple if statements that must be satisfied in order to execute non-uniform scaling. These were examined first. The if statements successfully determined whether the user is performing uniform or non-uniform scaling. Now it is known that the routine which calculates non-uniform inertia is called upon.

```
if (axis == 0) {
    rad_sqr = radius * (scaleFactors[1]) * radius * (scaleFactors[2]);
    inertia[0][0] = 0.5 * get_mass() * rad_sqr;
    inertia[1][1] = get_mass() * ((length * length / 12.0) + 0.25 * rad_sqr);
    inertia[2][2] = get_mass() * ((length * length / 12.0) + 0.25 * rad_sqr);

} else if (axis == 1) {
    rad_sqr = radius * (scaleFactors[0]) * radius * (scaleFactors[2]);
    inertia[0][0] = get_mass() * ((length * length / 12.0) + 0.25 * rad_sqr);
    inertia[1][1] = 0.5 * get_mass() * rad_sqr;
    inertia[2][2] = get_mass() * ((length * length / 12.0) + 0.25 * rad_sqr);

} else {
    rad_sqr = radius * (scaleFactors[0]) * radius * (scaleFactors[1]);
    inertia[0][0] = get_mass() * ((length * length / 12.0) + 0.25 * rad_sqr);
    inertia[1][1] = get_mass() * ((length * length / 12.0) + 0.25 * rad_sqr);
    inertia[2][2] = 0.5 * get_mass() * rad_sqr;

    cout << "Radius:    " << radius << endl;
    cout << "Length:    " << length << endl;
    cout << "Mass:      " << get_mass() << endl;
    cout << "Radius^2:  " << rad_sqr << endl;
    cout << "Scale factors:  " << scaleFactors << endl;
    cout << "Inertia:      " << inertia << endl;

}
```

Figure 3: Non-Uniform Inertia Scaling Equations

All variables within the scaled inertia equations were verified with the values calculated in the spreadsheet. Radius, length, scaling factors, mass, and radius squared were all being gathered or computed properly. The if statements are determining which axis of the segment is the cylinder axis. Print statements are

inserted within the condition for cylinder axis being the Z axis. Pelvis is the only segment which has it's smallest inertia (cylinder axis) only on the Z axis. We expect the properties of pelvis to be printed.

```
[info] Step 2: Scaling generic model
[info] Loaded marker file subject01_static.trc
Radius:    0.0991601
Length:    0.286543
Mass:      11.777
Radius^2:  0.0104228
Scale factors:  ~[1.02458,1.03458,1.04458]
Inertia:
[0.1028 * *]
[0 0.0871 *]
[0 0 0.0579]
```

(a) OpenSim Command Line Output

PELVIS	
Radius	0.09916009683
Length	0.2865424662
Mass	11.777
Radius^2	0.01042272564
Scale Factors	[1.024577, 1.034577, 1.044577]
Inertia - xx	0.11127
Inertia - yy	0.11127
Inertia - zz	0.06137

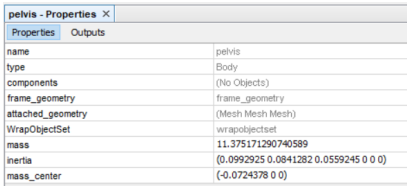
(b) Calculated

Figure 4: Pelvis Properties

The pelvis properties besides inertia matched. Figure 4(a) shows the command line output when utilizing the scaling tool with the print statements shown in figure 3. Calculated properties in 4(b) are from the spreadsheet calculations. Interestingly the inertia does not update from the OpenSim routine. It remains to be the original inertia which is found in figure 2. This tells us that the inertia calculations are not being processed. This was also confirmed by inertia[0][0] being set to a specific number such as 1, yet it prints out as 0.1028 which is the original inertia. At this point, it is known that inertia calculations for non-uniform scaling are not being computed.

```
Mass:  11.3752
Center of Mass:  ~[-0.0724378,0,0]
Inertia:
[0.0992925,0,0]
[0,0.0841282,0]
[0,0,0.0559245]
```

(a) Command Line



(b) OpenSim 4.1 GUI

Figure 5: Pelvis Mass Properties

Function getMassProperties() in file body.cpp provides the final scaled mass, center of mass, and inertia. Printing within this function provides us with the final mass properties that we see in the OpenSim GUI. Notice that the mass and inertia are now different than the previous pelvis properties in figure 4(a), this is because the mass scaling factor was applied after the first set of print statements. We now know where to print the final scaled mass properties that are shown in the OpenSim GUI.

4 Solution

After a few print statements and digging around, the issue narrowed down to the inertia variable definition. The inertia is defined as a 3 by 3 symmetric matrix which for an unknown reason is not allowing the redefinition of its elements in the manner shown in figure 3. Changing the variable type to a standard 3 by 3 matrix solves our problem.

```
218    //   SimTK::SymMat33 inertia = _inertia.asSymMat33();
219
220       SimTK::Mat33 inertia = _inertia.toMat33();
```

Figure 6: Inertia variable Type

Line 218 in body.cpp is the original code setting the symmetric matrix. It is replaced by line 220 for correction in the calculations.

```
[info] Step 2: Scaling generic model
[info] Loaded marker file subject01_static.trc
Radius:     0.0991601
Length:     0.286543
Mass:       11.777
Radius^2:   0.0104228
Scale factors:  ~[1.02458,1.03458,1.04458]
Inertia:
[0.111269,0,0]
[0,0.111269,0]
[0,0,0.0613746]
```

(a) OpenSim Command Line Output

PELVIS	
Radius	0.09916009683
Length	0.2865424662
Mass	11.777
Radius^2	0.01042272564
Scale Factors	[1.024577, 1.034577, 1.044577]
Inertia - xx	0.11127
Inertia - yy	0.11127
Inertia - zz	0.06137

(b) Calculated

Figure 7: Pelvis Properties

Printing from the scaleInertialProperties() function, the inertia is now outputting the proper and expected result before the application of mass scale factor.

```
Mass:  11.3752
Center of Mass:  ~[-0.0724378,0,0]
Inertia:
[0.107472,0,0]
[0,0.107472,0]
[0,0,0.0592805]
```

(a) Command Line Mass Prop

pelvis - Properties

Properties	
name	pelvis
type	Body
mass	11.375171290740585
mass_center	(-0.0724378 0 0)
inertia_xx	0.10747206448435286
inertia_yy	0.10747206448435286
inertia_zz	0.0592804902333872

(b) OpenSim 3.3 GUI

Figure 8: Pelvis Mass Properties

Printing from the getMassProperties() function, the inertia is now outputting the proper and expected final result. The printed mass properties now match OpenSim 3.3 which was previously verified to be true.