VICON LIFE SCIENCES

VICON PROCALC GENERATING VSKS TUTORIAL

WHAT'S INSIDE?

About this tutorial 2

About VSK/VST files 3

About the data for this tutorial 4

About the files for this tutorial 5

About the Conventional Gait Model 2.3 6

Updating the joints in the VST 8

Defining the hip model 10

Defining the femur segments 16

Defining the tibia segments 18

Defining the foot segments 19

Defining the toe segments 20

Complete variable scheme 21

Using the variable scheme to update VSKs 22

Troubleshooting generating VSKs with ProCalc 24

Vicon Motion Systems is an Oxford Metrics plc company.
Email: support@vicon.com Web: http://www.vicon.com

About this tutorial

ProCalc 1.3 and later enables you to generate subject-specific Vicon Skeleton (VSK) files based on calculations specified within a ProCalc variables scheme. This tutorial explains how to do this and the benefits of this approach.

This tutorial uses as a Vicon Skeleton Template (VST) file that is based on version 2.3 of the python Conventional Gait Model (see About the Conventional Gait Model 2.3, page 6). The segments are the same but the knee joint type was changed from a 1-degree-of-freedom (hinge) joint to a 3-degree-of-freedom (ball) joint. If you want to use a different VST, make sure all joint types are set to ball joints (see Updating the joints in the VST, page 8).

The goal of this tutorial is to:

- Learn how to build a biomechanical model that can be used to generate a VSK
- Learn how to generate a VSK if you already have a suitable, existing ProCalc variables scheme

About VSK/VST files

The Vicon VSK/VST file format is primarily used in the Vicon Nexus software to specify *labeling skeleton templates.* The labeling skeleton contains a set of relationships between motion-captured markers and an underlying skeletal structure, which Vicon Nexus uses to automatically label (auto-label) markers. Users of Vicon Nexus select a template (VST), and then scale/calibrate the template to the current subject that is being captured (which generates a corresponding VSK file), so that the auto-labeler can do its job.

The VST/VSK file format is a generic format that can be used to specify any relationship between markers and segments, including those that define biomechanical models – in other words, where the segments' positions estimate anatomical segments. ProCalc enables you to specify the marker-segment relationships that are normally used in a biomechanical model in the VST/VSK file format without having to manually edit a text file. Instead, you generate a VSK from a model that is defined in ProCalc.

Generating a VSK from ProCalc has two major benefits:

- When you set up a customized biomechanical model in ProCalc, you can use the same model for labeling.
- You can improve the labeling for any VST, or generate real-time kinematics.

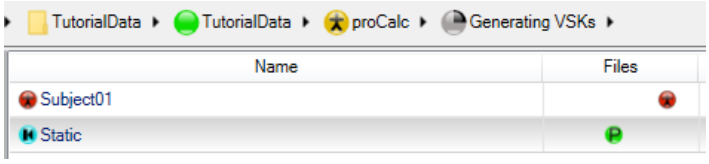
> **Important**
> Some of the default VST files that are supplied with Nexus use a 1-degree-of-freedom (hinge) joint at the knee. Before using the VSK with ProCalc as described in this tutorial, replace the 1-degree-of-freedom joint with a 3-degree-of-freedom (ball) joint. For more information, see Updating the joints in the VST, page 8.

About the data for this tutorial

To follow this tutorial, navigate to *C:\Program Files\Vicon\ProCalc\Help* and unzip the *ProCalc Tutorials Data* file. Choose a path or directory that has sufficient read/write permissions and that you can remember and access easily (e.g. *C:\Users\Public\Documents\Vicon*). If you have completed a ProCalc tutorial in the past, you may have already completed this step.

For this tutorial, we will use the *Generating VSKs* session. Within this folder, there is one trial (*Static.c3d*), two ProCalc schemes and a VST file (see About the files for this tutorial, page 5).

To access the trials in ProCalc, your hierarchy must look like the following example, though the specific path will depend on where you extracted your data.



About the files for this tutorial

This tutorial folder features other files that must be moved to the appropriate folders. With ProCalc closed, copy the files from the session folder into the following locations:

File name	Description	Location
GeneratingVSKs.vst	Labeling skeleton template	C:\Users\Public\Documents\Vicon\Nexus2.x\ModelTemplates
GeneratingVSKs.InputParamScheme	Input parameters scheme	C:\Users\Public\Documents\Vicon\Eclipse\InputParamSchemes
GeneratingVSKs.VarScheme	Variables scheme	C:\Users\Public\Documents\Vicon\Eclipse\VariableSchemes

These files contain complete information for this tutorial, so you can follow the instructions without having to enter additional data.

About the Conventional Gait Model 2.3

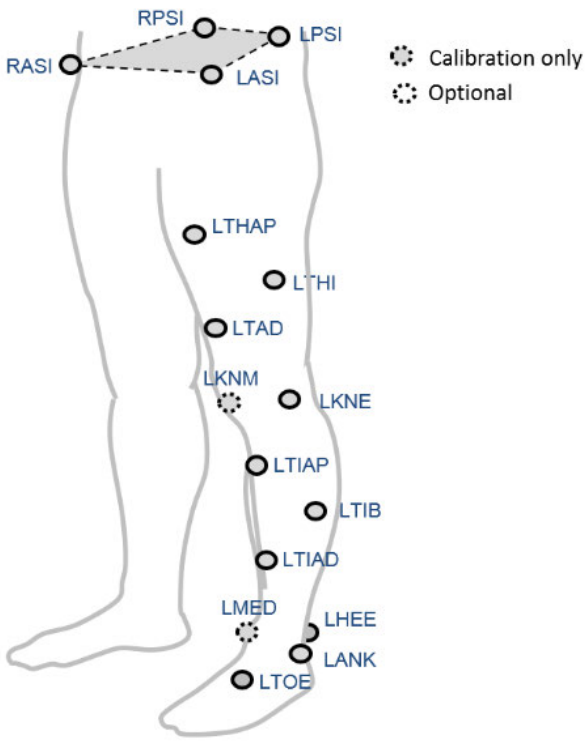
One of the main applications of ProCalc is to build and apply biomechanical models. If this is your primary aim, it is important to know which particular biomechanical model you wish to use, as this will govern the types of calculations defined within ProCalc.

In this tutorial, we have selected the Python Conventional Gait Model (CGM) version 2.3, also known as CGM2.3 or pyCGM2.3. This version of CGM has been developed by Fabien Leboeuf at Salford University, UK.

For more information, see:

https://pycgm2.github.io/pages/CGM23-Overview.html

The marker set for CGM2.3 looks like this (for clarity, only the left side is shown):



The marker set is similar to earlier versions of CGM, but uses medial markers at the knee and ankle to determine the frontal planes of the femur and tibia.

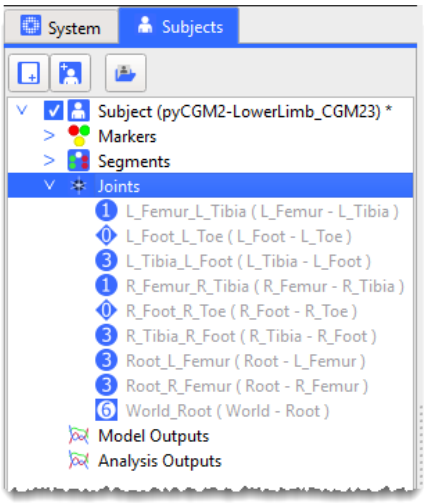
Extra tracking markers are used on the anterior aspect of the thighs and shanks.

Updating the joints in the VST

Several of the default labeling templates, including the one that is supplied for CGM2.3, use a hinge joint (a single degree-of-freedom joint) at the knee. Hinge joints work well to constrain the VST when it's used for labeling only, but because CGM uses a ball joint (three degrees of freedom) for all joints, you must update the labeling template (VST) to use ball joints.

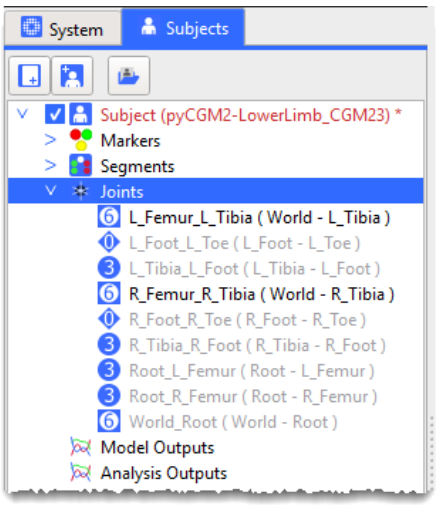
To update the joints in a VST:

1. Load a static trial and create a new subject based on the labeling template.
2. Reconstruct and label the static trial.
3. Run a **Kinematic Fit** operation on the trial.
   The current segments and joints are fitted to the marker data.
4. In the **Subjects Resources** pane, expand the subject node and then expand the **Joints** node to display the joints in the template.

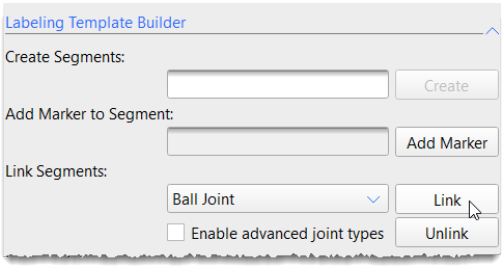


   The hinge joints are clearly labeled with a **1** symbol.

5. Select the hinge joints, then right-click and select **Unlink Joint**.
   The symbols change from 1 to 6 to indicate that the hinge joints have been replaced with 6-degree of freedom joints:



6. Re-link these joints using the **Link Segments** tool in the Labeling Template Builder. To do this, on the **Subject Preparation Tools** pane, in the **Labeling Template Builder** section, select **Ball Joint** and click **Link**:



A tip next to the mouse pointer prompts you to select the parent segment.

7. In the **Subjects Resources** pane, expand the **Segments** node of the subject.
8. Click on the **L_Femur** segment to select it.
   You are prompted to select the child segment.
9. Click on the **L_Tibia** segment.
   The list of joints now shows that the **L_Femur_L_Tibia** joint is now a ball joint with the symbol **3**.
10. Repeat Steps 8-9 for the right side.
11. Right-click the subject and select **Save Labeling Skeleton as Template**.
12. Select a new name for the template, then save.
    The template has now been updated with ball joints instead of hinge joints for the knees.

Defining the hip model

Before you begin, ensure you have unzipped the ProCalc Tutorials data as described in About the data for this tutorial, page 4.

Start ProCalc, and on the **Data Management** tab, navigate to the **Generating VSKs** folder and double-click the trial called **Static** to load it.

When the trial is loaded, the labeled markers that reflect the image illustrated in About the Conventional Gait Model 2.3, page 7 are displayed.

You now need to create two new schemes:

- Input parameters scheme
- Variables scheme

To create the schemes, on the relevant tabs (**Input Parameters** and **Variables**), click the Create button  to the right of the **Scheme** drop-down menu and enter an appropriate name for the scheme, eg, **CGM2.3**.

CGM2.1 (on which 2.3 is based) introduces the hip model – it is based on a paper by *Hara et al* (2016) that proposes a simple regression equation, using the Leg Length as the only variable:

$$HJC_X = 11 - 0.063LL$$
$$HJC_Y = 8 + 0.086LL$$
$$HJC_Z = -9 - 0.078LL \quad \text{(with LL: leg length in mm)}$$

Define the input parameters

First, you need to define the subject parameter named Leg Length. If you used the standard CGM2 VST file when you defined your subject in Nexus, as in the supplied tutorial files, add the following parameters. (If you used a different VST, ensure you have replaced any hinge joints with ball joints, as described in Updating the joints in the VST, page 8.) :

1. On the **Input Parameter** tab, create or select the CGM2.3 scheme.
2. To add a parameter, click the Edit button .
3. Click the **Add** button Add in the bottom left corner to add another parameter.
4. From the **Subject Measurements** drop-down menu at the bottom, select **LLegLength**, then click **Use**.
5. Ensure that the following properties are set:
   - **Quantity**: Length
   - **Unit**: mm
6. Repeat steps 3-5, but this time with **RLegLength**.
7. With the two parameters in the list, click the Save button at the top to save the scheme.

Add hip model variables to the variables scheme

To define the hip model variables, complete the following steps:

- Define the hip joint center regression equations, page 13
- Define the reference coordinate system for the hip joint center, page 14
- Define the hip joint centers, page 15

Define the hip joint center regression equations

To define the hip joint center regression equations:

1. On the **Variables** tab, create or select the CGM2.3 scheme.
2. Click the **Add** button 

to add a new variable.
3. Name the variable **LHJCx**.
4. Select function **Arithmetic** and then **Add: A + B**.
5. For **A**, choose type **Length** and then **LLegLength**.
6. For **B**, choose type **Length** and then **1mm**.
7. In the **Factor** column for **A**, enter **-0.063**.
8. In the **Factor** column for **B**, enter **11**.
   The calculated value is displayed in the log.
9. Repeat steps 3-8 for LHJCy and LHJCz, making sure that you enter the correct factors according to the above equations.
10. Repeat steps 3-8 for RHJCx, RHJCy and RHJCz. However, make sure that you *negate* the factors for RHJCy (Y is the mediolateral direction, so will be the opposite for left and right).

> **Tip**
> To generate a right-side variable that is equivalent to a left-side variable that you have defined (or vice versa), use the **Copy** button 

and then the Mirror button 

.

At the end of this step, six variables are defined, as follows:

Variable Name	Type	Definition
H		
LHJCx	Length	Add: LLegLength*-0.063 + 1mm*11
LHJCy	Length	Add: LLegLength*0.086 + 1mm*8
LHJCz	Length	Add: LLegLength*-0.078 + 1mm*-9
RHJCx	Length	Add: RLegLength*-0.063 + 1mm*11
RHJCy	Length	Add: RLegLength*-0.086 + 1mm*-8
RHJCz	Length	Add: RLegLength*-0.078 + 1mm*-9

Define the reference coordinate system for the hip joint center

The reference coordinate system will be the *Pelvis* coordinate system. You define the Pelvis coordinate system using the mid-point between the ASIS points as the origin, the RASI-LASI line as the main defining (Y) axis, and the mid-point between LPSI and RPSI to define the pelvic plane:

1. Define a new point named **PelvisFront** with:
   - Function: **Point** then **Halfway between A and B**
   - A – Type: **Point**, Input Variable: **LASI**
   - B – Type: **Point**, Input Variable: **RASI**
2. Similarly, define a new point named **PelvisBack** halfway between **LPSI** and **RPSI**.
3. Create a new point named **PelvisOrigin** with:
   - Function: **Point** and **Distance A from point B towards Point C**
   - A – Type: **Length**, Input Variable: **1mm**, Fact: A factor that corresponds to half the marker diameter plus the thickness of the marker's plastic base. For example, if you use 14 mm markers with a plastic base that is 1 mm thick, specify a factor of 8. The idea is to offset the pelvic origin posteriorly to account for the diameter of the LASI/RASI markers.
   - B - Type: **Point**, Input Variable: **PelvisFront**
   - C - Type: **Point**, Input Variable: **PelvisBack**
4. Create the Pelvis segment, which must be named **Root** (This is because the segment is named **Root** in the VSK file that we want to update, and the names must match.) To do this, you must create the two vectors that are needed to define the segment, the mediolateral axis and the anterior-posterior one. First, add a new variable named **PelvisML** with:
   - Function: **Vector** then **From point A to point B**
   - A – Type: **Point**, Input Variable: **RASI**
   - B – Type: **Point**, Input Variable: **LASI** (the vector must point towards the left)
5. Add a similar variable **PelvisAP**, but with A: **PelvisFront** and B: **PelvisBack**.
6. Add a new variable named **Root** with:
   - Function: **Segment** and **Origin A, Y-Axis=B, Z-Axis=B x C**
   - A: **PelvisOrigin**,

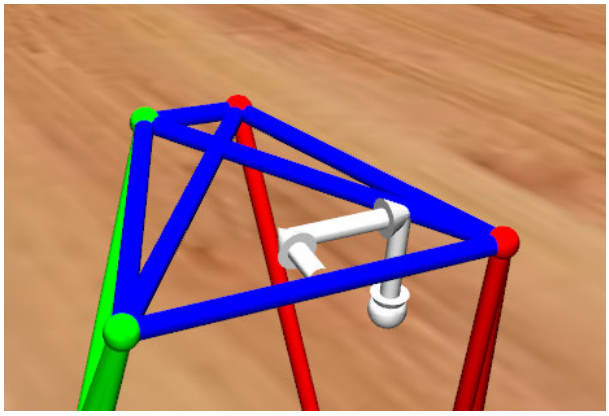
- B: PelvisML
- C: PelvisAP

Because the cross-product of the ML and AP vectors points up, the pelvis segment is now correctly defined, with the x-axis pointing forward, the y-axis pointing left and the z-axis up.

Define the hip joint centers

1. Add a new variable named LHJC with:
   - Function: Point and {A, B, C} in Segment D's local coordinates
   - A, B and C: LHJCx, LHJCy and LHJCz
   - D: Root
2. Repeat 1-2 for a new variable named RHJC.
3. Make sure that the LHJC and RHJC are displayed in reasonable positions in the 3D workspace.

The following image shows the LHJC relative to the pelvic origin.



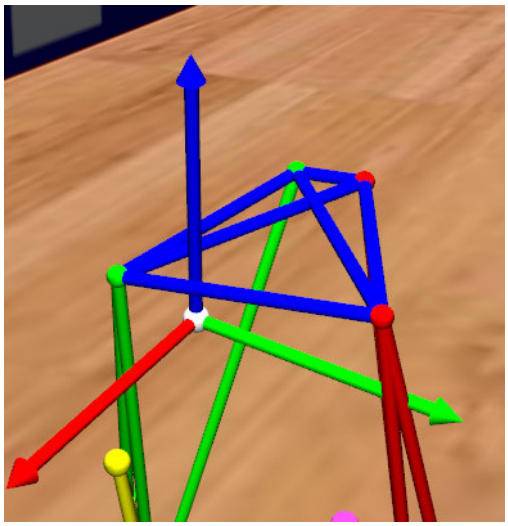
Defining the femur segments

> **Important**
> Segments must be defined with the origin at the proximal end. Sometimes, biomechanical models use the distal end as the origin for segments (eg, the knee joint center for the femur). For successful VSK generation, you must define all segments using the proximal end as the origin (i.e. the hip joint center for the femur).

You already have the origins for the two femur segments defined: LHJC and RHJC (see Defining the hip model, page 10). For the distal end, you need the knee joint centers. These are defined as the half-way point between the lateral and medial knee markers. You can then define the axes that define the femur segments:

1. On the **Variables** tab, with the CGM2.3 scheme selected, add a new variable named **LKJC**. Define this as a point halfway between **LKNE** and **LKNM**.
2. Repeat Step 1 for the right side.
3. Add a new vector named **LFemurPD** from point A=**LKJC** to point B=**LHJC.**
4. Add a new vector named **LFemurML** from point A=**LKNE** to point B=**LKNM.**
5. Use these two vectors to define the femur segment, which must be named **L_Femur** (to match the name of the segment in the VSK file). Add a new segment, name it **L_Femur**, select **Origin A, Z-axis=B, X-Axis=BxC**, then A=**LHJC** (remember, at the proximal end), B=**LFemurPD** and C=**LFemurML**.
6. Repeat Steps 3-5 for the right side to define **R_Femur**.
   Note that if you define the **RFemurML** from **RKNE** to **RKNM**, this vector points in the opposite direction (left) compared to the left side, which means that the **R_Femur** will have its Y-axis pointing right and its X-axis pointing backwards. To correct this, either use a factor -1 for the **RFemurML** in the **R_Femur**'s specification, or flip the **RFemurML** vector itself.

The following image shows the coordinate system of the R_Femur segment:



Defining the tibia segments

Define the tibia segments in the same way as the femur segments, page 16:

- Calculate the ankle joint centers halfway between the lateral and medial ankle markers.
- Use the LAJC/RAJC to LKJC/RKJC vectors as the proximal/distal (major) axes.
- Use the LANK/RMED to LMED/RANK vectors as the mediolateral (minor) axes for the L_Tibia and R_Tibia segments.

Defining the foot segments

Define the foot segments using the anterior-posterior axis LHEE/RHEE to LTOE/RTOE as their major axes.

Note that the minor axis in CGM is not yet defined in CGM2.3: this is added in the next version (CGM2.4). (This can also be implemented in ProCalc, but is not part of this tutorial.)

To define the feet:

1. Create two vectors, LFootAP and RFootAP, from LHEE/RHEE to LTOE/RTOE.
2. Create two segments, L_Foot and R_Foot, using Origin A, X-axis=B, Z-Axis=BxC, specifying A=LAJC/RAJC, B=LFootAP/RFootAP and C=LTibiaML/RTibiaML.
   When defining the L_Foot, flip the LTibiaML.

> **Tip**
> To define foot flat for the static trial, when defining the feet, change the XYZ drop-down for the LFootAP/RFootAP and LTibiaML/RTibiaML vectors to XY instead of XYZ.
> The Z-coordinate of the vectors is ignored, and the foot is modeled as flat on the ground in the static trial.

Defining the toe segments

The toe segments are used only as leaf segments for display purposes: they do not have any other function in the model. Therefore, they are aligned with the foot segment, but displaced to have their origins near the LTOE/RTOE markers.

To define the toes:

1. Define a line called **LeftFootLine** using the function type **Line** and **From point A in direction of Vector B**.
2. Select A = **LAJC** (the left ankle joint center), change B's Type to **Segment**, the Input Variable to **L_Foot** and the XYZ to **X**. In other words, the line starts in the ankle joint center, and is defined in the direction of the left foot's X-axis.
3. Add a new point **LTJC** using Type **Point** and **Project: A onto B**. For A, select **LTOE** and for B, select the **LeftFootLine** defined above.
4. Create a new segment **L_Toe** using the exact same definition as for **L_Foot** already defined except that the origin point should be **LTJC** instead of LAJC.
5. Repeat Steps 1-4 for the right side.

Complete variable scheme

This is the complete scheme for this tutorial, when sorted by **Type** (to sort, on the **Variables** tab, click the table header):

Variable Name	Type	Definition
Length		
LHJCx	Length	Add: LLegLength*-0.063 + 1mm*11
LHJCy	Length	Add: LLegLength*0.086 + 1mm*8
LHJCz	Length	Add: LLegLength*-0.078 + 1mm*-9
RHJCx	Length	Add: RLegLength*-0.063 + 1mm*11
RHJCy	Length	Add: RLegLength*-0.086 + 1mm*-8
RHJCz	Length	Add: RLegLength*-0.078 + 1mm*-9
Point		
LAJC	Point	Halfway between LANK and LMED
LHJC	Point	{LHJCx, LHJCy, LHJCz} in Segment Root's local coordinates
LKJC	Point	Halfway between LKNE and LKNM
PelvisBack	Point	Halfway between LPSI and RPSI
PelvisFront	Point	Halfway between LASI and RASI
PelvisOrigin	Point	Distance 1mm*7 from point PelvisFront towards point PelvisBack
RAJC	Point	Halfway between RANK and RMED
RHJC	Point	{RHJCx, RHJCy, RHJCz} in Segment Root's local coordinates
RKJC	Point	Halfway between RKNE and RKNM
Segment		
L_Femur	Segment	Origin LHJC, Z-Axis=LFemurPD, X-Axis=LFemurPD x LFemurML
L_Foot	Segment	Origin LAJC, X-Axis=LFootAP(XY), Z-Axis=LFootAP(XY) x LTibiaML(XY)*-1
R_Foot	Segment	Origin RAJC, X-Axis=RFootAP(XY), Z-Axis=RFootAP(XY) x RTibiaML(XY)
L_Tibia	Segment	Origin LKJC, Z-Axis=LTibiaPD, X-Axis=LTibiaPD x LTibiaML
R_Femur	Segment	Origin RHJC, Z-Axis=RFemurPD, X-Axis=RFemurPD x RFemurML*-1
R_Tibia	Segment	Origin RKJC, Z-Axis=RTibiaPD, X-Axis=RTibiaPD x RTibiaML*-1
Root	Segment	Origin PelvisOrigin, Y-Axis=PelvisML, Z-Axis=PelvisML x PelvisAP
Vector		
LFemurML	Vector	From point LKNE to point LKNM
LFemurPD	Vector	From point LKJC to point LHJC
LFootAP	Vector	From point LHEE to point LTOE
LTibiaML	Vector	From point LANK to point LMED
LTibiaPD	Vector	From point LAJC to point LKJC
PelvisAP	Vector	From point PelvisFront to point PelvisBack
PelvisML	Vector	From point RASI to point LASI
RFemurML	Vector	From point RKNE to point RKNM
RFemurPD	Vector	From point RKJC to point RHJC
RFootAP	Vector	From point RHEE to point RTOE
RTibiaML	Vector	From point RANK to point RMED
RTibiaPD	Vector	From point RAJC to point RKJC

This is everything that's needed to generate a VSK that contains the marker-segment relationships defined in CGM2.3.

Using the variable scheme to update VSKs

When your variable scheme is finished, you can use it to update VSK with your new joint centers and segment orientations.

To update a VSK with the variable scheme:

1. In Nexus, create a subject in the normal way (ie, from a labeling template VST file), and enter any required subject measurements.
2. Capture or load a static trial, and label it, either by using the **Autolabel Static Frame** function, or manually.

   > **Important**
   > It is NOT necessary to run the other operations that scale/calibrate the subject.

3. When the static trial has been correctly labeled, save the trial by clicking the **Save** button, so that both the subject's VSK and the C3D are saved.
4. In ProCalc 1.3 or later, load the trial.
5. Specify any relevant **Input Parameters** or **Variables** schemes that are used in defining your model
6. On the **Data Management** tab, click the **Update VSK** button.
7. Review and accept any prompts that are displayed.
   When the operation starts, information about the progress is output to ProCalc's log.
8. In Nexus, do one of the following:
   - Nexus 2.9 and later: Right-click the subject and select **Refresh Subject** then **From VSK**.
     or
   - Earlier versions of Nexus: Exit the session and open the session again (or re-start Nexus).
9. In the **Subject Viewer**, make sure that the subject is updated.

You can now use the updated VSK to:

- Label dynamic trials.
- Generate kinematics in real-time or offline.
  To generate real-time kinematics, make sure the processing level in Nexus is set to Kinematic Fit (**Local Vicon System** > **Processing Level**).

  To generate offline kinematics, either click the Nexus **KinFit** button 

  or run the **Kinemetic Fit** pipeline.

Troubleshooting generating VSKs with ProCalc

The following tips will help you to generate and use VSKs successfully:

- On the Vicon ProCalc **Input Parameters** tab and **Variables** tab, make sure that you have selected the required option from the **Scheme** drop-down menu.
- Make sure that the static trial has been correctly labeled.
- Ensure that all the segments that you have defined in ProCalc have exactly the same names as the segments in the VST/VSK.
- After you've clicked **Update VSK**, review the log in ProCalc.
  All markers and segments that have been updated are reported.
- Check that the VST has *only 3- or 6-degree-of-freedom links* between the different segments. If the VST includes a 1- or 2-degree-of-freedom joint, replace it with a 3-degree of freedom joint.
  To do this, in Nexus, in the **Subjects Resources** tree, right-click the joint, and select **Unlink Joint**. To re-link the two segments, use the **Labeling Template Builder** on the **Subject Preparation Tools** tab. For details, see Updating the joints in the VST, page 8.
- To make sure that you have updated the subject after ProCalc has run:
  - Nexus 2.9 and later: Right-click the subject and select **Refresh Subject** then **From VSK**.
  - Earlier versions of Nexus: Exit and re-enter the session (or re-start Nexus).