



Documents



OpenSim Developer's Jamboree

January 14-16, 2009,
Stanford University

Website: SimTK.org/home/opensim

OpenSim Jamboree Agenda (Day 1)
Clark Center Room S282 (below Peet's)

DAY 1 – Wednesday, January 14, 2009

8:30am – 9:00am	OpenSim 1.8 installation and setup – <i>Arrive early if you need to install software</i>
9:00am – 9:15am	Welcome and goals of workshop – <i>Scott</i>
9:15am – 10:15am	Participant information and goals – <i>You</i>
10:15am – 10:25am	BREAK & Introduction to SimTK Team
10:25am – 10:40am	OpenSim as a research tool (<i>e.g.</i> , strengths, limitations, best practice) – <i>Scott</i>
10:40am – 11:10am	Elements of an OpenSim model (<i>e.g.</i> , organization, joints, constraints) – <i>Ajay</i>
11:10am – 11:50pm	Model editing and adding components (<i>e.g.</i> , body, actuator) – <i>Jeff & You</i>
11:50am – 1:00pm	LUNCH
1:00pm – 1:20pm	Behind computed muscle control (<i>e.g.</i> , theory, implementation) – <i>Ajay</i>
1:20pm – 1:50pm	Using computed muscle control (<i>e.g.</i> , choosing settings) – <i>Jeff & You</i>
1:50pm – 2:15pm	Available analyses and what they do (<i>e.g.</i> , perturbation) – <i>Sam & Chand</i>
2:15pm – 2:45pm	Questions and Discussion – <i>Everyone</i>
2:45pm – 3:30pm	Form groups and create project plans – <i>Scott</i>
3:30pm – 4:20pm	BREAK, work on project plan, or Simbios Talk (S360)
4:20pm – 5:00pm	Presentations of group project plans for Jamboree – <i>You</i>
6:00pm	Social Dinner (TBD)

OpenSim Jamboree Agenda (Days 2 & 3)

DAY 2 – Thursday, January 15, 2009

- | | |
|-------------------|--|
| 8:30am – 9:00am | Individual group feedback
– <i>OpenSim Team</i> |
| 9:00am – 11:45am | Work on projects
– <i>You & OpenSim Team</i> |
| 11:45am – 12:45pm | LUNCH |
| 12:45pm – 1:00pm | Open discussion of common issues
– <i>Scott & You</i> |
| 1:00pm – 4:45pm | Work on projects
– <i>You & OpenSim Team</i> |
| 4:45pm – 5:00pm | Open discussion of common issues
– <i>Scott & You</i> |

DAY 3 – Friday, January 16, 2009

- | | |
|-------------------|--|
| 8:30am – 8:45am | Open discussion of tips and comments
– <i>Scott & You</i> |
| 8:45am – 11:45am | Work on projects
– <i>You & OpenSim Team</i> |
| 11:45am – 12:45pm | LUNCH |
| 12:45pm – 2:45pm | Presentations of progress, hurdles, feedback, and future plans
– <i>You</i> |
| 2:45pm – 3:00pm | Closing remarks
– <i>Scott</i> |
| 3:00pm – 4:30pm | RECEPTION |

Trademarks and Copyright and Permission Notice

SimTK and Simbios are trademarks of Stanford University. The documentation for OpenSim is freely available and distributable under the MIT License.

Copyright (c) 2008 Stanford University

Permission is hereby granted, free of charge, to any person obtaining a copy of this document (the "Document"), to deal in the Document without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Document, and to permit persons to whom the Document is furnished to do so, subject to the following conditions:

This copyright and permission notice shall be included in all copies or substantial portions of the Document.

THE DOCUMENT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS, CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE DOCUMENT OR THE USE OR OTHER DEALINGS IN THE DOCUMENT.

Acknowledgments

[OpenSim](#) was developed as a part of [SimTK](#) and funded by the [Symbios](#) National Center for Biomedical Computing through the National Institutes of Health and the NIH Roadmap for Medical Research, Grant U54 GM072970. Information on the National Centers can be found at <http://nihroadmap.nih.gov/bioinformatics>.

Table of Contents

1	ELEMENTS OF A MODEL	9
1.1	What is a musculoskeletal model in OpenSim?	10
1.2	Organization of the OpenSim model file	10
1.3	Specifying a Body and its Joint	10
1.3.1	The CustomJoint Transform	12
1.3.2	Kinematic Constraints in OpenSim	13
1.4	Actuator (Forces) in OpenSim	14
1.4.1	The Muscle Actuator	14
1.4.2	Other Common Forces.....	15
2	EXAMPLE: MODEL EDITING	17
2.1	Connecting an Additional Segment to the Model	17
2.2	Adding an Additional Actuator.....	21
3	COMPUTED MUSCLE CONTROL.....	23
3.1	Overview: Computed Muscle Control (Residual Reduction):.....	23
3.2	Theoretical background:.....	23
3.3	Performing RRA in OpenSim: Key Elements	25
4	EXAMPLE: COMPUTED MUSCLE CONTROL	29
4.1	Using Computed Muscle Control.....	29
4.2	Changing the Desired Kinematics.....	30
4.3	Changing the Tracking Tasks	30
4.4	Changing the Actuator Constraints.....	31
4.5	Changing the CMC Look-Ahead Window	31

1 Elements of a Model

1.1 What is a musculoskeletal model in OpenSim?

In OpenSim a skeleton is comprised of rigid bodies, which are interconnected by joints. Joints define how a body can move with respect to its parent body. All bodies have a parent and are connected to a parent via a joint, except for ground. Constraints limit the motion of bodies.

Muscles are specialized actuators that act at muscle points connected to rigid bodies. The force of a muscle is typically dependent on the path (length) through muscle points and the rate of change of that length (velocity). OpenSim also has a variety of other actuators, which represent externally controlled forces, torques and generalized forces.

1.2 Organization of the OpenSim model file

In formulating the equations-of-motion (the system dynamics), OpenSim employs Simbody via the SimbodyEngine, where the body is the primary element, and all bodies in a model live in a BodySet. Each body in turn owns a joint and that joint defines the coordinates and kinematic transforms that govern the motion of that body. The ConstraintsSet contains all the kinematic constraints that act on bodies (and/or their coordinates) in the model. User forces acting on the model are all included in an ActuatorSet

```

<Model name="subject01">
  <defaults>...
  <credits> Delp S.L., Loan J.P., Hoy M.G., Zajac F.E., Topp E.L., Rosen J.M., Thelen D.G., Anderson F.C., Seth A.
  <publications>...
  <ContactForceSet name="">...
  <DynamicsEngine>
    <SimbodyEngine name="default">
      <!--Acceleration due to gravity.-->
      <gravity> 0.00000000 -9.80665000 0.00000000 </gravity>
      <!--Bodies in the model.-->
      <BodySet name="">...
      <!--Constraints in the model.-->
      <ConstraintSet name="">...
      <!--Markers in the model.-->
      <MarkerSet name="">...
    </SimbodyEngine>
  </DynamicsEngine>
  <ActuatorSet name="">...
  <length_units> meters </length_units>
  <force_units> N </force_units>
</Model>

```

Figure 1: High level OpenSim Model organization.

1.3 Specifying a Body and its Joint

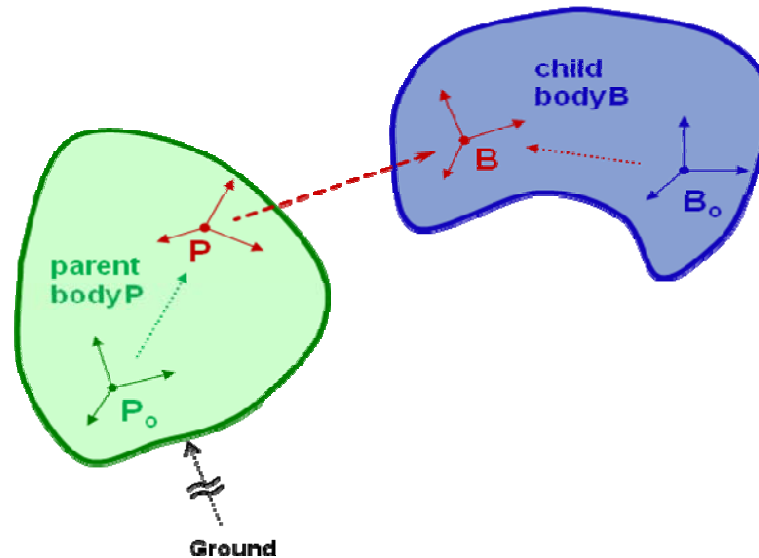


Figure 2: A joint (in red) defines the kinematic relationship between frames on two rigid-bodies (a parent (P) and child body (B)) parameterized by joint coordinates.

A body is a moving reference frame (B_o) in which its center-of-mass and inertia are defined, and the location of a joint frame (B) fixed to the body can be specified.

```

<Body name="tibia_r">
  <mass> 3.58100090 </mass>
  <mass_center> 0.00000000 -0.18455724 0.00000000 </mass_center>
  <inertia_xx> 0.04756937 </inertia_xx>
  ...
  <inertia_yz> 0.00000000 </inertia_yz>
  <!--Joint that connects this body with the parent body.-->
  <Joint>
    <CustomJoint name="knee_r">
      <parent_body> femur_r </parent_body>
      <location_in_parent> 0.00000000 0.00000000 0.00000000 </location_in_parent>
      <orientation_in_parent> 0.00000000 0.00000000 0.00000000 </orientation_in_parent>
      <location> 0.00000000 0.00000000 0.00000000 </location>
      <orientation> 0.00000000 0.00000000 0.00000000 </orientation>
      <!--Generalized coordinates parameterizing this joint.-->
      <CoordinateSet name="">
        <objects>
          <Coordinate name="knee_angle_r">
            <range> -2.09439510 0.17453293 </range>
            <clamped> true </clamped>
            <locked> false </locked>
          </Coordinate>
        </objects>
      </CoordinateSet>
    </CustomJoint>
  </Joint>
</Body>

```

Figure 3: Sample Body and Joint Definitions in OpenSim. The right knee joint is governed by one coordinate, the knee_angle_r.

1.3.1 The CustomJoint Transform

Most joints in an OpenSim model are custom joints since they can be used to model conventional (pins, slider, universal, etc...) as well as biomechanical joints. The user must define the transform (rotation and translation) of the child in the parent as a function of the generalized coordinates listed in the Joint's CoordinateSet (Fig. 3). Consider the transform:

$${}^P \mathbf{X}(q)^B = \begin{bmatrix} {}^P \mathbf{R}^B(x_1, x_2, x_3) & x_4 \\ & x_5 \\ & x_6 \end{bmatrix}, \text{ where } x(q) = \begin{Bmatrix} f_1(q_1, q_2, \dots, q_m) \\ f_2(q_1, q_2, \dots, q_m) \\ \vdots \\ f_6(q_1, q_2, \dots, q_m) \end{Bmatrix}$$

and q are the coordinates of the joint and x determine the individual rotations and translation along user-defined axes according to functions f . A CustomJoint can have up to 6 user-defined TransformAxes (in a single TransformAxisSet) to enable a maximum of 6 degrees-of-freedom. Each transform axis specifies a joint coordinate to operate along or about an axis. If a function is provided then $f(q)$ is used as the value for that axis, and the

user must specify if it is a rotation or not. Therefore, coupled motion, such as tibial translation with knee flexion is easily handled, as in the gait2354.osim model.

```

<TransformAxisSet name="">
  <objects>
    <TransformAxis name="r1">
      <function/>
      <coordinate> knee_angle_r </coordinate>
      <is_rotation> true </is_rotation>
      <axis> 0.00000000 0.00000000 1.00000000 </axis>
    </TransformAxis>
    <TransformAxis name="tx">
      <function>
        <natCubicSpline name="">...</natCubicSpline>
      </function>
      <coordinate> knee_angle_r </coordinate>
      <is_rotation> false </is_rotation>
      <axis> 1.00000000 0.00000000 0.00000000 </axis>
    </TransformAxis>
    <TransformAxis name="ty">
      ...
    </TransformAxis>
  </objects>
</TransformAxisSet>
</CustomJoint>
</Joint>
<VisibleObject name="">...</VisibleObject>
</Body>

```

Figure 4: Transform definition for the implementation of a knee joint.

1.3.2 Kinematic Constraints in OpenSim

OpenSim currently supports two types of constraints: the CoordinateCouplerConstraint and the WeldConstraint. A coordinate coupler relates the generalized coordinate of a given joint (the dependent coordinate) to any other coordinates in the model (independent coordinates). The user must supply a function that returns a dependent value based on independent values. A weld constraint fixes the relative location and orientation of two bodies throughout a simulation. The following example implements the motion of the patella as a function of the knee ankle and welds the foot to ground.

```

<!--Constraints in the model.-->
<ConstraintSet name="">
  <objects>
    <CoordinateCouplerConstraint name="pat_tx_r">
      <isDisabled> false </isDisabled>
      <coupled_coordinates_function>
        <natCubicSpline name="">...
      </coupled_coordinates_function>
      <independent_coordinate_names> knee_angle_r </independent_coordinate_names>
      <dependent_coordinate_name> pat_tx_r </dependent_coordinate_name>
    </CoordinateCouplerConstraint>
    <CoordinateCouplerConstraint name="pat_ty_r">...
    <CoordinateCouplerConstraint name="pat_angle_r">...
    <WeldConstraint name="">
      <isDisabled> false </isDisabled>
      <body_1> ground </body_1>
      <body_2> calcn_r </body_2>
      <location_body_1>          0.0000000000      0.0000000000      0.0840000000
      <orientation_body_1>      0.0000000000      0.0000000000      0.0000000000
      <location_body_2>          0.0000000000      0.0000000000      0.0000000000
      <orientation_body_2>      0.0000000000      0.0000000000      0.0000000000
    </WeldConstraint>
  </objects>
</groups/>
</ConstraintSet>

```

Figure 5: Example of constraints in OpenSim

1.4 Actuator (Forces) in OpenSim

Sources of force in OpenSim are referred to as actuators because they are (in general) under active control and their force output must be determined.

1.4.1 The Muscle Actuator

There are several muscle models in OpenSim. All muscles include a set of muscle points where the muscle is connected to bones (bodies) and provide utilities for calculating muscle-actuator lengths and velocities. Internally muscle models may differ in the number and type of parameters. Muscles typically include muscle activation and contraction dynamics and their own states. The control values are typically bounded (0,1) excitations which lead to a change in activation and then force.

```

<Thelen2003Muscle name="BIClong">
  <MusclePointSet>
    <objects>
      <MusclePoint>
        <location>-0.039235000000 0.003470000000 0.147950000000</location>
        <body>ground</body>
      </MusclePoint>
      ...
      <MusclePoint>
        <location>0.022800000000 -0.175400000000 -0.006300000000</location>
        <body>r_humerus</body>
      </MusclePoint>
      <MusclePoint>
        <location>0.007510000000 -0.048390000000 0.021790000000</location>
        <body>r_ulna_radius_hand</body>
      </MusclePoint>
    </objects>
  </MusclePointSet>
  <max_isometric_force>624.300000000000</max_isometric_force>
  <optimal_fiber_length>0.115700000000</optimal_fiber_length>
  <tendon_slack_length>0.272300000000</tendon_slack_length>
  <pennation_angle>0.000000000000</pennation_angle>
  <muscle_model>9</muscle_model>
  <MuscleWrapSet>
    <objects>
      <MuscleWrap>
        <wrap_object> BIClonghh </wrap_object>
        <method> hybrid </method>
        <range> 2 3 </range>
      </MuscleWrap>
    </objects>
  </MuscleWrapSet>
</Thelen2003Muscle>

```

Figure 6: Sample muscle actuator from an ActuatorSet

1.4.2 Other Common Forces

OpenSim also includes “ideal” actuators which apply pure forces or torques that are directly proportional to the input control (excitation) via its optimal force. Forces and torques are applied between bodies and generalized forces are applied along the axis of a generalized coordinate.

```

<Force name="FY">
  <optimal_force> 8.0 </optimal_force>
  <body_A> ground </body_A>
  <point_A> 0.000 0.000 0.000 </point_A>
  <direction_A> 0.000 -1.000 0.000 </direction_A>
  <body_B> pelvis </body_B>
  <point_B> -0.0724376 0.00000000 0.00000000 </point_B>
</Force>

<Torque name="M2">
  <optimal_force> 2.0 </optimal_force>
  <body_A> ground </body_A>
  <direction_A> 0.000 0.000 -1.000 </direction_A>
  <body_B> pelvis </body_B>
</Torque>

<GeneralizedForce name="knee_angle_r">
  <coordinate> knee_angle_r </coordinate>
  <optimal_force> 300.0 </optimal_force>
</GeneralizedForce>

```

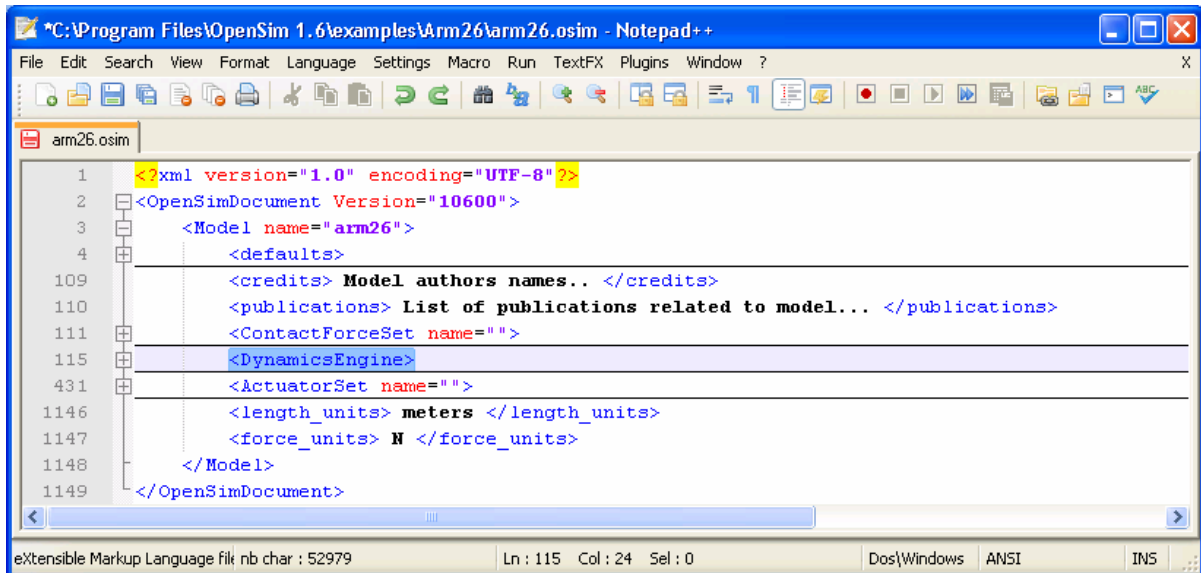
Figure 7: Sample of ideal forces and torques as they appear in the ActuatorSet of an OpenSim model

.

2 Example: Model Editing

2.1 Connecting an Additional Segment to the Model

1. **Save Model File for Editing.** To save the model, select **Save Model As...** from the **File** menu. Specify a **File name** (e.g., *arm26_editable.osim*) and click the **Save** button.
2. **Open Model File for Editing.** Use an XML editor (e.g., Notepad++) to open the OpenSim model file (e.g., *arm26_editable.osim*). When collapsed to the 3rd level (e.g., **Alt+3** in Notepad++), you should see the following (*Note: the **DynamicsEngine** tag has been highlighted*):



```
*C:\Program Files\OpenSim 1.6\examples\Arm26\arm26.osim - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
arm26.osim
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <OpenSimDocument Version="10600">
3   <Model name="arm26">
4     <defaults>
109   <credits> Model authors names.. </credits>
110   <publications> List of publications related to model... </publications>
111   <ContactForceSet name="">
115   <DynamicsEngine>
431   <ActuatorSet name="">
1146   <length_units> meters </length_units>
1147   <force_units> N </force_units>
1148   </Model>
1149 </OpenSimDocument>
```

eXtensible Markup Language file nb char : 52979 Ln : 115 Col : 24 Sel : 0 Dos\Windows ANSI INS

3. **Explore DynamicsEngine Children.** The **DynamicsEngine** tag has four children named **gravity**, **BodySet**, **ConstraintSet**, and **MarkerSet**. *Note: the **BodySet** tag has been highlighted.*

```

115 | <DynamicsEngine>
116 |   <SimbodyEngine name="default">
117 |     <!--Acceleration due to gravity-->
118 |     <gravity>      0.00000000    -9.80665000    0.00000000 </gravity>
119 |     <!--Bodies in the model-->
120 |     <BodySet name="" >
355 |   <!--Constraints in the model-->
356 |   <ConstraintSet name="" >
360 |   <!--Markers in the model-->
361 |   <MarkerSet name="" >
429 | </SimbodyEngine>
430 | </DynamicsEngine>
431 | <ActuatorSet name="" >

```

4. **Explore BodySet Children.** The **BodySet** tag has three grandchildren **Body** objects named **ground**, **r_humerus**, and **r_ulna_radius_hand**. *Note: the **Body** tag named **r_ulna_radius_hand** has been highlighted.*

```

115 | <DynamicsEngine>
116 |   <SimbodyEngine name="default">
117 |     <!--Acceleration due to gravity-->
118 |     <gravity>      0.00000000    -9.80665000    0.00000000 </gravity>
119 |     <!--Bodies in the model-->
120 |     <BodySet name="" >
121 |       <objects>
122 |         <Body name="ground">
167 |         <Body name="r humerus">
286 |         <Body name="r ulna radius hand">
352 |       </objects>
353 |     <groups/>
354 |   </BodySet>

```

5. **Add New Body.** Highlight the **Body** named **r_ulna_radius_hand** along with all of its children and copy (**Ctrl+C**) to the Clipboard. Paste the Clipboard contents immediately below the **Body** named **r_ulna_radius_hand**. Rename the new **Body** to **bucket**. *Note: the new **Body** tag named **bucket** has been highlighted.*

```

115 | <DynamicsEngine>
116 |   <SimbodyEngine name="default">
117 |     <!--Acceleration due to gravity-->
118 |     <gravity>      0.00000000    -9.80665000    0.00000000 </gravity>
119 |     <!--Bodies in the model-->
120 |     <BodySet name="" >
121 |       <objects>
122 |         <Body name="ground">
167 |         <Body name="r humerus">
286 |         <Body name="r ulna radius hand">
352 |         <Body name="bucket">
418 |       </objects>
419 |     <groups/>

```

6. **Specify Mass Properties.** Enter values for the **mass**, **mass_center**, **inertia_xx**, **inertia_yy**, and **inertia_zz** of the bucket as seen below:

```

352 | <Body name="bucket">
353 |   <WrapObjectSet name="">
357 |     <mass> 1.00000000 </mass> <!-- Specify mass and inertia as desired -->
358 |     <mass_center> 0.00000000 -0.10000000 0.00000000 </mass_center>
359 |     <inertia_xx> 0.00223958 </inertia_xx> <!-- Ixx = m/12(3r^2+h^2) = m*0.00223958333 -->
360 |     <inertia_yy> 0.00281250 </inertia_yy> <!-- Iyy = m/2(r^2) = m*0.0028125 -->
361 |     <inertia_zz> 0.00223958 </inertia_zz> <!-- Izz = m/12(3r^2+h^2) = m*0.004375 -->
362 |     <inertia_xy> 0.00000000 </inertia_xy>
363 |     <inertia_xz> 0.00000000 </inertia_xz>
364 |     <inertia_yz> 0.00000000 </inertia_yz>
365 |     <!-- Joint that connects this body with the parent body.-->
366 |     <Joint>
407 |   </VisibleObject name="">

```

7. **Specify Joint.** Enter names for the **CustomJoint** and **parent_body**, and enter values for **location_in_parent** as seen below:

```

365 | <!-- Joint that connects this body with the parent body.-->
366 | <Joint>
367 |   <CustomJoint name="r_handle"> <!-- Specify joint name (e.g., r_handle)-->
368 |     <parent_body> r_ulna_radius_hand </parent_body> <!-- Specify parent body name (e.g., r
369 |     <location_in_parent> 0.03100000 -0.31000000 0.07000000
370 |     <orientation_in_parent> 0.00000000 0.00000000 0.000000
371 |     <location> 0.00000000 0.00000000 0.00000000 </location
372 |     <orientation> 0.00000000 0.00000000 0.00000000 </orie
373 |     <!-- Generalized coordinates parameterizing this joint.-->
374 |     <CoordinateSet name="">
394 |     <TransformAxisSet name="">
405 |   </CustomJoint>
406 | </Joint>

```

8. **Specify Generalized Coordinate.** Enter name for the **Coordinate**, and enter values for **range** as seen below:

```

373 | <!-- Generalized coordinates parameterizing this joint.-->
374 | <CoordinateSet name="">
375 |   <objects>
376 |     <Coordinate name="r_handle_rot"> <!-- Specify coordinate name (e.g., r_handle_rot)
377 |       <default_value> 0.00000000 </default_value>
378 |       <initial_value> 0.00000000 </initial_value>
379 |       <tolerance> 0.00000000 </tolerance>
380 |       <stiffness> 0.00000000 </stiffness>
381 |       <range> -3.14159265 3.14159265 </range> <!-- Specify
382 |       <Keys> e_key leftmouse_button </keys>
383 |       <clamped> true </clamped>
384 |       <locked> false </locked>
385 |       <restraint function/>

```

9. **Specify Joint Axis.** Enter name for **coordinate**, and enter values for **axis** as seen below:

```

394 | <TransformAxisSet name="">
395 |   <objects>
396 |     <TransformAxis name="r3">
397 |       <function/>
398 |       <coordinate> r_handle_rot </coordinate> <!-- Specify coordinate name (
399 |       <is_rotation> true </is_rotation>
400 |       <axis> 0.04940001 0.03660001 0.99810825 </
401 |     </TransformAxis>
402 |   </objects>
403 | </groups/>
404 | </TransformAxisSet>
405 | </CustomJoint>
406 | </Joint>

```

10. **Specify Geometry File.** Enter name for **geometry_files** as seen below:

```

407 | <VisibleObject name="">
408 |   <geometry_files> bucket.vtp </geometry_files> <!-- Specify geometry file name (e.g., bucket.v
409 |   <VisibleProperties name="">
410 |     <display_preference> 4 </display_preference>
411 |     <show_normals> false </show_normals>
412 |     <show_axes> false </show_axes>
413 |     <material_name> DEFAULT </material_name>
414 |   </VisibleProperties>
415 |   <scale_factors> 1.00000000 1.00000000 1.00000000 </scale_
416 | </VisibleObject>
417 | </Body>
418 | </objects>
419 | </groups/>

```

11. **Save Model File.** From the XML editor, save the OpenSim model file (e.g., *arm26_with_bucket.osim*).

2.2 Adding an Additional Actuator

1. **Explore ActuatorSet Children.** The **ActuatorSet** tag has six grandchildren **Thelen2003Muscle** objects named **TRIlong**, **TRIlat**, **TRImed**, **BIClong**, **BICshort**, and **BRA**.

```

497 | <ActuatorSet name="" >
498 |   <objects>
499 |     <Thelen2003Muscle name="TRIlong">
622 |     <Thelen2003Muscle name="TRIlat">
735 |     <Thelen2003Muscle name="TRImed">
848 |     <Thelen2003Muscle name="BIClong">
017 |     <Thelen2003Muscle name="BICshort">
138 |     <Thelen2003Muscle name="BRA">
209 |   </objects>
210 |   <groups/>
211 | </ActuatorSet>

```

2. **Add New Actuator.** Add a **GeneralizedForce** object named **r_handle_rot_force** immediately below the **Thelen2003Muscle** named **BRA**. Associate this **GeneralizedForce** with the **r_handle_rot** coordinate and specify an **optimal_force** of 1000. *Note: the new **Actuator** tag named **r_handle_rot_force** has been highlighted.*

```

497 | <ActuatorSet name="" >
498 |   <objects>
499 |     <Thelen2003Muscle name="TRIlong">
622 |     <Thelen2003Muscle name="TRIlat">
735 |     <Thelen2003Muscle name="TRImed">
848 |     <Thelen2003Muscle name="BIClong">
017 |     <Thelen2003Muscle name="BICshort">
138 |     <Thelen2003Muscle name="BRA">
209 |     <GeneralizedForce name="r_handle_rot_force">
210 |       <coordinate> r_handle_rot </coordinate>
211 |       <optimal_force> 1000.0 </optimal_force>
212 |     </GeneralizedForce>
213 |   </objects>
214 |   <groups/>
215 | </ActuatorSet>

```


3 Computed Muscle Control

3.1 Overview: Computed Muscle Control (Residual Reduction):

What are CMC and RRA?

Both computed muscle control (CMC) and the residual reduction algorithm (RRA) are feedback-control based methods of estimating actuator controls such that actuator forces generate motion that tracks a desired trajectory (i.e. from experiment). Their similarity results in the algorithms sharing the same implementation (cmc.exe in OpenSim). They differ from standard feedback control methods in that the controls (forces) are solved via static optimization which enables redundant actuation and some deviation from “desired” kinematics. CMC also accounts for the delay in the generation of forces.

In RRA, static optimization is exploited to allow small deviations in kinematics (joint angles) that effect accelerations and further reduce the magnitude of residuals from inverse dynamics. In a separate step the total model mass of the model and location of the COM of a targeted segment (typically the most massive, e.g. torso) are adjusted to eliminate offsets in residual forces and moments.

3.2 Theoretical background:

1. We cast the inverse dynamics problem (joint torques from kinematics and external forces) as a tracking problem:

- a. Unlike IVD, tracking requires a dynamical system
 - $\mathbf{M}\ddot{q} = D(q, \dot{q}) + f$, joint angles and velocities determined by integrating accelerations
- b. Solve inverse system: f is computed based on accelerations and states, but
 - accelerations are difficult to measure but positions are easy
 - Double differentiation provides acceleration but also amplifies noise
 - Instantaneous forces and continuity over time not ensured
 - What if we use feedback to get better estimates of accelerations?

2. Feedback control:

1. (1) $\tilde{q} = \hat{q} - k_v(\dot{q} - \hat{\dot{q}}) - k_p(q - \hat{q})$ estimate acceleration using feedback
- (2) $f = \mathbf{M}\tilde{q} - D(q, \dot{q})$ compute controls (gen. forces, f)
- (3) integrate $\mathbf{M}\ddot{q} = D(q, \dot{q}) + f$

When actuator forces are torques, $f = \tau$, this is the method of computed torques.

2. For CMC, the production of torque is not instantaneous. The required torque at a given instant must be generated (in the case of muscles) over some prior time. To account for this, CMC changes the feedback law slightly, such that:

$$\tilde{q}(t+T) = \hat{q}(t+T) - k_v(\dot{q} - \hat{q}) - k_p(q - \hat{q})$$

where the desired acceleration at small time in the future (T) is based on the experimental acceleration at the future time and the current position and velocity errors.

3. In the general case, “controls” (gen. forces, f) are computed from static optimization:

$$\text{Min: } J(x) = \sum_{i=1}^{nx} \left(\frac{f_i(x_i)}{f_i^{opt}} \right)^2 + \sum_{j=1}^{nq} w_{q_j} (\tilde{q}_j(t+T) - \ddot{q}_j(t+T))^2$$

$$\text{Subject to: } \mathbf{M}\ddot{q} = D(q, \dot{q}) + f(x)$$

RRA: Compute torques and residuals (τ, R) that track the desired kinematics.

$$\tau, R \in f(x) = F^{opt} x$$

Min:

$$J(\tau, R) = \sum_{i=1}^{n\tau} \left(\frac{\tau_i^{opt} x_i}{\tau_i} \right)^2 + \sum_{k=1}^6 w_{R_k} \left(\frac{R_k^{opt} x_k}{R_k} \right)^2 + \sum_{j=1}^{nq} w_{q_j} (\tilde{q}_j(t+T) - \ddot{q}_j(t+T))^2$$

$$\text{subject to: } \mathbf{M}\ddot{q} = F(q, \dot{q}) + \tau + R$$

Enables us to vary tracking behavior between minimizing the residuals and tracking the kinematics and is solved as a bounded (limits on torques) nonlinear optimization. Since torques have no dynamics, T , can be small (e.g. $T=0.001s$)

CMC: Controls (x) are now excitations and torque production now has muscle dynamics:

$$\tau = [\mathbf{A}(q)]f_m(a(x), l, q, \dot{q}) + r$$

Joint torques, τ , are now replaced by the product of a muscle moment arm matrix, \mathbf{A} , and muscle forces, f_m , which is a function of muscle activation, a , which in turn is a function of excitation, x , the controls. (NOTE: “reserve” torques (actuators), r , remain in case muscle forces alone are unable to achieve the necessary accelerations.

Min:

$$J(\tilde{f}^m, x_{r,R}) = \sum_{i=1}^{nm} \left(\frac{\tilde{f}_i^m}{f_i^{opt}(\hat{q}, \hat{\dot{q}})} \right)^2 + \sum_{j=1}^{n\tau} \left(\frac{r_j^{opt} x_j}{r_j^{opt}} \right)^2 + \sum_{k=1}^6 \left(\frac{R_k^{opt} x_k}{R_k^{opt}} \right)^2 \Big|_{t+T}$$

Subject to: $\ddot{q}(t+T) = \tilde{\ddot{q}}(t+T)$ (using constraint is “fast target” in optimization)

$$\mathbf{M}\ddot{q} = F(q, \dot{q}) + \mathbf{A}f^m + r + R \Big|_{t+T}$$

$$f^{\min} \leq \tilde{f}^m \leq f^{\max} \Big|_{t+T}$$

Algorithm:

1. Set states: $\{a, l, q, \dot{q}\}$ at current time, t
2. Estimate future acceleration:
 $\tilde{\ddot{q}}(t+T) = \hat{\ddot{q}}(t+T) - k_v(\dot{q} - \hat{\dot{q}}) - k_p(q - \hat{q})$ $T = 0.01s$ provides sufficient time for a muscle to reach a target force.
3. Integrate muscle dynamics only with $(\hat{q}, \hat{\dot{q}})$ from t to $t+T$, with $x_i = [0,1]$ to get bounds on muscle forces $(f_i^{\min}, f_i^{\max}) \Big|_{t+T}$
4. Solve static optimization to get the desired muscle forces, \tilde{f}_i^m , at $t+T$.
5. Root solve for x from desired muscle forces, given:
 $f^m(a(x), l, q, \dot{q}) = \tilde{f}_i^m \Big|_{t+T}$
6. Integrate muscle and skeletal (multibody) dynamics for states at $t+T$ and loop.

Adjusting Mass Distribution:

1. residual force offsets can be reduced by adjusting total body mass
2. residual moment offsets can be reduced by adjusting moment due to gravity by moving COM location

3.3 Performing RRA in OpenSim: Key Elements

1. Actuators: specify the actuators (torques, for CMC they are muscles)
 - a. Forces and moments acting on the pelvis from ground are residual actuators
 - b. Internal joints are actuated by torques
 - c. Optimal forces are the maximum output of ideal actuators (torques, linear forces).
 1. Torque (force) applied is `optimal_force x control_value`

2. ControlConstraints: specify the bounds on the controls
 - a. Joint torques (and muscles) have a maximum magnitude of 1.
 - b. Residuals have bounds exceeding their anticipated force requirement.
 1. Weightings are implicit in this description. A high optimal_force means that large output force (torque) does not require a large control value (i.e. low cost). Conversely, residuals with low optimal force require high control values that incur higher costs.

3. Tasks: specify k_p , k_v , and w_{q_j} for each kinematic coordinate being tracked.
 - a. Selection of k_p and k_v are not arbitrary. They define the behavior of the error dynamics for each q as a second order linear system. We can write the k_p and k_v for the desired system behavior in terms of system poles, λ . For a (stable) critically damped system (real negative poles) $k_p = \lambda^2$ and $k_v = -2\lambda$.
 - b. w_{q_j} enables kinematics of joints (coordinates) for which we have high confidence (e.g. knee flexion, hip flexion) to be weighted more heavily compared to those of less confidence (e.g. hip internal rotation and ankle inversion).

4. RRA Setup File:
 - a. Model (lock subtalar and mtp joints, why?)
 - b. Actuators (NOTE: residual at pelvis should be applied at scaled location of COM)
 - c. Tasks (how strongly/loosely to track individual joint kinematics)
 - d. Adjust COM flags
 - e. Desired Kinematics
 - f. Control Constraints (bounds on the
 - g. External loads
 - Specify bodies that external loads are applied to.

5. Run RRA/CMC with slow walking IK and GRF data, and previously scaled model.

6. Trouble-shooting:
 - a. 4 important things to do in any case
 1. Check the pelvis COM location in Actuator files
 2. "Lock" the subtalar and mtp joints in *.osim file
 3. Make sure "use_fast_optimization_target" is
 - false (unchecked) for RRA
 - true (checked) for CMC
 4. Make sure "cmc_time_window" is
 - `<cmc_time_window> = T`

- 0.001 s for RRA
 - ~0.01 s for CMC
- b. Are the residual actuators (FX, FY, FZ, MX, MY, MZ) saturating?
1. Try increasing their min/max ranges in the ControlConstraints file.
 2. Check that the GRF are being applied.
- c. Is a dof tracking poorly?
- Consider increasing its weight in the Task file
- d. Does an actuator seem too weak?
- Consider increasing its optimal force in the Actuator file (torque actuators) or *.osim file (muscle actuators), but provide good justification for this. How does the optimal force influence the objective function?
- e. Post your question on the ME/BIOE 485 discussion forum. The teaching staff is notified when questions are posted.

4 Example: Computed Muscle Control

4.1 Using Computed Muscle Control

1. **Open Computed Muscle Control Tool.** To open the tool (**Error! Reference source not found.**), select **Compute Muscle Control...** from the **Tools** menu.
2. **Specify Filtered Input Motion.** Browse to the inverse kinematics directory and select the **Desired kinematics** file (e.g., `..\InverseKinematics\arm26_InverseKinematics.mot`), check the **Filter kinematics** option, and enter a cutoff frequency of 6 Hz.
3. **Specify Tracking Tasks.** Browse to select the tasks file (e.g., `arm26_ComputedMuscleControl_Tasks.xml`) specifying the joint coordinates for CMC to track and their relative weightings as well as the Kp and Kv gains on the errors.
4. **Specify Actuator Constraints.** Browse to select the constraints file (e.g., `arm26_ControlConstraints.xml`) specifying the maximum (or minimum) value used to constrain the allowed values of the actuator controls.
5. **Uncheck Adjust Model and Adjust Kinematics.** These options are used when performing residual reduction to obtain more dynamically consistent simulations. For our *Arm26* example, residual reduction is not necessary.
6. **Specify Time Range.** The **time range** for the forward simulation is specified and these should be set from 0 to 1 sec to correspond to the interval upon which the motion was found from inverse kinematics.
7. **Set CMC look-ahead window.** A time window of 0.01 is generally sufficient for muscle activations to change enough to produce the desired accelerations.
8. **Specify Output Directory.** Set the output **Directory** (e.g., `..\ComputedMuscleControl\Results`), so that you are able to compare the results of a CMC simulation with Forward Dynamics simulations using Nonphysiological and Physiological controls generated earlier by Static Optimization.

9. **Specify Additional Model Actuators.** Set the actuator settings to **Append** (rather than replace) and edit the **Additional actuator set files** field by adding an actuator set file (e.g., *arm26_Reserve_Actuators.xml*) containing reserve torques.
10. **Save Settings.** Use the **Settings** > button to save your settings to a setup file (e.g., *arm26_Setup_ComputedMuscleControl.xml*).
11. **Run CMC Tool.** You will see the model begin to move as muscles contract and accelerate the model. When the simulation has completed by reaching the end of the specified time range, the specified output directory will be populated by states files (e.g., *arm26_states_degrees.mot*) and the corresponding motion file (e.g., *arm26_states*) will be associated with the model in the GUI.

4.2 Changing the Desired Kinematics

1. **Create New Desired Kinematics.** Edit the original **Desired kinematics** file (e.g., *..\InverseKinematics\arm26_InverseKinematics.mot*) by specifying different shoulder elevation angle versus time (e.g., copy and paste elbow flexion values to shoulder elevation). Save the new file (e.g., *arm26_SpecifiedShoulder.mot*)
2. **Specify New Input Motion.** Browse and select the newly created **Desired kinematics** file (e.g., *arm26_SpecifiedShoulder.mot*), check the **Filter kinematics** option, and enter a cutoff frequency of 6 Hz.
3. **Run CMC Tool.** You will see the model begin to move to match the new kinematics.

4.3 Changing the Tracking Tasks

1. **Change Tracking Weight.** Edit the tasks file (e.g., *arm26_ComputedMuscleControl_Tasks.xml*) or use the Property Editor to reduce the *r_shoulder_elev weight* from 1.0 to 0.0.
2. **Specify New Tracking Tasks.** Browse and select the new tasks file (e.g., *arm26_ComputedMuscleControl_Tasks.xml*) or save new task from the Property Editor.
3. **Run CMC Tool.** You will see the model begin to move while tracking the elbow flexion well but not the shoulder elevation.

4.4 Changing the Actuator Constraints

1. **Change Actuator Constraint.** Edit the constraints file (e.g., *arm26_ControlConstraints.xml*) or use the Excitation Editor to limit BIClong control to a maximum of 0.05.
2. **Specify Actuator Constraints.** Browse and select the new constraints file (e.g., *arm26_ControlConstraints.xml*) or save new constraint from the Excitation Editor.
3. **Run CMC Tool.** You will see the model begin to move by using other actuators in lieu of BIClong.

4.5 Changing the CMC Look-Ahead Window

1. **Change CMC Look-Ahead Window.** Set the look-ahead time to 0.5 rather than 0.01.
2. **Run CMC Tool.** You will see the model begin to move, but the control will be too coarse to allow for good tracking.