Virtual Lab Direction of Progression Method 3

This script is explained in the Virtual Lab Direction of Progression tutorial. This script outlines Method 3.

Before running this script:

- A static trial must be loaded into the workspace
- A model template (with modified virtual lab) must be loaded into the workspace
- A dynamic trial must be loaded into the workspace

```
**__Add_a_Comment__**
! /COMMENT= Step 1: Direction of Progression
;

!  Find the location of the model's center of mass at the beginning of the
trial
Event_Explicit
/EVENT_NAME=TRIAL_START
/FRAME=50
! /TIME=
;

Event_Explicit
/EVENT_NAME=TRIAL_END
/FRAME=EOF-40
! /TIME=
;

Compute_Model_Based_Data
/RESULT_NAME=MODEL_COG
/FUNCTION=MODEL_COG
/SEGMENT=
/REFERENCE_SEGMENT=
! /RESOLUTION_COORDINATE_SYSTEM=LAB
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
```

```
;

Metric_Signal_Value_At_Event
/SIGNAL_TYPES=LINK_MODEL_BASED
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=MODEL_COG
/RESULT_METRIC_FOLDER=PROG
/RESULT_METRIC_NAME=MODEL_COG_START
! /APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
/EVENT_NAME=TRIAL_START
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /GENERATE_VECTOR_LENGTH_METRIC=FALSE
! /RETAIN_NO_DATA_VALUES=FALSE
;

Metric_Signal_Value_At_Event
/SIGNAL_TYPES=LINK_MODEL_BASED
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=MODEL_COG
/RESULT_METRIC_FOLDER=PROG
/RESULT_METRIC_NAME=MODEL_COG_END
! /APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
/EVENT_NAME=TRIAL_END
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /GENERATE_VECTOR_LENGTH_METRIC=FALSE
! /RETAIN_NO_DATA_VALUES=FALSE
;

Evaluate_Expression
/EXPRESSION= ( METRIC::PROG::MODEL_COG_END - METRIC::PROG::MODEL_COG_START )
/RESULT_TYPE=METRIC
/RESULT_FOLDER=PROG
/RESULT_NAME=DIR_PROG_Y_2
;

! Create a unit vector of the differences
!   There should be no progression in the vertical direction so set Z=0

Evaluate_Expression
/EXPRESSION= UNIT_VECTOR ( METRIC::PROG::DIR_PROG_Y_2::X ,
METRIC::PROG::DIR_PROG_Y_2::Y , 0 )
/RESULT_TYPE=METRIC
/RESULT_FOLDER=PROG
/RESULT_NAME=DIR_PROG
```

```
;

**__Add_a_Comment__**
! /COMMENT= Step 3: Define the trunk angle relative to the virtual lab
;

! Define rotation from Lab to VLab
!    Z component is perpendicular to the ground
!    Y component is the direction of progression
!    X component is Y cross Z

! The rotation from Lab to VLab is defined using the direction of
progression define in Step 1 and these values will be used to define the
joint angles

! Define rotation from Lab to Trunk
!    Using Compute Model Based Data command select the Joint_Rotation
function and resolve the Thorax/Ab rotation in the Lab coordinate system

Compute_Model_Based_Data
/RESULT_NAME=TRUNK_ROT
/FUNCTION=JOINT_ROTATION
/SEGMENT=RTA
! /REFERENCE_SEGMENT=LAB
/RESOLUTION_COORDINATE_SYSTEM=
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
;

**__Add_a_Comment__**
! /COMMENT=
;

!  Solve for the rotation between VLab and the trunk segment

Evaluate_Expression
/EXPRESSION= ( METRIC::PROG::DIR_PROG::1 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::7 ) + ( METRIC::PROG::DIR_PROG::2 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::8 )
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=R32
;
```

```
Evaluate_Expression
/EXPRESSION= LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::9
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=R33
;

Evaluate_Expression
/EXPRESSION= ( METRIC::PROG::DIR_PROG::2 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::7 ) - ( METRIC::PROG::DIR_PROG::1 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::8 )
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=R31
;

Evaluate_Expression
/EXPRESSION= ( METRIC::PROG::DIR_PROG::2 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::1 ) - ( METRIC::PROG::DIR_PROG::1 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::2 )
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=R11
;

Evaluate_Expression
/EXPRESSION= ( METRIC::PROG::DIR_PROG::2 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::4 ) - ( METRIC::PROG::DIR_PROG::1 *
LINK_MODEL_BASED::ORIGINAL::TRUNK_ROT::5 )
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=R21
;

! Solve for the joint angles:
!    Solve for Theta X
!    Solve for Theta Y
!    Solve for Theta Z

Evaluate_Expression
/EXPRESSION= ATAN ( - DERIVED::TRUNK_ANGLE::R32 / DERIVED::TRUNK_ANGLE::R33
)
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=THETA_X
;

Evaluate_Expression
/EXPRESSION= ATAN ( DERIVED::TRUNK_ANGLE::R31 / SQRT (
(DERIVED::TRUNK_ANGLE::R11 ^ 2) + (DERIVED::TRUNK_ANGLE::R21 ^ 2) ) )
```

```
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=THETA_Y
;

Evaluate_Expression
/EXPRESSION= ATAN ( - DERIVED::TRUNK_ANGLE::R21 / DERIVED::TRUNK_ANGLE::R11
)
/RESULT_TYPE=DERIVED
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAME=THETA_Z
;

! Convert from radians to degrees

Multiply_Signals_By_Constant
/SIGNAL_TYPES=DERIVED+DERIVED+DERIVED
/SIGNAL_FOLDER=TRUNK_ANGLE+TRUNK_ANGLE+TRUNK_ANGLE
/SIGNAL_NAMES=THETA_X+THETA_Y+THETA_Z
! /RESULT_TYPES=
/RESULT_FOLDER=TRUNK_ANGLE
/RESULT_NAMES=++
/RESULT_SUFFIX=_DEG
! /SIGNAL_COMPONENTS=
/CONSTANT= ( 180 / pi() )
;
```