Vector Coding

The **Vector Coding** command can be used to determine inter-segment coordination with the output being an angle from 0-360 degrees known as the coupling angle. This command is based on work from Needham 2015. More information available in his paper here.

Vector Coding Process

The Following section will outline the equations and steps followed while conducting the vector coding.

Coupling Angle Calculation

The command is based on changes in consecutive segment angles for proximal and distal segments of a given joint. **Equation 1** and **Equation 2** are used to calculate the coupling angle, with the equation used determined by the difference in consecutive angles for the proximal segment. If this result is greater than zero **Equation 1** is used and if the result is less than zero **Equation 2** is used.

$$\gamma_i = \left(\frac{\theta_{D(i+1)} - \theta_{Di}}{\theta_{P(i+1)} - \theta_{Pi}} \right) * \frac{180}{\pi} \qquad \theta_{P(i+1)} - \theta_{Pi} > 0 \qquad (1)$$

$$\gamma_i = \left(\frac{\theta_{D(i+1)} - \theta_{Di}}{\theta_{P(i+1)} - \theta_{Pi}} \right) * \frac{180}{\pi} + 180 \qquad \theta_{P(i+1)} - \theta_{Pi} < 0 \qquad (2)$$

Angle Conditions

Equation 3 displays conditions that are applied to the angle to avoid inaccuracies from special cases where there is no change in angle at consecutive points.

$$\gamma_i \begin{cases} \gamma_i = 90 & \theta_{D(i+1)} - \theta_{Di} = 0 \quad and \quad \theta_{P(i+1)} - \theta_{Pi} > 0 \\ \gamma_i = -90 & \theta_{D(i+1)} - \theta_{Di} = 0 \quad and \quad \theta_{P(i+1)} - \theta_{Pi} < 0 \\ \gamma_i = -180 & \theta_{D(i+1)} - \theta_{Di} < 0 \quad and \quad \theta_{P(i+1)} - \theta_{Pi} = 0 \\ \gamma_i = Undefined & \theta_{D(i+1)} - \theta_{Di} = 0 \quad and \quad \theta_{P(i+1)} - \theta_{Pi} > 0 \end{cases} \qquad (3)$$

Angle Correction

The result from **Equations 1 and 2** must be corrected to a value between 0 and 360 degrees. **Equation 4** is used to make this correction.

$$\gamma_i \begin{cases} \gamma_i + 360 & \gamma_i < 0 \\ \gamma_i & \gamma_i \geq 0 \end{cases} \quad (4)$$

Pipeline Command

The **Vector_Coding** command can be found in the Pipeline Workshop within the **Signal Process** folder as so:

```
Vector_Coding
! /SIGNAL_OWNER1=
/SIGNAL_TYPE1=
/SIGNAL_FOLDER1=
/SIGNAL_NAME1=
! /SIGNAL_COMPONENT1=
! /SIGNAL_OWNER2=
/SIGNAL_TYPE2=
/SIGNAL_FOLDER2=
/SIGNAL_NAME2=
! /SIGNAL_COMPONENT2=
! /RESULT_OWNER=
! /RESULT_FOLDER= PROCESSED
/RESULT_NAME=
;
```

Command Parameters

The parameters that can be used to control the command are as follows:

Signal Owner	The name of the file containing the given signal
Signal Type	The type of signal (force, target, link_model_based, etc.)
Signal Folder	The name of the folder containing the given signal
Signal Component	Which component of the signal is used
Result Owner	The name of the file that will contain the result
Result Folder	The name of the folder containing the resulting signal(s)
Result Name	The name of the resulting signal

Understand the parameters of the Vector_Coding command:

SIGNAL1

The /SIGNAL_OWNER1, /SIGNAL_TYPE1, /SIGNAL_FOLDER1, and /SIGNAL_NAME1 parameters allow the user to specify which signal (by file, data type, folder, and name) should be used as the first signal for the Vector_Coding calculation. The proximal segment angle should be chosen as signal 1.

SIGNAL2

Similarly, the /SIGNAL_OWNER2, /SIGNAL_TYPE2, /SIGNAL_FOLDER2, and /SIGNAL_NAME2 parameters allow the user to specify the second signal for the Vector_Coding calculation. The distal segment angle should be chosen as signal 2.

RESULT

The /RESULT_OWNER, /RESULT_FOLDER, and /RESULT_NAME parameters allow the user to specify where the result of the vector coding should be saved.

- The result's data type the DERIVED folder.
- The result's folder defaults to being the PROCESSED folder.

Example: Performing Vector Coding on the Ankle

```
Compute_Model_Based_Data
/RESULT_NAME= RIGHT_FOOT_ANGLE
/FUNCTION=SEG_PROGRESSION_ANGLE
/SEGMENT=Right Foot
! /REFERENCE_SEGMENT=LAB
! /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
! /INCLUDE_REMOTE_ANGULAR_MOMENTUM=FALSE
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;

Compute_Model_Based_Data
/RESULT_NAME= RIGHT_SHANK_ANGLE
```

```
/FUNCTION=SEG_PROGRESSION_ANGLE
/SEGMENT= Right Shank
! /REFERENCE_SEGMENT=LAB
! /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
! /INCLUDE_REMOTE_ANGULAR_MOMENTUM=FALSE
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;

Vector_Coding
/SIGNAL_OWNER1=*.c3d
/SIGNAL_TYPE1=LINK_MODEL_BASED
/SIGNAL_FOLDER1=ORIGINAL
/SIGNAL_NAME1=RIGHT_SHANK_ANGLE
/SIGNAL_COMPONENT1=X
/SIGNAL_OWNER2=*.c3d
/SIGNAL_TYPE2=LINK_MODEL_BASED
/SIGNAL_FOLDER2=ORIGINAL
/SIGNAL_NAME2=RIGHT_FOOT_ANGLE
/SIGNAL_COMPONENT2=X
/RESULT_OWNER=*.c3d
!/RESULT_FOLDER=PROCESSED
/RESULT_NAME=Ankle_Couple
;
```