2021

Biomechanical Assessment of the Lower Extremity Following an Unexpected Ankle Perturbation

Emilia Gladys Schempp
Illinois State University, egschempp@gmail.com

BIOMECHANICAL ASSESSMENT OF THE LOWER EXTREMITY FOLLOWING AN UNEXPECTED ANKLE PERTURBATION

EMILIA GLADYS SCHEMPP

297 Pages

A deeper understanding of the response to an unexpected ankle perturbation could help provide insight into ankle sprain injury avoidance strategies. This study compared lower extremity electromyographic and kinematic measures between unexpected ankle perturbation and normal walking conditions, including muscle amplitudes and reaction times, dominant limb joint angles and velocities. Trial conditions included a walking control condition, unexpected 30° inversion, and unexpected 18° combined inversion and plantarflexion. With randomized trials, 20 healthy volunteers walked along the custom-built walkway that unexpectedly released into the described ankle perturbations. Significant differences were found for peak and average EMG of the peroneus longus and rectus femoris, time to peak EMG and reaction time of the ipsilateral and contralateral gluteus medius, peak joint angles, peak joint velocities, and time to peak joint angles. The differences found provide evidence that points towards a systemic response with involvement at each joint of the lower extremity and muscles within the kinetic chain.

KEYWORDS: ankle, electromyography, kinematics

BIOMECHANICAL ASSESSMENT OF THE LOWER EXTREMITY FOLLOWING AN UNEXPECTED ANKLE PERTURBATION

EMILIA GLADYS SCHEMPP

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Kinesiology and Recreation

ILLINOIS STATE UNIVERSITY

2021

BIOMECHANICAL ASSESSMENT OF THE LOWER EXTREMITY FOLLOWING AN UNEXPECTED ANKLE PERTURBATION

EMILIA GLADYS SCHEMPP

COMMITTEE MEMBERS:

Adam Jagodinsky, Chair

Justin Stanek

Michael Torry

ACKNOWLEDGMENTS

I am thankful to everyone who has helped me on this journey and made this project possible. First, I would like to thank my advisor, Dr. J, for his continued support and assistance throughout this process. Thank you also to Dr. Stanek and Dr. Torry for sharing their expertise with advice and contributions. I appreciate the research team for their dedication to the project, especially Erin, Jenny, and Zach, as well as all those that assisted in the early stages of the research. I am grateful to each one of my family members and friends who have offered encouragement personally and academically. Lastly, I am forever indebted to Blake for his persistence in challenging me to strive for excellence.

E.G.S.

CONTENTS

TABLES

FIGURES

CHAPTER I: EMG ASSESSMENT

Introduction

It has been reported that in sports the most frequently injured joint is the ankle.(Fong et al., 2007) Within the ankle injuries reported the majority, 80%, were ligamentous sprains.(Fong et al., 2009) A further problem occurs when the ankle becomes unstable due to the damage of these lateral ligaments, this is called chronic ankle instability and is reported in 20% of patients following acute ligament rupture.(Ferran et al., 2009) Ankle instability becomes increasingly common if an individual suffers numerous of these injuries.(Hertel, 2002) As with any injury the mechanism and natural response of the body must be understood to develop methods for injury prevention. Further understanding of the neuromuscular response by investigating EMG activity and the response within the kinetic chain could help develop rehabilitation programs after injury as well as improve injury prevention techniques. The fact that these ankle injuries still occur daily(Chu et al., 2010; Hall et al., 2016), even with some prevention programs in place establishes the need for further investigation of the mechanism and response.

With the prevalence of ankle injuries, specifically sprains, it is important to understand how they occur. One type of sprain is a lateral or supination ankle sprain which occurs with plantarflexion and inversion at the ankle. More specifically, when the subtalar joint exceeds 40° of inversion there is damage to the lateral and subtalar ligaments and it is termed an ankle inversion injury.(Konradsen et al., 1997) The mechanism of injury for a lateral ankle sprain, involves abnormal supination motions. Typically, during an unexpected perturbation at the ankle, the local muscles react by contracting to slow down or oppose the motion to protect the body from injury. However, if the stabilization response from the ligaments is not fast enough, injury can occur. Investigating this natural response to an unexpected ankle perturbation is one way to

study the mechanism of ankle injuries since the recording of the true mechanism is unethical and the literature lacks a substantial amount of actual injury cases with biomechanical analyses. Thus, if a model can be developed that demonstrates similar mechanics, the neuromuscular response can be studied more in depth within a laboratory setting to again provide information that may be beneficial when developing injury prevention techniques.

To support the mechanism of injury there have only been few biomechanically recorded accounts of actual ankle sprain injuries. These available biomechanical case studies of ankle sprain injury cases did not include the investigation of muscle activity.(Fong et al., 2009; E. Kristianslund et al., 2011; Mok et al., 2011) However, one case study did report the inhibition of the tibialis anterior and peroneus longus during the landing phase before the injury occurred, but similar activation to non-injury trials following this window of low activity.(Gehring et al., 2013).

Other studies have investigated this response and originally research of simulated ankle sprain motion involved the use of unexpected inversion perturbations from a static standing position.(Konradsen et al., 1997; Liu et al., 2012; Ty Hopkins et al., 2007) However, it was found that the muscular restraints of the ankle were not fast enough to counteract the motion and prevent ligament damage. During another standing simulation ankle muscles were inhibited, and proximal muscles were used to unload the inverted ankle, which served as an unloading reaction (Santos et al., 2008), indicating that more than just ankle musculature is involved in the perturbation response. Thus, with proximal muscles acting to unload the ankle the response becomes systemic. More recent research has allowed ankle sprain motion to be studied during walking gait.(Stanek et al., 2011; Ty Hopkins et al., 2007) These investigations demonstrated that in preparation for heel strike there is pre-activity of muscles in the lower extremity,

specifically the tibialis anterior.(McLoda et al., 2004) This showed that in preparation for foot contact the system itself anticipates what is to come, and could help provide adequate response time for unexpected perturbations. Additionally, this reinforced the importance of studying a dynamic activity as opposed to static simulations, especially since most athletic events involve dynamic motion.

Although the previous research identified the necessity for dynamic motion, it did not incorporate the perturbation mechanism. One investigation studied ankle mechanics with a trap door that dropped into 25 degrees of inversion during walking and jumping.(Nieuwenhuijzen et al., 2002) Here two reflex responses were seen for various lower extremity muscles including the tibialis anterior, peroneus longus, and soleus; one response with a latency of around 40ms and a second with a latency of around 100ms, but the magnitudes of the responses were not reported. More recent investigations of simulated ankle sprains with walking gait have implemented the unexpected perturbation within a walkway by a trapdoor mechanism that releases into 30° of inversion when switched on.(Stanek et al., 2011; Ty Hopkins et al., 2007) In this mechanism the rate of ankle inversion was calculated to be 261 deg/s.(Ty Hopkins et al., 2007) Here the response to the sudden inversion perturbation was first seen with activation of the peroneal muscles to counteract the inversion moment.(Konradsen et al., 1997; Ty Hopkins et al., 2007) Following this the muscles of the thigh were recruited, and the contralateral limb was brought to the plane of the center of gravity to lessen the load on the inverted foot.(Ty Hopkins et al., 2007) The attempt to avert injury occurred by removing the weight from the inverted limb through ipsilateral hip and knee flexion. This research showed greater activation of the thigh and trunk muscles, specifically the rectus femoris, gluteus maximus, and contralateral oblique, when comparing the perturbed gait to a normal walking condition. Thus, demonstrating that muscles

further up the kinetic chain and other postural muscles are involved in the response to the unexpected ankle inversion perturbation.(Stanek et al., 2011)

The purpose of this experiment was to compare the neuromuscular response following normal walking, an unexpected inversion perturbation during walking, and a combined unexpected inversion and plantarflexion perturbation during walking. With recorded lower extremity muscle activity, it was expected that there would be increased muscle activation following muscle onset with significantly greater peak EMG values for the perturbation trials as compared to the normal walking condition. Specifically, greater peak EMG values were expected for the muscles of the hip and trunk as was previously found.(Stanek et al., 2011) Additionally, it was predicted that there would be shorter time to peak and reaction time for the studied muscles during the perturbations as compared to control.

Methods

Variables

The independent variable is represented by the trial condition; unperturbed walking, inversion drop, or plantarflexion drop. Electromyographic dependent variables include peak electromyography (EMG), time to peak EMG, average EMG, and muscle reaction time with the following lower extremity muscles: peroneus longus, tibialis anterior, soleus, biceps femoris, rectus femoris, ipsilateral gluteus medius, contralateral gluteus medius, and ipsilateral gluteus maximus.

Apparatus

The apparatus used for this experiment is that which was used in previous research.(Stanek et al., 2011; Ty Hopkins et al., 2007) The custom walkway is 6.10 m long, 0.25 m tall, 0.76 m wide. There are five 1.22 m segments that are secured together by clips on the

platform sides that can be unlatched to rearrange the segments. One of the segments includes a panel with levers to control a different segment that includes the trap door which drops into 30° of inversion on the right or left side. Once the lever is put in the 'on' position, full support under the trap door is removed and the door rests on spring-ball plungers until force of ≥ 4.4 N is applied. When this occurs, the trapdoor falls 30° into inversion about the center hinge of the walkway. A similar lever system is used to control a segment with a trap door that drops into 18° of combined plantar flexion and inversion. There are electromagnetic switches on each trapdoor that provide a sine wave electric signal to record the time of 'release'. The walking surface of the runway has two nonslip strips to prevent slipping when the trapdoor falls. Handrails span the perimeter of the runway to provide support if necessary. See Appendix D Fig. 23 for a visual of the apparatus.

Participants

For this study 20 healthy volunteers (10 female and 10 male, age 24 ± 6 years, height 1.73 ± .10 m, mass 73.58 ± 14.64 kg) that engage in at least thirty minutes of physical activity three days a week were recruited. Participants did not have any lower extremity injury at the time of collection or in the last six months and did not have any previous lower extremity surgery. Subjects were asked to complete the pre-participation questionnaire which includes basic demographic information that can be found in Appendix C, as well as review and sign the informed consent form.

Data Collection

After an overview of the data collection process, surface EMG electrodes were placed on the subject to collect muscle data using the BIOPAC System (1000 Hz). These were placed on the subject's dominant leg, defined as the leg that would be used to kick a ball, for the following

muscles: peroneus longus, tibialis anterior, soleus, biceps femoris, rectus femoris, ipsilateral gluteus medius, contralateral gluteus medius, and ipsilateral gluteus maximus. For appropriate contact, the skin was shaved, abraded with 220-grit sandpaper, and cleaned with 70% alcohol at each electrode placement. The electrodes were placed two centimeters apart measured from the center of the electrode and in line with the muscle fibers. Placements included for the peroneus longus 3 cm distal to the head of the fibula, tibialis anterior approximately 1 cm lateral to the anterior border of the tibia and 8 cm distal to the tibial tuberosity, soleus two thirds the distance between the medial condyle of the femur and the medial malleolus, Biceps femoris halfway between the ischial tuberosity and the medial epicondyle, rectus femoris halfway between the anterosuperior iliac spine and the superior border of the patella, gluteus medius 3 cm below the midpoint of the iliac crest, and gluteus maximus halfway between the sacral vertebrae and the greater trochanter. See Appendix D Fig. 24 for sample placements on a subject. Then maximal voluntary isometric contractions (MVIC) of each muscle were recorded and saved to scale the EMG trial data. Three contractions lasting three seconds each were recorded. The maximum value from all three bursts were averaged, stored, and used as the scaling value. MVIC procedures were followed for each muscle: peroneus longus resisting motions of eversion and plantarflexion applied through the lateral and sole of the foot with support just above the ankle, tibialis anterior resting motions of dorsiflexion and inversion applied through the medial and dorsal side of the foot with support just above the ankle, soleus resisting motions of plantarflexion applied through the metatarsal heads and calcaneus with the knee flexed at 90° and the subject lying prone, Biceps femoris resisting knee extension applied just above the ankle with support on the hamstring and the subject lying prone, rectus femoris resisting hip extension with applied below the hip and just above the knee with the subject sitting at the edge of the

table, gluteus medius resisting hip abduction applied through the distal lateral thigh and support just above the knee and on the hip with the subject lying on their side with the test side up, gluteus maximus resisting hip flexion applied through the thigh with support on the lower back and the subject lying prone.

Following this the subject put on dribbling goggles to prevent viewing of the inferior field of vision and were instructed to start at the beginning of the walkway and walk the length of the walkway to the cadence of a metronome set to 90 beats per minute while looking straight ahead. The exact starting location was adjusted to account for the stride length to ensure the subject consistently contacts the trapdoor locations with their dominant leg.

The subject walked the length of the runway sixty-three times and EMG data was recorded for nine of these trials. Three trials were with the inversion perturbation, three were with the inversion and plantarflexion perturbation, and three were recorded without a perturbation as a normal walking condition. The remaining trials were used to normalize gait designated as wash trials, six wash trials were used in between each of the perturbation and normal walking trials as a preliminary analysis of stride length and stride with indicated that gait normalized after six wash trials following perturbation. For each subject, the perturbation and normal walking trials were randomized using a random number generator, therefore after each set of wash trials a subject randomly experienced any one of the three conditions: inversion perturbation, combined inversion/plantarflexion perturbation, or normal walking.

Data Processing and Analysis

Processing included trimming the MVIC trials in Vicon. The unfiltered data was then be exported from Vicon and saved. With Visual 3D data was filtered using a low pass filter with a cutoff frequency of 300Hz. Average peak MVIC values were computed for each muscle as the

average of the maximum values from each contraction from the rectified MVIC recordings and were used for further trial EMG normalization. Specific event labels were created including gait events, muscle onset indicators, pre and post drop events. Data was exported from Visual 3D in ASCII format, saved as a text file as well as downloaded in excel and saved as an excel file. Then files were read into Matlab and computations for peak EMG, average EMG, and time to peak EMG were completed.

Muscle onset was calculated between the heel strike event and the end of the trial and recorded when the signal reached two standard deviations above the mean of the signal 150ms prior to heel strike. Reaction time was calculated as the difference between onset time and the time of the corresponding heel strike event. Peak and average EMG was calculated between muscle onset and the 350ms following. Time to peak EMG was computed as the time between peak EMG and muscle onset. See Fig. 25 in Appendix D for a sample EMG signal with labeled events and windows of analysis.

See Appendix A for Visual 3D pipelines and Appendix B for Matlab Codes.

Statistical Analysis

A statistical test of a one-way repeated measures MANOVA in SPSS was used to quantitatively compare the normal walking, inversion perturbation, and inversion and plantarflexion conditions for the EMG variables of interest. This was used to detect differences among conditions and post hoc tests were used with the Bonferroni adjustment to determine specific group differences when significant differences were found in the MANOVA. The alpha level was $p \leq 0.05$. The first set of MANOVAs included peak and average EMG and was ran separately for each muscle. Temporal variables of time to peak EMG and muscle reaction time were ran together for each muscle in their own MANOVA.

Results

Peak EMG

No significant differences in peak EMG (%MVC) were found across conditions for the following muscles: Tibialis Anterior; Soleus; Biceps Femoris; Ipsilateral Gluteus Medius; Contralateral Gluteus Medius; Gluteus Maximus.

Significant differences in peak EMG (%MVC) were found across conditions for the following muscles: Peroneus Longus; Rectus Femoris. A repeated measures ANOVA determined that the variables of peak EMG and average EMG differed significantly between trial conditions ($F(4,74) = 2.609$, $p = .042$) for the peroneus longus. The within subjects effect with sphericity assumed for peak EMG was also significant ($F(2) = 3.927$, $p = .028$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .038$) reduction in peak EMG for the peroneus longus for the combined inversion/plantarflexion trial condition (15.98 ±13.41) as compared to normal walking (28.36±20.96). A second repeated measures ANOVA determined that the variables of peak EMG and average EMG differed significantly between trial conditions ($F(4,54) = 2.718$, $p = .039$) for the rectus femoris. The within subjects effect with sphericity assumed for average EMG was also significant ($F(2) = 5.005$, $p = .014$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .006$) reduction in peak EMG for the rectus femoris for the inversion trial condition (5.04 ±10.35) as compared to normal walking (9.17±11.00).

Table 1

Peak EMG Descriptive Statistics. Values reported as %MVIC. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

	INV	**INV/PF**	**NW**
Muscle	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Tibialis Anterior	17.62 ± 16.13	18.86 ± 13.92	18.39 ± 8.65
Soleus	35.47 ± 29.63	25.15 ± 30.86	39.52 ± 48.66
Biceps Femoris	79.34 ± 184.49	61.69 ± 137.12	62.28 ± 131.12
Ipsilateral Gluteus Medius	25.71 ± 49.06	15.83 ± 36.95	28.03 ± 60.42
Contralateral Gluteus Medius	10.39 ± 8.87	11.94 ± 11.61	11.16 ± 13.37
Gluteus Maximus	5.77 ± 8.34	25.91 ± 78.21	6.85 ± 4.43
Peroneus Longus	26.14 ± 19.97	15.98 ± 13.41	28.36 ± 20.96
Rectus Femoris	5.04 ± 10.35	8.11 ± 7.98	9.17 ± 11.00

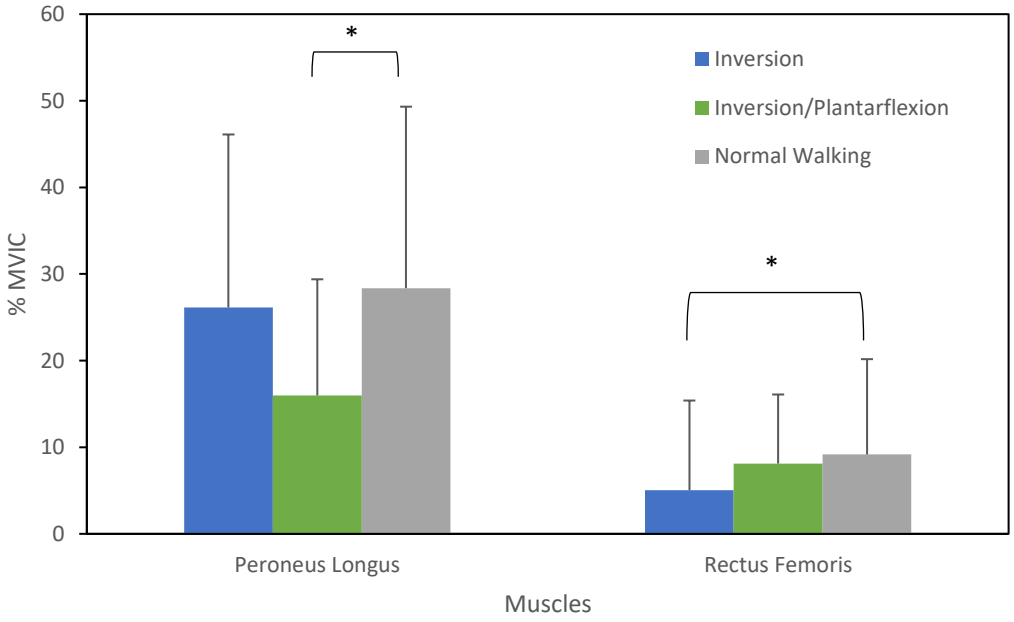

Figure 1. Descriptive statistics of normalized EMG peak amplitudes. * = Significant differences, $p < .05$

Average EMG

No significant differences in average EMG (%MVC) were found across conditions for the following muscles: Tibialis Anterior; Soleus; Biceps Femoris; Ipsilateral Gluteus Medius; Contralateral Gluteus Medius; Gluteus Maximus.

Significant differences in average EMG (%MVC) were found across conditions for the following muscles: Peroneus Longus; Rectus Femoris. A repeated measures ANOVA determined that the variables of peak EMG and average EMG differed significantly between trial conditions ($F(4,74) = 2.609$, $p = .042$) for the peroneus longus. The within subjects effect with a Greenhouse-Geisser correction for average EMG was also significant ($F(1.516) = 5.020$, $p = .020$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .022$) reduction in average EMG for the peroneus longus for the combined inversion/plantarflexion trial condition (9.62 ±9.14) as compared to normal walking (17.23±11.64). A second repeated measures ANOVA determined that the variables of peak EMG and average EMG differed significantly between trial conditions ($F(4,54) = 2.718$, $p = .039$) for the rectus femoris. The within subjects effect with a Greenhouse-Geisser correction for average EMG was not significant ($p = .062$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .019$) reduction in average EMG for the rectus femoris for the inversion trial condition (3.44 ±7.73) as compared to normal walking (5.82±6.91).

Table 2

Average EMG Descriptive Statistics. Values reported as %MVIC. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

	INV	INV/PF	NW
Muscle	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Tibialis Anterior	11.60 ± 11.54	12.12 ± 8.70	11.98 ± 5.68
Soleus	21.39 ± 17.92	14.68 ± 20.73	24.00 ± 27.79
Biceps Femoris	54.74 ± 131.86	38.04 ± 88.12	44.02 ± 95.51
Ipsilateral Gluteus Medius	15.81 ± 32.47	10.90 ± 27.94	18.54 ± 40.76
Contralateral Gluteus Medius	6.17 ± 4.71	7.91 ± 8.36	6.76 ± 8.34
Gluteus Maximus	3.93 ± 5.29	13.43 ± 38.38	4.35 ± 2.95
Peroneus Longus	14.72 ± 9.86	9.62 ± 9.14	17.23 ± 11.64
Rectus Femoris	3.44 ± 7.73	5.02 ± 5.19	5.82 ± 6.91

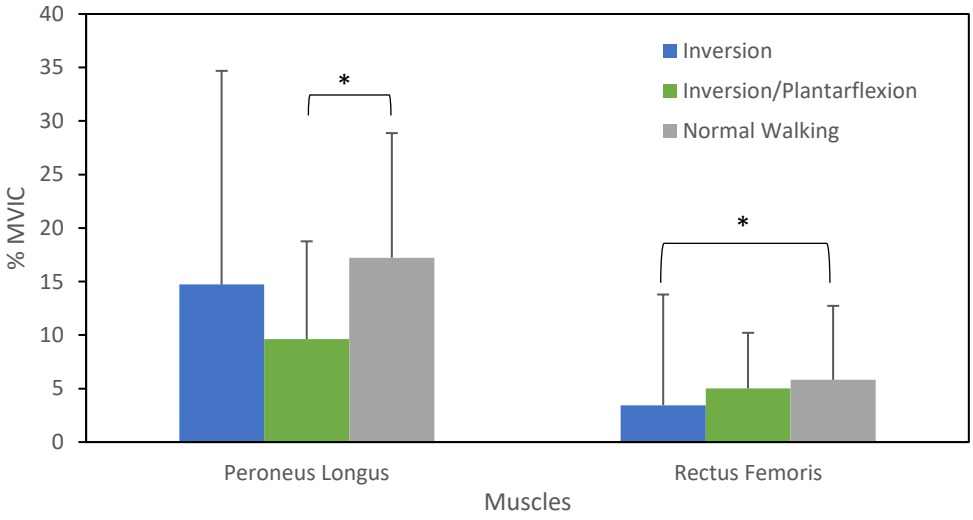

Figure 2. Descriptive statistics of normalized EMG average amplitudes. * = Significant differences, $p < .05$

Time to Peak EMG

No significant differences in time to peak EMG (seconds) were found across conditions for six of the eight muscles analyzed, including: Peroneus Longus; Tibialis Anterior; Soleus; Biceps Femoris; Rectus Femoris; Gluteus Maximus.

Significant differences in time to peak EMG (seconds) were found across conditions for two of the eight muscles analyzed, including: Ipsilateral Gluteus Medius; Contralateral Gluteus

Medius. A repeated measures ANOVA determined that the temporal variables of time to peak EMG and reaction time differed significantly between trial conditions ($F(4,74) = 3.632$, $p = .009$) for the ipsilateral gluteus medius. The within subjects effect for time to peak EMG with a Greenhouse-Geisser correction was also significant ($F(1.629) = 5.454$, $p = .013$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .003$) reduction in time to peak EMG for the ipsilateral gluteus medius for the inversion trial condition (.13 ±.07) as compared to normal walking (.22±.06). A second repeated measures ANOVA determined that the temporal variables of time to peak EMG and reaction time differed significantly between trial conditions ($F(4,74) = 18.875$, $p < .001$) for the contralateral gluteus medius. The within subjects effect with Greenhouse-Geisser correction for time to peak EMG was not significant ($p = .103$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .002$) reduction in time to peak EMG for the contralateral gluteus medius for the combined inversion/plantarflexion trial condition (.02 ±.20) as compared to inversion (.17±.12).

Table 3

Time to peak EMG Descriptive Statistics. Values reported in seconds. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

	INV	**INV/PF**	**NW**
Muscle	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Peroneus Longus	.15 ± .06	.15 ± .10	.20 ± .09
Tibialis Anterior	.13 ± .08	.17 ± .12	.16 ± .10
Soleus	.13 ± .07	.17 ± .10	.18 ± .14
Biceps Femoris	.17 ± .06	.17 ± .12	.22 ± .10
Rectus Femoris	.14 ± .13	.07 ± .29	.15 ± .13
Gluteus Maximus	.18 ± .08	.04 ± .34	.06 ± .24
Ipsilateral Gluteus Medius	.13 ± .07	.13 ± .14	.22 ± .06
Contralateral Gluteus Medius	.17 ± .12	.02 ± .20	.04 ± .38

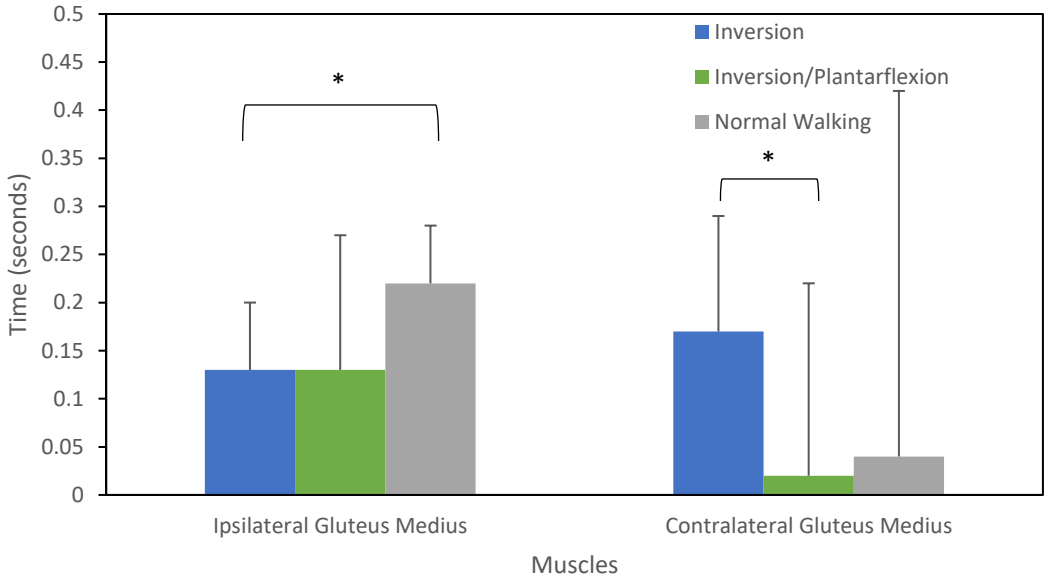

Figure 3. Descriptive statistics of time to peak EMG. * = Significant differences, $p < .05$

Reaction Time

No significant differences in EMG reaction time (seconds) were found across conditions for the following muscles: Peroneus Longus; Tibialis Anterior; Soleus; Biceps Femoris; Rectus Femoris; Gluteus Maximus.

Significant differences in EMG reaction time (seconds) were found across conditions for the following muscles: Ipsilateral Gluteus Medius; Contralateral Gluteus Medius. A repeated measures ANOVA determined that the temporal variables of time to peak EMG and reaction time differed significantly between trial conditions ($F(4,74) = 3.632$, $p = .009$) for the ipsilateral gluteus medius. The within subjects effect for reaction time with a Greenhouse-Geisser correction was not significant ($p = .080$). Post hoc tests using the Bonferroni correction revealed that there were no significant differences in reaction time for the ipsilateral gluteus medius between trial conditions A second repeated measures ANOVA determined that that the temporal variables of time to peak EMG and reaction time differed significantly between trial conditions ($F(4,74) = 18.875$, $p < .001$) for the contralateral gluteus medius. The within-subjects effect for

reaction time with Greenhouse-Geisser correction was significant ($F(1.616) = 50.655$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p < .001$) reduction in reaction time of the contralateral gluteus medius for both the inversion trial condition (.24±.10) and combined inversion/plantarflexion (.30 ±.13) as compared to normal walking (.64±.19).

Table 4

EMG Reaction Time Descriptive Statistics. Values reported in seconds. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

	INV	**INV/PF**	**NW**
Muscle	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Peroneus Longus	.08 ± .05	.06 ± .04	.12 ± .19
Tibialis Anterior	.07 ± .04	.09 ± .07	.16 ± .18
Soleus	.06 ± .03	.07 ± .05	.06 ± .02
Biceps Femoris	.08 ± .05	.08 ± .04	.12 ± .15
Rectus Femoris	.06 ± .03	.05 ± .02	.08 ± .12
Gluteus Maximus	.04 ± .03	.09 ± .11	.08 ± .08
Ipsilateral Gluteus Medius	.06 ± .03	.06 ± .02	.10 ± .12
Contralateral Gluteus Medius	.24 ± .10	.30 ± .13	.64 ± .19

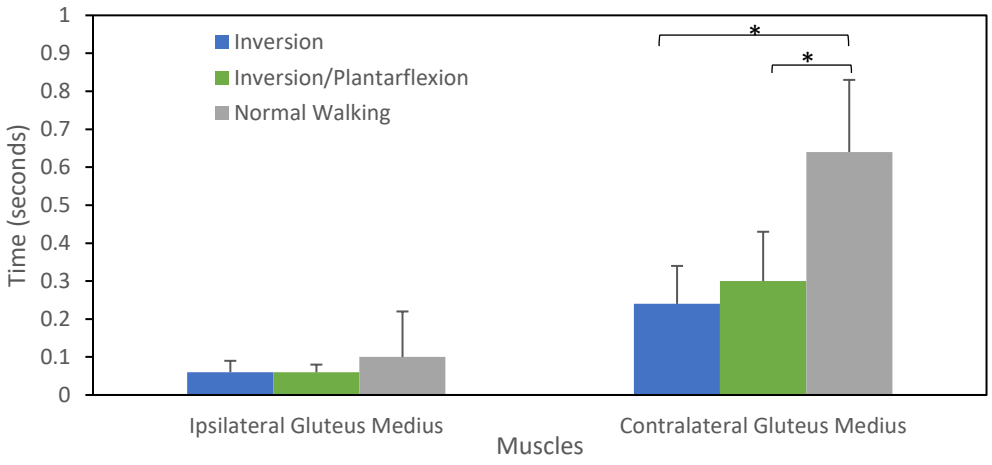

Figure 4. Descriptive statistics of muscle reaction times. * = Significant differences, $p < .05$

Discussion

The results of this study, specifically peak and average EMG revealed significantly lower activity for the rectus femoris during the inversion trial condition as compared to the control of normal walking. This finding may indicate that the rectus femoris tends to be more active during the control of normal walking than following the inversion perturbation, although the opposite finding occurred in previous research.(Stanek et al., 2011) In the present study the calculation of peak and average EMG included normalization to the MVIC for adequate comparison across subjects. This scaling of the trial muscle data may help explain why these results are not supported by previous the previous study, where similar scaling was not used; instead, data values were reported and compared in millivolts. Additionally, in the current study the window of analysis was expanded to capture more of the response with a 350ms window following muscle onset instead of the 200ms window previously used (Stanek et al., 2011). With an expanded window analysis, providing more signal analyzed and used in the average calculations it also allows for further variation to be included and could lower the chance for statistical findings in the same direction.

Another finding showed significantly lower peak and average EMG for the peroneus longus during the combined inversion/plantarflexion condition as compared to normal walking. Activation of the peroneus longus was expected during the normal walking condition as supported in previous findings(Ty Hopkins et al., 2007), it was also predicted to follow both perturbations to evert the foot and counteract the inversion mechanism of the drop. However, the magnitude of the peroneus longus for the inversion condition and normal walking were similar indicating that the peroneus longus is still active just not significantly more active. Less activity for the inversion/plantarflexion condition as compared to normal walking indicates that the

peroneus longus is not as active as it is during normal walking perhaps because the mechanism itself puts the foot into plantarflexion as opposed to the muscle activating to make this change.

One trend that is present in the descriptive statistics of peak and average EMG, Tables 1 and 2, is that of the gluteus maximus. Although there were no significant findings with this muscle's activity the mean magnitude and standard deviations were both much larger for the combined inversion/plantarflexion condition than the inversion alone or the control of normal walking. This could show a trend that some subjects could have had more variable activity in their gluteus maximus during following this combined inversion/plantarflexion perturbation. Alternatively, the soleus shows a trend somewhat of the contrary for the combined inversion/plantarflexion condition. Again, observing the descriptive statistics in Table 1 and 2, the soleus has a smaller mean magnitude for the inversion/plantarflexion condition as compared to similar mean values for the inversion and normal walking conditions. Together these trends could lend to an observation that the mechanics of the inversion/plantarflexion condition could potentially be different in terms of the gluteus maximus and soleus muscle activity. Thus, utilizing more sensitive metrics by differentiating phases could be beneficial in observing and reporting the mechanism and the response as some studies employed (Gehring et al., 2013; Eirik Kristianslund et al., 2011), rather than the one undivided 350ms window of analysis used in this study.

Time to peak findings included a significantly shorter time to peak for the ipsilateral gluteus medius during the inversion condition as compared to the control normal walking condition. As the gluteus medius is an important stabilizer and helps with single leg support this could demonstrate a more rapid response of the neuromuscular system to the inversion condition activating as a stabilization response. Similar findings occurred previously with a faster time to

peak during the inversion perturbation for the contralateral and external obliques during the previous study.(Stanek et al., 2011) For the combined inversion/plantarflexion condition significantly shorter time to peak values were found for the contralateral gluteus medius compared to the inversion condition. Perhaps due to a more involved neuromuscular response to the inversion condition the contralateral gluteus medius time to peak is delayed.

One trend in the time to peak descriptive statistics can be seen in Table 3 in the row of the gluteus maximus. Although not significant, the mean of the gluteus maximus is larger for the inversion condition than the similar smaller means of the inversion/plantarflexion condition and normal walking. The gluteus maximus also has a smaller standard deviation for the inversion condition, perhaps demonstrating that for the inversion condition the gluteus maximus responded in a more similar way among subjects than for that of the other conditions. Additionally, the mean values found (INV .18 ± .08, PF .04 ± .34, NW .06 ± .24) for time to peak EMG (seconds) for the gluteus maximus were similar to previous findings (INV .15 ± .04, NW .10 ± .06).(Stanek et al., 2011) Another trend present in Table 3 can be seen with the rectus femoris data. Here for the rectus femoris, the mean is smaller for the inversion/plantarflexion condition compared to the similar larger values for the other two conditions. This could show that the rectus femoris often reached peak activity sooner for the inversion/plantarflexion condition, although since this difference is not significant it cannot be certain. Additionally, the mean values found (INV .14 ± .13, PF .07 ± .29, NW .15 ± .13) for time to peak EMG (seconds) for the rectus femoris were similar to previous findings (INV .15 ± .04, NW .15 ± .05).(Stanek et al., 2011) However, lack of significant findings for these muscles with the time to peak variable also occurred previously.

Reaction time results included a significantly shorter reaction time for the contralateral gluteus medius in the inversion condition over the control of normal walking. Additionally, the

contralateral gluteus medius had a significantly shorter reaction time during the combined inversion/plantarflexion condition over the control of normal walking. This could indicate that the contralateral limb is involved in the response as predicted. Which could show that for these perturbation conditions the contralateral limb is acting faster to help stabilize the system potentially for an unloading reaction. With more significant involvement of this proximal musculature during the perturbation conditions it could suggest that the response travels through the kinetic chain eliciting a response that is more systemic in nature than localized.

In looking at Table 4 a few more trends can be observed which were not revealed significantly. In the tibialis anterior row, the mean and standard deviation both increased for the normal walking condition as compared to either perturbation. This indicates that the time to peak activity of the tibialis anterior was not as consistent among subjects for the normal walking condition as was seen during the perturbation trials. A final trend can be observed in the peroneus longus row of Table 4. Similar to the tibialis anterior the peroneus longus showed both a larger mean and standard deviation during the normal walking trials than was present for either perturbation condition. Thus, perhaps indicating that there was more variability and less consistency among subjects in the time to peak activity of the peroneus longus during normal walking than that of the perturbation conditions. The mean values found for reaction time reported in seconds of the peroneus longus for the perturbation conditions (INV .08 ± .05, PF .06 ± .04) were similar to previous findings: (10° INV .051)(Konradsen et al., 1997), (20° INV .057 ± .01)(Vaes et al., 2002), (15° INV .097)(Fernandes et al., 2000).

With large standard deviations of the EMG signals along with large standard deviations present in kinematic data reported in Chapter 2, it indicates high variability in the signals. This could suggest that the neuromuscular strategy is not set for each perturbation among subjects.

Even so, the EMG findings of this study show that different perturbations during dynamic activity can elicit different neuromuscular responses and proximal musculature may play a role in stabilization of the body as was shown in the temporal findings.

Limitations of this study include potential electrode sliding during the dynamic activity. Additionally, since subjects were aware of the possibility of perturbation at any time during any trial their muscle activity may be different in anticipation of the perturbation. With the undivided window of analysis used, comparison to other research that detailed phased responses is limited, thus more sensitive metrics may be beneficial for studying phases of the perturbation itself and the response.

In closing, the significant results of the study for peak and average EMG provide new normalized reference values for the muscle activity for two perturbations with a larger window than was previously reported. The temporal EMG findings with significant ipsilateral and contralateral gluteus medius involvement during the perturbation conditions support the conclusion that proximal musculature may be involved in a stabilization response in an effort to counteract the unexpected perturbation.

CHAPTER II: KINEMATIC ASSESSMENT

Introduction

It has been reported that in sports the most frequently injured joint is the ankle.(Fong et al., 2007) Within the ankle injuries reported the majority, 80%, were ligamentous sprains.(Fong et al., 2009). A further problem occurs when the ankle becomes unstable due to the damage of these lateral ligaments, this is called chronic ankle instability and is reported in 20% of patients following acute ligament rupture.(Ferran et al., 2009) Ankle instability becomes increasingly common if an individual suffers numerous of these injuries.(Hertel, 2002) As with any injury the mechanism and natural response of the body must be understood to develop methods for injury prevention. Further understanding of the movement response by investigating lower extremity kinematics and the response at each joint in the kinetic chain could help develop rehabilitation programs after injury as well as improve injury prevention techniques. The fact that these ankle injuries still occur daily(Chu et al., 2010; Hall et al., 2016), even with some prevention programs in place establishes the need for further investigation of the mechanism and response.

With the prevalence of ankle injuries, specifically sprains, it is important to understand how they occur. One type of sprain is a lateral or supination ankle sprain which occurs with plantarflexion and inversion at the ankle. More specifically, when the subtalar joint exceeds 40° of inversion there is damage to the ligaments and it is termed an ankle inversion injury.(Konradsen et al., 1997) Additionally, one study has suggested a safe inversion rate threshold of 300 deg/s from the analysis of common sporting motions and sprain motions, where they suggest that anything beyond that threshold may cause damage.(Chu et al., 2010)

The mechanism of injury for a lateral ankle sprain, as detailed above, involves abnormal supination motions. Externally, ankle sprains can occur due to an unexpected perturbation at the

ankle that externally rotates or forces the ankle into this motion. Typically, during an unexpected perturbation at the ankle, the muscles react by contracting to slow down or oppose the motion to protect the body from injury. However, if the stabilization response from the ligaments is not fast enough, injury can occur. Investigating this natural response to an unexpected ankle perturbation is one way to study the mechanism of ankle injuries since the recording of the true mechanism is unethical and the literature lacks a substantial amount of actual injury cases with biomechanical analyses. Thus, if a model can be developed that demonstrates similar mechanics, the kinematic response can be studied more in depth within a laboratory setting to again provide information that may be beneficial when developing injury prevention techniques.

To support the mechanism of injury there have only been few biomechanically recorded accounts of actual ankle sprain injuries. These available biomechanical case studies of ankle sprain injury cases support an inversion perturbation as they lack plantarflexion during the injury mechanism.(Fong et al., 2009; E. Kristianslund et al., 2011; Mok et al., 2011) There was one case where both inversion and plantarflexion were reported, however this athlete had ankle instability.(Gehring et al., 2013) Support for flexion reflexes or unloading reaction described later also comes from a case study of a lateral ankle sprain in which there was an unloading of the foot and reduced ground reaction force.(E. Kristianslund et al., 2011) These case studies and previous literature described prepares for further research to provide detail of the response beyond the muscle activation with kinematic data as well as implement a similar trapdoor mechanism that includes both plantarflexion and inversion to represent and understand another potential method of a lateral ankle injury.

Many studies have investigated this response and originally research of simulated ankle sprain motion involved the use of unexpected inversion perturbations from a static standing

position.(Konradsen et al., 1997; Liu et al., 2012; Ty Hopkins et al., 2007) However, it was found that the muscular restraints of the ankle were not fast enough to counteract the motion and prevent ligament damage. During another standing simulation ankle muscles were inhibited, and proximal muscles were used to unload the inverted ankle, which served as an unloading reaction.(Santos et al., 2008), indicating that more than just ankle musculature is involved in the perturbation response. Thus, with proximal muscles acting to unload the ankle the response becomes systemic. More recent research has allowed ankle sprain motion to be studied during walking gait.(Stanek et al., 2011; Ty Hopkins et al., 2007) These investigations demonstrated that in preparation for heel strike there is pre-activity of muscles in the lower extremity, specifically the tibialis anterior.(McLoda et al., 2004) This showed that in preparation for foot contact the system itself anticipates what is to come, and could help provide adequate response time for unexpected perturbations. Additionally, this reinforced the importance of studying a dynamic activity as opposed to static simulations, especially since most athletic events involve dynamic motion.

Although the previous research identified the necessity for dynamic motion it did not incorporate the perturbation mechanism. One investigation studied ankle mechanics with a trap door that dropped into 25 degrees of inversion during walking and jumping.(Nieuwenhuijzen et al., 2002) Here the inversion velocity for the walking task was measured at 403 ± 18 deg/s and for the jumping task at 595 ± 27 deg/s. More recent investigations of simulated ankle sprains with walking gait have implemented the unexpected perturbation within a walkway by a trapdoor mechanism that releases into 30° of inversion when switched on.(Stanek et al., 2011; Ty Hopkins et al., 2007) In this mechanism the rate of ankle inversion was calculated to be 261 deg/s.(Ty Hopkins et al., 2007) Here the response to the sudden inversion perturbation was first seen with

activation of the peroneal muscles to counteract the inversion moment.(Konradsen et al., 1997; Ty Hopkins et al., 2007) Following this the muscles of the thigh were recruited, and the contralateral limb was brought to the plane of the center of gravity to lessen the load on the inverted foot.(Ty Hopkins et al., 2007) The attempt to avert injury occurred by removing the weight from the inverted limb through ipsilateral hip and knee flexion. This research showed greater activation of the thigh and trunk muscles, specifically the rectus femoris, gluteus maximus, and contralateral oblique, when comparing the perturbed gait to a normal walking condition. Thus, demonstrating that muscles further up the kinetic chain and other postural muscles are involved in the response to the unexpected ankle inversion perturbation.(Stanek et al., 2011) Another investigation with a similar walkway and trap door apparatus that dropped into 30 degrees of inversion calculated an inversion velocity of 209.1 ± 48.2 deg/s.(Hall et al., 2016)

The purpose of this experiment was to descriptively compare the kinematic response following normal walking, an unexpected inversion perturbation during walking, and a combined unexpected inversion and plantarflexion perturbation during walking. With recorded lower extremity kinematics, it was expected that the experiment would serve as a dynamic unloading reaction test for both unexpected perturbation conditions. This would include ipsilateral flexion reflexes, detailed by significantly increased in ankle, knee, and hip flexion angles as compared to normal walking trial. It is also expected that the perturbations will have greater inversion velocity values than normal walking because of the apparatus dropping into 30 degrees of inversion and 18 degrees of combined inversion and plantarflexion and will near the values of calculated previously(Hall et al., 2016; Nieuwenhuijzen et al., 2002; Ty Hopkins et al., 2007).

Methods

Variables

The independent variable for both hypotheses are represented by the trial condition; unperturbed walking, inversion drop, or plantarflexion drop. Kinematic dependent variables included peak ankle, knee, and hip angles in all planes of motion as well as time to peak angle and peak velocity for the dominant limb.

Apparatus

The apparatus used for this experiment is that which was used in previous research.(Stanek et al., 2011; Ty Hopkins et al., 2007) The custom walkway is 6.10 m long, 0.25 m tall, 0.76 m wide. There are five 1.22 m segments that are secured together by clips on the platform sides that can be unlatched to rearrange the segments. One of the segments includes a panel with levers to control a different segment that includes the trap door which drops into 30° of inversion on the right or left side. Once the lever is put in the 'on' position, full support under the trap door is removed and the door rests on spring-ball plungers until force of $\geq$ 4.4 N is applied. When this occurs, the trapdoor falls 30° into inversion about the center hinge of the walkway. A similar lever system is used to control a segment with a trap door that drops into 18° of combined plantar flexion and inversion. There are electromagnetic switches on each trapdoor that provide a sine wave electric signal to record the time of 'release'. The walking surface of the runway has two nonslip strips to prevent slipping when the trapdoor falls. Handrails span the perimeter of the runway to provide support if necessary. See Appendix D Fig. 23 for a visual of the apparatus.

Participants

For this study 20 healthy volunteers (10 female and 10 male, age 24 ± 6 years, height 1.73 ± .10 m, mass 73.58 ± 14.64 kg) that engage in at least thirty minutes of physical activity three days a week were recruited. Participants did not have any lower extremity injury at the time of collection or in the last six months and did not have had any previous lower extremity surgery. Subjects were asked to complete the pre-participation questionnaire which included basic demographic information that can be found in Appendix C, as well as review and sign the informed consent form.

Data Collection

After an overview of the data collection process, reflective markers were placed on the lower extremity segments of the subject following the 33 marker lower extremity model to track the body segment positions. The six-camera Vicon motion capture system (200Hz) was used to record the trials. Before the trials, the cameras were calibrated to reduce error and a static calibration trial was recorded. See Appendix D Fig. 26 for marker placements.

Following this the subject put on dribbling goggles to prevent viewing of the inferior field of vision and was asked to start at the beginning of the walkway and walk the length of the walkway to the cadence of a metronome set to 90 beats per minute while looking straight ahead. The exact starting location was adjusted to account for the stride length to ensure the subject consistently contacts the trapdoor locations with their dominant leg.

The subject walked the length of the runway sixty-three times and kinematic data was recorded for nine of these trials. Three trials were with the inversion perturbation, three were with the inversion and plantarflexion perturbation, and three were recorded without a perturbation as a normal walking condition. The remaining trials were used to normalize gait

designated as wash trials, six wash trials were used in between each of the perturbation and normal walking trials as a preliminary analysis of stride length and stride with indicated that gait normalized after six wash trials following perturbation. For each subject, the perturbation and normal walking trials were randomized using a random number generator, therefore after each set of wash trials a subject randomly experienced any one of the three conditions: inversion perturbation, combined inversion/plantarflexion perturbation, or normal walking.

Data Processing and Analysis

Processing included digitizing each trial including the static calibration with marker placements within Vicon. See Fig. 27 in Appendix D for a screenshot of the static calibration in Vicon. The unfiltered data was then be exported from Vicon and saved. Visual 3D was then used to compile and construct the lower extremity model by assigning the appropriate static calibration data along with the subject's height and mass. See Fig. 28 in Appendix D for a screenshot of the assigned model in Visual 3D. Within Visual 3D data was filtered using a fourth order Butterworth filter. Lower extremity kinematics were calculated in Visual 3D using model-based computations. Specific event labels were created including gait events, pre and post drop events. Data was exported from Visual 3D as a text file, saved as a text file as well as downloaded in excel and saved as an excel file. Then files were read into Matlab and peak angle, time to peak angle, and peak velocity in all three planes of motion at the ankle, knee, and hip were computed.

Visual 3D joint conventions included: Ankle x (+ dorsiflexion / - plantarflexion), y (+ inversion / - eversion), z (+ internal rotation / - external rotation); Knee x (+ extension / - flexion), y (+ adduction,/- abduction), z (+ internal rotation / - external rotation); Hip x (+ flexion / - extension), y (+ adduction / - abduction), z (+ internal rotation / - external rotation). The joint

angles reported are as exported from Visual 3D where they are defined as the orientation of one segment relative to another. For the ankle angle the segment used was the foot and the reference segment was the shank. The knee angle used the shank segment and a reference segment of the thigh. Then the hip angle used the thigh segment and a reference segment of the pelvis. Note that specifically for the ankle, an angle of zero degrees does not correspond to anatomical neutral as it is only an expression of the orientation of the corresponding segments. Joint normalization was not utilized, and the default Cardan sequence (x, y, z) was used, with y as the anterior-posterior axis and z in the axial direction.

Peak angle and peak velocity were calculated between the heel strike event and the 350ms following. Time to peak angle was calculated as the difference in time from the heel strike event to the peak angle.

See Appendix A for Visual 3D pipelines and Appendix B for Matlab Codes.

Statistical Analysis

A statistical test of a one-way repeated measures MANOVA in SPSS was used to quantitatively compare the normal walking, inversion perturbation, and inversion and plantarflexion conditions for the kinematic variables of interest. This was used to detect differences among conditions and post hoc tests were used with the Bonferroni adjustment to determine specific group differences when significant differences were found in the MANOVA. The alpha level was $p \leq 0.05$.

Results

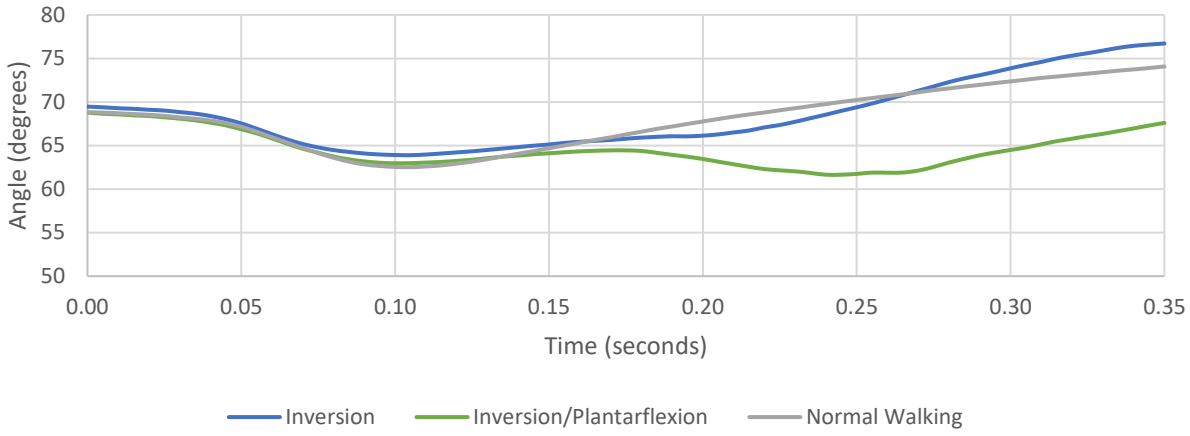

Figure 5. Time series of ankle angles in the x direction. Conventions: x (+ dorsiflexion / - plantarflexion). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

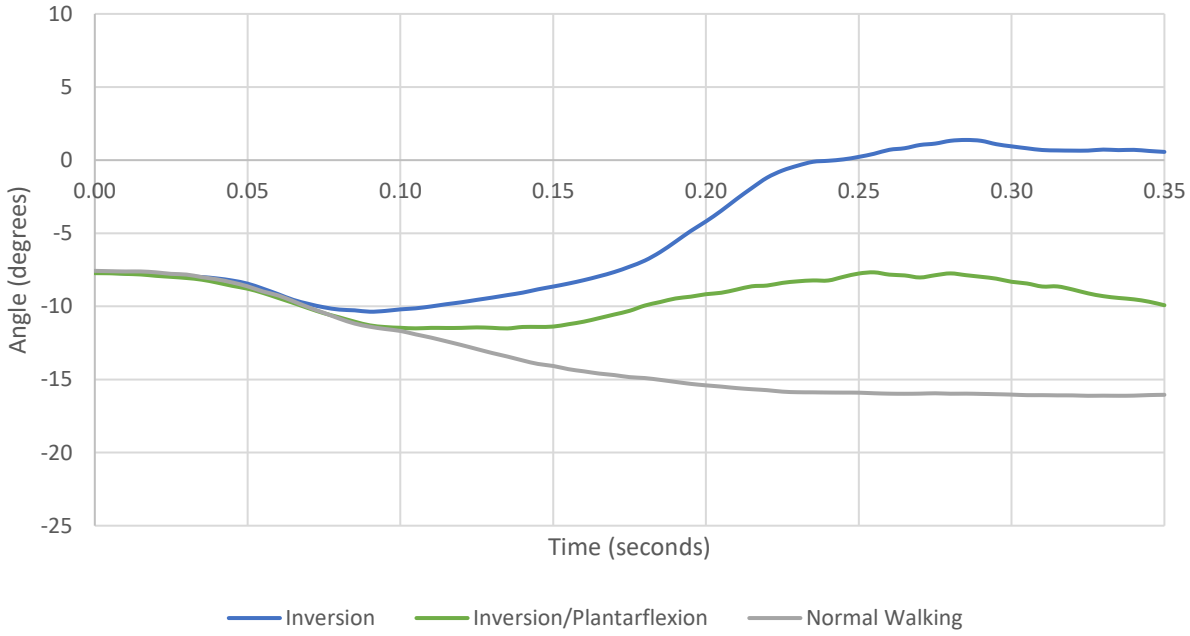

Figure 6. Time series of ankle angles in the y direction. Conventions: y (+ inversion / - eversion). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

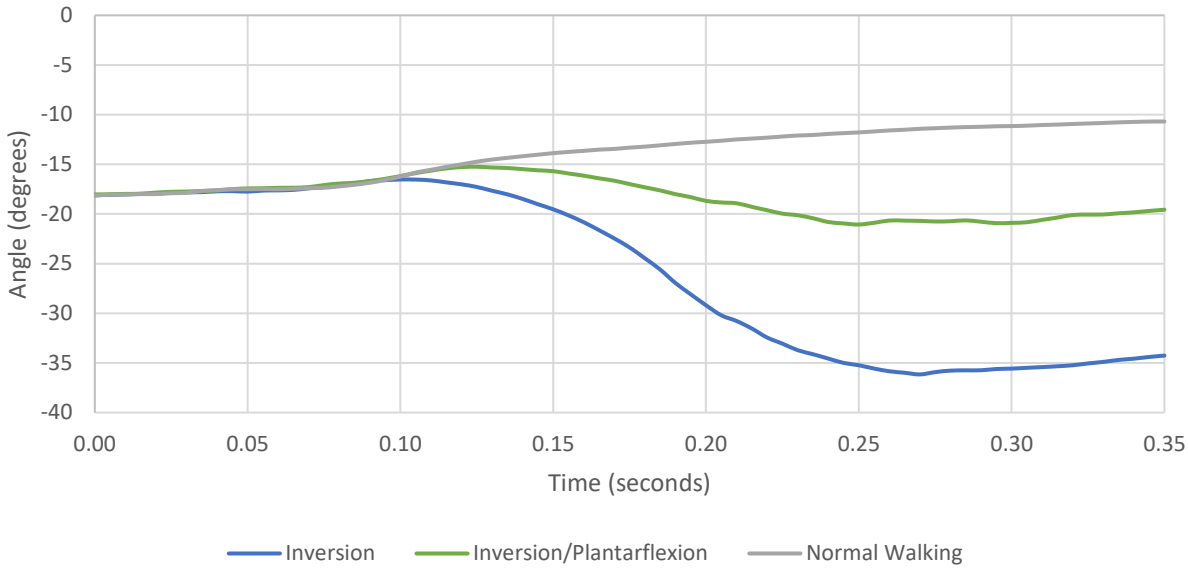

Figure 7. Time series of ankle angles in the z direction. Conventions: z (+ internal rotation / - external rotation). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

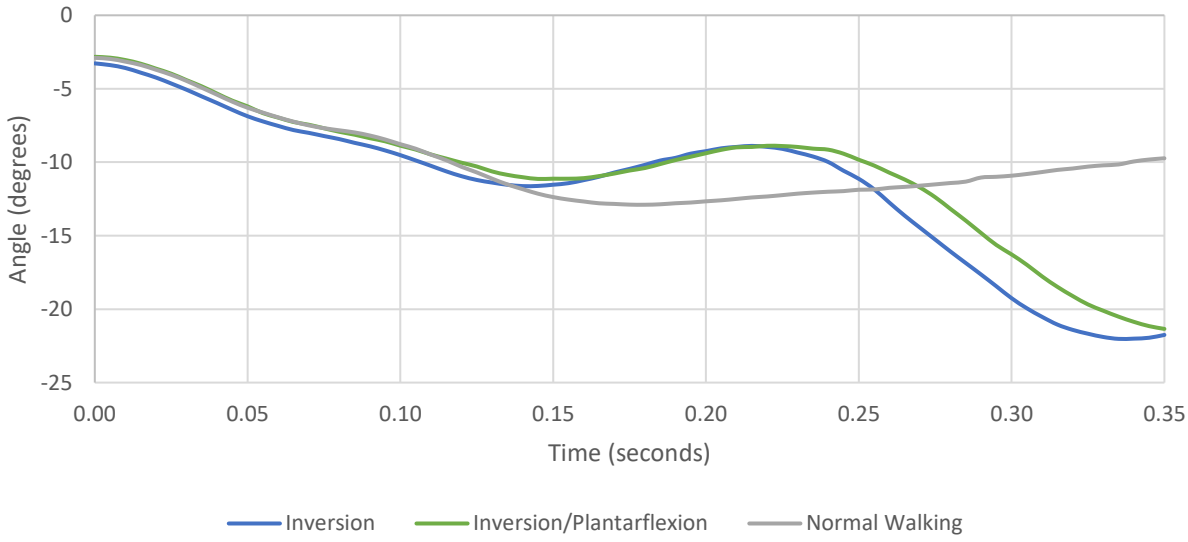

Figure 8. Time series of knee angles in the x direction. Conventions: x (+ extension / - flexion). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

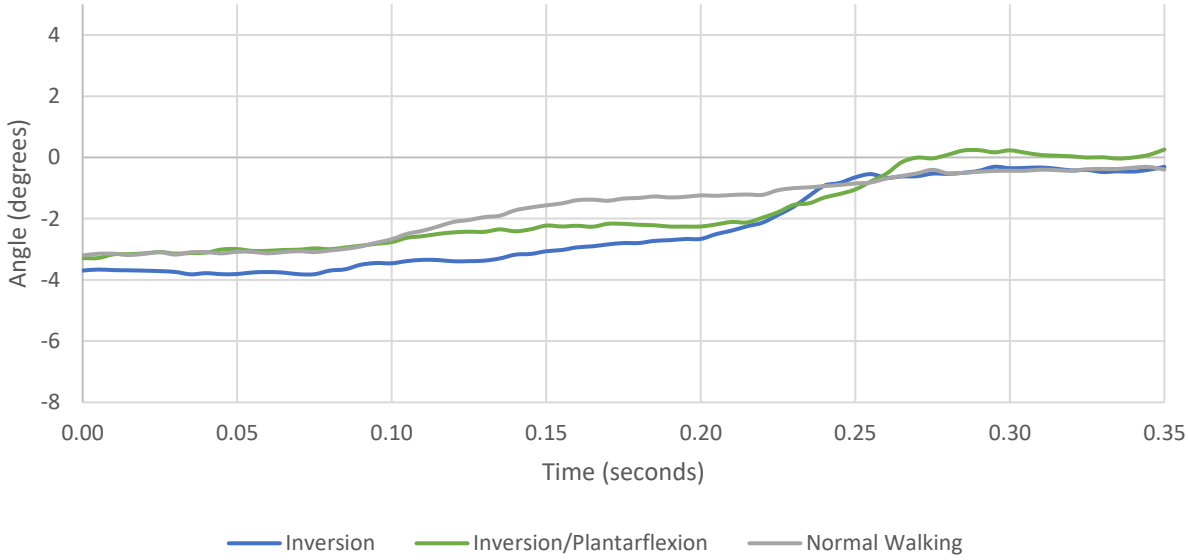

Figure 9. Time series of knee angles in the y direction. Conventions: y (+ adduction /- abduction). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

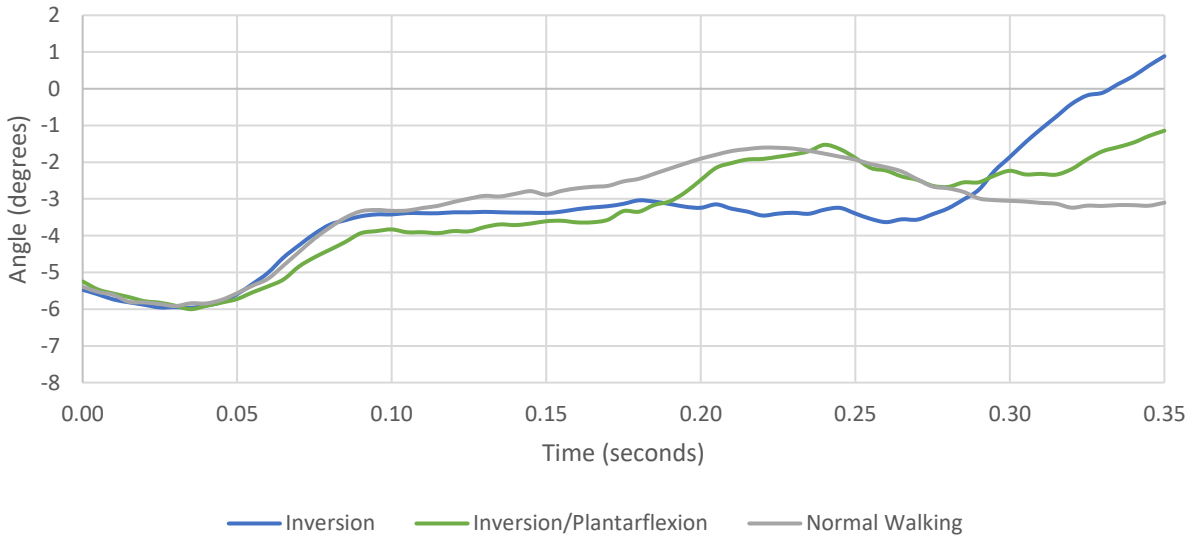

Figure 10. Time series of knee angles in the z direction. Conventions: z (+ internal rotation / - external rotation). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

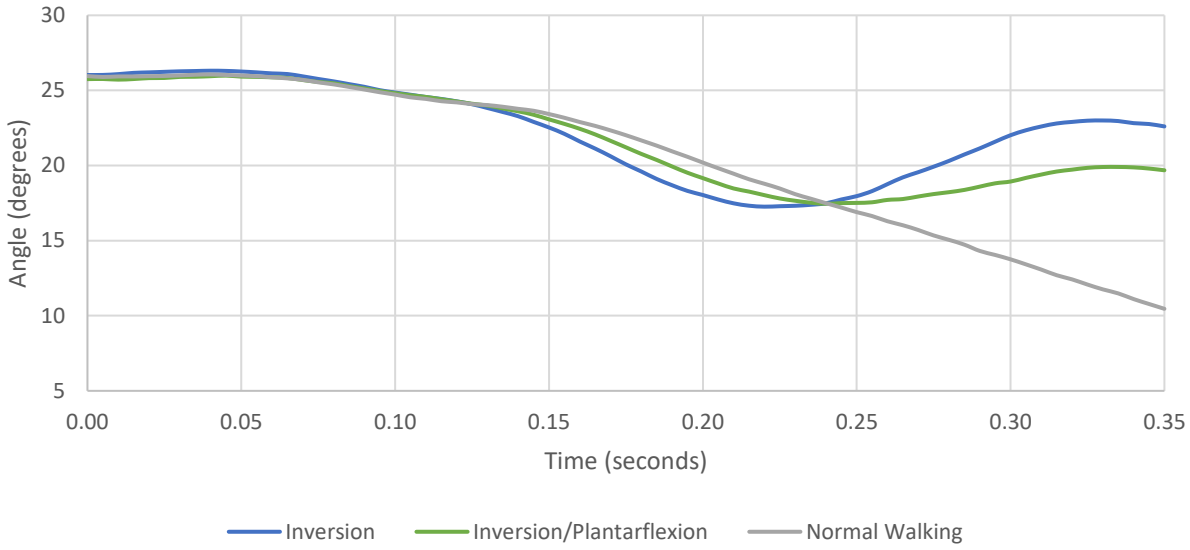

Figure 11. Time series of hip angles in the x direction. Conventions: x (+ flexion / - extension). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

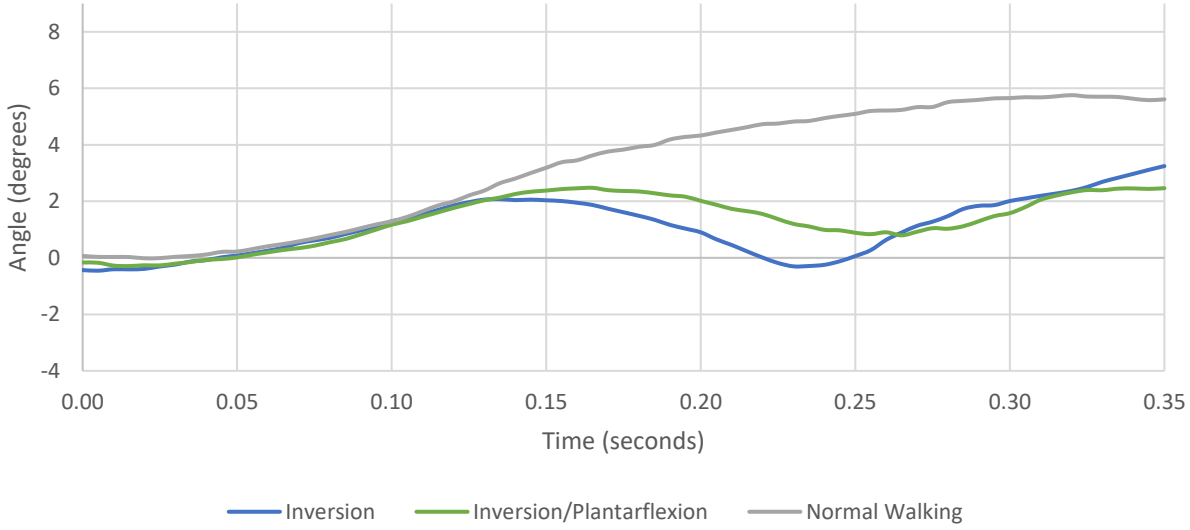

Figure 12. Time series of hip angles in the y direction. Conventions: y (+ adduction / - abduction). Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

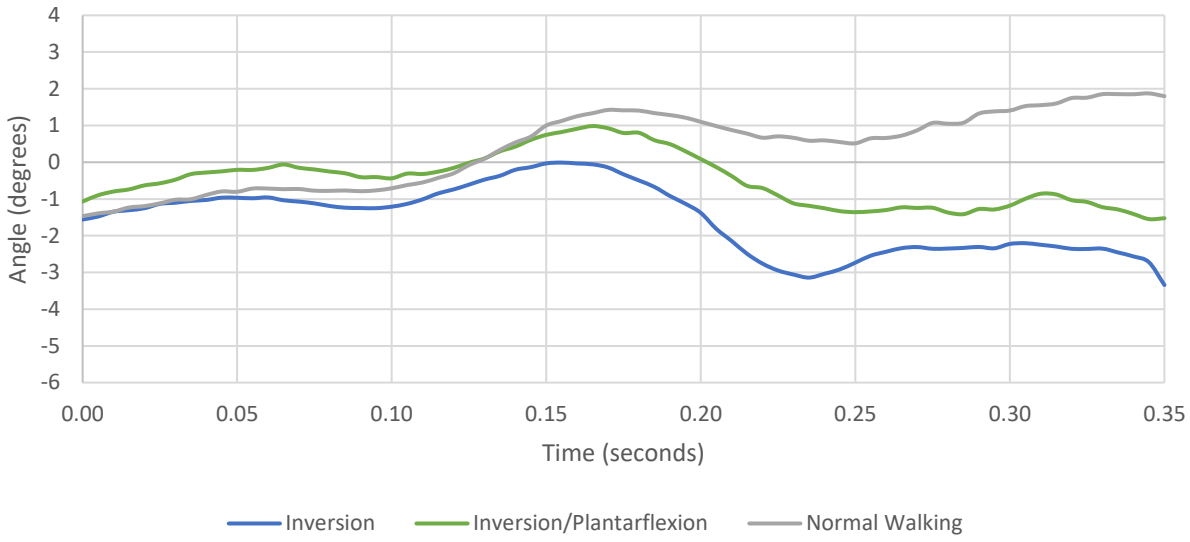

Figure 13. Time series of hip angles in the z direction. Conventions: z (+ internal rotation / - external rotation. Plot begins at heel strike labeled as zero seconds to 350 milliseconds after.

Peak Angle

Significant differences in peak angle (degrees) were found across conditions for all joint levels and directions.

Table 5

Peak Angle Descriptive Statistics. Angles reported in degrees. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

Variable		**INV**	**INV/PF**	**NW**
Joint	**Direction**	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Dominant Ankle	X	77.33 ± 6.15	70.55 ± 4.51	74.18 ± 4.93
	Y	-5.66 ± 14.12	-12.23 ± 11.99	-17.27 ± 7.94
	Z	-38.59 ± 5.76	-26.52 ± 7.33	-19.28 ± 5.92
Dominant Knee	X	-23.61 ± 6.40	-22.78 ± 10.34	-13.36 ± 7.87
	Y	-1.57 ± 6.17	-1.77 ± 6.06	-2.58 ± 4.83
	Z	-6.72 ± 15.43	-8.01 ± 14.05	-6.64 ± 13.59
Dominant Hip	X	27.78 ± 9.92	27.33 ± 9.54	26.84 ± 10.48
	Y	2.22 ± 7.63	2.33 ± 7.09	4.98 ± 6.12
	Z	-2.41 ± 10.60	-.99 ± 10.86	.74 ± 9.90

Ankle.

A repeated measures ANOVA determined that that the variables of peak ankle angle differed significantly between trial conditions ($F(6,72) = 48.822$, $p < .001$).

The within-subjects effect with the Greenhouse-Geisser correction for the x direction was significant ($F(1.381) = 29.558$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p < .001$) reduction in angle in the x direction for the combined inversion/plantarflexion (70.55±4.51) trial condition as compared to the inversion (77.33±5.15). Similarly, a significant ($p < .001$) reduction in angle in the x direction for the combined inversion/plantarflexion (70.55±4.51) trial condition as compared to normal walking (74.18±4.93). It also showed a significant ($p = .025$) reduction in angle in the x direction for the normal walking (74.18±4.93) trial condition as compared to inversion (77.33±6.15).

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 21.892$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .002$) increase in angle in the y direction for the inversion (-5.66±14.12) trial condition as compared to the combined inversion/plantarflexion (-12.23±11.99). Similarly, a significant ($p = .011$) increase in angle in the y direction for the combined inversion/plantarflexion (-12.23±11.99) trial condition as compared to normal walking (-17.27±7.94). It also showed a significant ($p < .001$) increase in angle in the y direction for the inversion (-5.66±14.12) trial condition as compared to normal walking (-17.27±7.94).

The within-subjects effect with the Greenhouse-Geisser correction for the z direction was significant ($F(1.601) = 212.967$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p < .001$) reduction in angle in the z direction for the inversion (-38.59±5.76) trial condition as compared to the combined inversion/plantarflexion (-

26.52±7.33). Similarly, a significant ($p < .001$) reduction in angle in the z direction for the combined inversion/plantarflexion (-26.52±7.33) trial condition as compared to normal walking (-19.28±5.92). It also showed a significant ($p < .001$) reduction in angle in the z direction for the inversion (-38.59±5.76) trial condition as compared to normal walking (-19.28±5.92).

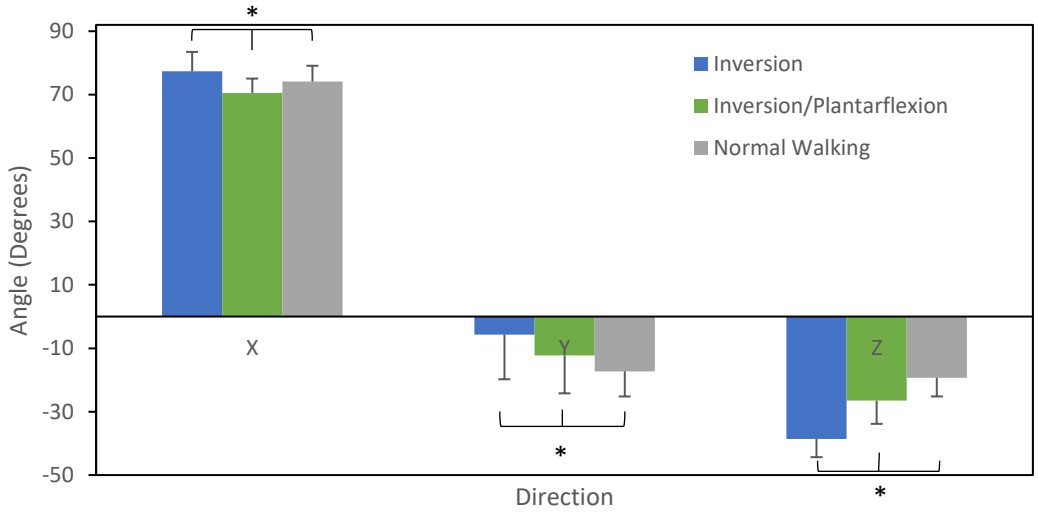

Figure 14. Descriptive statistics of peak ankle angles. Conventions: x (+ dorsiflexion / - plantarflexion), y (+ inversion / - eversion), z (+ internal rotation / - external rotation). * = Significant differences, $p < .05$

Knee.

Another repeated measures ANOVA determined that that the variables of peak knee angle differed significantly between trial conditions ($F(6,72) = 9.603$, $p < .001$).

The within-subjects effect with sphericity assumed for the x direction was significant ($F(2) = 28.639$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p < .001$) reduction in angle in the x direction for the inversion (-23.61±6.40) trial condition as compared to the normal walking (-13.36±10.34). Similarly, a significant ($p < .001$)

reduction in angle in the x direction for the combined inversion/plantarflexion (-22.78±10.34) trial condition as compared to normal walking (-13.36±7.87).

The within-subjects effect with sphericity assumed for the y direction was not significant ($p = .083$). Post hoc tests using the Bonferroni correction also showed that there were no significant differences in the y direction between trial conditions.

The within-subjects effect with sphericity assumed for the z direction was not significant ($p = .363$). Post hoc tests using the Bonferroni correction also showed no significant differences in the z direction between trial conditions.

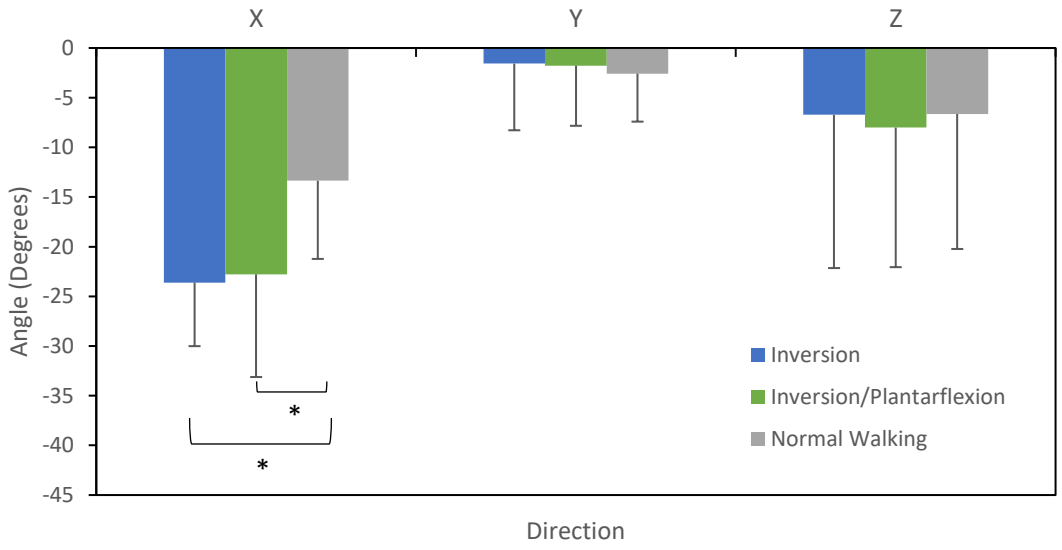

Figure 15. Descriptive statistics of peak knee angles. Conventions: x (+ extension / - flexion), y (+ adduction,/- abduction), z (+ internal rotation / - external rotation). * = Significant differences, $p < .05$

Hip.

The next measures ANOVA determined that that the variables of peak hip angle differed significantly between trial conditions ($F(6,72) = 5.350$, $p < .001$).

The within-subjects effect with sphericity assumed for the x direction was not significant ($p = .092$). Post hoc tests using the Bonferroni correction confirmed that there were no significant differences in angle in the x direction between trial conditions.

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 12.289$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .003$) reduction in angle in the y direction for the inversion (2.22±7.63) trial condition as compared to the normal walking (4.98±6.12). Similarly, a significant ($p < .001$) reduction in angle in the y direction for the combined inversion/plantarflexion (2.33±7.09) trial condition as compared to normal walking (4.98±6.12).

The within-subjects effect with sphericity assumed for the z direction was significant ($F(2) = 5.367$, $p = .009$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .002$) reduction in angle in the z direction for the inversion (-2.41±10.60) trial condition as compared to the normal walking (.74±9.90).

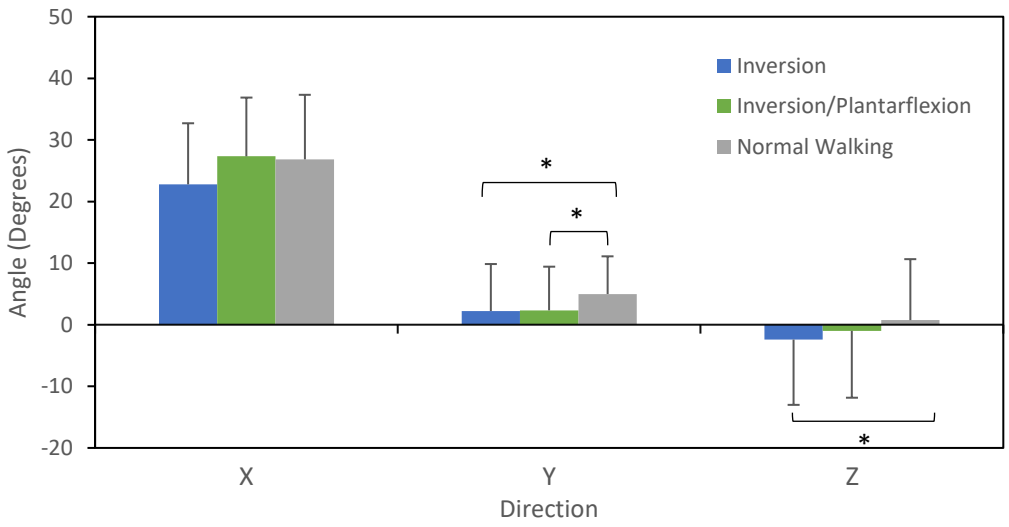

Figure 16. Descriptive statistics of peak hip angles. Conventions: x (+ flexion / - extension), y (+ adduction / - abduction), z (+ internal rotation / - external rotation). * = Significant differences, $p < .05$

Peak Velocity

Significant differences in peak velocity (deg/s) were found across conditions for all joint levels and directions.

Table 6

Peak Velocity Descriptive Statistics. Velocities reported in degrees/second. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

Variable		**INV**	**INV/PF**	**NW**
Joint	**Direction**	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Dominant Ankle	X	27.60 ± 199.93	-225.94 ± 69.03	-200.75 ± 69.03
	Y	559.05 ± 246.75	11.07 ± 383.92	-180.07 ± 112.39
	Z	183.59 ± 298.69	-22.91 ± 351.38	-117.02 ± 99.51
Dominant Knee	X	-307.15 ± 100.86	-305.44 ± 141.92	-111.45 ± 127.50
	Y	206.14 ± 129.90	160.27 ± 154.71	47.04 ± 67.51
	Z	44.62 ± 331.07	-147.26 ± 365.84	48.68 ± 152.82
Dominant Hip	X	154.09 ±152.57	56.76 ± 119.35	-122.83 ± 60.14
	Y	115.35 ± 154.99	32.21 ± 116.20	-109.35 ± 83.13
	Z	59.10 ± 191.97	64.41 ± 160.48	-109.35 ± 83.13

Ankle.

A repeated measures ANOVA determined that that the variables of peak ankle velocity differed significantly between trial conditions ($F(6,72) = 12.024$, $p < .001$).

The within-subjects effect with the Greenhouse-Geisser correction for the x direction was significant ($F(1.549) = 10.646$, $p = .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .009$) decrease in velocity in the x direction for the combined inversion/plantarflexion (-225.94±258.43) trial condition as compared to the inversion (27.60±199.93). Similarly, a significant ($p = .011$) increase in velocity in the x direction for the inversion (27.60±199.93) trial condition as compared to normal walking (-200.75±69.03).

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 41.775$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a

significant (p <.001) increase in velocity in the y direction for the inversion (559.05±246.75) trial condition as compared to the combined inversion/plantarflexion (11.07±383.92). Similarly, a significant (p < .001) increase in velocity in the y direction for the inversion (559.05±246.75) trial condition as compared to normal walking (-180.07±112.39).

The within-subjects effect with sphericity assumed for the z direction was significant (F (2) = 6.951, p = .003). Post hoc tests using the Bonferroni correction revealed that there was a significant (p = .002) increase in velocity in the z direction for the inversion (183.59±298.69) trial condition as compared to the normal walking (-117.02±99.51).

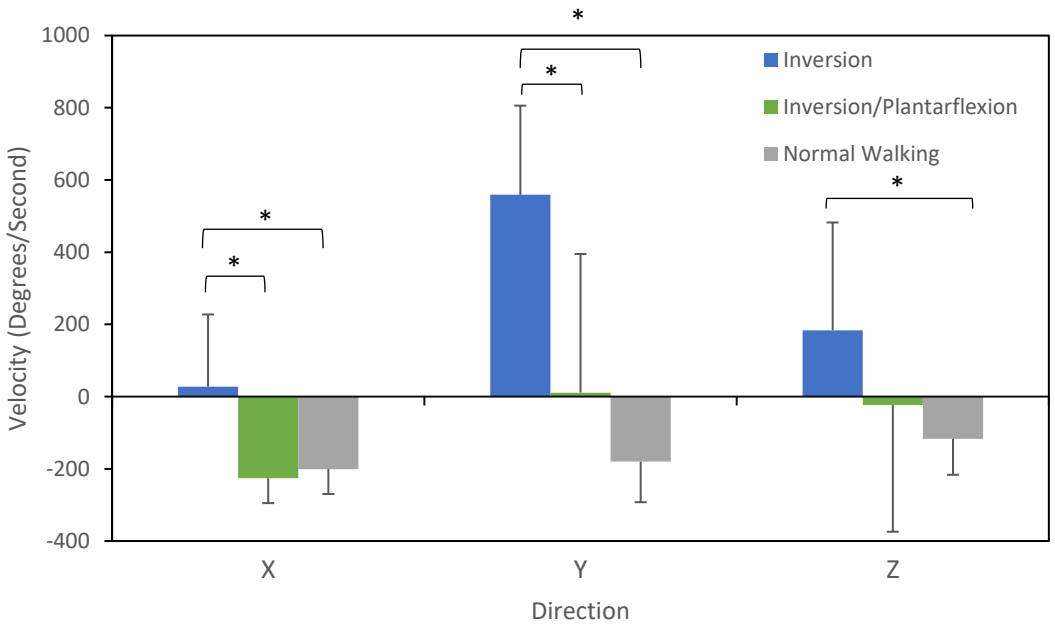

Figure 17. Descriptive statistics of peak ankle velocities. Conventions: x (+ dorsiflexion / - plantarflexion), y (+ inversion / - eversion), z (+ internal rotation / - external rotation). * = Significant differences, p < .05

Knee.

Another repeated measures ANOVA determined that that the variables of peak knee velocity differed significantly between trial conditions (F (6,72) = 8.857, p <.001).

The within-subjects effect with sphericity assumed for the x direction was significant ($F(2) = 22.631$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p < .001$) reduction in velocity in the x direction for the inversion (-307.15±100.86) trial condition as compared to the normal walking (-111.45±127.50). Similarly, a significant ($p < .001$) reduction in velocity in the x direction for the combined inversion/plantarflexion (-305.44±141.92) trial condition as compared to normal walking (-111.45±127.50).

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 10.773$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p <.001$) increase in velocity in the y direction for the inversion (206.14±129.90) trial condition as compared to the normal walking (47.04±67.51). Similarly, a significant ($p =.022$) increase in velocity in the y direction for the combined inversion/plantarflexion (160.27±154.71) trial condition as compared to normal walking (47.04±67.51).

The within-subjects effect with sphericity assumed for the z direction was not significant ($p = .055$). Post hoc tests using the Bonferroni correction confirmed no significant differences in velocity the z direction between conditions.

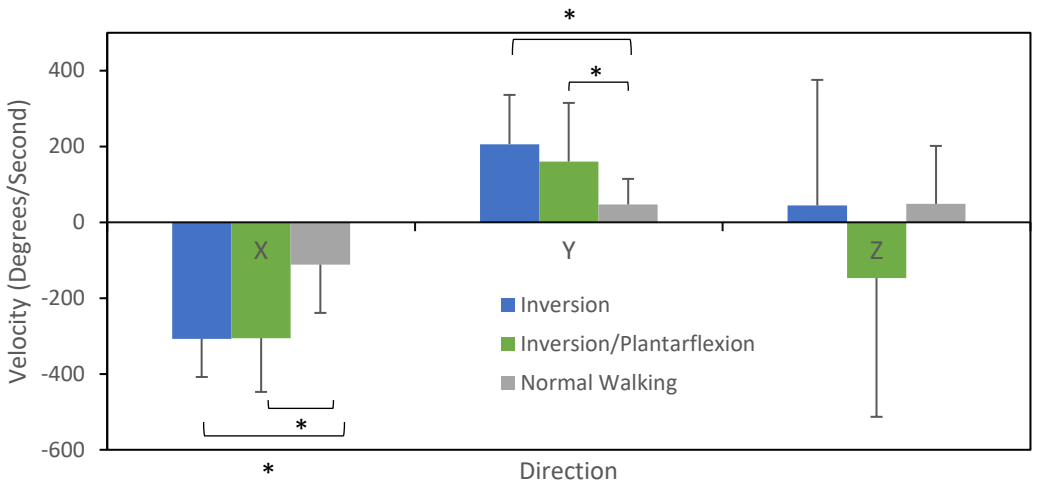

Figure 18. Descriptive statistics of peak knee velocities. Conventions: x (+ extension / - flexion), y (+ adduction,/- abduction), z (+ internal rotation / - external rotation). * = Significant differences, $p < .05$

Hip.

The next measures ANOVA determined that that the variables of peak hip velocity differed significantly between trial conditions ($F(6,72) = 11.945$, $p < .001$).

The within-subjects effect with sphericity assumed for the x direction was significant ($F(2) = 37.884$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .015$) increase in velocity in the x direction for the inversion (154.09±152.57) trial condition as compared to the combined inversion/plantarflexion (56.76±119.35). Similarly, a significant ($p < .001$) increase in velocity in the x direction for the combined inversion/plantarflexion (56.76±119.35) trial condition as compared to normal walking (-122.83±60.14). It also showed a significant ($p < .001$) increase in velocity in the x direction for the inversion (154.09±152.57) trial condition as compared to normal walking (-122.83±60.14).

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 19.97$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a

significant (p < .001) increase in velocity in the y direction for the inversion (115.35±154.99) trial condition as compared to the normal walking (-109.35±83.13). Similarly, a significant (p =.002) increase in velocity in the y direction for the combined inversion/plantarflexion (32.21±116.20) trial condition as compared to normal walking (-109.35±83.13).

The within-subjects effect with sphericity assumed for the z direction was significant (F (2) = 9.078, p = .001). Post hoc tests using the Bonferroni correction revealed that there was a significant (p = .010) increase in velocity in the z direction for the inversion (59.10±191.97) trial condition as compared to the normal walking (-109.35±83.13). Similarly, a significant (p = .003) increase in velocity in the z direction for the combined inversion/plantarflexion (64.41±160.48) trial condition as compared to normal walking (-109.35±83.13).

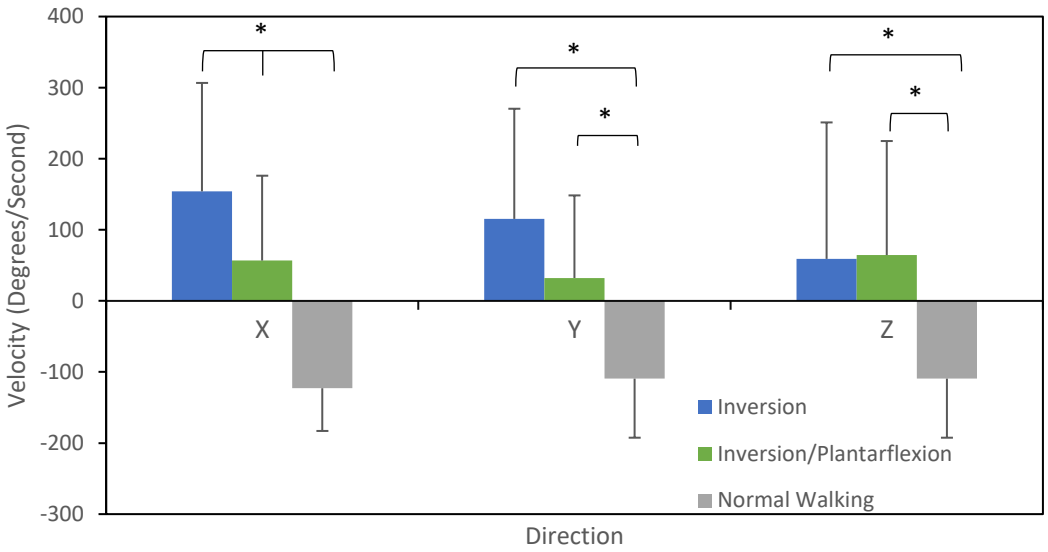

Figure 19. Descriptive statistics of peak hip velocities. Conventions: x (+ flexion / - extension), y (+ adduction / - abduction), z (+ internal rotation / - external rotation). * = Significant differences, p < .05

Time to Peak Angle

Ankle.

Significant differences in time to peak angle (s) were found across conditions for all joint levels and directions.

Table 7

Time to Peak Angle Descriptive Statistics. Values reported in seconds. INV = Inversion perturbation, INV/PF = combined inversion/plantarflexion perturbation, NW = normal walking condition.

Variable		INV	INV/PF	NW
Joint	**Direction**	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
Dominant Ankle	X	.31 ± .06	.17 ± .11	.33 ± .06
	Y	.16 ± .07	.21 ± .06	.17 ± .07
	Z	.26 ± .04	.22 ± .05	.05 ± .04
Dominant Knee	X	.32 ± .02	.29 ± .04	.19 ± .05
	Y	.18 ± .12	.17 ± .11	.14 ± .11
	Z	.19 ± .10	.17 ± .10	.14 ± .11
Dominant Hip	X	.14 ± .10	.09 ± .09	.05 ± .04
	Y	.27 ± .06	.25 ± .07	.14 ± .10
	Z	.20 ± .09	.19 ± .08	.18 ± .12

A repeated measures ANOVA determined that that the variables of time to peak ankle angle differed significantly between trial conditions ($F(6,72) = 52.348$, $p < .001$).

The within-subjects effect with sphericity assumed for the x direction was significant ($F(2) = 25.952$ $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p < .001$) reduction in time to peak angle in the x direction for the combined inversion/plantarflexion (.17±.11) trial condition as compared to the inversion (.31±.06). Similarly, a significant ($p < .001$) reduction in time to peak angle in the x direction for the combined inversion/plantarflexion (.17±.11) trial condition as compared to normal walking (.34±.06).

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 13.334$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .001$) reduction in time to peak angle in the y direction for the inversion (.16±.07) trial condition as compared to the combined normal walking (.27±.07). Similarly, a significant ($p = .002$) reduction in time to peak angle in the y direction for the combined inversion/plantarflexion (.21±.06) trial condition as compared to normal walking (.27±.07).

The within-subjects effect with sphericity assumed for the z direction was significant ($F(2) = 222.219$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .017$) increase in time to peak angle in the z direction for the inversion (.26±.04) trial condition as compared to the combined inversion/plantarflexion (.22±.05). Similarly, a significant ($p < .001$) increase in time to peak angle in the z direction for the combined inversion/plantarflexion (.22±.05) trial condition as compared to normal walking (.05±.04). It also showed a significant ($p < .001$) increase in time to peak angle in the z direction for the inversion (.26±.04) trial condition as compared to normal walking (.05±.04).

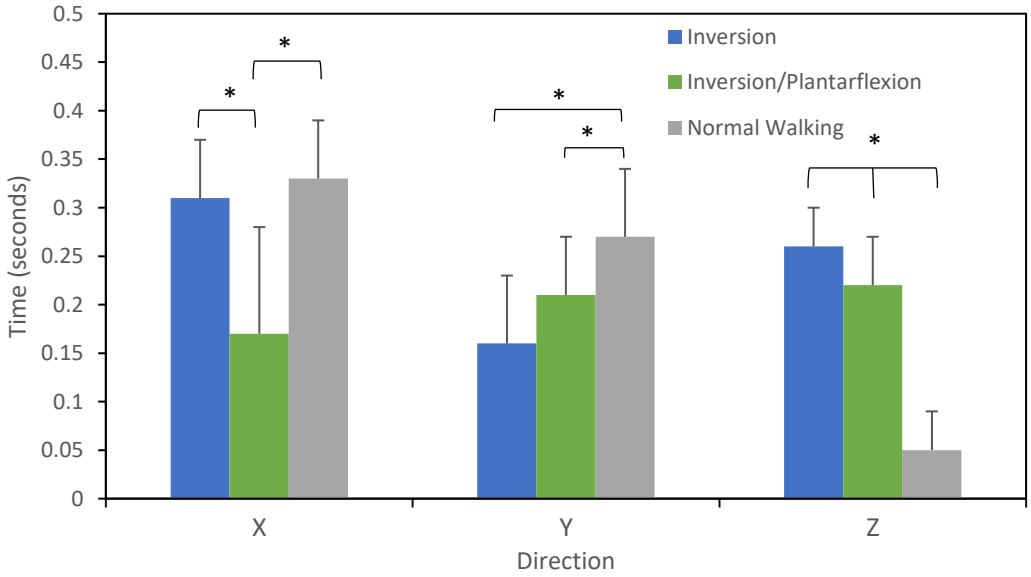

Figure 20. Descriptive statistics of time to peak ankle angle. * = Significant differences, $p < .05$

Knee.

A repeated measures ANOVA determined that that the variables of time to peak knee angle differed significantly between trial conditions ($F(6,72) = 11.998$, $p < .001$).

The within-subjects effect with sphericity assumed for the x direction was significant ($F(2) = 54.278$, $p < .001$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .012$) reduction in time to peak angle in the x direction for the combined inversion/plantarflexion (.29±.04) trial condition as compared to the inversion (.32±.02). There was a significant ($p < .001$) increase in time to peak angle in the x direction for the combined inversion/plantarflexion (.29±.04) trial condition as compared to normal walking (.19±.05). It also showed a significant ($p = .013$) reduction in time to peak angle in the x direction for the normal walking (.19±.05) trial condition as compared to inversion (.32±.02).

The within-subjects effect with sphericity assumed for the y direction was significant ($F(2) = 4.258$, $p = .021$). Post hoc tests using the Bonferroni correction revealed that there was a significant ($p = .029$) increase in time to peak angle in the y direction for the inversion (.18±.12) trial condition as compared to the normal walking (.14±.12).

The within-subjects effect with sphericity assumed for the z direction was significant ($F(2) = 3.883$, $p = .029$). Post hoc tests using the Bonferroni correction did not show any significant differences between conditions.

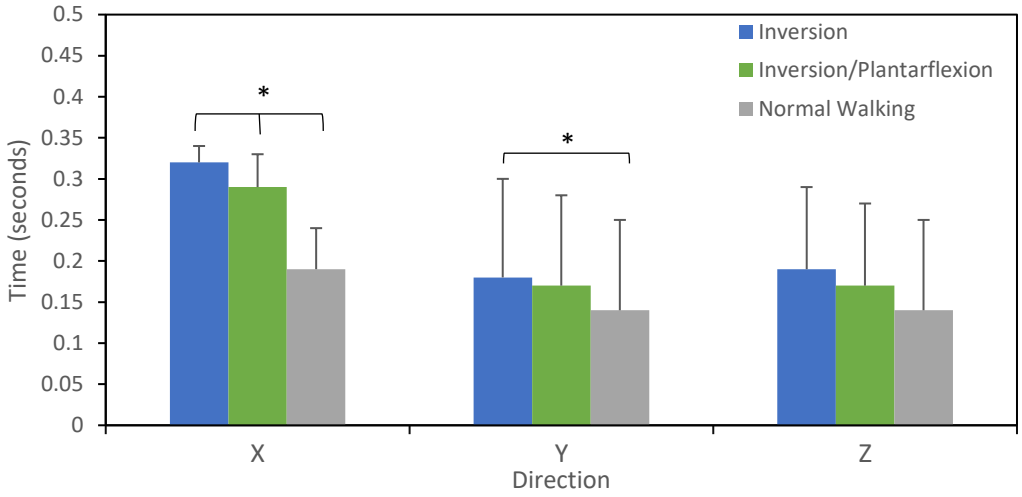

Figure 21. Descriptive statistics of time to peak knee angle. * = Significant differences, p < .05

Hip.

A repeated measures ANOVA determined that that the variables of peak ankle angle differed significantly between trial conditions (F (6,72) = 3.871, p =.002).

The within-subjects effect with sphericity assumed for the x direction was significant (F(2) = 12.314, p < .001). Post hoc tests using the Bonferroni correction revealed that there was a significant (p = .001) increase in time to peak angle in the x direction for the inversion (.14±.10) trial condition as compared to the normal walking (.05±.04). Similarly, a significant (p = .011) increase in time to peak angle in the x direction for the combined inversion/plantarflexion (.09±.09) trial condition as compared to normal walking (.05±.04).

The within-subjects effect with sphericity assumed for the y direction was not significant (p = .497).

The within-subjects effect with the Greenhouse-Geisser correction for the z direction was not significant (p = .794).

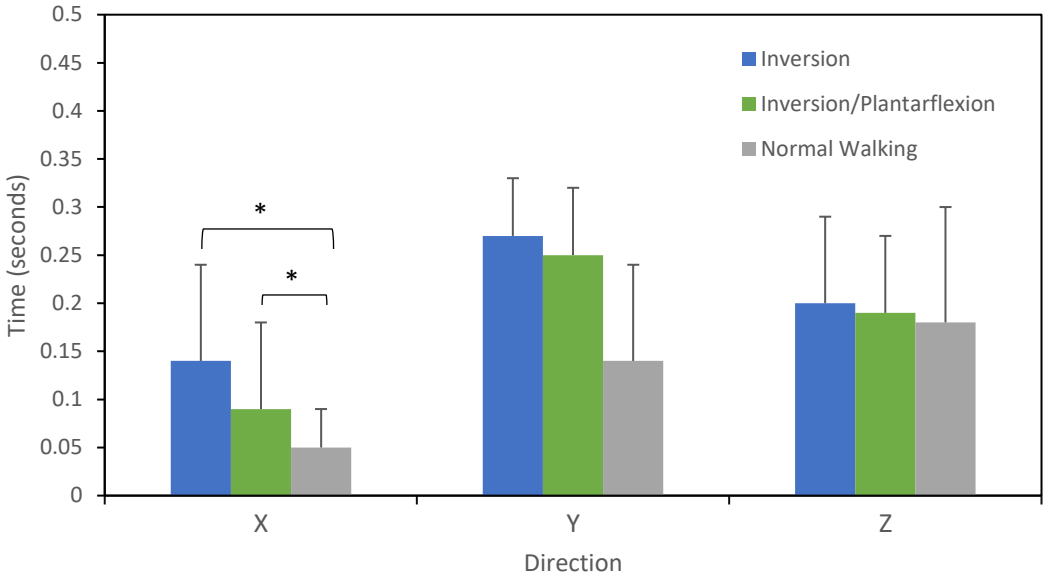

Figure 22. Descriptive statistics of time to peak hip angle. * = Significant differences, $p < .05$

Discussion

With a purpose to descriptively compare the kinematic response following unexpected ankle perturbations during walking, this study investigated kinematics at the ankle, knee, and hip on the perturbed side. Significant findings at each joint level suggested systemic involvement and an unloading response. The kinematic findings of this study help to both provide detail about the response to the perturbation and give information about mechanics of the perturbation itself.

Beginning with the peak ankle angle results, the inversion trial condition showed more dorsiflexion than the combined inversion and plantarflexion condition as well as the control of normal walking. However, the combined inversion/plantarflexion condition was significantly less dorsiflexed than the normal walking control. This could be indictive of the difference in perturbation where during the inversion perturbation there is a greater dorsiflexion angle than the inversion/plantarflexion condition simply because the mechanism involves more plantarflexion. Or it could be an indication of the extended response to the perturbation, as the peak

plantarflexion angle does not occur immediately following the response as evidenced by the time to peak angle (seconds) (INV .31 ± .06, PF .17 ± .11, NW .33 ± .06).

The mechanics in the x direction seem to agree with the injury analyses that also reported dorsiflexion angles, lacking plantarflexion in the injury mechanism.(Fong et al., 2009; Eirik Kristianslund et al., 2011; Mok et al., 2011) In the y direction all conditions resulted in a peak eversion angle. Differences included significantly less eversion for the inversion trial as compared to the other two conditions. Also, the combined inversion/plantarflexion condition was significantly less everted than the normal walking control. Supported by the physical mechanism with the apparatus dropping the ankle into inversion and inversion/plantarflexion it would be expected that the perturbations would elicit less eversion or more inversion than the normal walking control.

During these perturbations, the foot is neutral before contact then as it contacts the platform it pushes into eversion against the drop, similar to the loading response that occurs in eversion with normal walking which also recorded a peak eversion angle. Additionally, in the z direction during the inversion perturbation, the ankle is significantly more externally rotated than the other two conditions. The inversion/plantarflexion condition also shows a significantly more externally rotated ankle than the control. In a similar manner this could indicate that the subject is acting in opposition to the perturbation that would naturally act to internally rotate. Moving up to the knee significant differences were found in the x direction; there were significantly greater flexion angles during both perturbation conditions than normal walking, but no significant difference between perturbations. This result supports the expected response with the activation of flexion reflexes following perturbation, as were present in previous literature.(E. Kristianslund et al., 2011; Santos et al., 2008) As both perturbations elicited the knee flexion reflexes, but the

flexion angles were not significantly different it does not provide support to differentiate the perturbations if only studying the response. The hip joint showed significantly less adduction for both perturbation conditions over the normal walking condition. With less adduction during the response, it could be to aid in realigning the center of mass over the base of support, shifting over the perturbed limb, but analysis of the center of mass would be needed to support this. There was also significantly more external rotation during the inversion trial as compared to normal walking, which aligns with the results found at the ankle joint level potentially to counteract the perturbation. These findings demonstrate that the response is active, in general the subjects seem to respond by everting and externally rotating as supported by the ankle and hip kinematic data as well as flexion reflexes activated at the knee. The subjects do not seem to fall passively into the perturbations, instead they seem to respond in opposition.

Peak velocity data captured at the ankle in the x direction demonstrated that the inversion condition had a significantly faster rate of dorsiflexion than both other trial conditions which demonstrated a peak plantarflexion velocity instead. There was no significant difference between the combined inversion/plantarflexion condition and the normal walking for their peak plantarflexion velocities, perhaps indicating that the inversion/plantarflexion condition does not perturb the ankle joint in the sagittal plane any more than is normally experienced in walking. In the y direction, the peak velocity data supports the intended mechanism with a finding of significantly faster rate of ankle inversion for the inversion condition as compared to the combined inversion/plantarflexion and the normal walking control. However, it should also be noted that the combined inversion/plantarflexion condition did not have a significantly different inversion velocity than normal walking. This peak velocity in the y direction for the inversion condition was recorded at 559.05 ± 246.75 degrees/second, which falls above the previously

suggested safe threshold of 300 degrees/second as well as some common sporting motions which were reported as less than 100 degrees/second(Chu et al., 2010).

With no injuries reported during collection this suggest that the 300 degrees/second threshold is conservative. Additionally, the inversion velocity found for the inversion condition still falls below some of the inversion velocities reported in available case studies: 1752 deg/s from a high jump injury(Mok et al., 2011), 1397 deg/s from a field hockey case(Mok et al., 2011), 1290 deg/s from a running and cutting injury(Gehring et al., 2013), and close but still below the reported 638 deg/s in another running and cutting injury.(Fong et al., 2009) The inversion velocity found in the current study was more similarly aligned with that during walking (403 deg/s) and jumping (595 deg/s) as reported with a trap door mechanism dropping into 25 degrees of inversion.(Nieuwenhuijzen et al., 2002)

In the z direction the ankle joint demonstrated a significantly faster rate of internal rotation during the inversion condition than normal walking. Both the normal walking and the combined inversion/plantarflexion condition showed peak external rotation velocities instead, with no significant difference between each other. This demonstrates that potentially the inversion/plantarflexion condition does not perturb the ankle joint in the transverse plan any more than is normally experienced with walking. Moving up to the knee joint, a faster flexion velocity is experienced for both perturbation conditions as compared to normal walking. This again supports the potential finding of flexion reflexes following the perturbation as predicted and supported by other findings.(Eirik Kristianslund et al., 2011; Santos et al., 2008)

In the y direction the knee has a significantly faster adduction velocity for both perturbation conditions as compared to normal walking. This finding aligns with the potential suggestion that the response is to oppose the perturbation and act in the opposite direction.

Further up the kinetic chain at the hip joint, there were significantly faster flexion velocities for the inversion condition as compared to both other conditions and a significantly faster flexion velocity for the inversion/plantarflexion perturbation over normal walking. Although, no significant difference or indication of the flexion reflexes were found in the hip angular recordings, the velocity data here does support the flexion response with these increased flexion velocities. In the y direction the hip responds with a significantly faster adduction velocity for both perturbations over normal walking. In the z direction there was a significantly faster rate of internal rotation for both perturbation conditions over normal walking as well.

Time to peak findings appear to be relevant with the similarity in the knee's angular response, as all conditions demonstrated flexion angles with significant results. Time to peak knee flexion angle was shorter for normal walking than either perturbation condition. Also, time to peak knee flexion angle was shorter for the inversion/plantarflexion condition than the inversion trial. Thus, temporally it appears that the perturbations elicit a different type of unloading response with perhaps a slower reaction time for the flexion reflex during the inversion condition as opposed to the inversion/plantarflexion condition.

With large standard deviations of the kinematic data along with large standard deviations present in EMG signals reported in Chapter 1, it indicates high variability in the motions. This could suggest that the kinematic strategy is not set for each perturbation among subjects. However, in general the results suggest that the kinematic response to unexpected perturbations involves adjustments at all joint levels, indicating a systemic response that can be further classified.

Limitations of the kinematic analysis include the computation of the angular data. The peak inversion ankle angle was not reported due to the eversion ankle angle having a larger

magnitude, which was then classified as the peak angle due to the commands in the loop within the Matlab code. Descriptively it would be beneficial to compare the peak inversion ankle angle to that of other literature with corresponding available data. Additionally, more sensitive metrics may be beneficial for studying phases of the perturbation itself and the response. Other limitations include the digitization of the marker data as some subject's height allowed for occlusion of the pelvis markers by the handrails. Moreover, since subjects were aware of the possibility of perturbation at any time during any trial their mechanics may be different in anticipation of the perturbation.

This study provided a description of the kinematics following two unexpected perturbation conditions. The response to both perturbations showed ipsilateral flexion reflexes in support of an unloading reaction and demonstrated involvement at each joint level. The differences in inversion velocity may suggest that inversion condition elicits a more dramatic perturbation whereas the inversion/plantarflexion and normal walking conditions do not result such substantial angular velocities. While providing an initial descriptive analysis of the conditions, refinement of the window of analysis may aid in providing more context to the phases of the perturbation and response.

REFERENCES

Chu, V.W.-S., Fong, D.T.-P., Chan, Y.-Y., Yung, P.S.-H., Fung, K.-Y., Chan, K.-M., 2010. Differentiation of ankle sprain motion and common sporting motion by ankle inversion velocity. Journal of Biomechanics 43, 2035–2038. https://doi.org/10.1016/j.jbiomech.2010.03.029

Fernandes, N., Allison, G., Hopper, D., 2000. Peroneal latency in normal and injured ankles at varying angles of perturbation. Clin Orthop 193–201.

Ferran, N.A., Oliva, F., Maffulli, N., 2009. Ankle Instability. Sports Medicine and Arthroscopy Review 17, 139–145. https://doi.org/10.1097/JSA.0b013e3181a3d790

Fong, D.T.-P., Hong, Y., Chan, L.-K., Yung, P.S.-H., Chan, K.-M., 2007. A Systematic Review on Ankle Injury and Ankle Sprain in Sports: Sports Medicine 37, 73–94. https://doi.org/10.2165/00007256-200737010-00006

Fong, D.T.-P., Hong, Y., Shima, Y., Krosshaug, T., Yung, P.S.-H., Chan, K.-M., 2009. Biomechanics of Supination Ankle Sprain: A Case Report of an Accidental Injury Event in the Laboratory. Am J Sports Med 37, 822–827. https://doi.org/10.1177/0363546508328102

Gehring, D., Wissler, S., Mornieux, G., Gollhofer, A., 2013. How to sprain your ankle – a biomechanical case report of an inversion trauma. Journal of Biomechanics 46, 175–178. https://doi.org/10.1016/j.jbiomech.2012.09.016

Hall, E.A., Simon, J.E., Docherty, C.L., 2016. Using Ankle Bracing and Taping to Decrease Range of Motion and Velocity During Inversion Perturbation While Walking. Journal of Athletic Training 51, 283–290. https://doi.org/10.4085/1062-6050-51.5.06

Hertel, J., n.d. Functional Anatomy, Pathomechanics, and Pathophysiology of Lateral Ankle Instability 12.

Konradsen, L., Voigt, M., Hojsgaard, C., 1997. Ankle Inversion Injuries: The Role of the Dynamic Defense Mechanism. Am J Sports Med 25, 54–58. https://doi.org/10.1177/036354659702500110

Kristianslund, E., Bahr, R., Krosshaug, T., 2011. Kinematics and kinetics of accidental ankle sprain in 3D motion analysis lab. British Journal of Sports Medicine 45, 329–329. https://doi.org/10.1136/bjsm.2011.084038.55

Kristianslund, Eirik, Bahr, R., Krosshaug, T., 2011. Kinematics and kinetics of an accidental lateral ankle sprain. Journal of Biomechanics 44, 2576–2578. https://doi.org/10.1016/j.jbiomech.2011.07.014

Liu, W., Jain, T., Wauneka, C., 2012. A New Dimension in the Study of Human Functional Joint Instability. AMM 249–250, 1271–1276. https://doi.org/10.4028/www.scientific.net/AMM.249-250.1271

McLoda, T., Hansen, A.J., Birrer, D., 2004. EMG analysis of peroneal and tibialis anterior muscle activity prior to foot contact during functional activities. Electromyogr clin Neurophysiol 223–227.

Mok, K.-M., Fong, D.T.-P., Krosshaug, T., Engebretsen, L., Hung, A.S.-L., Yung, P.S.-H., Chan, K.-M., 2011. Kinematics Analysis of Ankle Inversion Ligamentous Sprain Injuries in Sports: 2 Cases During the 2008 Beijing Olympics. Am J Sports Med 39, 1548–1552. https://doi.org/10.1177/0363546511399384

Nieuwenhuijzen, P.H.J.A., Grüneberg, C., Duysens, J., 2002. Mechanically induced ankle inversion during human walking and jumping. Journal of Neuroscience Methods 117, 133–140. https://doi.org/10.1016/S0165-0270(02)00089-4

Santos, M.J., Liu, H., Liu, W., 2008. Unloading reactions in functional ankle instability. Gait & Posture 27, 589–594. https://doi.org/10.1016/j.gaitpost.2007.08.001

Stanek, J.M., McLoda, T.A., Csiszer, V.J., Hansen, A.J., 2011. Hip- and Trunk-Muscle Activation Patterns During Perturbed Gait. Journal of Sport Rehabilitation 20, 287–295. https://doi.org/10.1123/jsr.20.3.287

Ty Hopkins, J., McLoda, T., McCaw, S., 2007. Muscle activation following sudden ankle inversion during standing and walking. Eur J Appl Physiol 99, 371–378. https://doi.org/10.1007/s00421-006-0356-9

Vaes, P., Duquet, W., Gheluwe, B.V., 2002. Peroneal Reaction Times and Eversion Motor Response in Healthy and Unstable Ankles 7.

Comments are notated in red above their corresponding pipeline commands.
The following is saved as 8 on Part 1
The first part of the pipeline is considered a load pipeline. All files are loaded into the workspace and the subject specific model is created and applied to each file.
Here the user is prompted to load the subject calibration file into the workspace.
Pipeline_Breakpoint
/PAUSE_MESSAGE= C3D for Hybrid Model
;
Further documentation of this command can be found at: https://www.c-motion.com/v3dwiki/index.php?title=Create_Hybrid_Model
Create_Hybrid_Model
! /CALIBRATION_FILE=
! /SUFFIX=
! /RANGE=ALL_FRAMES
! /SET_PROMPT=Open standing file
;
The kettlebell model was used, found here: T:\AEJ\Models
Pipeline_Breakpoint
/PAUSE_MESSAGE= Apply Model Template
;
Further documentation of this command can be found at: https://www.c-motion.com/v3dwiki/index.php?title=Apply_Model_Template
Apply_Model_Template
! /CALIBRATION_FILE=
! /MODEL_TEMPLATE=
! /SET_PROMPT=Open model file
! /VIEW_BUILDMODEL_RESULTS=2
;
Subject height and mass were read from the shared Ankle Drop Study Google Sheet
Set_Subject_Height
! /CALIBRATION_FILE=
! /HEIGHT=
! /UNITS=m
;
Set_Subject_Mass
! /CALIBRATION_FILE=
! /WEIGHT=
! /UNITS=Kg
;
All MVC, ID, IPD, and NW were loaded into the workspace.
Pipeline_Breakpoint
/PAUSE_MESSAGE= Open Files
;

```
Opens the file that is selected by specifying the file and path. Further documentation:
https://www.c-motion.com/v3dwiki/index.php?title=File_Open
File_Open
! /FILE_NAME=
! /SUFFIX=
! /SET_PROMPT=File_Open
! /FILTER=
;
Assigns the model to the specified movement files. Further documentation:
https://www.c-motion.com/v3dwiki/index.php?title=Assign_Model_File
Assign_Model_File
! /CALIBRATION_FILE=
! /MOTION_FILE_NAMES=
! /REMOVE_EXISTING_ASSIGNMENTS=FALSE
;
The next part is a Tagging pipeline where the name of the file is read and tagged appropriately.
These tags include: ID (inversion drop trial), IPD (inversion plantarflexion drop trial), NW
(normal walking trial), Movement (all walking or drop trials), MVC (maximum voluntary
contraction file), and MVC_Muscle (specific maximum voluntary contraction files for each
muscle). Further documentation:
https://www.c-motion.com/v3dwiki/index.php?title=Assign_Tags_To_File
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*NW*
! /QUERY=
/TAGS=NW + Movement
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Rec_Fem*
! /QUERY=
/TAGS=MVC_Rec_Fem + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*IPD*
! /QUERY=
/TAGS=IPD + Movement
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*ID*
! /QUERY=
/TAGS=ID + Movement
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*Wash*
! /QUERY=
/TAGS=Wash + Movement
;
```

```
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Contralateral_Glut_Med*
! /QUERY=
/TAGS=MVC_Cont_Glut_Med + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Glut_Max*
! /QUERY=
/TAGS=MVC_Glute_Max + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Ipsilateral_Glut_Med*
! /QUERY=
/TAGS=MVC_Ipsi_Glut_Med + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Bicepss_Fem*
! /QUERY=
/TAGS=MVC_Bicepss_Fem + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Peroneus_Longus*
! /QUERY=
/TAGS=MVC_Peroneus_Longus + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Soleus*
! /QUERY=
/TAGS=MVC_Soleus + MVC
;
Assign_Tags_To_Files
/MOTION_FILE_NAMES=*MVC_Tib_Anterior*
! /QUERY=
/TAGS=MVC_Tib_Anterior + MVC
;
```

The next few pipeline commands are renaming the voltage signal channels to the corresponding muscle they were recording. Further documentation: https://www.c-motion.com/v3dwiki/index.php?title=Rename_Signals

```
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 1
/NEW_SIGNAL_NAME=Tibialis Anterior
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
```

```
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 2
/NEW_SIGNAL_NAME=Peroneus Longus
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 3
/NEW_SIGNAL_NAME=Bicepss Femoris
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 4
/NEW_SIGNAL_NAME=Contra. Gluteus Medius
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 5
/NEW_SIGNAL_NAME=Soleus
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 6
/NEW_SIGNAL_NAME=Rectus Femoris
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
```

```
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 7
/NEW_SIGNAL_NAME=Ipsilateral Gluteus Medius
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.Channel 8
/NEW_SIGNAL_NAME=Gluteus Max
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
```

The next two commands rename the analog switch to the specific platform they record.

```
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.ID
/NEW_SIGNAL_NAME=ID Switch
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
Rename_Signals
! /FILE_NAME=
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Voltage.IPD
/NEW_SIGNAL_NAME=IPD Switch
! /APPEND_TO_OLD_SIGNAL_NAME=FALSE
! /INCLUDE_CALFILE=FALSE
;
```

The next part includes a filter pipeline, where the according signals are filtered using the specifications set.

```
Select_Active_File
/FILE_NAME=ALL_FILES
! /QUERY=
;
Lowpass_Filter
/SIGNAL_TYPES=TARGET
! /SIGNAL_FOLDER=ORIGINAL
```

! /SIGNAL_NAMES=
! /RESULT_FOLDER=PROCESSED
! /RESULT_SUFFIX=
! /FILTER_CLASS=BUTTERWORTH
! /FREQUENCY_CUTOFF=6.0
! /NUM_REFLECTED=6
! /NUM_EXTRAPOLATED=0
! /TOTAL_BUFFER_SIZE=6
! /NUM_BIDIRECTIONAL_PASSES=1
;
This is using a lowpass filter with a cutoff frequency of 300Hz on the signal recorded from each muscle. Since it is a lowpass filter it passes signals below this cutoff frequency. This helps to get rid of extra noise outside this expected signal range of 300Hz. Further documentation of the filter pipeline commands can be found here:
https://www.c-motion.com/v3dwiki/index.php?title=Lowpass_Filter
Lowpass_Filter
/SIGNAL_TYPES=ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG
/SIGNAL_FOLDER=ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL
/SIGNAL_NAMES=Contra. Gluteus Medius+Bicepss Femoris+Ipsilateral Gluteus Medius+Rectus Femoris+Peroneus Longus+Soleus+Tibialis Anterior+Gluteus Max
! /RESULT_FOLDER=PROCESSED
! /RESULT_SUFFIX=
! /FILTER_CLASS=BUTTERWORTH
/FREQUENCY_CUTOFF=300
! /NUM_REFLECTED=6
! /NUM_EXTRAPOLATED=0
! /TOTAL_BUFFER_SIZE=6
! /NUM_BIDIRECTIONAL_PASSES=1
;
Select_Active_File
/FILE_NAME=Movement
! /QUERY=
;
Creating gait events based on the calculation of foot velocity relative to the pelvis. Further documentation of this method:
https://www.c-motion.com/v3dwiki/index.php?title=Events:Example_6
Compute_Model_Based_Data
/RESULT_NAME=RHEEL_WRT_PELVIS
/FUNCTION=SEG_PROXIMAL_JOINT
/SEGMENT=RFT
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE

```
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=LHEEL_WRT_PELVIS
/FUNCTION=SEG_PROXIMAL_JOINT
/SEGMENT=LFT
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=RTOE_WRT_PELVIS
/FUNCTION=SEG_DISTAL_JOINT
/SEGMENT=RFT
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
```

```
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=LTOE_WRT_PELVIS
/FUNCTION=SEG_DISTAL_JOINT
/SEGMENT=LFT
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
First_Derivative
/SIGNAL_TYPES=LINK_MODEL_BASED+LINK_MODEL_BASED+LINK_MODEL_BAS
ED+LINK_MODEL_BASED
/SIGNAL_FOLDER=ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL
/SIGNAL_NAMES=LHeel_Wrt_Pelvis+LToe_Wrt_Pelvis+RHeel_Wrt_Pelvis+RToe_Wrt_Pelv
is
/RESULT_TYPES=DERIVED
/RESULT_FOLDERS=EVENTS
/RESULT_NAME=_Vel
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
;
Here an event is created for right heel strike (RHS). Below, similar commands are used to create
left heel strike event (LHS), right toe off (RTO), and left toe off events (LTO).
Event_Threshold
/RESULT_EVENT_NAME=RHS
/SIGNAL_TYPES=DERIVED
/SIGNAL_FOLDER=EVENTS
/SIGNAL_NAMES=RHeel_Wrt_Pelvis_Vel
/SIGNAL_COMPONENTS=Y
! /FRAME_OFFSET=0
```

```
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD=0
/ON_ASCENT=FALSE
/ON_DESCENT=TRUE
! /FRAME_WINDOW=8
/ENSURE_FRAMES_BEFORE=TRUE
/ENSURE_FRAMES_AFTER=TRUE
;
Event_Threshold
/RESULT_EVENT_NAME=LHS
/SIGNAL_TYPES=DERIVED
/SIGNAL_FOLDER=EVENTS
/SIGNAL_NAMES=LHeel_Wrt_Pelvis_Vel
/SIGNAL_COMPONENTS=Y
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD=0
/ON_ASCENT=FALSE
/ON_DESCENT=TRUE
! /FRAME_WINDOW=8
/ENSURE_FRAMES_BEFORE=TRUE
/ENSURE_FRAMES_AFTER=TRUE
;
Event_Threshold
/RESULT_EVENT_NAME=RTO
/SIGNAL_TYPES=DERIVED
/SIGNAL_FOLDER=EVENTS
/SIGNAL_NAMES=RToe_Wrt_Pelvis_Vel
/SIGNAL_COMPONENTS=Y
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
```

```
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD=0
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
/ENSURE_FRAMES_BEFORE=TRUE
/ENSURE_FRAMES_AFTER=TRUE
;
Event_Threshold
/RESULT_EVENT_NAME=LTO
/SIGNAL_TYPES=DERIVED
/SIGNAL_FOLDER=EVENTS
/SIGNAL_NAMES=LToe_Wrt_Pelvis_Vel
/SIGNAL_COMPONENTS=Y
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD=0
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
/ENSURE_FRAMES_BEFORE=TRUE
/ENSURE_FRAMES_AFTER=TRUE
;
The next series of commands is considered the drop event pipeline as it creates events for the ID
and IPD drops based off the analog signal from the switch.
For_Each
/ITERATION_PARAMETER_NAME=ID_DROP
! /ITERATION_PARAMETER_COUNT_NAME=
/ITEMS=ID
;
Select_Active_File
/FILE_NAME=ID
! /QUERY=
;
```

Further documenation: https://www.c-motion.com/v3dwiki/index.php?title=Moving_RMS
Moving_RMS
/SIGNAL_TYPES=ANALOG
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=ID Switch
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
! /RESULT_NAME=
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
/NUM_WINDOW_FRAMES=25
;
Event_Explicit
/EVENT_NAME=Rest1
/FRAME=1
! /TIME=
;
Event_Explicit
/EVENT_NAME=Rest2
/FRAME=50
! /TIME=
;
Computes the maximum value of the signal and stores as a metric value:
https://c-motion.com/v3dwiki/index.php/Metric_Maximum
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_MAX
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=ID Switch
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Rest1+Rest2
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Computes the standard deviation of the signal and stores it as a metric value: https://www.c-motion.com/v3dwiki/index.php?title=Metric_StdDev
Metric_StdDev
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_SD
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED

/SIGNAL_NAMES=ID Switch
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Rest1+Rest2
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Places an event label at the frame where a signal crosses the specified value: https://www.c-motion.com/v3dwiki/index.php?title=Event_Threshold
Creating an event called ID_Drop that happens when the signal passes this threshold of the maximum value plus three standard deviations. A similar series of commands is used to create an event called IPD_Drop.
Event_Threshold
/RESULT_EVENT_NAME=ID_Drop
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=ID Switch
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::ID Switch_Max+3*METRIC::PROCESSED::ID Switch_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
/ENSURE_FRAMES_AFTER=TRUE
;
End_For_Each
/ITERATION_PARAMETER_NAME=ID_DROP
;
Event_Threshold
/RESULT_EVENT_NAME=IPD_Drop
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=IPD Switch
/SIGNAL_COMPONENTS=ALL

```
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /EVENT_SEQUENCE_INSTANCE=0
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
! /EVENT_SUBSEQUENCE_INSTANCE=0
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::IPD Switch_Max+3*METRIC::PROCESSED::IPD
Switch_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
/ENSURE_FRAMES_AFTER=TRUE
;
End_For_Each
/ITERATION_PARAMETER_NAME=IPD_DROP
;
Here begins 8 on Part 2
This is considered the drop event part of the pipeline. It labels what is considered the true drop.
Select_Active_File

/FILE_NAME=Movement
! /QUERY=
;
Event_Explicit
/EVENT_NAME=END
/FRAME=EOF
! /TIME=
;
This command creates an event between two events that already exist and were created. The
event sequence specifies that this new event will be created between the right heel strike and the
ID Drop event. A similar command sequence is used to create the True IPD drop. Further
documentation: https://www.c-motion.com/v3dwiki/index.php?title=Event_Between
Event_Between
/NEW_EVENT_NAME=TRUE_ID_DROP
/RANGE_INSTANCE=0
/EVENT_SEQUENCE=RHS+ID_DROP
! /EXCLUDE_EVENTS=
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /PERCENT_OFFSET=
;
Event_Between
```

```
/NEW_EVENT_NAME=TRUE_IPD
/RANGE_INSTANCE=0
/EVENT_SEQUENCE=RHS+IPD_DROP
! /EXCLUDE_EVENTS=
! /FRAME_OFFSET=0
! /TIME_OFFSET=
! /PERCENT_OFFSET=
;
```

The following command create events that mark time points following and prior to the true drop events. These are created 150, 200, 250, and 300 milliseconds before and after.

```
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=Pre_ID_150
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.150
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=Pre_ID_200
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.200
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=Pre_ID_250
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
```

```
/TIME_OFFSET=-.250
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=Pre_ID_300
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.300
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=POST_ID_150
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.150
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=POST_ID_200
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.200
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=POST_ID_250
! /EVENT_INSTANCE=0
```

```
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.250
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=POST_ID_300
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.300
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_ID_DROP
/NEW_EVENT_NAME=POST_ID_350
/EVENT_INSTANCE=1
/RANGE_INSTANCE=1
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.350
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=Pre_IPD_150
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.150
```

```
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=Pre_IPD_200
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.200
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=Pre_IPD_250
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.250
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=Pre_IPD_300
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.300
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=POST_IPD_150
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
```

```
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.150
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=POST_IPD_200
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.200
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=POST_IPD_250
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.250
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=POST_IPD_300
! /EVENT_INSTANCE=0
! /RANGE_INSTANCE=0
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.300
! /PERCENT_OFFSET=
```

```
;
Event_Copy
/EVENT_NAME=TRUE_IPD
/NEW_EVENT_NAME=POST_IPD_350
/EVENT_INSTANCE=1
/RANGE_INSTANCE=1
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.350
! /PERCENT_OFFSET=
;
```

The following section is considered the EMG MVC pipeline. This includes processing the signals from these MVC trials and the finding the average peak value for each muscle to store that can be referenced later for normalization of EMG trial data.

```
Select_Active_File
/FILE_NAME=MVC
! /QUERY=
;
```

Takes the processed signal and computes the root mean square of the signal for each frame with a moving window of 25 centered around the current frame. Further documentation: www.https://c-motion.com/v3dwiki/index.php/Moving_RMS

```
Moving_RMS
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
! /SIGNAL_NAMES=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_RMS
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/NUM_WINDOW_FRAMES=25
;
Select_Active_File
/FILE_NAME=Movement
! /QUERY=
;
Moving_RMS
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
! /SIGNAL_NAMES=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
! /RESULT_NAME=
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
```

```
/NUM_WINDOW_FRAMES=25
;
For_Each
/ITERATION_PARAMETER_NAME=MVC_Frames
! /ITERATION_PARAMETER_COUNT_NAME=
/ITEMS=MVC
;
Select_Active_File
/FILE_NAME=MVC
! /QUERY=
;
Computes the maximum value of the signal and stores it as a metric value. Further
documentation: https://www.c-motion.com/v3dwiki/index.php?title=Metric_Maximum
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=Total_Frames
! /APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
/SIGNAL_TYPES=FRAME_NUMBERS
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=FRAMES
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Event_Explicit
/EVENT_NAME=Beginning
/FRAME=1
! /TIME=
;
Creates an event after the first burst of MVC EMG signal. A similar event is created after the
second break.
Event_Explicit
/EVENT_NAME=First_Break
/FRAME=(1/3)*METRIC::PROCESSED::Total_Frames
! /TIME=
;
Event_Explicit
/EVENT_NAME=Second_Break
/FRAME=(2/3)*METRIC::PROCESSED::Total_Frames
! /TIME=
;
```

```
Event_Explicit
/EVENT_NAME=End
/FRAME=METRIC::PROCESSED::Total_Frames
! /TIME=
;
End_For_Each
/ITERATION_PARAMETER_NAME=MVC_Frames
;
Select_Active_File
/FILE_NAME=MVC_Peroneus_Longus
! /QUERY=
;
```

Determines the maximum value for each burst by using the events created to tell V3D to find the max between each of these events for each MVC EMG muscle file. The next commands do this for the other two bursts.

```
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
```

```
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Adding together the maximum values that were computed above.
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus_RMS_Peak1+Peroneus Longus_RMS_Peak2+Peroneus
Longus_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Taking the sum value and dividing it by the three to find the average maximum value from the
MVC EMG signal.
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
Select_Active_File
/FILE_NAME=MVC_Cont_Glut_Med
! /QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
```

```
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
```

```
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius_RMS_Peak1+Contra. Gluteus
Medius_RMS_Peak2+Contra. Gluteus Medius_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
Select_Active_File
/FILE_NAME=MVC_Ipsi_Glut_Med
! /QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Ipsilateral Gluteus Medius_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
```

```
/SIGNAL_NAMES=Ipsilateral Gluteus Medius_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Ipsilateral Gluteus Medius_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Ipsilateral Gluteus Medius_RMS_Peak1+Ipsilateral Gluteus
Medius_RMS_Peak2+Ipsilateral Gluteus Medius_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
```

```
Select_Active_File
/FILE_NAME=MVC_Soleus
! /QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
```

```
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus_RMS_Peak1+Soleus_RMS_Peak2+Soleus_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
Select_Active_File
/FILE_NAME=MVC_Glute_Max
! /QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
```

```
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max_RMS_Peak1+Gluteus Max_RMS_Peak2+Gluteus
Max_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
```

```
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
Select_Active_File
/FILE_NAME=MVC_Bicepss_Fem
! /QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
```

```
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris_RMS_Peak1+Bicepss Femoris_RMS_Peak2+Bicepss
Femoris_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
Select_Active_File
/FILE_NAME=MVC_Tib_Anterior
! /QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
```

```
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior_RMS_Peak1+Tibialis Anterior_RMS_Peak2+Tibialis
Anterior_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
```

```
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
Select_Active_File
/FILE_NAME=MVC_Rec_Fem
!/QUERY=
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak1
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Beginning+First_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak2
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=First_Break+Second_Break
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
```

```
;
Metric_Maximum
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_Peak3
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris_RMS
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Second_Break+End
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
! /CREATE_GLOBAL_MAXIMUM=FALSE
;
Add_Signals
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris_RMS_Peak1+Rectus Femoris_RMS_Peak2+Rectus
Femoris_RMS_Peak3
! /COMPONENT_SEQUENCE=
! /RESULT_FOLDER=PROCESSED
/RESULT_NAME=Peak_Sum
;
Divide_Signal_By_Constant
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum
! /SIGNAL_COMPONENTS=
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
/RESULT_NAME=_MVC
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/CONSTANT=3
;
For_Each
/ITERATION_PARAMETER_NAME=Movement_Files
! /ITERATION_PARAMETER_COUNT_NAME=
/ITEMS=MOVEMENT
;
The following portion is considered the kinematics pipeline. Here the kinematics are computed
for each lower extremity joint in all three planes of motion. This is done for each of the
movement trials.
Select_Active_File
```

```
/FILE_NAME=::Movement_Files
! /QUERY=
;
Compute_Model_Based_Data
/RESULT_NAME=L Ankle Angle
/FUNCTION=JOINT_ANGLE
/SEGMENT=LFT
/REFERENCE_SEGMENT=LSK
/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
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=L Knee Angle
/FUNCTION=JOINT_ANGLE
/SEGMENT=LSK
/REFERENCE_SEGMENT=LTH
/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
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=L Hip Angle
/FUNCTION=JOINT_ANGLE
```

```
/SEGMENT=LTH
/REFERENCE_SEGMENT=RPV
/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
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R Ankle Angle
/FUNCTION=JOINT_ANGLE
/SEGMENT=RFT
/REFERENCE_SEGMENT=RSK
/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
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R Knee Angle
/FUNCTION=JOINT_ANGLE
/SEGMENT=RSK
/REFERENCE_SEGMENT=RTH
/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
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R Hip Angle
/FUNCTION=JOINT_ANGLE
/SEGMENT=RTH
/REFERENCE_SEGMENT=RPV
/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
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
**__Add_a_Comment__**
/COMMENT= velocity
;
Here the velocity for each joint is calculated. The resolution coordinate system is typically the
segment proximal to the joint.
Compute_Model_Based_Data
/RESULT_NAME=L_Ankle_Velocity
/FUNCTION=JOINT_VELOCITY
/SEGMENT=LFT
/REFERENCE_SEGMENT=LSK
/RESOLUTION_COORDINATE_SYSTEM=LSK
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
```

```
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R_Ankle_Velocity
/FUNCTION=JOINT_VELOCITY
/SEGMENT=RFT
/REFERENCE_SEGMENT=RSK
/RESOLUTION_COORDINATE_SYSTEM=RSK
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=L_Knee_Velocity
/FUNCTION=JOINT_VELOCITY
/SEGMENT=LSK
/REFERENCE_SEGMENT=LTH
/RESOLUTION_COORDINATE_SYSTEM=LTH
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
```

```
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R_Knee_Velocity
/FUNCTION=JOINT_VELOCITY
/SEGMENT=RSK
/REFERENCE_SEGMENT=RTH
/RESOLUTION_COORDINATE_SYSTEM=RTH
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=L_Hip_Velocity
/FUNCTION=JOINT_VELOCITY
/SEGMENT=LTH
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R_Hip_Velocity
```

```
/FUNCTION=JOINT_VELOCITY
/SEGMENT=RTH
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
**__Add_a_Comment__**
/COMMENT=Accel
;
The acceleration data is computed but not exported.
Compute_Model_Based_Data
/RESULT_NAME=L_Hip_Accel
/FUNCTION=JOINT_ACCELERATION
/SEGMENT=LTH
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R_Hip_Accel
/FUNCTION=JOINT_ACCELERATION
/SEGMENT=RTH
```

```
/REFERENCE_SEGMENT=RPV
/RESOLUTION_COORDINATE_SYSTEM=RPV
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=L_Knee_Accel
/FUNCTION=JOINT_ACCELERATION
/SEGMENT=LSK
/REFERENCE_SEGMENT=LTH
/RESOLUTION_COORDINATE_SYSTEM=LTH
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R_Knee_Accel
/FUNCTION=JOINT_ACCELERATION
/SEGMENT=RSK
/REFERENCE_SEGMENT=RTH
/RESOLUTION_COORDINATE_SYSTEM=RTH
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
```

```
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=L_Ankle_Accel
/FUNCTION=JOINT_ACCELERATION
/SEGMENT=LFT
/REFERENCE_SEGMENT=LSK
/RESOLUTION_COORDINATE_SYSTEM=LSK
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
Compute_Model_Based_Data
/RESULT_NAME=R_Ankle_Accel
/FUNCTION=JOINT_ACCELERATION
/SEGMENT=RFT
/REFERENCE_SEGMENT=RSK
/RESOLUTION_COORDINATE_SYSTEM=RSK
! /USE_CARDAN_SEQUENCE=FALSE
! /NORMALIZATION=FALSE
! /NORMALIZATION_METHOD=
! /NORMALIZATION_METRIC=
! /NEGATEX=FALSE
! /NEGATEY=FALSE
! /NEGATEZ=FALSE
! /AXIS1=X
! /AXIS2=Y
! /AXIS3=Z
```

```
! /TREADMILL_DATA=FALSE
! /TREADMILL_DIRECTION=UNIT_VECTOR(0,1,0)
! /TREADMILL_SPEED=0.0
;
End_For_Each
/ITERATION_PARAMETER_NAME=Movement_Files
;
The next portion is considered part of the reaction time pipeline. Here the mean between 150ms
before the true drop event and the drop event for each EMG signal is calculated.
Metric_Mean
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_MEAN
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus
Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Pre_ID_150+TRUE_ID_DROP
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Here the standard deviation between 150ms before the true drop event and the drop event for
each EMG signal is calculated.
Metric_Mean
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_MEAN_IPD
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus
Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Pre_IPD_150+TRUE_IPD
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_StdDev
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_SD
```

/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Pre_ID_150+TRUE_ID_DROP
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_StdDev
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_SD_IPD
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Pre_IPD_150+TRUE_IPD
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
The event threshold for the reaction onset is defined as 2 standard deviations above the mean value during the 150ms prior to the true drop event. This value is searched for starting at the time of the drop to the end of the trial. A new event is created at the onset for each muscle during each trial.
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_BicepsFem
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=

```
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Bicepss Femoris_MEAN+2 *
METRIC::PROCESSED::Bicepss Femoris_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_ContraGM
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Contra. Gluteus Medius_MEAN+2 *
METRIC::PROCESSED::Contra. Gluteus Medius_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_GluteMax
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
```

```
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Gluteus Max_MEAN+2 *
METRIC::PROCESSED::Gluteus Max_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_IpsiGM
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Ipsilateral Gluteus Medius
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Ipsilateral Gluteus Medius_MEAN+2 *
METRIC::PROCESSED::Ipsilateral Gluteus Medius_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_PL
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
```

```
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Peroneus Longus_MEAN+2 *
METRIC::PROCESSED::Peroneus Longus_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_Soleus
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Soleus_MEAN+2 *
METRIC::PROCESSED::Soleus_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;

Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_RecFem
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
```

```
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Rectus Femoris_MEAN+2 *
METRIC::PROCESSED::Rectus Femoris_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_TibAnt
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_ID_DROP+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Tibialis Anterior_MEAN+2 *
METRIC::PROCESSED::Tibialis Anterior_SD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_BicepsFem_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
```

```
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Bicepss Femoris_MEAN_IPD+2 *
METRIC::PROCESSED::Bicepss Femoris_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_ContraGM_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Contra. Gluteus Medius_MEAN_IPD+2 *
METRIC::PROCESSED::Contra. Gluteus Medius_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_GluteMax_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
```

```
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Gluteus Max_MEAN_IPD+2 *
METRIC::PROCESSED::Gluteus Max_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_IpsiGM_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Ipsilateral Gluteus Medius
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Ipsilateral Gluteus Medius_MEAN_IPD+2 *
METRIC::PROCESSED::Ipsilateral Gluteus Medius_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_PL_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
```

```
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Peroneus Longus_MEAN_IPD+2 *
METRIC::PROCESSED::Peroneus Longus_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_Soleus_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Soleus_MEAN_IPD+2 *
METRIC::PROCESSED::Soleus_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_RecFem_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
```

```
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Rectus Femoris_MEAN_IPD+2 *
METRIC::PROCESSED::Rectus Femoris_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_TibAnt_IPD
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE=TRUE_IPD+ END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Tibialis Anterior_MEAN_IPD+2 *
METRIC::PROCESSED::Tibialis Anterior_SD_IPD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Here the reaction time is calculated between the true drop event and the reaction onset for each
muscle during each trial. The calculated value is stored in the metric folder.
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_BicepsFem
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_BicepsFem
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_ContraGM
```

```
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_ContraGM
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_GluteMax
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_GluteMax
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_IpsiGM
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_IpsiGM
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_PL
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_PL
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_RecFem
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_RecFem
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_Soleus
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_Soleus
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
```

```
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_TibAnt
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_ID_DROP+REACTION_ONSET_TibAnt
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_BicepsFem_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_BicepsFem_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_ContraGM_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_ContraGM_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_GluteMax_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_GluteMax_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_IpsiGM_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_IpsiGM_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_PL_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_PL_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
```

```
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_RecFem_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_RecFem_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_Soleus_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_Soleus_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_TibAnt_IPD
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE=TRUE_IPD+REACTION_ONSET_TibAnt_IPD
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Select_Active_File
/FILE_NAME=NW
! /QUERY=
;
Moving_RMS
/SIGNAL_TYPES=ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANA
LOG+ANALOG
/SIGNAL_FOLDER=ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGI
NAL+ORIGINAL+ORIGINAL
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus
Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
! /RESULT_TYPES=
! /RESULT_FOLDERS=PROCESSED
! /RESULT_NAME=
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=FALSE
/NUM_WINDOW_FRAMES=25
;
Select_Active_File
/FILE_NAME=NW
! /QUERY=
```

```
;
Event_Copy
/EVENT_NAME=RHS
/NEW_EVENT_NAME=PRE_HS_150
/EVENT_INSTANCE=1
/RANGE_INSTANCE=1
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=-.150
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=RHS
/NEW_EVENT_NAME=POST_HS_200
/EVENT_INSTANCE=1
/RANGE_INSTANCE=1
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.200
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=RHS
/NEW_EVENT_NAME=POST_HS_350
/EVENT_INSTANCE=1
/RANGE_INSTANCE=1
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=.350
! /PERCENT_OFFSET=
;
Event_Copy
/EVENT_NAME=RHS
/NEW_EVENT_NAME=REAL_RHS
/EVENT_INSTANCE=1
/RANGE_INSTANCE=1
/EVENT_SEQUENCE=PRE_HS_150+POST_HS_200
```

! /EXCLUDE_EVENTS=
! /START_AT_EVENT=
! /END_AT_EVENT=
! /FRAME_OFFSET=
/TIME_OFFSET=0
! /PERCENT_OFFSET=
;
The reaction time pipeline for the normal walking trials is very similar but uses the appropriate heel strike event instead of the true drop event.
Metric_Mean
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_NWMEAN
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG
/SIGNAL_FOLDER=PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Pre_HS_150+ REAL_RHS
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_StdDev
! /RESULT_METRIC_FOLDER=PROCESSED
/RESULT_METRIC_NAME=_NWSD
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
/SIGNAL_TYPES=ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG
/SIGNAL_FOLDER=PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED+PROCESSED
/SIGNAL_NAMES=Bicepss Femoris+Contra. Gluteus Medius+Gluteus Max+Ipsilateral Gluteus Medius+Peroneus Longus+Rectus Femoris+Soleus+Tibialis Anterior
/COMPONENT_SEQUENCE=ALL
/EVENT_SEQUENCE=Pre_HS_150+ REAL_RHS
/EXCLUDE_EVENTS=
/SEQUENCE_PERCENT_START=
/SEQUENCE_PERCENT_END=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Event_Threshold

```
/RESULT_EVENT_NAME=REACTION_ONSET_BicepsFem_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Bicepss Femoris
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Bicepss Femoris_NWMEAN+2 *
METRIC::PROCESSED::Bicepss Femoris_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_ContraGM_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Contra. Gluteus Medius
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Contra. Gluteus Medius_NWMEAN+2 *
METRIC::PROCESSED::Contra. Gluteus Medius_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
```

```
/RESULT_EVENT_NAME=REACTION_ONSET_GluteMax_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Gluteus Max
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Gluteus Max_NWMEAN+2 *
METRIC::PROCESSED::Gluteus Max_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_IpsiGM_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Ipsilateral Gluteus Medius
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Ipsilateral Gluteus Medius_NWMEAN+2 *
METRIC::PROCESSED::Ipsilateral Gluteus Medius_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
```

```
/RESULT_EVENT_NAME=REACTION_ONSET_PL_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peroneus Longus
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Peroneus Longus_NWMEAN+2 *
METRIC::PROCESSED::Peroneus Longus_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_Soleus_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Soleus
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Soleus_NWMEAN+2 *
METRIC::PROCESSED::Soleus_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
Event_Threshold
```

```
/RESULT_EVENT_NAME=REACTION_ONSET_RecFem_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rectus Femoris
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Rectus Femoris_NWMEAN+2 *
METRIC::PROCESSED::Rectus Femoris_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;

Event_Threshold
/RESULT_EVENT_NAME=REACTION_ONSET_TibAnt_NW
/SIGNAL_TYPES=ANALOG
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Tibialis Anterior
/SIGNAL_COMPONENTS=ALL
! /FRAME_OFFSET=0
! /TIME_OFFSET=
/EVENT_SEQUENCE= REAL_RHS + END
! /EXCLUDE_EVENTS=
/EVENT_SEQUENCE_INSTANCE=1
! /EVENT_SUBSEQUENCE=
! /SUBSEQUENCE_EXCLUDE_EVENTS=
/EVENT_SUBSEQUENCE_INSTANCE=1
! /EVENT_INSTANCE=0
/THRESHOLD=METRIC::PROCESSED::Tibialis Anterior_NWMEAN+2 *
METRIC::PROCESSED::Tibialis Anterior_NWSD
/ON_ASCENT=TRUE
/ON_DESCENT=FALSE
! /FRAME_WINDOW=8
! /ENSURE_FRAMES_BEFORE=FALSE
! /ENSURE_FRAMES_AFTER=FALSE
;
```

```
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_BicepsFem_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_BicepsFem_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_ContraGM_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_ContraGM_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_GluteMax_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_GluteMax_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_IpsiGM_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_IpsiGM_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_PL_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_PL_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_RecFem_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_RecFem_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
```

```
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_Soleus_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_Soleus_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Metric_Time_Between_Events
/RESULT_METRIC_NAME=Rxn_Time_TibAnt_NW
! /RESULT_METRIC_FOLDER=PROCESSED
/EVENT_SEQUENCE= REAL_RHS +REACTION_ONSET_TibAnt_NW
/EXCLUDE_EVENTS=
/GENERATE_MEAN_AND_STDDEV=FALSE
! /APPEND_TO_EXISTING_VALUES=FALSE
;
Below are the export commands. This pipeline is saved separately. This exports the data to a text file to be opened into excel or read into Matlab.
Pipeline_Breakpoint
/PAUSE_MESSAGE=Rename Export Filename
;
Exporting the MVC values for each muscle so that trial data can be normalized.
Export_Data_To_Ascii_File
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Peak_Sum_MVC
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
/EXPORT_WITHOUT_HEADER=TRUE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
```

```
Exporting the event labels to be read into Matlab to specifiy event windows for the appropriate
variables.
Export_Data_To_Ascii_File
/SIGNAL_TYPES=EVENT_LABEL
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=Pre_ID_150+TRUE_ID_DROP+POST_ID_150+POST_ID_200+POST_ID
_250+POST_ID_300+POST_ID_350+Pre_IPD_150+TRUE_IPD+POST_IPD_150+POST_IPD
_200+POST_IPD_250+POST_IPD_300+POST_IPD_350+PRE_HS_150+RHS+POST_HS_200
+POST_HS_350
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
/EXPORT_WITHOUT_HEADER=TRUE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Exporting the kinematic data from each movement trial.
Export_Data_To_Ascii_File
/SIGNAL_TYPES=FRAME_NUMBERS+LINK_MODEL_BASED+LINK_MODEL_BASED+
LINK_MODEL_BASED+LINK_MODEL_BASED+LINK_MODEL_BASED+LINK_MODEL
_BASED+LINK_MODEL_BASED+LINK_MODEL_BASED+LINK_MODEL_BASED+LIN
K_MODEL_BASED+LINK_MODEL_BASED+LINK_MODEL_BASED
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=TIME+L Ankle Angle+L_Ankle_Velocity+L Knee
Angle+L_Knee_Velocity+L Hip Angle+L_Hip_Velocity+R Ankle Angle+R_Ankle_Velocity+R
Knee Angle+R_Knee_Velocity+R Hip Angle+R_Hip_Velocity
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
```

```
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
/EXPORT_WITHOUT_HEADER=TRUE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Select_Active_File
/FILE_NAME= Movement
! /QUERY=
;
Exporting the EMG signal from each muscle for each movement trial.
Export_Data_To_Ascii_File
/SIGNAL_TYPES=FRAME_NUMBERS+ANALOG+ANALOG+ANALOG+ANALOG+ANA
LOG+ANALOG+ANALOG+ANALOG
/SIGNAL_FOLDER=ORIGINAL+PROCESSED+PROCESSED+PROCESSED+PROCESSED
+PROCESSED+PROCESSED+PROCESSED+PROCESSED
/SIGNAL_NAMES=TIME+Contra. Gluteus Medius+Ipsilateral Gluteus Medius+Tibialis
Anterior+Soleus+Peroneus Longus+Bicepss Femoris+Gluteus Max+Rectus Femoris
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
/EXPORT_EMPTY_SIGNALS=TRUE
/EXPORT_WITHOUT_HEADER=TRUE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Next the event labels for all of the reaction onsets as well as the calculated reaction times are
exported for each movement trial.
Select_Active_File
/FILE_NAME= ID
! /QUERY=
;
Export_Data_To_Ascii_File
```

```
/SIGNAL_TYPES=EVENT_LABEL
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=REACTION_ONSET_BicepsFem+REACTION_ONSET_ContraGM+RE
ACTION_ONSET_GluteMax+REACTION_ONSET_IpsiGM+REACTION_ONSET_PL+REA
CTION_ONSET_RecFem+REACTION_ONSET_Soleus+REACTION_ONSET_TibAnt
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
! /EXPORT_WITHOUT_HEADER=FALSE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Export_Data_To_Ascii_File
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rxn_Time_BicepsFem+Rxn_Time_ContraGM+Rxn_Time_GluteMax+Rx
n_Time_IpsiGM+Rxn_Time_PL+Rxn_Time_RecFem+Rxn_Time_Soleus+Rxn_Time_TibAnt
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
! /EXPORT_WITHOUT_HEADER=FALSE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
```

```
Select_Active_File
/FILE_NAME= IPD
! /QUERY=
;
Export_Data_To_Ascii_File
/SIGNAL_TYPES=EVENT_LABEL
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=REACTION_ONSET_BicepsFem_IPD+REACTION_ONSET_ContraGM
_IPD+REACTION_ONSET_GluteMax_IPD+REACTION_ONSET_IpsiGM_IPD+REACTION
_ONSET_PL_IPD+REACTION_ONSET_RecFem_IPD+REACTION_ONSET_Soleus_IPD+R
EACTION_ONSET_TibAnt_IPD
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
! /EXPORT_WITHOUT_HEADER=FALSE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Export_Data_To_Ascii_File
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rxn_Time_BicepsFem_IPD+Rxn_Time_ContraGM_IPD+Rxn_Time_Glut
eMax_IPD+Rxn_Time_IpsiGM_IPD+Rxn_Time_PL_IPD+Rxn_Time_RecFem_IPD+Rxn_Tim
e_Soleus_IPD+Rxn_Time_TibAnt_IPD
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
```

```
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
! /EXPORT_WITHOUT_HEADER=FALSE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Select_Active_File
/FILE_NAME= NW
! /QUERY=
;
Export_Data_To_Ascii_File
/SIGNAL_TYPES=EVENT_LABEL
! /SIGNAL_FOLDER=ORIGINAL
/SIGNAL_NAMES=REACTION_ONSET_BicepsFem_NW+REACTION_ONSET_ContraGM
_NW+REACTION_ONSET_GluteMax_NW+REACTION_ONSET_IpsiGM_NW+REACTION
_ONSET_PL_NW+REACTION_ONSET_RecFem_NW+REACTION_ONSET_Soleus_NW+R
EACTION_ONSET_TibAnt_NW
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
! /EXPORT_WITHOUT_HEADER=FALSE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
Export_Data_To_Ascii_File
/SIGNAL_TYPES=METRIC
/SIGNAL_FOLDER=PROCESSED
/SIGNAL_NAMES=Rxn_Time_BicepsFem_NW+Rxn_Time_ContraGM_NW+Rxn_Time_Glut
eMax_NW+Rxn_Time_IpsiGM_NW+Rxn_Time_PL_NW+Rxn_Time_RecFem_NW+Rxn_Tim
e_Soleus_NW+Rxn_Time_TibAnt_NW
! /FILE_NAME=
! /SIGNAL_COMPONENTS=
/COMPONENT_SEQUENCE=ALL
! /SIGNAL_PRECISION=
```

```
! /EVENT_SEQUENCE=
! /EXCLUDE_EVENTS=
! /USE_POINT_RATE=FALSE
! /NORMALIZE_DATA=FALSE
! /NORMALIZE_POINTS=101
! /EXPORT_MEAN_AND_STD_DEV=FALSE
! /USE_P2D_FORMAT=FALSE
! /USE_XML_FORMAT=FALSE
! /USE_SHORT_FILENAME=FALSE
! /EXPORT_EMPTY_SIGNALS=FALSE
! /EXPORT_WITHOUT_HEADER=FALSE
! /EXPORT_NAN=FALSE
! /USE_SCIENTIFIC_NOTATION=FALSE
;
```

APPENDIX B: MATLAB CODES

Below is the code written for Matlab to calculate the necessary variables. Coding comments are notated in green and begin with a percent sign. Two different programs were created and used. The first one shown was used for EMG data processing and the second on beginning on B-40 was used for kinematic data processing.

```
close all;
clear all;
clc;

% EMG

%Start
cd 'C:\Users\Emilia\Documents\Documents\Grad School\Thesis\Exports\';
% First import the MVCs for each muscle and store it as a matrix
MVCs = dlmread('7_a.txt'); %This is a
% Create variables for each MVC so that we can normalize EMG data
MVC_Biceps_Fem = MVCs(2:2);
MVC_Cont_Glut_Med = MVCs(3:3);
MVC_Glut_Max = MVCs(4:4);
MVC_Ipsi_Glut_Med = MVCs(5:5);
MVC_Peron_Long = MVCs(6:6);
MVC_Rec_Fem = MVCs(7:7);
MVC_Soleus = MVCs(8:8);
MVC_Tib_Ant = MVCs(9:9);


% Then import the EMG from each muscle for each trial as a matrix
EMGs = xlsread('7_d.xlsx');

Time = EMGs(:,2);

ID1_Cont_Glut_Med = EMGs(:,3);
ID1_Ipsi_Glut_Med = EMGs(:,4);
ID1_Tib_Ant = EMGs(:,5);
ID1_Soleus = EMGs(:,6);
ID1_Peron_Long = EMGs(:,7);
ID1_Biceps_Fem = EMGs(:,8);
ID1_Glut_Max = EMGs(:,9);
ID1_Rec_Fem = EMGs(:,10);

ID2_Cont_Glut_Med = EMGs(:,12);
ID2_Ipsi_Glut_Med = EMGs(:,13);
ID2_Tib_Ant = EMGs(:,14);
ID2_Soleus = EMGs(:,15);
```

```
ID2_Peron_Long = EMGs(:,16);
ID2_Biceps_Fem = EMGs(:,17);
ID2_Glut_Max = EMGs(:,18);
ID2_Rec_Fem = EMGs(:,19);

ID3_Cont_Glut_Med = EMGs(:,21);
ID3_Ipsi_Glut_Med = EMGs(:,22);
ID3_Tib_Ant = EMGs(:,23);
ID3_Soleus = EMGs(:,24);
ID3_Peron_Long = EMGs(:,25);
ID3_Biceps_Fem = EMGs(:,26);
ID3_Glut_Max = EMGs(:,27);
ID3_Rec_Fem = EMGs(:,28);

IPD1_Cont_Glut_Med = EMGs(:,30);
IPD1_Ipsi_Glut_Med = EMGs(:,31);
IPD1_Tib_Ant = EMGs(:,32);
IPD1_Soleus = EMGs(:,33);
IPD1_Peron_Long = EMGs(:,34);
IPD1_Biceps_Fem = EMGs(:,35);
IPD1_Glut_Max = EMGs(:,36);
IPD1_Rec_Fem = EMGs(:,37);

IPD2_Cont_Glut_Med = EMGs(:,39);
IPD2_Ipsi_Glut_Med = EMGs(:,40);
IPD2_Tib_Ant = EMGs(:,41);
IPD2_Soleus = EMGs(:,42);
IPD2_Peron_Long = EMGs(:,43);
IPD2_Biceps_Fem = EMGs(:,44);
IPD2_Glut_Max = EMGs(:,45);
IPD2_Rec_Fem = EMGs(:,46);

IPD3_Cont_Glut_Med = EMGs(:,48);
IPD3_Ipsi_Glut_Med = EMGs(:,49);
IPD3_Tib_Ant = EMGs(:,50);
IPD3_Soleus = EMGs(:,51);
IPD3_Peron_Long = EMGs(:,52);
IPD3_Biceps_Fem = EMGs(:,53);
IPD3_Glut_Max = EMGs(:,54);
IPD3_Rec_Fem = EMGs(:,55);

NW1_Cont_Glut_Med = EMGs(:,57);
NW1_Ipsi_Glut_Med = EMGs(:,58);
NW1_Tib_Ant = EMGs(:,59);
NW1_Soleus = EMGs(:,60);
NW1_Peron_Long = EMGs(:,61);
```

```
NW1_Biceps_Fem = EMGs(:,62);
NW1_Glut_Max = EMGs(:,63);
NW1_Rec_Fem = EMGs(:,64);

NW2_Cont_Glut_Med = EMGs(:,66);
NW2_Ipsi_Glut_Med = EMGs(:,67);
NW2_Tib_Ant = EMGs(:,68);
NW2_Soleus = EMGs(:,69);
NW2_Peron_Long = EMGs(:,70);
NW2_Biceps_Fem = EMGs(:,71);
NW2_Glut_Max = EMGs(:,72);
NW2_Rec_Fem = EMGs(:,73);

NW3_Cont_Glut_Med = EMGs(:,75);
NW3_Ipsi_Glut_Med = EMGs(:,76);
NW3_Tib_Ant = EMGs(:,77);
NW3_Soleus = EMGs(:,78);
NW3_Peron_Long = EMGs(:,79);
NW3_Biceps_Fem = EMGs(:,80);
NW3_Glut_Max = EMGs(:,81);
NW3_Rec_Fem = EMGs(:,82);


% Now normalize all the EMG data and store it as it's own matrix
Norm_ID1_Biceps_Fem = ((ID1_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_ID1_Cont_Glut_Med = ((ID1_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_ID1_Glut_Max = ((ID1_Glut_Max/MVC_Glut_Max)*100);
Norm_ID1_Ipsi_Glut_Med = ((ID1_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_ID1_Peron_Long = ((ID1_Peron_Long/MVC_Peron_Long)*100);
Norm_ID1_Rec_Fem = ((ID1_Rec_Fem/MVC_Rec_Fem)*100);
Norm_ID1_Soleus = ((ID1_Soleus/MVC_Soleus)*100);
Norm_ID1_Tib_Ant = ((ID1_Tib_Ant/MVC_Tib_Ant)*100);

Norm_ID2_Biceps_Fem = ((ID2_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_ID2_Cont_Glut_Med = ((ID2_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_ID2_Glut_Max = ((ID2_Glut_Max/MVC_Glut_Max)*100);
Norm_ID2_Ipsi_Glut_Med = ((ID2_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_ID2_Peron_Long = ((ID2_Peron_Long/MVC_Peron_Long)*100);
Norm_ID2_Rec_Fem = ((ID2_Rec_Fem/MVC_Rec_Fem)*100);
Norm_ID2_Soleus = ((ID2_Soleus/MVC_Soleus)*100);
Norm_ID2_Tib_Ant = ((ID2_Tib_Ant/MVC_Tib_Ant)*100);

Norm_ID3_Biceps_Fem = ((ID3_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_ID3_Cont_Glut_Med = ((ID3_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_ID3_Glut_Max = ((ID3_Glut_Max/MVC_Glut_Max)*100);
Norm_ID3_Ipsi_Glut_Med = ((ID3_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
```

```
Norm_ID3_Peron_Long = ((ID3_Peron_Long/MVC_Peron_Long)*100);
Norm_ID3_Rec_Fem = ((ID3_Rec_Fem/MVC_Rec_Fem)*100);
Norm_ID3_Soleus = ((ID3_Soleus/MVC_Soleus)*100);
Norm_ID3_Tib_Ant = ((ID3_Tib_Ant/MVC_Tib_Ant)*100);

Norm_IPD1_Biceps_Fem = ((IPD1_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_IPD1_Cont_Glut_Med = ((IPD1_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_IPD1_Glut_Max = ((IPD1_Glut_Max/MVC_Glut_Max)*100);
Norm_IPD1_Ipsi_Glut_Med = ((IPD1_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_IPD1_Peron_Long = ((IPD1_Peron_Long/MVC_Peron_Long)*100);
Norm_IPD1_Rec_Fem = ((IPD1_Rec_Fem/MVC_Rec_Fem)*100);
Norm_IPD1_Soleus = ((IPD1_Soleus/MVC_Soleus)*100);
Norm_IPD1_Tib_Ant = ((IPD1_Tib_Ant/MVC_Tib_Ant)*100);

Norm_IPD2_Biceps_Fem = ((IPD2_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_IPD2_Cont_Glut_Med = ((IPD2_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_IPD2_Glut_Max = ((IPD2_Glut_Max/MVC_Glut_Max)*100);
Norm_IPD2_Ipsi_Glut_Med = ((IPD2_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_IPD2_Peron_Long = ((IPD2_Peron_Long/MVC_Peron_Long)*100);
Norm_IPD2_Rec_Fem = ((IPD2_Rec_Fem/MVC_Rec_Fem)*100);
Norm_IPD2_Soleus = ((IPD2_Soleus/MVC_Soleus)*100);
Norm_IPD2_Tib_Ant = ((IPD2_Tib_Ant/MVC_Tib_Ant)*100);

Norm_IPD3_Biceps_Fem = ((IPD3_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_IPD3_Cont_Glut_Med = ((IPD3_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_IPD3_Glut_Max = ((IPD3_Glut_Max/MVC_Glut_Max)*100);
Norm_IPD3_Ipsi_Glut_Med = ((IPD3_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_IPD3_Peron_Long = ((IPD3_Peron_Long/MVC_Peron_Long)*100);
Norm_IPD3_Rec_Fem = ((IPD3_Rec_Fem/MVC_Rec_Fem)*100);
Norm_IPD3_Soleus = ((IPD3_Soleus/MVC_Soleus)*100);
Norm_IPD3_Tib_Ant = ((IPD3_Tib_Ant/MVC_Tib_Ant)*100);

Norm_NW1_Biceps_Fem = ((NW1_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_NW1_Cont_Glut_Med = ((NW1_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_NW1_Glut_Max = ((NW1_Glut_Max/MVC_Glut_Max)*100);
Norm_NW1_Ipsi_Glut_Med = ((NW1_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_NW1_Peron_Long = ((NW1_Peron_Long/MVC_Peron_Long)*100);
Norm_NW1_Rec_Fem = ((NW1_Rec_Fem/MVC_Rec_Fem)*100);
Norm_NW1_Soleus = ((NW1_Soleus/MVC_Soleus)*100);
Norm_NW1_Tib_Ant = ((NW1_Tib_Ant/MVC_Tib_Ant)*100);

Norm_NW2_Biceps_Fem = ((NW2_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_NW2_Cont_Glut_Med = ((NW2_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_NW2_Glut_Max = ((NW2_Glut_Max/MVC_Glut_Max)*100);
Norm_NW2_Ipsi_Glut_Med = ((NW2_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_NW2_Peron_Long = ((NW2_Peron_Long/MVC_Peron_Long)*100);
```

```
Norm_NW2_Rec_Fem = ((NW2_Rec_Fem/MVC_Rec_Fem)*100);
Norm_NW2_Soleus = ((NW2_Soleus/MVC_Soleus)*100);
Norm_NW2_Tib_Ant = ((NW2_Tib_Ant/MVC_Tib_Ant)*100);

Norm_NW3_Biceps_Fem = ((NW3_Biceps_Fem/MVC_Biceps_Fem)*100);
Norm_NW3_Cont_Glut_Med = ((NW3_Cont_Glut_Med/MVC_Cont_Glut_Med)*100);
Norm_NW3_Glut_Max = ((NW3_Glut_Max/MVC_Glut_Max)*100);
Norm_NW3_Ipsi_Glut_Med = ((NW3_Ipsi_Glut_Med/MVC_Ipsi_Glut_Med)*100);
Norm_NW3_Peron_Long = ((NW3_Peron_Long/MVC_Peron_Long)*100);
Norm_NW3_Rec_Fem = ((NW3_Rec_Fem/MVC_Rec_Fem)*100);
Norm_NW3_Soleus = ((NW3_Soleus/MVC_Soleus)*100);
Norm_NW3_Tib_Ant = ((NW3_Tib_Ant/MVC_Tib_Ant)*100);

% Create an output matrix to store all the values, each cell starts as zero
A = zeros(9,24);

% ID1
disp('ID1');

% What is the onset time for the muscle?
onset_prompt_ID1_BF = 'Biceps Fem Onset?';
ID1_Onset_Biceps_Fem =  input(onset_prompt_ID1_BF);
if ID1_Onset_Biceps_Fem == 0 % Enter zero if the onset doesn't exist, don't calculate variables
and just store zeros in output matrix
   x = 0;
else
   ID1_Onset_Biceps_Fem_Window = ID1_Onset_Biceps_Fem + 0.350;
   ID1_Onset_Biceps_Fem_Frame1 = round((ID1_Onset_Biceps_Fem/.005)+1);
   ID1_Onset_Biceps_Fem_FrameN = round((ID1_Onset_Biceps_Fem_Window/.005)+1);

   % Create subset that contains only data from onset to 200 ms after
   ID1_Biceps_Fem_subset =
Norm_ID1_Biceps_Fem(ID1_Onset_Biceps_Fem_Frame1:ID1_Onset_Biceps_Fem_FrameN);

   % Calculate the peak EMG between onset and the 200ms after
   pEMG_ID1_Biceps_Fem = max(ID1_Biceps_Fem_subset);

   % Calculate the time to peak: compute the time between onset and peak EMG
   % First need to return the index of the peak value
   pEMG_idx_ID1_Biceps_Fem = find( Norm_ID1_Biceps_Fem == pEMG_ID1_Biceps_Fem);
   t2p_ID1_Biceps_Fem =(((pEMG_idx_ID1_Biceps_Fem-1)*0.005) -
ID1_Onset_Biceps_Fem);

   % Calculate the average EMG between onset and the 200ms after
   AvgEMG_ID1_Biceps_Fem = mean(ID1_Biceps_Fem_subset);
```

```
    % Store the variable values in the according cells of the matrix
    A(1,1) = pEMG_ID1_Biceps_Fem;
    A(1,9) = AvgEMG_ID1_Biceps_Fem;
    A(1,17) = t2p_ID1_Biceps_Fem(1,1);
end


onset_prompt_ID1_CGM = 'Cont Glut Med Onset?';
ID1_Onset_Cont_Glut_Med =  input(onset_prompt_ID1_CGM);
 if ID1_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    ID1_Onset_Cont_Glut_Med_Window = ID1_Onset_Cont_Glut_Med + 0.350;
    ID1_Onset_Cont_Glut_Med_Frame1 = round((ID1_Onset_Cont_Glut_Med/.005)+1);
    ID1_Onset_Cont_Glut_Med_FrameN =
round((ID1_Onset_Cont_Glut_Med_Window/.005)+1);
    ID1_Cont_Glut_Med_subset =
Norm_ID1_Cont_Glut_Med(ID1_Onset_Cont_Glut_Med_Frame1:ID1_Onset_Cont_Glut_Med_
FrameN);
    pEMG_ID1_Cont_Glut_Med = max(ID1_Cont_Glut_Med_subset);
    pEMG_idx_ID1_Cont_Glut_Med = find(Norm_ID1_Cont_Glut_Med ==
pEMG_ID1_Cont_Glut_Med);
    t2p_ID1_Cont_Glut_Med =(((pEMG_idx_ID1_Cont_Glut_Med-1)*0.005) -
ID1_Onset_Cont_Glut_Med);
    AvgEMG_ID1_Cont_Glut_Med = mean(ID1_Cont_Glut_Med_subset);
    A(1,2) = pEMG_ID1_Cont_Glut_Med;
    A(1,10) = AvgEMG_ID1_Cont_Glut_Med;
    A(1,18) = t2p_ID1_Cont_Glut_Med(1,1);
 end


onset_prompt_ID1_GM = 'Glut Max Onset?';
ID1_Onset_Glut_Max =  input(onset_prompt_ID1_GM);
 if ID1_Onset_Glut_Max == 0
    x = 0;
 else
    ID1_Onset_Glut_Max_Window = ID1_Onset_Glut_Max + 0.350;
    ID1_Onset_Glut_Max_Frame1 = round((ID1_Onset_Biceps_Fem/.005)+1);
    ID1_Onset_Glut_Max_FrameN = round((ID1_Onset_Glut_Max_Window/.005)+1);
    ID1_Glut_Max_subset =
Norm_ID1_Glut_Max(ID1_Onset_Glut_Max_Frame1:ID1_Onset_Glut_Max_FrameN);
    pEMG_ID1_Glut_Max = max(ID1_Glut_Max_subset);
    pEMG_idx_ID1_Glut_Max = find(Norm_ID1_Glut_Max == pEMG_ID1_Glut_Max);
    t2p_ID1_Glut_Max =(((pEMG_idx_ID1_Glut_Max-1)*0.005) - ID1_Onset_Glut_Max);
    AvgEMG_ID1_Glut_Max = mean(ID1_Glut_Max_subset);
    A(1,3) = pEMG_ID1_Glut_Max;
```

```
    A(1,11) = AvgEMG_ID1_Glut_Max;
    A(1,19) = t2p_ID1_Glut_Max(1,1);
 end


onset_prompt_ID1_IGM = 'Ipsi Glut Med Onset?';
ID1_Onset_Ipsi_Glut_Med =  input(onset_prompt_ID1_IGM);
 if ID1_Onset_Ipsi_Glut_Med == 0
    x = 0;
 else
    ID1_Onset_Ipsi_Glut_Med_Window = ID1_Onset_Ipsi_Glut_Med + 0.350;
    ID1_Onset_Ipsi_Glut_Med_Frame1 = round((ID1_Onset_Ipsi_Glut_Med/.005)+1);
    ID1_Onset_Ipsi_Glut_Med_FrameN = round((ID1_Onset_Ipsi_Glut_Med_Window/.005)+1);
    ID1_Ipsi_Glut_Med_subset =
Norm_ID1_Ipsi_Glut_Med(ID1_Onset_Ipsi_Glut_Med_Frame1:ID1_Onset_Ipsi_Glut_Med_Fra
meN);
    pEMG_ID1_Ipsi_Glut_Med = max(ID1_Ipsi_Glut_Med_subset);
    pEMG_idx_ID1_Ipsi_Glut_Med = find(Norm_ID1_Ipsi_Glut_Med ==
pEMG_ID1_Ipsi_Glut_Med);
    t2p_ID1_Ipsi_Glut_Med =(((pEMG_idx_ID1_Ipsi_Glut_Med-1)*0.005) -
ID1_Onset_Ipsi_Glut_Med);
    AvgEMG_ID1_Ipsi_Glut_Med = mean(ID1_Ipsi_Glut_Med_subset);
    A(1,4) = pEMG_ID1_Ipsi_Glut_Med;
    A(1,12) = AvgEMG_ID1_Ipsi_Glut_Med;
    A(1,20) = t2p_ID1_Ipsi_Glut_Med(1,1);
 end

onset_prompt_ID1_PL = 'Peron Long Onset?';
ID1_Onset_Peron_Long =  input(onset_prompt_ID1_PL);
 if ID1_Onset_Peron_Long == 0
    x = 0;
 else
    ID1_Onset_Peron_Long_Window = ID1_Onset_Peron_Long + 0.350;
    ID1_Onset_Peron_Long_Frame1 = round((ID1_Onset_Peron_Long/.005)+1);
    ID1_Onset_Peron_Long_FrameN = round((ID1_Onset_Peron_Long_Window/.005)+1);
    ID1_Peron_Long_subset =
Norm_ID1_Peron_Long(ID1_Onset_Peron_Long_Frame1:ID1_Onset_Peron_Long_FrameN);
    pEMG_ID1_Peron_Long = max(ID1_Peron_Long_subset);
    pEMG_idx_ID1_Peron_Long = find(Norm_ID1_Peron_Long == pEMG_ID1_Peron_Long);
    t2p_ID1_Peron_Long =(((pEMG_idx_ID1_Peron_Long-1)*0.005) -
ID1_Onset_Peron_Long);
    AvgEMG_ID1_Peron_Long = mean(ID1_Peron_Long_subset);
    A(1,5) = pEMG_ID1_Peron_Long;
    A(1,13) = AvgEMG_ID1_Peron_Long;
    A(1,21) = t2p_ID1_Peron_Long(1,1);
 end
```

```
onset_prompt_ID1_RF = 'Rec Fem Onset?';
ID1_Onset_Rec_Fem =  input(onset_prompt_ID1_RF);
 if ID1_Onset_Rec_Fem == 0
    x = 0;
 else
    ID1_Onset_Rec_Fem_Window = ID1_Onset_Rec_Fem + 0.350;
    ID1_Onset_Rec_Fem_Frame1 = round((ID1_Onset_Rec_Fem/.005)+1);
    ID1_Onset_Rec_Fem_FrameN = round((ID1_Onset_Rec_Fem_Window/.005)+1);
    ID1_Rec_Fem_subset =
Norm_ID1_Rec_Fem(ID1_Onset_Rec_Fem_Frame1:ID1_Onset_Rec_Fem_FrameN);
    pEMG_ID1_Rec_Fem = max(ID1_Rec_Fem_subset);
    pEMG_idx_ID1_Rec_Fem = find(Norm_ID1_Rec_Fem == pEMG_ID1_Rec_Fem);
    t2p_ID1_Rec_Fem =(((pEMG_idx_ID1_Rec_Fem-1)*0.005) - ID1_Onset_Rec_Fem);
    AvgEMG_ID1_Rec_Fem = mean(ID1_Rec_Fem_subset);
    A(1,6) = pEMG_ID1_Rec_Fem;
    A(1,14) = AvgEMG_ID1_Rec_Fem;
    A(1,22) = t2p_ID1_Rec_Fem(1,1);
 end


onset_prompt_ID1_SOL = 'Soleus Onset?';
ID1_Onset_Soleus =  input(onset_prompt_ID1_SOL);
 if ID1_Onset_Soleus == 0
    x = 0;
 else
    ID1_Onset_Soleus_Window = ID1_Onset_Soleus + 0.350;
    ID1_Onset_Soleus_Frame1 = round((ID1_Onset_Soleus/.005)+1);
    ID1_Onset_Soleus_FrameN = round((ID1_Onset_Soleus_Window/.005)+1);
    ID1_Soleus_subset =
Norm_ID1_Soleus(ID1_Onset_Soleus_Frame1:ID1_Onset_Soleus_FrameN);
    pEMG_ID1_Soleus = max(ID1_Soleus_subset);
    pEMG_idx_ID1_Soleus = find(Norm_ID1_Soleus == pEMG_ID1_Soleus);
    t2p_ID1_Soleus =(((pEMG_idx_ID1_Soleus-1)*0.005) - ID1_Onset_Soleus);
    AvgEMG_ID1_Soleus = mean(ID1_Soleus_subset);
    A(1,7) = pEMG_ID1_Soleus;
    A(1,15) = AvgEMG_ID1_Soleus;
    A(1,23) = t2p_ID1_Soleus(1,1);
 end


onset_prompt_ID1_TA = 'Tib Ant Onset?';
ID1_Onset_Tib_Ant =  input(onset_prompt_ID1_TA);
 if ID1_Onset_Tib_Ant == 0
    x = 0;
```

```
 else
    ID1_Onset_Tib_Ant_Window = ID1_Onset_Tib_Ant + 0.350;
    ID1_Onset_Tib_Ant_Frame1 = round((ID1_Onset_Tib_Ant/.005)+1);
    ID1_Onset_Tib_Ant_FrameN = round((ID1_Onset_Tib_Ant_Window/.005)+1);
    ID1_Tib_Ant_subset =
Norm_ID1_Tib_Ant(ID1_Onset_Tib_Ant_Frame1:ID1_Onset_Tib_Ant_FrameN);
    pEMG_ID1_Tib_Ant = max(ID1_Tib_Ant_subset);
    pEMG_idx_ID1_Tib_Ant = find(Norm_ID1_Tib_Ant == pEMG_ID1_Tib_Ant);
    t2p_ID1_Tib_Ant =(((pEMG_idx_ID1_Tib_Ant-1)*0.005) - ID1_Onset_Tib_Ant);
    AvgEMG_ID1_Tib_Ant = mean(ID1_Tib_Ant_subset);
    A(1,8) = pEMG_ID1_Tib_Ant;
    A(1,16) = AvgEMG_ID1_Tib_Ant;
    A(1,24) = t2p_ID1_Tib_Ant(1,1);
 end


disp('ID2');
onset_prompt_ID2_BF = 'Biceps Fem Onset?';
ID2_Onset_Biceps_Fem =  input(onset_prompt_ID2_BF);
if ID2_Onset_Biceps_Fem == 0
    x = 0;
else
    ID2_Onset_Biceps_Fem_Window = ID2_Onset_Biceps_Fem + 0.350;
    ID2_Onset_Biceps_Fem_Frame1 = round((ID2_Onset_Biceps_Fem/.005)+1);
    ID2_Onset_Biceps_Fem_FrameN = round((ID2_Onset_Biceps_Fem_Window/.005)+1);
    ID2_Biceps_Fem_subset =
Norm_ID2_Biceps_Fem(ID2_Onset_Biceps_Fem_Frame1:ID2_Onset_Biceps_Fem_FrameN);
    pEMG_ID2_Biceps_Fem = max(ID2_Biceps_Fem_subset);
    pEMG_idx_ID2_Biceps_Fem = find( Norm_ID2_Biceps_Fem == pEMG_ID2_Biceps_Fem);
    t2p_ID2_Biceps_Fem =(((pEMG_idx_ID2_Biceps_Fem-1)*0.005) -
ID2_Onset_Biceps_Fem);
    AvgEMG_ID2_Biceps_Fem = mean(ID2_Biceps_Fem_subset);
    A(2,1) = pEMG_ID2_Biceps_Fem;
    A(2,9) = AvgEMG_ID2_Biceps_Fem;
    A(2,17) = t2p_ID2_Biceps_Fem(1,1);
end

onset_prompt_ID2_CGM = 'Cont Glut Med Onset?';
ID2_Onset_Cont_Glut_Med =  input(onset_prompt_ID2_CGM);
 if ID2_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    ID2_Onset_Cont_Glut_Med_Window = ID2_Onset_Cont_Glut_Med + 0.350;
    ID2_Onset_Cont_Glut_Med_Frame1 = round((ID2_Onset_Cont_Glut_Med/.005)+1);
    ID2_Onset_Cont_Glut_Med_FrameN =
round((ID2_Onset_Cont_Glut_Med_Window/.005)+1);
```

```
   ID2_Cont_Glut_Med_subset =
Norm_ID2_Cont_Glut_Med(ID2_Onset_Cont_Glut_Med_Frame1:ID2_Onset_Cont_Glut_Med_
FrameN);
   pEMG_ID2_Cont_Glut_Med = max(ID2_Cont_Glut_Med_subset);
   pEMG_idx_ID2_Cont_Glut_Med = find(Norm_ID2_Cont_Glut_Med ==
pEMG_ID2_Cont_Glut_Med);
   t2p_ID2_Cont_Glut_Med =(((pEMG_idx_ID2_Cont_Glut_Med-1)*0.005) -
ID2_Onset_Cont_Glut_Med);
   AvgEMG_ID2_Cont_Glut_Med = mean(ID2_Cont_Glut_Med_subset);
   A(2,2) = pEMG_ID2_Cont_Glut_Med;
   A(2,10) = AvgEMG_ID2_Cont_Glut_Med;
   A(2,18) = t2p_ID2_Cont_Glut_Med(1,1);
 end


onset_prompt_ID2_GM = 'Glut Max Onset?';
ID2_Onset_Glut_Max =  input(onset_prompt_ID2_GM);
 if ID2_Onset_Glut_Max == 0
   x = 0;
 else
   ID2_Onset_Glut_Max_Window = ID2_Onset_Glut_Max + 0.350;
   ID2_Onset_Glut_Max_Frame1 = round((ID2_Onset_Biceps_Fem/.005)+1);
   ID2_Onset_Glut_Max_FrameN = round((ID2_Onset_Glut_Max_Window/.005)+1);
   ID2_Glut_Max_subset =
Norm_ID2_Glut_Max(ID2_Onset_Glut_Max_Frame1:ID2_Onset_Glut_Max_FrameN);
   pEMG_ID2_Glut_Max = max(ID2_Glut_Max_subset);
   pEMG_idx_ID2_Glut_Max = find(Norm_ID2_Glut_Max == pEMG_ID2_Glut_Max);
   t2p_ID2_Glut_Max =(((pEMG_idx_ID2_Glut_Max-1)*0.005) - ID2_Onset_Glut_Max);
   AvgEMG_ID2_Glut_Max = mean(ID2_Glut_Max_subset);
   A(2,3) = pEMG_ID2_Glut_Max;
   A(2,11) = AvgEMG_ID2_Glut_Max;
   A(2,19) = t2p_ID2_Glut_Max(1,1);
 end


onset_prompt_ID2_IGM = 'Ipsi Glut Med Onset?';
ID2_Onset_Ipsi_Glut_Med =  input(onset_prompt_ID2_IGM);
 if ID2_Onset_Ipsi_Glut_Med == 0
   x = 0;
 else
   ID2_Onset_Ipsi_Glut_Med_Window = ID2_Onset_Ipsi_Glut_Med + 0.350;
   ID2_Onset_Ipsi_Glut_Med_Frame1 = round((ID2_Onset_Ipsi_Glut_Med/.005)+1);
   ID2_Onset_Ipsi_Glut_Med_FrameN = round((ID2_Onset_Ipsi_Glut_Med_Window/.005)+1);
   ID2_Ipsi_Glut_Med_subset =
Norm_ID2_Ipsi_Glut_Med(ID2_Onset_Ipsi_Glut_Med_Frame1:ID2_Onset_Ipsi_Glut_Med_Fra
meN);
```

```
    pEMG_ID2_Ipsi_Glut_Med = max(ID2_Ipsi_Glut_Med_subset);
    pEMG_idx_ID2_Ipsi_Glut_Med = find(Norm_ID2_Ipsi_Glut_Med ==
pEMG_ID2_Ipsi_Glut_Med);
    t2p_ID2_Ipsi_Glut_Med =(((pEMG_idx_ID2_Ipsi_Glut_Med-1)*0.005) -
ID2_Onset_Ipsi_Glut_Med);
    AvgEMG_ID2_Ipsi_Glut_Med = mean(ID2_Ipsi_Glut_Med_subset);
    A(2,4) = pEMG_ID2_Ipsi_Glut_Med;
    A(2,12) = AvgEMG_ID2_Ipsi_Glut_Med;
    A(2,20) = t2p_ID2_Ipsi_Glut_Med(1,1);
 end


onset_prompt_ID2_PL = 'Peron Long Onset?';
ID2_Onset_Peron_Long =  input(onset_prompt_ID2_PL);
 if ID2_Onset_Peron_Long == 0
    x = 0;
 else
    ID2_Onset_Peron_Long_Window = ID2_Onset_Peron_Long + 0.350;
    ID2_Onset_Peron_Long_Frame1 = round((ID2_Onset_Peron_Long/.005)+1);
    ID2_Onset_Peron_Long_FrameN = round((ID2_Onset_Peron_Long_Window/.005)+1);
    ID2_Peron_Long_subset =
Norm_ID2_Peron_Long(ID2_Onset_Peron_Long_Frame1:ID2_Onset_Peron_Long_FrameN);
    pEMG_ID2_Peron_Long = max(ID2_Peron_Long_subset);
    pEMG_idx_ID2_Peron_Long = find(Norm_ID2_Peron_Long == pEMG_ID2_Peron_Long);
    t2p_ID2_Peron_Long =(((pEMG_idx_ID2_Peron_Long-1)*0.005) -
ID2_Onset_Peron_Long);
    AvgEMG_ID2_Peron_Long = mean(ID2_Peron_Long_subset);
    A(2,5) = pEMG_ID2_Peron_Long;
    A(2,13) = AvgEMG_ID2_Peron_Long;
    A(2,21) = t2p_ID2_Peron_Long(1,1);
 end


onset_prompt_ID2_RF = 'Rec Fem Onset?';
ID2_Onset_Rec_Fem =  input(onset_prompt_ID2_RF);
 if ID2_Onset_Rec_Fem == 0
    x = 0;
 else
    ID2_Onset_Rec_Fem_Window = ID2_Onset_Rec_Fem + 0.350;
    ID2_Onset_Rec_Fem_Frame1 = round((ID2_Onset_Rec_Fem/.005)+1);
    ID2_Onset_Rec_Fem_FrameN = round((ID2_Onset_Rec_Fem_Window/.005)+1);
    ID2_Rec_Fem_subset =
Norm_ID2_Rec_Fem(ID2_Onset_Rec_Fem_Frame1:ID2_Onset_Rec_Fem_FrameN);
    pEMG_ID2_Rec_Fem = max(ID2_Rec_Fem_subset);
    pEMG_idx_ID2_Rec_Fem = find(Norm_ID2_Rec_Fem == pEMG_ID2_Rec_Fem);
    t2p_ID2_Rec_Fem =(((pEMG_idx_ID2_Rec_Fem-1)*0.005) - ID2_Onset_Rec_Fem);
```

```
    AvgEMG_ID2_Rec_Fem = mean(ID2_Rec_Fem_subset);
    A(2,6) = pEMG_ID2_Rec_Fem;
    A(2,14) = AvgEMG_ID2_Rec_Fem;
    A(2,22) = t2p_ID2_Rec_Fem(1,1);
 end


onset_prompt_ID2_SOL = 'Soleus Onset?';
ID2_Onset_Soleus =  input(onset_prompt_ID2_SOL);
 if ID2_Onset_Soleus == 0
    x = 0;
 else
    ID2_Onset_Soleus_Window = ID2_Onset_Soleus + 0.350;
    ID2_Onset_Soleus_Frame1 = round((ID2_Onset_Soleus/.005)+1);
    ID2_Onset_Soleus_FrameN = round((ID2_Onset_Soleus_Window/.005)+1);
    ID2_Soleus_subset =
Norm_ID2_Soleus(ID2_Onset_Soleus_Frame1:ID2_Onset_Soleus_FrameN);
    pEMG_ID2_Soleus = max(ID2_Soleus_subset);
    pEMG_idx_ID2_Soleus = find(Norm_ID2_Soleus == pEMG_ID2_Soleus);
    t2p_ID2_Soleus =(((pEMG_idx_ID2_Soleus-1)*0.005) - ID2_Onset_Soleus);
    AvgEMG_ID2_Soleus = mean(ID2_Soleus_subset);
    A(2,7) = pEMG_ID2_Soleus;
    A(2,15) = AvgEMG_ID2_Soleus;
    A(2,23) = t2p_ID2_Soleus(1,1);
 end


onset_prompt_ID2_TA = 'Tib Ant Onset?';
ID2_Onset_Tib_Ant =  input(onset_prompt_ID2_TA);
 if ID2_Onset_Tib_Ant == 0
    x = 0;
 else
    ID2_Onset_Tib_Ant_Window = ID2_Onset_Tib_Ant + 0.350;
    ID2_Onset_Tib_Ant_Frame1 = round((ID2_Onset_Tib_Ant/.005)+1);
    ID2_Onset_Tib_Ant_FrameN = round((ID2_Onset_Tib_Ant_Window/.005)+1);
    ID2_Tib_Ant_subset =
Norm_ID2_Tib_Ant(ID2_Onset_Tib_Ant_Frame1:ID2_Onset_Tib_Ant_FrameN);
    pEMG_ID2_Tib_Ant = max(ID2_Tib_Ant_subset);
    pEMG_idx_ID2_Tib_Ant = find(Norm_ID2_Tib_Ant == pEMG_ID2_Tib_Ant);
    t2p_ID2_Tib_Ant =(((pEMG_idx_ID2_Tib_Ant-1)*0.005) - ID2_Onset_Tib_Ant);
    AvgEMG_ID2_Tib_Ant = mean(ID2_Tib_Ant_subset);
    A(2,8) = pEMG_ID2_Tib_Ant;
    A(2,16) = AvgEMG_ID2_Tib_Ant;
    A(2,24) = t2p_ID2_Tib_Ant(1,1);
 end
```

```
disp('ID3');
onset_prompt_ID3_BF = 'Biceps Fem Onset?';
ID3_Onset_Biceps_Fem =  input(onset_prompt_ID3_BF);
 if ID3_Onset_Biceps_Fem == 0
    x = 0;
 else
    ID3_Onset_Biceps_Fem_Window = ID3_Onset_Biceps_Fem + 0.350;
    ID3_Onset_Biceps_Fem_Frame1 = round((ID3_Onset_Biceps_Fem/.005)+1);
    ID3_Onset_Biceps_Fem_FrameN = round((ID3_Onset_Biceps_Fem_Window/.005)+1);
    ID3_Biceps_Fem_subset =
Norm_ID3_Biceps_Fem(ID3_Onset_Biceps_Fem_Frame1:ID3_Onset_Biceps_Fem_FrameN);
    pEMG_ID3_Biceps_Fem = max(ID3_Biceps_Fem_subset);
    pEMG_idx_ID3_Biceps_Fem = find( Norm_ID3_Biceps_Fem == pEMG_ID3_Biceps_Fem);
    t2p_ID3_Biceps_Fem =(((pEMG_idx_ID3_Biceps_Fem-1)*0.005) -
ID3_Onset_Biceps_Fem);
    AvgEMG_ID3_Biceps_Fem = mean(ID3_Biceps_Fem_subset);
    A(3,1) = pEMG_ID3_Biceps_Fem;
    A(3,9) = AvgEMG_ID3_Biceps_Fem;
    A(3,17) = t2p_ID3_Biceps_Fem(1,1);
 end


onset_prompt_ID3_CGM = 'Cont Glut Med Onset?';
ID3_Onset_Cont_Glut_Med =  input(onset_prompt_ID3_CGM);
 if ID3_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    ID3_Onset_Cont_Glut_Med_Window = ID3_Onset_Cont_Glut_Med + 0.350;
    ID3_Onset_Cont_Glut_Med_Frame1 = round((ID3_Onset_Cont_Glut_Med/.005)+1);
    ID3_Onset_Cont_Glut_Med_FrameN =
round((ID3_Onset_Cont_Glut_Med_Window/.005)+1);
    ID3_Cont_Glut_Med_subset =
Norm_ID3_Cont_Glut_Med(ID3_Onset_Cont_Glut_Med_Frame1:ID3_Onset_Cont_Glut_Med_
FrameN);
    pEMG_ID3_Cont_Glut_Med = max(ID3_Cont_Glut_Med_subset);
    pEMG_idx_ID3_Cont_Glut_Med = find(Norm_ID3_Cont_Glut_Med ==
pEMG_ID3_Cont_Glut_Med);
    t2p_ID3_Cont_Glut_Med =(((pEMG_idx_ID3_Cont_Glut_Med-1)*0.005) -
ID3_Onset_Cont_Glut_Med);
    AvgEMG_ID3_Cont_Glut_Med = mean(ID3_Cont_Glut_Med_subset);
    A(3,2) = pEMG_ID3_Cont_Glut_Med;
    A(3,10) = AvgEMG_ID3_Cont_Glut_Med;
    A(3,18) = t2p_ID3_Cont_Glut_Med(1,1);
 end
```

```
onset_prompt_ID3_GM = 'Glut Max Onset?';
ID3_Onset_Glut_Max =  input(onset_prompt_ID3_GM);
 if ID3_Onset_Glut_Max == 0
    x = 0;
 else
    ID3_Onset_Glut_Max_Window = ID3_Onset_Glut_Max + 0.350;
    ID3_Onset_Glut_Max_Frame1 = round((ID3_Onset_Biceps_Fem/.005)+1);
    ID3_Onset_Glut_Max_FrameN = round((ID3_Onset_Glut_Max_Window/.005)+1);
    ID3_Glut_Max_subset =
Norm_ID3_Glut_Max(ID3_Onset_Glut_Max_Frame1:ID3_Onset_Glut_Max_FrameN);
    pEMG_ID3_Glut_Max = max(ID3_Glut_Max_subset);
    pEMG_idx_ID3_Glut_Max = find(Norm_ID3_Glut_Max == pEMG_ID3_Glut_Max);
    t2p_ID3_Glut_Max =(((pEMG_idx_ID3_Glut_Max-1)*0.005) - ID3_Onset_Glut_Max);
    AvgEMG_ID3_Glut_Max = mean(ID3_Glut_Max_subset);
    A(3,3) = pEMG_ID3_Glut_Max;
    A(3,11) = AvgEMG_ID3_Glut_Max;
    A(3,19) = t2p_ID3_Glut_Max(1,1);
 end


onset_prompt_ID3_IGM = 'Ipsi Glut Med Onset?';
ID3_Onset_Ipsi_Glut_Med =  input(onset_prompt_ID3_IGM);
 if ID3_Onset_Ipsi_Glut_Med == 0
    x = 0;
 else
    ID3_Onset_Ipsi_Glut_Med_Window = ID3_Onset_Ipsi_Glut_Med + 0.350;
    ID3_Onset_Ipsi_Glut_Med_Frame1 = round((ID3_Onset_Ipsi_Glut_Med/.005)+1);
    ID3_Onset_Ipsi_Glut_Med_FrameN = round((ID3_Onset_Ipsi_Glut_Med_Window/.005)+1);
    ID3_Ipsi_Glut_Med_subset =
Norm_ID3_Ipsi_Glut_Med(ID3_Onset_Ipsi_Glut_Med_Frame1:ID3_Onset_Ipsi_Glut_Med_Fra
meN);
    pEMG_ID3_Ipsi_Glut_Med = max(ID3_Ipsi_Glut_Med_subset);
    pEMG_idx_ID3_Ipsi_Glut_Med = find(Norm_ID3_Ipsi_Glut_Med ==
pEMG_ID3_Ipsi_Glut_Med);
    t2p_ID3_Ipsi_Glut_Med =(((pEMG_idx_ID3_Ipsi_Glut_Med-1)*0.005) -
ID3_Onset_Ipsi_Glut_Med);
    AvgEMG_ID3_Ipsi_Glut_Med = mean(ID3_Ipsi_Glut_Med_subset);
    A(3,4) = pEMG_ID3_Ipsi_Glut_Med;
    A(3,12) = AvgEMG_ID3_Ipsi_Glut_Med;
    A(3,20) = t2p_ID3_Ipsi_Glut_Med(1,1);
 end


onset_prompt_ID3_PL = 'Peron Long Onset?';
```

```
ID3_Onset_Peron_Long =  input(onset_prompt_ID3_PL);
 if ID3_Onset_Peron_Long == 0
    x = 0;
 else
    ID3_Onset_Peron_Long_Window = ID3_Onset_Peron_Long + 0.350;
    ID3_Onset_Peron_Long_Frame1 = round((ID3_Onset_Peron_Long/.005)+1);
    ID3_Onset_Peron_Long_FrameN = round((ID3_Onset_Peron_Long_Window/.005)+1);
    ID3_Peron_Long_subset =
Norm_ID3_Peron_Long(ID3_Onset_Peron_Long_Frame1:ID3_Onset_Peron_Long_FrameN);
    pEMG_ID3_Peron_Long = max(ID3_Peron_Long_subset);
    pEMG_idx_ID3_Peron_Long = find(Norm_ID3_Peron_Long == pEMG_ID3_Peron_Long);
    t2p_ID3_Peron_Long =(((pEMG_idx_ID3_Peron_Long-1)*0.005) -
ID3_Onset_Peron_Long);
    AvgEMG_ID3_Peron_Long = mean(ID3_Peron_Long_subset);
    A(3,5) = pEMG_ID3_Peron_Long;
    A(3,13) = AvgEMG_ID3_Peron_Long;
    A(3,21) = t2p_ID3_Peron_Long(1,1);
 end


onset_prompt_ID3_RF = 'Rec Fem Onset?';
ID3_Onset_Rec_Fem =  input(onset_prompt_ID3_RF);
 if ID3_Onset_Rec_Fem == 0
    x = 0;
 else
    ID3_Onset_Rec_Fem_Window = ID3_Onset_Rec_Fem + 0.350;
    ID3_Onset_Rec_Fem_Frame1 = round((ID3_Onset_Rec_Fem/.005)+1);
    ID3_Onset_Rec_Fem_FrameN = round((ID3_Onset_Rec_Fem_Window/.005)+1);
    ID3_Rec_Fem_subset =
Norm_ID3_Rec_Fem(ID3_Onset_Rec_Fem_Frame1:ID3_Onset_Rec_Fem_FrameN);
    pEMG_ID3_Rec_Fem = max(ID3_Rec_Fem_subset);
    pEMG_idx_ID3_Rec_Fem = find(Norm_ID3_Rec_Fem == pEMG_ID3_Rec_Fem);
    t2p_ID3_Rec_Fem =(((pEMG_idx_ID3_Rec_Fem-1)*0.005) - ID3_Onset_Rec_Fem);
    AvgEMG_ID3_Rec_Fem = mean(ID3_Rec_Fem_subset);
    A(3,6) = pEMG_ID3_Rec_Fem;
    A(3,14) = AvgEMG_ID3_Rec_Fem;
    A(3,22) = t2p_ID3_Rec_Fem(1,1);
 end


onset_prompt_ID3_SOL = 'Soleus Onset?';
ID3_Onset_Soleus =  input(onset_prompt_ID3_SOL);
 if ID3_Onset_Soleus == 0
    x = 0;
 else
    ID3_Onset_Soleus_Window = ID3_Onset_Soleus + 0.350;
```

```
   ID3_Onset_Soleus_Frame1 = round((ID3_Onset_Soleus/.005)+1);
   ID3_Onset_Soleus_FrameN = round((ID3_Onset_Soleus_Window/.005)+1);
   ID3_Soleus_subset =
Norm_ID3_Soleus(ID3_Onset_Soleus_Frame1:ID3_Onset_Soleus_FrameN);
   pEMG_ID3_Soleus = max(ID3_Soleus_subset);
   pEMG_idx_ID3_Soleus = find(Norm_ID3_Soleus == pEMG_ID3_Soleus);
   t2p_ID3_Soleus =(((pEMG_idx_ID3_Soleus-1)*0.005) - ID3_Onset_Soleus);
   AvgEMG_ID3_Soleus = mean(ID3_Soleus_subset);
   A(3,7) = pEMG_ID3_Soleus;
   A(3,15) = AvgEMG_ID3_Soleus;
   A(3,23) = t2p_ID3_Soleus(1,1);
 end

onset_prompt_ID3_TA = 'Tib Ant Onset?';
ID3_Onset_Tib_Ant =  input(onset_prompt_ID3_TA);
 if ID3_Onset_Tib_Ant == 0
   x = 0;
 else
   ID3_Onset_Tib_Ant_Window = ID3_Onset_Tib_Ant + 0.350;
   ID3_Onset_Tib_Ant_Frame1 = round((ID3_Onset_Tib_Ant/.005)+1);
   ID3_Onset_Tib_Ant_FrameN = round((ID3_Onset_Tib_Ant_Window/.005)+1);
   ID3_Tib_Ant_subset =
Norm_ID3_Tib_Ant(ID3_Onset_Tib_Ant_Frame1:ID3_Onset_Tib_Ant_FrameN);
   pEMG_ID3_Tib_Ant = max(ID3_Tib_Ant_subset);
   pEMG_idx_ID3_Tib_Ant = find(Norm_ID3_Tib_Ant == pEMG_ID3_Tib_Ant);
   t2p_ID3_Tib_Ant =(((pEMG_idx_ID3_Tib_Ant-1)*0.005) - ID3_Onset_Tib_Ant);
   AvgEMG_ID3_Tib_Ant = mean(ID3_Tib_Ant_subset);
   A(3,8) = pEMG_ID3_Tib_Ant;
   A(3,16) = AvgEMG_ID3_Tib_Ant;
   A(3,24) = t2p_ID3_Tib_Ant(1,1);
 end


disp('IPD1');
onset_prompt_IPD1_BF = 'Biceps Fem Onset?';
IPD1_Onset_Biceps_Fem =  input(onset_prompt_IPD1_BF);
 if IPD1_Onset_Biceps_Fem == 0
   x = 0;
 else
   IPD1_Onset_Biceps_Fem_Window = IPD1_Onset_Biceps_Fem + 0.350;
   IPD1_Onset_Biceps_Fem_Frame1 = round((IPD1_Onset_Biceps_Fem/.005)+1);
   IPD1_Onset_Biceps_Fem_FrameN = round((IPD1_Onset_Biceps_Fem_Window/.005)+1);
   IPD1_Biceps_Fem_subset =
Norm_IPD1_Biceps_Fem(IPD1_Onset_Biceps_Fem_Frame1:IPD1_Onset_Biceps_Fem_Frame
N);
   pEMG_IPD1_Biceps_Fem = max(IPD1_Biceps_Fem_subset);
```

```
   pEMG_idx_IPD1_Biceps_Fem = find(Norm_IPD1_Biceps_Fem ==
pEMG_IPD1_Biceps_Fem);
   t2p_IPD1_Biceps_Fem =(((pEMG_idx_IPD1_Biceps_Fem-1)*0.005) -
IPD1_Onset_Biceps_Fem);
   AvgEMG_IPD1_Biceps_Fem = mean(IPD1_Biceps_Fem_subset);
   A(4,1) = pEMG_IPD1_Biceps_Fem;
   A(4,9) = AvgEMG_IPD1_Biceps_Fem;
   A(4,17) = t2p_IPD1_Biceps_Fem(1,1);
 end


onset_prompt_IPD1_CGM = 'Cont Glut Med Onset?';
IPD1_Onset_Cont_Glut_Med =  input(onset_prompt_IPD1_CGM);
 if IPD1_Onset_Cont_Glut_Med == 0
   x = 0;
 else
   IPD1_Onset_Cont_Glut_Med_Window = IPD1_Onset_Cont_Glut_Med + 0.350;
   IPD1_Onset_Cont_Glut_Med_Frame1 = round((IPD1_Onset_Cont_Glut_Med/.005)+1);
   IPD1_Onset_Cont_Glut_Med_FrameN =
round((IPD1_Onset_Cont_Glut_Med_Window/.005)+1);
   IPD1_Cont_Glut_Med_subset =
Norm_IPD1_Cont_Glut_Med(IPD1_Onset_Cont_Glut_Med_Frame1:IPD1_Onset_Cont_Glut_
Med_FrameN);
   pEMG_IPD1_Cont_Glut_Med = max(IPD1_Cont_Glut_Med_subset);
   pEMG_idx_IPD1_Cont_Glut_Med = find(Norm_IPD1_Cont_Glut_Med ==
pEMG_IPD1_Cont_Glut_Med);
   t2p_IPD1_Cont_Glut_Med =(((pEMG_idx_IPD1_Cont_Glut_Med-1)*0.005) -
IPD1_Onset_Cont_Glut_Med);
   AvgEMG_IPD1_Cont_Glut_Med = mean(IPD1_Cont_Glut_Med_subset);
   A(4,2) = pEMG_IPD1_Cont_Glut_Med;
   A(4,10) = AvgEMG_IPD1_Cont_Glut_Med;
   A(4,18) = t2p_IPD1_Cont_Glut_Med(1,1);
 end


onset_prompt_IPD1_GM = 'Glut Max Onset?';
IPD1_Onset_Glut_Max =  input(onset_prompt_IPD1_GM);
 if IPD1_Onset_Glut_Max == 0
   x = 0;
 else
   IPD1_Onset_Glut_Max_Window = IPD1_Onset_Glut_Max + 0.350;
   IPD1_Onset_Glut_Max_Frame1 = round((IPD1_Onset_Biceps_Fem/.005)+1);
   IPD1_Onset_Glut_Max_FrameN = round((IPD1_Onset_Glut_Max_Window/.005)+1);
   IPD1_Glut_Max_subset =
Norm_IPD1_Glut_Max(IPD1_Onset_Glut_Max_Frame1:IPD1_Onset_Glut_Max_FrameN);
   pEMG_IPD1_Glut_Max = max(IPD1_Glut_Max_subset);
```

```
   pEMG_idx_IPD1_Glut_Max = find(Norm_IPD1_Glut_Max == pEMG_IPD1_Glut_Max);
   t2p_IPD1_Glut_Max =(((pEMG_idx_IPD1_Glut_Max-1)*0.005) - IPD1_Onset_Glut_Max);
   AvgEMG_IPD1_Glut_Max = mean(IPD1_Glut_Max_subset);
   A(4,3) = pEMG_IPD1_Glut_Max;
   A(4,11) = AvgEMG_IPD1_Glut_Max;
   A(4,19) = t2p_IPD1_Glut_Max(1,1);
 end


onset_prompt_IPD1_IGM = 'Ipsi Glut Med Onset?';
IPD1_Onset_Ipsi_Glut_Med =  input(onset_prompt_IPD1_IGM);
 if IPD1_Onset_Ipsi_Glut_Med == 0
   x = 0;
 else
   IPD1_Onset_Ipsi_Glut_Med_Window = IPD1_Onset_Ipsi_Glut_Med + 0.350;
   IPD1_Onset_Ipsi_Glut_Med_Frame1 = round((IPD1_Onset_Ipsi_Glut_Med/.005)+1);
   IPD1_Onset_Ipsi_Glut_Med_FrameN =
round((IPD1_Onset_Ipsi_Glut_Med_Window/.005)+1);
   IPD1_Ipsi_Glut_Med_subset =
Norm_IPD1_Ipsi_Glut_Med(IPD1_Onset_Ipsi_Glut_Med_Frame1:IPD1_Onset_Ipsi_Glut_Med
_FrameN);
   pEMG_IPD1_Ipsi_Glut_Med = max(IPD1_Ipsi_Glut_Med_subset);
   pEMG_idx_IPD1_Ipsi_Glut_Med = find(Norm_IPD1_Ipsi_Glut_Med ==
pEMG_IPD1_Ipsi_Glut_Med);
   t2p_IPD1_Ipsi_Glut_Med =(((pEMG_idx_IPD1_Ipsi_Glut_Med-1)*0.005) -
IPD1_Onset_Ipsi_Glut_Med);
   AvgEMG_IPD1_Ipsi_Glut_Med = mean(IPD1_Ipsi_Glut_Med_subset);
   A(4,4) = pEMG_IPD1_Ipsi_Glut_Med;
   A(4,12) = AvgEMG_IPD1_Ipsi_Glut_Med;
   A(4,20) = t2p_IPD1_Ipsi_Glut_Med(1,1);
 end


onset_prompt_IPD1_PL = 'Peron Long Onset?';
IPD1_Onset_Peron_Long =  input(onset_prompt_IPD1_PL);
 if IPD1_Onset_Peron_Long == 0
   x = 0;
 else
   IPD1_Onset_Peron_Long_Window = IPD1_Onset_Peron_Long + 0.350;
   IPD1_Onset_Peron_Long_Frame1 = round((IPD1_Onset_Peron_Long/.005)+1);
   IPD1_Onset_Peron_Long_FrameN = round((IPD1_Onset_Peron_Long_Window/.005)+1);
   IPD1_Peron_Long_subset =
Norm_IPD1_Peron_Long(IPD1_Onset_Peron_Long_Frame1:IPD1_Onset_Peron_Long_Frame
N);
   pEMG_IPD1_Peron_Long = max(IPD1_Peron_Long_subset);
```

```
   pEMG_idx_IPD1_Peron_Long = find(Norm_IPD1_Peron_Long ==
pEMG_IPD1_Peron_Long);
   t2p_IPD1_Peron_Long =(((pEMG_idx_IPD1_Peron_Long-1)*0.005) -
IPD1_Onset_Peron_Long);
   AvgEMG_IPD1_Peron_Long = mean(IPD1_Peron_Long_subset);
   A(4,5) = pEMG_IPD1_Peron_Long;
   A(4,13) = AvgEMG_IPD1_Peron_Long;
   A(4,21) = t2p_IPD1_Peron_Long(1,1);
 end


onset_prompt_IPD1_RF = 'Rec Fem Onset?';
IPD1_Onset_Rec_Fem =  input(onset_prompt_IPD1_RF);
 if IPD1_Onset_Rec_Fem == 0
   x = 0;
 else
   IPD1_Onset_Rec_Fem_Window = IPD1_Onset_Rec_Fem + 0.350;
   IPD1_Onset_Rec_Fem_Frame1 = round((IPD1_Onset_Rec_Fem/.005)+1);
   IPD1_Onset_Rec_Fem_FrameN = round((IPD1_Onset_Rec_Fem_Window/.005)+1);
   IPD1_Rec_Fem_subset =
Norm_IPD1_Rec_Fem(IPD1_Onset_Rec_Fem_Frame1:IPD1_Onset_Rec_Fem_FrameN);
   pEMG_IPD1_Rec_Fem = max(IPD1_Rec_Fem_subset);
   pEMG_idx_IPD1_Rec_Fem = find(Norm_IPD1_Rec_Fem == pEMG_IPD1_Rec_Fem);
   t2p_IPD1_Rec_Fem =(((pEMG_idx_IPD1_Rec_Fem-1)*0.005) - IPD1_Onset_Rec_Fem);
   AvgEMG_IPD1_Rec_Fem = mean(IPD1_Rec_Fem_subset);
   A(4,6) = pEMG_IPD1_Rec_Fem;
   A(4,14) = AvgEMG_IPD1_Rec_Fem;
   A(4,22) = t2p_IPD1_Rec_Fem(1,1);
 end


onset_prompt_IPD1_SOL = 'Soleus Onset?';
IPD1_Onset_Soleus =  input(onset_prompt_IPD1_SOL);
 if IPD1_Onset_Soleus == 0
   x = 0;
 else
   IPD1_Onset_Soleus_Window = IPD1_Onset_Soleus + 0.350;
   IPD1_Onset_Soleus_Frame1 = round((IPD1_Onset_Soleus/.005)+1);
   IPD1_Onset_Soleus_FrameN = round((IPD1_Onset_Soleus_Window/.005)+1);
   IPD1_Soleus_subset =
Norm_IPD1_Soleus(IPD1_Onset_Soleus_Frame1:IPD1_Onset_Soleus_FrameN);
   pEMG_IPD1_Soleus = max(IPD1_Soleus_subset);
   pEMG_idx_IPD1_Soleus = find(Norm_IPD1_Soleus == pEMG_IPD1_Soleus);
   t2p_IPD1_Soleus =(((pEMG_idx_IPD1_Soleus-1)*0.005) - IPD1_Onset_Soleus);
   AvgEMG_IPD1_Soleus = mean(IPD1_Soleus_subset);
   A(4,7) = pEMG_IPD1_Soleus;
```

```
    A(4,15) = AvgEMG_IPD1_Soleus;
    A(4,23) = t2p_IPD1_Soleus(1,1);
 end


onset_prompt_IPD1_TA = 'Tib Ant Onset?';
IPD1_Onset_Tib_Ant =  input(onset_prompt_IPD1_TA);
 if IPD1_Onset_Tib_Ant == 0
    x = 0;
 else
    IPD1_Onset_Tib_Ant_Window = IPD1_Onset_Tib_Ant + 0.350;
    IPD1_Onset_Tib_Ant_Frame1 = round((IPD1_Onset_Tib_Ant/.005)+1);
    IPD1_Onset_Tib_Ant_FrameN = round((IPD1_Onset_Tib_Ant_Window/.005)+1);
    IPD1_Tib_Ant_subset =
Norm_IPD1_Tib_Ant(IPD1_Onset_Tib_Ant_Frame1:IPD1_Onset_Tib_Ant_FrameN);
    pEMG_IPD1_Tib_Ant = max(IPD1_Tib_Ant_subset);
    pEMG_idx_IPD1_Tib_Ant = find(Norm_IPD1_Tib_Ant == pEMG_IPD1_Tib_Ant);
    t2p_IPD1_Tib_Ant =(((pEMG_idx_IPD1_Tib_Ant-1)*0.005) - IPD1_Onset_Tib_Ant);
    AvgEMG_IPD1_Tib_Ant = mean(IPD1_Tib_Ant_subset);
    A(4,8) = pEMG_IPD1_Tib_Ant;
    A(4,16) = AvgEMG_IPD1_Tib_Ant;
    A(4,24) = t2p_IPD1_Tib_Ant(1,1);
 end


disp('IPD2');
onset_prompt_IPD2_BF = 'Biceps Fem Onset?';
IPD2_Onset_Biceps_Fem =  input(onset_prompt_IPD2_BF);
 if IPD2_Onset_Biceps_Fem == 0
    x = 0;
 else
    IPD2_Onset_Biceps_Fem_Window = IPD2_Onset_Biceps_Fem + 0.350;
    IPD2_Onset_Biceps_Fem_Frame1 = round((IPD2_Onset_Biceps_Fem/.005)+1);
    IPD2_Onset_Biceps_Fem_FrameN = round((IPD2_Onset_Biceps_Fem_Window/.005)+1);
    IPD2_Biceps_Fem_subset =
Norm_IPD2_Biceps_Fem(IPD2_Onset_Biceps_Fem_Frame1:IPD2_Onset_Biceps_Fem_Frame
N);
    pEMG_IPD2_Biceps_Fem = max(IPD2_Biceps_Fem_subset);
    pEMG_idx_IPD2_Biceps_Fem = find(Norm_IPD2_Biceps_Fem ==
pEMG_IPD2_Biceps_Fem);
    t2p_IPD2_Biceps_Fem =(((pEMG_idx_IPD2_Biceps_Fem-1)*0.005) -
IPD2_Onset_Biceps_Fem);
    AvgEMG_IPD2_Biceps_Fem = mean(IPD2_Biceps_Fem_subset);
    A(5,1) = pEMG_IPD2_Biceps_Fem;
    A(5,9) = AvgEMG_IPD2_Biceps_Fem;
    A(5,17) = t2p_IPD2_Biceps_Fem(1,1);
```

```
 end


onset_prompt_IPD2_CGM = 'Cont Glut Med Onset?';
IPD2_Onset_Cont_Glut_Med =  input(onset_prompt_IPD2_CGM);
 if IPD2_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    IPD2_Onset_Cont_Glut_Med_Window = IPD2_Onset_Cont_Glut_Med + 0.350;
    IPD2_Onset_Cont_Glut_Med_Frame1 = round((IPD2_Onset_Cont_Glut_Med/.005)+1);
    IPD2_Onset_Cont_Glut_Med_FrameN =
round((IPD2_Onset_Cont_Glut_Med_Window/.005)+1);
    IPD2_Cont_Glut_Med_subset =
Norm_IPD2_Cont_Glut_Med(IPD2_Onset_Cont_Glut_Med_Frame1:IPD2_Onset_Cont_Glut_
Med_FrameN);
    pEMG_IPD2_Cont_Glut_Med = max(IPD2_Cont_Glut_Med_subset);
    pEMG_idx_IPD2_Cont_Glut_Med = find(Norm_IPD2_Cont_Glut_Med ==
pEMG_IPD2_Cont_Glut_Med);
    t2p_IPD2_Cont_Glut_Med =(((pEMG_idx_IPD2_Cont_Glut_Med-1)*0.005) -
IPD2_Onset_Cont_Glut_Med);
    AvgEMG_IPD2_Cont_Glut_Med = mean(IPD2_Cont_Glut_Med_subset);
    A(5,2) = pEMG_IPD2_Cont_Glut_Med;
    A(5,10) = AvgEMG_IPD2_Cont_Glut_Med;
    A(5,18) = t2p_IPD2_Cont_Glut_Med(1,1);
 end


onset_prompt_IPD2_GM = 'Glut Max Onset?';
IPD2_Onset_Glut_Max =  input(onset_prompt_IPD2_GM);
 if IPD2_Onset_Glut_Max == 0
    x = 0;
 else
    IPD2_Onset_Glut_Max_Window = IPD2_Onset_Glut_Max + 0.350;
    IPD2_Onset_Glut_Max_Frame1 = round((IPD2_Onset_Biceps_Fem/.005)+1);
    IPD2_Onset_Glut_Max_FrameN = round((IPD2_Onset_Glut_Max_Window/.005)+1);
    IPD2_Glut_Max_subset =
Norm_IPD2_Glut_Max(IPD2_Onset_Glut_Max_Frame1:IPD2_Onset_Glut_Max_FrameN);
    pEMG_IPD2_Glut_Max = max(IPD2_Glut_Max_subset);
    pEMG_idx_IPD2_Glut_Max = find(Norm_IPD2_Glut_Max == pEMG_IPD2_Glut_Max);
    t2p_IPD2_Glut_Max =(((pEMG_idx_IPD2_Glut_Max-1)*0.005) - IPD2_Onset_Glut_Max);
    AvgEMG_IPD2_Glut_Max = mean(IPD2_Glut_Max_subset);
    A(5,3) = pEMG_IPD2_Glut_Max;
    A(5,11) = AvgEMG_IPD2_Glut_Max;
    A(5,19) = t2p_IPD2_Glut_Max(1,1);
 end
```

```
onset_prompt_IPD2_IGM = 'Ipsi Glut Med Onset?';
IPD2_Onset_Ipsi_Glut_Med =  input(onset_prompt_IPD2_IGM);
 if IPD2_Onset_Ipsi_Glut_Med == 0
    x = 0;
 else
    IPD2_Onset_Ipsi_Glut_Med_Window = IPD2_Onset_Ipsi_Glut_Med + 0.350;
    IPD2_Onset_Ipsi_Glut_Med_Frame1 = round((IPD2_Onset_Ipsi_Glut_Med/.005)+1);
    IPD2_Onset_Ipsi_Glut_Med_FrameN =
round((IPD2_Onset_Ipsi_Glut_Med_Window/.005)+1);
    IPD2_Ipsi_Glut_Med_subset =
Norm_IPD2_Ipsi_Glut_Med(IPD2_Onset_Ipsi_Glut_Med_Frame1:IPD2_Onset_Ipsi_Glut_Med
_FrameN);
    pEMG_IPD2_Ipsi_Glut_Med = max(IPD2_Ipsi_Glut_Med_subset);
    pEMG_idx_IPD2_Ipsi_Glut_Med = find(Norm_IPD2_Ipsi_Glut_Med ==
pEMG_IPD2_Ipsi_Glut_Med);
    t2p_IPD2_Ipsi_Glut_Med =(((pEMG_idx_IPD2_Ipsi_Glut_Med-1)*0.005) -
IPD2_Onset_Ipsi_Glut_Med);
    AvgEMG_IPD2_Ipsi_Glut_Med = mean(IPD2_Ipsi_Glut_Med_subset);
    A(5,4) = pEMG_IPD2_Ipsi_Glut_Med;
    A(5,12) = AvgEMG_IPD2_Ipsi_Glut_Med;
    A(5,20) = t2p_IPD2_Ipsi_Glut_Med(1,1);
 end


onset_prompt_IPD2_PL = 'Peron Long Onset?';
IPD2_Onset_Peron_Long =  input(onset_prompt_IPD2_PL);
 if IPD2_Onset_Peron_Long == 0
    x = 0;
 else
    IPD2_Onset_Peron_Long_Window = IPD2_Onset_Peron_Long + 0.350;
    IPD2_Onset_Peron_Long_Frame1 = round((IPD2_Onset_Peron_Long/.005)+1);
    IPD2_Onset_Peron_Long_FrameN = round((IPD2_Onset_Peron_Long_Window/.005)+1);
    IPD2_Peron_Long_subset =
Norm_IPD2_Peron_Long(IPD2_Onset_Peron_Long_Frame1:IPD2_Onset_Peron_Long_Frame
N);
    pEMG_IPD2_Peron_Long = max(IPD2_Peron_Long_subset);
    pEMG_idx_IPD2_Peron_Long = find(Norm_IPD2_Peron_Long ==
pEMG_IPD2_Peron_Long);
    t2p_IPD2_Peron_Long =(((pEMG_idx_IPD2_Peron_Long-1)*0.005) -
IPD2_Onset_Peron_Long);
    AvgEMG_IPD2_Peron_Long = mean(IPD2_Peron_Long_subset);
    A(5,5) = pEMG_IPD2_Peron_Long;
    A(5,13) = AvgEMG_IPD2_Peron_Long;
    A(5,21) = t2p_IPD2_Peron_Long(1,1);
 end
```

```
onset_prompt_IPD2_RF = 'Rec Fem Onset?';
IPD2_Onset_Rec_Fem =  input(onset_prompt_IPD2_RF);
 if IPD2_Onset_Rec_Fem == 0
    x = 0;
 else
    IPD2_Onset_Rec_Fem_Window = IPD2_Onset_Rec_Fem + 0.350;
    IPD2_Onset_Rec_Fem_Frame1 = round((IPD2_Onset_Rec_Fem/.005)+1);
    IPD2_Onset_Rec_Fem_FrameN = round((IPD2_Onset_Rec_Fem_Window/.005)+1);
    IPD2_Rec_Fem_subset =
Norm_IPD2_Rec_Fem(IPD2_Onset_Rec_Fem_Frame1:IPD2_Onset_Rec_Fem_FrameN);
    pEMG_IPD2_Rec_Fem = max(IPD2_Rec_Fem_subset);
    pEMG_idx_IPD2_Rec_Fem = find(Norm_IPD2_Rec_Fem == pEMG_IPD2_Rec_Fem);
    t2p_IPD2_Rec_Fem =(((pEMG_idx_IPD2_Rec_Fem-1)*0.005) - IPD2_Onset_Rec_Fem);
    AvgEMG_IPD2_Rec_Fem = mean(IPD2_Rec_Fem_subset);
    A(5,6) = pEMG_IPD2_Rec_Fem;
    A(5,14) = AvgEMG_IPD2_Rec_Fem;
    A(5,22) = t2p_IPD2_Rec_Fem(1,1);
 end


onset_prompt_IPD2_SOL = 'Soleus Onset?';
IPD2_Onset_Soleus =  input(onset_prompt_IPD2_SOL);
 if IPD2_Onset_Soleus == 0
    x = 0;
 else
    IPD2_Onset_Soleus_Window = IPD2_Onset_Soleus + 0.350;
    IPD2_Onset_Soleus_Frame1 = round((IPD2_Onset_Soleus/.005)+1);
    IPD2_Onset_Soleus_FrameN = round((IPD2_Onset_Soleus_Window/.005)+1);
    IPD2_Soleus_subset =
Norm_IPD2_Soleus(IPD2_Onset_Soleus_Frame1:IPD2_Onset_Soleus_FrameN);
    pEMG_IPD2_Soleus = max(IPD2_Soleus_subset);
    pEMG_idx_IPD2_Soleus = find(Norm_IPD2_Soleus == pEMG_IPD2_Soleus);
    t2p_IPD2_Soleus =(((pEMG_idx_IPD2_Soleus-1)*0.005) - IPD2_Onset_Soleus);
    AvgEMG_IPD2_Soleus = mean(IPD2_Soleus_subset);
    A(5,7) = pEMG_IPD2_Soleus;
    A(5,15) = AvgEMG_IPD2_Soleus;
    A(5,23) = t2p_IPD2_Soleus(1,1);
 end


onset_prompt_IPD2_TA = 'Tib Ant Onset?';
IPD2_Onset_Tib_Ant =  input(onset_prompt_IPD2_TA);
 if IPD2_Onset_Tib_Ant == 0
    x = 0;
```

```
 else
    IPD2_Onset_Tib_Ant_Window = IPD2_Onset_Tib_Ant + 0.350;
    IPD2_Onset_Tib_Ant_Frame1 = round((IPD2_Onset_Tib_Ant/.005)+1);
    IPD2_Onset_Tib_Ant_FrameN = round((IPD2_Onset_Tib_Ant_Window/.005)+1);
    IPD2_Tib_Ant_subset =
Norm_IPD2_Tib_Ant(IPD2_Onset_Tib_Ant_Frame1:IPD2_Onset_Tib_Ant_FrameN);
    pEMG_IPD2_Tib_Ant = max(IPD2_Tib_Ant_subset);
    pEMG_idx_IPD2_Tib_Ant = find(Norm_IPD2_Tib_Ant == pEMG_IPD2_Tib_Ant);
    t2p_IPD2_Tib_Ant =(((pEMG_idx_IPD2_Tib_Ant-1)*0.005) - IPD2_Onset_Tib_Ant);
    AvgEMG_IPD2_Tib_Ant = mean(IPD2_Tib_Ant_subset);
    A(5,8) = pEMG_IPD2_Tib_Ant;
    A(5,16) = AvgEMG_IPD2_Tib_Ant;
    A(5,24) = t2p_IPD2_Tib_Ant(1,1);
 end


disp('IPD3');
onset_prompt_IPD3_BF = 'Biceps Fem Onset?';
IPD3_Onset_Biceps_Fem =  input(onset_prompt_IPD3_BF);
 if IPD3_Onset_Biceps_Fem == 0
    x = 0;
 else
    IPD3_Onset_Biceps_Fem_Window = IPD3_Onset_Biceps_Fem + 0.350;
    IPD3_Onset_Biceps_Fem_Frame1 = round((IPD3_Onset_Biceps_Fem/.005)+1);
    IPD3_Onset_Biceps_Fem_FrameN = round((IPD3_Onset_Biceps_Fem_Window/.005)+1);
    IPD3_Biceps_Fem_subset =
Norm_IPD3_Biceps_Fem(IPD3_Onset_Biceps_Fem_Frame1:IPD3_Onset_Biceps_Fem_Frame
N);
    pEMG_IPD3_Biceps_Fem = max(IPD3_Biceps_Fem_subset);
    pEMG_idx_IPD3_Biceps_Fem = find( Norm_IPD3_Biceps_Fem ==
pEMG_IPD3_Biceps_Fem);
    t2p_IPD3_Biceps_Fem =(((pEMG_idx_IPD3_Biceps_Fem-1)*0.005) -
IPD3_Onset_Biceps_Fem);
    AvgEMG_IPD3_Biceps_Fem = mean(IPD3_Biceps_Fem_subset);
    A(6,1) = pEMG_IPD3_Biceps_Fem;
    A(6,9) = AvgEMG_IPD3_Biceps_Fem;
    A(6,17) = t2p_IPD3_Biceps_Fem(1,1);
 end


onset_prompt_IPD3_CGM = 'Cont Glut Med Onset?';
IPD3_Onset_Cont_Glut_Med =  input(onset_prompt_IPD3_CGM);
 if IPD3_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    IPD3_Onset_Cont_Glut_Med_Window = IPD3_Onset_Cont_Glut_Med + 0.350;
```

```
   IPD3_Onset_Cont_Glut_Med_Frame1 = round((IPD3_Onset_Cont_Glut_Med/.005)+1);
   IPD3_Onset_Cont_Glut_Med_FrameN =
round((IPD3_Onset_Cont_Glut_Med_Window/.005)+1);
   IPD3_Cont_Glut_Med_subset =
Norm_IPD3_Cont_Glut_Med(IPD3_Onset_Cont_Glut_Med_Frame1:IPD3_Onset_Cont_Glut_
Med_FrameN);
   pEMG_IPD3_Cont_Glut_Med = max(IPD3_Cont_Glut_Med_subset);
   pEMG_idx_IPD3_Cont_Glut_Med = find(Norm_IPD3_Cont_Glut_Med ==
pEMG_IPD3_Cont_Glut_Med);
   t2p_IPD3_Cont_Glut_Med =(((pEMG_idx_IPD3_Cont_Glut_Med-1)*0.005) -
IPD3_Onset_Cont_Glut_Med);
   AvgEMG_IPD3_Cont_Glut_Med = mean(IPD3_Cont_Glut_Med_subset);
   A(6,2) = pEMG_IPD3_Cont_Glut_Med;
   A(6,10) = AvgEMG_IPD3_Cont_Glut_Med;
   A(6,18) = t2p_IPD3_Cont_Glut_Med(1,1);
 end


onset_prompt_IPD3_GM = 'Glut Max Onset?';
IPD3_Onset_Glut_Max =  input(onset_prompt_IPD3_GM);
 if IPD3_Onset_Glut_Max == 0
   x = 0;
 else
   IPD3_Onset_Glut_Max_Window = IPD3_Onset_Glut_Max + 0.350;
   IPD3_Onset_Glut_Max_Frame1 = round((IPD3_Onset_Biceps_Fem/.005)+1);
   IPD3_Onset_Glut_Max_FrameN = round((IPD3_Onset_Glut_Max_Window/.005)+1);
   IPD3_Glut_Max_subset =
Norm_IPD3_Glut_Max(IPD3_Onset_Glut_Max_Frame1:IPD3_Onset_Glut_Max_FrameN);
   pEMG_IPD3_Glut_Max = max(IPD3_Glut_Max_subset);
   pEMG_idx_IPD3_Glut_Max = find(Norm_IPD3_Glut_Max == pEMG_IPD3_Glut_Max);
   t2p_IPD3_Glut_Max =(((pEMG_idx_IPD3_Glut_Max-1)*0.005) - IPD3_Onset_Glut_Max);
   AvgEMG_IPD3_Glut_Max = mean(IPD3_Glut_Max_subset);
   A(6,3) = pEMG_IPD3_Glut_Max;
   A(6,11) = AvgEMG_IPD3_Glut_Max;
   A(6,19) = t2p_IPD3_Glut_Max(1,1);
 end


onset_prompt_IPD3_IGM = 'Ipsi Glut Med Onset?';
IPD3_Onset_Ipsi_Glut_Med =  input(onset_prompt_IPD3_IGM);
 if IPD3_Onset_Ipsi_Glut_Med == 0
   x = 0;
 else
   IPD3_Onset_Ipsi_Glut_Med_Window = IPD3_Onset_Ipsi_Glut_Med + 0.350;
   IPD3_Onset_Ipsi_Glut_Med_Frame1 = round((IPD3_Onset_Ipsi_Glut_Med/.005)+1);
```

```
   IPD3_Onset_Ipsi_Glut_Med_FrameN =
round((IPD3_Onset_Ipsi_Glut_Med_Window/.005)+1);
   IPD3_Ipsi_Glut_Med_subset =
Norm_IPD3_Ipsi_Glut_Med(IPD3_Onset_Ipsi_Glut_Med_Frame1:IPD3_Onset_Ipsi_Glut_Med
_FrameN);
   pEMG_IPD3_Ipsi_Glut_Med = max(IPD3_Ipsi_Glut_Med_subset);
   pEMG_idx_IPD3_Ipsi_Glut_Med = find(Norm_IPD3_Ipsi_Glut_Med ==
pEMG_IPD3_Ipsi_Glut_Med);
   t2p_IPD3_Ipsi_Glut_Med =(((pEMG_idx_IPD3_Ipsi_Glut_Med-1)*0.005) -
IPD3_Onset_Ipsi_Glut_Med);
   AvgEMG_IPD3_Ipsi_Glut_Med = mean(IPD3_Ipsi_Glut_Med_subset);
   A(6,4) = pEMG_IPD3_Ipsi_Glut_Med;
   A(6,12) = AvgEMG_IPD3_Ipsi_Glut_Med;
   A(6,20) = t2p_IPD3_Ipsi_Glut_Med(1,1);
 end


onset_prompt_IPD3_PL = 'Peron Long Onset?';
IPD3_Onset_Peron_Long =  input(onset_prompt_IPD3_PL);
 if IPD3_Onset_Peron_Long == 0
   x = 0;
 else
   IPD3_Onset_Peron_Long_Window = IPD3_Onset_Peron_Long + 0.350;
   IPD3_Onset_Peron_Long_Frame1 = round((IPD3_Onset_Peron_Long/.005)+1);
   IPD3_Onset_Peron_Long_FrameN = round((IPD3_Onset_Peron_Long_Window/.005)+1);
   IPD3_Peron_Long_subset =
Norm_IPD3_Peron_Long(IPD3_Onset_Peron_Long_Frame1:IPD3_Onset_Peron_Long_Frame
N);
   pEMG_IPD3_Peron_Long = max(IPD3_Peron_Long_subset);
   pEMG_idx_IPD3_Peron_Long = find(Norm_IPD3_Peron_Long ==
pEMG_IPD3_Peron_Long);
   t2p_IPD3_Peron_Long =(((pEMG_idx_IPD3_Peron_Long-1)*0.005) -
IPD3_Onset_Peron_Long);
   AvgEMG_IPD3_Peron_Long = mean(IPD3_Peron_Long_subset);
   A(6,5) = pEMG_IPD3_Peron_Long;
   A(6,13) = AvgEMG_IPD3_Peron_Long;
   A(6,21) = t2p_IPD3_Peron_Long(1,1);
 end


onset_prompt_IPD3_RF = 'Rec Fem Onset?';
IPD3_Onset_Rec_Fem =  input(onset_prompt_IPD3_RF);
 if IPD3_Onset_Rec_Fem == 0
   x = 0;
 else
   IPD3_Onset_Rec_Fem_Window = IPD3_Onset_Rec_Fem + 0.350;
```

```
    IPD3_Onset_Rec_Fem_Frame1 = round((IPD3_Onset_Rec_Fem/.005)+1);
    IPD3_Onset_Rec_Fem_FrameN = round((IPD3_Onset_Rec_Fem_Window/.005)+1);
    IPD3_Rec_Fem_subset =
Norm_IPD3_Rec_Fem(IPD3_Onset_Rec_Fem_Frame1:IPD3_Onset_Rec_Fem_FrameN);
    pEMG_IPD3_Rec_Fem = max(IPD3_Rec_Fem_subset);
    pEMG_idx_IPD3_Rec_Fem = find(Norm_IPD3_Rec_Fem == pEMG_IPD3_Rec_Fem);
    t2p_IPD3_Rec_Fem =(((pEMG_idx_IPD3_Rec_Fem-1)*0.005) - IPD3_Onset_Rec_Fem);
    AvgEMG_IPD3_Rec_Fem = mean(IPD3_Rec_Fem_subset);
    A(6,6) = pEMG_IPD3_Rec_Fem;
    A(6,14) = AvgEMG_IPD3_Rec_Fem;
    A(6,22) = t2p_IPD3_Rec_Fem(1,1);
 end


onset_prompt_IPD3_SOL = 'Soleus Onset?';
IPD3_Onset_Soleus =  input(onset_prompt_IPD3_SOL);
 if IPD3_Onset_Soleus == 0
    x = 0;
 else
    IPD3_Onset_Soleus_Window = IPD3_Onset_Soleus + 0.350;
    IPD3_Onset_Soleus_Frame1 = round((IPD3_Onset_Soleus/.005)+1);
    IPD3_Onset_Soleus_FrameN = round((IPD3_Onset_Soleus_Window/.005)+1);
    IPD3_Soleus_subset =
Norm_IPD3_Soleus(IPD3_Onset_Soleus_Frame1:IPD3_Onset_Soleus_FrameN);
    pEMG_IPD3_Soleus = max(IPD3_Soleus_subset);
    pEMG_idx_IPD3_Soleus = find(Norm_IPD3_Soleus == pEMG_IPD3_Soleus);
    t2p_IPD3_Soleus =(((pEMG_idx_IPD3_Soleus-1)*0.005) - IPD3_Onset_Soleus);
    AvgEMG_IPD3_Soleus = mean(IPD3_Soleus_subset);
    A(6,7) = pEMG_IPD3_Soleus;
    A(6,15) = AvgEMG_IPD3_Soleus;
    A(6,23) = t2p_IPD3_Soleus(1,1);
 end


onset_prompt_IPD3_TA = 'Tib Ant Onset?';
IPD3_Onset_Tib_Ant =  input(onset_prompt_IPD3_TA);
 if IPD3_Onset_Tib_Ant == 0
    x = 0;
 else
    IPD3_Onset_Tib_Ant_Window = IPD3_Onset_Tib_Ant + 0.350;
    IPD3_Onset_Tib_Ant_Frame1 = round((IPD3_Onset_Tib_Ant/.005)+1);
    IPD3_Onset_Tib_Ant_FrameN = round((IPD3_Onset_Tib_Ant_Window/.005)+1);
    IPD3_Tib_Ant_subset =
Norm_IPD3_Tib_Ant(IPD3_Onset_Tib_Ant_Frame1:IPD3_Onset_Tib_Ant_FrameN);
    pEMG_IPD3_Tib_Ant = max(IPD3_Tib_Ant_subset);
    pEMG_idx_IPD3_Tib_Ant = find(Norm_IPD3_Tib_Ant == pEMG_IPD3_Tib_Ant);
```

```
    t2p_IPD3_Tib_Ant =(((pEMG_idx_IPD3_Tib_Ant-1)*0.005) - IPD3_Onset_Tib_Ant);
    AvgEMG_IPD3_Tib_Ant = mean(IPD3_Tib_Ant_subset);
    A(6,8) = pEMG_IPD3_Tib_Ant;
    A(6,16) = AvgEMG_IPD3_Tib_Ant;
    A(6,24) = t2p_IPD3_Tib_Ant(1,1);
 end


disp('NW1');
onset_prompt_NW1_BF = 'Biceps Fem Onset?';
NW1_Onset_Biceps_Fem =  input(onset_prompt_NW1_BF);
 if NW1_Onset_Biceps_Fem == 0
    x = 0;
 else
    NW1_Onset_Biceps_Fem_Window = NW1_Onset_Biceps_Fem + 0.350;
    NW1_Onset_Biceps_Fem_Frame1 = round((NW1_Onset_Biceps_Fem/.005)+1);
    NW1_Onset_Biceps_Fem_FrameN = round((NW1_Onset_Biceps_Fem_Window/.005)+1);
    NW1_Biceps_Fem_subset =
Norm_NW1_Biceps_Fem(NW1_Onset_Biceps_Fem_Frame1:NW1_Onset_Biceps_Fem_Frame
N);
    pEMG_NW1_Biceps_Fem = max(NW1_Biceps_Fem_subset);
    pEMG_idx_NW1_Biceps_Fem = find(Norm_NW1_Biceps_Fem ==
pEMG_NW1_Biceps_Fem);
    t2p_NW1_Biceps_Fem =(((pEMG_idx_NW1_Biceps_Fem-1)*0.005) -
NW1_Onset_Biceps_Fem);
    AvgEMG_NW1_Biceps_Fem = mean(NW1_Biceps_Fem_subset);
    A(7,1) = pEMG_NW1_Biceps_Fem;
    A(7,9) = AvgEMG_NW1_Biceps_Fem;
    A(7,17) = t2p_NW1_Biceps_Fem(1,1);
 end


onset_prompt_NW1_CGM = 'Cont Glut Med Onset?';
NW1_Onset_Cont_Glut_Med =  input(onset_prompt_NW1_CGM);
 if NW1_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    NW1_Onset_Cont_Glut_Med_Window = NW1_Onset_Cont_Glut_Med + 0.350;
    NW1_Onset_Cont_Glut_Med_Frame1 = round((NW1_Onset_Cont_Glut_Med/.005)+1);
    NW1_Onset_Cont_Glut_Med_FrameN =
round((NW1_Onset_Cont_Glut_Med_Window/.005)+1);
    NW1_Cont_Glut_Med_subset =
Norm_NW1_Cont_Glut_Med(NW1_Onset_Cont_Glut_Med_Frame1:NW1_Onset_Cont_Glut_
Med_FrameN);
    pEMG_NW1_Cont_Glut_Med = max(NW1_Cont_Glut_Med_subset);
```

```
   pEMG_idx_NW1_Cont_Glut_Med = find(Norm_NW1_Cont_Glut_Med ==
pEMG_NW1_Cont_Glut_Med);
   t2p_NW1_Cont_Glut_Med =(((pEMG_idx_NW1_Cont_Glut_Med-1)*0.005) -
NW1_Onset_Cont_Glut_Med);
   AvgEMG_NW1_Cont_Glut_Med = mean(NW1_Cont_Glut_Med_subset);
   A(7,2) = pEMG_NW1_Cont_Glut_Med;
   A(7,10) = AvgEMG_NW1_Cont_Glut_Med;
   A(7,18) = t2p_NW1_Cont_Glut_Med(1,1);
 end


onset_prompt_NW1_GM = 'Glut Max Onset?';
NW1_Onset_Glut_Max =  input(onset_prompt_NW1_GM);
 if NW1_Onset_Glut_Max == 0
   x = 0;
 else
   NW1_Onset_Glut_Max_Window = NW1_Onset_Glut_Max + 0.350;
   NW1_Onset_Glut_Max_Frame1 = round((NW1_Onset_Biceps_Fem/.005)+1);
   NW1_Onset_Glut_Max_FrameN = round((NW1_Onset_Glut_Max_Window/.005)+1);
   NW1_Glut_Max_subset =
Norm_NW1_Glut_Max(NW1_Onset_Glut_Max_Frame1:NW1_Onset_Glut_Max_FrameN);
   pEMG_NW1_Glut_Max = max(NW1_Glut_Max_subset);
   pEMG_idx_NW1_Glut_Max = find(Norm_NW1_Glut_Max == pEMG_NW1_Glut_Max);
   t2p_NW1_Glut_Max =(((pEMG_idx_NW1_Glut_Max-1)*0.005) - NW1_Onset_Glut_Max);
   AvgEMG_NW1_Glut_Max = mean(NW1_Glut_Max_subset);
   A(7,3) = pEMG_NW1_Glut_Max;
   A(7,11) = AvgEMG_NW1_Glut_Max;
   A(7,19) = t2p_NW1_Glut_Max(1,1);
 end


onset_prompt_NW1_IGM = 'Ipsi Glut Med Onset?';
NW1_Onset_Ipsi_Glut_Med =  input(onset_prompt_NW1_IGM);
 if NW1_Onset_Ipsi_Glut_Med == 0
   x = 0;
 else
   NW1_Onset_Ipsi_Glut_Med_Window = NW1_Onset_Ipsi_Glut_Med + 0.350;
   NW1_Onset_Ipsi_Glut_Med_Frame1 = round((NW1_Onset_Ipsi_Glut_Med/.005)+1);
   NW1_Onset_Ipsi_Glut_Med_FrameN =
round((NW1_Onset_Ipsi_Glut_Med_Window/.005)+1);
   NW1_Ipsi_Glut_Med_subset =
Norm_NW1_Ipsi_Glut_Med(NW1_Onset_Ipsi_Glut_Med_Frame1:NW1_Onset_Ipsi_Glut_Med
_FrameN);
   pEMG_NW1_Ipsi_Glut_Med = max(NW1_Ipsi_Glut_Med_subset);
   pEMG_idx_NW1_Ipsi_Glut_Med = find(Norm_NW1_Ipsi_Glut_Med ==
pEMG_NW1_Ipsi_Glut_Med);
```

```
   t2p_NW1_Ipsi_Glut_Med =(((pEMG_idx_NW1_Ipsi_Glut_Med-1)*0.005) -
NW1_Onset_Ipsi_Glut_Med);
   AvgEMG_NW1_Ipsi_Glut_Med = mean(NW1_Ipsi_Glut_Med_subset);
   A(7,4) = pEMG_NW1_Ipsi_Glut_Med;
   A(7,12) = AvgEMG_NW1_Ipsi_Glut_Med;
   A(7,20) = t2p_NW1_Ipsi_Glut_Med(1,1);
 end


onset_prompt_NW1_PL = 'Peron Long Onset?';
NW1_Onset_Peron_Long =  input(onset_prompt_NW1_PL);
 if NW1_Onset_Peron_Long == 0
   x = 0;
 else
   NW1_Onset_Peron_Long_Window = NW1_Onset_Peron_Long + 0.350;
   NW1_Onset_Peron_Long_Frame1 = round((NW1_Onset_Peron_Long/.005)+1);
   NW1_Onset_Peron_Long_FrameN = round((NW1_Onset_Peron_Long_Window/.005)+1);
   NW1_Peron_Long_subset =
Norm_NW1_Peron_Long(NW1_Onset_Peron_Long_Frame1:NW1_Onset_Peron_Long_Frame
N);
   pEMG_NW1_Peron_Long = max(NW1_Peron_Long_subset);
   pEMG_idx_NW1_Peron_Long = find(Norm_NW1_Peron_Long ==
pEMG_NW1_Peron_Long);
   t2p_NW1_Peron_Long =(((pEMG_idx_NW1_Peron_Long-1)*0.005) -
NW1_Onset_Peron_Long);
   AvgEMG_NW1_Peron_Long = mean(NW1_Peron_Long_subset);
   A(7,5) = pEMG_NW1_Peron_Long;
   A(7,13) = AvgEMG_NW1_Peron_Long;
   A(7,21) = t2p_NW1_Peron_Long(1,1);
 end

onset_prompt_NW1_RF = 'Rec Fem Onset?';
NW1_Onset_Rec_Fem =  input(onset_prompt_NW1_RF);
 if NW1_Onset_Rec_Fem == 0
   x = 0;
 else
   NW1_Onset_Rec_Fem_Window = NW1_Onset_Rec_Fem + 0.350;
   NW1_Onset_Rec_Fem_Frame1 = round((NW1_Onset_Rec_Fem/.005)+1);
   NW1_Onset_Rec_Fem_FrameN = round((NW1_Onset_Rec_Fem_Window/.005)+1);
   NW1_Rec_Fem_subset =
Norm_NW1_Rec_Fem(NW1_Onset_Rec_Fem_Frame1:NW1_Onset_Rec_Fem_FrameN);
   pEMG_NW1_Rec_Fem = max(NW1_Rec_Fem_subset);
   pEMG_idx_NW1_Rec_Fem = find(Norm_NW1_Rec_Fem == pEMG_NW1_Rec_Fem);
   t2p_NW1_Rec_Fem =(((pEMG_idx_NW1_Rec_Fem-1)*0.005) - NW1_Onset_Rec_Fem);
   AvgEMG_NW1_Rec_Fem = mean(NW1_Rec_Fem_subset);
   A(7,6) = pEMG_NW1_Rec_Fem;
```

```
   A(7,14) = AvgEMG_NW1_Rec_Fem;
   A(7,22) = t2p_NW1_Rec_Fem(1,1);
 end


onset_prompt_NW1_SOL = 'Soleus Onset?';
NW1_Onset_Soleus =  input(onset_prompt_NW1_SOL);
 if NW1_Onset_Soleus == 0
   x = 0;
 else
   NW1_Onset_Soleus_Window = NW1_Onset_Soleus + 0.350;
   NW1_Onset_Soleus_Frame1 = round((NW1_Onset_Soleus/.005)+1);
   NW1_Onset_Soleus_FrameN = round((NW1_Onset_Soleus_Window/.005)+1);
   NW1_Soleus_subset =
Norm_NW1_Soleus(NW1_Onset_Soleus_Frame1:NW1_Onset_Soleus_FrameN);
   pEMG_NW1_Soleus = max(NW1_Soleus_subset);
   pEMG_idx_NW1_Soleus = find(Norm_NW1_Soleus == pEMG_NW1_Soleus);
   t2p_NW1_Soleus =(((pEMG_idx_NW1_Soleus-1)*0.005) - NW1_Onset_Soleus);
   AvgEMG_NW1_Soleus = mean(NW1_Soleus_subset);
   A(7,7) = pEMG_NW1_Soleus;
   A(7,15) = AvgEMG_NW1_Soleus;
   A(7,23) = t2p_NW1_Soleus(1,1);
 end


onset_prompt_NW1_TA = 'Tib Ant Onset?';
NW1_Onset_Tib_Ant =  input(onset_prompt_NW1_TA);
 if NW1_Onset_Tib_Ant == 0
   x = 0;
 else
   NW1_Onset_Tib_Ant_Window = NW1_Onset_Tib_Ant + 0.350;
   NW1_Onset_Tib_Ant_Frame1 = round((NW1_Onset_Tib_Ant/.005)+1);
   NW1_Onset_Tib_Ant_FrameN = round((NW1_Onset_Tib_Ant_Window/.005)+1);
   NW1_Tib_Ant_subset =
Norm_NW1_Tib_Ant(NW1_Onset_Tib_Ant_Frame1:NW1_Onset_Tib_Ant_FrameN);
   pEMG_NW1_Tib_Ant = max(NW1_Tib_Ant_subset);
   pEMG_idx_NW1_Tib_Ant = find(Norm_NW1_Tib_Ant == pEMG_NW1_Tib_Ant);
   t2p_NW1_Tib_Ant =(((pEMG_idx_NW1_Tib_Ant-1)*0.005) - NW1_Onset_Tib_Ant);
   AvgEMG_NW1_Tib_Ant = mean(NW1_Tib_Ant_subset);
   A(7,8) = pEMG_NW1_Tib_Ant;
   A(7,16) = AvgEMG_NW1_Tib_Ant;
   A(7,24) = t2p_NW1_Tib_Ant(1,1);
 end


disp('NW2');
```

```
onset_prompt_NW2_BF = 'Biceps Fem Onset?';
NW2_Onset_Biceps_Fem =  input(onset_prompt_NW2_BF);
 if NW2_Onset_Biceps_Fem == 0
    x = 0;
 else
    NW2_Onset_Biceps_Fem_Window = NW2_Onset_Biceps_Fem + 0.350;
    NW2_Onset_Biceps_Fem_Frame1 = round((NW2_Onset_Biceps_Fem/.005)+1);
    NW2_Onset_Biceps_Fem_FrameN = round((NW2_Onset_Biceps_Fem_Window/.005)+1);
    NW2_Biceps_Fem_subset =
Norm_NW2_Biceps_Fem(NW2_Onset_Biceps_Fem_Frame1:NW2_Onset_Biceps_Fem_Frame
N);
    pEMG_NW2_Biceps_Fem = max(NW2_Biceps_Fem_subset);
    pEMG_idx_NW2_Biceps_Fem = find(Norm_NW2_Biceps_Fem ==
pEMG_NW2_Biceps_Fem);
    t2p_NW2_Biceps_Fem =(((pEMG_idx_NW2_Biceps_Fem-1)*0.005) -
NW2_Onset_Biceps_Fem);
    AvgEMG_NW2_Biceps_Fem = mean(NW2_Biceps_Fem_subset);
    A(8,1) = pEMG_NW2_Biceps_Fem;
    A(8,9) = AvgEMG_NW2_Biceps_Fem;
    A(8,17) = t2p_NW2_Biceps_Fem(1,1);
 end


onset_prompt_NW2_CGM = 'Cont Glut Med Onset?';
NW2_Onset_Cont_Glut_Med =  input(onset_prompt_NW2_CGM);
 if NW2_Onset_Cont_Glut_Med == 0
    x = 0;
 else
    NW2_Onset_Cont_Glut_Med_Window = NW2_Onset_Cont_Glut_Med + 0.350;
    NW2_Onset_Cont_Glut_Med_Frame1 = round((NW2_Onset_Cont_Glut_Med/.005)+1);
    NW2_Onset_Cont_Glut_Med_FrameN =
round((NW2_Onset_Cont_Glut_Med_Window/.005)+1);
    NW2_Cont_Glut_Med_subset =
Norm_NW2_Cont_Glut_Med(NW2_Onset_Cont_Glut_Med_Frame1:NW2_Onset_Cont_Glut_
Med_FrameN);
    pEMG_NW2_Cont_Glut_Med = max(NW2_Cont_Glut_Med_subset);
    pEMG_idx_NW2_Cont_Glut_Med = find(Norm_NW2_Cont_Glut_Med ==
pEMG_NW2_Cont_Glut_Med);
    t2p_NW2_Cont_Glut_Med =(((pEMG_idx_NW2_Cont_Glut_Med-1)*0.005) -
NW2_Onset_Cont_Glut_Med);
    AvgEMG_NW2_Cont_Glut_Med = mean(NW2_Cont_Glut_Med_subset);
    A(8,2) = pEMG_NW2_Cont_Glut_Med;
    A(8,10) = AvgEMG_NW2_Cont_Glut_Med;
    A(8,18) = t2p_NW2_Cont_Glut_Med(1,1);
 end
```

```
onset_prompt_NW2_GM = 'Glut Max Onset?';
NW2_Onset_Glut_Max =  input(onset_prompt_NW2_GM);
 if NW2_Onset_Glut_Max == 0
    x = 0;
 else
    NW2_Onset_Glut_Max_Window = NW2_Onset_Glut_Max + 0.350;
    NW2_Onset_Glut_Max_Frame1 = round((NW2_Onset_Biceps_Fem/.005)+1);
    NW2_Onset_Glut_Max_FrameN = round((NW2_Onset_Glut_Max_Window/.005)+1);
    NW2_Glut_Max_subset =
Norm_NW2_Glut_Max(NW2_Onset_Glut_Max_Frame1:NW2_Onset_Glut_Max_FrameN);
    pEMG_NW2_Glut_Max = max(NW2_Glut_Max_subset);
    pEMG_idx_NW2_Glut_Max = find(Norm_NW2_Glut_Max == pEMG_NW2_Glut_Max);
    t2p_NW2_Glut_Max =(((pEMG_idx_NW2_Glut_Max-1)*0.005) - NW2_Onset_Glut_Max);
    AvgEMG_NW2_Glut_Max = mean(NW2_Glut_Max_subset);
    A(8,3) = pEMG_NW2_Glut_Max;
    A(8,11) = AvgEMG_NW2_Glut_Max;
    A(8,19) = t2p_NW2_Glut_Max(1,1);
 end


onset_prompt_NW2_IGM = 'Ipsi Glut Med Onset?';
NW2_Onset_Ipsi_Glut_Med =  input(onset_prompt_NW2_IGM);
 if NW2_Onset_Ipsi_Glut_Med == 0
    x = 0;
 else
    NW2_Onset_Ipsi_Glut_Med_Window = NW2_Onset_Ipsi_Glut_Med + 0.350;
    NW2_Onset_Ipsi_Glut_Med_Frame1 = round((NW2_Onset_Ipsi_Glut_Med/.005)+1);
    NW2_Onset_Ipsi_Glut_Med_FrameN =
round((NW2_Onset_Ipsi_Glut_Med_Window/.005)+1);
    NW2_Ipsi_Glut_Med_subset =
Norm_NW2_Ipsi_Glut_Med(NW2_Onset_Ipsi_Glut_Med_Frame1:NW2_Onset_Ipsi_Glut_Med
_FrameN);
    pEMG_NW2_Ipsi_Glut_Med = max(NW2_Ipsi_Glut_Med_subset);
    pEMG_idx_NW2_Ipsi_Glut_Med = find(Norm_NW2_Ipsi_Glut_Med ==
pEMG_NW2_Ipsi_Glut_Med);
    t2p_NW2_Ipsi_Glut_Med =(((pEMG_idx_NW2_Ipsi_Glut_Med-1)*0.005) -
NW2_Onset_Ipsi_Glut_Med);
    AvgEMG_NW2_Ipsi_Glut_Med = mean(NW2_Ipsi_Glut_Med_subset);
    A(8,4) = pEMG_NW2_Ipsi_Glut_Med;
    A(8,12) = AvgEMG_NW2_Ipsi_Glut_Med;
    A(8,20) = t2p_NW2_Ipsi_Glut_Med(1,1);
 end


onset_prompt_NW2_PL = 'Peron Long Onset?';
```

```
NW2_Onset_Peron_Long =  input(onset_prompt_NW2_PL);
 if NW2_Onset_Peron_Long == 0
    x = 0;
 else
    NW2_Onset_Peron_Long_Window = NW2_Onset_Peron_Long + 0.350;
    NW2_Onset_Peron_Long_Frame1 = round((NW2_Onset_Peron_Long/.005)+1);
    NW2_Onset_Peron_Long_FrameN = round((NW2_Onset_Peron_Long_Window/.005)+1);
    NW2_Peron_Long_subset =
Norm_NW2_Peron_Long(NW2_Onset_Peron_Long_Frame1:NW2_Onset_Peron_Long_Frame
N);
    pEMG_NW2_Peron_Long = max(NW2_Peron_Long_subset);
    pEMG_idx_NW2_Peron_Long = find(Norm_NW2_Peron_Long ==
pEMG_NW2_Peron_Long);
    t2p_NW2_Peron_Long =(((pEMG_idx_NW2_Peron_Long-1)*0.005) -
NW2_Onset_Peron_Long);
    AvgEMG_NW2_Peron_Long = mean(NW2_Peron_Long_subset);
    A(8,5) = pEMG_NW2_Peron_Long;
    A(8,13) = AvgEMG_NW2_Peron_Long;
    A(8,21) = t2p_NW2_Peron_Long(1,1);
 end


onset_prompt_NW2_RF = 'Rec Fem Onset?';
NW2_Onset_Rec_Fem =  input(onset_prompt_NW2_RF);
 if NW2_Onset_Rec_Fem == 0
    x = 0;
 else
    NW2_Onset_Rec_Fem_Window = NW2_Onset_Rec_Fem + 0.350;
    NW2_Onset_Rec_Fem_Frame1 = round((NW2_Onset_Rec_Fem/.005)+1);
    NW2_Onset_Rec_Fem_FrameN = round((NW2_Onset_Rec_Fem_Window/.005)+1);
    NW2_Rec_Fem_subset =
Norm_NW2_Rec_Fem(NW2_Onset_Rec_Fem_Frame1:NW2_Onset_Rec_Fem_FrameN);
    pEMG_NW2_Rec_Fem = max(NW2_Rec_Fem_subset);
    pEMG_idx_NW2_Rec_Fem = find(Norm_NW2_Rec_Fem == pEMG_NW2_Rec_Fem);
    t2p_NW2_Rec_Fem =(((pEMG_idx_NW2_Rec_Fem-1)*0.005) - NW2_Onset_Rec_Fem);
    AvgEMG_NW2_Rec_Fem = mean(NW2_Rec_Fem_subset);
    A(8,6) = pEMG_NW2_Rec_Fem;
    A(8,14) = AvgEMG_NW2_Rec_Fem;
    A(8,22) = t2p_NW2_Rec_Fem(1,1);
 end


onset_prompt_NW2_SOL = 'Soleus Onset?';
NW2_Onset_Soleus =  input(onset_prompt_NW2_SOL);
 if NW2_Onset_Soleus == 0
    x = 0;
```

```
 else
    NW2_Onset_Soleus_Window = NW2_Onset_Soleus + 0.350;
    NW2_Onset_Soleus_Frame1 = round((NW2_Onset_Soleus/.005)+1);
    NW2_Onset_Soleus_FrameN = round((NW2_Onset_Soleus_Window/.005)+1);
    NW2_Soleus_subset =
Norm_NW2_Soleus(NW2_Onset_Soleus_Frame1:NW2_Onset_Soleus_FrameN);
    pEMG_NW2_Soleus = max(NW2_Soleus_subset);
    pEMG_idx_NW2_Soleus = find(Norm_NW2_Soleus == pEMG_NW2_Soleus);
    t2p_NW2_Soleus =(((pEMG_idx_NW2_Soleus-1)*0.005) - NW2_Onset_Soleus);
    AvgEMG_NW2_Soleus = mean(NW2_Soleus_subset);
    A(8,7) = pEMG_NW2_Soleus;
    A(8,15) = AvgEMG_NW2_Soleus;
    A(8,23) = t2p_NW2_Soleus(1,1);
 end


onset_prompt_NW2_TA = 'Tib Ant Onset?';
NW2_Onset_Tib_Ant =  input(onset_prompt_NW2_TA);
 if NW2_Onset_Tib_Ant == 0
    x = 0;
 else
    NW2_Onset_Tib_Ant_Window = NW2_Onset_Tib_Ant + 0.350;
    NW2_Onset_Tib_Ant_Frame1 = round((NW2_Onset_Tib_Ant/.005)+1);
    NW2_Onset_Tib_Ant_FrameN = round((NW2_Onset_Tib_Ant_Window/.005)+1);
    NW2_Tib_Ant_subset =
Norm_NW2_Tib_Ant(NW2_Onset_Tib_Ant_Frame1:NW2_Onset_Tib_Ant_FrameN);
    pEMG_NW2_Tib_Ant = max(NW2_Tib_Ant_subset);
    pEMG_idx_NW2_Tib_Ant = find(Norm_NW2_Tib_Ant == pEMG_NW2_Tib_Ant);
    t2p_NW2_Tib_Ant =(((pEMG_idx_NW2_Tib_Ant-1)*0.005) - NW2_Onset_Tib_Ant);
    AvgEMG_NW2_Tib_Ant = mean(NW2_Tib_Ant_subset);
    A(8,8) = pEMG_NW2_Tib_Ant;
    A(8,16) = AvgEMG_NW2_Tib_Ant;
    A(8,24) = t2p_NW2_Tib_Ant(1,1);
 end


disp('NW3');
onset_prompt_NW3_BF = 'Biceps Fem Onset?';
NW3_Onset_Biceps_Fem =  input(onset_prompt_NW3_BF);
 if NW3_Onset_Biceps_Fem == 0
    x = 0;
 else
    NW3_Onset_Biceps_Fem_Window = NW3_Onset_Biceps_Fem + 0.350;
    NW3_Onset_Biceps_Fem_Frame1 = round((NW3_Onset_Biceps_Fem/.005)+1);
    NW3_Onset_Biceps_Fem_FrameN = round((NW3_Onset_Biceps_Fem_Window/.005)+1);
```

```
   NW3_Biceps_Fem_subset =
Norm_NW3_Biceps_Fem(NW3_Onset_Biceps_Fem_Frame1:NW3_Onset_Biceps_Fem_Frame
N);
   pEMG_NW3_Biceps_Fem = max(NW3_Biceps_Fem_subset);
   pEMG_idx_NW3_Biceps_Fem = find(Norm_NW3_Biceps_Fem ==
pEMG_NW3_Biceps_Fem);
   t2p_NW3_Biceps_Fem =(((pEMG_idx_NW3_Biceps_Fem-1)*0.005) -
NW3_Onset_Biceps_Fem);
   AvgEMG_NW3_Biceps_Fem = mean(NW3_Biceps_Fem_subset);
   A(9,1) = pEMG_NW3_Biceps_Fem;
   A(9,9) = AvgEMG_NW3_Biceps_Fem;
   A(9,17) = t2p_NW3_Biceps_Fem(1,1);
 end


onset_prompt_NW3_CGM = 'Cont Glut Med Onset?';
NW3_Onset_Cont_Glut_Med =  input(onset_prompt_NW3_CGM);
 if NW3_Onset_Cont_Glut_Med == 0
   x = 0;
 else
   NW3_Onset_Cont_Glut_Med_Window = NW3_Onset_Cont_Glut_Med + 0.350;
   NW3_Onset_Cont_Glut_Med_Frame1 = round((NW3_Onset_Cont_Glut_Med/.005)+1);
   NW3_Onset_Cont_Glut_Med_FrameN =
round((NW3_Onset_Cont_Glut_Med_Window/.005)+1);
   NW3_Cont_Glut_Med_subset =
Norm_NW3_Cont_Glut_Med(NW3_Onset_Cont_Glut_Med_Frame1:NW3_Onset_Cont_Glut_
Med_FrameN);
   pEMG_NW3_Cont_Glut_Med = max(NW3_Cont_Glut_Med_subset);
   pEMG_idx_NW3_Cont_Glut_Med = find(Norm_NW3_Cont_Glut_Med ==
pEMG_NW3_Cont_Glut_Med);
   t2p_NW3_Cont_Glut_Med =(((pEMG_idx_NW3_Cont_Glut_Med-1)*0.005) -
NW3_Onset_Cont_Glut_Med);
   AvgEMG_NW3_Cont_Glut_Med = mean(NW3_Cont_Glut_Med_subset);
   A(9,2) = pEMG_NW3_Cont_Glut_Med;
   A(9,10) = AvgEMG_NW3_Cont_Glut_Med;
   A(9,18) = t2p_NW3_Cont_Glut_Med(1,1);
 end


onset_prompt_NW3_GM = 'Glut Max Onset?';
NW3_Onset_Glut_Max =  input(onset_prompt_NW3_GM);
 if NW3_Onset_Glut_Max == 0
   x = 0;
 else
   NW3_Onset_Glut_Max_Window = NW3_Onset_Glut_Max + 0.350;
   NW3_Onset_Glut_Max_Frame1 = round((NW3_Onset_Biceps_Fem/.005)+1);
```

```
    NW3_Onset_Glut_Max_FrameN = round((NW3_Onset_Glut_Max_Window/.005)+1);
    NW3_Glut_Max_subset =
Norm_NW3_Glut_Max(NW3_Onset_Glut_Max_Frame1:NW3_Onset_Glut_Max_FrameN);
    pEMG_NW3_Glut_Max = max(NW3_Glut_Max_subset);
    pEMG_idx_NW3_Glut_Max = find(Norm_NW3_Glut_Max == pEMG_NW3_Glut_Max);
    t2p_NW3_Glut_Max =(((pEMG_idx_NW3_Glut_Max-1)*0.005) - NW3_Onset_Glut_Max);
    AvgEMG_NW3_Glut_Max = mean(NW3_Glut_Max_subset);
    A(9,3) = pEMG_NW3_Glut_Max;
    A(9,11) = AvgEMG_NW3_Glut_Max;
    A(9,19) = t2p_NW3_Glut_Max(1,1);
 end


onset_prompt_NW3_IGM = 'Ipsi Glut Med Onset?';
NW3_Onset_Ipsi_Glut_Med =  input(onset_prompt_NW3_IGM);
 if NW3_Onset_Ipsi_Glut_Med == 0
    x = 0;
 else
    NW3_Onset_Ipsi_Glut_Med_Window = NW3_Onset_Ipsi_Glut_Med + 0.350;
    NW3_Onset_Ipsi_Glut_Med_Frame1 = round((NW3_Onset_Ipsi_Glut_Med/.005)+1);
    NW3_Onset_Ipsi_Glut_Med_FrameN =
round((NW3_Onset_Ipsi_Glut_Med_Window/.005)+1);
    NW3_Ipsi_Glut_Med_subset =
Norm_NW3_Ipsi_Glut_Med(NW3_Onset_Ipsi_Glut_Med_Frame1:NW3_Onset_Ipsi_Glut_Med
_FrameN);
    pEMG_NW3_Ipsi_Glut_Med = max(NW3_Ipsi_Glut_Med_subset);
    pEMG_idx_NW3_Ipsi_Glut_Med = find(Norm_NW3_Ipsi_Glut_Med ==
pEMG_NW3_Ipsi_Glut_Med);
    t2p_NW3_Ipsi_Glut_Med =(((pEMG_idx_NW3_Ipsi_Glut_Med-1)*0.005) -
NW3_Onset_Ipsi_Glut_Med);
    AvgEMG_NW3_Ipsi_Glut_Med = mean(NW3_Ipsi_Glut_Med_subset);
    A(9,4) = pEMG_NW3_Ipsi_Glut_Med;
    A(9,12) = AvgEMG_NW3_Ipsi_Glut_Med;
    A(9,20) = t2p_NW3_Ipsi_Glut_Med(1,1);
 end


onset_prompt_NW3_PL = 'Peron Long Onset?';
NW3_Onset_Peron_Long =  input(onset_prompt_NW3_PL);
 if NW3_Onset_Peron_Long == 0
    x = 0;
 else
    NW3_Onset_Peron_Long_Window = NW3_Onset_Peron_Long + 0.350;
    NW3_Onset_Peron_Long_Frame1 = round((NW3_Onset_Peron_Long/.005)+1);
    NW3_Onset_Peron_Long_FrameN = round((NW3_Onset_Peron_Long_Window/.005)+1);
```

```
   NW3_Peron_Long_subset =
Norm_NW3_Peron_Long(NW3_Onset_Peron_Long_Frame1:NW3_Onset_Peron_Long_Frame
N);
   pEMG_NW3_Peron_Long = max(NW3_Peron_Long_subset);
   pEMG_idx_NW3_Peron_Long = find(Norm_NW3_Peron_Long ==
pEMG_NW3_Peron_Long);
   t2p_NW3_Peron_Long =(((pEMG_idx_NW3_Peron_Long-1)*0.005) -
NW3_Onset_Peron_Long);
   AvgEMG_NW3_Peron_Long = mean(NW3_Peron_Long_subset);
   A(9,5) = pEMG_NW3_Peron_Long;
   A(9,13) = AvgEMG_NW3_Peron_Long;
   A(9,21) = t2p_NW3_Peron_Long(1,1);
 end


onset_prompt_NW3_RF = 'Rec Fem Onset?';
NW3_Onset_Rec_Fem =  input(onset_prompt_NW3_RF);
 if NW3_Onset_Rec_Fem == 0
   x = 0;
 else
   NW3_Onset_Rec_Fem_Window = NW3_Onset_Rec_Fem + 0.350;
   NW3_Onset_Rec_Fem_Frame1 = round((NW3_Onset_Rec_Fem/.005)+1);
   NW3_Onset_Rec_Fem_FrameN = round((NW3_Onset_Rec_Fem_Window/.005)+1);
   NW3_Rec_Fem_subset =
Norm_NW3_Rec_Fem(NW3_Onset_Rec_Fem_Frame1:NW3_Onset_Rec_Fem_FrameN);
   pEMG_NW3_Rec_Fem = max(NW3_Rec_Fem_subset);
   pEMG_idx_NW3_Rec_Fem = find(Norm_NW3_Rec_Fem == pEMG_NW3_Rec_Fem);
   t2p_NW3_Rec_Fem =(((pEMG_idx_NW3_Rec_Fem-1)*0.005) - NW3_Onset_Rec_Fem);
   AvgEMG_NW3_Rec_Fem = mean(NW3_Rec_Fem_subset);
   A(9,6) = pEMG_NW3_Rec_Fem;
   A(9,14) = AvgEMG_NW3_Rec_Fem;
   A(9,22) = t2p_NW3_Rec_Fem(1,1);
 end


onset_prompt_NW3_SOL = 'Soleus Onset?';
NW3_Onset_Soleus =  input(onset_prompt_NW3_SOL);
 if NW3_Onset_Soleus == 0
   x = 0;
 else
   NW3_Onset_Soleus_Window = NW3_Onset_Soleus + 0.350;
   NW3_Onset_Soleus_Frame1 = round((NW3_Onset_Soleus/.005)+1);
   NW3_Onset_Soleus_FrameN = round((NW3_Onset_Soleus_Window/.005)+1);
   NW3_Soleus_subset =
Norm_NW3_Soleus(NW3_Onset_Soleus_Frame1:NW3_Onset_Soleus_FrameN);
   pEMG_NW3_Soleus = max(NW3_Soleus_subset);
```

```
   pEMG_idx_NW3_Soleus = find(Norm_NW3_Soleus == pEMG_NW3_Soleus);
   t2p_NW3_Soleus =(((pEMG_idx_NW3_Soleus-1)*0.005) - NW3_Onset_Soleus);
   AvgEMG_NW3_Soleus = mean(NW3_Soleus_subset);
   A(9,7) = pEMG_NW3_Soleus;
   A(9,15) = AvgEMG_NW3_Soleus;
   A(9,23) = t2p_NW3_Soleus(1,1);
 end


onset_prompt_NW3_TA = 'Tib Ant Onset?';
NW3_Onset_Tib_Ant =  input(onset_prompt_NW3_TA);
 if NW3_Onset_Tib_Ant == 0
   x = 0;
 else
   NW3_Onset_Tib_Ant_Window = NW3_Onset_Tib_Ant + 0.350;
   NW3_Onset_Tib_Ant_Frame1 = round((NW3_Onset_Tib_Ant/.005)+1);
   NW3_Onset_Tib_Ant_FrameN = round((NW3_Onset_Tib_Ant_Window/.005)+1);
   NW3_Tib_Ant_subset =
Norm_NW3_Tib_Ant(NW3_Onset_Tib_Ant_Frame1:NW3_Onset_Tib_Ant_FrameN);
   pEMG_NW3_Tib_Ant = max(NW3_Tib_Ant_subset);
   pEMG_idx_NW3_Tib_Ant = find(Norm_NW3_Tib_Ant == pEMG_NW3_Tib_Ant);
   t2p_NW3_Tib_Ant =(((pEMG_idx_NW3_Tib_Ant-1)*0.005) - NW3_Onset_Tib_Ant);
   AvgEMG_NW3_Tib_Ant = mean(NW3_Tib_Ant_subset);
   A(9,8) = pEMG_NW3_Tib_Ant;
   A(9,16) = AvgEMG_NW3_Tib_Ant;
   A(9,24) = t2p_NW3_Tib_Ant(1,1);
 end


%End









close all;
clear all;
clc;

% Kinematics
```

```
cd 'C:\Users\Emilia\Documents\Documents\Grad School\Thesis\Exports\
% First import the Kinematic data and store it as a matrix
% Doesn't work with the dlmread for a txt file because the file is too
% large and the data gets messed up?
Kinematics = xlsread('6_c.xlsx');%This is c

% Create variables or matrices for each column of the data to distinguish
Time = Kinematics(:,2);

ID1_L_Ankle_Angle_x = Kinematics(:,11);
ID1_L_Ankle_Angle_y = Kinematics(:,12);
ID1_L_Ankle_Angle_z = Kinematics(:,13);
ID1_L_Ankle_Velocity_x = Kinematics(:,14);
ID1_L_Ankle_Velocity_y = Kinematics(:,15);
ID1_L_Ankle_Velocity_z = Kinematics(:,16);
ID1_L_Knee_Angle_x = Kinematics(:,17);
ID1_L_Knee_Angle_y = Kinematics(:,18);
ID1_L_Knee_Angle_z = Kinematics(:,19);
ID1_L_Knee_Velocity_x = Kinematics(:,20);
ID1_L_Knee_Velocity_y = Kinematics(:,21);
ID1_L_Knee_Velocity_z = Kinematics(:,22);
ID1_L_Hip_Angle_x = Kinematics(:,23);
ID1_L_Hip_Angle_y = Kinematics(:,24);
ID1_L_Hip_Angle_z = Kinematics(:,25);
ID1_L_Hip_Velocity_x = Kinematics(:,26);
ID1_L_Hip_Velocity_y = Kinematics(:,27);
ID1_L_Hip_Velocity_z = Kinematics(:,28);
ID1_R_Ankle_Angle_x = Kinematics(:,29);
ID1_R_Ankle_Angle_y = Kinematics(:,30);
ID1_R_Ankle_Angle_z = Kinematics(:,31);
ID1_R_Ankle_Velocity_x = Kinematics(:,32);
ID1_R_Ankle_Velocity_y = Kinematics(:,33);
ID1_R_Ankle_Velocity_z = Kinematics(:,34);
ID1_R_Knee_Angle_x = Kinematics(:,35);
ID1_R_Knee_Angle_y = Kinematics(:,36);
ID1_R_Knee_Angle_z = Kinematics(:,37);
ID1_R_Knee_Velocity_x = Kinematics(:,38);
ID1_R_Knee_Velocity_y = Kinematics(:,39);
ID1_R_Knee_Velocity_z = Kinematics(:,40);
ID1_R_Hip_Angle_x = Kinematics(:,41);
ID1_R_Hip_Angle_y = Kinematics(:,42);
ID1_R_Hip_Angle_z = Kinematics(:,43);
ID1_R_Hip_Velocity_x = Kinematics(:,44);
ID1_R_Hip_Velocity_y = Kinematics(:,45);
ID1_R_Hip_Velocity_z = Kinematics(:,46);
```

```
ID2_L_Ankle_Angle_x = Kinematics(:,48);
ID2_L_Ankle_Angle_y = Kinematics(:,49);
ID2_L_Ankle_Angle_z = Kinematics(:,50);
ID2_L_Ankle_Velocity_x = Kinematics(:,51);
ID2_L_Ankle_Velocity_y = Kinematics(:,52);
ID2_L_Ankle_Velocity_z = Kinematics(:,53);
ID2_L_Knee_Angle_x = Kinematics(:,54);
ID2_L_Knee_Angle_y = Kinematics(:,55);
ID2_L_Knee_Angle_z = Kinematics(:,56);
ID2_L_Knee_Velocity_x = Kinematics(:,57);
ID2_L_Knee_Velocity_y = Kinematics(:,58);
ID2_L_Knee_Velocity_z = Kinematics(:,59);
ID2_L_Hip_Angle_x = Kinematics(:,60);
ID2_L_Hip_Angle_y = Kinematics(:,61);
ID2_L_Hip_Angle_z = Kinematics(:,62);
ID2_L_Hip_Velocity_x = Kinematics(:,63);
ID2_L_Hip_Velocity_y = Kinematics(:,64);
ID2_L_Hip_Velocity_z = Kinematics(:,65);
ID2_R_Ankle_Angle_x = Kinematics(:,66);
ID2_R_Ankle_Angle_y = Kinematics(:,67);
ID2_R_Ankle_Angle_z = Kinematics(:,68);
ID2_R_Ankle_Velocity_x = Kinematics(:,69);
ID2_R_Ankle_Velocity_y = Kinematics(:,70);
ID2_R_Ankle_Velocity_z = Kinematics(:,71);
ID2_R_Knee_Angle_x = Kinematics(:,72);
ID2_R_Knee_Angle_y = Kinematics(:,73);
ID2_R_Knee_Angle_z = Kinematics(:,74);
ID2_R_Knee_Velocity_x = Kinematics(:,75);
ID2_R_Knee_Velocity_y = Kinematics(:,76);
ID2_R_Knee_Velocity_z = Kinematics(:,77);
ID2_R_Hip_Angle_x = Kinematics(:,78);
ID2_R_Hip_Angle_y = Kinematics(:,79);
ID2_R_Hip_Angle_z = Kinematics(:,80);
ID2_R_Hip_Velocity_x = Kinematics(:,81);
ID2_R_Hip_Velocity_y = Kinematics(:,82);
ID2_R_Hip_Velocity_z = Kinematics(:,83);

ID3_L_Ankle_Angle_x = Kinematics(:,85);
ID3_L_Ankle_Angle_y = Kinematics(:,86);
ID3_L_Ankle_Angle_z = Kinematics(:,87);
ID3_L_Ankle_Velocity_x = Kinematics(:,88);
ID3_L_Ankle_Velocity_y = Kinematics(:,89);
ID3_L_Ankle_Velocity_z = Kinematics(:,90);
ID3_L_Knee_Angle_x = Kinematics(:,91);
ID3_L_Knee_Angle_y = Kinematics(:,92);
ID3_L_Knee_Angle_z = Kinematics(:,93);
```

```
ID3_L_Knee_Velocity_x = Kinematics(:,94);
ID3_L_Knee_Velocity_y = Kinematics(:,95);
ID3_L_Knee_Velocity_z = Kinematics(:,96);
ID3_L_Hip_Angle_x = Kinematics(:,97);
ID3_L_Hip_Angle_y = Kinematics(:,98);
ID3_L_Hip_Angle_z = Kinematics(:,99);
ID3_L_Hip_Velocity_x = Kinematics(:,100);
ID3_L_Hip_Velocity_y = Kinematics(:,101);
ID3_L_Hip_Velocity_z = Kinematics(:,102);
ID3_R_Ankle_Angle_x = Kinematics(:,103);
ID3_R_Ankle_Angle_y = Kinematics(:,104);
ID3_R_Ankle_Angle_z = Kinematics(:,105);
ID3_R_Ankle_Velocity_x = Kinematics(:,106);
ID3_R_Ankle_Velocity_y = Kinematics(:,107);
ID3_R_Ankle_Velocity_z = Kinematics(:,108);
ID3_R_Knee_Angle_x = Kinematics(:,109);
ID3_R_Knee_Angle_y = Kinematics(:,110);
ID3_R_Knee_Angle_z = Kinematics(:,111);
ID3_R_Knee_Velocity_x = Kinematics(:,112);
ID3_R_Knee_Velocity_y = Kinematics(:,113);
ID3_R_Knee_Velocity_z = Kinematics(:,114);
ID3_R_Hip_Angle_x = Kinematics(:,115);
ID3_R_Hip_Angle_y = Kinematics(:,116);
ID3_R_Hip_Angle_z = Kinematics(:,117);
ID3_R_Hip_Velocity_x = Kinematics(:,118);
ID3_R_Hip_Velocity_y = Kinematics(:,119);
ID3_R_Hip_Velocity_z = Kinematics(:,120);

IPD1_L_Ankle_Angle_x = Kinematics(:,122);
IPD1_L_Ankle_Angle_y = Kinematics(:,123);
IPD1_L_Ankle_Angle_z = Kinematics(:,124);
IPD1_L_Ankle_Velocity_x = Kinematics(:,125);
IPD1_L_Ankle_Velocity_y = Kinematics(:,126);
IPD1_L_Ankle_Velocity_z = Kinematics(:,127);
IPD1_L_Knee_Angle_x = Kinematics(:,128);
IPD1_L_Knee_Angle_y = Kinematics(:,129);
IPD1_L_Knee_Angle_z = Kinematics(:,130);
IPD1_L_Knee_Velocity_x = Kinematics(:,131);
IPD1_L_Knee_Velocity_y = Kinematics(:,132);
IPD1_L_Knee_Velocity_z = Kinematics(:,133);
IPD1_L_Hip_Angle_x = Kinematics(:,134);
IPD1_L_Hip_Angle_y = Kinematics(:,135);
IPD1_L_Hip_Angle_z = Kinematics(:,136);
IPD1_L_Hip_Velocity_x = Kinematics(:,137);
IPD1_L_Hip_Velocity_y = Kinematics(:,138);
IPD1_L_Hip_Velocity_z = Kinematics(:,139);
```

```
IPD1_R_Ankle_Angle_x = Kinematics(:,140);
IPD1_R_Ankle_Angle_y = Kinematics(:,141);
IPD1_R_Ankle_Angle_z = Kinematics(:,142);
IPD1_R_Ankle_Velocity_x = Kinematics(:,143);
IPD1_R_Ankle_Velocity_y = Kinematics(:,144);
IPD1_R_Ankle_Velocity_z = Kinematics(:,145);
IPD1_R_Knee_Angle_x = Kinematics(:,146);
IPD1_R_Knee_Angle_y = Kinematics(:,147);
IPD1_R_Knee_Angle_z = Kinematics(:,148);
IPD1_R_Knee_Velocity_x = Kinematics(:,149);
IPD1_R_Knee_Velocity_y = Kinematics(:,150);
IPD1_R_Knee_Velocity_z = Kinematics(:,151);
IPD1_R_Hip_Angle_x = Kinematics(:,152);
IPD1_R_Hip_Angle_y = Kinematics(:,153);
IPD1_R_Hip_Angle_z = Kinematics(:,154);
IPD1_R_Hip_Velocity_x = Kinematics(:,155);
IPD1_R_Hip_Velocity_y = Kinematics(:,156);
IPD1_R_Hip_Velocity_z = Kinematics(:,157);

IPD2_L_Ankle_Angle_x = Kinematics(:,159);
IPD2_L_Ankle_Angle_y = Kinematics(:,160);
IPD2_L_Ankle_Angle_z = Kinematics(:,161);
IPD2_L_Ankle_Velocity_x = Kinematics(:,162);
IPD2_L_Ankle_Velocity_y = Kinematics(:,163);
IPD2_L_Ankle_Velocity_z = Kinematics(:,164);
IPD2_L_Knee_Angle_x = Kinematics(:,165);
IPD2_L_Knee_Angle_y = Kinematics(:,166);
IPD2_L_Knee_Angle_z = Kinematics(:,167);
IPD2_L_Knee_Velocity_x = Kinematics(:,168);
IPD2_L_Knee_Velocity_y = Kinematics(:,169);
IPD2_L_Knee_Velocity_z = Kinematics(:,170);
IPD2_L_Hip_Angle_x = Kinematics(:,171);
IPD2_L_Hip_Angle_y = Kinematics(:,172);
IPD2_L_Hip_Angle_z = Kinematics(:,173);
IPD2_L_Hip_Velocity_x = Kinematics(:,174);
IPD2_L_Hip_Velocity_y = Kinematics(:,175);
IPD2_L_Hip_Velocity_z = Kinematics(:,176);
IPD2_R_Ankle_Angle_x = Kinematics(:,177);
IPD2_R_Ankle_Angle_y = Kinematics(:,178);
IPD2_R_Ankle_Angle_z = Kinematics(:,179);
IPD2_R_Ankle_Velocity_x = Kinematics(:,180);
IPD2_R_Ankle_Velocity_y = Kinematics(:,181);
IPD2_R_Ankle_Velocity_z = Kinematics(:,182);
IPD2_R_Knee_Angle_x = Kinematics(:,183);
IPD2_R_Knee_Angle_y = Kinematics(:,184);
IPD2_R_Knee_Angle_z = Kinematics(:,185);
```

```
IPD2_R_Knee_Velocity_x = Kinematics(:,186);
IPD2_R_Knee_Velocity_y = Kinematics(:,187);
IPD2_R_Knee_Velocity_z = Kinematics(:,188);
IPD2_R_Hip_Angle_x = Kinematics(:,189);
IPD2_R_Hip_Angle_y = Kinematics(:,190);
IPD2_R_Hip_Angle_z = Kinematics(:,191);
IPD2_R_Hip_Velocity_x = Kinematics(:,192);
IPD2_R_Hip_Velocity_y = Kinematics(:,193);
IPD2_R_Hip_Velocity_z = Kinematics(:,194);

IPD3_L_Ankle_Angle_x = Kinematics(:,196);
IPD3_L_Ankle_Angle_y = Kinematics(:,197);
IPD3_L_Ankle_Angle_z = Kinematics(:,198);
IPD3_L_Ankle_Velocity_x = Kinematics(:,199);
IPD3_L_Ankle_Velocity_y = Kinematics(:,200);
IPD3_L_Ankle_Velocity_z = Kinematics(:,201);
IPD3_L_Knee_Angle_x = Kinematics(:,202);
IPD3_L_Knee_Angle_y = Kinematics(:,203);
IPD3_L_Knee_Angle_z = Kinematics(:,204);
IPD3_L_Knee_Velocity_x = Kinematics(:,205);
IPD3_L_Knee_Velocity_y = Kinematics(:,206);
IPD3_L_Knee_Velocity_z = Kinematics(:,207);
IPD3_L_Hip_Angle_x = Kinematics(:,208);
IPD3_L_Hip_Angle_y = Kinematics(:,209);
IPD3_L_Hip_Angle_z = Kinematics(:,210);
IPD3_L_Hip_Velocity_x = Kinematics(:,211);
IPD3_L_Hip_Velocity_y = Kinematics(:,212);
IPD3_L_Hip_Velocity_z = Kinematics(:,213);
IPD3_R_Ankle_Angle_x = Kinematics(:,214);
IPD3_R_Ankle_Angle_y = Kinematics(:,215);
IPD3_R_Ankle_Angle_z = Kinematics(:,216);
IPD3_R_Ankle_Velocity_x = Kinematics(:,217);
IPD3_R_Ankle_Velocity_y = Kinematics(:,218);
IPD3_R_Ankle_Velocity_z = Kinematics(:,219);
IPD3_R_Knee_Angle_x = Kinematics(:,220);
IPD3_R_Knee_Angle_y = Kinematics(:,221);
IPD3_R_Knee_Angle_z = Kinematics(:,222);
IPD3_R_Knee_Velocity_x = Kinematics(:,223);
IPD3_R_Knee_Velocity_y = Kinematics(:,224);
IPD3_R_Knee_Velocity_z = Kinematics(:,225);
IPD3_R_Hip_Angle_x = Kinematics(:,226);
IPD3_R_Hip_Angle_y = Kinematics(:,227);
IPD3_R_Hip_Angle_z = Kinematics(:,228);
IPD3_R_Hip_Velocity_x = Kinematics(:,229);
IPD3_R_Hip_Velocity_y = Kinematics(:,230);
IPD3_R_Hip_Velocity_z = Kinematics(:,231);
```

```
NW1_L_Ankle_Angle_x = Kinematics(:,233);
NW1_L_Ankle_Angle_y = Kinematics(:,234);
NW1_L_Ankle_Angle_z = Kinematics(:,235);
NW1_L_Ankle_Velocity_x = Kinematics(:,236);
NW1_L_Ankle_Velocity_y = Kinematics(:,237);
NW1_L_Ankle_Velocity_z = Kinematics(:,238);
NW1_L_Knee_Angle_x = Kinematics(:,239);
NW1_L_Knee_Angle_y = Kinematics(:,240);
NW1_L_Knee_Angle_z = Kinematics(:,241);
NW1_L_Knee_Velocity_x = Kinematics(:,242);
NW1_L_Knee_Velocity_y = Kinematics(:,243);
NW1_L_Knee_Velocity_z = Kinematics(:,244);
NW1_L_Hip_Angle_x = Kinematics(:,245);
NW1_L_Hip_Angle_y = Kinematics(:,246);
NW1_L_Hip_Angle_z = Kinematics(:,247);
NW1_L_Hip_Velocity_x = Kinematics(:,248);
NW1_L_Hip_Velocity_y = Kinematics(:,249);
NW1_L_Hip_Velocity_z = Kinematics(:,250);
NW1_R_Ankle_Angle_x = Kinematics(:,251);
NW1_R_Ankle_Angle_y = Kinematics(:,252);
NW1_R_Ankle_Angle_z = Kinematics(:,253);
NW1_R_Ankle_Velocity_x = Kinematics(:,254);
NW1_R_Ankle_Velocity_y = Kinematics(:,255);
NW1_R_Ankle_Velocity_z = Kinematics(:,256);
NW1_R_Knee_Angle_x = Kinematics(:,257);
NW1_R_Knee_Angle_y = Kinematics(:,258);
NW1_R_Knee_Angle_z = Kinematics(:,259);
NW1_R_Knee_Velocity_x = Kinematics(:,260);
NW1_R_Knee_Velocity_y = Kinematics(:,261);
NW1_R_Knee_Velocity_z = Kinematics(:,262);
NW1_R_Hip_Angle_x = Kinematics(:,263);
NW1_R_Hip_Angle_y = Kinematics(:,264);
NW1_R_Hip_Angle_z = Kinematics(:,265);
NW1_R_Hip_Velocity_x = Kinematics(:,266);
NW1_R_Hip_Velocity_y = Kinematics(:,267);
NW1_R_Hip_Velocity_z = Kinematics(:,268);

NW2_L_Ankle_Angle_x = Kinematics(:,270);
NW2_L_Ankle_Angle_y = Kinematics(:,271);
NW2_L_Ankle_Angle_z = Kinematics(:,272);
NW2_L_Ankle_Velocity_x = Kinematics(:,273);
NW2_L_Ankle_Velocity_y = Kinematics(:,274);
NW2_L_Ankle_Velocity_z = Kinematics(:,275);
NW2_L_Knee_Angle_x = Kinematics(:,276);
NW2_L_Knee_Angle_y = Kinematics(:,277);
```

```
NW2_L_Knee_Angle_z = Kinematics(:,278);
NW2_L_Knee_Velocity_x = Kinematics(:,279);
NW2_L_Knee_Velocity_y = Kinematics(:,280);
NW2_L_Knee_Velocity_z = Kinematics(:,281);
NW2_L_Hip_Angle_x = Kinematics(:,282);
NW2_L_Hip_Angle_y = Kinematics(:,283);
NW2_L_Hip_Angle_z = Kinematics(:,284);
NW2_L_Hip_Velocity_x = Kinematics(:,285);
NW2_L_Hip_Velocity_y = Kinematics(:,286);
NW2_L_Hip_Velocity_z = Kinematics(:,287);
NW2_R_Ankle_Angle_x = Kinematics(:,288);
NW2_R_Ankle_Angle_y = Kinematics(:,289);
NW2_R_Ankle_Angle_z = Kinematics(:,290);
NW2_R_Ankle_Velocity_x = Kinematics(:,291);
NW2_R_Ankle_Velocity_y = Kinematics(:,292);
NW2_R_Ankle_Velocity_z = Kinematics(:,293);
NW2_R_Knee_Angle_x = Kinematics(:,294);
NW2_R_Knee_Angle_y = Kinematics(:,295);
NW2_R_Knee_Angle_z = Kinematics(:,296);
NW2_R_Knee_Velocity_x = Kinematics(:,297);
NW2_R_Knee_Velocity_y = Kinematics(:,298);
NW2_R_Knee_Velocity_z = Kinematics(:,299);
NW2_R_Hip_Angle_x = Kinematics(:,300);
NW2_R_Hip_Angle_y = Kinematics(:,301);
NW2_R_Hip_Angle_z = Kinematics(:,302);
NW2_R_Hip_Velocity_x = Kinematics(:,303);
NW2_R_Hip_Velocity_y = Kinematics(:,304);
NW2_R_Hip_Velocity_z = Kinematics(:,305);

NW3_L_Ankle_Angle_x = Kinematics(:,307);
NW3_L_Ankle_Angle_y = Kinematics(:,308);
NW3_L_Ankle_Angle_z = Kinematics(:,309);
NW3_L_Ankle_Velocity_x = Kinematics(:,310);
NW3_L_Ankle_Velocity_y = Kinematics(:,311);
NW3_L_Ankle_Velocity_z = Kinematics(:,312);
NW3_L_Knee_Angle_x = Kinematics(:,313);
NW3_L_Knee_Angle_y = Kinematics(:,314);
NW3_L_Knee_Angle_z = Kinematics(:,315);
NW3_L_Knee_Velocity_x = Kinematics(:,316);
NW3_L_Knee_Velocity_y = Kinematics(:,317);
NW3_L_Knee_Velocity_z = Kinematics(:,318);
NW3_L_Hip_Angle_x = Kinematics(:,319);
NW3_L_Hip_Angle_y = Kinematics(:,320);
NW3_L_Hip_Angle_z = Kinematics(:,321);
NW3_L_Hip_Velocity_x = Kinematics(:,322);
NW3_L_Hip_Velocity_y = Kinematics(:,323);
```

```
NW3_L_Hip_Velocity_z = Kinematics(:,324);
NW3_R_Ankle_Angle_x = Kinematics(:,325);
NW3_R_Ankle_Angle_y = Kinematics(:,326);
NW3_R_Ankle_Angle_z = Kinematics(:,327);
NW3_R_Ankle_Velocity_x = Kinematics(:,328);
NW3_R_Ankle_Velocity_y = Kinematics(:,329);
NW3_R_Ankle_Velocity_z = Kinematics(:,330);
NW3_R_Knee_Angle_x = Kinematics(:,331);
NW3_R_Knee_Angle_y = Kinematics(:,332);
NW3_R_Knee_Angle_z = Kinematics(:,333);
NW3_R_Knee_Velocity_x = Kinematics(:,334);
NW3_R_Knee_Velocity_y = Kinematics(:,335);
NW3_R_Knee_Velocity_z = Kinematics(:,336);
NW3_R_Hip_Angle_x = Kinematics(:,337);
NW3_R_Hip_Angle_y = Kinematics(:,338);
NW3_R_Hip_Angle_z = Kinematics(:,339);
NW3_R_Hip_Velocity_x = Kinematics(:,340);
NW3_R_Hip_Velocity_y = Kinematics(:,341);
NW3_R_Hip_Velocity_z = Kinematics(:,342);


% Now import the event labels and store it as a matrix
Events = xlsread('6_b.xlsx'); %This is b

% Create variables or matrices for each event time needed
ID1_True_Drop = Events(1,3);
ID1_Post_350 = Events(1,8);
ID2_True_Drop = Events(1,11);
ID2_Post_350 = Events(1,16);
ID3_True_Drop = Events(1,19);
ID3_Post_350 = Events(1,24);

IPD1_True_Drop = Events(1,27);
IPD1_Post_350 = Events(1,32);
IPD2_True_Drop = Events(1,35);
IPD2_Post_350 = Events(1,40);
IPD3_True_Drop = Events(1,43);
IPD3_Post_350 = Events(1,48);

NW1_HS = Events(1,51);
NW1_Post_350 = Events(1,53);
NW2_HS = Events(1,55);
NW2_Post_350 = Events(1,57);
NW3_HS = Events(1,59);
NW3_Post_350 = Events(1,61);
```

```
% Create an output matrix to store all the values, each cell starts as zero
B = zeros(9,54);

% Compute kinematic variables of interest
   % These include peak angle, time to peak angle, and peak velocity
   % This is for each joint (ankle,knee,hip) + all 3 planes(xyz)

% The window of time for peak angle and velocity is between the heel strike of the drop and
350ms after
 %Angles
 %Left side
 %ID1
 ID1_L_Ankle_Angle_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Ankle_Angle_x_FrameN = round((ID1_Post_350/.005)+1);

 % Create subset that contains only data from heelstrike/true drop to 350 ms after
 ID1_L_Ankle_Angle_x_subset =
ID1_L_Ankle_Angle_x(ID1_L_Ankle_Angle_x_Frame1:ID1_L_Ankle_Angle_x_FrameN);

 % Calculate the peak angle between heelstrike/true drop and the 350ms after
 maxAngle_ID1_L_Ankle_Angle_x = max(ID1_L_Ankle_Angle_x_subset);
 minAngle_ID1_L_Ankle_Angle_x = min(ID1_L_Ankle_Angle_x_subset);
 if maxAngle_ID1_L_Ankle_Angle_x >(abs(minAngle_ID1_L_Ankle_Angle_x))
    pAngle_ID1_L_Ankle_Angle_x = maxAngle_ID1_L_Ankle_Angle_x;
 else
    pAngle_ID1_L_Ankle_Angle_x = minAngle_ID1_L_Ankle_Angle_x;
 end

 % Calculate the time to peak: compute the time between onset and peak EMG
 % First need to return the index of the peak value
 pAnlge_idx_ID1_L_Ankle_Angle_x = find(ID1_L_Ankle_Angle_x ==
pAngle_ID1_L_Ankle_Angle_x);
 t2p_ID1_L_Ankle_Angle_x = (((pAnlge_idx_ID1_L_Ankle_Angle_x-1)*0.005)-
ID1_True_Drop);
 B(1,1) = pAngle_ID1_L_Ankle_Angle_x;
 B(1,2) = t2p_ID1_L_Ankle_Angle_x(1,1);


 ID1_L_Ankle_Angle_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Ankle_Angle_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Ankle_Angle_y_subset =
ID1_L_Ankle_Angle_y(ID1_L_Ankle_Angle_y_Frame1:ID1_L_Ankle_Angle_y_FrameN);
 maxAngle_ID1_L_Ankle_Angle_y = max(ID1_L_Ankle_Angle_y_subset);
 minAngle_ID1_L_Ankle_Angle_y = min(ID1_L_Ankle_Angle_y_subset);
 if maxAngle_ID1_L_Ankle_Angle_y >(abs(minAngle_ID1_L_Ankle_Angle_y))
    pAngle_ID1_L_Ankle_Angle_y = maxAngle_ID1_L_Ankle_Angle_y;
```

```
 else
    pAngle_ID1_L_Ankle_Angle_y = minAngle_ID1_L_Ankle_Angle_y;
 end
 pAnlge_idx_ID1_L_Ankle_Angle_y = find(ID1_L_Ankle_Angle_y ==
pAngle_ID1_L_Ankle_Angle_y);
 t2p_ID1_L_Ankle_Angle_y = (((pAnlge_idx_ID1_L_Ankle_Angle_y-1)*0.005)-
ID1_True_Drop);
 B(1,3) = pAngle_ID1_L_Ankle_Angle_y;
 B(1,4) = t2p_ID1_L_Ankle_Angle_y;

 ID1_L_Ankle_Angle_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Ankle_Angle_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Ankle_Angle_z_subset =
ID1_L_Ankle_Angle_z(ID1_L_Ankle_Angle_z_Frame1:ID1_L_Ankle_Angle_z_FrameN);
 maxAngle_ID1_L_Ankle_Angle_z = max(ID1_L_Ankle_Angle_z_subset);
 minAngle_ID1_L_Ankle_Angle_z = min(ID1_L_Ankle_Angle_z_subset);
 if maxAngle_ID1_L_Ankle_Angle_z >(abs(minAngle_ID1_L_Ankle_Angle_z))
    pAngle_ID1_L_Ankle_Angle_z = maxAngle_ID1_L_Ankle_Angle_z;
 else
    pAngle_ID1_L_Ankle_Angle_z = minAngle_ID1_L_Ankle_Angle_z;
 end
 pAnlge_idx_ID1_L_Ankle_Angle_z = find(ID1_L_Ankle_Angle_z ==
pAngle_ID1_L_Ankle_Angle_z);
 t2p_ID1_L_Ankle_Angle_z = (((pAnlge_idx_ID1_L_Ankle_Angle_z-1)*0.005)-
ID1_True_Drop);
 B(1,5) = pAngle_ID1_L_Ankle_Angle_z;
 B(1,6) = t2p_ID1_L_Ankle_Angle_z;

 ID1_L_Knee_Angle_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Knee_Angle_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Knee_Angle_x_subset =
ID1_L_Knee_Angle_x(ID1_L_Knee_Angle_x_Frame1:ID1_L_Knee_Angle_x_FrameN);
 maxAngle_ID1_L_Knee_Angle_x = max(ID1_L_Knee_Angle_x_subset);
 minAngle_ID1_L_Knee_Angle_x = min(ID1_L_Knee_Angle_x_subset);
 if maxAngle_ID1_L_Knee_Angle_x >(abs(minAngle_ID1_L_Knee_Angle_x))
    pAngle_ID1_L_Knee_Angle_x = maxAngle_ID1_L_Knee_Angle_x;
 else
    pAngle_ID1_L_Knee_Angle_x = minAngle_ID1_L_Knee_Angle_x;
 end
 pAnlge_idx_ID1_L_Knee_Angle_x = find(ID1_L_Knee_Angle_x ==
pAngle_ID1_L_Knee_Angle_x);
 t2p_ID1_L_Knee_Angle_x = (((pAnlge_idx_ID1_L_Knee_Angle_x-1)*0.005)-
ID1_True_Drop);
 B(1,7) = pAngle_ID1_L_Knee_Angle_x;
 B(1,8) = t2p_ID1_L_Knee_Angle_x;
```

```
 ID1_L_Knee_Angle_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Knee_Angle_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Knee_Angle_y_subset =
ID1_L_Knee_Angle_y(ID1_L_Knee_Angle_y_Frame1:ID1_L_Knee_Angle_y_FrameN);
 maxAngle_ID1_L_Knee_Angle_y = max(ID1_L_Knee_Angle_y_subset);
 minAngle_ID1_L_Knee_Angle_y = min(ID1_L_Knee_Angle_y_subset);
 if maxAngle_ID1_L_Knee_Angle_y >(abs(minAngle_ID1_L_Knee_Angle_y))
    pAngle_ID1_L_Knee_Angle_y = maxAngle_ID1_L_Knee_Angle_y;
 else
    pAngle_ID1_L_Knee_Angle_y = minAngle_ID1_L_Knee_Angle_y;
 end
 pAnlge_idx_ID1_L_Knee_Angle_y = find(ID1_L_Knee_Angle_y ==
pAngle_ID1_L_Knee_Angle_y);
 t2p_ID1_L_Knee_Angle_y = (((pAnlge_idx_ID1_L_Knee_Angle_y-1)*0.005)-
ID1_True_Drop);
 B(1,9) = pAngle_ID1_L_Knee_Angle_y;
 B(1,10) = t2p_ID1_L_Knee_Angle_y;

 ID1_L_Knee_Angle_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Knee_Angle_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Knee_Angle_z_subset =
ID1_L_Knee_Angle_z(ID1_L_Knee_Angle_z_Frame1:ID1_L_Knee_Angle_z_FrameN);
 maxAngle_ID1_L_Knee_Angle_z = max(ID1_L_Knee_Angle_z_subset);
 minAngle_ID1_L_Knee_Angle_z = min(ID1_L_Knee_Angle_z_subset);
 if maxAngle_ID1_L_Knee_Angle_z >(abs(minAngle_ID1_L_Knee_Angle_z))
    pAngle_ID1_L_Knee_Angle_z = maxAngle_ID1_L_Knee_Angle_z;
 else
    pAngle_ID1_L_Knee_Angle_z = minAngle_ID1_L_Knee_Angle_z;
 end
 pAnlge_idx_ID1_L_Knee_Angle_z = find(ID1_L_Knee_Angle_z ==
pAngle_ID1_L_Knee_Angle_z);
 t2p_ID1_L_Knee_Angle_z = (((pAnlge_idx_ID1_L_Knee_Angle_z-1)*0.005)-
ID1_True_Drop);
 B(1,11) = pAngle_ID1_L_Knee_Angle_z;
 B(1,12) = t2p_ID1_L_Knee_Angle_z;

 ID1_L_Hip_Angle_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Hip_Angle_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Hip_Angle_x_subset =
ID1_L_Hip_Angle_x(ID1_L_Hip_Angle_x_Frame1:ID1_L_Hip_Angle_x_FrameN);
 maxAngle_ID1_L_Hip_Angle_x = max(ID1_L_Hip_Angle_x_subset);
 minAngle_ID1_L_Hip_Angle_x = min(ID1_L_Hip_Angle_x_subset);
 if maxAngle_ID1_L_Hip_Angle_x >(abs(minAngle_ID1_L_Hip_Angle_x))
    pAngle_ID1_L_Hip_Angle_x = maxAngle_ID1_L_Hip_Angle_x;
 else
    pAngle_ID1_L_Hip_Angle_x = minAngle_ID1_L_Hip_Angle_x;
```

```
 end
 pAnlge_idx_ID1_L_Hip_Angle_x = find(ID1_L_Hip_Angle_x ==
pAngle_ID1_L_Hip_Angle_x);
 t2p_ID1_L_Hip_Angle_x = (((pAnlge_idx_ID1_L_Hip_Angle_x-1)*0.005)-ID1_True_Drop);
 B(1,13) = pAngle_ID1_L_Hip_Angle_x;
 B(1,14) = t2p_ID1_L_Hip_Angle_x;

 ID1_L_Hip_Angle_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Hip_Angle_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Hip_Angle_y_subset =
ID1_L_Hip_Angle_y(ID1_L_Hip_Angle_y_Frame1:ID1_L_Hip_Angle_y_FrameN);
 maxAngle_ID1_L_Hip_Angle_y = max(ID1_L_Hip_Angle_y_subset);
 minAngle_ID1_L_Hip_Angle_y = min(ID1_L_Hip_Angle_y_subset);
 if maxAngle_ID1_L_Hip_Angle_y >(abs(minAngle_ID1_L_Hip_Angle_y))
    pAngle_ID1_L_Hip_Angle_y = maxAngle_ID1_L_Hip_Angle_y;
 else
    pAngle_ID1_L_Hip_Angle_y = minAngle_ID1_L_Hip_Angle_y;
 end
 pAnlge_idx_ID1_L_Hip_Angle_y = find(ID1_L_Hip_Angle_y ==
pAngle_ID1_L_Hip_Angle_y);
 t2p_ID1_L_Hip_Angle_y = (((pAnlge_idx_ID1_L_Hip_Angle_y-1)*0.005)-ID1_True_Drop);
 B(1,15) = pAngle_ID1_L_Hip_Angle_y;
 B(1,16) = t2p_ID1_L_Hip_Angle_y;

 ID1_L_Hip_Angle_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Hip_Angle_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Hip_Angle_z_subset =
ID1_L_Hip_Angle_z(ID1_L_Hip_Angle_z_Frame1:ID1_L_Hip_Angle_z_FrameN);
 maxAngle_ID1_L_Hip_Angle_z = max(ID1_L_Hip_Angle_z_subset);
 minAngle_ID1_L_Hip_Angle_z = min(ID1_L_Hip_Angle_z_subset);
 if maxAngle_ID1_L_Hip_Angle_z >(abs(minAngle_ID1_L_Hip_Angle_z))
    pAngle_ID1_L_Hip_Angle_z = maxAngle_ID1_L_Hip_Angle_z;
 else
    pAngle_ID1_L_Hip_Angle_z = minAngle_ID1_L_Hip_Angle_z;
 end
 pAnlge_idx_ID1_L_Hip_Angle_z = find(ID1_L_Hip_Angle_z ==
pAngle_ID1_L_Hip_Angle_z);
 t2p_ID1_L_Hip_Angle_z = (((pAnlge_idx_ID1_L_Hip_Angle_z-1)*0.005)-ID1_True_Drop);
 B(1,17) = pAngle_ID1_L_Hip_Angle_z;
 B(1,18) = t2p_ID1_L_Hip_Angle_z;

 % ID2
 ID2_L_Ankle_Angle_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Ankle_Angle_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Ankle_Angle_x_subset =
ID2_L_Ankle_Angle_x(ID2_L_Ankle_Angle_x_Frame1:ID2_L_Ankle_Angle_x_FrameN);
```

```
 maxAngle_ID2_L_Ankle_Angle_x = max(ID2_L_Ankle_Angle_x_subset);
 minAngle_ID2_L_Ankle_Angle_x = min(ID2_L_Ankle_Angle_x_subset);
 if maxAngle_ID2_L_Ankle_Angle_x >(abs(minAngle_ID2_L_Ankle_Angle_x))
    pAngle_ID2_L_Ankle_Angle_x = maxAngle_ID2_L_Ankle_Angle_x;
 else
    pAngle_ID2_L_Ankle_Angle_x = minAngle_ID2_L_Ankle_Angle_x;
 end
 pAnlge_idx_ID2_L_Ankle_Angle_x = find(ID2_L_Ankle_Angle_x ==
pAngle_ID2_L_Ankle_Angle_x);
 t2p_ID2_L_Ankle_Angle_x = (((pAnlge_idx_ID2_L_Ankle_Angle_x-1)*0.005)-
ID2_True_Drop);


 ID2_L_Ankle_Angle_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Ankle_Angle_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Ankle_Angle_y_subset =
ID2_L_Ankle_Angle_y(ID2_L_Ankle_Angle_y_Frame1:ID2_L_Ankle_Angle_y_FrameN);
 maxAngle_ID2_L_Ankle_Angle_y = max(ID2_L_Ankle_Angle_y_subset);
 minAngle_ID2_L_Ankle_Angle_y = min(ID2_L_Ankle_Angle_y_subset);
 if maxAngle_ID2_L_Ankle_Angle_y >(abs(minAngle_ID2_L_Ankle_Angle_y))
    pAngle_ID2_L_Ankle_Angle_y = maxAngle_ID2_L_Ankle_Angle_y;
 else
    pAngle_ID2_L_Ankle_Angle_y = minAngle_ID2_L_Ankle_Angle_y;
 end
 pAnlge_idx_ID2_L_Ankle_Angle_y = find(ID2_L_Ankle_Angle_y ==
pAngle_ID2_L_Ankle_Angle_y);
 t2p_ID2_L_Ankle_Angle_y = (((pAnlge_idx_ID2_L_Ankle_Angle_y-1)*0.005)-
ID2_True_Drop);


 ID2_L_Ankle_Angle_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Ankle_Angle_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Ankle_Angle_z_subset =
ID2_L_Ankle_Angle_z(ID2_L_Ankle_Angle_z_Frame1:ID2_L_Ankle_Angle_z_FrameN);
 maxAngle_ID2_L_Ankle_Angle_z = max(ID2_L_Ankle_Angle_z_subset);
 minAngle_ID2_L_Ankle_Angle_z = min(ID2_L_Ankle_Angle_z_subset);
 if maxAngle_ID2_L_Ankle_Angle_z >(abs(minAngle_ID2_L_Ankle_Angle_z))
    pAngle_ID2_L_Ankle_Angle_z = maxAngle_ID2_L_Ankle_Angle_z;
 else
    pAngle_ID2_L_Ankle_Angle_z = minAngle_ID2_L_Ankle_Angle_z;
 end
 pAnlge_idx_ID2_L_Ankle_Angle_z = find(ID2_L_Ankle_Angle_z ==
pAngle_ID2_L_Ankle_Angle_z);
 t2p_ID2_L_Ankle_Angle_z = (((pAnlge_idx_ID2_L_Ankle_Angle_z-1)*0.005)-
ID2_True_Drop);
```

```
 ID2_L_Knee_Angle_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Knee_Angle_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Knee_Angle_x_subset =
ID2_L_Knee_Angle_x(ID2_L_Knee_Angle_x_Frame1:ID2_L_Knee_Angle_x_FrameN);
 maxAngle_ID2_L_Knee_Angle_x = max(ID2_L_Knee_Angle_x_subset);
 minAngle_ID2_L_Knee_Angle_x = min(ID2_L_Knee_Angle_x_subset);
 if maxAngle_ID2_L_Knee_Angle_x >(abs(minAngle_ID2_L_Knee_Angle_x))
    pAngle_ID2_L_Knee_Angle_x = maxAngle_ID2_L_Knee_Angle_x;
 else
    pAngle_ID2_L_Knee_Angle_x = minAngle_ID2_L_Knee_Angle_x;
 end
 pAnlge_idx_ID2_L_Knee_Angle_x = find(ID2_L_Knee_Angle_x ==
pAngle_ID2_L_Knee_Angle_x);
 t2p_ID2_L_Knee_Angle_x = (((pAnlge_idx_ID2_L_Knee_Angle_x-1)*0.005)-
ID2_True_Drop);


 ID2_L_Knee_Angle_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Knee_Angle_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Knee_Angle_y_subset =
ID2_L_Knee_Angle_y(ID2_L_Knee_Angle_y_Frame1:ID2_L_Knee_Angle_y_FrameN);
 maxAngle_ID2_L_Knee_Angle_y = max(ID2_L_Knee_Angle_y_subset);
 minAngle_ID2_L_Knee_Angle_y = min(ID2_L_Knee_Angle_y_subset);
 if maxAngle_ID2_L_Knee_Angle_y >(abs(minAngle_ID2_L_Knee_Angle_y))
    pAngle_ID2_L_Knee_Angle_y = maxAngle_ID2_L_Knee_Angle_y;
 else
    pAngle_ID2_L_Knee_Angle_y = minAngle_ID2_L_Knee_Angle_y;
 end
 pAnlge_idx_ID2_L_Knee_Angle_y = find(ID2_L_Knee_Angle_y ==
pAngle_ID2_L_Knee_Angle_y);
 t2p_ID2_L_Knee_Angle_y = (((pAnlge_idx_ID2_L_Knee_Angle_y-1)*0.005)-
ID2_True_Drop);


 ID2_L_Knee_Angle_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Knee_Angle_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Knee_Angle_z_subset =
ID2_L_Knee_Angle_z(ID2_L_Knee_Angle_z_Frame1:ID2_L_Knee_Angle_z_FrameN);
 maxAngle_ID2_L_Knee_Angle_z = max(ID2_L_Knee_Angle_z_subset);
 minAngle_ID2_L_Knee_Angle_z = min(ID2_L_Knee_Angle_z_subset);
 if maxAngle_ID2_L_Knee_Angle_z >(abs(minAngle_ID2_L_Knee_Angle_z))
    pAngle_ID2_L_Knee_Angle_z = maxAngle_ID2_L_Knee_Angle_z;
 else
    pAngle_ID2_L_Knee_Angle_z = minAngle_ID2_L_Knee_Angle_z;
 end
```

```
 pAnlge_idx_ID2_L_Knee_Angle_z = find(ID2_L_Knee_Angle_z ==
pAngle_ID2_L_Knee_Angle_z);
 t2p_ID2_L_Knee_Angle_z = (((pAnlge_idx_ID2_L_Knee_Angle_z-1)*0.005)-
ID2_True_Drop);


 ID2_L_Hip_Angle_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Hip_Angle_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Hip_Angle_x_subset =
ID2_L_Hip_Angle_x(ID2_L_Hip_Angle_x_Frame1:ID2_L_Hip_Angle_x_FrameN);
 maxAngle_ID2_L_Hip_Angle_x = max(ID2_L_Hip_Angle_x_subset);
 minAngle_ID2_L_Hip_Angle_x = min(ID2_L_Hip_Angle_x_subset);
 if maxAngle_ID2_L_Hip_Angle_x >(abs(minAngle_ID2_L_Hip_Angle_x))
    pAngle_ID2_L_Hip_Angle_x = maxAngle_ID2_L_Hip_Angle_x;
 else
    pAngle_ID2_L_Hip_Angle_x = minAngle_ID2_L_Hip_Angle_x;
 end
 pAnlge_idx_ID2_L_Hip_Angle_x = find(ID2_L_Hip_Angle_x ==
pAngle_ID2_L_Hip_Angle_x);
 t2p_ID2_L_Hip_Angle_x = (((pAnlge_idx_ID2_L_Hip_Angle_x-1)*0.005)-ID2_True_Drop);


 ID2_L_Hip_Angle_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Hip_Angle_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Hip_Angle_y_subset =
ID2_L_Hip_Angle_y(ID2_L_Hip_Angle_y_Frame1:ID2_L_Hip_Angle_y_FrameN);
 maxAngle_ID2_L_Hip_Angle_y = max(ID2_L_Hip_Angle_y_subset);
 minAngle_ID2_L_Hip_Angle_y = min(ID2_L_Hip_Angle_y_subset);
 if maxAngle_ID2_L_Hip_Angle_y >(abs(minAngle_ID2_L_Hip_Angle_y))
    pAngle_ID2_L_Hip_Angle_y = maxAngle_ID2_L_Hip_Angle_y;
 else
    pAngle_ID2_L_Hip_Angle_y = minAngle_ID2_L_Hip_Angle_y;
 end
 pAnlge_idx_ID2_L_Hip_Angle_y = find(ID2_L_Hip_Angle_y ==
pAngle_ID2_L_Hip_Angle_y);
 t2p_ID2_L_Hip_Angle_y = (((pAnlge_idx_ID2_L_Hip_Angle_y-1)*0.005)-ID2_True_Drop);


 ID2_L_Hip_Angle_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Hip_Angle_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Hip_Angle_z_subset =
ID2_L_Hip_Angle_z(ID2_L_Hip_Angle_z_Frame1:ID2_L_Hip_Angle_z_FrameN);
 maxAngle_ID2_L_Hip_Angle_z = max(ID2_L_Hip_Angle_z_subset);
 minAngle_ID2_L_Hip_Angle_z = min(ID2_L_Hip_Angle_z_subset);
 if maxAngle_ID2_L_Hip_Angle_z >(abs(minAngle_ID2_L_Hip_Angle_z))
    pAngle_ID2_L_Hip_Angle_z = maxAngle_ID2_L_Hip_Angle_z;
```

```
 else
    pAngle_ID2_L_Hip_Angle_z = minAngle_ID2_L_Hip_Angle_z;
 end
 pAnlge_idx_ID2_L_Hip_Angle_z = find(ID2_L_Hip_Angle_z ==
pAngle_ID2_L_Hip_Angle_z);
 t2p_ID2_L_Hip_Angle_z = (((pAnlge_idx_ID2_L_Hip_Angle_z-1)*0.005)-ID2_True_Drop);

 %ID3
 ID3_L_Ankle_Angle_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Ankle_Angle_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Ankle_Angle_x_subset =
ID3_L_Ankle_Angle_x(ID3_L_Ankle_Angle_x_Frame1:ID3_L_Ankle_Angle_x_FrameN);
 maxAngle_ID3_L_Ankle_Angle_x = max(ID3_L_Ankle_Angle_x_subset);
 minAngle_ID3_L_Ankle_Angle_x = min(ID3_L_Ankle_Angle_x_subset);
 if maxAngle_ID3_L_Ankle_Angle_x >(abs(minAngle_ID3_L_Ankle_Angle_x))
    pAngle_ID3_L_Ankle_Angle_x = maxAngle_ID3_L_Ankle_Angle_x;
 else
    pAngle_ID3_L_Ankle_Angle_x = minAngle_ID3_L_Ankle_Angle_x;
 end
 pAnlge_idx_ID3_L_Ankle_Angle_x = find(ID3_L_Ankle_Angle_x ==
pAngle_ID3_L_Ankle_Angle_x);
 t2p_ID3_L_Ankle_Angle_x = (((pAnlge_idx_ID3_L_Ankle_Angle_x-1)*0.005)-
ID3_True_Drop);


 ID3_L_Ankle_Angle_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Ankle_Angle_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Ankle_Angle_y_subset =
ID3_L_Ankle_Angle_y(ID3_L_Ankle_Angle_y_Frame1:ID3_L_Ankle_Angle_y_FrameN);
 maxAngle_ID3_L_Ankle_Angle_y = max(ID3_L_Ankle_Angle_y_subset);
 minAngle_ID3_L_Ankle_Angle_y = min(ID3_L_Ankle_Angle_y_subset);
 if maxAngle_ID3_L_Ankle_Angle_y >(abs(minAngle_ID3_L_Ankle_Angle_y))
    pAngle_ID3_L_Ankle_Angle_y = maxAngle_ID3_L_Ankle_Angle_y;
 else
    pAngle_ID3_L_Ankle_Angle_y = minAngle_ID3_L_Ankle_Angle_y;
 end
 pAnlge_idx_ID3_L_Ankle_Angle_y = find(ID3_L_Ankle_Angle_y ==
pAngle_ID3_L_Ankle_Angle_y);
 t2p_ID3_L_Ankle_Angle_y = (((pAnlge_idx_ID3_L_Ankle_Angle_y-1)*0.005)-
ID3_True_Drop);


 ID3_L_Ankle_Angle_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Ankle_Angle_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Ankle_Angle_z_subset =
ID3_L_Ankle_Angle_z(ID3_L_Ankle_Angle_z_Frame1:ID3_L_Ankle_Angle_z_FrameN);
```

```
 maxAngle_ID3_L_Ankle_Angle_z = max(ID3_L_Ankle_Angle_z_subset);
 minAngle_ID3_L_Ankle_Angle_z = min(ID3_L_Ankle_Angle_z_subset);
 if maxAngle_ID3_L_Ankle_Angle_z >(abs(minAngle_ID3_L_Ankle_Angle_z))
    pAngle_ID3_L_Ankle_Angle_z = maxAngle_ID3_L_Ankle_Angle_z;
 else
    pAngle_ID3_L_Ankle_Angle_z = minAngle_ID3_L_Ankle_Angle_z;
 end
 pAnlge_idx_ID3_L_Ankle_Angle_z = find(ID3_L_Ankle_Angle_z ==
pAngle_ID3_L_Ankle_Angle_z);
 t2p_ID3_L_Ankle_Angle_z = (((pAnlge_idx_ID3_L_Ankle_Angle_z-1)*0.005)-
ID3_True_Drop);


 ID3_L_Knee_Angle_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Knee_Angle_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Knee_Angle_x_subset =
ID3_L_Knee_Angle_x(ID3_L_Knee_Angle_x_Frame1:ID3_L_Knee_Angle_x_FrameN);
 maxAngle_ID3_L_Knee_Angle_x = max(ID3_L_Knee_Angle_x_subset);
 minAngle_ID3_L_Knee_Angle_x = min(ID3_L_Knee_Angle_x_subset);
 if maxAngle_ID3_L_Knee_Angle_x >(abs(minAngle_ID3_L_Knee_Angle_x))
    pAngle_ID3_L_Knee_Angle_x = maxAngle_ID3_L_Knee_Angle_x;
 else
    pAngle_ID3_L_Knee_Angle_x = minAngle_ID3_L_Knee_Angle_x;
 end
 pAnlge_idx_ID3_L_Knee_Angle_x = find(ID3_L_Knee_Angle_x ==
pAngle_ID3_L_Knee_Angle_x);
 t2p_ID3_L_Knee_Angle_x = (((pAnlge_idx_ID3_L_Knee_Angle_x-1)*0.005)-
ID3_True_Drop);


 ID3_L_Knee_Angle_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Knee_Angle_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Knee_Angle_y_subset =
ID3_L_Knee_Angle_y(ID3_L_Knee_Angle_y_Frame1:ID3_L_Knee_Angle_y_FrameN);
 maxAngle_ID3_L_Knee_Angle_y = max(ID3_L_Knee_Angle_y_subset);
 minAngle_ID3_L_Knee_Angle_y = min(ID3_L_Knee_Angle_y_subset);
 if maxAngle_ID3_L_Knee_Angle_y >(abs(minAngle_ID3_L_Knee_Angle_y))
    pAngle_ID3_L_Knee_Angle_y = maxAngle_ID3_L_Knee_Angle_y;
 else
    pAngle_ID3_L_Knee_Angle_y = minAngle_ID3_L_Knee_Angle_y;
 end
 pAnlge_idx_ID3_L_Knee_Angle_y = find(ID3_L_Knee_Angle_y ==
pAngle_ID3_L_Knee_Angle_y);
 t2p_ID3_L_Knee_Angle_y = (((pAnlge_idx_ID3_L_Knee_Angle_y-1)*0.005)-
ID3_True_Drop);
```

```
 ID3_L_Knee_Angle_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Knee_Angle_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Knee_Angle_z_subset =
ID3_L_Knee_Angle_z(ID3_L_Knee_Angle_z_Frame1:ID3_L_Knee_Angle_z_FrameN);
 maxAngle_ID3_L_Knee_Angle_z = max(ID3_L_Knee_Angle_z_subset);
 minAngle_ID3_L_Knee_Angle_z = min(ID3_L_Knee_Angle_z_subset);
 if maxAngle_ID3_L_Knee_Angle_z >(abs(minAngle_ID3_L_Knee_Angle_z))
    pAngle_ID3_L_Knee_Angle_z = maxAngle_ID3_L_Knee_Angle_z;
 else
    pAngle_ID3_L_Knee_Angle_z = minAngle_ID3_L_Knee_Angle_z;
 end
 pAnlge_idx_ID3_L_Knee_Angle_z = find(ID3_L_Knee_Angle_z ==
pAngle_ID3_L_Knee_Angle_z);
 t2p_ID3_L_Knee_Angle_z = (((pAnlge_idx_ID3_L_Knee_Angle_z-1)*0.005)-
ID3_True_Drop);


 ID3_L_Hip_Angle_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Hip_Angle_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Hip_Angle_x_subset =
ID3_L_Hip_Angle_x(ID3_L_Hip_Angle_x_Frame1:ID3_L_Hip_Angle_x_FrameN);
 maxAngle_ID3_L_Hip_Angle_x = max(ID3_L_Hip_Angle_x_subset);
 minAngle_ID3_L_Hip_Angle_x = min(ID3_L_Hip_Angle_x_subset);
 if maxAngle_ID3_L_Hip_Angle_x >(abs(minAngle_ID3_L_Hip_Angle_x))
    pAngle_ID3_L_Hip_Angle_x = maxAngle_ID3_L_Hip_Angle_x;
 else
    pAngle_ID3_L_Hip_Angle_x = minAngle_ID3_L_Hip_Angle_x;
 end
 pAnlge_idx_ID3_L_Hip_Angle_x = find(ID3_L_Hip_Angle_x ==
pAngle_ID3_L_Hip_Angle_x);
 t2p_ID3_L_Hip_Angle_x = (((pAnlge_idx_ID3_L_Hip_Angle_x-1)*0.005)-ID3_True_Drop);


 ID3_L_Hip_Angle_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Hip_Angle_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Hip_Angle_y_subset =
ID3_L_Hip_Angle_y(ID3_L_Hip_Angle_y_Frame1:ID3_L_Hip_Angle_y_FrameN);
 maxAngle_ID3_L_Hip_Angle_y = max(ID3_L_Hip_Angle_y_subset);
 minAngle_ID3_L_Hip_Angle_y = min(ID3_L_Hip_Angle_y_subset);
 if maxAngle_ID3_L_Hip_Angle_y >(abs(minAngle_ID3_L_Hip_Angle_y))
    pAngle_ID3_L_Hip_Angle_y = maxAngle_ID3_L_Hip_Angle_y;
 else
    pAngle_ID3_L_Hip_Angle_y = minAngle_ID3_L_Hip_Angle_y;
 end
```

```
 pAnlge_idx_ID3_L_Hip_Angle_y = find(ID3_L_Hip_Angle_y ==
pAngle_ID3_L_Hip_Angle_y);
 t2p_ID3_L_Hip_Angle_y = (((pAnlge_idx_ID3_L_Hip_Angle_y-1)*0.005)-ID3_True_Drop);


 ID3_L_Hip_Angle_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Hip_Angle_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Hip_Angle_z_subset =
ID3_L_Hip_Angle_z(ID3_L_Hip_Angle_z_Frame1:ID3_L_Hip_Angle_z_FrameN);
 maxAngle_ID3_L_Hip_Angle_z = max(ID3_L_Hip_Angle_z_subset);
 minAngle_ID3_L_Hip_Angle_z = min(ID3_L_Hip_Angle_z_subset);
 if maxAngle_ID3_L_Hip_Angle_z >(abs(minAngle_ID3_L_Hip_Angle_z))
    pAngle_ID3_L_Hip_Angle_z = maxAngle_ID3_L_Hip_Angle_z;
 else
    pAngle_ID3_L_Hip_Angle_z = minAngle_ID3_L_Hip_Angle_z;
 end
 pAnlge_idx_ID3_L_Hip_Angle_z = find(ID3_L_Hip_Angle_z ==
pAngle_ID3_L_Hip_Angle_z);
 t2p_ID3_L_Hip_Angle_z = (((pAnlge_idx_ID3_L_Hip_Angle_z-1)*0.005)-ID3_True_Drop);



 %IPD1
  IPD1_L_Ankle_Angle_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Ankle_Angle_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Ankle_Angle_x_subset =
IPD1_L_Ankle_Angle_x(IPD1_L_Ankle_Angle_x_Frame1:IPD1_L_Ankle_Angle_x_FrameN);
 maxAngle_IPD1_L_Ankle_Angle_x = max(IPD1_L_Ankle_Angle_x_subset);
 minAngle_IPD1_L_Ankle_Angle_x = min(IPD1_L_Ankle_Angle_x_subset);
 if maxAngle_IPD1_L_Ankle_Angle_x >(abs(minAngle_IPD1_L_Ankle_Angle_x))
    pAngle_IPD1_L_Ankle_Angle_x = maxAngle_IPD1_L_Ankle_Angle_x;
 else
    pAngle_IPD1_L_Ankle_Angle_x = minAngle_IPD1_L_Ankle_Angle_x;
 end
 pAnlge_idx_IPD1_L_Ankle_Angle_x = find(IPD1_L_Ankle_Angle_x ==
pAngle_IPD1_L_Ankle_Angle_x);
 t2p_IPD1_L_Ankle_Angle_x = (((pAnlge_idx_IPD1_L_Ankle_Angle_x-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Ankle_Angle_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Ankle_Angle_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Ankle_Angle_y_subset =
IPD1_L_Ankle_Angle_y(IPD1_L_Ankle_Angle_y_Frame1:IPD1_L_Ankle_Angle_y_FrameN);
 maxAngle_IPD1_L_Ankle_Angle_y = max(IPD1_L_Ankle_Angle_y_subset);
 minAngle_IPD1_L_Ankle_Angle_y = min(IPD1_L_Ankle_Angle_y_subset);
```

```
 if maxAngle_IPD1_L_Ankle_Angle_y >(abs(minAngle_IPD1_L_Ankle_Angle_y))
    pAngle_IPD1_L_Ankle_Angle_y = maxAngle_IPD1_L_Ankle_Angle_y;
 else
    pAngle_IPD1_L_Ankle_Angle_y = minAngle_IPD1_L_Ankle_Angle_y;
 end
 pAnlge_idx_IPD1_L_Ankle_Angle_y = find(IPD1_L_Ankle_Angle_y ==
pAngle_IPD1_L_Ankle_Angle_y);
 t2p_IPD1_L_Ankle_Angle_y = (((pAnlge_idx_IPD1_L_Ankle_Angle_y-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Ankle_Angle_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Ankle_Angle_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Ankle_Angle_z_subset =
IPD1_L_Ankle_Angle_z(IPD1_L_Ankle_Angle_z_Frame1:IPD1_L_Ankle_Angle_z_FrameN);
 maxAngle_IPD1_L_Ankle_Angle_z = max(IPD1_L_Ankle_Angle_z_subset);
 minAngle_IPD1_L_Ankle_Angle_z = min(IPD1_L_Ankle_Angle_z_subset);
 if maxAngle_IPD1_L_Ankle_Angle_z >(abs(minAngle_IPD1_L_Ankle_Angle_z))
    pAngle_IPD1_L_Ankle_Angle_z = maxAngle_IPD1_L_Ankle_Angle_z;
 else
    pAngle_IPD1_L_Ankle_Angle_z = minAngle_IPD1_L_Ankle_Angle_z;
 end
 pAnlge_idx_IPD1_L_Ankle_Angle_z = find(IPD1_L_Ankle_Angle_z ==
pAngle_IPD1_L_Ankle_Angle_z);
 t2p_IPD1_L_Ankle_Angle_z = (((pAnlge_idx_IPD1_L_Ankle_Angle_z-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Knee_Angle_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Knee_Angle_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Knee_Angle_x_subset =
IPD1_L_Knee_Angle_x(IPD1_L_Knee_Angle_x_Frame1:IPD1_L_Knee_Angle_x_FrameN);
 maxAngle_IPD1_L_Knee_Angle_x = max(IPD1_L_Knee_Angle_x_subset);
 minAngle_IPD1_L_Knee_Angle_x = min(IPD1_L_Knee_Angle_x_subset);
 if maxAngle_IPD1_L_Knee_Angle_x >(abs(minAngle_IPD1_L_Knee_Angle_x))
    pAngle_IPD1_L_Knee_Angle_x = maxAngle_IPD1_L_Knee_Angle_x;
 else
    pAngle_IPD1_L_Knee_Angle_x = minAngle_IPD1_L_Knee_Angle_x;
 end
 pAnlge_idx_IPD1_L_Knee_Angle_x = find(IPD1_L_Knee_Angle_x ==
pAngle_IPD1_L_Knee_Angle_x);
 t2p_IPD1_L_Knee_Angle_x = (((pAnlge_idx_IPD1_L_Knee_Angle_x-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Knee_Angle_y_Frame1 = round((IPD1_True_Drop/.005)+1);
```

```
 IPD1_L_Knee_Angle_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Knee_Angle_y_subset =
IPD1_L_Knee_Angle_y(IPD1_L_Knee_Angle_y_Frame1:IPD1_L_Knee_Angle_y_FrameN);
 maxAngle_IPD1_L_Knee_Angle_y = max(IPD1_L_Knee_Angle_y_subset);
 minAngle_IPD1_L_Knee_Angle_y = min(IPD1_L_Knee_Angle_y_subset);
 if maxAngle_IPD1_L_Knee_Angle_y >(abs(minAngle_IPD1_L_Knee_Angle_y))
    pAngle_IPD1_L_Knee_Angle_y = maxAngle_IPD1_L_Knee_Angle_y;
 else
    pAngle_IPD1_L_Knee_Angle_y = minAngle_IPD1_L_Knee_Angle_y;
 end
 pAnlge_idx_IPD1_L_Knee_Angle_y = find(IPD1_L_Knee_Angle_y ==
pAngle_IPD1_L_Knee_Angle_y);
 t2p_IPD1_L_Knee_Angle_y = (((pAnlge_idx_IPD1_L_Knee_Angle_y-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Knee_Angle_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Knee_Angle_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Knee_Angle_z_subset =
IPD1_L_Knee_Angle_z(IPD1_L_Knee_Angle_z_Frame1:IPD1_L_Knee_Angle_z_FrameN);
 maxAngle_IPD1_L_Knee_Angle_z = max(IPD1_L_Knee_Angle_z_subset);
 minAngle_IPD1_L_Knee_Angle_z = min(IPD1_L_Knee_Angle_z_subset);
 if maxAngle_IPD1_L_Knee_Angle_z >(abs(minAngle_IPD1_L_Knee_Angle_z))
    pAngle_IPD1_L_Knee_Angle_z = maxAngle_IPD1_L_Knee_Angle_z;
 else
    pAngle_IPD1_L_Knee_Angle_z = minAngle_IPD1_L_Knee_Angle_z;
 end
 pAnlge_idx_IPD1_L_Knee_Angle_z = find(IPD1_L_Knee_Angle_z ==
pAngle_IPD1_L_Knee_Angle_z);
 t2p_IPD1_L_Knee_Angle_z = (((pAnlge_idx_IPD1_L_Knee_Angle_z-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Hip_Angle_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Hip_Angle_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Hip_Angle_x_subset =
IPD1_L_Hip_Angle_x(IPD1_L_Hip_Angle_x_Frame1:IPD1_L_Hip_Angle_x_FrameN);
 maxAngle_IPD1_L_Hip_Angle_x = max(IPD1_L_Hip_Angle_x_subset);
 minAngle_IPD1_L_Hip_Angle_x = min(IPD1_L_Hip_Angle_x_subset);
 if maxAngle_IPD1_L_Hip_Angle_x >(abs(minAngle_IPD1_L_Hip_Angle_x))
    pAngle_IPD1_L_Hip_Angle_x = maxAngle_IPD1_L_Hip_Angle_x;
 else
    pAngle_IPD1_L_Hip_Angle_x = minAngle_IPD1_L_Hip_Angle_x;
 end
 pAnlge_idx_IPD1_L_Hip_Angle_x = find(IPD1_L_Hip_Angle_x ==
pAngle_IPD1_L_Hip_Angle_x);
```

```
 t2p_IPD1_L_Hip_Angle_x = (((pAnlge_idx_IPD1_L_Hip_Angle_x-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Hip_Angle_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Hip_Angle_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Hip_Angle_y_subset =
IPD1_L_Hip_Angle_y(IPD1_L_Hip_Angle_y_Frame1:IPD1_L_Hip_Angle_y_FrameN);
 maxAngle_IPD1_L_Hip_Angle_y = max(IPD1_L_Hip_Angle_y_subset);
 minAngle_IPD1_L_Hip_Angle_y = min(IPD1_L_Hip_Angle_y_subset);
 if maxAngle_IPD1_L_Hip_Angle_y >(abs(minAngle_IPD1_L_Hip_Angle_y))
    pAngle_IPD1_L_Hip_Angle_y = maxAngle_IPD1_L_Hip_Angle_y;
 else
    pAngle_IPD1_L_Hip_Angle_y = minAngle_IPD1_L_Hip_Angle_y;
 end
 pAnlge_idx_IPD1_L_Hip_Angle_y = find(IPD1_L_Hip_Angle_y ==
pAngle_IPD1_L_Hip_Angle_y);
 t2p_IPD1_L_Hip_Angle_y = (((pAnlge_idx_IPD1_L_Hip_Angle_y-1)*0.005)-
IPD1_True_Drop);


 IPD1_L_Hip_Angle_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Hip_Angle_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Hip_Angle_z_subset =
IPD1_L_Hip_Angle_z(IPD1_L_Hip_Angle_z_Frame1:IPD1_L_Hip_Angle_z_FrameN);
 maxAngle_IPD1_L_Hip_Angle_z = max(IPD1_L_Hip_Angle_z_subset);
 minAngle_IPD1_L_Hip_Angle_z = min(IPD1_L_Hip_Angle_z_subset);
 if maxAngle_IPD1_L_Hip_Angle_z >(abs(minAngle_IPD1_L_Hip_Angle_z))
    pAngle_IPD1_L_Hip_Angle_z = maxAngle_IPD1_L_Hip_Angle_z;
 else
    pAngle_IPD1_L_Hip_Angle_z = minAngle_IPD1_L_Hip_Angle_z;
 end
 pAnlge_idx_IPD1_L_Hip_Angle_z = find(IPD1_L_Hip_Angle_z ==
pAngle_IPD1_L_Hip_Angle_z);
 t2p_IPD1_L_Hip_Angle_z = (((pAnlge_idx_IPD1_L_Hip_Angle_z-1)*0.005)-
IPD1_True_Drop);

 %IPD2
 IPD2_L_Ankle_Angle_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Ankle_Angle_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Ankle_Angle_x_subset =
IPD2_L_Ankle_Angle_x(IPD2_L_Ankle_Angle_x_Frame1:IPD2_L_Ankle_Angle_x_FrameN);
 maxAngle_IPD2_L_Ankle_Angle_x = max(IPD2_L_Ankle_Angle_x_subset);
 minAngle_IPD2_L_Ankle_Angle_x = min(IPD2_L_Ankle_Angle_x_subset);
 if maxAngle_IPD2_L_Ankle_Angle_x >(abs(minAngle_IPD2_L_Ankle_Angle_x))
    pAngle_IPD2_L_Ankle_Angle_x = maxAngle_IPD2_L_Ankle_Angle_x;
```

```
 else
    pAngle_IPD2_L_Ankle_Angle_x = minAngle_IPD2_L_Ankle_Angle_x;
 end
 pAnlge_idx_IPD2_L_Ankle_Angle_x = find(IPD2_L_Ankle_Angle_x ==
pAngle_IPD2_L_Ankle_Angle_x);
 t2p_IPD2_L_Ankle_Angle_x = (((pAnlge_idx_IPD2_L_Ankle_Angle_x-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Ankle_Angle_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Ankle_Angle_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Ankle_Angle_y_subset =
IPD2_L_Ankle_Angle_y(IPD2_L_Ankle_Angle_y_Frame1:IPD2_L_Ankle_Angle_y_FrameN);
 maxAngle_IPD2_L_Ankle_Angle_y = max(IPD2_L_Ankle_Angle_y_subset);
 minAngle_IPD2_L_Ankle_Angle_y = min(IPD2_L_Ankle_Angle_y_subset);
 if maxAngle_IPD2_L_Ankle_Angle_y >(abs(minAngle_IPD2_L_Ankle_Angle_y))
    pAngle_IPD2_L_Ankle_Angle_y = maxAngle_IPD2_L_Ankle_Angle_y;
 else
    pAngle_IPD2_L_Ankle_Angle_y = minAngle_IPD2_L_Ankle_Angle_y;
 end
 pAnlge_idx_IPD2_L_Ankle_Angle_y = find(IPD2_L_Ankle_Angle_y ==
pAngle_IPD2_L_Ankle_Angle_y);
 t2p_IPD2_L_Ankle_Angle_y = (((pAnlge_idx_IPD2_L_Ankle_Angle_y-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Ankle_Angle_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Ankle_Angle_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Ankle_Angle_z_subset =
IPD2_L_Ankle_Angle_z(IPD2_L_Ankle_Angle_z_Frame1:IPD2_L_Ankle_Angle_z_FrameN);
 maxAngle_IPD2_L_Ankle_Angle_z = max(IPD2_L_Ankle_Angle_z_subset);
 minAngle_IPD2_L_Ankle_Angle_z = min(IPD2_L_Ankle_Angle_z_subset);
 if maxAngle_IPD2_L_Ankle_Angle_z >(abs(minAngle_IPD2_L_Ankle_Angle_z))
    pAngle_IPD2_L_Ankle_Angle_z = maxAngle_IPD2_L_Ankle_Angle_z;
 else
    pAngle_IPD2_L_Ankle_Angle_z = minAngle_IPD2_L_Ankle_Angle_z;
 end
 pAnlge_idx_IPD2_L_Ankle_Angle_z = find(IPD2_L_Ankle_Angle_z ==
pAngle_IPD2_L_Ankle_Angle_z);
 t2p_IPD2_L_Ankle_Angle_z = (((pAnlge_idx_IPD2_L_Ankle_Angle_z-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Knee_Angle_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Knee_Angle_x_FrameN = round((IPD2_Post_350/.005)+1);
```

```
 IPD2_L_Knee_Angle_x_subset =
IPD2_L_Knee_Angle_x(IPD2_L_Knee_Angle_x_Frame1:IPD2_L_Knee_Angle_x_FrameN);
 maxAngle_IPD2_L_Knee_Angle_x = max(IPD2_L_Knee_Angle_x_subset);
 minAngle_IPD2_L_Knee_Angle_x = min(IPD2_L_Knee_Angle_x_subset);
 if maxAngle_IPD2_L_Knee_Angle_x >(abs(minAngle_IPD2_L_Knee_Angle_x))
    pAngle_IPD2_L_Knee_Angle_x = maxAngle_IPD2_L_Knee_Angle_x;
 else
    pAngle_IPD2_L_Knee_Angle_x = minAngle_IPD2_L_Knee_Angle_x;
 end
 pAnlge_idx_IPD2_L_Knee_Angle_x = find(IPD2_L_Knee_Angle_x ==
pAngle_IPD2_L_Knee_Angle_x);
 t2p_IPD2_L_Knee_Angle_x = (((pAnlge_idx_IPD2_L_Knee_Angle_x-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Knee_Angle_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Knee_Angle_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Knee_Angle_y_subset =
IPD2_L_Knee_Angle_y(IPD2_L_Knee_Angle_y_Frame1:IPD2_L_Knee_Angle_y_FrameN);
 maxAngle_IPD2_L_Knee_Angle_y = max(IPD2_L_Knee_Angle_y_subset);
 minAngle_IPD2_L_Knee_Angle_y = min(IPD2_L_Knee_Angle_y_subset);
 if maxAngle_IPD2_L_Knee_Angle_y >(abs(minAngle_IPD2_L_Knee_Angle_y))
    pAngle_IPD2_L_Knee_Angle_y = maxAngle_IPD2_L_Knee_Angle_y;
 else
    pAngle_IPD2_L_Knee_Angle_y = minAngle_IPD2_L_Knee_Angle_y;
 end
 pAnlge_idx_IPD2_L_Knee_Angle_y = find(IPD2_L_Knee_Angle_y ==
pAngle_IPD2_L_Knee_Angle_y);
 t2p_IPD2_L_Knee_Angle_y = (((pAnlge_idx_IPD2_L_Knee_Angle_y-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Knee_Angle_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Knee_Angle_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Knee_Angle_z_subset =
IPD2_L_Knee_Angle_z(IPD2_L_Knee_Angle_z_Frame1:IPD2_L_Knee_Angle_z_FrameN);
 maxAngle_IPD2_L_Knee_Angle_z = max(IPD2_L_Knee_Angle_z_subset);
 minAngle_IPD2_L_Knee_Angle_z = min(IPD2_L_Knee_Angle_z_subset);
 if maxAngle_IPD2_L_Knee_Angle_z >(abs(minAngle_IPD2_L_Knee_Angle_z))
    pAngle_IPD2_L_Knee_Angle_z = maxAngle_IPD2_L_Knee_Angle_z;
 else
    pAngle_IPD2_L_Knee_Angle_z = minAngle_IPD2_L_Knee_Angle_z;
 end
 pAnlge_idx_IPD2_L_Knee_Angle_z = find(IPD2_L_Knee_Angle_z ==
pAngle_IPD2_L_Knee_Angle_z);
```

```
 t2p_IPD2_L_Knee_Angle_z = (((pAnlge_idx_IPD2_L_Knee_Angle_z-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Hip_Angle_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Hip_Angle_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Hip_Angle_x_subset =
IPD2_L_Hip_Angle_x(IPD2_L_Hip_Angle_x_Frame1:IPD2_L_Hip_Angle_x_FrameN);
 maxAngle_IPD2_L_Hip_Angle_x = max(IPD2_L_Hip_Angle_x_subset);
 minAngle_IPD2_L_Hip_Angle_x = min(IPD2_L_Hip_Angle_x_subset);
 if maxAngle_IPD2_L_Hip_Angle_x >(abs(minAngle_IPD2_L_Hip_Angle_x))
    pAngle_IPD2_L_Hip_Angle_x = maxAngle_IPD2_L_Hip_Angle_x;
 else
    pAngle_IPD2_L_Hip_Angle_x = minAngle_IPD2_L_Hip_Angle_x;
 end
 pAnlge_idx_IPD2_L_Hip_Angle_x = find(IPD2_L_Hip_Angle_x ==
pAngle_IPD2_L_Hip_Angle_x);
 t2p_IPD2_L_Hip_Angle_x = (((pAnlge_idx_IPD2_L_Hip_Angle_x-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Hip_Angle_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Hip_Angle_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Hip_Angle_y_subset =
IPD2_L_Hip_Angle_y(IPD2_L_Hip_Angle_y_Frame1:IPD2_L_Hip_Angle_y_FrameN);
 maxAngle_IPD2_L_Hip_Angle_y = max(IPD2_L_Hip_Angle_y_subset);
 minAngle_IPD2_L_Hip_Angle_y = min(IPD2_L_Hip_Angle_y_subset);
 if maxAngle_IPD2_L_Hip_Angle_y >(abs(minAngle_IPD2_L_Hip_Angle_y))
    pAngle_IPD2_L_Hip_Angle_y = maxAngle_IPD2_L_Hip_Angle_y;
 else
    pAngle_IPD2_L_Hip_Angle_y = minAngle_IPD2_L_Hip_Angle_y;
 end
 pAnlge_idx_IPD2_L_Hip_Angle_y = find(IPD2_L_Hip_Angle_y ==
pAngle_IPD2_L_Hip_Angle_y);
 t2p_IPD2_L_Hip_Angle_y = (((pAnlge_idx_IPD2_L_Hip_Angle_y-1)*0.005)-
IPD2_True_Drop);


 IPD2_L_Hip_Angle_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Hip_Angle_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Hip_Angle_z_subset =
IPD2_L_Hip_Angle_z(IPD2_L_Hip_Angle_z_Frame1:IPD2_L_Hip_Angle_z_FrameN);
 maxAngle_IPD2_L_Hip_Angle_z = max(IPD2_L_Hip_Angle_z_subset);
 minAngle_IPD2_L_Hip_Angle_z = min(IPD2_L_Hip_Angle_z_subset);
 if maxAngle_IPD2_L_Hip_Angle_z >(abs(minAngle_IPD2_L_Hip_Angle_z))
    pAngle_IPD2_L_Hip_Angle_z = maxAngle_IPD2_L_Hip_Angle_z;
```

```
 else
    pAngle_IPD2_L_Hip_Angle_z = minAngle_IPD2_L_Hip_Angle_z;
 end
 pAnlge_idx_IPD2_L_Hip_Angle_z = find(IPD2_L_Hip_Angle_z ==
pAngle_IPD2_L_Hip_Angle_z);
 t2p_IPD2_L_Hip_Angle_z = (((pAnlge_idx_IPD2_L_Hip_Angle_z-1)*0.005)-
IPD2_True_Drop);

 %IPD3
 IPD3_L_Ankle_Angle_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Ankle_Angle_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Ankle_Angle_x_subset =
IPD3_L_Ankle_Angle_x(IPD3_L_Ankle_Angle_x_Frame1:IPD3_L_Ankle_Angle_x_FrameN);
 maxAngle_IPD3_L_Ankle_Angle_x = max(IPD3_L_Ankle_Angle_x_subset);
 minAngle_IPD3_L_Ankle_Angle_x = min(IPD3_L_Ankle_Angle_x_subset);
 if maxAngle_IPD3_L_Ankle_Angle_x >(abs(minAngle_IPD3_L_Ankle_Angle_x))
    pAngle_IPD3_L_Ankle_Angle_x = maxAngle_IPD3_L_Ankle_Angle_x;
 else
    pAngle_IPD3_L_Ankle_Angle_x = minAngle_IPD3_L_Ankle_Angle_x;
 end
 pAnlge_idx_IPD3_L_Ankle_Angle_x = find(IPD3_L_Ankle_Angle_x ==
pAngle_IPD3_L_Ankle_Angle_x);
 t2p_IPD3_L_Ankle_Angle_x = (((pAnlge_idx_IPD3_L_Ankle_Angle_x-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Ankle_Angle_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Ankle_Angle_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Ankle_Angle_y_subset =
IPD3_L_Ankle_Angle_y(IPD3_L_Ankle_Angle_y_Frame1:IPD3_L_Ankle_Angle_y_FrameN);
 maxAngle_IPD3_L_Ankle_Angle_y = max(IPD3_L_Ankle_Angle_y_subset);
 minAngle_IPD3_L_Ankle_Angle_y = min(IPD3_L_Ankle_Angle_y_subset);
 if maxAngle_IPD3_L_Ankle_Angle_y >(abs(minAngle_IPD3_L_Ankle_Angle_y))
    pAngle_IPD3_L_Ankle_Angle_y = maxAngle_IPD3_L_Ankle_Angle_y;
 else
    pAngle_IPD3_L_Ankle_Angle_y = minAngle_IPD3_L_Ankle_Angle_y;
 end
 pAnlge_idx_IPD3_L_Ankle_Angle_y = find(IPD3_L_Ankle_Angle_y ==
pAngle_IPD3_L_Ankle_Angle_y);
 t2p_IPD3_L_Ankle_Angle_y = (((pAnlge_idx_IPD3_L_Ankle_Angle_y-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Ankle_Angle_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Ankle_Angle_z_FrameN = round((IPD3_Post_350/.005)+1);
```

```
 IPD3_L_Ankle_Angle_z_subset =
IPD3_L_Ankle_Angle_z(IPD3_L_Ankle_Angle_z_Frame1:IPD3_L_Ankle_Angle_z_FrameN);
 maxAngle_IPD3_L_Ankle_Angle_z = max(IPD3_L_Ankle_Angle_z_subset);
 minAngle_IPD3_L_Ankle_Angle_z = min(IPD3_L_Ankle_Angle_z_subset);
 if maxAngle_IPD3_L_Ankle_Angle_z >(abs(minAngle_IPD3_L_Ankle_Angle_z))
    pAngle_IPD3_L_Ankle_Angle_z = maxAngle_IPD3_L_Ankle_Angle_z;
 else
    pAngle_IPD3_L_Ankle_Angle_z = minAngle_IPD3_L_Ankle_Angle_z;
 end
 pAnlge_idx_IPD3_L_Ankle_Angle_z = find(IPD3_L_Ankle_Angle_z ==
pAngle_IPD3_L_Ankle_Angle_z);
 t2p_IPD3_L_Ankle_Angle_z = (((pAnlge_idx_IPD3_L_Ankle_Angle_z-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Knee_Angle_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Knee_Angle_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Knee_Angle_x_subset =
IPD3_L_Knee_Angle_x(IPD3_L_Knee_Angle_x_Frame1:IPD3_L_Knee_Angle_x_FrameN);
 maxAngle_IPD3_L_Knee_Angle_x = max(IPD3_L_Knee_Angle_x_subset);
 minAngle_IPD3_L_Knee_Angle_x = min(IPD3_L_Knee_Angle_x_subset);
 if maxAngle_IPD3_L_Knee_Angle_x >(abs(minAngle_IPD3_L_Knee_Angle_x))
    pAngle_IPD3_L_Knee_Angle_x = maxAngle_IPD3_L_Knee_Angle_x;
 else
    pAngle_IPD3_L_Knee_Angle_x = minAngle_IPD3_L_Knee_Angle_x;
 end
 pAnlge_idx_IPD3_L_Knee_Angle_x = find(IPD3_L_Knee_Angle_x ==
pAngle_IPD3_L_Knee_Angle_x);
 t2p_IPD3_L_Knee_Angle_x = (((pAnlge_idx_IPD3_L_Knee_Angle_x-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Knee_Angle_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Knee_Angle_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Knee_Angle_y_subset =
IPD3_L_Knee_Angle_y(IPD3_L_Knee_Angle_y_Frame1:IPD3_L_Knee_Angle_y_FrameN);
 maxAngle_IPD3_L_Knee_Angle_y = max(IPD3_L_Knee_Angle_y_subset);
 minAngle_IPD3_L_Knee_Angle_y = min(IPD3_L_Knee_Angle_y_subset);
 if maxAngle_IPD3_L_Knee_Angle_y >(abs(minAngle_IPD3_L_Knee_Angle_y))
    pAngle_IPD3_L_Knee_Angle_y = maxAngle_IPD3_L_Knee_Angle_y;
 else
    pAngle_IPD3_L_Knee_Angle_y = minAngle_IPD3_L_Knee_Angle_y;
 end
 pAnlge_idx_IPD3_L_Knee_Angle_y = find(IPD3_L_Knee_Angle_y ==
pAngle_IPD3_L_Knee_Angle_y);
```

```
 t2p_IPD3_L_Knee_Angle_y = (((pAnlge_idx_IPD3_L_Knee_Angle_y-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Knee_Angle_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Knee_Angle_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Knee_Angle_z_subset =
IPD3_L_Knee_Angle_z(IPD3_L_Knee_Angle_z_Frame1:IPD3_L_Knee_Angle_z_FrameN);
 maxAngle_IPD3_L_Knee_Angle_z = max(IPD3_L_Knee_Angle_z_subset);
 minAngle_IPD3_L_Knee_Angle_z = min(IPD3_L_Knee_Angle_z_subset);
 if maxAngle_IPD3_L_Knee_Angle_z >(abs(minAngle_IPD3_L_Knee_Angle_z))
    pAngle_IPD3_L_Knee_Angle_z = maxAngle_IPD3_L_Knee_Angle_z;
 else
    pAngle_IPD3_L_Knee_Angle_z = minAngle_IPD3_L_Knee_Angle_z;
 end
 pAnlge_idx_IPD3_L_Knee_Angle_z = find(IPD3_L_Knee_Angle_z ==
pAngle_IPD3_L_Knee_Angle_z);
 t2p_IPD3_L_Knee_Angle_z = (((pAnlge_idx_IPD3_L_Knee_Angle_z-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Hip_Angle_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Hip_Angle_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Hip_Angle_x_subset =
IPD3_L_Hip_Angle_x(IPD3_L_Hip_Angle_x_Frame1:IPD3_L_Hip_Angle_x_FrameN);
 maxAngle_IPD3_L_Hip_Angle_x = max(IPD3_L_Hip_Angle_x_subset);
 minAngle_IPD3_L_Hip_Angle_x = min(IPD3_L_Hip_Angle_x_subset);
 if maxAngle_IPD3_L_Hip_Angle_x >(abs(minAngle_IPD3_L_Hip_Angle_x))
    pAngle_IPD3_L_Hip_Angle_x = maxAngle_IPD3_L_Hip_Angle_x;
 else
    pAngle_IPD3_L_Hip_Angle_x = minAngle_IPD3_L_Hip_Angle_x;
 end
 pAnlge_idx_IPD3_L_Hip_Angle_x = find(IPD3_L_Hip_Angle_x ==
pAngle_IPD3_L_Hip_Angle_x);
 t2p_IPD3_L_Hip_Angle_x = (((pAnlge_idx_IPD3_L_Hip_Angle_x-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Hip_Angle_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Hip_Angle_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Hip_Angle_y_subset =
IPD3_L_Hip_Angle_y(IPD3_L_Hip_Angle_y_Frame1:IPD3_L_Hip_Angle_y_FrameN);
 maxAngle_IPD3_L_Hip_Angle_y = max(IPD3_L_Hip_Angle_y_subset);
 minAngle_IPD3_L_Hip_Angle_y = min(IPD3_L_Hip_Angle_y_subset);
 if maxAngle_IPD3_L_Hip_Angle_y >(abs(minAngle_IPD3_L_Hip_Angle_y))
    pAngle_IPD3_L_Hip_Angle_y = maxAngle_IPD3_L_Hip_Angle_y;
```

```
 else
    pAngle_IPD3_L_Hip_Angle_y = minAngle_IPD3_L_Hip_Angle_y;
 end
 pAnlge_idx_IPD3_L_Hip_Angle_y = find(IPD3_L_Hip_Angle_y ==
pAngle_IPD3_L_Hip_Angle_y);
 t2p_IPD3_L_Hip_Angle_y = (((pAnlge_idx_IPD3_L_Hip_Angle_y-1)*0.005)-
IPD3_True_Drop);


 IPD3_L_Hip_Angle_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Hip_Angle_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Hip_Angle_z_subset =
IPD3_L_Hip_Angle_z(IPD3_L_Hip_Angle_z_Frame1:IPD3_L_Hip_Angle_z_FrameN);
 maxAngle_IPD3_L_Hip_Angle_z = max(IPD3_L_Hip_Angle_z_subset);
 minAngle_IPD3_L_Hip_Angle_z = min(IPD3_L_Hip_Angle_z_subset);
 if maxAngle_IPD3_L_Hip_Angle_z >(abs(minAngle_IPD3_L_Hip_Angle_z))
    pAngle_IPD3_L_Hip_Angle_z = maxAngle_IPD3_L_Hip_Angle_z;
 else
    pAngle_IPD3_L_Hip_Angle_z = minAngle_IPD3_L_Hip_Angle_z;
 end
 pAnlge_idx_IPD3_L_Hip_Angle_z = find(IPD3_L_Hip_Angle_z ==
pAngle_IPD3_L_Hip_Angle_z);
 t2p_IPD3_L_Hip_Angle_z = (((pAnlge_idx_IPD3_L_Hip_Angle_z-1)*0.005)-
IPD3_True_Drop);


 %NW1
 NW1_L_Ankle_Angle_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Ankle_Angle_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Ankle_Angle_x_subset =
NW1_L_Ankle_Angle_x(NW1_L_Ankle_Angle_x_Frame1:NW1_L_Ankle_Angle_x_FrameN)
;
 maxAngle_NW1_L_Ankle_Angle_x = max(NW1_L_Ankle_Angle_x_subset);
 minAngle_NW1_L_Ankle_Angle_x = min(NW1_L_Ankle_Angle_x_subset);
 if maxAngle_NW1_L_Ankle_Angle_x >(abs(minAngle_NW1_L_Ankle_Angle_x))
    pAngle_NW1_L_Ankle_Angle_x = maxAngle_NW1_L_Ankle_Angle_x;
 else
    pAngle_NW1_L_Ankle_Angle_x = minAngle_NW1_L_Ankle_Angle_x;
 end
 pAnlge_idx_NW1_L_Ankle_Angle_x = find(NW1_L_Ankle_Angle_x ==
pAngle_NW1_L_Ankle_Angle_x);
 t2p_NW1_L_Ankle_Angle_x = (((pAnlge_idx_NW1_L_Ankle_Angle_x-1)*0.005)-NW1_HS);


 NW1_L_Ankle_Angle_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Ankle_Angle_y_FrameN = round((NW1_Post_350/.005)+1);
```

```
 NW1_L_Ankle_Angle_y_subset =
NW1_L_Ankle_Angle_y(NW1_L_Ankle_Angle_y_Frame1:NW1_L_Ankle_Angle_y_FrameN)
;
 maxAngle_NW1_L_Ankle_Angle_y = max(NW1_L_Ankle_Angle_y_subset);
 minAngle_NW1_L_Ankle_Angle_y = min(NW1_L_Ankle_Angle_y_subset);
 if maxAngle_NW1_L_Ankle_Angle_y >(abs(minAngle_NW1_L_Ankle_Angle_y))
    pAngle_NW1_L_Ankle_Angle_y = maxAngle_NW1_L_Ankle_Angle_y;
 else
    pAngle_NW1_L_Ankle_Angle_y = minAngle_NW1_L_Ankle_Angle_y;
 end
 pAnlge_idx_NW1_L_Ankle_Angle_y = find(NW1_L_Ankle_Angle_y ==
pAngle_NW1_L_Ankle_Angle_y);
 t2p_NW1_L_Ankle_Angle_y = (((pAnlge_idx_NW1_L_Ankle_Angle_y-1)*0.005)-NW1_HS);


 NW1_L_Ankle_Angle_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Ankle_Angle_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Ankle_Angle_z_subset =
NW1_L_Ankle_Angle_z(NW1_L_Ankle_Angle_z_Frame1:NW1_L_Ankle_Angle_z_FrameN);
 maxAngle_NW1_L_Ankle_Angle_z = max(NW1_L_Ankle_Angle_z_subset);
 minAngle_NW1_L_Ankle_Angle_z = min(NW1_L_Ankle_Angle_z_subset);
 if maxAngle_NW1_L_Ankle_Angle_z >(abs(minAngle_NW1_L_Ankle_Angle_z))
    pAngle_NW1_L_Ankle_Angle_z = maxAngle_NW1_L_Ankle_Angle_z;
 else
    pAngle_NW1_L_Ankle_Angle_z = minAngle_NW1_L_Ankle_Angle_z;
 end
 pAnlge_idx_NW1_L_Ankle_Angle_z = find(NW1_L_Ankle_Angle_z ==
pAngle_NW1_L_Ankle_Angle_z);
 t2p_NW1_L_Ankle_Angle_z = (((pAnlge_idx_NW1_L_Ankle_Angle_z-1)*0.005)-NW1_HS);


 NW1_L_Knee_Angle_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Knee_Angle_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Knee_Angle_x_subset =
NW1_L_Knee_Angle_x(NW1_L_Knee_Angle_x_Frame1:NW1_L_Knee_Angle_x_FrameN);
 maxAngle_NW1_L_Knee_Angle_x = max(NW1_L_Knee_Angle_x_subset);
 minAngle_NW1_L_Knee_Angle_x = min(NW1_L_Knee_Angle_x_subset);
 if maxAngle_NW1_L_Knee_Angle_x >(abs(minAngle_NW1_L_Knee_Angle_x))
    pAngle_NW1_L_Knee_Angle_x = maxAngle_NW1_L_Knee_Angle_x;
 else
    pAngle_NW1_L_Knee_Angle_x = minAngle_NW1_L_Knee_Angle_x;
 end
 pAnlge_idx_NW1_L_Knee_Angle_x = find(NW1_L_Knee_Angle_x ==
pAngle_NW1_L_Knee_Angle_x);
 t2p_NW1_L_Knee_Angle_x = (((pAnlge_idx_NW1_L_Knee_Angle_x-1)*0.005)-NW1_HS);
```

```
 NW1_L_Knee_Angle_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Knee_Angle_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Knee_Angle_y_subset =
NW1_L_Knee_Angle_y(NW1_L_Knee_Angle_y_Frame1:NW1_L_Knee_Angle_y_FrameN);
 maxAngle_NW1_L_Knee_Angle_y = max(NW1_L_Knee_Angle_y_subset);
 minAngle_NW1_L_Knee_Angle_y = min(NW1_L_Knee_Angle_y_subset);
 if maxAngle_NW1_L_Knee_Angle_y >(abs(minAngle_NW1_L_Knee_Angle_y))
    pAngle_NW1_L_Knee_Angle_y = maxAngle_NW1_L_Knee_Angle_y;
 else
    pAngle_NW1_L_Knee_Angle_y = minAngle_NW1_L_Knee_Angle_y;
 end
 pAnlge_idx_NW1_L_Knee_Angle_y = find(NW1_L_Knee_Angle_y ==
pAngle_NW1_L_Knee_Angle_y);
 t2p_NW1_L_Knee_Angle_y = (((pAnlge_idx_NW1_L_Knee_Angle_y-1)*0.005)-NW1_HS);


 NW1_L_Knee_Angle_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Knee_Angle_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Knee_Angle_z_subset =
NW1_L_Knee_Angle_z(NW1_L_Knee_Angle_z_Frame1:NW1_L_Knee_Angle_z_FrameN);
 maxAngle_NW1_L_Knee_Angle_z = max(NW1_L_Knee_Angle_z_subset);
 minAngle_NW1_L_Knee_Angle_z = min(NW1_L_Knee_Angle_z_subset);
 if maxAngle_NW1_L_Knee_Angle_z >(abs(minAngle_NW1_L_Knee_Angle_z))
    pAngle_NW1_L_Knee_Angle_z = maxAngle_NW1_L_Knee_Angle_z;
 else
    pAngle_NW1_L_Knee_Angle_z = minAngle_NW1_L_Knee_Angle_z;
 end
 pAnlge_idx_NW1_L_Knee_Angle_z = find(NW1_L_Knee_Angle_z ==
pAngle_NW1_L_Knee_Angle_z);
 t2p_NW1_L_Knee_Angle_z = (((pAnlge_idx_NW1_L_Knee_Angle_z-1)*0.005)-NW1_HS);


 NW1_L_Hip_Angle_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Hip_Angle_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Hip_Angle_x_subset =
NW1_L_Hip_Angle_x(NW1_L_Hip_Angle_x_Frame1:NW1_L_Hip_Angle_x_FrameN);
 maxAngle_NW1_L_Hip_Angle_x = max(NW1_L_Hip_Angle_x_subset);
 minAngle_NW1_L_Hip_Angle_x = min(NW1_L_Hip_Angle_x_subset);
 if maxAngle_NW1_L_Hip_Angle_x >(abs(minAngle_NW1_L_Hip_Angle_x))
    pAngle_NW1_L_Hip_Angle_x = maxAngle_NW1_L_Hip_Angle_x;
 else
    pAngle_NW1_L_Hip_Angle_x = minAngle_NW1_L_Hip_Angle_x;
 end
 pAnlge_idx_NW1_L_Hip_Angle_x = find(NW1_L_Hip_Angle_x ==
pAngle_NW1_L_Hip_Angle_x);
```

```
t2p_NW1_L_Hip_Angle_x = (((pAnlge_idx_NW1_L_Hip_Angle_x-1)*0.005)-NW1_HS);


NW1_L_Hip_Angle_y_Frame1 = round((NW1_HS/.005)+1);
NW1_L_Hip_Angle_y_FrameN = round((NW1_Post_350/.005)+1);
NW1_L_Hip_Angle_y_subset =
NW1_L_Hip_Angle_y(NW1_L_Hip_Angle_y_Frame1:NW1_L_Hip_Angle_y_FrameN);
maxAngle_NW1_L_Hip_Angle_y = max(NW1_L_Hip_Angle_y_subset);
minAngle_NW1_L_Hip_Angle_y = min(NW1_L_Hip_Angle_y_subset);
if maxAngle_NW1_L_Hip_Angle_y >(abs(minAngle_NW1_L_Hip_Angle_y))
   pAngle_NW1_L_Hip_Angle_y = maxAngle_NW1_L_Hip_Angle_y;
else
   pAngle_NW1_L_Hip_Angle_y = minAngle_NW1_L_Hip_Angle_y;
end
pAnlge_idx_NW1_L_Hip_Angle_y = find(NW1_L_Hip_Angle_y ==
pAngle_NW1_L_Hip_Angle_y);
t2p_NW1_L_Hip_Angle_y = (((pAnlge_idx_NW1_L_Hip_Angle_y-1)*0.005)-NW1_HS);


NW1_L_Hip_Angle_z_Frame1 = round((NW1_HS/.005)+1);
NW1_L_Hip_Angle_z_FrameN = round((NW1_Post_350/.005)+1);
NW1_L_Hip_Angle_z_subset =
NW1_L_Hip_Angle_z(NW1_L_Hip_Angle_z_Frame1:NW1_L_Hip_Angle_z_FrameN);
maxAngle_NW1_L_Hip_Angle_z = max(NW1_L_Hip_Angle_z_subset);
minAngle_NW1_L_Hip_Angle_z = min(NW1_L_Hip_Angle_z_subset);
if maxAngle_NW1_L_Hip_Angle_z >(abs(minAngle_NW1_L_Hip_Angle_z))
   pAngle_NW1_L_Hip_Angle_z = maxAngle_NW1_L_Hip_Angle_z;
else
   pAngle_NW1_L_Hip_Angle_z = minAngle_NW1_L_Hip_Angle_z;
end
pAnlge_idx_NW1_L_Hip_Angle_z = find(NW1_L_Hip_Angle_z ==
pAngle_NW1_L_Hip_Angle_z);
t2p_NW1_L_Hip_Angle_z = (((pAnlge_idx_NW1_L_Hip_Angle_z-1)*0.005)-NW1_HS);



%NW2
NW2_L_Ankle_Angle_x_Frame1 = round((NW2_HS/.005)+1);
NW2_L_Ankle_Angle_x_FrameN = round((NW2_Post_350/.005)+1);
NW2_L_Ankle_Angle_x_subset =
NW2_L_Ankle_Angle_x(NW2_L_Ankle_Angle_x_Frame1:NW2_L_Ankle_Angle_x_FrameN)
;
maxAngle_NW2_L_Ankle_Angle_x = max(NW2_L_Ankle_Angle_x_subset);
minAngle_NW2_L_Ankle_Angle_x = min(NW2_L_Ankle_Angle_x_subset);
if maxAngle_NW2_L_Ankle_Angle_x >(abs(minAngle_NW2_L_Ankle_Angle_x))
   pAngle_NW2_L_Ankle_Angle_x = maxAngle_NW2_L_Ankle_Angle_x;
```

```
 else
    pAngle_NW2_L_Ankle_Angle_x = minAngle_NW2_L_Ankle_Angle_x;
 end
 pAnlge_idx_NW2_L_Ankle_Angle_x = find(NW2_L_Ankle_Angle_x ==
pAngle_NW2_L_Ankle_Angle_x);
 t2p_NW2_L_Ankle_Angle_x = (((pAnlge_idx_NW2_L_Ankle_Angle_x-1)*0.005)-NW2_HS);


 NW2_L_Ankle_Angle_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Ankle_Angle_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Ankle_Angle_y_subset =
NW2_L_Ankle_Angle_y(NW2_L_Ankle_Angle_y_Frame1:NW2_L_Ankle_Angle_y_FrameN)
;
 maxAngle_NW2_L_Ankle_Angle_y = max(NW2_L_Ankle_Angle_y_subset);
 minAngle_NW2_L_Ankle_Angle_y = min(NW2_L_Ankle_Angle_y_subset);
 if maxAngle_NW2_L_Ankle_Angle_y >(abs(minAngle_NW2_L_Ankle_Angle_y))
    pAngle_NW2_L_Ankle_Angle_y = maxAngle_NW2_L_Ankle_Angle_y;
 else
    pAngle_NW2_L_Ankle_Angle_y = minAngle_NW2_L_Ankle_Angle_y;
 end
 pAnlge_idx_NW2_L_Ankle_Angle_y = find(NW2_L_Ankle_Angle_y ==
pAngle_NW2_L_Ankle_Angle_y);
 t2p_NW2_L_Ankle_Angle_y = (((pAnlge_idx_NW2_L_Ankle_Angle_y-1)*0.005)-NW2_HS);


 NW2_L_Ankle_Angle_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Ankle_Angle_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Ankle_Angle_z_subset =
NW2_L_Ankle_Angle_z(NW2_L_Ankle_Angle_z_Frame1:NW2_L_Ankle_Angle_z_FrameN);
 maxAngle_NW2_L_Ankle_Angle_z = max(NW2_L_Ankle_Angle_z_subset);
 minAngle_NW2_L_Ankle_Angle_z = min(NW2_L_Ankle_Angle_z_subset);
 if maxAngle_NW2_L_Ankle_Angle_z >(abs(minAngle_NW2_L_Ankle_Angle_z))
    pAngle_NW2_L_Ankle_Angle_z = maxAngle_NW2_L_Ankle_Angle_z;
 else
    pAngle_NW2_L_Ankle_Angle_z = minAngle_NW2_L_Ankle_Angle_z;
 end
 pAnlge_idx_NW2_L_Ankle_Angle_z = find(NW2_L_Ankle_Angle_z ==
pAngle_NW2_L_Ankle_Angle_z);
 t2p_NW2_L_Ankle_Angle_z = (((pAnlge_idx_NW2_L_Ankle_Angle_z-1)*0.005)-NW2_HS);


 NW2_L_Knee_Angle_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Knee_Angle_x_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Knee_Angle_x_subset =
NW2_L_Knee_Angle_x(NW2_L_Knee_Angle_x_Frame1:NW2_L_Knee_Angle_x_FrameN);
 maxAngle_NW2_L_Knee_Angle_x = max(NW2_L_Knee_Angle_x_subset);
```

```
minAngle_NW2_L_Knee_Angle_x = min(NW2_L_Knee_Angle_x_subset);
if maxAngle_NW2_L_Knee_Angle_x >(abs(minAngle_NW2_L_Knee_Angle_x))
   pAngle_NW2_L_Knee_Angle_x = maxAngle_NW2_L_Knee_Angle_x;
else
   pAngle_NW2_L_Knee_Angle_x = minAngle_NW2_L_Knee_Angle_x;
end
pAnlge_idx_NW2_L_Knee_Angle_x = find(NW2_L_Knee_Angle_x ==
pAngle_NW2_L_Knee_Angle_x);
t2p_NW2_L_Knee_Angle_x = (((pAnlge_idx_NW2_L_Knee_Angle_x-1)*0.005)-NW2_HS);


NW2_L_Knee_Angle_y_Frame1 = round((NW2_HS/.005)+1);
NW2_L_Knee_Angle_y_FrameN = round((NW2_Post_350/.005)+1);
NW2_L_Knee_Angle_y_subset =
NW2_L_Knee_Angle_y(NW2_L_Knee_Angle_y_Frame1:NW2_L_Knee_Angle_y_FrameN);
maxAngle_NW2_L_Knee_Angle_y = max(NW2_L_Knee_Angle_y_subset);
minAngle_NW2_L_Knee_Angle_y = min(NW2_L_Knee_Angle_y_subset);
if maxAngle_NW2_L_Knee_Angle_y >(abs(minAngle_NW2_L_Knee_Angle_y))
   pAngle_NW2_L_Knee_Angle_y = maxAngle_NW2_L_Knee_Angle_y;
else
   pAngle_NW2_L_Knee_Angle_y = minAngle_NW2_L_Knee_Angle_y;
end
pAnlge_idx_NW2_L_Knee_Angle_y = find(NW2_L_Knee_Angle_y ==
pAngle_NW2_L_Knee_Angle_y);
t2p_NW2_L_Knee_Angle_y = (((pAnlge_idx_NW2_L_Knee_Angle_y-1)*0.005)-NW2_HS);


NW2_L_Knee_Angle_z_Frame1 = round((NW2_HS/.005)+1);
NW2_L_Knee_Angle_z_FrameN = round((NW2_Post_350/.005)+1);
NW2_L_Knee_Angle_z_subset =
NW2_L_Knee_Angle_z(NW2_L_Knee_Angle_z_Frame1:NW2_L_Knee_Angle_z_FrameN);
maxAngle_NW2_L_Knee_Angle_z = max(NW2_L_Knee_Angle_z_subset);
minAngle_NW2_L_Knee_Angle_z = min(NW2_L_Knee_Angle_z_subset);
if maxAngle_NW2_L_Knee_Angle_z >(abs(minAngle_NW2_L_Knee_Angle_z))
   pAngle_NW2_L_Knee_Angle_z = maxAngle_NW2_L_Knee_Angle_z;
else
   pAngle_NW2_L_Knee_Angle_z = minAngle_NW2_L_Knee_Angle_z;
end
pAnlge_idx_NW2_L_Knee_Angle_z = find(NW2_L_Knee_Angle_z ==
pAngle_NW2_L_Knee_Angle_z);
t2p_NW2_L_Knee_Angle_z = (((pAnlge_idx_NW2_L_Knee_Angle_z-1)*0.005)-NW2_HS);


NW2_L_Hip_Angle_x_Frame1 = round((NW2_HS/.005)+1);
NW2_L_Hip_Angle_x_FrameN = round((NW2_Post_350/.005)+1);
```

```
 NW2_L_Hip_Angle_x_subset =
NW2_L_Hip_Angle_x(NW2_L_Hip_Angle_x_Frame1:NW2_L_Hip_Angle_x_FrameN);
 maxAngle_NW2_L_Hip_Angle_x = max(NW2_L_Hip_Angle_x_subset);
 minAngle_NW2_L_Hip_Angle_x = min(NW2_L_Hip_Angle_x_subset);
 if maxAngle_NW2_L_Hip_Angle_x >(abs(minAngle_NW2_L_Hip_Angle_x))
    pAngle_NW2_L_Hip_Angle_x = maxAngle_NW2_L_Hip_Angle_x;
 else
    pAngle_NW2_L_Hip_Angle_x = minAngle_NW2_L_Hip_Angle_x;
 end
 pAnlge_idx_NW2_L_Hip_Angle_x = find(NW2_L_Hip_Angle_x ==
pAngle_NW2_L_Hip_Angle_x);
 t2p_NW2_L_Hip_Angle_x = (((pAnlge_idx_NW2_L_Hip_Angle_x-1)*0.005)-NW2_HS);


 NW2_L_Hip_Angle_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Hip_Angle_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Hip_Angle_y_subset =
NW2_L_Hip_Angle_y(NW2_L_Hip_Angle_y_Frame1:NW2_L_Hip_Angle_y_FrameN);
 maxAngle_NW2_L_Hip_Angle_y = max(NW2_L_Hip_Angle_y_subset);
 minAngle_NW2_L_Hip_Angle_y = min(NW2_L_Hip_Angle_y_subset);
 if maxAngle_NW2_L_Hip_Angle_y >(abs(minAngle_NW2_L_Hip_Angle_y))
    pAngle_NW2_L_Hip_Angle_y = maxAngle_NW2_L_Hip_Angle_y;
 else
    pAngle_NW2_L_Hip_Angle_y = minAngle_NW2_L_Hip_Angle_y;
 end
 pAnlge_idx_NW2_L_Hip_Angle_y = find(NW2_L_Hip_Angle_y ==
pAngle_NW2_L_Hip_Angle_y);
 t2p_NW2_L_Hip_Angle_y = (((pAnlge_idx_NW2_L_Hip_Angle_y-1)*0.005)-NW2_HS);


 NW2_L_Hip_Angle_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Hip_Angle_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Hip_Angle_z_subset =
NW2_L_Hip_Angle_z(NW2_L_Hip_Angle_z_Frame1:NW2_L_Hip_Angle_z_FrameN);
 maxAngle_NW2_L_Hip_Angle_z = max(NW2_L_Hip_Angle_z_subset);
 minAngle_NW2_L_Hip_Angle_z = min(NW2_L_Hip_Angle_z_subset);
 if maxAngle_NW2_L_Hip_Angle_z >(abs(minAngle_NW2_L_Hip_Angle_z))
    pAngle_NW2_L_Hip_Angle_z = maxAngle_NW2_L_Hip_Angle_z;
 else
    pAngle_NW2_L_Hip_Angle_z = minAngle_NW2_L_Hip_Angle_z;
 end
 pAnlge_idx_NW2_L_Hip_Angle_z = find(NW2_L_Hip_Angle_z ==
pAngle_NW2_L_Hip_Angle_z);
 t2p_NW2_L_Hip_Angle_z = (((pAnlge_idx_NW2_L_Hip_Angle_z-1)*0.005)-NW2_HS);
```

```
%NW3
 NW3_L_Ankle_Angle_x_Frame1 = round((NW3_HS/.005)+1);
NW3_L_Ankle_Angle_x_FrameN = round((NW3_Post_350/.005)+1);
NW3_L_Ankle_Angle_x_subset =
NW3_L_Ankle_Angle_x(NW3_L_Ankle_Angle_x_Frame1:NW3_L_Ankle_Angle_x_FrameN)
;
maxAngle_NW3_L_Ankle_Angle_x = max(NW3_L_Ankle_Angle_x_subset);
minAngle_NW3_L_Ankle_Angle_x = min(NW3_L_Ankle_Angle_x_subset);
if maxAngle_NW3_L_Ankle_Angle_x >(abs(minAngle_NW3_L_Ankle_Angle_x))
   pAngle_NW3_L_Ankle_Angle_x = maxAngle_NW3_L_Ankle_Angle_x;
else
   pAngle_NW3_L_Ankle_Angle_x = minAngle_NW3_L_Ankle_Angle_x;
end
pAnlge_idx_NW3_L_Ankle_Angle_x = find(NW3_L_Ankle_Angle_x ==
pAngle_NW3_L_Ankle_Angle_x);
t2p_NW3_L_Ankle_Angle_x = (((pAnlge_idx_NW3_L_Ankle_Angle_x-1)*0.005)-NW3_HS);


NW3_L_Ankle_Angle_y_Frame1 = round((NW3_HS/.005)+1);
NW3_L_Ankle_Angle_y_FrameN = round((NW3_Post_350/.005)+1);
NW3_L_Ankle_Angle_y_subset =
NW3_L_Ankle_Angle_y(NW3_L_Ankle_Angle_y_Frame1:NW3_L_Ankle_Angle_y_FrameN)
;
maxAngle_NW3_L_Ankle_Angle_y = max(NW3_L_Ankle_Angle_y_subset);
minAngle_NW3_L_Ankle_Angle_y = min(NW3_L_Ankle_Angle_y_subset);
if maxAngle_NW3_L_Ankle_Angle_y >(abs(minAngle_NW3_L_Ankle_Angle_y))
   pAngle_NW3_L_Ankle_Angle_y = maxAngle_NW3_L_Ankle_Angle_y;
else
   pAngle_NW3_L_Ankle_Angle_y = minAngle_NW3_L_Ankle_Angle_y;
end
pAnlge_idx_NW3_L_Ankle_Angle_y = find(NW3_L_Ankle_Angle_y ==
pAngle_NW3_L_Ankle_Angle_y);
t2p_NW3_L_Ankle_Angle_y = (((pAnlge_idx_NW3_L_Ankle_Angle_y-1)*0.005)-NW3_HS);


NW3_L_Ankle_Angle_z_Frame1 = round((NW3_HS/.005)+1);
NW3_L_Ankle_Angle_z_FrameN = round((NW3_Post_350/.005)+1);
NW3_L_Ankle_Angle_z_subset =
NW3_L_Ankle_Angle_z(NW3_L_Ankle_Angle_z_Frame1:NW3_L_Ankle_Angle_z_FrameN);
maxAngle_NW3_L_Ankle_Angle_z = max(NW3_L_Ankle_Angle_z_subset);
minAngle_NW3_L_Ankle_Angle_z = min(NW3_L_Ankle_Angle_z_subset);
if maxAngle_NW3_L_Ankle_Angle_z >(abs(minAngle_NW3_L_Ankle_Angle_z))
   pAngle_NW3_L_Ankle_Angle_z = maxAngle_NW3_L_Ankle_Angle_z;
else
   pAngle_NW3_L_Ankle_Angle_z = minAngle_NW3_L_Ankle_Angle_z;
```

```
 end
 pAnlge_idx_NW3_L_Ankle_Angle_z = find(NW3_L_Ankle_Angle_z ==
pAngle_NW3_L_Ankle_Angle_z);
 t2p_NW3_L_Ankle_Angle_z = (((pAnlge_idx_NW3_L_Ankle_Angle_z-1)*0.005)-NW3_HS);


 NW3_L_Knee_Angle_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Knee_Angle_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Knee_Angle_x_subset =
NW3_L_Knee_Angle_x(NW3_L_Knee_Angle_x_Frame1:NW3_L_Knee_Angle_x_FrameN);
 maxAngle_NW3_L_Knee_Angle_x = max(NW3_L_Knee_Angle_x_subset);
 minAngle_NW3_L_Knee_Angle_x = min(NW3_L_Knee_Angle_x_subset);
 if maxAngle_NW3_L_Knee_Angle_x >(abs(minAngle_NW3_L_Knee_Angle_x))
    pAngle_NW3_L_Knee_Angle_x = maxAngle_NW3_L_Knee_Angle_x;
 else
    pAngle_NW3_L_Knee_Angle_x = minAngle_NW3_L_Knee_Angle_x;
 end
 pAnlge_idx_NW3_L_Knee_Angle_x = find(NW3_L_Knee_Angle_x ==
pAngle_NW3_L_Knee_Angle_x);
 t2p_NW3_L_Knee_Angle_x = (((pAnlge_idx_NW3_L_Knee_Angle_x-1)*0.005)-NW3_HS);


 NW3_L_Knee_Angle_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Knee_Angle_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Knee_Angle_y_subset =
NW3_L_Knee_Angle_y(NW3_L_Knee_Angle_y_Frame1:NW3_L_Knee_Angle_y_FrameN);
 maxAngle_NW3_L_Knee_Angle_y = max(NW3_L_Knee_Angle_y_subset);
 minAngle_NW3_L_Knee_Angle_y = min(NW3_L_Knee_Angle_y_subset);
 if maxAngle_NW3_L_Knee_Angle_y >(abs(minAngle_NW3_L_Knee_Angle_y))
    pAngle_NW3_L_Knee_Angle_y = maxAngle_NW3_L_Knee_Angle_y;
 else
    pAngle_NW3_L_Knee_Angle_y = minAngle_NW3_L_Knee_Angle_y;
 end
 pAnlge_idx_NW3_L_Knee_Angle_y = find(NW3_L_Knee_Angle_y ==
pAngle_NW3_L_Knee_Angle_y);
 t2p_NW3_L_Knee_Angle_y = (((pAnlge_idx_NW3_L_Knee_Angle_y-1)*0.005)-NW3_HS);


 NW3_L_Knee_Angle_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Knee_Angle_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Knee_Angle_z_subset =
NW3_L_Knee_Angle_z(NW3_L_Knee_Angle_z_Frame1:NW3_L_Knee_Angle_z_FrameN);
 maxAngle_NW3_L_Knee_Angle_z = max(NW3_L_Knee_Angle_z_subset);
 minAngle_NW3_L_Knee_Angle_z = min(NW3_L_Knee_Angle_z_subset);
 if maxAngle_NW3_L_Knee_Angle_z >(abs(minAngle_NW3_L_Knee_Angle_z))
    pAngle_NW3_L_Knee_Angle_z = maxAngle_NW3_L_Knee_Angle_z;
```

```
 else
    pAngle_NW3_L_Knee_Angle_z = minAngle_NW3_L_Knee_Angle_z;
 end
 pAnlge_idx_NW3_L_Knee_Angle_z = find(NW3_L_Knee_Angle_z ==
pAngle_NW3_L_Knee_Angle_z);
 t2p_NW3_L_Knee_Angle_z = (((pAnlge_idx_NW3_L_Knee_Angle_z-1)*0.005)-NW3_HS);


 NW3_L_Hip_Angle_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Hip_Angle_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Hip_Angle_x_subset =
NW3_L_Hip_Angle_x(NW3_L_Hip_Angle_x_Frame1:NW3_L_Hip_Angle_x_FrameN);
 maxAngle_NW3_L_Hip_Angle_x = max(NW3_L_Hip_Angle_x_subset);
 minAngle_NW3_L_Hip_Angle_x = min(NW3_L_Hip_Angle_x_subset);
 if maxAngle_NW3_L_Hip_Angle_x >(abs(minAngle_NW3_L_Hip_Angle_x))
    pAngle_NW3_L_Hip_Angle_x = maxAngle_NW3_L_Hip_Angle_x;
 else
    pAngle_NW3_L_Hip_Angle_x = minAngle_NW3_L_Hip_Angle_x;
 end
 pAnlge_idx_NW3_L_Hip_Angle_x = find(NW3_L_Hip_Angle_x ==
pAngle_NW3_L_Hip_Angle_x);
 t2p_NW3_L_Hip_Angle_x = (((pAnlge_idx_NW3_L_Hip_Angle_x-1)*0.005)-NW3_HS);


 NW3_L_Hip_Angle_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Hip_Angle_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Hip_Angle_y_subset =
NW3_L_Hip_Angle_y(NW3_L_Hip_Angle_y_Frame1:NW3_L_Hip_Angle_y_FrameN);
 maxAngle_NW3_L_Hip_Angle_y = max(NW3_L_Hip_Angle_y_subset);
 minAngle_NW3_L_Hip_Angle_y = min(NW3_L_Hip_Angle_y_subset);
 if maxAngle_NW3_L_Hip_Angle_y >(abs(minAngle_NW3_L_Hip_Angle_y))
    pAngle_NW3_L_Hip_Angle_y = maxAngle_NW3_L_Hip_Angle_y;
 else
    pAngle_NW3_L_Hip_Angle_y = minAngle_NW3_L_Hip_Angle_y;
 end
 pAnlge_idx_NW3_L_Hip_Angle_y = find(NW3_L_Hip_Angle_y ==
pAngle_NW3_L_Hip_Angle_y);
 t2p_NW3_L_Hip_Angle_y = (((pAnlge_idx_NW3_L_Hip_Angle_y-1)*0.005)-NW3_HS);


 NW3_L_Hip_Angle_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Hip_Angle_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Hip_Angle_z_subset =
NW3_L_Hip_Angle_z(NW3_L_Hip_Angle_z_Frame1:NW3_L_Hip_Angle_z_FrameN);
 maxAngle_NW3_L_Hip_Angle_z = max(NW3_L_Hip_Angle_z_subset);
 minAngle_NW3_L_Hip_Angle_z = min(NW3_L_Hip_Angle_z_subset);
```

```
 if maxAngle_NW3_L_Hip_Angle_z >(abs(minAngle_NW3_L_Hip_Angle_z))
    pAngle_NW3_L_Hip_Angle_z = maxAngle_NW3_L_Hip_Angle_z;
 else
    pAngle_NW3_L_Hip_Angle_z = minAngle_NW3_L_Hip_Angle_z;
 end
 pAnlge_idx_NW3_L_Hip_Angle_z = find(NW3_L_Hip_Angle_z ==
pAngle_NW3_L_Hip_Angle_z);
 t2p_NW3_L_Hip_Angle_z = (((pAnlge_idx_NW3_L_Hip_Angle_z-1)*0.005)-NW3_HS);


 %Right Side
 %ID1
 ID1_R_Ankle_Angle_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Ankle_Angle_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Ankle_Angle_x_subset =
ID1_R_Ankle_Angle_x(ID1_R_Ankle_Angle_x_Frame1:ID1_R_Ankle_Angle_x_FrameN);
 maxAngle_ID1_R_Ankle_Angle_x = max(ID1_R_Ankle_Angle_x_subset);
 minAngle_ID1_R_Ankle_Angle_x = min(ID1_R_Ankle_Angle_x_subset);
 if maxAngle_ID1_R_Ankle_Angle_x >(abs(minAngle_ID1_R_Ankle_Angle_x))
    pAngle_ID1_R_Ankle_Angle_x = maxAngle_ID1_R_Ankle_Angle_x;
 else
    pAngle_ID1_R_Ankle_Angle_x = minAngle_ID1_R_Ankle_Angle_x;
 end
 pAnlge_idx_ID1_R_Ankle_Angle_x = find(ID1_R_Ankle_Angle_x ==
pAngle_ID1_R_Ankle_Angle_x);
 t2p_ID1_R_Ankle_Angle_x = (((pAnlge_idx_ID1_R_Ankle_Angle_x-1)*0.005)-
ID1_True_Drop);

 ID1_R_Ankle_Angle_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Ankle_Angle_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Ankle_Angle_y_subset =
ID1_R_Ankle_Angle_y(ID1_R_Ankle_Angle_y_Frame1:ID1_R_Ankle_Angle_y_FrameN);
 maxAngle_ID1_R_Ankle_Angle_y = max(ID1_R_Ankle_Angle_y_subset);
 minAngle_ID1_R_Ankle_Angle_y = min(ID1_R_Ankle_Angle_y_subset);
 if maxAngle_ID1_R_Ankle_Angle_y >(abs(minAngle_ID1_R_Ankle_Angle_y))
    pAngle_ID1_R_Ankle_Angle_y = maxAngle_ID1_R_Ankle_Angle_y;
 else
    pAngle_ID1_R_Ankle_Angle_y = minAngle_ID1_R_Ankle_Angle_y;
 end
 pAnlge_idx_ID1_R_Ankle_Angle_y = find(ID1_R_Ankle_Angle_y ==
pAngle_ID1_R_Ankle_Angle_y);
 t2p_ID1_R_Ankle_Angle_y = (((pAnlge_idx_ID1_R_Ankle_Angle_y-1)*0.005)-
ID1_True_Drop);


 ID1_R_Ankle_Angle_z_Frame1 = round((ID1_True_Drop/.005)+1);
```

```
 ID1_R_Ankle_Angle_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Ankle_Angle_z_subset =
ID1_R_Ankle_Angle_z(ID1_R_Ankle_Angle_z_Frame1:ID1_R_Ankle_Angle_z_FrameN);
 maxAngle_ID1_R_Ankle_Angle_z = max(ID1_R_Ankle_Angle_z_subset);
 minAngle_ID1_R_Ankle_Angle_z = min(ID1_R_Ankle_Angle_z_subset);
 if maxAngle_ID1_R_Ankle_Angle_z >(abs(minAngle_ID1_R_Ankle_Angle_z))
    pAngle_ID1_R_Ankle_Angle_z = maxAngle_ID1_R_Ankle_Angle_z;
 else
    pAngle_ID1_R_Ankle_Angle_z = minAngle_ID1_R_Ankle_Angle_z;
 end
 pAnlge_idx_ID1_R_Ankle_Angle_z = find(ID1_R_Ankle_Angle_z ==
pAngle_ID1_R_Ankle_Angle_z);
 t2p_ID1_R_Ankle_Angle_z = (((pAnlge_idx_ID1_R_Ankle_Angle_z-1)*0.005)-
ID1_True_Drop);


 ID1_R_Knee_Angle_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Knee_Angle_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Knee_Angle_x_subset =
ID1_R_Knee_Angle_x(ID1_R_Knee_Angle_x_Frame1:ID1_R_Knee_Angle_x_FrameN);
 maxAngle_ID1_R_Knee_Angle_x = max(ID1_R_Knee_Angle_x_subset);
 minAngle_ID1_R_Knee_Angle_x = min(ID1_R_Knee_Angle_x_subset);
 if maxAngle_ID1_R_Knee_Angle_x >(abs(minAngle_ID1_R_Knee_Angle_x))
    pAngle_ID1_R_Knee_Angle_x = maxAngle_ID1_R_Knee_Angle_x;
 else
    pAngle_ID1_R_Knee_Angle_x = minAngle_ID1_R_Knee_Angle_x;
 end
 pAnlge_idx_ID1_R_Knee_Angle_x = find(ID1_R_Knee_Angle_x ==
pAngle_ID1_R_Knee_Angle_x);
 t2p_ID1_R_Knee_Angle_x = (((pAnlge_idx_ID1_R_Knee_Angle_x-1)*0.005)-
ID1_True_Drop);


 ID1_R_Knee_Angle_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Knee_Angle_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Knee_Angle_y_subset =
ID1_R_Knee_Angle_y(ID1_R_Knee_Angle_y_Frame1:ID1_R_Knee_Angle_y_FrameN);
 maxAngle_ID1_R_Knee_Angle_y = max(ID1_R_Knee_Angle_y_subset);
 minAngle_ID1_R_Knee_Angle_y = min(ID1_R_Knee_Angle_y_subset);
 if maxAngle_ID1_R_Knee_Angle_y >(abs(minAngle_ID1_R_Knee_Angle_y))
    pAngle_ID1_R_Knee_Angle_y = maxAngle_ID1_R_Knee_Angle_y;
 else
    pAngle_ID1_R_Knee_Angle_y = minAngle_ID1_R_Knee_Angle_y;
 end
 pAnlge_idx_ID1_R_Knee_Angle_y = find(ID1_R_Knee_Angle_y ==
pAngle_ID1_R_Knee_Angle_y);
```

```
 t2p_ID1_R_Knee_Angle_y = (((pAnlge_idx_ID1_R_Knee_Angle_y-1)*0.005)-
ID1_True_Drop);


 ID1_R_Knee_Angle_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Knee_Angle_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Knee_Angle_z_subset =
ID1_R_Knee_Angle_z(ID1_R_Knee_Angle_z_Frame1:ID1_R_Knee_Angle_z_FrameN);
 maxAngle_ID1_R_Knee_Angle_z = max(ID1_R_Knee_Angle_z_subset);
 minAngle_ID1_R_Knee_Angle_z = min(ID1_R_Knee_Angle_z_subset);
 if maxAngle_ID1_R_Knee_Angle_z >(abs(minAngle_ID1_R_Knee_Angle_z))
    pAngle_ID1_R_Knee_Angle_z = maxAngle_ID1_R_Knee_Angle_z;
 else
    pAngle_ID1_R_Knee_Angle_z = minAngle_ID1_R_Knee_Angle_z;
 end
 pAnlge_idx_ID1_R_Knee_Angle_z = find(ID1_R_Knee_Angle_z ==
pAngle_ID1_R_Knee_Angle_z);
 t2p_ID1_R_Knee_Angle_z = (((pAnlge_idx_ID1_R_Knee_Angle_z-1)*0.005)-
ID1_True_Drop);


 ID1_R_Hip_Angle_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Hip_Angle_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Hip_Angle_x_subset =
ID1_R_Hip_Angle_x(ID1_R_Hip_Angle_x_Frame1:ID1_R_Hip_Angle_x_FrameN);
 maxAngle_ID1_R_Hip_Angle_x = max(ID1_R_Hip_Angle_x_subset);
 minAngle_ID1_R_Hip_Angle_x = min(ID1_R_Hip_Angle_x_subset);
 if maxAngle_ID1_R_Hip_Angle_x >(abs(minAngle_ID1_R_Hip_Angle_x))
    pAngle_ID1_R_Hip_Angle_x = maxAngle_ID1_R_Hip_Angle_x;
 else
    pAngle_ID1_R_Hip_Angle_x = minAngle_ID1_R_Hip_Angle_x;
 end
 pAnlge_idx_ID1_R_Hip_Angle_x = find(ID1_R_Hip_Angle_x ==
pAngle_ID1_R_Hip_Angle_x);
 t2p_ID1_R_Hip_Angle_x = (((pAnlge_idx_ID1_R_Hip_Angle_x-1)*0.005)-ID1_True_Drop);


 ID1_R_Hip_Angle_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Hip_Angle_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Hip_Angle_y_subset =
ID1_R_Hip_Angle_y(ID1_R_Hip_Angle_y_Frame1:ID1_R_Hip_Angle_y_FrameN);
 maxAngle_ID1_R_Hip_Angle_y = max(ID1_R_Hip_Angle_y_subset);
 minAngle_ID1_R_Hip_Angle_y = min(ID1_R_Hip_Angle_y_subset);
 if maxAngle_ID1_R_Hip_Angle_y >(abs(minAngle_ID1_R_Hip_Angle_y))
    pAngle_ID1_R_Hip_Angle_y = maxAngle_ID1_R_Hip_Angle_y;
 else
```

```
    pAngle_ID1_R_Hip_Angle_y = minAngle_ID1_R_Hip_Angle_y;
 end
 pAnlge_idx_ID1_R_Hip_Angle_y = find(ID1_R_Hip_Angle_y ==
pAngle_ID1_R_Hip_Angle_y);
 t2p_ID1_R_Hip_Angle_y = (((pAnlge_idx_ID1_R_Hip_Angle_y-1)*0.005)-ID1_True_Drop);


 ID1_R_Hip_Angle_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Hip_Angle_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Hip_Angle_z_subset =
ID1_R_Hip_Angle_z(ID1_R_Hip_Angle_z_Frame1:ID1_R_Hip_Angle_z_FrameN);
 maxAngle_ID1_R_Hip_Angle_z = max(ID1_R_Hip_Angle_z_subset);
 minAngle_ID1_R_Hip_Angle_z = min(ID1_R_Hip_Angle_z_subset);
 if maxAngle_ID1_R_Hip_Angle_z >(abs(minAngle_ID1_R_Hip_Angle_z))
    pAngle_ID1_R_Hip_Angle_z = maxAngle_ID1_R_Hip_Angle_z;
 else
    pAngle_ID1_R_Hip_Angle_z = minAngle_ID1_R_Hip_Angle_z;
 end
 pAnlge_idx_ID1_R_Hip_Angle_z = find(ID1_R_Hip_Angle_z ==
pAngle_ID1_R_Hip_Angle_z);
 t2p_ID1_R_Hip_Angle_z = (((pAnlge_idx_ID1_R_Hip_Angle_z-1)*0.005)-ID1_True_Drop);


 % ID2
 ID2_R_Ankle_Angle_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Ankle_Angle_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Ankle_Angle_x_subset =
ID2_R_Ankle_Angle_x(ID2_R_Ankle_Angle_x_Frame1:ID2_R_Ankle_Angle_x_FrameN);
 maxAngle_ID2_R_Ankle_Angle_x = max(ID2_R_Ankle_Angle_x_subset);
 minAngle_ID2_R_Ankle_Angle_x = min(ID2_R_Ankle_Angle_x_subset);
 if maxAngle_ID2_R_Ankle_Angle_x >(abs(minAngle_ID2_R_Ankle_Angle_x))
    pAngle_ID2_R_Ankle_Angle_x = maxAngle_ID2_R_Ankle_Angle_x;
 else
    pAngle_ID2_R_Ankle_Angle_x = minAngle_ID2_R_Ankle_Angle_x;
 end
 pAnlge_idx_ID2_R_Ankle_Angle_x = find(ID2_R_Ankle_Angle_x ==
pAngle_ID2_R_Ankle_Angle_x);
 t2p_ID2_R_Ankle_Angle_x = (((pAnlge_idx_ID2_R_Ankle_Angle_x-1)*0.005)-
ID2_True_Drop);


 ID2_R_Ankle_Angle_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Ankle_Angle_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Ankle_Angle_y_subset =
ID2_R_Ankle_Angle_y(ID2_R_Ankle_Angle_y_Frame1:ID2_R_Ankle_Angle_y_FrameN);
 maxAngle_ID2_R_Ankle_Angle_y = max(ID2_R_Ankle_Angle_y_subset);
```

```
 minAngle_ID2_R_Ankle_Angle_y = min(ID2_R_Ankle_Angle_y_subset);
 if maxAngle_ID2_R_Ankle_Angle_y >(abs(minAngle_ID2_R_Ankle_Angle_y))
    pAngle_ID2_R_Ankle_Angle_y = maxAngle_ID2_R_Ankle_Angle_y;
 else
    pAngle_ID2_R_Ankle_Angle_y = minAngle_ID2_R_Ankle_Angle_y;
 end
 pAnlge_idx_ID2_R_Ankle_Angle_y = find(ID2_R_Ankle_Angle_y ==
pAngle_ID2_R_Ankle_Angle_y);
 t2p_ID2_R_Ankle_Angle_y = (((pAnlge_idx_ID2_R_Ankle_Angle_y-1)*0.005)-
ID2_True_Drop);


 ID2_R_Ankle_Angle_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Ankle_Angle_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Ankle_Angle_z_subset =
ID2_R_Ankle_Angle_z(ID2_R_Ankle_Angle_z_Frame1:ID2_R_Ankle_Angle_z_FrameN);
 maxAngle_ID2_R_Ankle_Angle_z = max(ID2_R_Ankle_Angle_z_subset);
 minAngle_ID2_R_Ankle_Angle_z = min(ID2_R_Ankle_Angle_z_subset);
 if maxAngle_ID2_R_Ankle_Angle_z >(abs(minAngle_ID2_R_Ankle_Angle_z))
    pAngle_ID2_R_Ankle_Angle_z = maxAngle_ID2_R_Ankle_Angle_z;
 else
    pAngle_ID2_R_Ankle_Angle_z = minAngle_ID2_R_Ankle_Angle_z;
 end
 pAnlge_idx_ID2_R_Ankle_Angle_z = find(ID2_R_Ankle_Angle_z ==
pAngle_ID2_R_Ankle_Angle_z);
 t2p_ID2_R_Ankle_Angle_z = (((pAnlge_idx_ID2_R_Ankle_Angle_z-1)*0.005)-
ID2_True_Drop);


 ID2_R_Knee_Angle_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Knee_Angle_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Knee_Angle_x_subset =
ID2_R_Knee_Angle_x(ID2_R_Knee_Angle_x_Frame1:ID2_R_Knee_Angle_x_FrameN);
 maxAngle_ID2_R_Knee_Angle_x = max(ID2_R_Knee_Angle_x_subset);
 minAngle_ID2_R_Knee_Angle_x = min(ID2_R_Knee_Angle_x_subset);
 if maxAngle_ID2_R_Knee_Angle_x >(abs(minAngle_ID2_R_Knee_Angle_x))
    pAngle_ID2_R_Knee_Angle_x = maxAngle_ID2_R_Knee_Angle_x;
 else
    pAngle_ID2_R_Knee_Angle_x = minAngle_ID2_R_Knee_Angle_x;
 end
 pAnlge_idx_ID2_R_Knee_Angle_x = find(ID2_R_Knee_Angle_x ==
pAngle_ID2_R_Knee_Angle_x);
 t2p_ID2_R_Knee_Angle_x = (((pAnlge_idx_ID2_R_Knee_Angle_x-1)*0.005)-
ID2_True_Drop);
```

```
 ID2_R_Knee_Angle_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Knee_Angle_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Knee_Angle_y_subset =
ID2_R_Knee_Angle_y(ID2_R_Knee_Angle_y_Frame1:ID2_R_Knee_Angle_y_FrameN);
 maxAngle_ID2_R_Knee_Angle_y = max(ID2_R_Knee_Angle_y_subset);
 minAngle_ID2_R_Knee_Angle_y = min(ID2_R_Knee_Angle_y_subset);
 if maxAngle_ID2_R_Knee_Angle_y >(abs(minAngle_ID2_R_Knee_Angle_y))
    pAngle_ID2_R_Knee_Angle_y = maxAngle_ID2_R_Knee_Angle_y;
 else
    pAngle_ID2_R_Knee_Angle_y = minAngle_ID2_R_Knee_Angle_y;
 end
 pAnlge_idx_ID2_R_Knee_Angle_y = find(ID2_R_Knee_Angle_y ==
pAngle_ID2_R_Knee_Angle_y);
 t2p_ID2_R_Knee_Angle_y = (((pAnlge_idx_ID2_R_Knee_Angle_y-1)*0.005)-
ID2_True_Drop);


 ID2_R_Knee_Angle_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Knee_Angle_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Knee_Angle_z_subset =
ID2_R_Knee_Angle_z(ID2_R_Knee_Angle_z_Frame1:ID2_R_Knee_Angle_z_FrameN);
 maxAngle_ID2_R_Knee_Angle_z = max(ID2_R_Knee_Angle_z_subset);
 minAngle_ID2_R_Knee_Angle_z = min(ID2_R_Knee_Angle_z_subset);
 if maxAngle_ID2_R_Knee_Angle_z >(abs(minAngle_ID2_R_Knee_Angle_z))
    pAngle_ID2_R_Knee_Angle_z = maxAngle_ID2_R_Knee_Angle_z;
 else
    pAngle_ID2_R_Knee_Angle_z = minAngle_ID2_R_Knee_Angle_z;
 end
 pAnlge_idx_ID2_R_Knee_Angle_z = find(ID2_R_Knee_Angle_z ==
pAngle_ID2_R_Knee_Angle_z);
 t2p_ID2_R_Knee_Angle_z = (((pAnlge_idx_ID2_R_Knee_Angle_z-1)*0.005)-
ID2_True_Drop);


 ID2_R_Hip_Angle_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Hip_Angle_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Hip_Angle_x_subset =
ID2_R_Hip_Angle_x(ID2_R_Hip_Angle_x_Frame1:ID2_R_Hip_Angle_x_FrameN);
 maxAngle_ID2_R_Hip_Angle_x = max(ID2_R_Hip_Angle_x_subset);
 minAngle_ID2_R_Hip_Angle_x = min(ID2_R_Hip_Angle_x_subset);
 if maxAngle_ID2_R_Hip_Angle_x >(abs(minAngle_ID2_R_Hip_Angle_x))
    pAngle_ID2_R_Hip_Angle_x = maxAngle_ID2_R_Hip_Angle_x;
 else
    pAngle_ID2_R_Hip_Angle_x = minAngle_ID2_R_Hip_Angle_x;
 end
```

```
 pAnlge_idx_ID2_R_Hip_Angle_x = find(ID2_R_Hip_Angle_x ==
pAngle_ID2_R_Hip_Angle_x);
 t2p_ID2_R_Hip_Angle_x = (((pAnlge_idx_ID2_R_Hip_Angle_x-1)*0.005)-ID2_True_Drop);


 ID2_R_Hip_Angle_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Hip_Angle_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Hip_Angle_y_subset =
ID2_R_Hip_Angle_y(ID2_R_Hip_Angle_y_Frame1:ID2_R_Hip_Angle_y_FrameN);
 maxAngle_ID2_R_Hip_Angle_y = max(ID2_R_Hip_Angle_y_subset);
 minAngle_ID2_R_Hip_Angle_y = min(ID2_R_Hip_Angle_y_subset);
 if maxAngle_ID2_R_Hip_Angle_y >(abs(minAngle_ID2_R_Hip_Angle_y))
    pAngle_ID2_R_Hip_Angle_y = maxAngle_ID2_R_Hip_Angle_y;
 else
    pAngle_ID2_R_Hip_Angle_y = minAngle_ID2_R_Hip_Angle_y;
 end
 pAnlge_idx_ID2_R_Hip_Angle_y = find(ID2_R_Hip_Angle_y ==
pAngle_ID2_R_Hip_Angle_y);
 t2p_ID2_R_Hip_Angle_y = (((pAnlge_idx_ID2_R_Hip_Angle_y-1)*0.005)-ID2_True_Drop);


 ID2_R_Hip_Angle_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Hip_Angle_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Hip_Angle_z_subset =
ID2_R_Hip_Angle_z(ID2_R_Hip_Angle_z_Frame1:ID2_R_Hip_Angle_z_FrameN);
 maxAngle_ID2_R_Hip_Angle_z = max(ID2_R_Hip_Angle_z_subset);
 minAngle_ID2_R_Hip_Angle_z = min(ID2_R_Hip_Angle_z_subset);
 if maxAngle_ID2_R_Hip_Angle_z >(abs(minAngle_ID2_R_Hip_Angle_z))
    pAngle_ID2_R_Hip_Angle_z = maxAngle_ID2_R_Hip_Angle_z;
 else
    pAngle_ID2_R_Hip_Angle_z = minAngle_ID2_R_Hip_Angle_z;
 end
 pAnlge_idx_ID2_R_Hip_Angle_z = find(ID2_R_Hip_Angle_z ==
pAngle_ID2_R_Hip_Angle_z);
 t2p_ID2_R_Hip_Angle_z = (((pAnlge_idx_ID2_R_Hip_Angle_z-1)*0.005)-ID2_True_Drop);

 %ID3
 ID3_R_Ankle_Angle_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Ankle_Angle_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Ankle_Angle_x_subset =
ID3_R_Ankle_Angle_x(ID3_R_Ankle_Angle_x_Frame1:ID3_R_Ankle_Angle_x_FrameN);
 maxAngle_ID3_R_Ankle_Angle_x = max(ID3_R_Ankle_Angle_x_subset);
 minAngle_ID3_R_Ankle_Angle_x = min(ID3_R_Ankle_Angle_x_subset);
 if maxAngle_ID3_R_Ankle_Angle_x >(abs(minAngle_ID3_R_Ankle_Angle_x))
    pAngle_ID3_R_Ankle_Angle_x = maxAngle_ID3_R_Ankle_Angle_x;
 else
```

```
    pAngle_ID3_R_Ankle_Angle_x = minAngle_ID3_R_Ankle_Angle_x;
 end
 pAnlge_idx_ID3_R_Ankle_Angle_x = find(ID3_R_Ankle_Angle_x ==
pAngle_ID3_R_Ankle_Angle_x);
 t2p_ID3_R_Ankle_Angle_x = (((pAnlge_idx_ID3_R_Ankle_Angle_x-1)*0.005)-
ID3_True_Drop);


 ID3_R_Ankle_Angle_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Ankle_Angle_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Ankle_Angle_y_subset =
ID3_R_Ankle_Angle_y(ID3_R_Ankle_Angle_y_Frame1:ID3_R_Ankle_Angle_y_FrameN);
 maxAngle_ID3_R_Ankle_Angle_y = max(ID3_R_Ankle_Angle_y_subset);
 minAngle_ID3_R_Ankle_Angle_y = min(ID3_R_Ankle_Angle_y_subset);
 if maxAngle_ID3_R_Ankle_Angle_y >(abs(minAngle_ID3_R_Ankle_Angle_y))
    pAngle_ID3_R_Ankle_Angle_y = maxAngle_ID3_R_Ankle_Angle_y;
 else
    pAngle_ID3_R_Ankle_Angle_y = minAngle_ID3_R_Ankle_Angle_y;
 end
 pAnlge_idx_ID3_R_Ankle_Angle_y = find(ID3_R_Ankle_Angle_y ==
pAngle_ID3_R_Ankle_Angle_y);
 t2p_ID3_R_Ankle_Angle_y = (((pAnlge_idx_ID3_R_Ankle_Angle_y-1)*0.005)-
ID3_True_Drop);


 ID3_R_Ankle_Angle_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Ankle_Angle_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Ankle_Angle_z_subset =
ID3_R_Ankle_Angle_z(ID3_R_Ankle_Angle_z_Frame1:ID3_R_Ankle_Angle_z_FrameN);
 maxAngle_ID3_R_Ankle_Angle_z = max(ID3_R_Ankle_Angle_z_subset);
 minAngle_ID3_R_Ankle_Angle_z = min(ID3_R_Ankle_Angle_z_subset);
 if maxAngle_ID3_R_Ankle_Angle_z >(abs(minAngle_ID3_R_Ankle_Angle_z))
    pAngle_ID3_R_Ankle_Angle_z = maxAngle_ID3_R_Ankle_Angle_z;
 else
    pAngle_ID3_R_Ankle_Angle_z = minAngle_ID3_R_Ankle_Angle_z;
 end
 pAnlge_idx_ID3_R_Ankle_Angle_z = find(ID3_R_Ankle_Angle_z ==
pAngle_ID3_R_Ankle_Angle_z);
 t2p_ID3_R_Ankle_Angle_z = (((pAnlge_idx_ID3_R_Ankle_Angle_z-1)*0.005)-
ID3_True_Drop);


 ID3_R_Knee_Angle_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Knee_Angle_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Knee_Angle_x_subset =
ID3_R_Knee_Angle_x(ID3_R_Knee_Angle_x_Frame1:ID3_R_Knee_Angle_x_FrameN);
```

```
 maxAngle_ID3_R_Knee_Angle_x = max(ID3_R_Knee_Angle_x_subset);
 minAngle_ID3_R_Knee_Angle_x = min(ID3_R_Knee_Angle_x_subset);
 if maxAngle_ID3_R_Knee_Angle_x >(abs(minAngle_ID3_R_Knee_Angle_x))
    pAngle_ID3_R_Knee_Angle_x = maxAngle_ID3_R_Knee_Angle_x;
 else
    pAngle_ID3_R_Knee_Angle_x = minAngle_ID3_R_Knee_Angle_x;
 end
 pAnlge_idx_ID3_R_Knee_Angle_x = find(ID3_R_Knee_Angle_x ==
pAngle_ID3_R_Knee_Angle_x);
 t2p_ID3_R_Knee_Angle_x = (((pAnlge_idx_ID3_R_Knee_Angle_x-1)*0.005)-
ID3_True_Drop);


 ID3_R_Knee_Angle_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Knee_Angle_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Knee_Angle_y_subset =
ID3_R_Knee_Angle_y(ID3_R_Knee_Angle_y_Frame1:ID3_R_Knee_Angle_y_FrameN);
 maxAngle_ID3_R_Knee_Angle_y = max(ID3_R_Knee_Angle_y_subset);
 minAngle_ID3_R_Knee_Angle_y = min(ID3_R_Knee_Angle_y_subset);
 if maxAngle_ID3_R_Knee_Angle_y >(abs(minAngle_ID3_R_Knee_Angle_y))
    pAngle_ID3_R_Knee_Angle_y = maxAngle_ID3_R_Knee_Angle_y;
 else
    pAngle_ID3_R_Knee_Angle_y = minAngle_ID3_R_Knee_Angle_y;
 end
 pAnlge_idx_ID3_R_Knee_Angle_y = find(ID3_R_Knee_Angle_y ==
pAngle_ID3_R_Knee_Angle_y);
 t2p_ID3_R_Knee_Angle_y = (((pAnlge_idx_ID3_R_Knee_Angle_y-1)*0.005)-
ID3_True_Drop);


 ID3_R_Knee_Angle_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Knee_Angle_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Knee_Angle_z_subset =
ID3_R_Knee_Angle_z(ID3_R_Knee_Angle_z_Frame1:ID3_R_Knee_Angle_z_FrameN);
 maxAngle_ID3_R_Knee_Angle_z = max(ID3_R_Knee_Angle_z_subset);
 minAngle_ID3_R_Knee_Angle_z = min(ID3_R_Knee_Angle_z_subset);
 if maxAngle_ID3_R_Knee_Angle_z >(abs(minAngle_ID3_R_Knee_Angle_z))
    pAngle_ID3_R_Knee_Angle_z = maxAngle_ID3_R_Knee_Angle_z;
 else
    pAngle_ID3_R_Knee_Angle_z = minAngle_ID3_R_Knee_Angle_z;
 end
 pAnlge_idx_ID3_R_Knee_Angle_z = find(ID3_R_Knee_Angle_z ==
pAngle_ID3_R_Knee_Angle_z);
 t2p_ID3_R_Knee_Angle_z = (((pAnlge_idx_ID3_R_Knee_Angle_z-1)*0.005)-
ID3_True_Drop);
```

```
 ID3_R_Hip_Angle_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Hip_Angle_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Hip_Angle_x_subset =
ID3_R_Hip_Angle_x(ID3_R_Hip_Angle_x_Frame1:ID3_R_Hip_Angle_x_FrameN);
 maxAngle_ID3_R_Hip_Angle_x = max(ID3_R_Hip_Angle_x_subset);
 minAngle_ID3_R_Hip_Angle_x = min(ID3_R_Hip_Angle_x_subset);
 if maxAngle_ID3_R_Hip_Angle_x >(abs(minAngle_ID3_R_Hip_Angle_x))
    pAngle_ID3_R_Hip_Angle_x = maxAngle_ID3_R_Hip_Angle_x;
 else
    pAngle_ID3_R_Hip_Angle_x = minAngle_ID3_R_Hip_Angle_x;
 end
 pAnlge_idx_ID3_R_Hip_Angle_x = find(ID3_R_Hip_Angle_x ==
pAngle_ID3_R_Hip_Angle_x);
 t2p_ID3_R_Hip_Angle_x = (((pAnlge_idx_ID3_R_Hip_Angle_x-1)*0.005)-ID3_True_Drop);


 ID3_R_Hip_Angle_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Hip_Angle_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Hip_Angle_y_subset =
ID3_R_Hip_Angle_y(ID3_R_Hip_Angle_y_Frame1:ID3_R_Hip_Angle_y_FrameN);
 maxAngle_ID3_R_Hip_Angle_y = max(ID3_R_Hip_Angle_y_subset);
 minAngle_ID3_R_Hip_Angle_y = min(ID3_R_Hip_Angle_y_subset);
 if maxAngle_ID3_R_Hip_Angle_y >(abs(minAngle_ID3_R_Hip_Angle_y))
    pAngle_ID3_R_Hip_Angle_y = maxAngle_ID3_R_Hip_Angle_y;
 else
    pAngle_ID3_R_Hip_Angle_y = minAngle_ID3_R_Hip_Angle_y;
 end
 pAnlge_idx_ID3_R_Hip_Angle_y = find(ID3_R_Hip_Angle_y ==
pAngle_ID3_R_Hip_Angle_y);
 t2p_ID3_R_Hip_Angle_y = (((pAnlge_idx_ID3_R_Hip_Angle_y-1)*0.005)-ID3_True_Drop);


 ID3_R_Hip_Angle_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Hip_Angle_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Hip_Angle_z_subset =
ID3_R_Hip_Angle_z(ID3_R_Hip_Angle_z_Frame1:ID3_R_Hip_Angle_z_FrameN);
 maxAngle_ID3_R_Hip_Angle_z = max(ID3_R_Hip_Angle_z_subset);
 minAngle_ID3_R_Hip_Angle_z = min(ID3_R_Hip_Angle_z_subset);
 if maxAngle_ID3_R_Hip_Angle_z >(abs(minAngle_ID3_R_Hip_Angle_z))
    pAngle_ID3_R_Hip_Angle_z = maxAngle_ID3_R_Hip_Angle_z;
 else
    pAngle_ID3_R_Hip_Angle_z = minAngle_ID3_R_Hip_Angle_z;
 end
 pAnlge_idx_ID3_R_Hip_Angle_z = find(ID3_R_Hip_Angle_z ==
pAngle_ID3_R_Hip_Angle_z);
```

```
t2p_ID3_R_Hip_Angle_z = (((pAnlge_idx_ID3_R_Hip_Angle_z-1)*0.005)-ID3_True_Drop);


%IPD1
 IPD1_R_Ankle_Angle_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Ankle_Angle_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Ankle_Angle_x_subset =
IPD1_R_Ankle_Angle_x(IPD1_R_Ankle_Angle_x_Frame1:IPD1_R_Ankle_Angle_x_FrameN)
;
 maxAngle_IPD1_R_Ankle_Angle_x = max(IPD1_R_Ankle_Angle_x_subset);
 minAngle_IPD1_R_Ankle_Angle_x = min(IPD1_R_Ankle_Angle_x_subset);
 if maxAngle_IPD1_R_Ankle_Angle_x >(abs(minAngle_IPD1_R_Ankle_Angle_x))
    pAngle_IPD1_R_Ankle_Angle_x = maxAngle_IPD1_R_Ankle_Angle_x;
 else
    pAngle_IPD1_R_Ankle_Angle_x = minAngle_IPD1_R_Ankle_Angle_x;
 end
 pAnlge_idx_IPD1_R_Ankle_Angle_x = find(IPD1_R_Ankle_Angle_x ==
pAngle_IPD1_R_Ankle_Angle_x);
 t2p_IPD1_R_Ankle_Angle_x = (((pAnlge_idx_IPD1_R_Ankle_Angle_x-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Ankle_Angle_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Ankle_Angle_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Ankle_Angle_y_subset =
IPD1_R_Ankle_Angle_y(IPD1_R_Ankle_Angle_y_Frame1:IPD1_R_Ankle_Angle_y_FrameN)
;
 maxAngle_IPD1_R_Ankle_Angle_y = max(IPD1_R_Ankle_Angle_y_subset);
 minAngle_IPD1_R_Ankle_Angle_y = min(IPD1_R_Ankle_Angle_y_subset);
 if maxAngle_IPD1_R_Ankle_Angle_y >(abs(minAngle_IPD1_R_Ankle_Angle_y))
    pAngle_IPD1_R_Ankle_Angle_y = maxAngle_IPD1_R_Ankle_Angle_y;
 else
    pAngle_IPD1_R_Ankle_Angle_y = minAngle_IPD1_R_Ankle_Angle_y;
 end
 pAnlge_idx_IPD1_R_Ankle_Angle_y = find(IPD1_R_Ankle_Angle_y ==
pAngle_IPD1_R_Ankle_Angle_y);
 t2p_IPD1_R_Ankle_Angle_y = (((pAnlge_idx_IPD1_R_Ankle_Angle_y-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Ankle_Angle_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Ankle_Angle_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Ankle_Angle_z_subset =
IPD1_R_Ankle_Angle_z(IPD1_R_Ankle_Angle_z_Frame1:IPD1_R_Ankle_Angle_z_FrameN);
 maxAngle_IPD1_R_Ankle_Angle_z = max(IPD1_R_Ankle_Angle_z_subset);
```

```
 minAngle_IPD1_R_Ankle_Angle_z = min(IPD1_R_Ankle_Angle_z_subset);
 if maxAngle_IPD1_R_Ankle_Angle_z >(abs(minAngle_IPD1_R_Ankle_Angle_z))
    pAngle_IPD1_R_Ankle_Angle_z = maxAngle_IPD1_R_Ankle_Angle_z;
 else
    pAngle_IPD1_R_Ankle_Angle_z = minAngle_IPD1_R_Ankle_Angle_z;
 end
 pAnlge_idx_IPD1_R_Ankle_Angle_z = find(IPD1_R_Ankle_Angle_z ==
pAngle_IPD1_R_Ankle_Angle_z);
 t2p_IPD1_R_Ankle_Angle_z = (((pAnlge_idx_IPD1_R_Ankle_Angle_z-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Knee_Angle_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Knee_Angle_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Knee_Angle_x_subset =
IPD1_R_Knee_Angle_x(IPD1_R_Knee_Angle_x_Frame1:IPD1_R_Knee_Angle_x_FrameN);
 maxAngle_IPD1_R_Knee_Angle_x = max(IPD1_R_Knee_Angle_x_subset);
 minAngle_IPD1_R_Knee_Angle_x = min(IPD1_R_Knee_Angle_x_subset);
 if maxAngle_IPD1_R_Knee_Angle_x >(abs(minAngle_IPD1_R_Knee_Angle_x))
    pAngle_IPD1_R_Knee_Angle_x = maxAngle_IPD1_R_Knee_Angle_x;
 else
    pAngle_IPD1_R_Knee_Angle_x = minAngle_IPD1_R_Knee_Angle_x;
 end
 pAnlge_idx_IPD1_R_Knee_Angle_x = find(IPD1_R_Knee_Angle_x ==
pAngle_IPD1_R_Knee_Angle_x);
 t2p_IPD1_R_Knee_Angle_x = (((pAnlge_idx_IPD1_R_Knee_Angle_x-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Knee_Angle_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Knee_Angle_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Knee_Angle_y_subset =
IPD1_R_Knee_Angle_y(IPD1_R_Knee_Angle_y_Frame1:IPD1_R_Knee_Angle_y_FrameN);
 maxAngle_IPD1_R_Knee_Angle_y = max(IPD1_R_Knee_Angle_y_subset);
 minAngle_IPD1_R_Knee_Angle_y = min(IPD1_R_Knee_Angle_y_subset);
 if maxAngle_IPD1_R_Knee_Angle_y >(abs(minAngle_IPD1_R_Knee_Angle_y))
    pAngle_IPD1_R_Knee_Angle_y = maxAngle_IPD1_R_Knee_Angle_y;
 else
    pAngle_IPD1_R_Knee_Angle_y = minAngle_IPD1_R_Knee_Angle_y;
 end
 pAnlge_idx_IPD1_R_Knee_Angle_y = find(IPD1_R_Knee_Angle_y ==
pAngle_IPD1_R_Knee_Angle_y);
 t2p_IPD1_R_Knee_Angle_y = (((pAnlge_idx_IPD1_R_Knee_Angle_y-1)*0.005)-
IPD1_True_Drop);
```

```
 IPD1_R_Knee_Angle_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Knee_Angle_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Knee_Angle_z_subset =
IPD1_R_Knee_Angle_z(IPD1_R_Knee_Angle_z_Frame1:IPD1_R_Knee_Angle_z_FrameN);
 maxAngle_IPD1_R_Knee_Angle_z = max(IPD1_R_Knee_Angle_z_subset);
 minAngle_IPD1_R_Knee_Angle_z = min(IPD1_R_Knee_Angle_z_subset);
 if maxAngle_IPD1_R_Knee_Angle_z >(abs(minAngle_IPD1_R_Knee_Angle_z))
    pAngle_IPD1_R_Knee_Angle_z = maxAngle_IPD1_R_Knee_Angle_z;
 else
    pAngle_IPD1_R_Knee_Angle_z = minAngle_IPD1_R_Knee_Angle_z;
 end
 pAnlge_idx_IPD1_R_Knee_Angle_z = find(IPD1_R_Knee_Angle_z ==
pAngle_IPD1_R_Knee_Angle_z);
 t2p_IPD1_R_Knee_Angle_z = (((pAnlge_idx_IPD1_R_Knee_Angle_z-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Hip_Angle_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Hip_Angle_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Hip_Angle_x_subset =
IPD1_R_Hip_Angle_x(IPD1_R_Hip_Angle_x_Frame1:IPD1_R_Hip_Angle_x_FrameN);
 maxAngle_IPD1_R_Hip_Angle_x = max(IPD1_R_Hip_Angle_x_subset);
 minAngle_IPD1_R_Hip_Angle_x = min(IPD1_R_Hip_Angle_x_subset);
 if maxAngle_IPD1_R_Hip_Angle_x >(abs(minAngle_IPD1_R_Hip_Angle_x))
    pAngle_IPD1_R_Hip_Angle_x = maxAngle_IPD1_R_Hip_Angle_x;
 else
    pAngle_IPD1_R_Hip_Angle_x = minAngle_IPD1_R_Hip_Angle_x;
 end
 pAnlge_idx_IPD1_R_Hip_Angle_x = find(IPD1_R_Hip_Angle_x ==
pAngle_IPD1_R_Hip_Angle_x);
 t2p_IPD1_R_Hip_Angle_x = (((pAnlge_idx_IPD1_R_Hip_Angle_x-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Hip_Angle_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Hip_Angle_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Hip_Angle_y_subset =
IPD1_R_Hip_Angle_y(IPD1_R_Hip_Angle_y_Frame1:IPD1_R_Hip_Angle_y_FrameN);
 maxAngle_IPD1_R_Hip_Angle_y = max(IPD1_R_Hip_Angle_y_subset);
 minAngle_IPD1_R_Hip_Angle_y = min(IPD1_R_Hip_Angle_y_subset);
 if maxAngle_IPD1_R_Hip_Angle_y >(abs(minAngle_IPD1_R_Hip_Angle_y))
    pAngle_IPD1_R_Hip_Angle_y = maxAngle_IPD1_R_Hip_Angle_y;
 else
    pAngle_IPD1_R_Hip_Angle_y = minAngle_IPD1_R_Hip_Angle_y;
 end
```

```
 pAnlge_idx_IPD1_R_Hip_Angle_y = find(IPD1_R_Hip_Angle_y ==
pAngle_IPD1_R_Hip_Angle_y);
 t2p_IPD1_R_Hip_Angle_y = (((pAnlge_idx_IPD1_R_Hip_Angle_y-1)*0.005)-
IPD1_True_Drop);


 IPD1_R_Hip_Angle_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Hip_Angle_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Hip_Angle_z_subset =
IPD1_R_Hip_Angle_z(IPD1_R_Hip_Angle_z_Frame1:IPD1_R_Hip_Angle_z_FrameN);
 maxAngle_IPD1_R_Hip_Angle_z = max(IPD1_R_Hip_Angle_z_subset);
 minAngle_IPD1_R_Hip_Angle_z = min(IPD1_R_Hip_Angle_z_subset);
 if maxAngle_IPD1_R_Hip_Angle_z >(abs(minAngle_IPD1_R_Hip_Angle_z))
    pAngle_IPD1_R_Hip_Angle_z = maxAngle_IPD1_R_Hip_Angle_z;
 else
    pAngle_IPD1_R_Hip_Angle_z = minAngle_IPD1_R_Hip_Angle_z;
 end
 pAnlge_idx_IPD1_R_Hip_Angle_z = find(IPD1_R_Hip_Angle_z ==
pAngle_IPD1_R_Hip_Angle_z);
 t2p_IPD1_R_Hip_Angle_z = (((pAnlge_idx_IPD1_R_Hip_Angle_z-1)*0.005)-
IPD1_True_Drop);

 %IPD2
 IPD2_R_Ankle_Angle_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Ankle_Angle_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Ankle_Angle_x_subset =
IPD2_R_Ankle_Angle_x(IPD2_R_Ankle_Angle_x_Frame1:IPD2_R_Ankle_Angle_x_FrameN)
;
 maxAngle_IPD2_R_Ankle_Angle_x = max(IPD2_R_Ankle_Angle_x_subset);
 minAngle_IPD2_R_Ankle_Angle_x = min(IPD2_R_Ankle_Angle_x_subset);
 if maxAngle_IPD2_R_Ankle_Angle_x >(abs(minAngle_IPD2_R_Ankle_Angle_x))
    pAngle_IPD2_R_Ankle_Angle_x = maxAngle_IPD2_R_Ankle_Angle_x;
 else
    pAngle_IPD2_R_Ankle_Angle_x = minAngle_IPD2_R_Ankle_Angle_x;
 end
 pAnlge_idx_IPD2_R_Ankle_Angle_x = find(IPD2_R_Ankle_Angle_x ==
pAngle_IPD2_R_Ankle_Angle_x);
 t2p_IPD2_R_Ankle_Angle_x = (((pAnlge_idx_IPD2_R_Ankle_Angle_x-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Ankle_Angle_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Ankle_Angle_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Ankle_Angle_y_subset =
IPD2_R_Ankle_Angle_y(IPD2_R_Ankle_Angle_y_Frame1:IPD2_R_Ankle_Angle_y_FrameN)
;
```

```
 maxAngle_IPD2_R_Ankle_Angle_y = max(IPD2_R_Ankle_Angle_y_subset);
 minAngle_IPD2_R_Ankle_Angle_y = min(IPD2_R_Ankle_Angle_y_subset);
 if maxAngle_IPD2_R_Ankle_Angle_y >(abs(minAngle_IPD2_R_Ankle_Angle_y))
    pAngle_IPD2_R_Ankle_Angle_y = maxAngle_IPD2_R_Ankle_Angle_y;
 else
    pAngle_IPD2_R_Ankle_Angle_y = minAngle_IPD2_R_Ankle_Angle_y;
 end
 pAnlge_idx_IPD2_R_Ankle_Angle_y = find(IPD2_R_Ankle_Angle_y ==
pAngle_IPD2_R_Ankle_Angle_y);
 t2p_IPD2_R_Ankle_Angle_y = (((pAnlge_idx_IPD2_R_Ankle_Angle_y-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Ankle_Angle_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Ankle_Angle_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Ankle_Angle_z_subset =
IPD2_R_Ankle_Angle_z(IPD2_R_Ankle_Angle_z_Frame1:IPD2_R_Ankle_Angle_z_FrameN);
 maxAngle_IPD2_R_Ankle_Angle_z = max(IPD2_R_Ankle_Angle_z_subset);
 minAngle_IPD2_R_Ankle_Angle_z = min(IPD2_R_Ankle_Angle_z_subset);
 if maxAngle_IPD2_R_Ankle_Angle_z >(abs(minAngle_IPD2_R_Ankle_Angle_z))
    pAngle_IPD2_R_Ankle_Angle_z = maxAngle_IPD2_R_Ankle_Angle_z;
 else
    pAngle_IPD2_R_Ankle_Angle_z = minAngle_IPD2_R_Ankle_Angle_z;
 end
 pAnlge_idx_IPD2_R_Ankle_Angle_z = find(IPD2_R_Ankle_Angle_z ==
pAngle_IPD2_R_Ankle_Angle_z);
 t2p_IPD2_R_Ankle_Angle_z = (((pAnlge_idx_IPD2_R_Ankle_Angle_z-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Knee_Angle_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Knee_Angle_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Knee_Angle_x_subset =
IPD2_R_Knee_Angle_x(IPD2_R_Knee_Angle_x_Frame1:IPD2_R_Knee_Angle_x_FrameN);
 maxAngle_IPD2_R_Knee_Angle_x = max(IPD2_R_Knee_Angle_x_subset);
 minAngle_IPD2_R_Knee_Angle_x = min(IPD2_R_Knee_Angle_x_subset);
 if maxAngle_IPD2_R_Knee_Angle_x >(abs(minAngle_IPD2_R_Knee_Angle_x))
    pAngle_IPD2_R_Knee_Angle_x = maxAngle_IPD2_R_Knee_Angle_x;
 else
    pAngle_IPD2_R_Knee_Angle_x = minAngle_IPD2_R_Knee_Angle_x;
 end
 pAnlge_idx_IPD2_R_Knee_Angle_x = find(IPD2_R_Knee_Angle_x ==
pAngle_IPD2_R_Knee_Angle_x);
 t2p_IPD2_R_Knee_Angle_x = (((pAnlge_idx_IPD2_R_Knee_Angle_x-1)*0.005)-
IPD2_True_Drop);
```

```
 IPD2_R_Knee_Angle_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Knee_Angle_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Knee_Angle_y_subset =
IPD2_R_Knee_Angle_y(IPD2_R_Knee_Angle_y_Frame1:IPD2_R_Knee_Angle_y_FrameN);
 maxAngle_IPD2_R_Knee_Angle_y = max(IPD2_R_Knee_Angle_y_subset);
 minAngle_IPD2_R_Knee_Angle_y = min(IPD2_R_Knee_Angle_y_subset);
 if maxAngle_IPD2_R_Knee_Angle_y >(abs(minAngle_IPD2_R_Knee_Angle_y))
    pAngle_IPD2_R_Knee_Angle_y = maxAngle_IPD2_R_Knee_Angle_y;
 else
    pAngle_IPD2_R_Knee_Angle_y = minAngle_IPD2_R_Knee_Angle_y;
 end
 pAnlge_idx_IPD2_R_Knee_Angle_y = find(IPD2_R_Knee_Angle_y ==
pAngle_IPD2_R_Knee_Angle_y);
 t2p_IPD2_R_Knee_Angle_y = (((pAnlge_idx_IPD2_R_Knee_Angle_y-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Knee_Angle_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Knee_Angle_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Knee_Angle_z_subset =
IPD2_R_Knee_Angle_z(IPD2_R_Knee_Angle_z_Frame1:IPD2_R_Knee_Angle_z_FrameN);
 maxAngle_IPD2_R_Knee_Angle_z = max(IPD2_R_Knee_Angle_z_subset);
 minAngle_IPD2_R_Knee_Angle_z = min(IPD2_R_Knee_Angle_z_subset);
 if maxAngle_IPD2_R_Knee_Angle_z >(abs(minAngle_IPD2_R_Knee_Angle_z))
    pAngle_IPD2_R_Knee_Angle_z = maxAngle_IPD2_R_Knee_Angle_z;
 else
    pAngle_IPD2_R_Knee_Angle_z = minAngle_IPD2_R_Knee_Angle_z;
 end
 pAnlge_idx_IPD2_R_Knee_Angle_z = find(IPD2_R_Knee_Angle_z ==
pAngle_IPD2_R_Knee_Angle_z);
 t2p_IPD2_R_Knee_Angle_z = (((pAnlge_idx_IPD2_R_Knee_Angle_z-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Hip_Angle_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Hip_Angle_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Hip_Angle_x_subset =
IPD2_R_Hip_Angle_x(IPD2_R_Hip_Angle_x_Frame1:IPD2_R_Hip_Angle_x_FrameN);
 maxAngle_IPD2_R_Hip_Angle_x = max(IPD2_R_Hip_Angle_x_subset);
 minAngle_IPD2_R_Hip_Angle_x = min(IPD2_R_Hip_Angle_x_subset);
 if maxAngle_IPD2_R_Hip_Angle_x >(abs(minAngle_IPD2_R_Hip_Angle_x))
    pAngle_IPD2_R_Hip_Angle_x = maxAngle_IPD2_R_Hip_Angle_x;
 else
    pAngle_IPD2_R_Hip_Angle_x = minAngle_IPD2_R_Hip_Angle_x;
 end
```

```
 pAnlge_idx_IPD2_R_Hip_Angle_x = find(IPD2_R_Hip_Angle_x ==
pAngle_IPD2_R_Hip_Angle_x);
 t2p_IPD2_R_Hip_Angle_x = (((pAnlge_idx_IPD2_R_Hip_Angle_x-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Hip_Angle_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Hip_Angle_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Hip_Angle_y_subset =
IPD2_R_Hip_Angle_y(IPD2_R_Hip_Angle_y_Frame1:IPD2_R_Hip_Angle_y_FrameN);
 maxAngle_IPD2_R_Hip_Angle_y = max(IPD2_R_Hip_Angle_y_subset);
 minAngle_IPD2_R_Hip_Angle_y = min(IPD2_R_Hip_Angle_y_subset);
 if maxAngle_IPD2_R_Hip_Angle_y >(abs(minAngle_IPD2_R_Hip_Angle_y))
    pAngle_IPD2_R_Hip_Angle_y = maxAngle_IPD2_R_Hip_Angle_y;
 else
    pAngle_IPD2_R_Hip_Angle_y = minAngle_IPD2_R_Hip_Angle_y;
 end
 pAnlge_idx_IPD2_R_Hip_Angle_y = find(IPD2_R_Hip_Angle_y ==
pAngle_IPD2_R_Hip_Angle_y);
 t2p_IPD2_R_Hip_Angle_y = (((pAnlge_idx_IPD2_R_Hip_Angle_y-1)*0.005)-
IPD2_True_Drop);


 IPD2_R_Hip_Angle_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Hip_Angle_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Hip_Angle_z_subset =
IPD2_R_Hip_Angle_z(IPD2_R_Hip_Angle_z_Frame1:IPD2_R_Hip_Angle_z_FrameN);
 maxAngle_IPD2_R_Hip_Angle_z = max(IPD2_R_Hip_Angle_z_subset);
 minAngle_IPD2_R_Hip_Angle_z = min(IPD2_R_Hip_Angle_z_subset);
 if maxAngle_IPD2_R_Hip_Angle_z >(abs(minAngle_IPD2_R_Hip_Angle_z))
    pAngle_IPD2_R_Hip_Angle_z = maxAngle_IPD2_R_Hip_Angle_z;
 else
    pAngle_IPD2_R_Hip_Angle_z = minAngle_IPD2_R_Hip_Angle_z;
 end
 pAnlge_idx_IPD2_R_Hip_Angle_z = find(IPD2_R_Hip_Angle_z ==
pAngle_IPD2_R_Hip_Angle_z);
 t2p_IPD2_R_Hip_Angle_z = (((pAnlge_idx_IPD2_R_Hip_Angle_z-1)*0.005)-
IPD2_True_Drop);

 %IPD3
 IPD3_R_Ankle_Angle_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Ankle_Angle_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Ankle_Angle_x_subset =
IPD3_R_Ankle_Angle_x(IPD3_R_Ankle_Angle_x_Frame1:IPD3_R_Ankle_Angle_x_FrameN)
;
 maxAngle_IPD3_R_Ankle_Angle_x = max(IPD3_R_Ankle_Angle_x_subset);
```

```
 minAngle_IPD3_R_Ankle_Angle_x = min(IPD3_R_Ankle_Angle_x_subset);
 if maxAngle_IPD3_R_Ankle_Angle_x >(abs(minAngle_IPD3_R_Ankle_Angle_x))
    pAngle_IPD3_R_Ankle_Angle_x = maxAngle_IPD3_R_Ankle_Angle_x;
 else
    pAngle_IPD3_R_Ankle_Angle_x = minAngle_IPD3_R_Ankle_Angle_x;
 end
 pAnlge_idx_IPD3_R_Ankle_Angle_x = find(IPD3_R_Ankle_Angle_x ==
pAngle_IPD3_R_Ankle_Angle_x);
 t2p_IPD3_R_Ankle_Angle_x = (((pAnlge_idx_IPD3_R_Ankle_Angle_x-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Ankle_Angle_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Ankle_Angle_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Ankle_Angle_y_subset =
IPD3_R_Ankle_Angle_y(IPD3_R_Ankle_Angle_y_Frame1:IPD3_R_Ankle_Angle_y_FrameN)
;
 maxAngle_IPD3_R_Ankle_Angle_y = max(IPD3_R_Ankle_Angle_y_subset);
 minAngle_IPD3_R_Ankle_Angle_y = min(IPD3_R_Ankle_Angle_y_subset);
 if maxAngle_IPD3_R_Ankle_Angle_y >(abs(minAngle_IPD3_R_Ankle_Angle_y))
    pAngle_IPD3_R_Ankle_Angle_y = maxAngle_IPD3_R_Ankle_Angle_y;
 else
    pAngle_IPD3_R_Ankle_Angle_y = minAngle_IPD3_R_Ankle_Angle_y;
 end
 pAnlge_idx_IPD3_R_Ankle_Angle_y = find(IPD3_R_Ankle_Angle_y ==
pAngle_IPD3_R_Ankle_Angle_y);
 t2p_IPD3_R_Ankle_Angle_y = (((pAnlge_idx_IPD3_R_Ankle_Angle_y-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Ankle_Angle_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Ankle_Angle_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Ankle_Angle_z_subset =
IPD3_R_Ankle_Angle_z(IPD3_R_Ankle_Angle_z_Frame1:IPD3_R_Ankle_Angle_z_FrameN);
 maxAngle_IPD3_R_Ankle_Angle_z = max(IPD3_R_Ankle_Angle_z_subset);
 minAngle_IPD3_R_Ankle_Angle_z = min(IPD3_R_Ankle_Angle_z_subset);
 if maxAngle_IPD3_R_Ankle_Angle_z >(abs(minAngle_IPD3_R_Ankle_Angle_z))
    pAngle_IPD3_R_Ankle_Angle_z = maxAngle_IPD3_R_Ankle_Angle_z;
 else
    pAngle_IPD3_R_Ankle_Angle_z = minAngle_IPD3_R_Ankle_Angle_z;
 end
 pAnlge_idx_IPD3_R_Ankle_Angle_z = find(IPD3_R_Ankle_Angle_z ==
pAngle_IPD3_R_Ankle_Angle_z);
 t2p_IPD3_R_Ankle_Angle_z = (((pAnlge_idx_IPD3_R_Ankle_Angle_z-1)*0.005)-
IPD3_True_Drop);
```

```
 IPD3_R_Knee_Angle_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Knee_Angle_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Knee_Angle_x_subset =
IPD3_R_Knee_Angle_x(IPD3_R_Knee_Angle_x_Frame1:IPD3_R_Knee_Angle_x_FrameN);
 maxAngle_IPD3_R_Knee_Angle_x = max(IPD3_R_Knee_Angle_x_subset);
 minAngle_IPD3_R_Knee_Angle_x = min(IPD3_R_Knee_Angle_x_subset);
 if maxAngle_IPD3_R_Knee_Angle_x >(abs(minAngle_IPD3_R_Knee_Angle_x))
    pAngle_IPD3_R_Knee_Angle_x = maxAngle_IPD3_R_Knee_Angle_x;
 else
    pAngle_IPD3_R_Knee_Angle_x = minAngle_IPD3_R_Knee_Angle_x;
 end
 pAnlge_idx_IPD3_R_Knee_Angle_x = find(IPD3_R_Knee_Angle_x ==
pAngle_IPD3_R_Knee_Angle_x);
 t2p_IPD3_R_Knee_Angle_x = (((pAnlge_idx_IPD3_R_Knee_Angle_x-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Knee_Angle_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Knee_Angle_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Knee_Angle_y_subset =
IPD3_R_Knee_Angle_y(IPD3_R_Knee_Angle_y_Frame1:IPD3_R_Knee_Angle_y_FrameN);
 maxAngle_IPD3_R_Knee_Angle_y = max(IPD3_R_Knee_Angle_y_subset);
 minAngle_IPD3_R_Knee_Angle_y = min(IPD3_R_Knee_Angle_y_subset);
 if maxAngle_IPD3_R_Knee_Angle_y >(abs(minAngle_IPD3_R_Knee_Angle_y))
    pAngle_IPD3_R_Knee_Angle_y = maxAngle_IPD3_R_Knee_Angle_y;
 else
    pAngle_IPD3_R_Knee_Angle_y = minAngle_IPD3_R_Knee_Angle_y;
 end
 pAnlge_idx_IPD3_R_Knee_Angle_y = find(IPD3_R_Knee_Angle_y ==
pAngle_IPD3_R_Knee_Angle_y);
 t2p_IPD3_R_Knee_Angle_y = (((pAnlge_idx_IPD3_R_Knee_Angle_y-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Knee_Angle_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Knee_Angle_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Knee_Angle_z_subset =
IPD3_R_Knee_Angle_z(IPD3_R_Knee_Angle_z_Frame1:IPD3_R_Knee_Angle_z_FrameN);
 maxAngle_IPD3_R_Knee_Angle_z = max(IPD3_R_Knee_Angle_z_subset);
 minAngle_IPD3_R_Knee_Angle_z = min(IPD3_R_Knee_Angle_z_subset);
 if maxAngle_IPD3_R_Knee_Angle_z >(abs(minAngle_IPD3_R_Knee_Angle_z))
    pAngle_IPD3_R_Knee_Angle_z = maxAngle_IPD3_R_Knee_Angle_z;
 else
    pAngle_IPD3_R_Knee_Angle_z = minAngle_IPD3_R_Knee_Angle_z;
 end
```

```
 pAnlge_idx_IPD3_R_Knee_Angle_z = find(IPD3_R_Knee_Angle_z ==
pAngle_IPD3_R_Knee_Angle_z);
 t2p_IPD3_R_Knee_Angle_z = (((pAnlge_idx_IPD3_R_Knee_Angle_z-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Hip_Angle_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Hip_Angle_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Hip_Angle_x_subset =
IPD3_R_Hip_Angle_x(IPD3_R_Hip_Angle_x_Frame1:IPD3_R_Hip_Angle_x_FrameN);
 maxAngle_IPD3_R_Hip_Angle_x = max(IPD3_R_Hip_Angle_x_subset);
 minAngle_IPD3_R_Hip_Angle_x = min(IPD3_R_Hip_Angle_x_subset);
 if maxAngle_IPD3_R_Hip_Angle_x >(abs(minAngle_IPD3_R_Hip_Angle_x))
    pAngle_IPD3_R_Hip_Angle_x = maxAngle_IPD3_R_Hip_Angle_x;
 else
    pAngle_IPD3_R_Hip_Angle_x = minAngle_IPD3_R_Hip_Angle_x;
 end
 pAnlge_idx_IPD3_R_Hip_Angle_x = find(IPD3_R_Hip_Angle_x ==
pAngle_IPD3_R_Hip_Angle_x);
 t2p_IPD3_R_Hip_Angle_x = (((pAnlge_idx_IPD3_R_Hip_Angle_x-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Hip_Angle_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Hip_Angle_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Hip_Angle_y_subset =
IPD3_R_Hip_Angle_y(IPD3_R_Hip_Angle_y_Frame1:IPD3_R_Hip_Angle_y_FrameN);
 maxAngle_IPD3_R_Hip_Angle_y = max(IPD3_R_Hip_Angle_y_subset);
 minAngle_IPD3_R_Hip_Angle_y = min(IPD3_R_Hip_Angle_y_subset);
 if maxAngle_IPD3_R_Hip_Angle_y >(abs(minAngle_IPD3_R_Hip_Angle_y))
    pAngle_IPD3_R_Hip_Angle_y = maxAngle_IPD3_R_Hip_Angle_y;
 else
    pAngle_IPD3_R_Hip_Angle_y = minAngle_IPD3_R_Hip_Angle_y;
 end
 pAnlge_idx_IPD3_R_Hip_Angle_y = find(IPD3_R_Hip_Angle_y ==
pAngle_IPD3_R_Hip_Angle_y);
 t2p_IPD3_R_Hip_Angle_y = (((pAnlge_idx_IPD3_R_Hip_Angle_y-1)*0.005)-
IPD3_True_Drop);


 IPD3_R_Hip_Angle_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Hip_Angle_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Hip_Angle_z_subset =
IPD3_R_Hip_Angle_z(IPD3_R_Hip_Angle_z_Frame1:IPD3_R_Hip_Angle_z_FrameN);
 maxAngle_IPD3_R_Hip_Angle_z = max(IPD3_R_Hip_Angle_z_subset);
 minAngle_IPD3_R_Hip_Angle_z = min(IPD3_R_Hip_Angle_z_subset);
```

```
 if maxAngle_IPD3_R_Hip_Angle_z >(abs(minAngle_IPD3_R_Hip_Angle_z))
    pAngle_IPD3_R_Hip_Angle_z = maxAngle_IPD3_R_Hip_Angle_z;
 else
    pAngle_IPD3_R_Hip_Angle_z = minAngle_IPD3_R_Hip_Angle_z;
 end
 pAnlge_idx_IPD3_R_Hip_Angle_z = find(IPD3_R_Hip_Angle_z ==
pAngle_IPD3_R_Hip_Angle_z);
 t2p_IPD3_R_Hip_Angle_z = (((pAnlge_idx_IPD3_R_Hip_Angle_z-1)*0.005)-
IPD3_True_Drop);


 %NW1
 NW1_R_Ankle_Angle_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Ankle_Angle_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Ankle_Angle_x_subset =
NW1_R_Ankle_Angle_x(NW1_R_Ankle_Angle_x_Frame1:NW1_R_Ankle_Angle_x_FrameN)
;
 maxAngle_NW1_R_Ankle_Angle_x = max(NW1_R_Ankle_Angle_x_subset);
 minAngle_NW1_R_Ankle_Angle_x = min(NW1_R_Ankle_Angle_x_subset);
 if maxAngle_NW1_R_Ankle_Angle_x >(abs(minAngle_NW1_R_Ankle_Angle_x))
    pAngle_NW1_R_Ankle_Angle_x = maxAngle_NW1_R_Ankle_Angle_x;
 else
    pAngle_NW1_R_Ankle_Angle_x = minAngle_NW1_R_Ankle_Angle_x;
 end
 pAnlge_idx_NW1_R_Ankle_Angle_x = find(NW1_R_Ankle_Angle_x ==
pAngle_NW1_R_Ankle_Angle_x);
 t2p_NW1_R_Ankle_Angle_x = (((pAnlge_idx_NW1_R_Ankle_Angle_x-1)*0.005)-NW1_HS);


 NW1_R_Ankle_Angle_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Ankle_Angle_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Ankle_Angle_y_subset =
NW1_R_Ankle_Angle_y(NW1_R_Ankle_Angle_y_Frame1:NW1_R_Ankle_Angle_y_FrameN)
;
 maxAngle_NW1_R_Ankle_Angle_y = max(NW1_R_Ankle_Angle_y_subset);
 minAngle_NW1_R_Ankle_Angle_y = min(NW1_R_Ankle_Angle_y_subset);
 if maxAngle_NW1_R_Ankle_Angle_y >(abs(minAngle_NW1_R_Ankle_Angle_y))
    pAngle_NW1_R_Ankle_Angle_y = maxAngle_NW1_R_Ankle_Angle_y;
 else
    pAngle_NW1_R_Ankle_Angle_y = minAngle_NW1_R_Ankle_Angle_y;
 end
 pAnlge_idx_NW1_R_Ankle_Angle_y = find(NW1_R_Ankle_Angle_y ==
pAngle_NW1_R_Ankle_Angle_y);
 t2p_NW1_R_Ankle_Angle_y = (((pAnlge_idx_NW1_R_Ankle_Angle_y-1)*0.005)-NW1_HS);
```

```
 NW1_R_Ankle_Angle_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Ankle_Angle_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Ankle_Angle_z_subset =
NW1_R_Ankle_Angle_z(NW1_R_Ankle_Angle_z_Frame1:NW1_R_Ankle_Angle_z_FrameN)
;
 maxAngle_NW1_R_Ankle_Angle_z = max(NW1_R_Ankle_Angle_z_subset);
 minAngle_NW1_R_Ankle_Angle_z = min(NW1_R_Ankle_Angle_z_subset);
 if maxAngle_NW1_R_Ankle_Angle_z >(abs(minAngle_NW1_R_Ankle_Angle_z))
    pAngle_NW1_R_Ankle_Angle_z = maxAngle_NW1_R_Ankle_Angle_z;
 else
    pAngle_NW1_R_Ankle_Angle_z = minAngle_NW1_R_Ankle_Angle_z;
 end
 pAnlge_idx_NW1_R_Ankle_Angle_z = find(NW1_R_Ankle_Angle_z ==
pAngle_NW1_R_Ankle_Angle_z);
 t2p_NW1_R_Ankle_Angle_z = (((pAnlge_idx_NW1_R_Ankle_Angle_z-1)*0.005)-NW1_HS);


 NW1_R_Knee_Angle_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Knee_Angle_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Knee_Angle_x_subset =
NW1_R_Knee_Angle_x(NW1_R_Knee_Angle_x_Frame1:NW1_R_Knee_Angle_x_FrameN);
 maxAngle_NW1_R_Knee_Angle_x = max(NW1_R_Knee_Angle_x_subset);
 minAngle_NW1_R_Knee_Angle_x = min(NW1_R_Knee_Angle_x_subset);
 if maxAngle_NW1_R_Knee_Angle_x >(abs(minAngle_NW1_R_Knee_Angle_x))
    pAngle_NW1_R_Knee_Angle_x = maxAngle_NW1_R_Knee_Angle_x;
 else
    pAngle_NW1_R_Knee_Angle_x = minAngle_NW1_R_Knee_Angle_x;
 end
 pAnlge_idx_NW1_R_Knee_Angle_x = find(NW1_R_Knee_Angle_x ==
pAngle_NW1_R_Knee_Angle_x);
 t2p_NW1_R_Knee_Angle_x = (((pAnlge_idx_NW1_R_Knee_Angle_x-1)*0.005)-NW1_HS);


 NW1_R_Knee_Angle_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Knee_Angle_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Knee_Angle_y_subset =
NW1_R_Knee_Angle_y(NW1_R_Knee_Angle_y_Frame1:NW1_R_Knee_Angle_y_FrameN);
 maxAngle_NW1_R_Knee_Angle_y = max(NW1_R_Knee_Angle_y_subset);
 minAngle_NW1_R_Knee_Angle_y = min(NW1_R_Knee_Angle_y_subset);
 if maxAngle_NW1_R_Knee_Angle_y >(abs(minAngle_NW1_R_Knee_Angle_y))
    pAngle_NW1_R_Knee_Angle_y = maxAngle_NW1_R_Knee_Angle_y;
 else
    pAngle_NW1_R_Knee_Angle_y = minAngle_NW1_R_Knee_Angle_y;
 end
 pAnlge_idx_NW1_R_Knee_Angle_y = find(NW1_R_Knee_Angle_y ==
pAngle_NW1_R_Knee_Angle_y);
```

```
t2p_NW1_R_Knee_Angle_y = (((pAnlge_idx_NW1_R_Knee_Angle_y-1)*0.005)-NW1_HS);


NW1_R_Knee_Angle_z_Frame1 = round((NW1_HS/.005)+1);
NW1_R_Knee_Angle_z_FrameN = round((NW1_Post_350/.005)+1);
NW1_R_Knee_Angle_z_subset =
NW1_R_Knee_Angle_z(NW1_R_Knee_Angle_z_Frame1:NW1_R_Knee_Angle_z_FrameN);
maxAngle_NW1_R_Knee_Angle_z = max(NW1_R_Knee_Angle_z_subset);
minAngle_NW1_R_Knee_Angle_z = min(NW1_R_Knee_Angle_z_subset);
if maxAngle_NW1_R_Knee_Angle_z >(abs(minAngle_NW1_R_Knee_Angle_z))
   pAngle_NW1_R_Knee_Angle_z = maxAngle_NW1_R_Knee_Angle_z;
else
   pAngle_NW1_R_Knee_Angle_z = minAngle_NW1_R_Knee_Angle_z;
end
pAnlge_idx_NW1_R_Knee_Angle_z = find(NW1_R_Knee_Angle_z ==
pAngle_NW1_R_Knee_Angle_z);
t2p_NW1_R_Knee_Angle_z = (((pAnlge_idx_NW1_R_Knee_Angle_z-1)*0.005)-NW1_HS);


NW1_R_Hip_Angle_x_Frame1 = round((NW1_HS/.005)+1);
NW1_R_Hip_Angle_x_FrameN = round((NW1_Post_350/.005)+1);
NW1_R_Hip_Angle_x_subset =
NW1_R_Hip_Angle_x(NW1_R_Hip_Angle_x_Frame1:NW1_R_Hip_Angle_x_FrameN);
maxAngle_NW1_R_Hip_Angle_x = max(NW1_R_Hip_Angle_x_subset);
minAngle_NW1_R_Hip_Angle_x = min(NW1_R_Hip_Angle_x_subset);
if maxAngle_NW1_R_Hip_Angle_x >(abs(minAngle_NW1_R_Hip_Angle_x))
   pAngle_NW1_R_Hip_Angle_x = maxAngle_NW1_R_Hip_Angle_x;
else
   pAngle_NW1_R_Hip_Angle_x = minAngle_NW1_R_Hip_Angle_x;
end
pAnlge_idx_NW1_R_Hip_Angle_x = find(NW1_R_Hip_Angle_x ==
pAngle_NW1_R_Hip_Angle_x);
t2p_NW1_R_Hip_Angle_x = (((pAnlge_idx_NW1_R_Hip_Angle_x-1)*0.005)-NW1_HS);


NW1_R_Hip_Angle_y_Frame1 = round((NW1_HS/.005)+1);
NW1_R_Hip_Angle_y_FrameN = round((NW1_Post_350/.005)+1);
NW1_R_Hip_Angle_y_subset =
NW1_R_Hip_Angle_y(NW1_R_Hip_Angle_y_Frame1:NW1_R_Hip_Angle_y_FrameN);
maxAngle_NW1_R_Hip_Angle_y = max(NW1_R_Hip_Angle_y_subset);
minAngle_NW1_R_Hip_Angle_y = min(NW1_R_Hip_Angle_y_subset);
if maxAngle_NW1_R_Hip_Angle_y >(abs(minAngle_NW1_R_Hip_Angle_y))
   pAngle_NW1_R_Hip_Angle_y = maxAngle_NW1_R_Hip_Angle_y;
else
   pAngle_NW1_R_Hip_Angle_y = minAngle_NW1_R_Hip_Angle_y;
end
```

```
 pAnlge_idx_NW1_R_Hip_Angle_y = find(NW1_R_Hip_Angle_y ==
pAngle_NW1_R_Hip_Angle_y);
 t2p_NW1_R_Hip_Angle_y = (((pAnlge_idx_NW1_R_Hip_Angle_y-1)*0.005)-NW1_HS);


 NW1_R_Hip_Angle_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Hip_Angle_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Hip_Angle_z_subset =
NW1_R_Hip_Angle_z(NW1_R_Hip_Angle_z_Frame1:NW1_R_Hip_Angle_z_FrameN);
 maxAngle_NW1_R_Hip_Angle_z = max(NW1_R_Hip_Angle_z_subset);
 minAngle_NW1_R_Hip_Angle_z = min(NW1_R_Hip_Angle_z_subset);
 if maxAngle_NW1_R_Hip_Angle_z >(abs(minAngle_NW1_R_Hip_Angle_z))
    pAngle_NW1_R_Hip_Angle_z = maxAngle_NW1_R_Hip_Angle_z;
 else
    pAngle_NW1_R_Hip_Angle_z = minAngle_NW1_R_Hip_Angle_z;
 end
 pAnlge_idx_NW1_R_Hip_Angle_z = find(NW1_R_Hip_Angle_z ==
pAngle_NW1_R_Hip_Angle_z);
 t2p_NW1_R_Hip_Angle_z = (((pAnlge_idx_NW1_R_Hip_Angle_z-1)*0.005)-NW1_HS);



 %NW2
 NW2_R_Ankle_Angle_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Ankle_Angle_x_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Ankle_Angle_x_subset =
NW2_R_Ankle_Angle_x(NW2_R_Ankle_Angle_x_Frame1:NW2_R_Ankle_Angle_x_FrameN)
;
 maxAngle_NW2_R_Ankle_Angle_x = max(NW2_R_Ankle_Angle_x_subset);
 minAngle_NW2_R_Ankle_Angle_x = min(NW2_R_Ankle_Angle_x_subset);
 if maxAngle_NW2_R_Ankle_Angle_x >(abs(minAngle_NW2_R_Ankle_Angle_x))
    pAngle_NW2_R_Ankle_Angle_x = maxAngle_NW2_R_Ankle_Angle_x;
 else
    pAngle_NW2_R_Ankle_Angle_x = minAngle_NW2_R_Ankle_Angle_x;
 end
 pAnlge_idx_NW2_R_Ankle_Angle_x = find(NW2_R_Ankle_Angle_x ==
pAngle_NW2_R_Ankle_Angle_x);
 t2p_NW2_R_Ankle_Angle_x = (((pAnlge_idx_NW2_R_Ankle_Angle_x-1)*0.005)-NW2_HS);


 NW2_R_Ankle_Angle_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Ankle_Angle_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Ankle_Angle_y_subset =
NW2_R_Ankle_Angle_y(NW2_R_Ankle_Angle_y_Frame1:NW2_R_Ankle_Angle_y_FrameN)
;
 maxAngle_NW2_R_Ankle_Angle_y = max(NW2_R_Ankle_Angle_y_subset);
```

```
minAngle_NW2_R_Ankle_Angle_y = min(NW2_R_Ankle_Angle_y_subset);
if maxAngle_NW2_R_Ankle_Angle_y >(abs(minAngle_NW2_R_Ankle_Angle_y))
   pAngle_NW2_R_Ankle_Angle_y = maxAngle_NW2_R_Ankle_Angle_y;
else
   pAngle_NW2_R_Ankle_Angle_y = minAngle_NW2_R_Ankle_Angle_y;
end
pAnlge_idx_NW2_R_Ankle_Angle_y = find(NW2_R_Ankle_Angle_y ==
pAngle_NW2_R_Ankle_Angle_y);
t2p_NW2_R_Ankle_Angle_y = (((pAnlge_idx_NW2_R_Ankle_Angle_y-1)*0.005)-NW2_HS);


NW2_R_Ankle_Angle_z_Frame1 = round((NW2_HS/.005)+1);
NW2_R_Ankle_Angle_z_FrameN = round((NW2_Post_350/.005)+1);
NW2_R_Ankle_Angle_z_subset =
NW2_R_Ankle_Angle_z(NW2_R_Ankle_Angle_z_Frame1:NW2_R_Ankle_Angle_z_FrameN)
;
maxAngle_NW2_R_Ankle_Angle_z = max(NW2_R_Ankle_Angle_z_subset);
minAngle_NW2_R_Ankle_Angle_z = min(NW2_R_Ankle_Angle_z_subset);
if maxAngle_NW2_R_Ankle_Angle_z >(abs(minAngle_NW2_R_Ankle_Angle_z))
   pAngle_NW2_R_Ankle_Angle_z = maxAngle_NW2_R_Ankle_Angle_z;
else
   pAngle_NW2_R_Ankle_Angle_z = minAngle_NW2_R_Ankle_Angle_z;
end
pAnlge_idx_NW2_R_Ankle_Angle_z = find(NW2_R_Ankle_Angle_z ==
pAngle_NW2_R_Ankle_Angle_z);
t2p_NW2_R_Ankle_Angle_z = (((pAnlge_idx_NW2_R_Ankle_Angle_z-1)*0.005)-NW2_HS);


NW2_R_Knee_Angle_x_Frame1 = round((NW2_HS/.005)+1);
NW2_R_Knee_Angle_x_FrameN = round((NW2_Post_350/.005)+1);
NW2_R_Knee_Angle_x_subset =
NW2_R_Knee_Angle_x(NW2_R_Knee_Angle_x_Frame1:NW2_R_Knee_Angle_x_FrameN);
maxAngle_NW2_R_Knee_Angle_x = max(NW2_R_Knee_Angle_x_subset);
minAngle_NW2_R_Knee_Angle_x = min(NW2_R_Knee_Angle_x_subset);
if maxAngle_NW2_R_Knee_Angle_x >(abs(minAngle_NW2_R_Knee_Angle_x))
   pAngle_NW2_R_Knee_Angle_x = maxAngle_NW2_R_Knee_Angle_x;
else
   pAngle_NW2_R_Knee_Angle_x = minAngle_NW2_R_Knee_Angle_x;
end
pAnlge_idx_NW2_R_Knee_Angle_x = find(NW2_R_Knee_Angle_x ==
pAngle_NW2_R_Knee_Angle_x);
t2p_NW2_R_Knee_Angle_x = (((pAnlge_idx_NW2_R_Knee_Angle_x-1)*0.005)-NW2_HS);


NW2_R_Knee_Angle_y_Frame1 = round((NW2_HS/.005)+1);
NW2_R_Knee_Angle_y_FrameN = round((NW2_Post_350/.005)+1);
```

```
 NW2_R_Knee_Angle_y_subset =
NW2_R_Knee_Angle_y(NW2_R_Knee_Angle_y_Frame1:NW2_R_Knee_Angle_y_FrameN);
 maxAngle_NW2_R_Knee_Angle_y = max(NW2_R_Knee_Angle_y_subset);
 minAngle_NW2_R_Knee_Angle_y = min(NW2_R_Knee_Angle_y_subset);
 if maxAngle_NW2_R_Knee_Angle_y >(abs(minAngle_NW2_R_Knee_Angle_y))
    pAngle_NW2_R_Knee_Angle_y = maxAngle_NW2_R_Knee_Angle_y;
 else
    pAngle_NW2_R_Knee_Angle_y = minAngle_NW2_R_Knee_Angle_y;
 end
 pAnlge_idx_NW2_R_Knee_Angle_y = find(NW2_R_Knee_Angle_y ==
pAngle_NW2_R_Knee_Angle_y);
 t2p_NW2_R_Knee_Angle_y = (((pAnlge_idx_NW2_R_Knee_Angle_y-1)*0.005)-NW2_HS);


 NW2_R_Knee_Angle_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Knee_Angle_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Knee_Angle_z_subset =
NW2_R_Knee_Angle_z(NW2_R_Knee_Angle_z_Frame1:NW2_R_Knee_Angle_z_FrameN);
 maxAngle_NW2_R_Knee_Angle_z = max(NW2_R_Knee_Angle_z_subset);
 minAngle_NW2_R_Knee_Angle_z = min(NW2_R_Knee_Angle_z_subset);
 if maxAngle_NW2_R_Knee_Angle_z >(abs(minAngle_NW2_R_Knee_Angle_z))
    pAngle_NW2_R_Knee_Angle_z = maxAngle_NW2_R_Knee_Angle_z;
 else
    pAngle_NW2_R_Knee_Angle_z = minAngle_NW2_R_Knee_Angle_z;
 end
 pAnlge_idx_NW2_R_Knee_Angle_z = find(NW2_R_Knee_Angle_z ==
pAngle_NW2_R_Knee_Angle_z);
 t2p_NW2_R_Knee_Angle_z = (((pAnlge_idx_NW2_R_Knee_Angle_z-1)*0.005)-NW2_HS);


 NW2_R_Hip_Angle_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Hip_Angle_x_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Hip_Angle_x_subset =
NW2_R_Hip_Angle_x(NW2_R_Hip_Angle_x_Frame1:NW2_R_Hip_Angle_x_FrameN);
 maxAngle_NW2_R_Hip_Angle_x = max(NW2_R_Hip_Angle_x_subset);
 minAngle_NW2_R_Hip_Angle_x = min(NW2_R_Hip_Angle_x_subset);
 if maxAngle_NW2_R_Hip_Angle_x >(abs(minAngle_NW2_R_Hip_Angle_x))
    pAngle_NW2_R_Hip_Angle_x = maxAngle_NW2_R_Hip_Angle_x;
 else
    pAngle_NW2_R_Hip_Angle_x = minAngle_NW2_R_Hip_Angle_x;
 end
 pAnlge_idx_NW2_R_Hip_Angle_x = find(NW2_R_Hip_Angle_x ==
pAngle_NW2_R_Hip_Angle_x);
 t2p_NW2_R_Hip_Angle_x = (((pAnlge_idx_NW2_R_Hip_Angle_x-1)*0.005)-NW2_HS);
```

```
 NW2_R_Hip_Angle_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Hip_Angle_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Hip_Angle_y_subset =
NW2_R_Hip_Angle_y(NW2_R_Hip_Angle_y_Frame1:NW2_R_Hip_Angle_y_FrameN);
 maxAngle_NW2_R_Hip_Angle_y = max(NW2_R_Hip_Angle_y_subset);
 minAngle_NW2_R_Hip_Angle_y = min(NW2_R_Hip_Angle_y_subset);
 if maxAngle_NW2_R_Hip_Angle_y >(abs(minAngle_NW2_R_Hip_Angle_y))
    pAngle_NW2_R_Hip_Angle_y = maxAngle_NW2_R_Hip_Angle_y;
 else
    pAngle_NW2_R_Hip_Angle_y = minAngle_NW2_R_Hip_Angle_y;
 end
 pAnlge_idx_NW2_R_Hip_Angle_y = find(NW2_R_Hip_Angle_y ==
pAngle_NW2_R_Hip_Angle_y);
 t2p_NW2_R_Hip_Angle_y = (((pAnlge_idx_NW2_R_Hip_Angle_y-1)*0.005)-NW2_HS);


 NW2_R_Hip_Angle_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Hip_Angle_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Hip_Angle_z_subset =
NW2_R_Hip_Angle_z(NW2_R_Hip_Angle_z_Frame1:NW2_R_Hip_Angle_z_FrameN);
 maxAngle_NW2_R_Hip_Angle_z = max(NW2_R_Hip_Angle_z_subset);
 minAngle_NW2_R_Hip_Angle_z = min(NW2_R_Hip_Angle_z_subset);
 if maxAngle_NW2_R_Hip_Angle_z >(abs(minAngle_NW2_R_Hip_Angle_z))
    pAngle_NW2_R_Hip_Angle_z = maxAngle_NW2_R_Hip_Angle_z;
 else
    pAngle_NW2_R_Hip_Angle_z = minAngle_NW2_R_Hip_Angle_z;
 end
 pAnlge_idx_NW2_R_Hip_Angle_z = find(NW2_R_Hip_Angle_z ==
pAngle_NW2_R_Hip_Angle_z);
 t2p_NW2_R_Hip_Angle_z = (((pAnlge_idx_NW2_R_Hip_Angle_z-1)*0.005)-NW2_HS);



 %NW3
  NW3_R_Ankle_Angle_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Ankle_Angle_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Ankle_Angle_x_subset =
NW3_R_Ankle_Angle_x(NW3_R_Ankle_Angle_x_Frame1:NW3_R_Ankle_Angle_x_FrameN)
;
 maxAngle_NW3_R_Ankle_Angle_x = max(NW3_R_Ankle_Angle_x_subset);
 minAngle_NW3_R_Ankle_Angle_x = min(NW3_R_Ankle_Angle_x_subset);
 if maxAngle_NW3_R_Ankle_Angle_x >(abs(minAngle_NW3_R_Ankle_Angle_x))
    pAngle_NW3_R_Ankle_Angle_x = maxAngle_NW3_R_Ankle_Angle_x;
 else
    pAngle_NW3_R_Ankle_Angle_x = minAngle_NW3_R_Ankle_Angle_x;
 end
```

```
 pAnlge_idx_NW3_R_Ankle_Angle_x = find(NW3_R_Ankle_Angle_x ==
pAngle_NW3_R_Ankle_Angle_x);
 t2p_NW3_R_Ankle_Angle_x = (((pAnlge_idx_NW3_R_Ankle_Angle_x-1)*0.005)-NW3_HS);


 NW3_R_Ankle_Angle_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Ankle_Angle_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Ankle_Angle_y_subset =
NW3_R_Ankle_Angle_y(NW3_R_Ankle_Angle_y_Frame1:NW3_R_Ankle_Angle_y_FrameN)
;
 maxAngle_NW3_R_Ankle_Angle_y = max(NW3_R_Ankle_Angle_y_subset);
 minAngle_NW3_R_Ankle_Angle_y = min(NW3_R_Ankle_Angle_y_subset);
 if maxAngle_NW3_R_Ankle_Angle_y >(abs(minAngle_NW3_R_Ankle_Angle_y))
    pAngle_NW3_R_Ankle_Angle_y = maxAngle_NW3_R_Ankle_Angle_y;
 else
    pAngle_NW3_R_Ankle_Angle_y = minAngle_NW3_R_Ankle_Angle_y;
 end
 pAnlge_idx_NW3_R_Ankle_Angle_y = find(NW3_R_Ankle_Angle_y ==
pAngle_NW3_R_Ankle_Angle_y);
 t2p_NW3_R_Ankle_Angle_y = (((pAnlge_idx_NW3_R_Ankle_Angle_y-1)*0.005)-NW3_HS);


 NW3_R_Ankle_Angle_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Ankle_Angle_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Ankle_Angle_z_subset =
NW3_R_Ankle_Angle_z(NW3_R_Ankle_Angle_z_Frame1:NW3_R_Ankle_Angle_z_FrameN)
;
 maxAngle_NW3_R_Ankle_Angle_z = max(NW3_R_Ankle_Angle_z_subset);
 minAngle_NW3_R_Ankle_Angle_z = min(NW3_R_Ankle_Angle_z_subset);
 if maxAngle_NW3_R_Ankle_Angle_z >(abs(minAngle_NW3_R_Ankle_Angle_z))
    pAngle_NW3_R_Ankle_Angle_z = maxAngle_NW3_R_Ankle_Angle_z;
 else
    pAngle_NW3_R_Ankle_Angle_z = minAngle_NW3_R_Ankle_Angle_z;
 end
 pAnlge_idx_NW3_R_Ankle_Angle_z = find(NW3_R_Ankle_Angle_z ==
pAngle_NW3_R_Ankle_Angle_z);
 t2p_NW3_R_Ankle_Angle_z = (((pAnlge_idx_NW3_R_Ankle_Angle_z-1)*0.005)-NW3_HS);


 NW3_R_Knee_Angle_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Knee_Angle_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Knee_Angle_x_subset =
NW3_R_Knee_Angle_x(NW3_R_Knee_Angle_x_Frame1:NW3_R_Knee_Angle_x_FrameN);
 maxAngle_NW3_R_Knee_Angle_x = max(NW3_R_Knee_Angle_x_subset);
 minAngle_NW3_R_Knee_Angle_x = min(NW3_R_Knee_Angle_x_subset);
 if maxAngle_NW3_R_Knee_Angle_x >(abs(minAngle_NW3_R_Knee_Angle_x))
```

```
    pAngle_NW3_R_Knee_Angle_x = maxAngle_NW3_R_Knee_Angle_x;
 else
    pAngle_NW3_R_Knee_Angle_x = minAngle_NW3_R_Knee_Angle_x;
 end
 pAnlge_idx_NW3_R_Knee_Angle_x = find(NW3_R_Knee_Angle_x ==
pAngle_NW3_R_Knee_Angle_x);
 t2p_NW3_R_Knee_Angle_x = (((pAnlge_idx_NW3_R_Knee_Angle_x-1)*0.005)-NW3_HS);


 NW3_R_Knee_Angle_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Knee_Angle_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Knee_Angle_y_subset =
NW3_R_Knee_Angle_y(NW3_R_Knee_Angle_y_Frame1:NW3_R_Knee_Angle_y_FrameN);
 maxAngle_NW3_R_Knee_Angle_y = max(NW3_R_Knee_Angle_y_subset);
 minAngle_NW3_R_Knee_Angle_y = min(NW3_R_Knee_Angle_y_subset);
 if maxAngle_NW3_R_Knee_Angle_y >(abs(minAngle_NW3_R_Knee_Angle_y))
    pAngle_NW3_R_Knee_Angle_y = maxAngle_NW3_R_Knee_Angle_y;
 else
    pAngle_NW3_R_Knee_Angle_y = minAngle_NW3_R_Knee_Angle_y;
 end
 pAnlge_idx_NW3_R_Knee_Angle_y = find(NW3_R_Knee_Angle_y ==
pAngle_NW3_R_Knee_Angle_y);
 t2p_NW3_R_Knee_Angle_y = (((pAnlge_idx_NW3_R_Knee_Angle_y-1)*0.005)-NW3_HS);


 NW3_R_Knee_Angle_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Knee_Angle_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Knee_Angle_z_subset =
NW3_R_Knee_Angle_z(NW3_R_Knee_Angle_z_Frame1:NW3_R_Knee_Angle_z_FrameN);
 maxAngle_NW3_R_Knee_Angle_z = max(NW3_R_Knee_Angle_z_subset);
 minAngle_NW3_R_Knee_Angle_z = min(NW3_R_Knee_Angle_z_subset);
 if maxAngle_NW3_R_Knee_Angle_z >(abs(minAngle_NW3_R_Knee_Angle_z))
    pAngle_NW3_R_Knee_Angle_z = maxAngle_NW3_R_Knee_Angle_z;
 else
    pAngle_NW3_R_Knee_Angle_z = minAngle_NW3_R_Knee_Angle_z;
 end
 pAnlge_idx_NW3_R_Knee_Angle_z = find(NW3_R_Knee_Angle_z ==
pAngle_NW3_R_Knee_Angle_z);
 t2p_NW3_R_Knee_Angle_z = (((pAnlge_idx_NW3_R_Knee_Angle_z-1)*0.005)-NW3_HS);


 NW3_R_Hip_Angle_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Hip_Angle_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Hip_Angle_x_subset =
NW3_R_Hip_Angle_x(NW3_R_Hip_Angle_x_Frame1:NW3_R_Hip_Angle_x_FrameN);
 maxAngle_NW3_R_Hip_Angle_x = max(NW3_R_Hip_Angle_x_subset);
```

```
minAngle_NW3_R_Hip_Angle_x = min(NW3_R_Hip_Angle_x_subset);
if maxAngle_NW3_R_Hip_Angle_x >(abs(minAngle_NW3_R_Hip_Angle_x))
   pAngle_NW3_R_Hip_Angle_x = maxAngle_NW3_R_Hip_Angle_x;
else
   pAngle_NW3_R_Hip_Angle_x = minAngle_NW3_R_Hip_Angle_x;
end
pAnlge_idx_NW3_R_Hip_Angle_x = find(NW3_R_Hip_Angle_x ==
pAngle_NW3_R_Hip_Angle_x);
t2p_NW3_R_Hip_Angle_x = (((pAnlge_idx_NW3_R_Hip_Angle_x-1)*0.005)-NW3_HS);


NW3_R_Hip_Angle_y_Frame1 = round((NW3_HS/.005)+1);
NW3_R_Hip_Angle_y_FrameN = round((NW3_Post_350/.005)+1);
NW3_R_Hip_Angle_y_subset =
NW3_R_Hip_Angle_y(NW3_R_Hip_Angle_y_Frame1:NW3_R_Hip_Angle_y_FrameN);
maxAngle_NW3_R_Hip_Angle_y = max(NW3_R_Hip_Angle_y_subset);
minAngle_NW3_R_Hip_Angle_y = min(NW3_R_Hip_Angle_y_subset);
if maxAngle_NW3_R_Hip_Angle_y >(abs(minAngle_NW3_R_Hip_Angle_y))
   pAngle_NW3_R_Hip_Angle_y = maxAngle_NW3_R_Hip_Angle_y;
else
   pAngle_NW3_R_Hip_Angle_y = minAngle_NW3_R_Hip_Angle_y;
end
pAnlge_idx_NW3_R_Hip_Angle_y = find(NW3_R_Hip_Angle_y ==
pAngle_NW3_R_Hip_Angle_y);
t2p_NW3_R_Hip_Angle_y = (((pAnlge_idx_NW3_R_Hip_Angle_y-1)*0.005)-NW3_HS);


NW3_R_Hip_Angle_z_Frame1 = round((NW3_HS/.005)+1);
NW3_R_Hip_Angle_z_FrameN = round((NW3_Post_350/.005)+1);
NW3_R_Hip_Angle_z_subset =
NW3_R_Hip_Angle_z(NW3_R_Hip_Angle_z_Frame1:NW3_R_Hip_Angle_z_FrameN);
maxAngle_NW3_R_Hip_Angle_z = max(NW3_R_Hip_Angle_z_subset);
minAngle_NW3_R_Hip_Angle_z = min(NW3_R_Hip_Angle_z_subset);
if maxAngle_NW3_R_Hip_Angle_z >(abs(minAngle_NW3_R_Hip_Angle_z))
   pAngle_NW3_R_Hip_Angle_z = maxAngle_NW3_R_Hip_Angle_z;
else
   pAngle_NW3_R_Hip_Angle_z = minAngle_NW3_R_Hip_Angle_z;
end
pAnlge_idx_NW3_R_Hip_Angle_z = find(NW3_R_Hip_Angle_z ==
pAngle_NW3_R_Hip_Angle_z);
t2p_NW3_R_Hip_Angle_z = (((pAnlge_idx_NW3_R_Hip_Angle_z-1)*0.005)-NW3_HS);


 %Velocity
%Left Side
%ID1
```

```
 ID1_L_Ankle_Velocity_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Ankle_Velocity_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Ankle_Velocity_x_subset =
ID1_L_Ankle_Velocity_x(ID1_L_Ankle_Velocity_x_Frame1:ID1_L_Ankle_Velocity_x_Frame
N);
 maxVelocity_ID1_L_Ankle_Velocity_x = max(ID1_L_Ankle_Velocity_x_subset);
 minVelocity_ID1_L_Ankle_Velocity_x = min(ID1_L_Ankle_Velocity_x_subset);
 if maxVelocity_ID1_L_Ankle_Velocity_x >(abs(minVelocity_ID1_L_Ankle_Velocity_x))
    pVelocity_ID1_L_Ankle_Velocity_x = maxVelocity_ID1_L_Ankle_Velocity_x;
 else
    pVelocity_ID1_L_Ankle_Velocity_x = minVelocity_ID1_L_Ankle_Velocity_x;
 end

 ID1_L_Ankle_Velocity_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Ankle_Velocity_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Ankle_Velocity_y_subset =
ID1_L_Ankle_Velocity_y(ID1_L_Ankle_Velocity_y_Frame1:ID1_L_Ankle_Velocity_y_Frame
N);
 maxVelocity_ID1_L_Ankle_Velocity_y = max(ID1_L_Ankle_Velocity_y_subset);
 minVelocity_ID1_L_Ankle_Velocity_y = min(ID1_L_Ankle_Velocity_y_subset);
 if maxVelocity_ID1_L_Ankle_Velocity_y >(abs(minVelocity_ID1_L_Ankle_Velocity_y))
    pVelocity_ID1_L_Ankle_Velocity_y = maxVelocity_ID1_L_Ankle_Velocity_y;
 else
    pVelocity_ID1_L_Ankle_Velocity_y = minVelocity_ID1_L_Ankle_Velocity_y;
 end


 ID1_L_Ankle_Velocity_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Ankle_Velocity_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Ankle_Velocity_z_subset =
ID1_L_Ankle_Velocity_z(ID1_L_Ankle_Velocity_z_Frame1:ID1_L_Ankle_Velocity_z_Frame
N);
 maxVelocity_ID1_L_Ankle_Velocity_z = max(ID1_L_Ankle_Velocity_z_subset);
 minVelocity_ID1_L_Ankle_Velocity_z = min(ID1_L_Ankle_Velocity_z_subset);
 if maxVelocity_ID1_L_Ankle_Velocity_z >(abs(minVelocity_ID1_L_Ankle_Velocity_z))
    pVelocity_ID1_L_Ankle_Velocity_z = maxVelocity_ID1_L_Ankle_Velocity_z;
 else
    pVelocity_ID1_L_Ankle_Velocity_z = minVelocity_ID1_L_Ankle_Velocity_z;
 end


 ID1_L_Knee_Velocity_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Knee_Velocity_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Knee_Velocity_x_subset =
ID1_L_Knee_Velocity_x(ID1_L_Knee_Velocity_x_Frame1:ID1_L_Knee_Velocity_x_FrameN)
;
```

```
maxVelocity_ID1_L_Knee_Velocity_x = max(ID1_L_Knee_Velocity_x_subset);
minVelocity_ID1_L_Knee_Velocity_x = min(ID1_L_Knee_Velocity_x_subset);
if maxVelocity_ID1_L_Knee_Velocity_x >(abs(minVelocity_ID1_L_Knee_Velocity_x))
   pVelocity_ID1_L_Knee_Velocity_x = maxVelocity_ID1_L_Knee_Velocity_x;
else
   pVelocity_ID1_L_Knee_Velocity_x = minVelocity_ID1_L_Knee_Velocity_x;
end


ID1_L_Knee_Velocity_y_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_L_Knee_Velocity_y_FrameN = round((ID1_Post_350/.005)+1);
ID1_L_Knee_Velocity_y_subset =
ID1_L_Knee_Velocity_y(ID1_L_Knee_Velocity_y_Frame1:ID1_L_Knee_Velocity_y_FrameN)
;
maxVelocity_ID1_L_Knee_Velocity_y = max(ID1_L_Knee_Velocity_y_subset);
minVelocity_ID1_L_Knee_Velocity_y = min(ID1_L_Knee_Velocity_y_subset);
if maxVelocity_ID1_L_Knee_Velocity_y >(abs(minVelocity_ID1_L_Knee_Velocity_y))
   pVelocity_ID1_L_Knee_Velocity_y = maxVelocity_ID1_L_Knee_Velocity_y;
else
   pVelocity_ID1_L_Knee_Velocity_y = minVelocity_ID1_L_Knee_Velocity_y;
end


ID1_L_Knee_Velocity_z_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_L_Knee_Velocity_z_FrameN = round((ID1_Post_350/.005)+1);
ID1_L_Knee_Velocity_z_subset =
ID1_L_Knee_Velocity_z(ID1_L_Knee_Velocity_z_Frame1:ID1_L_Knee_Velocity_z_FrameN)
;
maxVelocity_ID1_L_Knee_Velocity_z = max(ID1_L_Knee_Velocity_z_subset);
minVelocity_ID1_L_Knee_Velocity_z = min(ID1_L_Knee_Velocity_z_subset);
if maxVelocity_ID1_L_Knee_Velocity_z >(abs(minVelocity_ID1_L_Knee_Velocity_z))
   pVelocity_ID1_L_Knee_Velocity_z = maxVelocity_ID1_L_Knee_Velocity_z;
else
   pVelocity_ID1_L_Knee_Velocity_z = minVelocity_ID1_L_Knee_Velocity_z;
end


ID1_L_Hip_Velocity_x_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_L_Hip_Velocity_x_FrameN = round((ID1_Post_350/.005)+1);
ID1_L_Hip_Velocity_x_subset =
ID1_L_Hip_Velocity_x(ID1_L_Hip_Velocity_x_Frame1:ID1_L_Hip_Velocity_x_FrameN);
maxVelocity_ID1_L_Hip_Velocity_x = max(ID1_L_Hip_Velocity_x_subset);
minVelocity_ID1_L_Hip_Velocity_x = min(ID1_L_Hip_Velocity_x_subset);
if maxVelocity_ID1_L_Hip_Velocity_x >(abs(minVelocity_ID1_L_Hip_Velocity_x))
   pVelocity_ID1_L_Hip_Velocity_x = maxVelocity_ID1_L_Hip_Velocity_x;
else
```

```
    pVelocity_ID1_L_Hip_Velocity_x = minVelocity_ID1_L_Hip_Velocity_x;
 end


 ID1_L_Hip_Velocity_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Hip_Velocity_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Hip_Velocity_y_subset =
ID1_L_Hip_Velocity_y(ID1_L_Hip_Velocity_y_Frame1:ID1_L_Hip_Velocity_y_FrameN);
 maxVelocity_ID1_L_Hip_Velocity_y = max(ID1_L_Hip_Velocity_y_subset);
 minVelocity_ID1_L_Hip_Velocity_y = min(ID1_L_Hip_Velocity_y_subset);
 if maxVelocity_ID1_L_Hip_Velocity_y >(abs(minVelocity_ID1_L_Hip_Velocity_y))
    pVelocity_ID1_L_Hip_Velocity_y = maxVelocity_ID1_L_Hip_Velocity_y;
 else
    pVelocity_ID1_L_Hip_Velocity_y = minVelocity_ID1_L_Hip_Velocity_y;
 end


 ID1_L_Hip_Velocity_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_L_Hip_Velocity_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_L_Hip_Velocity_z_subset =
ID1_L_Hip_Velocity_z(ID1_L_Hip_Velocity_z_Frame1:ID1_L_Hip_Velocity_z_FrameN);
 maxVelocity_ID1_L_Hip_Velocity_z = max(ID1_L_Hip_Velocity_z_subset);
 minVelocity_ID1_L_Hip_Velocity_z = min(ID1_L_Hip_Velocity_z_subset);
 if maxVelocity_ID1_L_Hip_Velocity_z >(abs(minVelocity_ID1_L_Hip_Velocity_z))
    pVelocity_ID1_L_Hip_Velocity_z = maxVelocity_ID1_L_Hip_Velocity_z;
 else
    pVelocity_ID1_L_Hip_Velocity_z = minVelocity_ID1_L_Hip_Velocity_z;
 end


 % ID2
 ID2_L_Ankle_Velocity_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Ankle_Velocity_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Ankle_Velocity_x_subset =
ID2_L_Ankle_Velocity_x(ID2_L_Ankle_Velocity_x_Frame1:ID2_L_Ankle_Velocity_x_Frame
N);
 maxVelocity_ID2_L_Ankle_Velocity_x = max(ID2_L_Ankle_Velocity_x_subset);
 minVelocity_ID2_L_Ankle_Velocity_x = min(ID2_L_Ankle_Velocity_x_subset);
 if maxVelocity_ID2_L_Ankle_Velocity_x >(abs(minVelocity_ID2_L_Ankle_Velocity_x))
    pVelocity_ID2_L_Ankle_Velocity_x = maxVelocity_ID2_L_Ankle_Velocity_x;
 else
    pVelocity_ID2_L_Ankle_Velocity_x = minVelocity_ID2_L_Ankle_Velocity_x;
 end


 ID2_L_Ankle_Velocity_y_Frame1 = round((ID2_True_Drop/.005)+1);
```

```
 ID2_L_Ankle_Velocity_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Ankle_Velocity_y_subset =
ID2_L_Ankle_Velocity_y(ID2_L_Ankle_Velocity_y_Frame1:ID2_L_Ankle_Velocity_y_Frame
N);
 maxVelocity_ID2_L_Ankle_Velocity_y = max(ID2_L_Ankle_Velocity_y_subset);
 minVelocity_ID2_L_Ankle_Velocity_y = min(ID2_L_Ankle_Velocity_y_subset);
 if maxVelocity_ID2_L_Ankle_Velocity_y >(abs(minVelocity_ID2_L_Ankle_Velocity_y))
    pVelocity_ID2_L_Ankle_Velocity_y = maxVelocity_ID2_L_Ankle_Velocity_y;
 else
    pVelocity_ID2_L_Ankle_Velocity_y = minVelocity_ID2_L_Ankle_Velocity_y;
 end


 ID2_L_Ankle_Velocity_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Ankle_Velocity_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Ankle_Velocity_z_subset =
ID2_L_Ankle_Velocity_z(ID2_L_Ankle_Velocity_z_Frame1:ID2_L_Ankle_Velocity_z_Frame
N);
 maxVelocity_ID2_L_Ankle_Velocity_z = max(ID2_L_Ankle_Velocity_z_subset);
 minVelocity_ID2_L_Ankle_Velocity_z = min(ID2_L_Ankle_Velocity_z_subset);
 if maxVelocity_ID2_L_Ankle_Velocity_z >(abs(minVelocity_ID2_L_Ankle_Velocity_z))
    pVelocity_ID2_L_Ankle_Velocity_z = maxVelocity_ID2_L_Ankle_Velocity_z;
 else
    pVelocity_ID2_L_Ankle_Velocity_z = minVelocity_ID2_L_Ankle_Velocity_z;
 end


 ID2_L_Knee_Velocity_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Knee_Velocity_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Knee_Velocity_x_subset =
ID2_L_Knee_Velocity_x(ID2_L_Knee_Velocity_x_Frame1:ID2_L_Knee_Velocity_x_FrameN)
;
 maxVelocity_ID2_L_Knee_Velocity_x = max(ID2_L_Knee_Velocity_x_subset);
 minVelocity_ID2_L_Knee_Velocity_x = min(ID2_L_Knee_Velocity_x_subset);
 if maxVelocity_ID2_L_Knee_Velocity_x >(abs(minVelocity_ID2_L_Knee_Velocity_x))
    pVelocity_ID2_L_Knee_Velocity_x = maxVelocity_ID2_L_Knee_Velocity_x;
 else
    pVelocity_ID2_L_Knee_Velocity_x = minVelocity_ID2_L_Knee_Velocity_x;
 end

 ID2_L_Knee_Velocity_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_L_Knee_Velocity_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_L_Knee_Velocity_y_subset =
ID2_L_Knee_Velocity_y(ID2_L_Knee_Velocity_y_Frame1:ID2_L_Knee_Velocity_y_FrameN)
;
 maxVelocity_ID2_L_Knee_Velocity_y = max(ID2_L_Knee_Velocity_y_subset);
```

```
minVelocity_ID2_L_Knee_Velocity_y = min(ID2_L_Knee_Velocity_y_subset);
if maxVelocity_ID2_L_Knee_Velocity_y >(abs(minVelocity_ID2_L_Knee_Velocity_y))
   pVelocity_ID2_L_Knee_Velocity_y = maxVelocity_ID2_L_Knee_Velocity_y;
else
   pVelocity_ID2_L_Knee_Velocity_y = minVelocity_ID2_L_Knee_Velocity_y;
end


ID2_L_Knee_Velocity_z_Frame1 = round((ID2_True_Drop/.005)+1);
ID2_L_Knee_Velocity_z_FrameN = round((ID2_Post_350/.005)+1);
ID2_L_Knee_Velocity_z_subset =
ID2_L_Knee_Velocity_z(ID2_L_Knee_Velocity_z_Frame1:ID2_L_Knee_Velocity_z_FrameN)
;
maxVelocity_ID2_L_Knee_Velocity_z = max(ID2_L_Knee_Velocity_z_subset);
minVelocity_ID2_L_Knee_Velocity_z = min(ID2_L_Knee_Velocity_z_subset);
if maxVelocity_ID2_L_Knee_Velocity_z >(abs(minVelocity_ID2_L_Knee_Velocity_z))
   pVelocity_ID2_L_Knee_Velocity_z = maxVelocity_ID2_L_Knee_Velocity_z;
else
   pVelocity_ID2_L_Knee_Velocity_z = minVelocity_ID2_L_Knee_Velocity_z;
end


ID2_L_Hip_Velocity_x_Frame1 = round((ID2_True_Drop/.005)+1);
ID2_L_Hip_Velocity_x_FrameN = round((ID2_Post_350/.005)+1);
ID2_L_Hip_Velocity_x_subset =
ID2_L_Hip_Velocity_x(ID2_L_Hip_Velocity_x_Frame1:ID2_L_Hip_Velocity_x_FrameN);
maxVelocity_ID2_L_Hip_Velocity_x = max(ID2_L_Hip_Velocity_x_subset);
minVelocity_ID2_L_Hip_Velocity_x = min(ID2_L_Hip_Velocity_x_subset);
if maxVelocity_ID2_L_Hip_Velocity_x >(abs(minVelocity_ID2_L_Hip_Velocity_x))
   pVelocity_ID2_L_Hip_Velocity_x = maxVelocity_ID2_L_Hip_Velocity_x;
else
   pVelocity_ID2_L_Hip_Velocity_x = minVelocity_ID2_L_Hip_Velocity_x;
end


ID2_L_Hip_Velocity_y_Frame1 = round((ID2_True_Drop/.005)+1);
ID2_L_Hip_Velocity_y_FrameN = round((ID2_Post_350/.005)+1);
ID2_L_Hip_Velocity_y_subset =
ID2_L_Hip_Velocity_y(ID2_L_Hip_Velocity_y_Frame1:ID2_L_Hip_Velocity_y_FrameN);
maxVelocity_ID2_L_Hip_Velocity_y = max(ID2_L_Hip_Velocity_y_subset);
minVelocity_ID2_L_Hip_Velocity_y = min(ID2_L_Hip_Velocity_y_subset);
if maxVelocity_ID2_L_Hip_Velocity_y >(abs(minVelocity_ID2_L_Hip_Velocity_y))
   pVelocity_ID2_L_Hip_Velocity_y = maxVelocity_ID2_L_Hip_Velocity_y;
else
   pVelocity_ID2_L_Hip_Velocity_y = minVelocity_ID2_L_Hip_Velocity_y;
end
```

```
ID2_L_Hip_Velocity_z_Frame1 = round((ID2_True_Drop/.005)+1);
ID2_L_Hip_Velocity_z_FrameN = round((ID2_Post_350/.005)+1);
ID2_L_Hip_Velocity_z_subset =
ID2_L_Hip_Velocity_z(ID2_L_Hip_Velocity_z_Frame1:ID2_L_Hip_Velocity_z_FrameN);
maxVelocity_ID2_L_Hip_Velocity_z = max(ID2_L_Hip_Velocity_z_subset);
minVelocity_ID2_L_Hip_Velocity_z = min(ID2_L_Hip_Velocity_z_subset);
if maxVelocity_ID2_L_Hip_Velocity_z >(abs(minVelocity_ID2_L_Hip_Velocity_z))
   pVelocity_ID2_L_Hip_Velocity_z = maxVelocity_ID2_L_Hip_Velocity_z;
else
   pVelocity_ID2_L_Hip_Velocity_z = minVelocity_ID2_L_Hip_Velocity_z;
end


%ID3
ID3_L_Ankle_Velocity_x_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_L_Ankle_Velocity_x_FrameN = round((ID3_Post_350/.005)+1);
ID3_L_Ankle_Velocity_x_subset =
ID3_L_Ankle_Velocity_x(ID3_L_Ankle_Velocity_x_Frame1:ID3_L_Ankle_Velocity_x_Frame
N);
maxVelocity_ID3_L_Ankle_Velocity_x = max(ID3_L_Ankle_Velocity_x_subset);
minVelocity_ID3_L_Ankle_Velocity_x = min(ID3_L_Ankle_Velocity_x_subset);
if maxVelocity_ID3_L_Ankle_Velocity_x >(abs(minVelocity_ID3_L_Ankle_Velocity_x))
   pVelocity_ID3_L_Ankle_Velocity_x = maxVelocity_ID3_L_Ankle_Velocity_x;
else
   pVelocity_ID3_L_Ankle_Velocity_x = minVelocity_ID3_L_Ankle_Velocity_x;
end


ID3_L_Ankle_Velocity_y_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_L_Ankle_Velocity_y_FrameN = round((ID3_Post_350/.005)+1);
ID3_L_Ankle_Velocity_y_subset =
ID3_L_Ankle_Velocity_y(ID3_L_Ankle_Velocity_y_Frame1:ID3_L_Ankle_Velocity_y_Frame
N);
maxVelocity_ID3_L_Ankle_Velocity_y = max(ID3_L_Ankle_Velocity_y_subset);
minVelocity_ID3_L_Ankle_Velocity_y = min(ID3_L_Ankle_Velocity_y_subset);
if maxVelocity_ID3_L_Ankle_Velocity_y >(abs(minVelocity_ID3_L_Ankle_Velocity_y))
   pVelocity_ID3_L_Ankle_Velocity_y = maxVelocity_ID3_L_Ankle_Velocity_y;
else
   pVelocity_ID3_L_Ankle_Velocity_y = minVelocity_ID3_L_Ankle_Velocity_y;
end


ID3_L_Ankle_Velocity_z_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_L_Ankle_Velocity_z_FrameN = round((ID3_Post_350/.005)+1);
```

```
 ID3_L_Ankle_Velocity_z_subset =
ID3_L_Ankle_Velocity_z(ID3_L_Ankle_Velocity_z_Frame1:ID3_L_Ankle_Velocity_z_Frame
N);
 maxVelocity_ID3_L_Ankle_Velocity_z = max(ID3_L_Ankle_Velocity_z_subset);
 minVelocity_ID3_L_Ankle_Velocity_z = min(ID3_L_Ankle_Velocity_z_subset);
 if maxVelocity_ID3_L_Ankle_Velocity_z >(abs(minVelocity_ID3_L_Ankle_Velocity_z))
    pVelocity_ID3_L_Ankle_Velocity_z = maxVelocity_ID3_L_Ankle_Velocity_z;
 else
    pVelocity_ID3_L_Ankle_Velocity_z = minVelocity_ID3_L_Ankle_Velocity_z;
 end


 ID3_L_Knee_Velocity_x_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Knee_Velocity_x_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Knee_Velocity_x_subset =
ID3_L_Knee_Velocity_x(ID3_L_Knee_Velocity_x_Frame1:ID3_L_Knee_Velocity_x_FrameN)
;
 maxVelocity_ID3_L_Knee_Velocity_x = max(ID3_L_Knee_Velocity_x_subset);
 minVelocity_ID3_L_Knee_Velocity_x = min(ID3_L_Knee_Velocity_x_subset);
 if maxVelocity_ID3_L_Knee_Velocity_x >(abs(minVelocity_ID3_L_Knee_Velocity_x))
    pVelocity_ID3_L_Knee_Velocity_x = maxVelocity_ID3_L_Knee_Velocity_x;
 else
    pVelocity_ID3_L_Knee_Velocity_x = minVelocity_ID3_L_Knee_Velocity_x;
 end


 ID3_L_Knee_Velocity_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Knee_Velocity_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Knee_Velocity_y_subset =
ID3_L_Knee_Velocity_y(ID3_L_Knee_Velocity_y_Frame1:ID3_L_Knee_Velocity_y_FrameN)
;
 maxVelocity_ID3_L_Knee_Velocity_y = max(ID3_L_Knee_Velocity_y_subset);
 minVelocity_ID3_L_Knee_Velocity_y = min(ID3_L_Knee_Velocity_y_subset);
 if maxVelocity_ID3_L_Knee_Velocity_y >(abs(minVelocity_ID3_L_Knee_Velocity_y))
    pVelocity_ID3_L_Knee_Velocity_y = maxVelocity_ID3_L_Knee_Velocity_y;
 else
    pVelocity_ID3_L_Knee_Velocity_y = minVelocity_ID3_L_Knee_Velocity_y;
 end


 ID3_L_Knee_Velocity_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_L_Knee_Velocity_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_L_Knee_Velocity_z_subset =
ID3_L_Knee_Velocity_z(ID3_L_Knee_Velocity_z_Frame1:ID3_L_Knee_Velocity_z_FrameN)
;
 maxVelocity_ID3_L_Knee_Velocity_z = max(ID3_L_Knee_Velocity_z_subset);
```

```
minVelocity_ID3_L_Knee_Velocity_z = min(ID3_L_Knee_Velocity_z_subset);
if maxVelocity_ID3_L_Knee_Velocity_z >(abs(minVelocity_ID3_L_Knee_Velocity_z))
   pVelocity_ID3_L_Knee_Velocity_z = maxVelocity_ID3_L_Knee_Velocity_z;
else
   pVelocity_ID3_L_Knee_Velocity_z = minVelocity_ID3_L_Knee_Velocity_z;
end


ID3_L_Hip_Velocity_x_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_L_Hip_Velocity_x_FrameN = round((ID3_Post_350/.005)+1);
ID3_L_Hip_Velocity_x_subset =
ID3_L_Hip_Velocity_x(ID3_L_Hip_Velocity_x_Frame1:ID3_L_Hip_Velocity_x_FrameN);
maxVelocity_ID3_L_Hip_Velocity_x = max(ID3_L_Hip_Velocity_x_subset);
minVelocity_ID3_L_Hip_Velocity_x = min(ID3_L_Hip_Velocity_x_subset);
if maxVelocity_ID3_L_Hip_Velocity_x >(abs(minVelocity_ID3_L_Hip_Velocity_x))
   pVelocity_ID3_L_Hip_Velocity_x = maxVelocity_ID3_L_Hip_Velocity_x;
else
   pVelocity_ID3_L_Hip_Velocity_x = minVelocity_ID3_L_Hip_Velocity_x;
end


ID3_L_Hip_Velocity_y_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_L_Hip_Velocity_y_FrameN = round((ID3_Post_350/.005)+1);
ID3_L_Hip_Velocity_y_subset =
ID3_L_Hip_Velocity_y(ID3_L_Hip_Velocity_y_Frame1:ID3_L_Hip_Velocity_y_FrameN);
maxVelocity_ID3_L_Hip_Velocity_y = max(ID3_L_Hip_Velocity_y_subset);
minVelocity_ID3_L_Hip_Velocity_y = min(ID3_L_Hip_Velocity_y_subset);
if maxVelocity_ID3_L_Hip_Velocity_y >(abs(minVelocity_ID3_L_Hip_Velocity_y))
   pVelocity_ID3_L_Hip_Velocity_y = maxVelocity_ID3_L_Hip_Velocity_y;
else
   pVelocity_ID3_L_Hip_Velocity_y = minVelocity_ID3_L_Hip_Velocity_y;
end


ID3_L_Hip_Velocity_z_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_L_Hip_Velocity_z_FrameN = round((ID3_Post_350/.005)+1);
ID3_L_Hip_Velocity_z_subset =
ID3_L_Hip_Velocity_z(ID3_L_Hip_Velocity_z_Frame1:ID3_L_Hip_Velocity_z_FrameN);
maxVelocity_ID3_L_Hip_Velocity_z = max(ID3_L_Hip_Velocity_z_subset);
minVelocity_ID3_L_Hip_Velocity_z = min(ID3_L_Hip_Velocity_z_subset);
if maxVelocity_ID3_L_Hip_Velocity_z >(abs(minVelocity_ID3_L_Hip_Velocity_z))
   pVelocity_ID3_L_Hip_Velocity_z = maxVelocity_ID3_L_Hip_Velocity_z;
else
   pVelocity_ID3_L_Hip_Velocity_z = minVelocity_ID3_L_Hip_Velocity_z;
end
```

```
%IPD1
 IPD1_L_Ankle_Velocity_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Ankle_Velocity_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Ankle_Velocity_x_subset =
IPD1_L_Ankle_Velocity_x(IPD1_L_Ankle_Velocity_x_Frame1:IPD1_L_Ankle_Velocity_x_Fr
ameN);
 maxVelocity_IPD1_L_Ankle_Velocity_x = max(IPD1_L_Ankle_Velocity_x_subset);
 minVelocity_IPD1_L_Ankle_Velocity_x = min(IPD1_L_Ankle_Velocity_x_subset);
 if maxVelocity_IPD1_L_Ankle_Velocity_x >(abs(minVelocity_IPD1_L_Ankle_Velocity_x))
    pVelocity_IPD1_L_Ankle_Velocity_x = maxVelocity_IPD1_L_Ankle_Velocity_x;
 else
    pVelocity_IPD1_L_Ankle_Velocity_x = minVelocity_IPD1_L_Ankle_Velocity_x;
 end



 IPD1_L_Ankle_Velocity_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Ankle_Velocity_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Ankle_Velocity_y_subset =
IPD1_L_Ankle_Velocity_y(IPD1_L_Ankle_Velocity_y_Frame1:IPD1_L_Ankle_Velocity_y_Fr
ameN);
 maxVelocity_IPD1_L_Ankle_Velocity_y = max(IPD1_L_Ankle_Velocity_y_subset);
 minVelocity_IPD1_L_Ankle_Velocity_y = min(IPD1_L_Ankle_Velocity_y_subset);
 if maxVelocity_IPD1_L_Ankle_Velocity_y >(abs(minVelocity_IPD1_L_Ankle_Velocity_y))
    pVelocity_IPD1_L_Ankle_Velocity_y = maxVelocity_IPD1_L_Ankle_Velocity_y;
 else
    pVelocity_IPD1_L_Ankle_Velocity_y = minVelocity_IPD1_L_Ankle_Velocity_y;
 end



 IPD1_L_Ankle_Velocity_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Ankle_Velocity_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Ankle_Velocity_z_subset =
IPD1_L_Ankle_Velocity_z(IPD1_L_Ankle_Velocity_z_Frame1:IPD1_L_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_IPD1_L_Ankle_Velocity_z = max(IPD1_L_Ankle_Velocity_z_subset);
 minVelocity_IPD1_L_Ankle_Velocity_z = min(IPD1_L_Ankle_Velocity_z_subset);
 if maxVelocity_IPD1_L_Ankle_Velocity_z >(abs(minVelocity_IPD1_L_Ankle_Velocity_z))
    pVelocity_IPD1_L_Ankle_Velocity_z = maxVelocity_IPD1_L_Ankle_Velocity_z;
 else
    pVelocity_IPD1_L_Ankle_Velocity_z = minVelocity_IPD1_L_Ankle_Velocity_z;
 end
```

```
 IPD1_L_Knee_Velocity_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Knee_Velocity_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Knee_Velocity_x_subset =
IPD1_L_Knee_Velocity_x(IPD1_L_Knee_Velocity_x_Frame1:IPD1_L_Knee_Velocity_x_Fra
meN);
 maxVelocity_IPD1_L_Knee_Velocity_x = max(IPD1_L_Knee_Velocity_x_subset);
 minVelocity_IPD1_L_Knee_Velocity_x = min(IPD1_L_Knee_Velocity_x_subset);
 if maxVelocity_IPD1_L_Knee_Velocity_x >(abs(minVelocity_IPD1_L_Knee_Velocity_x))
    pVelocity_IPD1_L_Knee_Velocity_x = maxVelocity_IPD1_L_Knee_Velocity_x;
 else
    pVelocity_IPD1_L_Knee_Velocity_x = minVelocity_IPD1_L_Knee_Velocity_x;
 end



 IPD1_L_Knee_Velocity_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Knee_Velocity_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Knee_Velocity_y_subset =
IPD1_L_Knee_Velocity_y(IPD1_L_Knee_Velocity_y_Frame1:IPD1_L_Knee_Velocity_y_Fra
meN);
 maxVelocity_IPD1_L_Knee_Velocity_y = max(IPD1_L_Knee_Velocity_y_subset);
 minVelocity_IPD1_L_Knee_Velocity_y = min(IPD1_L_Knee_Velocity_y_subset);
 if maxVelocity_IPD1_L_Knee_Velocity_y >(abs(minVelocity_IPD1_L_Knee_Velocity_y))
    pVelocity_IPD1_L_Knee_Velocity_y = maxVelocity_IPD1_L_Knee_Velocity_y;
 else
    pVelocity_IPD1_L_Knee_Velocity_y = minVelocity_IPD1_L_Knee_Velocity_y;
 end


 IPD1_L_Knee_Velocity_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Knee_Velocity_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Knee_Velocity_z_subset =
IPD1_L_Knee_Velocity_z(IPD1_L_Knee_Velocity_z_Frame1:IPD1_L_Knee_Velocity_z_Fram
eN);
 maxVelocity_IPD1_L_Knee_Velocity_z = max(IPD1_L_Knee_Velocity_z_subset);
 minVelocity_IPD1_L_Knee_Velocity_z = min(IPD1_L_Knee_Velocity_z_subset);
 if maxVelocity_IPD1_L_Knee_Velocity_z >(abs(minVelocity_IPD1_L_Knee_Velocity_z))
    pVelocity_IPD1_L_Knee_Velocity_z = maxVelocity_IPD1_L_Knee_Velocity_z;
 else
    pVelocity_IPD1_L_Knee_Velocity_z = minVelocity_IPD1_L_Knee_Velocity_z;
 end


 IPD1_L_Hip_Velocity_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Hip_Velocity_x_FrameN = round((IPD1_Post_350/.005)+1);
```

```
 IPD1_L_Hip_Velocity_x_subset =
IPD1_L_Hip_Velocity_x(IPD1_L_Hip_Velocity_x_Frame1:IPD1_L_Hip_Velocity_x_FrameN)
;
 maxVelocity_IPD1_L_Hip_Velocity_x = max(IPD1_L_Hip_Velocity_x_subset);
 minVelocity_IPD1_L_Hip_Velocity_x = min(IPD1_L_Hip_Velocity_x_subset);
 if maxVelocity_IPD1_L_Hip_Velocity_x >(abs(minVelocity_IPD1_L_Hip_Velocity_x))
    pVelocity_IPD1_L_Hip_Velocity_x = maxVelocity_IPD1_L_Hip_Velocity_x;
 else
    pVelocity_IPD1_L_Hip_Velocity_x = minVelocity_IPD1_L_Hip_Velocity_x;
 end


 IPD1_L_Hip_Velocity_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Hip_Velocity_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Hip_Velocity_y_subset =
IPD1_L_Hip_Velocity_y(IPD1_L_Hip_Velocity_y_Frame1:IPD1_L_Hip_Velocity_y_FrameN)
;
 maxVelocity_IPD1_L_Hip_Velocity_y = max(IPD1_L_Hip_Velocity_y_subset);
 minVelocity_IPD1_L_Hip_Velocity_y = min(IPD1_L_Hip_Velocity_y_subset);
 if maxVelocity_IPD1_L_Hip_Velocity_y >(abs(minVelocity_IPD1_L_Hip_Velocity_y))
    pVelocity_IPD1_L_Hip_Velocity_y = maxVelocity_IPD1_L_Hip_Velocity_y;
 else
    pVelocity_IPD1_L_Hip_Velocity_y = minVelocity_IPD1_L_Hip_Velocity_y;
 end


 IPD1_L_Hip_Velocity_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_L_Hip_Velocity_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_L_Hip_Velocity_z_subset =
IPD1_L_Hip_Velocity_z(IPD1_L_Hip_Velocity_z_Frame1:IPD1_L_Hip_Velocity_z_FrameN);
 maxVelocity_IPD1_L_Hip_Velocity_z = max(IPD1_L_Hip_Velocity_z_subset);
 minVelocity_IPD1_L_Hip_Velocity_z = min(IPD1_L_Hip_Velocity_z_subset);
 if maxVelocity_IPD1_L_Hip_Velocity_z >(abs(minVelocity_IPD1_L_Hip_Velocity_z))
    pVelocity_IPD1_L_Hip_Velocity_z = maxVelocity_IPD1_L_Hip_Velocity_z;
 else
    pVelocity_IPD1_L_Hip_Velocity_z = minVelocity_IPD1_L_Hip_Velocity_z;
 end

 %IPD2
 IPD2_L_Ankle_Velocity_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Ankle_Velocity_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Ankle_Velocity_x_subset =
IPD2_L_Ankle_Velocity_x(IPD2_L_Ankle_Velocity_x_Frame1:IPD2_L_Ankle_Velocity_x_Fr
ameN);
 maxVelocity_IPD2_L_Ankle_Velocity_x = max(IPD2_L_Ankle_Velocity_x_subset);
 minVelocity_IPD2_L_Ankle_Velocity_x = min(IPD2_L_Ankle_Velocity_x_subset);
```

```
 if maxVelocity_IPD2_L_Ankle_Velocity_x >(abs(minVelocity_IPD2_L_Ankle_Velocity_x))
    pVelocity_IPD2_L_Ankle_Velocity_x = maxVelocity_IPD2_L_Ankle_Velocity_x;
 else
    pVelocity_IPD2_L_Ankle_Velocity_x = minVelocity_IPD2_L_Ankle_Velocity_x;
 end


 IPD2_L_Ankle_Velocity_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Ankle_Velocity_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Ankle_Velocity_y_subset =
IPD2_L_Ankle_Velocity_y(IPD2_L_Ankle_Velocity_y_Frame1:IPD2_L_Ankle_Velocity_y_Fr
ameN);
 maxVelocity_IPD2_L_Ankle_Velocity_y = max(IPD2_L_Ankle_Velocity_y_subset);
 minVelocity_IPD2_L_Ankle_Velocity_y = min(IPD2_L_Ankle_Velocity_y_subset);
 if maxVelocity_IPD2_L_Ankle_Velocity_y >(abs(minVelocity_IPD2_L_Ankle_Velocity_y))
    pVelocity_IPD2_L_Ankle_Velocity_y = maxVelocity_IPD2_L_Ankle_Velocity_y;
 else
    pVelocity_IPD2_L_Ankle_Velocity_y = minVelocity_IPD2_L_Ankle_Velocity_y;
 end


 IPD2_L_Ankle_Velocity_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Ankle_Velocity_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Ankle_Velocity_z_subset =
IPD2_L_Ankle_Velocity_z(IPD2_L_Ankle_Velocity_z_Frame1:IPD2_L_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_IPD2_L_Ankle_Velocity_z = max(IPD2_L_Ankle_Velocity_z_subset);
 minVelocity_IPD2_L_Ankle_Velocity_z = min(IPD2_L_Ankle_Velocity_z_subset);
 if maxVelocity_IPD2_L_Ankle_Velocity_z >(abs(minVelocity_IPD2_L_Ankle_Velocity_z))
    pVelocity_IPD2_L_Ankle_Velocity_z = maxVelocity_IPD2_L_Ankle_Velocity_z;
 else
    pVelocity_IPD2_L_Ankle_Velocity_z = minVelocity_IPD2_L_Ankle_Velocity_z;
 end


 IPD2_L_Knee_Velocity_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Knee_Velocity_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Knee_Velocity_x_subset =
IPD2_L_Knee_Velocity_x(IPD2_L_Knee_Velocity_x_Frame1:IPD2_L_Knee_Velocity_x_Fra
meN);
 maxVelocity_IPD2_L_Knee_Velocity_x = max(IPD2_L_Knee_Velocity_x_subset);
 minVelocity_IPD2_L_Knee_Velocity_x = min(IPD2_L_Knee_Velocity_x_subset);
 if maxVelocity_IPD2_L_Knee_Velocity_x >(abs(minVelocity_IPD2_L_Knee_Velocity_x))
    pVelocity_IPD2_L_Knee_Velocity_x = maxVelocity_IPD2_L_Knee_Velocity_x;
 else
    pVelocity_IPD2_L_Knee_Velocity_x = minVelocity_IPD2_L_Knee_Velocity_x;
```

```
end


IPD2_L_Knee_Velocity_y_Frame1 = round((IPD2_True_Drop/.005)+1);
IPD2_L_Knee_Velocity_y_FrameN = round((IPD2_Post_350/.005)+1);
IPD2_L_Knee_Velocity_y_subset =
IPD2_L_Knee_Velocity_y(IPD2_L_Knee_Velocity_y_Frame1:IPD2_L_Knee_Velocity_y_Fra
meN);
maxVelocity_IPD2_L_Knee_Velocity_y = max(IPD2_L_Knee_Velocity_y_subset);
minVelocity_IPD2_L_Knee_Velocity_y = min(IPD2_L_Knee_Velocity_y_subset);
if maxVelocity_IPD2_L_Knee_Velocity_y >(abs(minVelocity_IPD2_L_Knee_Velocity_y))
   pVelocity_IPD2_L_Knee_Velocity_y = maxVelocity_IPD2_L_Knee_Velocity_y;
else
   pVelocity_IPD2_L_Knee_Velocity_y = minVelocity_IPD2_L_Knee_Velocity_y;
end


IPD2_L_Knee_Velocity_z_Frame1 = round((IPD2_True_Drop/.005)+1);
IPD2_L_Knee_Velocity_z_FrameN = round((IPD2_Post_350/.005)+1);
IPD2_L_Knee_Velocity_z_subset =
IPD2_L_Knee_Velocity_z(IPD2_L_Knee_Velocity_z_Frame1:IPD2_L_Knee_Velocity_z_Fram
eN);
maxVelocity_IPD2_L_Knee_Velocity_z = max(IPD2_L_Knee_Velocity_z_subset);
minVelocity_IPD2_L_Knee_Velocity_z = min(IPD2_L_Knee_Velocity_z_subset);
if maxVelocity_IPD2_L_Knee_Velocity_z >(abs(minVelocity_IPD2_L_Knee_Velocity_z))
   pVelocity_IPD2_L_Knee_Velocity_z = maxVelocity_IPD2_L_Knee_Velocity_z;
else
   pVelocity_IPD2_L_Knee_Velocity_z = minVelocity_IPD2_L_Knee_Velocity_z;
end


IPD2_L_Hip_Velocity_x_Frame1 = round((IPD2_True_Drop/.005)+1);
IPD2_L_Hip_Velocity_x_FrameN = round((IPD2_Post_350/.005)+1);
IPD2_L_Hip_Velocity_x_subset =
IPD2_L_Hip_Velocity_x(IPD2_L_Hip_Velocity_x_Frame1:IPD2_L_Hip_Velocity_x_FrameN)
;
maxVelocity_IPD2_L_Hip_Velocity_x = max(IPD2_L_Hip_Velocity_x_subset);
minVelocity_IPD2_L_Hip_Velocity_x = min(IPD2_L_Hip_Velocity_x_subset);
if maxVelocity_IPD2_L_Hip_Velocity_x >(abs(minVelocity_IPD2_L_Hip_Velocity_x))
   pVelocity_IPD2_L_Hip_Velocity_x = maxVelocity_IPD2_L_Hip_Velocity_x;
else
   pVelocity_IPD2_L_Hip_Velocity_x = minVelocity_IPD2_L_Hip_Velocity_x;
end


IPD2_L_Hip_Velocity_y_Frame1 = round((IPD2_True_Drop/.005)+1);
```

```
 IPD2_L_Hip_Velocity_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Hip_Velocity_y_subset =
IPD2_L_Hip_Velocity_y(IPD2_L_Hip_Velocity_y_Frame1:IPD2_L_Hip_Velocity_y_FrameN)
;
 maxVelocity_IPD2_L_Hip_Velocity_y = max(IPD2_L_Hip_Velocity_y_subset);
 minVelocity_IPD2_L_Hip_Velocity_y = min(IPD2_L_Hip_Velocity_y_subset);
 if maxVelocity_IPD2_L_Hip_Velocity_y >(abs(minVelocity_IPD2_L_Hip_Velocity_y))
    pVelocity_IPD2_L_Hip_Velocity_y = maxVelocity_IPD2_L_Hip_Velocity_y;
 else
    pVelocity_IPD2_L_Hip_Velocity_y = minVelocity_IPD2_L_Hip_Velocity_y;
 end


 IPD2_L_Hip_Velocity_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_L_Hip_Velocity_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_L_Hip_Velocity_z_subset =
IPD2_L_Hip_Velocity_z(IPD2_L_Hip_Velocity_z_Frame1:IPD2_L_Hip_Velocity_z_FrameN);
 maxVelocity_IPD2_L_Hip_Velocity_z = max(IPD2_L_Hip_Velocity_z_subset);
 minVelocity_IPD2_L_Hip_Velocity_z = min(IPD2_L_Hip_Velocity_z_subset);
 if maxVelocity_IPD2_L_Hip_Velocity_z >(abs(minVelocity_IPD2_L_Hip_Velocity_z))
    pVelocity_IPD2_L_Hip_Velocity_z = maxVelocity_IPD2_L_Hip_Velocity_z;
 else
    pVelocity_IPD2_L_Hip_Velocity_z = minVelocity_IPD2_L_Hip_Velocity_z;
 end


 %IPD3
 IPD3_L_Ankle_Velocity_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Ankle_Velocity_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Ankle_Velocity_x_subset =
IPD3_L_Ankle_Velocity_x(IPD3_L_Ankle_Velocity_x_Frame1:IPD3_L_Ankle_Velocity_x_Fr
ameN);
 maxVelocity_IPD3_L_Ankle_Velocity_x = max(IPD3_L_Ankle_Velocity_x_subset);
 minVelocity_IPD3_L_Ankle_Velocity_x = min(IPD3_L_Ankle_Velocity_x_subset);
 if maxVelocity_IPD3_L_Ankle_Velocity_x >(abs(minVelocity_IPD3_L_Ankle_Velocity_x))
    pVelocity_IPD3_L_Ankle_Velocity_x = maxVelocity_IPD3_L_Ankle_Velocity_x;
 else
    pVelocity_IPD3_L_Ankle_Velocity_x = minVelocity_IPD3_L_Ankle_Velocity_x;
 end


 IPD3_L_Ankle_Velocity_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Ankle_Velocity_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Ankle_Velocity_y_subset =
IPD3_L_Ankle_Velocity_y(IPD3_L_Ankle_Velocity_y_Frame1:IPD3_L_Ankle_Velocity_y_Fr
ameN);
```

```
maxVelocity_IPD3_L_Ankle_Velocity_y = max(IPD3_L_Ankle_Velocity_y_subset);
minVelocity_IPD3_L_Ankle_Velocity_y = min(IPD3_L_Ankle_Velocity_y_subset);
if maxVelocity_IPD3_L_Ankle_Velocity_y >(abs(minVelocity_IPD3_L_Ankle_Velocity_y))
   pVelocity_IPD3_L_Ankle_Velocity_y = maxVelocity_IPD3_L_Ankle_Velocity_y;
else
   pVelocity_IPD3_L_Ankle_Velocity_y = minVelocity_IPD3_L_Ankle_Velocity_y;
end


IPD3_L_Ankle_Velocity_z_Frame1 = round((IPD3_True_Drop/.005)+1);
IPD3_L_Ankle_Velocity_z_FrameN = round((IPD3_Post_350/.005)+1);
IPD3_L_Ankle_Velocity_z_subset =
IPD3_L_Ankle_Velocity_z(IPD3_L_Ankle_Velocity_z_Frame1:IPD3_L_Ankle_Velocity_z_Fr
ameN);
maxVelocity_IPD3_L_Ankle_Velocity_z = max(IPD3_L_Ankle_Velocity_z_subset);
minVelocity_IPD3_L_Ankle_Velocity_z = min(IPD3_L_Ankle_Velocity_z_subset);
if maxVelocity_IPD3_L_Ankle_Velocity_z >(abs(minVelocity_IPD3_L_Ankle_Velocity_z))
   pVelocity_IPD3_L_Ankle_Velocity_z = maxVelocity_IPD3_L_Ankle_Velocity_z;
else
   pVelocity_IPD3_L_Ankle_Velocity_z = minVelocity_IPD3_L_Ankle_Velocity_z;
end


IPD3_L_Knee_Velocity_x_Frame1 = round((IPD3_True_Drop/.005)+1);
IPD3_L_Knee_Velocity_x_FrameN = round((IPD3_Post_350/.005)+1);
IPD3_L_Knee_Velocity_x_subset =
IPD3_L_Knee_Velocity_x(IPD3_L_Knee_Velocity_x_Frame1:IPD3_L_Knee_Velocity_x_Fra
meN);
maxVelocity_IPD3_L_Knee_Velocity_x = max(IPD3_L_Knee_Velocity_x_subset);
minVelocity_IPD3_L_Knee_Velocity_x = min(IPD3_L_Knee_Velocity_x_subset);
if maxVelocity_IPD3_L_Knee_Velocity_x >(abs(minVelocity_IPD3_L_Knee_Velocity_x))
   pVelocity_IPD3_L_Knee_Velocity_x = maxVelocity_IPD3_L_Knee_Velocity_x;
else
   pVelocity_IPD3_L_Knee_Velocity_x = minVelocity_IPD3_L_Knee_Velocity_x;
end


IPD3_L_Knee_Velocity_y_Frame1 = round((IPD3_True_Drop/.005)+1);
IPD3_L_Knee_Velocity_y_FrameN = round((IPD3_Post_350/.005)+1);
IPD3_L_Knee_Velocity_y_subset =
IPD3_L_Knee_Velocity_y(IPD3_L_Knee_Velocity_y_Frame1:IPD3_L_Knee_Velocity_y_Fra
meN);
maxVelocity_IPD3_L_Knee_Velocity_y = max(IPD3_L_Knee_Velocity_y_subset);
minVelocity_IPD3_L_Knee_Velocity_y = min(IPD3_L_Knee_Velocity_y_subset);
if maxVelocity_IPD3_L_Knee_Velocity_y >(abs(minVelocity_IPD3_L_Knee_Velocity_y))
   pVelocity_IPD3_L_Knee_Velocity_y = maxVelocity_IPD3_L_Knee_Velocity_y;
```

```
else
   pVelocity_IPD3_L_Knee_Velocity_y = minVelocity_IPD3_L_Knee_Velocity_y;
end


IPD3_L_Knee_Velocity_z_Frame1 = round((IPD3_True_Drop/.005)+1);
IPD3_L_Knee_Velocity_z_FrameN = round((IPD3_Post_350/.005)+1);
IPD3_L_Knee_Velocity_z_subset =
IPD3_L_Knee_Velocity_z(IPD3_L_Knee_Velocity_z_Frame1:IPD3_L_Knee_Velocity_z_Fram
eN);
maxVelocity_IPD3_L_Knee_Velocity_z = max(IPD3_L_Knee_Velocity_z_subset);
minVelocity_IPD3_L_Knee_Velocity_z = min(IPD3_L_Knee_Velocity_z_subset);
if maxVelocity_IPD3_L_Knee_Velocity_z >(abs(minVelocity_IPD3_L_Knee_Velocity_z))
   pVelocity_IPD3_L_Knee_Velocity_z = maxVelocity_IPD3_L_Knee_Velocity_z;
else
   pVelocity_IPD3_L_Knee_Velocity_z = minVelocity_IPD3_L_Knee_Velocity_z;
end


IPD3_L_Hip_Velocity_x_Frame1 = round((IPD3_True_Drop/.005)+1);
IPD3_L_Hip_Velocity_x_FrameN = round((IPD3_Post_350/.005)+1);
IPD3_L_Hip_Velocity_x_subset =
IPD3_L_Hip_Velocity_x(IPD3_L_Hip_Velocity_x_Frame1:IPD3_L_Hip_Velocity_x_FrameN)
;
maxVelocity_IPD3_L_Hip_Velocity_x = max(IPD3_L_Hip_Velocity_x_subset);
minVelocity_IPD3_L_Hip_Velocity_x = min(IPD3_L_Hip_Velocity_x_subset);
if maxVelocity_IPD3_L_Hip_Velocity_x >(abs(minVelocity_IPD3_L_Hip_Velocity_x))
   pVelocity_IPD3_L_Hip_Velocity_x = maxVelocity_IPD3_L_Hip_Velocity_x;
else
   pVelocity_IPD3_L_Hip_Velocity_x = minVelocity_IPD3_L_Hip_Velocity_x;
end


IPD3_L_Hip_Velocity_y_Frame1 = round((IPD3_True_Drop/.005)+1);
IPD3_L_Hip_Velocity_y_FrameN = round((IPD3_Post_350/.005)+1);
IPD3_L_Hip_Velocity_y_subset =
IPD3_L_Hip_Velocity_y(IPD3_L_Hip_Velocity_y_Frame1:IPD3_L_Hip_Velocity_y_FrameN)
;
maxVelocity_IPD3_L_Hip_Velocity_y = max(IPD3_L_Hip_Velocity_y_subset);
minVelocity_IPD3_L_Hip_Velocity_y = min(IPD3_L_Hip_Velocity_y_subset);
if maxVelocity_IPD3_L_Hip_Velocity_y >(abs(minVelocity_IPD3_L_Hip_Velocity_y))
   pVelocity_IPD3_L_Hip_Velocity_y = maxVelocity_IPD3_L_Hip_Velocity_y;
else
   pVelocity_IPD3_L_Hip_Velocity_y = minVelocity_IPD3_L_Hip_Velocity_y;
end
```

```
 IPD3_L_Hip_Velocity_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_L_Hip_Velocity_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_L_Hip_Velocity_z_subset =
IPD3_L_Hip_Velocity_z(IPD3_L_Hip_Velocity_z_Frame1:IPD3_L_Hip_Velocity_z_FrameN);
 maxVelocity_IPD3_L_Hip_Velocity_z = max(IPD3_L_Hip_Velocity_z_subset);
 minVelocity_IPD3_L_Hip_Velocity_z = min(IPD3_L_Hip_Velocity_z_subset);
 if maxVelocity_IPD3_L_Hip_Velocity_z >(abs(minVelocity_IPD3_L_Hip_Velocity_z))
    pVelocity_IPD3_L_Hip_Velocity_z = maxVelocity_IPD3_L_Hip_Velocity_z;
 else
    pVelocity_IPD3_L_Hip_Velocity_z = minVelocity_IPD3_L_Hip_Velocity_z;
 end

 %NW1
 NW1_L_Ankle_Velocity_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Ankle_Velocity_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Ankle_Velocity_x_subset =
NW1_L_Ankle_Velocity_x(NW1_L_Ankle_Velocity_x_Frame1:NW1_L_Ankle_Velocity_x_F
rameN);
 maxVelocity_NW1_L_Ankle_Velocity_x = max(NW1_L_Ankle_Velocity_x_subset);
 minVelocity_NW1_L_Ankle_Velocity_x = min(NW1_L_Ankle_Velocity_x_subset);
 if maxVelocity_NW1_L_Ankle_Velocity_x >(abs(minVelocity_NW1_L_Ankle_Velocity_x))
    pVelocity_NW1_L_Ankle_Velocity_x = maxVelocity_NW1_L_Ankle_Velocity_x;
 else
    pVelocity_NW1_L_Ankle_Velocity_x = minVelocity_NW1_L_Ankle_Velocity_x;
 end


 NW1_L_Ankle_Velocity_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Ankle_Velocity_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Ankle_Velocity_y_subset =
NW1_L_Ankle_Velocity_y(NW1_L_Ankle_Velocity_y_Frame1:NW1_L_Ankle_Velocity_y_F
rameN);
 maxVelocity_NW1_L_Ankle_Velocity_y = max(NW1_L_Ankle_Velocity_y_subset);
 minVelocity_NW1_L_Ankle_Velocity_y = min(NW1_L_Ankle_Velocity_y_subset);
 if maxVelocity_NW1_L_Ankle_Velocity_y >(abs(minVelocity_NW1_L_Ankle_Velocity_y))
    pVelocity_NW1_L_Ankle_Velocity_y = maxVelocity_NW1_L_Ankle_Velocity_y;
 else
    pVelocity_NW1_L_Ankle_Velocity_y = minVelocity_NW1_L_Ankle_Velocity_y;
 end


 NW1_L_Ankle_Velocity_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Ankle_Velocity_z_FrameN = round((NW1_Post_350/.005)+1);
```

```
 NW1_L_Ankle_Velocity_z_subset =
NW1_L_Ankle_Velocity_z(NW1_L_Ankle_Velocity_z_Frame1:NW1_L_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_NW1_L_Ankle_Velocity_z = max(NW1_L_Ankle_Velocity_z_subset);
 minVelocity_NW1_L_Ankle_Velocity_z = min(NW1_L_Ankle_Velocity_z_subset);
 if maxVelocity_NW1_L_Ankle_Velocity_z >(abs(minVelocity_NW1_L_Ankle_Velocity_z))
    pVelocity_NW1_L_Ankle_Velocity_z = maxVelocity_NW1_L_Ankle_Velocity_z;
 else
    pVelocity_NW1_L_Ankle_Velocity_z = minVelocity_NW1_L_Ankle_Velocity_z;
 end


 NW1_L_Knee_Velocity_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Knee_Velocity_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Knee_Velocity_x_subset =
NW1_L_Knee_Velocity_x(NW1_L_Knee_Velocity_x_Frame1:NW1_L_Knee_Velocity_x_Fra
meN);
 maxVelocity_NW1_L_Knee_Velocity_x = max(NW1_L_Knee_Velocity_x_subset);
 minVelocity_NW1_L_Knee_Velocity_x = min(NW1_L_Knee_Velocity_x_subset);
 if maxVelocity_NW1_L_Knee_Velocity_x >(abs(minVelocity_NW1_L_Knee_Velocity_x))
    pVelocity_NW1_L_Knee_Velocity_x = maxVelocity_NW1_L_Knee_Velocity_x;
 else
    pVelocity_NW1_L_Knee_Velocity_x = minVelocity_NW1_L_Knee_Velocity_x;
 end


 NW1_L_Knee_Velocity_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Knee_Velocity_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Knee_Velocity_y_subset =
NW1_L_Knee_Velocity_y(NW1_L_Knee_Velocity_y_Frame1:NW1_L_Knee_Velocity_y_Fra
meN);
 maxVelocity_NW1_L_Knee_Velocity_y = max(NW1_L_Knee_Velocity_y_subset);
 minVelocity_NW1_L_Knee_Velocity_y = min(NW1_L_Knee_Velocity_y_subset);
 if maxVelocity_NW1_L_Knee_Velocity_y >(abs(minVelocity_NW1_L_Knee_Velocity_y))
    pVelocity_NW1_L_Knee_Velocity_y = maxVelocity_NW1_L_Knee_Velocity_y;
 else
    pVelocity_NW1_L_Knee_Velocity_y = minVelocity_NW1_L_Knee_Velocity_y;
 end


 NW1_L_Knee_Velocity_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_L_Knee_Velocity_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_L_Knee_Velocity_z_subset =
NW1_L_Knee_Velocity_z(NW1_L_Knee_Velocity_z_Frame1:NW1_L_Knee_Velocity_z_Fra
meN);
 maxVelocity_NW1_L_Knee_Velocity_z = max(NW1_L_Knee_Velocity_z_subset);
```

```
minVelocity_NW1_L_Knee_Velocity_z = min(NW1_L_Knee_Velocity_z_subset);
if maxVelocity_NW1_L_Knee_Velocity_z >(abs(minVelocity_NW1_L_Knee_Velocity_z))
   pVelocity_NW1_L_Knee_Velocity_z = maxVelocity_NW1_L_Knee_Velocity_z;
else
   pVelocity_NW1_L_Knee_Velocity_z = minVelocity_NW1_L_Knee_Velocity_z;
end


NW1_L_Hip_Velocity_x_Frame1 = round((NW1_HS/.005)+1);
NW1_L_Hip_Velocity_x_FrameN = round((NW1_Post_350/.005)+1);
NW1_L_Hip_Velocity_x_subset =
NW1_L_Hip_Velocity_x(NW1_L_Hip_Velocity_x_Frame1:NW1_L_Hip_Velocity_x_FrameN)
;
maxVelocity_NW1_L_Hip_Velocity_x = max(NW1_L_Hip_Velocity_x_subset);
minVelocity_NW1_L_Hip_Velocity_x = min(NW1_L_Hip_Velocity_x_subset);
if maxVelocity_NW1_L_Hip_Velocity_x >(abs(minVelocity_NW1_L_Hip_Velocity_x))
   pVelocity_NW1_L_Hip_Velocity_x = maxVelocity_NW1_L_Hip_Velocity_x;
else
   pVelocity_NW1_L_Hip_Velocity_x = minVelocity_NW1_L_Hip_Velocity_x;
end


NW1_L_Hip_Velocity_y_Frame1 = round((NW1_HS/.005)+1);
NW1_L_Hip_Velocity_y_FrameN = round((NW1_Post_350/.005)+1);
NW1_L_Hip_Velocity_y_subset =
NW1_L_Hip_Velocity_x(NW1_L_Hip_Velocity_y_Frame1:NW1_L_Hip_Velocity_y_FrameN)
;
maxVelocity_NW1_L_Hip_Velocity_y = max(NW1_L_Hip_Velocity_y_subset);
minVelocity_NW1_L_Hip_Velocity_y = min(NW1_L_Hip_Velocity_y_subset);
if maxVelocity_NW1_L_Hip_Velocity_y >(abs(minVelocity_NW1_L_Hip_Velocity_y))
   pVelocity_NW1_L_Hip_Velocity_y = maxVelocity_NW1_L_Hip_Velocity_y;
else
   pVelocity_NW1_L_Hip_Velocity_y = minVelocity_NW1_L_Hip_Velocity_y;
end


NW1_L_Hip_Velocity_z_Frame1 = round((NW1_HS/.005)+1);
NW1_L_Hip_Velocity_z_FrameN = round((NW1_Post_350/.005)+1);
NW1_L_Hip_Velocity_z_subset =
NW1_L_Hip_Velocity_x(NW1_L_Hip_Velocity_z_Frame1:NW1_L_Hip_Velocity_z_FrameN)
;
maxVelocity_NW1_L_Hip_Velocity_z = max(NW1_L_Hip_Velocity_z_subset);
minVelocity_NW1_L_Hip_Velocity_z = min(NW1_L_Hip_Velocity_z_subset);
if maxVelocity_NW1_L_Hip_Velocity_z >(abs(minVelocity_NW1_L_Hip_Velocity_z))
   pVelocity_NW1_L_Hip_Velocity_z = maxVelocity_NW1_L_Hip_Velocity_z;
else
```

```
    pVelocity_NW1_L_Hip_Velocity_z = minVelocity_NW1_L_Hip_Velocity_z;
 end


 %NW2
 NW2_L_Ankle_Velocity_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Ankle_Velocity_x_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Ankle_Velocity_x_subset =
NW2_L_Ankle_Velocity_x(NW2_L_Ankle_Velocity_x_Frame1:NW2_L_Ankle_Velocity_x_F
rameN);
 maxVelocity_NW2_L_Ankle_Velocity_x = max(NW2_L_Ankle_Velocity_x_subset);
 minVelocity_NW2_L_Ankle_Velocity_x = min(NW2_L_Ankle_Velocity_x_subset);
 if maxVelocity_NW2_L_Ankle_Velocity_x >(abs(minVelocity_NW2_L_Ankle_Velocity_x))
    pVelocity_NW2_L_Ankle_Velocity_x = maxVelocity_NW2_L_Ankle_Velocity_x;
 else
    pVelocity_NW2_L_Ankle_Velocity_x = minVelocity_NW2_L_Ankle_Velocity_x;
 end


 NW2_L_Ankle_Velocity_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Ankle_Velocity_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Ankle_Velocity_y_subset =
NW2_L_Ankle_Velocity_y(NW2_L_Ankle_Velocity_y_Frame1:NW2_L_Ankle_Velocity_y_F
rameN);
 maxVelocity_NW2_L_Ankle_Velocity_y = max(NW2_L_Ankle_Velocity_y_subset);
 minVelocity_NW2_L_Ankle_Velocity_y = min(NW2_L_Ankle_Velocity_y_subset);
 if maxVelocity_NW2_L_Ankle_Velocity_y >(abs(minVelocity_NW2_L_Ankle_Velocity_y))
    pVelocity_NW2_L_Ankle_Velocity_y = maxVelocity_NW2_L_Ankle_Velocity_y;
 else
    pVelocity_NW2_L_Ankle_Velocity_y = minVelocity_NW2_L_Ankle_Velocity_y;
 end


 NW2_L_Ankle_Velocity_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Ankle_Velocity_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Ankle_Velocity_z_subset =
NW2_L_Ankle_Velocity_z(NW2_L_Ankle_Velocity_z_Frame1:NW2_L_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_NW2_L_Ankle_Velocity_z = max(NW2_L_Ankle_Velocity_z_subset);
 minVelocity_NW2_L_Ankle_Velocity_z = min(NW2_L_Ankle_Velocity_z_subset);
 if maxVelocity_NW2_L_Ankle_Velocity_z >(abs(minVelocity_NW2_L_Ankle_Velocity_z))
    pVelocity_NW2_L_Ankle_Velocity_z = maxVelocity_NW2_L_Ankle_Velocity_z;
 else
    pVelocity_NW2_L_Ankle_Velocity_z = minVelocity_NW2_L_Ankle_Velocity_z;
 end
```

```
 NW2_L_Knee_Velocity_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Knee_Velocity_x_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Knee_Velocity_x_subset =
NW2_L_Knee_Velocity_x(NW2_L_Knee_Velocity_x_Frame1:NW2_L_Knee_Velocity_x_Fra
meN);
 maxVelocity_NW2_L_Knee_Velocity_x = max(NW2_L_Knee_Velocity_x_subset);
 minVelocity_NW2_L_Knee_Velocity_x = min(NW2_L_Knee_Velocity_x_subset);
 if maxVelocity_NW2_L_Knee_Velocity_x >(abs(minVelocity_NW2_L_Knee_Velocity_x))
    pVelocity_NW2_L_Knee_Velocity_x = maxVelocity_NW2_L_Knee_Velocity_x;
 else
    pVelocity_NW2_L_Knee_Velocity_x = minVelocity_NW2_L_Knee_Velocity_x;
 end


 NW2_L_Knee_Velocity_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Knee_Velocity_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Knee_Velocity_y_subset =
NW2_L_Knee_Velocity_y(NW2_L_Knee_Velocity_y_Frame1:NW2_L_Knee_Velocity_y_Fra
meN);
 maxVelocity_NW2_L_Knee_Velocity_y = max(NW2_L_Knee_Velocity_y_subset);
 minVelocity_NW2_L_Knee_Velocity_y = min(NW2_L_Knee_Velocity_y_subset);
 if maxVelocity_NW2_L_Knee_Velocity_y >(abs(minVelocity_NW2_L_Knee_Velocity_y))
    pVelocity_NW2_L_Knee_Velocity_y = maxVelocity_NW2_L_Knee_Velocity_y;
 else
    pVelocity_NW2_L_Knee_Velocity_y = minVelocity_NW2_L_Knee_Velocity_y;
 end


 NW2_L_Knee_Velocity_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Knee_Velocity_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Knee_Velocity_z_subset =
NW2_L_Knee_Velocity_z(NW2_L_Knee_Velocity_z_Frame1:NW2_L_Knee_Velocity_z_Fra
meN);
 maxVelocity_NW2_L_Knee_Velocity_z = max(NW2_L_Knee_Velocity_z_subset);
 minVelocity_NW2_L_Knee_Velocity_z = min(NW2_L_Knee_Velocity_z_subset);
 if maxVelocity_NW2_L_Knee_Velocity_z >(abs(minVelocity_NW2_L_Knee_Velocity_z))
    pVelocity_NW2_L_Knee_Velocity_z = maxVelocity_NW2_L_Knee_Velocity_z;
 else
    pVelocity_NW2_L_Knee_Velocity_z = minVelocity_NW2_L_Knee_Velocity_z;
 end


 NW2_L_Hip_Velocity_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Hip_Velocity_x_FrameN = round((NW2_Post_350/.005)+1);
```

```
 NW2_L_Hip_Velocity_x_subset =
NW2_L_Hip_Velocity_x(NW2_L_Hip_Velocity_x_Frame1:NW2_L_Hip_Velocity_x_FrameN)
;
 maxVelocity_NW2_L_Hip_Velocity_x = max(NW2_L_Hip_Velocity_x_subset);
 minVelocity_NW2_L_Hip_Velocity_x = min(NW2_L_Hip_Velocity_x_subset);
 if maxVelocity_NW2_L_Hip_Velocity_x >(abs(minVelocity_NW2_L_Hip_Velocity_x))
    pVelocity_NW2_L_Hip_Velocity_x = maxVelocity_NW2_L_Hip_Velocity_x;
 else
    pVelocity_NW2_L_Hip_Velocity_x = minVelocity_NW2_L_Hip_Velocity_x;
 end


 NW2_L_Hip_Velocity_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Hip_Velocity_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Hip_Velocity_y_subset =
NW2_L_Hip_Velocity_x(NW2_L_Hip_Velocity_y_Frame1:NW2_L_Hip_Velocity_y_FrameN)
;
 maxVelocity_NW2_L_Hip_Velocity_y = max(NW2_L_Hip_Velocity_y_subset);
 minVelocity_NW2_L_Hip_Velocity_y = min(NW2_L_Hip_Velocity_y_subset);
 if maxVelocity_NW2_L_Hip_Velocity_y >(abs(minVelocity_NW2_L_Hip_Velocity_y))
    pVelocity_NW2_L_Hip_Velocity_y = maxVelocity_NW2_L_Hip_Velocity_y;
 else
    pVelocity_NW2_L_Hip_Velocity_y = minVelocity_NW2_L_Hip_Velocity_y;
 end


 NW2_L_Hip_Velocity_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_L_Hip_Velocity_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_L_Hip_Velocity_z_subset =
NW2_L_Hip_Velocity_x(NW2_L_Hip_Velocity_z_Frame1:NW2_L_Hip_Velocity_z_FrameN)
;
 maxVelocity_NW2_L_Hip_Velocity_z = max(NW2_L_Hip_Velocity_z_subset);
 minVelocity_NW2_L_Hip_Velocity_z = min(NW2_L_Hip_Velocity_z_subset);
 if maxVelocity_NW2_L_Hip_Velocity_z >(abs(minVelocity_NW2_L_Hip_Velocity_z))
    pVelocity_NW2_L_Hip_Velocity_z = maxVelocity_NW2_L_Hip_Velocity_z;
 else
    pVelocity_NW2_L_Hip_Velocity_z = minVelocity_NW2_L_Hip_Velocity_z;
 end



 %NW3
  NW3_L_Ankle_Velocity_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Ankle_Velocity_x_FrameN = round((NW3_Post_350/.005)+1);
```

```
 NW3_L_Ankle_Velocity_x_subset =
NW3_L_Ankle_Velocity_x(NW3_L_Ankle_Velocity_x_Frame1:NW3_L_Ankle_Velocity_x_F
rameN);
 maxVelocity_NW3_L_Ankle_Velocity_x = max(NW3_L_Ankle_Velocity_x_subset);
 minVelocity_NW3_L_Ankle_Velocity_x = min(NW3_L_Ankle_Velocity_x_subset);
 if maxVelocity_NW3_L_Ankle_Velocity_x >(abs(minVelocity_NW3_L_Ankle_Velocity_x))
    pVelocity_NW3_L_Ankle_Velocity_x = maxVelocity_NW3_L_Ankle_Velocity_x;
 else
    pVelocity_NW3_L_Ankle_Velocity_x = minVelocity_NW3_L_Ankle_Velocity_x;
 end


 NW3_L_Ankle_Velocity_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Ankle_Velocity_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Ankle_Velocity_y_subset =
NW3_L_Ankle_Velocity_y(NW3_L_Ankle_Velocity_y_Frame1:NW3_L_Ankle_Velocity_y_F
rameN);
 maxVelocity_NW3_L_Ankle_Velocity_y = max(NW3_L_Ankle_Velocity_y_subset);
 minVelocity_NW3_L_Ankle_Velocity_y = min(NW3_L_Ankle_Velocity_y_subset);
 if maxVelocity_NW3_L_Ankle_Velocity_y >(abs(minVelocity_NW3_L_Ankle_Velocity_y))
    pVelocity_NW3_L_Ankle_Velocity_y = maxVelocity_NW3_L_Ankle_Velocity_y;
 else
    pVelocity_NW3_L_Ankle_Velocity_y = minVelocity_NW3_L_Ankle_Velocity_y;
 end


 NW3_L_Ankle_Velocity_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Ankle_Velocity_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Ankle_Velocity_z_subset =
NW3_L_Ankle_Velocity_z(NW3_L_Ankle_Velocity_z_Frame1:NW3_L_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_NW3_L_Ankle_Velocity_z = max(NW3_L_Ankle_Velocity_z_subset);
 minVelocity_NW3_L_Ankle_Velocity_z = min(NW3_L_Ankle_Velocity_z_subset);
 if maxVelocity_NW3_L_Ankle_Velocity_z >(abs(minVelocity_NW3_L_Ankle_Velocity_z))
    pVelocity_NW3_L_Ankle_Velocity_z = maxVelocity_NW3_L_Ankle_Velocity_z;
 else
    pVelocity_NW3_L_Ankle_Velocity_z = minVelocity_NW3_L_Ankle_Velocity_z;
 end


 NW3_L_Knee_Velocity_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Knee_Velocity_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Knee_Velocity_x_subset =
NW3_L_Knee_Velocity_x(NW3_L_Knee_Velocity_x_Frame1:NW3_L_Knee_Velocity_x_Fra
meN);
 maxVelocity_NW3_L_Knee_Velocity_x = max(NW3_L_Knee_Velocity_x_subset);
```

```
minVelocity_NW3_L_Knee_Velocity_x = min(NW3_L_Knee_Velocity_x_subset);
if maxVelocity_NW3_L_Knee_Velocity_x >(abs(minVelocity_NW3_L_Knee_Velocity_x))
   pVelocity_NW3_L_Knee_Velocity_x = maxVelocity_NW3_L_Knee_Velocity_x;
else
   pVelocity_NW3_L_Knee_Velocity_x = minVelocity_NW3_L_Knee_Velocity_x;
end


NW3_L_Knee_Velocity_y_Frame1 = round((NW3_HS/.005)+1);
NW3_L_Knee_Velocity_y_FrameN = round((NW3_Post_350/.005)+1);
NW3_L_Knee_Velocity_y_subset =
NW3_L_Knee_Velocity_y(NW3_L_Knee_Velocity_y_Frame1:NW3_L_Knee_Velocity_y_Fra
meN);
maxVelocity_NW3_L_Knee_Velocity_y = max(NW3_L_Knee_Velocity_y_subset);
minVelocity_NW3_L_Knee_Velocity_y = min(NW3_L_Knee_Velocity_y_subset);
if maxVelocity_NW3_L_Knee_Velocity_y >(abs(minVelocity_NW3_L_Knee_Velocity_y))
   pVelocity_NW3_L_Knee_Velocity_y = maxVelocity_NW3_L_Knee_Velocity_y;
else
   pVelocity_NW3_L_Knee_Velocity_y = minVelocity_NW3_L_Knee_Velocity_y;
end


NW3_L_Knee_Velocity_z_Frame1 = round((NW3_HS/.005)+1);
NW3_L_Knee_Velocity_z_FrameN = round((NW3_Post_350/.005)+1);
NW3_L_Knee_Velocity_z_subset =
NW3_L_Knee_Velocity_z(NW3_L_Knee_Velocity_z_Frame1:NW3_L_Knee_Velocity_z_Fra
meN);
maxVelocity_NW3_L_Knee_Velocity_z = max(NW3_L_Knee_Velocity_z_subset);
minVelocity_NW3_L_Knee_Velocity_z = min(NW3_L_Knee_Velocity_z_subset);
if maxVelocity_NW3_L_Knee_Velocity_z >(abs(minVelocity_NW3_L_Knee_Velocity_z))
   pVelocity_NW3_L_Knee_Velocity_z = maxVelocity_NW3_L_Knee_Velocity_z;
else
   pVelocity_NW3_L_Knee_Velocity_z = minVelocity_NW3_L_Knee_Velocity_z;
end


NW3_L_Hip_Velocity_x_Frame1 = round((NW3_HS/.005)+1);
NW3_L_Hip_Velocity_x_FrameN = round((NW3_Post_350/.005)+1);
NW3_L_Hip_Velocity_x_subset =
NW3_L_Hip_Velocity_x(NW3_L_Hip_Velocity_x_Frame1:NW3_L_Hip_Velocity_x_FrameN)
;
maxVelocity_NW3_L_Hip_Velocity_x = max(NW3_L_Hip_Velocity_x_subset);
minVelocity_NW3_L_Hip_Velocity_x = min(NW3_L_Hip_Velocity_x_subset);
if maxVelocity_NW3_L_Hip_Velocity_x >(abs(minVelocity_NW3_L_Hip_Velocity_x))
   pVelocity_NW3_L_Hip_Velocity_x = maxVelocity_NW3_L_Hip_Velocity_x;
else
```

```
    pVelocity_NW3_L_Hip_Velocity_x = minVelocity_NW3_L_Hip_Velocity_x;
 end


 NW3_L_Hip_Velocity_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Hip_Velocity_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Hip_Velocity_y_subset =
NW3_L_Hip_Velocity_x(NW3_L_Hip_Velocity_y_Frame1:NW3_L_Hip_Velocity_y_FrameN)
;
 maxVelocity_NW3_L_Hip_Velocity_y = max(NW3_L_Hip_Velocity_y_subset);
 minVelocity_NW3_L_Hip_Velocity_y = min(NW3_L_Hip_Velocity_y_subset);
 if maxVelocity_NW3_L_Hip_Velocity_y >(abs(minVelocity_NW3_L_Hip_Velocity_y))
    pVelocity_NW3_L_Hip_Velocity_y = maxVelocity_NW3_L_Hip_Velocity_y;
 else
    pVelocity_NW3_L_Hip_Velocity_y = minVelocity_NW3_L_Hip_Velocity_y;
 end


 NW3_L_Hip_Velocity_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_L_Hip_Velocity_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_L_Hip_Velocity_z_subset =
NW3_L_Hip_Velocity_x(NW3_L_Hip_Velocity_z_Frame1:NW3_L_Hip_Velocity_z_FrameN)
;
 maxVelocity_NW3_L_Hip_Velocity_z = max(NW3_L_Hip_Velocity_z_subset);
 minVelocity_NW3_L_Hip_Velocity_z = min(NW3_L_Hip_Velocity_z_subset);
 if maxVelocity_NW3_L_Hip_Velocity_z >(abs(minVelocity_NW3_L_Hip_Velocity_z))
    pVelocity_NW3_L_Hip_Velocity_z = maxVelocity_NW3_L_Hip_Velocity_z;
 else
    pVelocity_NW3_L_Hip_Velocity_z = minVelocity_NW3_L_Hip_Velocity_z;
 end


  %Velocity
 %Right Side
 %ID1
 ID1_R_Ankle_Velocity_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Ankle_Velocity_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Ankle_Velocity_x_subset =
ID1_R_Ankle_Velocity_x(ID1_R_Ankle_Velocity_x_Frame1:ID1_R_Ankle_Velocity_x_Fram
eN);
 maxVelocity_ID1_R_Ankle_Velocity_x = max(ID1_R_Ankle_Velocity_x_subset);
 minVelocity_ID1_R_Ankle_Velocity_x = min(ID1_R_Ankle_Velocity_x_subset);
 if maxVelocity_ID1_R_Ankle_Velocity_x >(abs(minVelocity_ID1_R_Ankle_Velocity_x))
    pVelocity_ID1_R_Ankle_Velocity_x = maxVelocity_ID1_R_Ankle_Velocity_x;
 else
    pVelocity_ID1_R_Ankle_Velocity_x = minVelocity_ID1_R_Ankle_Velocity_x;
```

```
end

ID1_R_Ankle_Velocity_y_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_R_Ankle_Velocity_y_FrameN = round((ID1_Post_350/.005)+1);
ID1_R_Ankle_Velocity_y_subset =
ID1_R_Ankle_Velocity_y(ID1_R_Ankle_Velocity_y_Frame1:ID1_R_Ankle_Velocity_y_Fram
eN);
maxVelocity_ID1_R_Ankle_Velocity_y = max(ID1_R_Ankle_Velocity_y_subset);
minVelocity_ID1_R_Ankle_Velocity_y = min(ID1_R_Ankle_Velocity_y_subset);
if maxVelocity_ID1_R_Ankle_Velocity_y >(abs(minVelocity_ID1_R_Ankle_Velocity_y))
   pVelocity_ID1_R_Ankle_Velocity_y = maxVelocity_ID1_R_Ankle_Velocity_y;
else
   pVelocity_ID1_R_Ankle_Velocity_y = minVelocity_ID1_R_Ankle_Velocity_y;
end


ID1_R_Ankle_Velocity_z_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_R_Ankle_Velocity_z_FrameN = round((ID1_Post_350/.005)+1);
ID1_R_Ankle_Velocity_z_subset =
ID1_R_Ankle_Velocity_z(ID1_R_Ankle_Velocity_z_Frame1:ID1_R_Ankle_Velocity_z_Frame
N);
maxVelocity_ID1_R_Ankle_Velocity_z = max(ID1_R_Ankle_Velocity_z_subset);
minVelocity_ID1_R_Ankle_Velocity_z = min(ID1_R_Ankle_Velocity_z_subset);
if maxVelocity_ID1_R_Ankle_Velocity_z >(abs(minVelocity_ID1_R_Ankle_Velocity_z))
   pVelocity_ID1_R_Ankle_Velocity_z = maxVelocity_ID1_R_Ankle_Velocity_z;
else
   pVelocity_ID1_R_Ankle_Velocity_z = minVelocity_ID1_R_Ankle_Velocity_z;
end


ID1_R_Knee_Velocity_x_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_R_Knee_Velocity_x_FrameN = round((ID1_Post_350/.005)+1);
ID1_R_Knee_Velocity_x_subset =
ID1_R_Knee_Velocity_x(ID1_R_Knee_Velocity_x_Frame1:ID1_R_Knee_Velocity_x_FrameN
);
maxVelocity_ID1_R_Knee_Velocity_x = max(ID1_R_Knee_Velocity_x_subset);
minVelocity_ID1_R_Knee_Velocity_x = min(ID1_R_Knee_Velocity_x_subset);
if maxVelocity_ID1_R_Knee_Velocity_x >(abs(minVelocity_ID1_R_Knee_Velocity_x))
   pVelocity_ID1_R_Knee_Velocity_x = maxVelocity_ID1_R_Knee_Velocity_x;
else
   pVelocity_ID1_R_Knee_Velocity_x = minVelocity_ID1_R_Knee_Velocity_x;
end


ID1_R_Knee_Velocity_y_Frame1 = round((ID1_True_Drop/.005)+1);
ID1_R_Knee_Velocity_y_FrameN = round((ID1_Post_350/.005)+1);
```

```
 ID1_R_Knee_Velocity_y_subset =
ID1_R_Knee_Velocity_y(ID1_R_Knee_Velocity_y_Frame1:ID1_R_Knee_Velocity_y_FrameN
);
 maxVelocity_ID1_R_Knee_Velocity_y = max(ID1_R_Knee_Velocity_y_subset);
 minVelocity_ID1_R_Knee_Velocity_y = min(ID1_R_Knee_Velocity_y_subset);
 if maxVelocity_ID1_R_Knee_Velocity_y >(abs(minVelocity_ID1_R_Knee_Velocity_y))
    pVelocity_ID1_R_Knee_Velocity_y = maxVelocity_ID1_R_Knee_Velocity_y;
 else
    pVelocity_ID1_R_Knee_Velocity_y = minVelocity_ID1_R_Knee_Velocity_y;
 end


 ID1_R_Knee_Velocity_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Knee_Velocity_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Knee_Velocity_z_subset =
ID1_R_Knee_Velocity_z(ID1_R_Knee_Velocity_z_Frame1:ID1_R_Knee_Velocity_z_FrameN)
;
 maxVelocity_ID1_R_Knee_Velocity_z = max(ID1_R_Knee_Velocity_z_subset);
 minVelocity_ID1_R_Knee_Velocity_z = min(ID1_R_Knee_Velocity_z_subset);
 if maxVelocity_ID1_R_Knee_Velocity_z >(abs(minVelocity_ID1_R_Knee_Velocity_z))
    pVelocity_ID1_R_Knee_Velocity_z = maxVelocity_ID1_R_Knee_Velocity_z;
 else
    pVelocity_ID1_R_Knee_Velocity_z = minVelocity_ID1_R_Knee_Velocity_z;
 end


 ID1_R_Hip_Velocity_x_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Hip_Velocity_x_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Hip_Velocity_x_subset =
ID1_R_Hip_Velocity_x(ID1_R_Hip_Velocity_x_Frame1:ID1_R_Hip_Velocity_x_FrameN);
 maxVelocity_ID1_R_Hip_Velocity_x = max(ID1_R_Hip_Velocity_x_subset);
 minVelocity_ID1_R_Hip_Velocity_x = min(ID1_R_Hip_Velocity_x_subset);
 if maxVelocity_ID1_R_Hip_Velocity_x >(abs(minVelocity_ID1_R_Hip_Velocity_x))
    pVelocity_ID1_R_Hip_Velocity_x = maxVelocity_ID1_R_Hip_Velocity_x;
 else
    pVelocity_ID1_R_Hip_Velocity_x = minVelocity_ID1_R_Hip_Velocity_x;
 end


 ID1_R_Hip_Velocity_y_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Hip_Velocity_y_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Hip_Velocity_y_subset =
ID1_R_Hip_Velocity_y(ID1_R_Hip_Velocity_y_Frame1:ID1_R_Hip_Velocity_y_FrameN);
 maxVelocity_ID1_R_Hip_Velocity_y = max(ID1_R_Hip_Velocity_y_subset);
 minVelocity_ID1_R_Hip_Velocity_y = min(ID1_R_Hip_Velocity_y_subset);
 if maxVelocity_ID1_R_Hip_Velocity_y >(abs(minVelocity_ID1_R_Hip_Velocity_y))
```

```
    pVelocity_ID1_R_Hip_Velocity_y = maxVelocity_ID1_R_Hip_Velocity_y;
 else
    pVelocity_ID1_R_Hip_Velocity_y = minVelocity_ID1_R_Hip_Velocity_y;
 end


 ID1_R_Hip_Velocity_z_Frame1 = round((ID1_True_Drop/.005)+1);
 ID1_R_Hip_Velocity_z_FrameN = round((ID1_Post_350/.005)+1);
 ID1_R_Hip_Velocity_z_subset =
ID1_R_Hip_Velocity_z(ID1_R_Hip_Velocity_z_Frame1:ID1_R_Hip_Velocity_z_FrameN);
 maxVelocity_ID1_R_Hip_Velocity_z = max(ID1_R_Hip_Velocity_z_subset);
 minVelocity_ID1_R_Hip_Velocity_z = min(ID1_R_Hip_Velocity_z_subset);
 if maxVelocity_ID1_R_Hip_Velocity_z >(abs(minVelocity_ID1_R_Hip_Velocity_z))
    pVelocity_ID1_R_Hip_Velocity_z = maxVelocity_ID1_R_Hip_Velocity_z;
 else
    pVelocity_ID1_R_Hip_Velocity_z = minVelocity_ID1_R_Hip_Velocity_z;
 end


 % ID2
 ID2_R_Ankle_Velocity_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Ankle_Velocity_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Ankle_Velocity_x_subset =
ID2_R_Ankle_Velocity_x(ID2_R_Ankle_Velocity_x_Frame1:ID2_R_Ankle_Velocity_x_Fram
eN);
 maxVelocity_ID2_R_Ankle_Velocity_x = max(ID2_R_Ankle_Velocity_x_subset);
 minVelocity_ID2_R_Ankle_Velocity_x = min(ID2_R_Ankle_Velocity_x_subset);
 if maxVelocity_ID2_R_Ankle_Velocity_x >(abs(minVelocity_ID2_R_Ankle_Velocity_x))
    pVelocity_ID2_R_Ankle_Velocity_x = maxVelocity_ID2_R_Ankle_Velocity_x;
 else
    pVelocity_ID2_R_Ankle_Velocity_x = minVelocity_ID2_R_Ankle_Velocity_x;
 end


 ID2_R_Ankle_Velocity_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Ankle_Velocity_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Ankle_Velocity_y_subset =
ID2_R_Ankle_Velocity_y(ID2_R_Ankle_Velocity_y_Frame1:ID2_R_Ankle_Velocity_y_Fram
eN);
 maxVelocity_ID2_R_Ankle_Velocity_y = max(ID2_R_Ankle_Velocity_y_subset);
 minVelocity_ID2_R_Ankle_Velocity_y = min(ID2_R_Ankle_Velocity_y_subset);
 if maxVelocity_ID2_R_Ankle_Velocity_y >(abs(minVelocity_ID2_R_Ankle_Velocity_y))
    pVelocity_ID2_R_Ankle_Velocity_y = maxVelocity_ID2_R_Ankle_Velocity_y;
 else
    pVelocity_ID2_R_Ankle_Velocity_y = minVelocity_ID2_R_Ankle_Velocity_y;
 end
```

```
 ID2_R_Ankle_Velocity_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Ankle_Velocity_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Ankle_Velocity_z_subset =
ID2_R_Ankle_Velocity_z(ID2_R_Ankle_Velocity_z_Frame1:ID2_R_Ankle_Velocity_z_Frame
N);
 maxVelocity_ID2_R_Ankle_Velocity_z = max(ID2_R_Ankle_Velocity_z_subset);
 minVelocity_ID2_R_Ankle_Velocity_z = min(ID2_R_Ankle_Velocity_z_subset);
 if maxVelocity_ID2_R_Ankle_Velocity_z >(abs(minVelocity_ID2_R_Ankle_Velocity_z))
    pVelocity_ID2_R_Ankle_Velocity_z = maxVelocity_ID2_R_Ankle_Velocity_z;
 else
    pVelocity_ID2_R_Ankle_Velocity_z = minVelocity_ID2_R_Ankle_Velocity_z;
 end


 ID2_R_Knee_Velocity_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Knee_Velocity_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Knee_Velocity_x_subset =
ID2_R_Knee_Velocity_x(ID2_R_Knee_Velocity_x_Frame1:ID2_R_Knee_Velocity_x_FrameN
);
 maxVelocity_ID2_R_Knee_Velocity_x = max(ID2_R_Knee_Velocity_x_subset);
 minVelocity_ID2_R_Knee_Velocity_x = min(ID2_R_Knee_Velocity_x_subset);
 if maxVelocity_ID2_R_Knee_Velocity_x >(abs(minVelocity_ID2_R_Knee_Velocity_x))
    pVelocity_ID2_R_Knee_Velocity_x = maxVelocity_ID2_R_Knee_Velocity_x;
 else
    pVelocity_ID2_R_Knee_Velocity_x = minVelocity_ID2_R_Knee_Velocity_x;
 end

 ID2_R_Knee_Velocity_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Knee_Velocity_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Knee_Velocity_y_subset =
ID2_R_Knee_Velocity_y(ID2_R_Knee_Velocity_y_Frame1:ID2_R_Knee_Velocity_y_FrameN
);
 maxVelocity_ID2_R_Knee_Velocity_y = max(ID2_R_Knee_Velocity_y_subset);
 minVelocity_ID2_R_Knee_Velocity_y = min(ID2_R_Knee_Velocity_y_subset);
 if maxVelocity_ID2_R_Knee_Velocity_y >(abs(minVelocity_ID2_R_Knee_Velocity_y))
    pVelocity_ID2_R_Knee_Velocity_y = maxVelocity_ID2_R_Knee_Velocity_y;
 else
    pVelocity_ID2_R_Knee_Velocity_y = minVelocity_ID2_R_Knee_Velocity_y;
 end


 ID2_R_Knee_Velocity_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Knee_Velocity_z_FrameN = round((ID2_Post_350/.005)+1);
```

```
 ID2_R_Knee_Velocity_z_subset =
ID2_R_Knee_Velocity_z(ID2_R_Knee_Velocity_z_Frame1:ID2_R_Knee_Velocity_z_FrameN)
;
 maxVelocity_ID2_R_Knee_Velocity_z = max(ID2_R_Knee_Velocity_z_subset);
 minVelocity_ID2_R_Knee_Velocity_z = min(ID2_R_Knee_Velocity_z_subset);
 if maxVelocity_ID2_R_Knee_Velocity_z >(abs(minVelocity_ID2_R_Knee_Velocity_z))
    pVelocity_ID2_R_Knee_Velocity_z = maxVelocity_ID2_R_Knee_Velocity_z;
 else
    pVelocity_ID2_R_Knee_Velocity_z = minVelocity_ID2_R_Knee_Velocity_z;
 end


 ID2_R_Hip_Velocity_x_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Hip_Velocity_x_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Hip_Velocity_x_subset =
ID2_R_Hip_Velocity_x(ID2_R_Hip_Velocity_x_Frame1:ID2_R_Hip_Velocity_x_FrameN);
 maxVelocity_ID2_R_Hip_Velocity_x = max(ID2_R_Hip_Velocity_x_subset);
 minVelocity_ID2_R_Hip_Velocity_x = min(ID2_R_Hip_Velocity_x_subset);
 if maxVelocity_ID2_R_Hip_Velocity_x >(abs(minVelocity_ID2_R_Hip_Velocity_x))
    pVelocity_ID2_R_Hip_Velocity_x = maxVelocity_ID2_R_Hip_Velocity_x;
 else
    pVelocity_ID2_R_Hip_Velocity_x = minVelocity_ID2_R_Hip_Velocity_x;
 end


 ID2_R_Hip_Velocity_y_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Hip_Velocity_y_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Hip_Velocity_y_subset =
ID2_R_Hip_Velocity_y(ID2_R_Hip_Velocity_y_Frame1:ID2_R_Hip_Velocity_y_FrameN);
 maxVelocity_ID2_R_Hip_Velocity_y = max(ID2_R_Hip_Velocity_y_subset);
 minVelocity_ID2_R_Hip_Velocity_y = min(ID2_R_Hip_Velocity_y_subset);
 if maxVelocity_ID2_R_Hip_Velocity_y >(abs(minVelocity_ID2_R_Hip_Velocity_y))
    pVelocity_ID2_R_Hip_Velocity_y = maxVelocity_ID2_R_Hip_Velocity_y;
 else
    pVelocity_ID2_R_Hip_Velocity_y = minVelocity_ID2_R_Hip_Velocity_y;
 end


 ID2_R_Hip_Velocity_z_Frame1 = round((ID2_True_Drop/.005)+1);
 ID2_R_Hip_Velocity_z_FrameN = round((ID2_Post_350/.005)+1);
 ID2_R_Hip_Velocity_z_subset =
ID2_R_Hip_Velocity_z(ID2_R_Hip_Velocity_z_Frame1:ID2_R_Hip_Velocity_z_FrameN);
 maxVelocity_ID2_R_Hip_Velocity_z = max(ID2_R_Hip_Velocity_z_subset);
 minVelocity_ID2_R_Hip_Velocity_z = min(ID2_R_Hip_Velocity_z_subset);
 if maxVelocity_ID2_R_Hip_Velocity_z >(abs(minVelocity_ID2_R_Hip_Velocity_z))
    pVelocity_ID2_R_Hip_Velocity_z = maxVelocity_ID2_R_Hip_Velocity_z;
```

```
else
   pVelocity_ID2_R_Hip_Velocity_z = minVelocity_ID2_R_Hip_Velocity_z;
end


%ID3
ID3_R_Ankle_Velocity_x_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Ankle_Velocity_x_FrameN = round((ID3_Post_350/.005)+1);
ID3_R_Ankle_Velocity_x_subset =
ID3_R_Ankle_Velocity_x(ID3_R_Ankle_Velocity_x_Frame1:ID3_R_Ankle_Velocity_x_Fram
eN);
maxVelocity_ID3_R_Ankle_Velocity_x = max(ID3_R_Ankle_Velocity_x_subset);
minVelocity_ID3_R_Ankle_Velocity_x = min(ID3_R_Ankle_Velocity_x_subset);
if maxVelocity_ID3_R_Ankle_Velocity_x >(abs(minVelocity_ID3_R_Ankle_Velocity_x))
   pVelocity_ID3_R_Ankle_Velocity_x = maxVelocity_ID3_R_Ankle_Velocity_x;
else
   pVelocity_ID3_R_Ankle_Velocity_x = minVelocity_ID3_R_Ankle_Velocity_x;
end


ID3_R_Ankle_Velocity_y_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Ankle_Velocity_y_FrameN = round((ID3_Post_350/.005)+1);
ID3_R_Ankle_Velocity_y_subset =
ID3_R_Ankle_Velocity_y(ID3_R_Ankle_Velocity_y_Frame1:ID3_R_Ankle_Velocity_y_Fram
eN);
maxVelocity_ID3_R_Ankle_Velocity_y = max(ID3_R_Ankle_Velocity_y_subset);
minVelocity_ID3_R_Ankle_Velocity_y = min(ID3_R_Ankle_Velocity_y_subset);
if maxVelocity_ID3_R_Ankle_Velocity_y >(abs(minVelocity_ID3_R_Ankle_Velocity_y))
   pVelocity_ID3_R_Ankle_Velocity_y = maxVelocity_ID3_R_Ankle_Velocity_y;
else
   pVelocity_ID3_R_Ankle_Velocity_y = minVelocity_ID3_R_Ankle_Velocity_y;
end


ID3_R_Ankle_Velocity_z_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Ankle_Velocity_z_FrameN = round((ID3_Post_350/.005)+1);
ID3_R_Ankle_Velocity_z_subset =
ID3_R_Ankle_Velocity_z(ID3_R_Ankle_Velocity_z_Frame1:ID3_R_Ankle_Velocity_z_Frame
N);
maxVelocity_ID3_R_Ankle_Velocity_z = max(ID3_R_Ankle_Velocity_z_subset);
minVelocity_ID3_R_Ankle_Velocity_z = min(ID3_R_Ankle_Velocity_z_subset);
if maxVelocity_ID3_R_Ankle_Velocity_z >(abs(minVelocity_ID3_R_Ankle_Velocity_z))
   pVelocity_ID3_R_Ankle_Velocity_z = maxVelocity_ID3_R_Ankle_Velocity_z;
else
   pVelocity_ID3_R_Ankle_Velocity_z = minVelocity_ID3_R_Ankle_Velocity_z;
end
```

```
ID3_R_Knee_Velocity_x_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Knee_Velocity_x_FrameN = round((ID3_Post_350/.005)+1);
ID3_R_Knee_Velocity_x_subset =
ID3_R_Knee_Velocity_x(ID3_R_Knee_Velocity_x_Frame1:ID3_R_Knee_Velocity_x_FrameN
);
maxVelocity_ID3_R_Knee_Velocity_x = max(ID3_R_Knee_Velocity_x_subset);
minVelocity_ID3_R_Knee_Velocity_x = min(ID3_R_Knee_Velocity_x_subset);
if maxVelocity_ID3_R_Knee_Velocity_x >(abs(minVelocity_ID3_R_Knee_Velocity_x))
   pVelocity_ID3_R_Knee_Velocity_x = maxVelocity_ID3_R_Knee_Velocity_x;
else
   pVelocity_ID3_R_Knee_Velocity_x = minVelocity_ID3_R_Knee_Velocity_x;
end


ID3_R_Knee_Velocity_y_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Knee_Velocity_y_FrameN = round((ID3_Post_350/.005)+1);
ID3_R_Knee_Velocity_y_subset =
ID3_R_Knee_Velocity_y(ID3_R_Knee_Velocity_y_Frame1:ID3_R_Knee_Velocity_y_FrameN
);
maxVelocity_ID3_R_Knee_Velocity_y = max(ID3_R_Knee_Velocity_y_subset);
minVelocity_ID3_R_Knee_Velocity_y = min(ID3_R_Knee_Velocity_y_subset);
if maxVelocity_ID3_R_Knee_Velocity_y >(abs(minVelocity_ID3_R_Knee_Velocity_y))
   pVelocity_ID3_R_Knee_Velocity_y = maxVelocity_ID3_R_Knee_Velocity_y;
else
   pVelocity_ID3_R_Knee_Velocity_y = minVelocity_ID3_R_Knee_Velocity_y;
end


ID3_R_Knee_Velocity_z_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Knee_Velocity_z_FrameN = round((ID3_Post_350/.005)+1);
ID3_R_Knee_Velocity_z_subset =
ID3_R_Knee_Velocity_z(ID3_R_Knee_Velocity_z_Frame1:ID3_R_Knee_Velocity_z_FrameN)
;
maxVelocity_ID3_R_Knee_Velocity_z = max(ID3_R_Knee_Velocity_z_subset);
minVelocity_ID3_R_Knee_Velocity_z = min(ID3_R_Knee_Velocity_z_subset);
if maxVelocity_ID3_R_Knee_Velocity_z >(abs(minVelocity_ID3_R_Knee_Velocity_z))
   pVelocity_ID3_R_Knee_Velocity_z = maxVelocity_ID3_R_Knee_Velocity_z;
else
   pVelocity_ID3_R_Knee_Velocity_z = minVelocity_ID3_R_Knee_Velocity_z;
end


ID3_R_Hip_Velocity_x_Frame1 = round((ID3_True_Drop/.005)+1);
ID3_R_Hip_Velocity_x_FrameN = round((ID3_Post_350/.005)+1);
```

```
 ID3_R_Hip_Velocity_x_subset =
ID3_R_Hip_Velocity_x(ID3_R_Hip_Velocity_x_Frame1:ID3_R_Hip_Velocity_x_FrameN);
 maxVelocity_ID3_R_Hip_Velocity_x = max(ID3_R_Hip_Velocity_x_subset);
 minVelocity_ID3_R_Hip_Velocity_x = min(ID3_R_Hip_Velocity_x_subset);
 if maxVelocity_ID3_R_Hip_Velocity_x >(abs(minVelocity_ID3_R_Hip_Velocity_x))
    pVelocity_ID3_R_Hip_Velocity_x = maxVelocity_ID3_R_Hip_Velocity_x;
 else
    pVelocity_ID3_R_Hip_Velocity_x = minVelocity_ID3_R_Hip_Velocity_x;
 end


 ID3_R_Hip_Velocity_y_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Hip_Velocity_y_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Hip_Velocity_y_subset =
ID3_R_Hip_Velocity_y(ID3_R_Hip_Velocity_y_Frame1:ID3_R_Hip_Velocity_y_FrameN);
 maxVelocity_ID3_R_Hip_Velocity_y = max(ID3_R_Hip_Velocity_y_subset);
 minVelocity_ID3_R_Hip_Velocity_y = min(ID3_R_Hip_Velocity_y_subset);
 if maxVelocity_ID3_R_Hip_Velocity_y >(abs(minVelocity_ID3_R_Hip_Velocity_y))
    pVelocity_ID3_R_Hip_Velocity_y = maxVelocity_ID3_R_Hip_Velocity_y;
 else
    pVelocity_ID3_R_Hip_Velocity_y = minVelocity_ID3_R_Hip_Velocity_y;
 end


 ID3_R_Hip_Velocity_z_Frame1 = round((ID3_True_Drop/.005)+1);
 ID3_R_Hip_Velocity_z_FrameN = round((ID3_Post_350/.005)+1);
 ID3_R_Hip_Velocity_z_subset =
ID3_R_Hip_Velocity_z(ID3_R_Hip_Velocity_z_Frame1:ID3_R_Hip_Velocity_z_FrameN);
 maxVelocity_ID3_R_Hip_Velocity_z = max(ID3_R_Hip_Velocity_z_subset);
 minVelocity_ID3_R_Hip_Velocity_z = min(ID3_R_Hip_Velocity_z_subset);
 if maxVelocity_ID3_R_Hip_Velocity_z >(abs(minVelocity_ID3_R_Hip_Velocity_z))
    pVelocity_ID3_R_Hip_Velocity_z = maxVelocity_ID3_R_Hip_Velocity_z;
 else
    pVelocity_ID3_R_Hip_Velocity_z = minVelocity_ID3_R_Hip_Velocity_z;
 end


 %IPD1
  IPD1_R_Ankle_Velocity_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Ankle_Velocity_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Ankle_Velocity_x_subset =
IPD1_R_Ankle_Velocity_x(IPD1_R_Ankle_Velocity_x_Frame1:IPD1_R_Ankle_Velocity_x_F
rameN);
 maxVelocity_IPD1_R_Ankle_Velocity_x = max(IPD1_R_Ankle_Velocity_x_subset);
 minVelocity_IPD1_R_Ankle_Velocity_x = min(IPD1_R_Ankle_Velocity_x_subset);
```

```
 if maxVelocity_IPD1_R_Ankle_Velocity_x >(abs(minVelocity_IPD1_R_Ankle_Velocity_x))
    pVelocity_IPD1_R_Ankle_Velocity_x = maxVelocity_IPD1_R_Ankle_Velocity_x;
 else
    pVelocity_IPD1_R_Ankle_Velocity_x = minVelocity_IPD1_R_Ankle_Velocity_x;
 end



 IPD1_R_Ankle_Velocity_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Ankle_Velocity_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Ankle_Velocity_y_subset =
IPD1_R_Ankle_Velocity_y(IPD1_R_Ankle_Velocity_y_Frame1:IPD1_R_Ankle_Velocity_y_F
rameN);
 maxVelocity_IPD1_R_Ankle_Velocity_y = max(IPD1_R_Ankle_Velocity_y_subset);
 minVelocity_IPD1_R_Ankle_Velocity_y = min(IPD1_R_Ankle_Velocity_y_subset);
 if maxVelocity_IPD1_R_Ankle_Velocity_y >(abs(minVelocity_IPD1_R_Ankle_Velocity_y))
    pVelocity_IPD1_R_Ankle_Velocity_y = maxVelocity_IPD1_R_Ankle_Velocity_y;
 else
    pVelocity_IPD1_R_Ankle_Velocity_y = minVelocity_IPD1_R_Ankle_Velocity_y;
 end



 IPD1_R_Ankle_Velocity_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Ankle_Velocity_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Ankle_Velocity_z_subset =
IPD1_R_Ankle_Velocity_z(IPD1_R_Ankle_Velocity_z_Frame1:IPD1_R_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_IPD1_R_Ankle_Velocity_z = max(IPD1_R_Ankle_Velocity_z_subset);
 minVelocity_IPD1_R_Ankle_Velocity_z = min(IPD1_R_Ankle_Velocity_z_subset);
 if maxVelocity_IPD1_R_Ankle_Velocity_z >(abs(minVelocity_IPD1_R_Ankle_Velocity_z))
    pVelocity_IPD1_R_Ankle_Velocity_z = maxVelocity_IPD1_R_Ankle_Velocity_z;
 else
    pVelocity_IPD1_R_Ankle_Velocity_z = minVelocity_IPD1_R_Ankle_Velocity_z;
 end



 IPD1_R_Knee_Velocity_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Knee_Velocity_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Knee_Velocity_x_subset =
IPD1_R_Knee_Velocity_x(IPD1_R_Knee_Velocity_x_Frame1:IPD1_R_Knee_Velocity_x_Fra
meN);
 maxVelocity_IPD1_R_Knee_Velocity_x = max(IPD1_R_Knee_Velocity_x_subset);
 minVelocity_IPD1_R_Knee_Velocity_x = min(IPD1_R_Knee_Velocity_x_subset);
 if maxVelocity_IPD1_R_Knee_Velocity_x >(abs(minVelocity_IPD1_R_Knee_Velocity_x))
    pVelocity_IPD1_R_Knee_Velocity_x = maxVelocity_IPD1_R_Knee_Velocity_x;
 else
```

```
    pVelocity_IPD1_R_Knee_Velocity_x = minVelocity_IPD1_R_Knee_Velocity_x;
 end



 IPD1_R_Knee_Velocity_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Knee_Velocity_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Knee_Velocity_y_subset =
IPD1_R_Knee_Velocity_y(IPD1_R_Knee_Velocity_y_Frame1:IPD1_R_Knee_Velocity_y_Fra
meN);
 maxVelocity_IPD1_R_Knee_Velocity_y = max(IPD1_R_Knee_Velocity_y_subset);
 minVelocity_IPD1_R_Knee_Velocity_y = min(IPD1_R_Knee_Velocity_y_subset);
 if maxVelocity_IPD1_R_Knee_Velocity_y >(abs(minVelocity_IPD1_R_Knee_Velocity_y))
    pVelocity_IPD1_R_Knee_Velocity_y = maxVelocity_IPD1_R_Knee_Velocity_y;
 else
    pVelocity_IPD1_R_Knee_Velocity_y = minVelocity_IPD1_R_Knee_Velocity_y;
 end



 IPD1_R_Knee_Velocity_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Knee_Velocity_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Knee_Velocity_z_subset =
IPD1_R_Knee_Velocity_z(IPD1_R_Knee_Velocity_z_Frame1:IPD1_R_Knee_Velocity_z_Fra
meN);
 maxVelocity_IPD1_R_Knee_Velocity_z = max(IPD1_R_Knee_Velocity_z_subset);
 minVelocity_IPD1_R_Knee_Velocity_z = min(IPD1_R_Knee_Velocity_z_subset);
 if maxVelocity_IPD1_R_Knee_Velocity_z >(abs(minVelocity_IPD1_R_Knee_Velocity_z))
    pVelocity_IPD1_R_Knee_Velocity_z = maxVelocity_IPD1_R_Knee_Velocity_z;
 else
    pVelocity_IPD1_R_Knee_Velocity_z = minVelocity_IPD1_R_Knee_Velocity_z;
 end



 IPD1_R_Hip_Velocity_x_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Hip_Velocity_x_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Hip_Velocity_x_subset =
IPD1_R_Hip_Velocity_x(IPD1_R_Hip_Velocity_x_Frame1:IPD1_R_Hip_Velocity_x_FrameN)
;
 maxVelocity_IPD1_R_Hip_Velocity_x = max(IPD1_R_Hip_Velocity_x_subset);
 minVelocity_IPD1_R_Hip_Velocity_x = min(IPD1_R_Hip_Velocity_x_subset);
 if maxVelocity_IPD1_R_Hip_Velocity_x >(abs(minVelocity_IPD1_R_Hip_Velocity_x))
    pVelocity_IPD1_R_Hip_Velocity_x = maxVelocity_IPD1_R_Hip_Velocity_x;
 else
    pVelocity_IPD1_R_Hip_Velocity_x = minVelocity_IPD1_R_Hip_Velocity_x;
 end
```

```
 IPD1_R_Hip_Velocity_y_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Hip_Velocity_y_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Hip_Velocity_y_subset =
IPD1_R_Hip_Velocity_y(IPD1_R_Hip_Velocity_y_Frame1:IPD1_R_Hip_Velocity_y_FrameN)
;
 maxVelocity_IPD1_R_Hip_Velocity_y = max(IPD1_R_Hip_Velocity_y_subset);
 minVelocity_IPD1_R_Hip_Velocity_y = min(IPD1_R_Hip_Velocity_y_subset);
 if maxVelocity_IPD1_R_Hip_Velocity_y >(abs(minVelocity_IPD1_R_Hip_Velocity_y))
    pVelocity_IPD1_R_Hip_Velocity_y = maxVelocity_IPD1_R_Hip_Velocity_y;
 else
    pVelocity_IPD1_R_Hip_Velocity_y = minVelocity_IPD1_R_Hip_Velocity_y;
 end


 IPD1_R_Hip_Velocity_z_Frame1 = round((IPD1_True_Drop/.005)+1);
 IPD1_R_Hip_Velocity_z_FrameN = round((IPD1_Post_350/.005)+1);
 IPD1_R_Hip_Velocity_z_subset =
IPD1_R_Hip_Velocity_z(IPD1_R_Hip_Velocity_z_Frame1:IPD1_R_Hip_Velocity_z_FrameN)
;
 maxVelocity_IPD1_R_Hip_Velocity_z = max(IPD1_R_Hip_Velocity_z_subset);
 minVelocity_IPD1_R_Hip_Velocity_z = min(IPD1_R_Hip_Velocity_z_subset);
 if maxVelocity_IPD1_R_Hip_Velocity_z >(abs(minVelocity_IPD1_R_Hip_Velocity_z))
    pVelocity_IPD1_R_Hip_Velocity_z = maxVelocity_IPD1_R_Hip_Velocity_z;
 else
    pVelocity_IPD1_R_Hip_Velocity_z = minVelocity_IPD1_R_Hip_Velocity_z;
 end

 %IPD2
 IPD2_R_Ankle_Velocity_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Ankle_Velocity_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Ankle_Velocity_x_subset =
IPD2_R_Ankle_Velocity_x(IPD2_R_Ankle_Velocity_x_Frame1:IPD2_R_Ankle_Velocity_x_F
rameN);
 maxVelocity_IPD2_R_Ankle_Velocity_x = max(IPD2_R_Ankle_Velocity_x_subset);
 minVelocity_IPD2_R_Ankle_Velocity_x = min(IPD2_R_Ankle_Velocity_x_subset);
 if maxVelocity_IPD2_R_Ankle_Velocity_x >(abs(minVelocity_IPD2_R_Ankle_Velocity_x))
    pVelocity_IPD2_R_Ankle_Velocity_x = maxVelocity_IPD2_R_Ankle_Velocity_x;
 else
    pVelocity_IPD2_R_Ankle_Velocity_x = minVelocity_IPD2_R_Ankle_Velocity_x;
 end


 IPD2_R_Ankle_Velocity_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Ankle_Velocity_y_FrameN = round((IPD2_Post_350/.005)+1);
```

```
 IPD2_R_Ankle_Velocity_y_subset =
IPD2_R_Ankle_Velocity_y(IPD2_R_Ankle_Velocity_y_Frame1:IPD2_R_Ankle_Velocity_y_F
rameN);
 maxVelocity_IPD2_R_Ankle_Velocity_y = max(IPD2_R_Ankle_Velocity_y_subset);
 minVelocity_IPD2_R_Ankle_Velocity_y = min(IPD2_R_Ankle_Velocity_y_subset);
 if maxVelocity_IPD2_R_Ankle_Velocity_y >(abs(minVelocity_IPD2_R_Ankle_Velocity_y))
    pVelocity_IPD2_R_Ankle_Velocity_y = maxVelocity_IPD2_R_Ankle_Velocity_y;
 else
    pVelocity_IPD2_R_Ankle_Velocity_y = minVelocity_IPD2_R_Ankle_Velocity_y;
 end


 IPD2_R_Ankle_Velocity_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Ankle_Velocity_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Ankle_Velocity_z_subset =
IPD2_R_Ankle_Velocity_z(IPD2_R_Ankle_Velocity_z_Frame1:IPD2_R_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_IPD2_R_Ankle_Velocity_z = max(IPD2_R_Ankle_Velocity_z_subset);
 minVelocity_IPD2_R_Ankle_Velocity_z = min(IPD2_R_Ankle_Velocity_z_subset);
 if maxVelocity_IPD2_R_Ankle_Velocity_z >(abs(minVelocity_IPD2_R_Ankle_Velocity_z))
    pVelocity_IPD2_R_Ankle_Velocity_z = maxVelocity_IPD2_R_Ankle_Velocity_z;
 else
    pVelocity_IPD2_R_Ankle_Velocity_z = minVelocity_IPD2_R_Ankle_Velocity_z;
 end


 IPD2_R_Knee_Velocity_x_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Knee_Velocity_x_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Knee_Velocity_x_subset =
IPD2_R_Knee_Velocity_x(IPD2_R_Knee_Velocity_x_Frame1:IPD2_R_Knee_Velocity_x_Fra
meN);
 maxVelocity_IPD2_R_Knee_Velocity_x = max(IPD2_R_Knee_Velocity_x_subset);
 minVelocity_IPD2_R_Knee_Velocity_x = min(IPD2_R_Knee_Velocity_x_subset);
 if maxVelocity_IPD2_R_Knee_Velocity_x >(abs(minVelocity_IPD2_R_Knee_Velocity_x))
    pVelocity_IPD2_R_Knee_Velocity_x = maxVelocity_IPD2_R_Knee_Velocity_x;
 else
    pVelocity_IPD2_R_Knee_Velocity_x = minVelocity_IPD2_R_Knee_Velocity_x;
 end


 IPD2_R_Knee_Velocity_y_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Knee_Velocity_y_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Knee_Velocity_y_subset =
IPD2_R_Knee_Velocity_y(IPD2_R_Knee_Velocity_y_Frame1:IPD2_R_Knee_Velocity_y_Fra
meN);
 maxVelocity_IPD2_R_Knee_Velocity_y = max(IPD2_R_Knee_Velocity_y_subset);
```

```
minVelocity_IPD2_R_Knee_Velocity_y = min(IPD2_R_Knee_Velocity_y_subset);
if maxVelocity_IPD2_R_Knee_Velocity_y >(abs(minVelocity_IPD2_R_Knee_Velocity_y))
   pVelocity_IPD2_R_Knee_Velocity_y = maxVelocity_IPD2_R_Knee_Velocity_y;
else
   pVelocity_IPD2_R_Knee_Velocity_y = minVelocity_IPD2_R_Knee_Velocity_y;
end


IPD2_R_Knee_Velocity_z_Frame1 = round((IPD2_True_Drop/.005)+1);
IPD2_R_Knee_Velocity_z_FrameN = round((IPD2_Post_350/.005)+1);
IPD2_R_Knee_Velocity_z_subset =
IPD2_R_Knee_Velocity_z(IPD2_R_Knee_Velocity_z_Frame1:IPD2_R_Knee_Velocity_z_Fra
meN);
maxVelocity_IPD2_R_Knee_Velocity_z = max(IPD2_R_Knee_Velocity_z_subset);
minVelocity_IPD2_R_Knee_Velocity_z = min(IPD2_R_Knee_Velocity_z_subset);
if maxVelocity_IPD2_R_Knee_Velocity_z >(abs(minVelocity_IPD2_R_Knee_Velocity_z))
   pVelocity_IPD2_R_Knee_Velocity_z = maxVelocity_IPD2_R_Knee_Velocity_z;
else
   pVelocity_IPD2_R_Knee_Velocity_z = minVelocity_IPD2_R_Knee_Velocity_z;
end


IPD2_R_Hip_Velocity_x_Frame1 = round((IPD2_True_Drop/.005)+1);
IPD2_R_Hip_Velocity_x_FrameN = round((IPD2_Post_350/.005)+1);
IPD2_R_Hip_Velocity_x_subset =
IPD2_R_Hip_Velocity_x(IPD2_R_Hip_Velocity_x_Frame1:IPD2_R_Hip_Velocity_x_FrameN)
;
maxVelocity_IPD2_R_Hip_Velocity_x = max(IPD2_R_Hip_Velocity_x_subset);
minVelocity_IPD2_R_Hip_Velocity_x = min(IPD2_R_Hip_Velocity_x_subset);
if maxVelocity_IPD2_R_Hip_Velocity_x >(abs(minVelocity_IPD2_R_Hip_Velocity_x))
   pVelocity_IPD2_R_Hip_Velocity_x = maxVelocity_IPD2_R_Hip_Velocity_x;
else
   pVelocity_IPD2_R_Hip_Velocity_x = minVelocity_IPD2_R_Hip_Velocity_x;
end


IPD2_R_Hip_Velocity_y_Frame1 = round((IPD2_True_Drop/.005)+1);
IPD2_R_Hip_Velocity_y_FrameN = round((IPD2_Post_350/.005)+1);
IPD2_R_Hip_Velocity_y_subset =
IPD2_R_Hip_Velocity_y(IPD2_R_Hip_Velocity_y_Frame1:IPD2_R_Hip_Velocity_y_FrameN)
;
maxVelocity_IPD2_R_Hip_Velocity_y = max(IPD2_R_Hip_Velocity_y_subset);
minVelocity_IPD2_R_Hip_Velocity_y = min(IPD2_R_Hip_Velocity_y_subset);
if maxVelocity_IPD2_R_Hip_Velocity_y >(abs(minVelocity_IPD2_R_Hip_Velocity_y))
   pVelocity_IPD2_R_Hip_Velocity_y = maxVelocity_IPD2_R_Hip_Velocity_y;
else
```

```
    pVelocity_IPD2_R_Hip_Velocity_y = minVelocity_IPD2_R_Hip_Velocity_y;
 end


 IPD2_R_Hip_Velocity_z_Frame1 = round((IPD2_True_Drop/.005)+1);
 IPD2_R_Hip_Velocity_z_FrameN = round((IPD2_Post_350/.005)+1);
 IPD2_R_Hip_Velocity_z_subset =
IPD2_R_Hip_Velocity_z(IPD2_R_Hip_Velocity_z_Frame1:IPD2_R_Hip_Velocity_z_FrameN)
;
 maxVelocity_IPD2_R_Hip_Velocity_z = max(IPD2_R_Hip_Velocity_z_subset);
 minVelocity_IPD2_R_Hip_Velocity_z = min(IPD2_R_Hip_Velocity_z_subset);
 if maxVelocity_IPD2_R_Hip_Velocity_z >(abs(minVelocity_IPD2_R_Hip_Velocity_z))
    pVelocity_IPD2_R_Hip_Velocity_z = maxVelocity_IPD2_R_Hip_Velocity_z;
 else
    pVelocity_IPD2_R_Hip_Velocity_z = minVelocity_IPD2_R_Hip_Velocity_z;
 end


 %IPD3
 IPD3_R_Ankle_Velocity_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Ankle_Velocity_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Ankle_Velocity_x_subset =
IPD3_R_Ankle_Velocity_x(IPD3_R_Ankle_Velocity_x_Frame1:IPD3_R_Ankle_Velocity_x_F
rameN);
 maxVelocity_IPD3_R_Ankle_Velocity_x = max(IPD3_R_Ankle_Velocity_x_subset);
 minVelocity_IPD3_R_Ankle_Velocity_x = min(IPD3_R_Ankle_Velocity_x_subset);
 if maxVelocity_IPD3_R_Ankle_Velocity_x >(abs(minVelocity_IPD3_R_Ankle_Velocity_x))
    pVelocity_IPD3_R_Ankle_Velocity_x = maxVelocity_IPD3_R_Ankle_Velocity_x;
 else
    pVelocity_IPD3_R_Ankle_Velocity_x = minVelocity_IPD3_R_Ankle_Velocity_x;
 end


 IPD3_R_Ankle_Velocity_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Ankle_Velocity_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Ankle_Velocity_y_subset =
IPD3_R_Ankle_Velocity_y(IPD3_R_Ankle_Velocity_y_Frame1:IPD3_R_Ankle_Velocity_y_F
rameN);
 maxVelocity_IPD3_R_Ankle_Velocity_y = max(IPD3_R_Ankle_Velocity_y_subset);
 minVelocity_IPD3_R_Ankle_Velocity_y = min(IPD3_R_Ankle_Velocity_y_subset);
 if maxVelocity_IPD3_R_Ankle_Velocity_y >(abs(minVelocity_IPD3_R_Ankle_Velocity_y))
    pVelocity_IPD3_R_Ankle_Velocity_y = maxVelocity_IPD3_R_Ankle_Velocity_y;
 else
    pVelocity_IPD3_R_Ankle_Velocity_y = minVelocity_IPD3_R_Ankle_Velocity_y;
 end
```

```
 IPD3_R_Ankle_Velocity_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Ankle_Velocity_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Ankle_Velocity_z_subset =
IPD3_R_Ankle_Velocity_z(IPD3_R_Ankle_Velocity_z_Frame1:IPD3_R_Ankle_Velocity_z_Fr
ameN);
 maxVelocity_IPD3_R_Ankle_Velocity_z = max(IPD3_R_Ankle_Velocity_z_subset);
 minVelocity_IPD3_R_Ankle_Velocity_z = min(IPD3_R_Ankle_Velocity_z_subset);
 if maxVelocity_IPD3_R_Ankle_Velocity_z >(abs(minVelocity_IPD3_R_Ankle_Velocity_z))
    pVelocity_IPD3_R_Ankle_Velocity_z = maxVelocity_IPD3_R_Ankle_Velocity_z;
 else
    pVelocity_IPD3_R_Ankle_Velocity_z = minVelocity_IPD3_R_Ankle_Velocity_z;
 end


 IPD3_R_Knee_Velocity_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Knee_Velocity_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Knee_Velocity_x_subset =
IPD3_R_Knee_Velocity_x(IPD3_R_Knee_Velocity_x_Frame1:IPD3_R_Knee_Velocity_x_Fra
meN);
 maxVelocity_IPD3_R_Knee_Velocity_x = max(IPD3_R_Knee_Velocity_x_subset);
 minVelocity_IPD3_R_Knee_Velocity_x = min(IPD3_R_Knee_Velocity_x_subset);
 if maxVelocity_IPD3_R_Knee_Velocity_x >(abs(minVelocity_IPD3_R_Knee_Velocity_x))
    pVelocity_IPD3_R_Knee_Velocity_x = maxVelocity_IPD3_R_Knee_Velocity_x;
 else
    pVelocity_IPD3_R_Knee_Velocity_x = minVelocity_IPD3_R_Knee_Velocity_x;
 end


 IPD3_R_Knee_Velocity_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Knee_Velocity_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Knee_Velocity_y_subset =
IPD3_R_Knee_Velocity_y(IPD3_R_Knee_Velocity_y_Frame1:IPD3_R_Knee_Velocity_y_Fra
meN);
 maxVelocity_IPD3_R_Knee_Velocity_y = max(IPD3_R_Knee_Velocity_y_subset);
 minVelocity_IPD3_R_Knee_Velocity_y = min(IPD3_R_Knee_Velocity_y_subset);
 if maxVelocity_IPD3_R_Knee_Velocity_y >(abs(minVelocity_IPD3_R_Knee_Velocity_y))
    pVelocity_IPD3_R_Knee_Velocity_y = maxVelocity_IPD3_R_Knee_Velocity_y;
 else
    pVelocity_IPD3_R_Knee_Velocity_y = minVelocity_IPD3_R_Knee_Velocity_y;
 end


 IPD3_R_Knee_Velocity_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Knee_Velocity_z_FrameN = round((IPD3_Post_350/.005)+1);
```

```
 IPD3_R_Knee_Velocity_z_subset =
IPD3_R_Knee_Velocity_z(IPD3_R_Knee_Velocity_z_Frame1:IPD3_R_Knee_Velocity_z_Fra
meN);
 maxVelocity_IPD3_R_Knee_Velocity_z = max(IPD3_R_Knee_Velocity_z_subset);
 minVelocity_IPD3_R_Knee_Velocity_z = min(IPD3_R_Knee_Velocity_z_subset);
 if maxVelocity_IPD3_R_Knee_Velocity_z >(abs(minVelocity_IPD3_R_Knee_Velocity_z))
    pVelocity_IPD3_R_Knee_Velocity_z = maxVelocity_IPD3_R_Knee_Velocity_z;
 else
    pVelocity_IPD3_R_Knee_Velocity_z = minVelocity_IPD3_R_Knee_Velocity_z;
 end


 IPD3_R_Hip_Velocity_x_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Hip_Velocity_x_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Hip_Velocity_x_subset =
IPD3_R_Hip_Velocity_x(IPD3_R_Hip_Velocity_x_Frame1:IPD3_R_Hip_Velocity_x_FrameN)
;
 maxVelocity_IPD3_R_Hip_Velocity_x = max(IPD3_R_Hip_Velocity_x_subset);
 minVelocity_IPD3_R_Hip_Velocity_x = min(IPD3_R_Hip_Velocity_x_subset);
 if maxVelocity_IPD3_R_Hip_Velocity_x >(abs(minVelocity_IPD3_R_Hip_Velocity_x))
    pVelocity_IPD3_R_Hip_Velocity_x = maxVelocity_IPD3_R_Hip_Velocity_x;
 else
    pVelocity_IPD3_R_Hip_Velocity_x = minVelocity_IPD3_R_Hip_Velocity_x;
 end


 IPD3_R_Hip_Velocity_y_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Hip_Velocity_y_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Hip_Velocity_y_subset =
IPD3_R_Hip_Velocity_y(IPD3_R_Hip_Velocity_y_Frame1:IPD3_R_Hip_Velocity_y_FrameN)
;
 maxVelocity_IPD3_R_Hip_Velocity_y = max(IPD3_R_Hip_Velocity_y_subset);
 minVelocity_IPD3_R_Hip_Velocity_y = min(IPD3_R_Hip_Velocity_y_subset);
 if maxVelocity_IPD3_R_Hip_Velocity_y >(abs(minVelocity_IPD3_R_Hip_Velocity_y))
    pVelocity_IPD3_R_Hip_Velocity_y = maxVelocity_IPD3_R_Hip_Velocity_y;
 else
    pVelocity_IPD3_R_Hip_Velocity_y = minVelocity_IPD3_R_Hip_Velocity_y;
 end


 IPD3_R_Hip_Velocity_z_Frame1 = round((IPD3_True_Drop/.005)+1);
 IPD3_R_Hip_Velocity_z_FrameN = round((IPD3_Post_350/.005)+1);
 IPD3_R_Hip_Velocity_z_subset =
IPD3_R_Hip_Velocity_z(IPD3_R_Hip_Velocity_z_Frame1:IPD3_R_Hip_Velocity_z_FrameN)
;
 maxVelocity_IPD3_R_Hip_Velocity_z = max(IPD3_R_Hip_Velocity_z_subset);
```

```
minVelocity_IPD3_R_Hip_Velocity_z = min(IPD3_R_Hip_Velocity_z_subset);
if maxVelocity_IPD3_R_Hip_Velocity_z >(abs(minVelocity_IPD3_R_Hip_Velocity_z))
   pVelocity_IPD3_R_Hip_Velocity_z = maxVelocity_IPD3_R_Hip_Velocity_z;
else
   pVelocity_IPD3_R_Hip_Velocity_z = minVelocity_IPD3_R_Hip_Velocity_z;
end

%NW1
NW1_R_Ankle_Velocity_x_Frame1 = round((NW1_HS/.005)+1);
NW1_R_Ankle_Velocity_x_FrameN = round((NW1_Post_350/.005)+1);
NW1_R_Ankle_Velocity_x_subset =
NW1_R_Ankle_Velocity_x(NW1_R_Ankle_Velocity_x_Frame1:NW1_R_Ankle_Velocity_x_F
rameN);
maxVelocity_NW1_R_Ankle_Velocity_x = max(NW1_R_Ankle_Velocity_x_subset);
minVelocity_NW1_R_Ankle_Velocity_x = min(NW1_R_Ankle_Velocity_x_subset);
if maxVelocity_NW1_R_Ankle_Velocity_x >(abs(minVelocity_NW1_R_Ankle_Velocity_x))
   pVelocity_NW1_R_Ankle_Velocity_x = maxVelocity_NW1_R_Ankle_Velocity_x;
else
   pVelocity_NW1_R_Ankle_Velocity_x = minVelocity_NW1_R_Ankle_Velocity_x;
end


NW1_R_Ankle_Velocity_y_Frame1 = round((NW1_HS/.005)+1);
NW1_R_Ankle_Velocity_y_FrameN = round((NW1_Post_350/.005)+1);
NW1_R_Ankle_Velocity_y_subset =
NW1_R_Ankle_Velocity_y(NW1_R_Ankle_Velocity_y_Frame1:NW1_R_Ankle_Velocity_y_F
rameN);
maxVelocity_NW1_R_Ankle_Velocity_y = max(NW1_R_Ankle_Velocity_y_subset);
minVelocity_NW1_R_Ankle_Velocity_y = min(NW1_R_Ankle_Velocity_y_subset);
if maxVelocity_NW1_R_Ankle_Velocity_y >(abs(minVelocity_NW1_R_Ankle_Velocity_y))
   pVelocity_NW1_R_Ankle_Velocity_y = maxVelocity_NW1_R_Ankle_Velocity_y;
else
   pVelocity_NW1_R_Ankle_Velocity_y = minVelocity_NW1_R_Ankle_Velocity_y;
end


NW1_R_Ankle_Velocity_z_Frame1 = round((NW1_HS/.005)+1);
NW1_R_Ankle_Velocity_z_FrameN = round((NW1_Post_350/.005)+1);
NW1_R_Ankle_Velocity_z_subset =
NW1_R_Ankle_Velocity_z(NW1_R_Ankle_Velocity_z_Frame1:NW1_R_Ankle_Velocity_z_F
rameN);
maxVelocity_NW1_R_Ankle_Velocity_z = max(NW1_R_Ankle_Velocity_z_subset);
minVelocity_NW1_R_Ankle_Velocity_z = min(NW1_R_Ankle_Velocity_z_subset);
if maxVelocity_NW1_R_Ankle_Velocity_z >(abs(minVelocity_NW1_R_Ankle_Velocity_z))
   pVelocity_NW1_R_Ankle_Velocity_z = maxVelocity_NW1_R_Ankle_Velocity_z;
else
```

```
    pVelocity_NW1_R_Ankle_Velocity_z = minVelocity_NW1_R_Ankle_Velocity_z;
 end


 NW1_R_Knee_Velocity_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Knee_Velocity_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Knee_Velocity_x_subset =
NW1_R_Knee_Velocity_x(NW1_R_Knee_Velocity_x_Frame1:NW1_R_Knee_Velocity_x_Fra
meN);
 maxVelocity_NW1_R_Knee_Velocity_x = max(NW1_R_Knee_Velocity_x_subset);
 minVelocity_NW1_R_Knee_Velocity_x = min(NW1_R_Knee_Velocity_x_subset);
 if maxVelocity_NW1_R_Knee_Velocity_x >(abs(minVelocity_NW1_R_Knee_Velocity_x))
    pVelocity_NW1_R_Knee_Velocity_x = maxVelocity_NW1_R_Knee_Velocity_x;
 else
    pVelocity_NW1_R_Knee_Velocity_x = minVelocity_NW1_R_Knee_Velocity_x;
 end


 NW1_R_Knee_Velocity_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Knee_Velocity_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Knee_Velocity_y_subset =
NW1_R_Knee_Velocity_y(NW1_R_Knee_Velocity_y_Frame1:NW1_R_Knee_Velocity_y_Fra
meN);
 maxVelocity_NW1_R_Knee_Velocity_y = max(NW1_R_Knee_Velocity_y_subset);
 minVelocity_NW1_R_Knee_Velocity_y = min(NW1_R_Knee_Velocity_y_subset);
 if maxVelocity_NW1_R_Knee_Velocity_y >(abs(minVelocity_NW1_R_Knee_Velocity_y))
    pVelocity_NW1_R_Knee_Velocity_y = maxVelocity_NW1_R_Knee_Velocity_y;
 else
    pVelocity_NW1_R_Knee_Velocity_y = minVelocity_NW1_R_Knee_Velocity_y;
 end


 NW1_R_Knee_Velocity_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Knee_Velocity_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Knee_Velocity_z_subset =
NW1_R_Knee_Velocity_z(NW1_R_Knee_Velocity_z_Frame1:NW1_R_Knee_Velocity_z_Fra
meN);
 maxVelocity_NW1_R_Knee_Velocity_z = max(NW1_R_Knee_Velocity_z_subset);
 minVelocity_NW1_R_Knee_Velocity_z = min(NW1_R_Knee_Velocity_z_subset);
 if maxVelocity_NW1_R_Knee_Velocity_z >(abs(minVelocity_NW1_R_Knee_Velocity_z))
    pVelocity_NW1_R_Knee_Velocity_z = maxVelocity_NW1_R_Knee_Velocity_z;
 else
    pVelocity_NW1_R_Knee_Velocity_z = minVelocity_NW1_R_Knee_Velocity_z;
 end
```

```
 NW1_R_Hip_Velocity_x_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Hip_Velocity_x_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Hip_Velocity_x_subset =
NW1_R_Hip_Velocity_x(NW1_R_Hip_Velocity_x_Frame1:NW1_R_Hip_Velocity_x_FrameN
);
 maxVelocity_NW1_R_Hip_Velocity_x = max(NW1_R_Hip_Velocity_x_subset);
 minVelocity_NW1_R_Hip_Velocity_x = min(NW1_R_Hip_Velocity_x_subset);
 if maxVelocity_NW1_R_Hip_Velocity_x >(abs(minVelocity_NW1_R_Hip_Velocity_x))
    pVelocity_NW1_R_Hip_Velocity_x = maxVelocity_NW1_R_Hip_Velocity_x;
 else
    pVelocity_NW1_R_Hip_Velocity_x = minVelocity_NW1_R_Hip_Velocity_x;
 end


 NW1_R_Hip_Velocity_y_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Hip_Velocity_y_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Hip_Velocity_y_subset =
NW1_R_Hip_Velocity_x(NW1_R_Hip_Velocity_y_Frame1:NW1_R_Hip_Velocity_y_FrameN
);
 maxVelocity_NW1_R_Hip_Velocity_y = max(NW1_R_Hip_Velocity_y_subset);
 minVelocity_NW1_R_Hip_Velocity_y = min(NW1_R_Hip_Velocity_y_subset);
 if maxVelocity_NW1_R_Hip_Velocity_y >(abs(minVelocity_NW1_R_Hip_Velocity_y))
    pVelocity_NW1_R_Hip_Velocity_y = maxVelocity_NW1_R_Hip_Velocity_y;
 else
    pVelocity_NW1_R_Hip_Velocity_y = minVelocity_NW1_R_Hip_Velocity_y;
 end


 NW1_R_Hip_Velocity_z_Frame1 = round((NW1_HS/.005)+1);
 NW1_R_Hip_Velocity_z_FrameN = round((NW1_Post_350/.005)+1);
 NW1_R_Hip_Velocity_z_subset =
NW1_R_Hip_Velocity_x(NW1_R_Hip_Velocity_z_Frame1:NW1_R_Hip_Velocity_z_FrameN)
;
 maxVelocity_NW1_R_Hip_Velocity_z = max(NW1_R_Hip_Velocity_z_subset);
 minVelocity_NW1_R_Hip_Velocity_z = min(NW1_R_Hip_Velocity_z_subset);
 if maxVelocity_NW1_R_Hip_Velocity_z >(abs(minVelocity_NW1_R_Hip_Velocity_z))
    pVelocity_NW1_R_Hip_Velocity_z = maxVelocity_NW1_R_Hip_Velocity_z;
 else
    pVelocity_NW1_R_Hip_Velocity_z = minVelocity_NW1_R_Hip_Velocity_z;
 end


 %NW2
 NW2_R_Ankle_Velocity_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Ankle_Velocity_x_FrameN = round((NW2_Post_350/.005)+1);
```

```
 NW2_R_Ankle_Velocity_x_subset =
NW2_R_Ankle_Velocity_x(NW2_R_Ankle_Velocity_x_Frame1:NW2_R_Ankle_Velocity_x_F
rameN);
 maxVelocity_NW2_R_Ankle_Velocity_x = max(NW2_R_Ankle_Velocity_x_subset);
 minVelocity_NW2_R_Ankle_Velocity_x = min(NW2_R_Ankle_Velocity_x_subset);
 if maxVelocity_NW2_R_Ankle_Velocity_x >(abs(minVelocity_NW2_R_Ankle_Velocity_x))
    pVelocity_NW2_R_Ankle_Velocity_x = maxVelocity_NW2_R_Ankle_Velocity_x;
 else
    pVelocity_NW2_R_Ankle_Velocity_x = minVelocity_NW2_R_Ankle_Velocity_x;
 end


 NW2_R_Ankle_Velocity_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Ankle_Velocity_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Ankle_Velocity_y_subset =
NW2_R_Ankle_Velocity_y(NW2_R_Ankle_Velocity_y_Frame1:NW2_R_Ankle_Velocity_y_F
rameN);
 maxVelocity_NW2_R_Ankle_Velocity_y = max(NW2_R_Ankle_Velocity_y_subset);
 minVelocity_NW2_R_Ankle_Velocity_y = min(NW2_R_Ankle_Velocity_y_subset);
 if maxVelocity_NW2_R_Ankle_Velocity_y >(abs(minVelocity_NW2_R_Ankle_Velocity_y))
    pVelocity_NW2_R_Ankle_Velocity_y = maxVelocity_NW2_R_Ankle_Velocity_y;
 else
    pVelocity_NW2_R_Ankle_Velocity_y = minVelocity_NW2_R_Ankle_Velocity_y;
 end


 NW2_R_Ankle_Velocity_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Ankle_Velocity_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Ankle_Velocity_z_subset =
NW2_R_Ankle_Velocity_z(NW2_R_Ankle_Velocity_z_Frame1:NW2_R_Ankle_Velocity_z_F
rameN);
 maxVelocity_NW2_R_Ankle_Velocity_z = max(NW2_R_Ankle_Velocity_z_subset);
 minVelocity_NW2_R_Ankle_Velocity_z = min(NW2_R_Ankle_Velocity_z_subset);
 if maxVelocity_NW2_R_Ankle_Velocity_z >(abs(minVelocity_NW2_R_Ankle_Velocity_z))
    pVelocity_NW2_R_Ankle_Velocity_z = maxVelocity_NW2_R_Ankle_Velocity_z;
 else
    pVelocity_NW2_R_Ankle_Velocity_z = minVelocity_NW2_R_Ankle_Velocity_z;
 end


 NW2_R_Knee_Velocity_x_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Knee_Velocity_x_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Knee_Velocity_x_subset =
NW2_R_Knee_Velocity_x(NW2_R_Knee_Velocity_x_Frame1:NW2_R_Knee_Velocity_x_Fra
meN);
 maxVelocity_NW2_R_Knee_Velocity_x = max(NW2_R_Knee_Velocity_x_subset);
```

```
minVelocity_NW2_R_Knee_Velocity_x = min(NW2_R_Knee_Velocity_x_subset);
if maxVelocity_NW2_R_Knee_Velocity_x >(abs(minVelocity_NW2_R_Knee_Velocity_x))
   pVelocity_NW2_R_Knee_Velocity_x = maxVelocity_NW2_R_Knee_Velocity_x;
else
   pVelocity_NW2_R_Knee_Velocity_x = minVelocity_NW2_R_Knee_Velocity_x;
end


NW2_R_Knee_Velocity_y_Frame1 = round((NW2_HS/.005)+1);
NW2_R_Knee_Velocity_y_FrameN = round((NW2_Post_350/.005)+1);
NW2_R_Knee_Velocity_y_subset =
NW2_R_Knee_Velocity_y(NW2_R_Knee_Velocity_y_Frame1:NW2_R_Knee_Velocity_y_Fra
meN);
maxVelocity_NW2_R_Knee_Velocity_y = max(NW2_R_Knee_Velocity_y_subset);
minVelocity_NW2_R_Knee_Velocity_y = min(NW2_R_Knee_Velocity_y_subset);
if maxVelocity_NW2_R_Knee_Velocity_y >(abs(minVelocity_NW2_R_Knee_Velocity_y))
   pVelocity_NW2_R_Knee_Velocity_y = maxVelocity_NW2_R_Knee_Velocity_y;
else
   pVelocity_NW2_R_Knee_Velocity_y = minVelocity_NW2_R_Knee_Velocity_y;
end


NW2_R_Knee_Velocity_z_Frame1 = round((NW2_HS/.005)+1);
NW2_R_Knee_Velocity_z_FrameN = round((NW2_Post_350/.005)+1);
NW2_R_Knee_Velocity_z_subset =
NW2_R_Knee_Velocity_z(NW2_R_Knee_Velocity_z_Frame1:NW2_R_Knee_Velocity_z_Fra
meN);
maxVelocity_NW2_R_Knee_Velocity_z = max(NW2_R_Knee_Velocity_z_subset);
minVelocity_NW2_R_Knee_Velocity_z = min(NW2_R_Knee_Velocity_z_subset);
if maxVelocity_NW2_R_Knee_Velocity_z >(abs(minVelocity_NW2_R_Knee_Velocity_z))
   pVelocity_NW2_R_Knee_Velocity_z = maxVelocity_NW2_R_Knee_Velocity_z;
else
   pVelocity_NW2_R_Knee_Velocity_z = minVelocity_NW2_R_Knee_Velocity_z;
end


NW2_R_Hip_Velocity_x_Frame1 = round((NW2_HS/.005)+1);
NW2_R_Hip_Velocity_x_FrameN = round((NW2_Post_350/.005)+1);
NW2_R_Hip_Velocity_x_subset =
NW2_R_Hip_Velocity_x(NW2_R_Hip_Velocity_x_Frame1:NW2_R_Hip_Velocity_x_FrameN
);
maxVelocity_NW2_R_Hip_Velocity_x = max(NW2_R_Hip_Velocity_x_subset);
minVelocity_NW2_R_Hip_Velocity_x = min(NW2_R_Hip_Velocity_x_subset);
if maxVelocity_NW2_R_Hip_Velocity_x >(abs(minVelocity_NW2_R_Hip_Velocity_x))
   pVelocity_NW2_R_Hip_Velocity_x = maxVelocity_NW2_R_Hip_Velocity_x;
else
```

```
    pVelocity_NW2_R_Hip_Velocity_x = minVelocity_NW2_R_Hip_Velocity_x;
 end


 NW2_R_Hip_Velocity_y_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Hip_Velocity_y_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Hip_Velocity_y_subset =
NW2_R_Hip_Velocity_x(NW2_R_Hip_Velocity_y_Frame1:NW2_R_Hip_Velocity_y_FrameN
);
 maxVelocity_NW2_R_Hip_Velocity_y = max(NW2_R_Hip_Velocity_y_subset);
 minVelocity_NW2_R_Hip_Velocity_y = min(NW2_R_Hip_Velocity_y_subset);
 if maxVelocity_NW2_R_Hip_Velocity_y >(abs(minVelocity_NW2_R_Hip_Velocity_y))
    pVelocity_NW2_R_Hip_Velocity_y = maxVelocity_NW2_R_Hip_Velocity_y;
 else
    pVelocity_NW2_R_Hip_Velocity_y = minVelocity_NW2_R_Hip_Velocity_y;
 end


 NW2_R_Hip_Velocity_z_Frame1 = round((NW2_HS/.005)+1);
 NW2_R_Hip_Velocity_z_FrameN = round((NW2_Post_350/.005)+1);
 NW2_R_Hip_Velocity_z_subset =
NW2_R_Hip_Velocity_x(NW2_R_Hip_Velocity_z_Frame1:NW2_R_Hip_Velocity_z_FrameN)
;
 maxVelocity_NW2_R_Hip_Velocity_z = max(NW2_R_Hip_Velocity_z_subset);
 minVelocity_NW2_R_Hip_Velocity_z = min(NW2_R_Hip_Velocity_z_subset);
 if maxVelocity_NW2_R_Hip_Velocity_z >(abs(minVelocity_NW2_R_Hip_Velocity_z))
    pVelocity_NW2_R_Hip_Velocity_z = maxVelocity_NW2_R_Hip_Velocity_z;
 else
    pVelocity_NW2_R_Hip_Velocity_z = minVelocity_NW2_R_Hip_Velocity_z;
 end



 %NW3
  NW3_R_Ankle_Velocity_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Ankle_Velocity_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Ankle_Velocity_x_subset =
NW3_R_Ankle_Velocity_x(NW3_R_Ankle_Velocity_x_Frame1:NW3_R_Ankle_Velocity_x_F
rameN);
 maxVelocity_NW3_R_Ankle_Velocity_x = max(NW3_R_Ankle_Velocity_x_subset);
 minVelocity_NW3_R_Ankle_Velocity_x = min(NW3_R_Ankle_Velocity_x_subset);
 if maxVelocity_NW3_R_Ankle_Velocity_x >(abs(minVelocity_NW3_R_Ankle_Velocity_x))
    pVelocity_NW3_R_Ankle_Velocity_x = maxVelocity_NW3_R_Ankle_Velocity_x;
 else
    pVelocity_NW3_R_Ankle_Velocity_x = minVelocity_NW3_R_Ankle_Velocity_x;
 end
```

```
 NW3_R_Ankle_Velocity_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Ankle_Velocity_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Ankle_Velocity_y_subset =
NW3_R_Ankle_Velocity_y(NW3_R_Ankle_Velocity_y_Frame1:NW3_R_Ankle_Velocity_y_F
rameN);
 maxVelocity_NW3_R_Ankle_Velocity_y = max(NW3_R_Ankle_Velocity_y_subset);
 minVelocity_NW3_R_Ankle_Velocity_y = min(NW3_R_Ankle_Velocity_y_subset);
 if maxVelocity_NW3_R_Ankle_Velocity_y >(abs(minVelocity_NW3_R_Ankle_Velocity_y))
    pVelocity_NW3_R_Ankle_Velocity_y = maxVelocity_NW3_R_Ankle_Velocity_y;
 else
    pVelocity_NW3_R_Ankle_Velocity_y = minVelocity_NW3_R_Ankle_Velocity_y;
 end


 NW3_R_Ankle_Velocity_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Ankle_Velocity_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Ankle_Velocity_z_subset =
NW3_R_Ankle_Velocity_z(NW3_R_Ankle_Velocity_z_Frame1:NW3_R_Ankle_Velocity_z_F
rameN);
 maxVelocity_NW3_R_Ankle_Velocity_z = max(NW3_R_Ankle_Velocity_z_subset);
 minVelocity_NW3_R_Ankle_Velocity_z = min(NW3_R_Ankle_Velocity_z_subset);
 if maxVelocity_NW3_R_Ankle_Velocity_z >(abs(minVelocity_NW3_R_Ankle_Velocity_z))
    pVelocity_NW3_R_Ankle_Velocity_z = maxVelocity_NW3_R_Ankle_Velocity_z;
 else
    pVelocity_NW3_R_Ankle_Velocity_z = minVelocity_NW3_R_Ankle_Velocity_z;
 end


 NW3_R_Knee_Velocity_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Knee_Velocity_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Knee_Velocity_x_subset =
NW3_R_Knee_Velocity_x(NW3_R_Knee_Velocity_x_Frame1:NW3_R_Knee_Velocity_x_Fra
meN);
 maxVelocity_NW3_R_Knee_Velocity_x = max(NW3_R_Knee_Velocity_x_subset);
 minVelocity_NW3_R_Knee_Velocity_x = min(NW3_R_Knee_Velocity_x_subset);
 if maxVelocity_NW3_R_Knee_Velocity_x >(abs(minVelocity_NW3_R_Knee_Velocity_x))
    pVelocity_NW3_R_Knee_Velocity_x = maxVelocity_NW3_R_Knee_Velocity_x;
 else
    pVelocity_NW3_R_Knee_Velocity_x = minVelocity_NW3_R_Knee_Velocity_x;
 end


 NW3_R_Knee_Velocity_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Knee_Velocity_y_FrameN = round((NW3_Post_350/.005)+1);
```

```
 NW3_R_Knee_Velocity_y_subset =
NW3_R_Knee_Velocity_y(NW3_R_Knee_Velocity_y_Frame1:NW3_R_Knee_Velocity_y_Fra
meN);
 maxVelocity_NW3_R_Knee_Velocity_y = max(NW3_R_Knee_Velocity_y_subset);
 minVelocity_NW3_R_Knee_Velocity_y = min(NW3_R_Knee_Velocity_y_subset);
 if maxVelocity_NW3_R_Knee_Velocity_y >(abs(minVelocity_NW3_R_Knee_Velocity_y))
    pVelocity_NW3_R_Knee_Velocity_y = maxVelocity_NW3_R_Knee_Velocity_y;
 else
    pVelocity_NW3_R_Knee_Velocity_y = minVelocity_NW3_R_Knee_Velocity_y;
 end


 NW3_R_Knee_Velocity_z_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Knee_Velocity_z_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Knee_Velocity_z_subset =
NW3_R_Knee_Velocity_z(NW3_R_Knee_Velocity_z_Frame1:NW3_R_Knee_Velocity_z_Fra
meN);
 maxVelocity_NW3_R_Knee_Velocity_z = max(NW3_R_Knee_Velocity_z_subset);
 minVelocity_NW3_R_Knee_Velocity_z = min(NW3_R_Knee_Velocity_z_subset);
 if maxVelocity_NW3_R_Knee_Velocity_z >(abs(minVelocity_NW3_R_Knee_Velocity_z))
    pVelocity_NW3_R_Knee_Velocity_z = maxVelocity_NW3_R_Knee_Velocity_z;
 else
    pVelocity_NW3_R_Knee_Velocity_z = minVelocity_NW3_R_Knee_Velocity_z;
 end


 NW3_R_Hip_Velocity_x_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Hip_Velocity_x_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Hip_Velocity_x_subset =
NW3_R_Hip_Velocity_x(NW3_R_Hip_Velocity_x_Frame1:NW3_R_Hip_Velocity_x_FrameN
);
 maxVelocity_NW3_R_Hip_Velocity_x = max(NW3_R_Hip_Velocity_x_subset);
 minVelocity_NW3_R_Hip_Velocity_x = min(NW3_R_Hip_Velocity_x_subset);
 if maxVelocity_NW3_R_Hip_Velocity_x >(abs(minVelocity_NW3_R_Hip_Velocity_x))
    pVelocity_NW3_R_Hip_Velocity_x = maxVelocity_NW3_R_Hip_Velocity_x;
 else
    pVelocity_NW3_R_Hip_Velocity_x = minVelocity_NW3_R_Hip_Velocity_x;
 end


 NW3_R_Hip_Velocity_y_Frame1 = round((NW3_HS/.005)+1);
 NW3_R_Hip_Velocity_y_FrameN = round((NW3_Post_350/.005)+1);
 NW3_R_Hip_Velocity_y_subset =
NW3_R_Hip_Velocity_x(NW3_R_Hip_Velocity_y_Frame1:NW3_R_Hip_Velocity_y_FrameN
);
 maxVelocity_NW3_R_Hip_Velocity_y = max(NW3_R_Hip_Velocity_y_subset);
```

```
minVelocity_NW3_R_Hip_Velocity_y = min(NW3_R_Hip_Velocity_y_subset);
if maxVelocity_NW3_R_Hip_Velocity_y >(abs(minVelocity_NW3_R_Hip_Velocity_y))
   pVelocity_NW3_R_Hip_Velocity_y = maxVelocity_NW3_R_Hip_Velocity_y;
else
   pVelocity_NW3_R_Hip_Velocity_y = minVelocity_NW3_R_Hip_Velocity_y;
end


NW3_R_Hip_Velocity_z_Frame1 = round((NW3_HS/.005)+1);
NW3_R_Hip_Velocity_z_FrameN = round((NW3_Post_350/.005)+1);
NW3_R_Hip_Velocity_z_subset =
NW3_R_Hip_Velocity_x(NW3_R_Hip_Velocity_z_Frame1:NW3_R_Hip_Velocity_z_FrameN)
;
maxVelocity_NW3_R_Hip_Velocity_z = max(NW3_R_Hip_Velocity_z_subset);
minVelocity_NW3_R_Hip_Velocity_z = min(NW3_R_Hip_Velocity_z_subset);
if maxVelocity_NW3_R_Hip_Velocity_z >(abs(minVelocity_NW3_R_Hip_Velocity_z))
   pVelocity_NW3_R_Hip_Velocity_z = maxVelocity_NW3_R_Hip_Velocity_z;
else
   pVelocity_NW3_R_Hip_Velocity_z = minVelocity_NW3_R_Hip_Velocity_z;
end

%Store variables in a matrix
B(2,1) = pAngle_ID2_L_Ankle_Angle_x;
B(2,2) = t2p_ID2_L_Ankle_Angle_x;
B(2,3) = pAngle_ID2_L_Ankle_Angle_y;
B(2,4) = t2p_ID2_L_Ankle_Angle_y;
B(2,5) = pAngle_ID2_L_Ankle_Angle_z;
B(2,6) = t2p_ID2_L_Ankle_Angle_z;
B(2,7) = pAngle_ID2_L_Knee_Angle_x;
B(2,8) = t2p_ID2_L_Knee_Angle_x;
B(2,9) = pAngle_ID2_L_Knee_Angle_y;
B(2,10) = t2p_ID2_L_Knee_Angle_y;
B(2,11) = pAngle_ID2_L_Knee_Angle_z;
B(2,12) = t2p_ID2_L_Knee_Angle_z;
B(2,13) = pAngle_ID2_L_Hip_Angle_x;
B(2,14) = t2p_ID2_L_Hip_Angle_x;
B(2,15) = pAngle_ID2_L_Hip_Angle_y;
B(2,16) = t2p_ID2_L_Hip_Angle_y;
B(2,17) = pAngle_ID2_L_Hip_Angle_z;
B(2,18) = t2p_ID2_L_Hip_Angle_z;
B(3,1) = pAngle_ID3_L_Ankle_Angle_x;
B(3,2) = t2p_ID3_L_Ankle_Angle_x;
B(3,3) = pAngle_ID3_L_Ankle_Angle_y;
B(3,4) = t2p_ID3_L_Ankle_Angle_y;
B(3,5) = pAngle_ID3_L_Ankle_Angle_z;
B(3,6) = t2p_ID3_L_Ankle_Angle_z;
```

B(3,7) = pAngle_ID3_L_Knee_Angle_x;
B(3,8) = t2p_ID3_L_Knee_Angle_x;
B(3,9) = pAngle_ID3_L_Knee_Angle_y;
B(3,10) = t2p_ID3_L_Knee_Angle_y;
B(3,11) = pAngle_ID3_L_Knee_Angle_z;
B(3,12) = t2p_ID3_L_Knee_Angle_z;
B(3,13) = pAngle_ID3_L_Hip_Angle_x;
B(3,14) = t2p_ID3_L_Hip_Angle_x;
B(3,15) = pAngle_ID3_L_Hip_Angle_y;
B(3,16) = t2p_ID3_L_Hip_Angle_y;
B(3,17) = pAngle_ID3_L_Hip_Angle_z;
B(3,18) = t2p_ID3_L_Hip_Angle_z;
B(4,1) = pAngle_IPD1_L_Ankle_Angle_x;
B(4,2) = t2p_IPD1_L_Ankle_Angle_x;
B(4,3) = pAngle_IPD1_L_Ankle_Angle_y;
B(4,4) = t2p_IPD1_L_Ankle_Angle_y;
B(4,5) = pAngle_IPD1_L_Ankle_Angle_z;
B(4,6) = t2p_IPD1_L_Ankle_Angle_z;
B(4,7) = pAngle_IPD1_L_Knee_Angle_x;
B(4,8) = t2p_IPD1_L_Knee_Angle_x;
B(4,9) = pAngle_IPD1_L_Knee_Angle_y;
B(4,10) = t2p_IPD1_L_Knee_Angle_y;
B(4,11) = pAngle_IPD1_L_Knee_Angle_z;
B(4,12) = t2p_IPD1_L_Knee_Angle_z;
B(4,13) = pAngle_IPD1_L_Hip_Angle_x;
B(4,14) = t2p_IPD1_L_Hip_Angle_x;
B(4,15) = pAngle_IPD1_L_Hip_Angle_y;
B(4,16) = t2p_IPD1_L_Hip_Angle_y;
B(4,17) = pAngle_IPD1_L_Hip_Angle_z;
B(4,18) = t2p_IPD1_L_Hip_Angle_z;
B(5,1) = pAngle_IPD2_L_Ankle_Angle_x;
B(5,2) = t2p_IPD2_L_Ankle_Angle_x;
B(5,3) = pAngle_IPD2_L_Ankle_Angle_y;
B(5,4) = t2p_IPD2_L_Ankle_Angle_y;
B(5,5) = pAngle_IPD2_L_Ankle_Angle_z;
B(5,6) = t2p_IPD2_L_Ankle_Angle_z;
B(5,7) = pAngle_IPD2_L_Knee_Angle_x;
B(5,8) = t2p_IPD2_L_Knee_Angle_x;
B(5,9) = pAngle_IPD2_L_Knee_Angle_y;
B(5,10) = t2p_IPD2_L_Knee_Angle_y;
B(5,11) = pAngle_IPD2_L_Knee_Angle_z;
B(5,12) = t2p_IPD2_L_Knee_Angle_z;
B(5,13) = pAngle_IPD2_L_Hip_Angle_x;
B(5,14) = t2p_IPD2_L_Hip_Angle_x;
B(5,15) = pAngle_IPD2_L_Hip_Angle_y;
B(5,16) = t2p_IPD2_L_Hip_Angle_y;

```
B(5,17) = pAngle_IPD2_L_Hip_Angle_z;
B(5,18) = t2p_IPD2_L_Hip_Angle_z;
B(6,1) = pAngle_IPD3_L_Ankle_Angle_x;
B(6,2) = t2p_IPD3_L_Ankle_Angle_x;
B(6,3) = pAngle_IPD3_L_Ankle_Angle_y;
B(6,4) = t2p_IPD3_L_Ankle_Angle_y;
B(6,5) = pAngle_IPD3_L_Ankle_Angle_z;
B(6,6) = t2p_IPD3_L_Ankle_Angle_z;
B(6,7) = pAngle_IPD3_L_Knee_Angle_x;
B(6,8) = t2p_IPD3_L_Knee_Angle_x;
B(6,9) = pAngle_IPD3_L_Knee_Angle_y;
B(6,10) = t2p_IPD3_L_Knee_Angle_y;
B(6,11) = pAngle_IPD3_L_Knee_Angle_z;
B(6,12) = t2p_IPD3_L_Knee_Angle_z;
B(6,13) = pAngle_IPD3_L_Hip_Angle_x;
B(6,14) = t2p_IPD3_L_Hip_Angle_x;
B(6,15) = pAngle_IPD3_L_Hip_Angle_y;
B(6,16) = t2p_IPD3_L_Hip_Angle_y;
B(6,17) = pAngle_IPD3_L_Hip_Angle_z;
B(6,18) = t2p_IPD3_L_Hip_Angle_z;
B(7,1) = pAngle_NW1_L_Ankle_Angle_x;
B(7,2) = t2p_NW1_L_Ankle_Angle_x;
B(7,3) = pAngle_NW1_L_Ankle_Angle_y;
B(7,4) = t2p_NW1_L_Ankle_Angle_y;
B(7,5) = pAngle_NW1_L_Ankle_Angle_z;
B(7,6) = t2p_NW1_L_Ankle_Angle_z;
B(7,7) = pAngle_NW1_L_Knee_Angle_x;
B(7,8) = t2p_NW1_L_Knee_Angle_x;
B(7,9) = pAngle_NW1_L_Knee_Angle_y;
B(7,10) = t2p_NW1_L_Knee_Angle_y;
B(7,11) = pAngle_NW1_L_Knee_Angle_z;
B(7,12) = t2p_NW1_L_Knee_Angle_z;
B(7,13) = pAngle_NW1_L_Hip_Angle_x;
B(7,14) = t2p_NW1_L_Hip_Angle_x;
B(7,15) = pAngle_NW1_L_Hip_Angle_y;
B(7,16) = t2p_NW1_L_Hip_Angle_y;
B(7,17) = pAngle_NW1_L_Hip_Angle_z;
B(7,18) = t2p_NW1_L_Hip_Angle_z;
B(8,1) = pAngle_NW2_L_Ankle_Angle_x;
B(8,2) = t2p_NW2_L_Ankle_Angle_x(1,1);
B(8,3) = pAngle_NW2_L_Ankle_Angle_y;
B(8,4) = t2p_NW2_L_Ankle_Angle_y;
B(8,5) = pAngle_NW2_L_Ankle_Angle_z;
B(8,6) = t2p_NW2_L_Ankle_Angle_z;
B(8,7) = pAngle_NW2_L_Knee_Angle_x;
B(8,8) = t2p_NW2_L_Knee_Angle_x;
```

```
B(8,9) = pAngle_NW2_L_Knee_Angle_y;
B(8,10) = t2p_NW2_L_Knee_Angle_y;
B(8,11) = pAngle_NW2_L_Knee_Angle_z;
B(8,12) = t2p_NW2_L_Knee_Angle_z;
B(8,13) = pAngle_NW2_L_Hip_Angle_x;
B(8,14) = t2p_NW2_L_Hip_Angle_x;
B(8,15) = pAngle_NW2_L_Hip_Angle_y;
B(8,16) = t2p_NW2_L_Hip_Angle_y;
B(8,17) = pAngle_NW2_L_Hip_Angle_z;
B(8,18) = t2p_NW2_L_Hip_Angle_z;
B(9,1) = pAngle_NW3_L_Ankle_Angle_x;
B(9,2) = t2p_NW3_L_Ankle_Angle_x;
B(9,3) = pAngle_NW3_L_Ankle_Angle_y;
B(9,4) = t2p_NW3_L_Ankle_Angle_y;
B(9,5) = pAngle_NW3_L_Ankle_Angle_z;
B(9,6) = t2p_NW3_L_Ankle_Angle_z;
B(9,7) = pAngle_NW3_L_Knee_Angle_x;
B(9,8) = t2p_NW3_L_Knee_Angle_x;
B(9,9) = pAngle_NW3_L_Knee_Angle_y;
B(9,10) = t2p_NW3_L_Knee_Angle_y;
B(9,11) = pAngle_NW3_L_Knee_Angle_z;
B(9,12) = t2p_NW3_L_Knee_Angle_z;
B(9,13) = pAngle_NW3_L_Hip_Angle_x;
B(9,14) = t2p_NW3_L_Hip_Angle_x;
B(9,15) = pAngle_NW3_L_Hip_Angle_y;
B(9,16) = t2p_NW3_L_Hip_Angle_y;
B(9,17) = pAngle_NW3_L_Hip_Angle_z;
B(9,18) = t2p_NW3_L_Hip_Angle_z;
B(1,19) = pAngle_ID1_R_Ankle_Angle_x;
B(1,20) = t2p_ID1_R_Ankle_Angle_x;
B(1,21) = pAngle_ID1_R_Ankle_Angle_y;
B(1,22) = t2p_ID1_R_Ankle_Angle_y;
B(1,23) = pAngle_ID1_R_Ankle_Angle_z;
B(1,24) = t2p_ID1_R_Ankle_Angle_z;
B(1,25) = pAngle_ID1_R_Knee_Angle_x;
B(1,26) = t2p_ID1_R_Knee_Angle_x;
B(1,27) = pAngle_ID1_R_Knee_Angle_y;
B(1,28) = t2p_ID1_R_Knee_Angle_y;
B(1,29) = pAngle_ID1_R_Knee_Angle_z;
B(1,30) = t2p_ID1_R_Knee_Angle_z;
B(1,31) = pAngle_ID1_R_Hip_Angle_x;
B(1,32) = t2p_ID1_R_Hip_Angle_x;
B(1,33) = pAngle_ID1_R_Hip_Angle_y;
B(1,34) = t2p_ID1_R_Hip_Angle_y;
B(1,35) = pAngle_ID1_R_Hip_Angle_z;
B(1,36) = t2p_ID1_R_Hip_Angle_z;
```

```
B(2,19) = pAngle_ID2_R_Ankle_Angle_x;
B(2,20) = t2p_ID2_R_Ankle_Angle_x;
B(2,21) = pAngle_ID2_R_Ankle_Angle_y;
B(2,22) = t2p_ID2_R_Ankle_Angle_y;
B(2,23) = pAngle_ID2_R_Ankle_Angle_z;
B(2,24) = t2p_ID2_R_Ankle_Angle_z;
B(2,25) = pAngle_ID2_R_Knee_Angle_x;
B(2,26) = t2p_ID2_R_Knee_Angle_x;
B(2,27) = pAngle_ID2_R_Knee_Angle_y;
B(2,28) = t2p_ID2_R_Knee_Angle_y;
B(2,29) = pAngle_ID2_R_Knee_Angle_z;
B(2,30) = t2p_ID2_R_Knee_Angle_z;
B(2,31) = pAngle_ID2_R_Hip_Angle_x;
B(2,32) = t2p_ID2_R_Hip_Angle_x;
B(2,33) = pAngle_ID2_R_Hip_Angle_y;
B(2,34) = t2p_ID2_R_Hip_Angle_y;
B(2,35) = pAngle_ID2_R_Hip_Angle_z;
B(2,36) = t2p_ID2_R_Hip_Angle_z;
B(3,19) = pAngle_ID3_R_Ankle_Angle_x;
B(3,20) = t2p_ID3_R_Ankle_Angle_x;
B(3,21) = pAngle_ID3_R_Ankle_Angle_y;
B(3,22) = t2p_ID3_R_Ankle_Angle_y;
B(3,23) = pAngle_ID3_R_Ankle_Angle_z;
B(3,24) = t2p_ID3_R_Ankle_Angle_z;
B(3,25) = pAngle_ID3_R_Knee_Angle_x;
B(3,26) = t2p_ID3_R_Knee_Angle_x;
B(3,27) = pAngle_ID3_R_Knee_Angle_y;
B(3,28) = t2p_ID3_R_Knee_Angle_y;
B(3,29) = pAngle_ID3_R_Knee_Angle_z;
B(3,30) = t2p_ID3_R_Knee_Angle_z;
B(3,31) = pAngle_ID3_R_Hip_Angle_x;
B(3,32) = t2p_ID3_R_Hip_Angle_x;
B(3,33) = pAngle_ID3_R_Hip_Angle_y;
B(3,34) = t2p_ID3_R_Hip_Angle_y;
B(3,35) = pAngle_ID3_R_Hip_Angle_z;
B(3,36) = t2p_ID3_R_Hip_Angle_z;
B(4,19) = pAngle_IPD1_R_Ankle_Angle_x;
B(4,20) = t2p_IPD1_R_Ankle_Angle_x;
B(4,21) = pAngle_IPD1_R_Ankle_Angle_y;
B(4,22) = t2p_IPD1_R_Ankle_Angle_y;
B(4,23) = pAngle_IPD1_R_Ankle_Angle_z;
B(4,24) = t2p_IPD1_R_Ankle_Angle_z;
B(4,25) = pAngle_IPD1_R_Knee_Angle_x;
B(4,26) = t2p_IPD1_R_Knee_Angle_x;
B(4,27) = pAngle_IPD1_R_Knee_Angle_y;
B(4,28) = t2p_IPD1_R_Knee_Angle_y;
```

B(4,29) = pAngle_IPD1_R_Knee_Angle_z;
B(4,30) = t2p_IPD1_R_Knee_Angle_z;
B(4,31) = pAngle_IPD1_R_Hip_Angle_x;
B(4,32) = t2p_IPD1_R_Hip_Angle_x;
B(4,33) = pAngle_IPD1_R_Hip_Angle_y;
B(4,34) = t2p_IPD1_R_Hip_Angle_y;
B(4,35) = pAngle_IPD1_R_Hip_Angle_z;
B(4,36) = t2p_IPD1_R_Hip_Angle_z;
B(5,19) = pAngle_IPD2_R_Ankle_Angle_x;
B(5,20) = t2p_IPD2_R_Ankle_Angle_x;
B(5,21) = pAngle_IPD2_R_Ankle_Angle_y;
B(5,22) = t2p_IPD2_R_Ankle_Angle_y;
B(5,23) = pAngle_IPD2_R_Ankle_Angle_z;
B(5,24) = t2p_IPD2_R_Ankle_Angle_z;
B(5,25) = pAngle_IPD2_R_Knee_Angle_x;
B(5,26) = t2p_IPD2_R_Knee_Angle_x;
B(5,27) = pAngle_IPD2_R_Knee_Angle_y;
B(5,28) = t2p_IPD2_R_Knee_Angle_y;
B(5,29) = pAngle_IPD2_R_Knee_Angle_z;
B(5,30) = t2p_IPD2_R_Knee_Angle_z;
B(5,31) = pAngle_IPD2_R_Hip_Angle_x;
B(5,32) = t2p_IPD2_R_Hip_Angle_x;
B(5,33) = pAngle_IPD2_R_Hip_Angle_y;
B(5,34) = t2p_IPD2_R_Hip_Angle_y;
B(5,35) = pAngle_IPD2_R_Hip_Angle_z;
B(5,36) = t2p_IPD2_R_Hip_Angle_z;
B(6,19) = pAngle_IPD3_R_Ankle_Angle_x;
B(6,20) = t2p_IPD3_R_Ankle_Angle_x;
B(6,21) = pAngle_IPD3_R_Ankle_Angle_y;
B(6,22) = t2p_IPD3_R_Ankle_Angle_y;
B(6,23) = pAngle_IPD3_R_Ankle_Angle_z;
B(6,24) = t2p_IPD3_R_Ankle_Angle_z;
B(6,25) = pAngle_IPD3_R_Knee_Angle_x;
B(6,26) = t2p_IPD3_R_Knee_Angle_x;
B(6,27) = pAngle_IPD3_R_Knee_Angle_y;
B(6,28) = t2p_IPD3_R_Knee_Angle_y;
B(6,29) = pAngle_IPD3_R_Knee_Angle_z;
B(6,30) = t2p_IPD3_R_Knee_Angle_z;
B(6,31) = pAngle_IPD3_R_Hip_Angle_x;
B(6,32) = t2p_IPD3_R_Hip_Angle_x;
B(6,33) = pAngle_IPD3_R_Hip_Angle_y;
B(6,34) = t2p_IPD3_R_Hip_Angle_y;
B(6,35) = pAngle_IPD3_R_Hip_Angle_z;
B(6,36) = t2p_IPD3_R_Hip_Angle_z;
B(7,19) = pAngle_NW1_R_Ankle_Angle_x;
B(7,20) = t2p_NW1_R_Ankle_Angle_x;

```
B(7,21) = pAngle_NW1_R_Ankle_Angle_y;
B(7,22) = t2p_NW1_R_Ankle_Angle_y;
B(7,23) = pAngle_NW1_R_Ankle_Angle_z;
B(7,24) = t2p_NW1_R_Ankle_Angle_z;
B(7,25) = pAngle_NW1_R_Knee_Angle_x;
B(7,26) = t2p_NW1_R_Knee_Angle_x;
B(7,27) = pAngle_NW1_R_Knee_Angle_y;
B(7,28) = t2p_NW1_R_Knee_Angle_y;
B(7,29) = pAngle_NW1_R_Knee_Angle_z;
B(7,30) = t2p_NW1_R_Knee_Angle_z;
B(7,31) = pAngle_NW1_R_Hip_Angle_x;
B(7,32) = t2p_NW1_R_Hip_Angle_x;
B(7,33) = pAngle_NW1_R_Hip_Angle_y;
B(7,34) = t2p_NW1_R_Hip_Angle_y;
B(7,35) = pAngle_NW1_R_Hip_Angle_z;
B(7,36) = t2p_NW1_R_Hip_Angle_z;
B(8,19) = pAngle_NW2_R_Ankle_Angle_x;
B(8,20) = t2p_NW2_R_Ankle_Angle_x;
B(8,21) = pAngle_NW2_R_Ankle_Angle_y;
B(8,22) = t2p_NW2_R_Ankle_Angle_y;
B(8,23) = pAngle_NW2_R_Ankle_Angle_z;
B(8,24) = t2p_NW2_R_Ankle_Angle_z;
B(8,25) = pAngle_NW2_R_Knee_Angle_x;
B(8,26) = t2p_NW2_R_Knee_Angle_x;
B(8,27) = pAngle_NW2_R_Knee_Angle_y;
B(8,28) = t2p_NW2_R_Knee_Angle_y;
B(8,29) = pAngle_NW2_R_Knee_Angle_z;
B(8,30) = t2p_NW2_R_Knee_Angle_z;
B(8,31) = pAngle_NW2_R_Hip_Angle_x;
B(8,32) = t2p_NW2_R_Hip_Angle_x;
B(8,33) = pAngle_NW2_R_Hip_Angle_y;
B(8,34) = t2p_NW2_R_Hip_Angle_y;
B(8,35) = pAngle_NW2_R_Hip_Angle_z;
B(8,36) = t2p_NW2_R_Hip_Angle_z;
B(9,19) = pAngle_NW3_R_Ankle_Angle_x;
B(9,20) = t2p_NW3_R_Ankle_Angle_x;
B(9,21) = pAngle_NW3_R_Ankle_Angle_y;
B(9,22) = t2p_NW3_R_Ankle_Angle_y;
B(9,23) = pAngle_NW3_R_Ankle_Angle_z;
B(9,24) = t2p_NW3_R_Ankle_Angle_z;
B(9,25) = pAngle_NW3_R_Knee_Angle_x;
B(9,26) = t2p_NW3_R_Knee_Angle_x;
B(9,27) = pAngle_NW3_R_Knee_Angle_y;
B(9,28) = t2p_NW3_R_Knee_Angle_y;
B(9,29) = pAngle_NW3_R_Knee_Angle_z;
B(9,30) = t2p_NW3_R_Knee_Angle_z;
```

B(9,31) = pAngle_NW3_R_Hip_Angle_x;
B(9,32) = t2p_NW3_R_Hip_Angle_x;
B(9,33) = pAngle_NW3_R_Hip_Angle_y;
B(9,34) = t2p_NW3_R_Hip_Angle_y;
B(9,35) = pAngle_NW3_R_Hip_Angle_z;
B(9,36) = t2p_NW3_R_Hip_Angle_z;
B(1,37) = pVelocity_ID1_L_Ankle_Velocity_x;
B(1,38) = pVelocity_ID1_L_Ankle_Velocity_y;
B(1,39) = pVelocity_ID1_L_Ankle_Velocity_z;
B(1,40) = pVelocity_ID1_L_Knee_Velocity_x;
B(1,41) = pVelocity_ID1_L_Knee_Velocity_y;
B(1,42) = pVelocity_ID1_L_Knee_Velocity_z;
B(1,43) = pVelocity_ID1_L_Hip_Velocity_x;
B(1,44) = pVelocity_ID1_L_Hip_Velocity_y;
B(1,45) = pVelocity_ID1_L_Hip_Velocity_z;
B(2,37) = pVelocity_ID2_L_Ankle_Velocity_x;
B(2,38) = pVelocity_ID2_L_Ankle_Velocity_y;
B(2,39) = pVelocity_ID2_L_Ankle_Velocity_z;
B(2,40) = pVelocity_ID2_L_Knee_Velocity_x;
B(2,41) = pVelocity_ID2_L_Knee_Velocity_y;
B(2,42) = pVelocity_ID2_L_Knee_Velocity_z;
B(2,43) = pVelocity_ID2_L_Hip_Velocity_x;
B(2,44) = pVelocity_ID2_L_Hip_Velocity_y;
B(2,45) = pVelocity_ID2_L_Hip_Velocity_z;
B(3,37) = pVelocity_ID3_L_Ankle_Velocity_x;
B(3,38) = pVelocity_ID3_L_Ankle_Velocity_y;
B(3,39) = pVelocity_ID3_L_Ankle_Velocity_z;
B(3,40) = pVelocity_ID3_L_Knee_Velocity_x;
B(3,41) = pVelocity_ID3_L_Knee_Velocity_y;
B(3,42) = pVelocity_ID3_L_Knee_Velocity_z;
B(3,43) = pVelocity_ID3_L_Hip_Velocity_x;
B(3,44) = pVelocity_ID3_L_Hip_Velocity_y;
B(3,45) = pVelocity_ID3_L_Hip_Velocity_z;
B(4,37) = pVelocity_IPD1_L_Ankle_Velocity_x;
B(4,38) = pVelocity_IPD1_L_Ankle_Velocity_y;
B(4,39) = pVelocity_IPD1_L_Ankle_Velocity_z;
B(4,40) = pVelocity_IPD1_L_Knee_Velocity_x;
B(4,41) = pVelocity_IPD1_L_Knee_Velocity_y;
B(4,42) = pVelocity_IPD1_L_Knee_Velocity_z;
B(4,43) = pVelocity_IPD1_L_Hip_Velocity_x;
B(4,44) = pVelocity_IPD1_L_Hip_Velocity_y;
B(4,45) = pVelocity_IPD1_L_Hip_Velocity_z;
B(5,37) = pVelocity_IPD2_L_Ankle_Velocity_x;
B(5,38) = pVelocity_IPD2_L_Ankle_Velocity_y;
B(5,39) = pVelocity_IPD2_L_Ankle_Velocity_z;
B(5,40) = pVelocity_IPD2_L_Knee_Velocity_x;

B(5,41) = pVelocity_IPD2_L_Knee_Velocity_y;
B(5,42) = pVelocity_IPD2_L_Knee_Velocity_z;
B(5,43) = pVelocity_IPD2_L_Hip_Velocity_x;
B(5,44) = pVelocity_IPD2_L_Hip_Velocity_y;
B(5,45) = pVelocity_IPD2_L_Hip_Velocity_z;
B(6,37) = pVelocity_IPD3_L_Ankle_Velocity_x;
B(6,38) = pVelocity_IPD3_L_Ankle_Velocity_y;
B(6,39) = pVelocity_IPD3_L_Ankle_Velocity_z;
B(6,40) = pVelocity_IPD3_L_Knee_Velocity_x;
B(6,41) = pVelocity_IPD3_L_Knee_Velocity_y;
B(6,42) = pVelocity_IPD3_L_Knee_Velocity_z;
B(6,43) = pVelocity_IPD3_L_Hip_Velocity_x;
B(6,44) = pVelocity_IPD3_L_Hip_Velocity_y;
B(6,45) = pVelocity_IPD3_L_Hip_Velocity_z;
B(7,37) = pVelocity_NW1_L_Ankle_Velocity_x;
B(7,38) = pVelocity_NW1_L_Ankle_Velocity_y;
B(7,39) = pVelocity_NW1_L_Ankle_Velocity_z;
B(7,40) = pVelocity_NW1_L_Knee_Velocity_x;
B(7,41) = pVelocity_NW1_L_Knee_Velocity_y;
B(7,42) = pVelocity_NW1_L_Knee_Velocity_z;
B(7,43) = pVelocity_NW1_L_Hip_Velocity_x;
B(7,44) = pVelocity_NW1_L_Hip_Velocity_y;
B(7,45) = pVelocity_NW1_L_Hip_Velocity_z;
B(8,37) = pVelocity_NW2_L_Ankle_Velocity_x;
B(8,38) = pVelocity_NW2_L_Ankle_Velocity_y;
B(8,39) = pVelocity_NW2_L_Ankle_Velocity_z;
B(8,40) = pVelocity_NW2_L_Knee_Velocity_x;
B(8,41) = pVelocity_NW2_L_Knee_Velocity_y;
B(8,42) = pVelocity_NW2_L_Knee_Velocity_z;
B(8,43) = pVelocity_NW2_L_Hip_Velocity_x;
B(8,44) = pVelocity_NW2_L_Hip_Velocity_y;
B(8,45) = pVelocity_NW2_L_Hip_Velocity_z;
B(9,37) = pVelocity_NW3_L_Ankle_Velocity_x;
B(9,38) = pVelocity_NW3_L_Ankle_Velocity_y;
B(9,39) = pVelocity_NW3_L_Ankle_Velocity_z;
B(9,40) = pVelocity_NW3_L_Knee_Velocity_x;
B(9,41) = pVelocity_NW3_L_Knee_Velocity_y;
B(9,42) = pVelocity_NW3_L_Knee_Velocity_z;
B(9,43) = pVelocity_NW3_L_Hip_Velocity_x;
B(9,44) = pVelocity_NW3_L_Hip_Velocity_y;
B(9,45) = pVelocity_NW3_L_Hip_Velocity_z;
B(1,46) = pVelocity_ID1_R_Ankle_Velocity_x;
B(1,47) = pVelocity_ID1_R_Ankle_Velocity_y;
B(1,48) = pVelocity_ID1_R_Ankle_Velocity_z;
B(1,49) = pVelocity_ID1_R_Knee_Velocity_x;
B(1,50) = pVelocity_ID1_R_Knee_Velocity_y;

B(1,51) = pVelocity_ID1_R_Knee_Velocity_z;
B(1,52) = pVelocity_ID1_R_Hip_Velocity_x;
B(1,53) = pVelocity_ID1_R_Hip_Velocity_y;
B(1,54) = pVelocity_ID1_R_Hip_Velocity_z;
B(2,46) = pVelocity_ID2_R_Ankle_Velocity_x;
B(2,47) = pVelocity_ID2_R_Ankle_Velocity_y;
B(2,48) = pVelocity_ID2_R_Ankle_Velocity_z;
B(2,49) = pVelocity_ID2_R_Knee_Velocity_x;
B(2,50) = pVelocity_ID2_R_Knee_Velocity_y;
B(2,51) = pVelocity_ID2_R_Knee_Velocity_z;
B(2,52) = pVelocity_ID2_R_Hip_Velocity_x;
B(2,53) = pVelocity_ID2_R_Hip_Velocity_y;
B(2,54) = pVelocity_ID2_R_Hip_Velocity_z;
B(3,46) = pVelocity_ID3_R_Ankle_Velocity_x;
B(3,47) = pVelocity_ID3_R_Ankle_Velocity_y;
B(3,48) = pVelocity_ID3_R_Ankle_Velocity_z;
B(3,49) = pVelocity_ID3_R_Knee_Velocity_x;
B(3,50) = pVelocity_ID3_R_Knee_Velocity_y;
B(3,51) = pVelocity_ID3_R_Knee_Velocity_z;
B(3,52) = pVelocity_ID3_R_Hip_Velocity_x;
B(3,53) = pVelocity_ID3_R_Hip_Velocity_y;
B(3,54) = pVelocity_ID3_R_Hip_Velocity_z;
B(4,46) = pVelocity_IPD1_R_Ankle_Velocity_x;
B(4,47) = pVelocity_IPD1_R_Ankle_Velocity_y;
B(4,48) = pVelocity_IPD1_R_Ankle_Velocity_z;
B(4,49) = pVelocity_IPD1_R_Knee_Velocity_x;
B(4,50) = pVelocity_IPD1_R_Knee_Velocity_y;
B(4,51) = pVelocity_IPD1_R_Knee_Velocity_z;
B(4,52) = pVelocity_IPD1_R_Hip_Velocity_x;
B(4,53) = pVelocity_IPD1_R_Hip_Velocity_y;
B(4,54) = pVelocity_IPD1_R_Hip_Velocity_z;
B(5,46) = pVelocity_IPD2_R_Ankle_Velocity_x;
B(5,47) = pVelocity_IPD2_R_Ankle_Velocity_y;
B(5,48) = pVelocity_IPD2_R_Ankle_Velocity_z;
B(5,49) = pVelocity_IPD2_R_Knee_Velocity_x;
B(5,50) = pVelocity_IPD2_R_Knee_Velocity_y;
B(5,51) = pVelocity_IPD2_R_Knee_Velocity_z;
B(5,52) = pVelocity_IPD2_R_Hip_Velocity_x;
B(5,53) = pVelocity_IPD2_R_Hip_Velocity_y;
B(5,54) = pVelocity_IPD2_R_Hip_Velocity_z;
B(6,46) = pVelocity_IPD3_R_Ankle_Velocity_x;
B(6,47) = pVelocity_IPD3_R_Ankle_Velocity_y;
B(6,48) = pVelocity_IPD3_R_Ankle_Velocity_z;
B(6,49) = pVelocity_IPD3_R_Knee_Velocity_x;
B(6,50) = pVelocity_IPD3_R_Knee_Velocity_y;
B(6,51) = pVelocity_IPD3_R_Knee_Velocity_z;

```
B(6,52) = pVelocity_IPD3_R_Hip_Velocity_x;
B(6,53) = pVelocity_IPD3_R_Hip_Velocity_y;
B(6,54) = pVelocity_IPD3_R_Hip_Velocity_z;
B(7,46) = pVelocity_NW1_R_Ankle_Velocity_x;
B(7,47) = pVelocity_NW1_R_Ankle_Velocity_y;
B(7,48) = pVelocity_NW1_R_Ankle_Velocity_z;
B(7,49) = pVelocity_NW1_R_Knee_Velocity_x;
B(7,50) = pVelocity_NW1_R_Knee_Velocity_y;
B(7,51) = pVelocity_NW1_R_Knee_Velocity_z;
B(7,52) = pVelocity_NW1_R_Hip_Velocity_x;
B(7,53) = pVelocity_NW1_R_Hip_Velocity_y;
B(7,54) = pVelocity_NW1_R_Hip_Velocity_z;
B(8,46) = pVelocity_NW2_R_Ankle_Velocity_x;
B(8,47) = pVelocity_NW2_R_Ankle_Velocity_y;
B(8,48) = pVelocity_NW2_R_Ankle_Velocity_z;
B(8,49) = pVelocity_NW2_R_Knee_Velocity_x;
B(8,50) = pVelocity_NW2_R_Knee_Velocity_y;
B(8,51) = pVelocity_NW2_R_Knee_Velocity_z;
B(8,52) = pVelocity_NW2_R_Hip_Velocity_x;
B(8,53) = pVelocity_NW2_R_Hip_Velocity_y;
B(8,54) = pVelocity_NW2_R_Hip_Velocity_z;
B(9,46) = pVelocity_NW3_R_Ankle_Velocity_x;
B(9,47) = pVelocity_NW3_R_Ankle_Velocity_y;
B(9,48) = pVelocity_NW3_R_Ankle_Velocity_z;
B(9,49) = pVelocity_NW3_R_Knee_Velocity_x;
B(9,50) = pVelocity_NW3_R_Knee_Velocity_y;
B(9,51) = pVelocity_NW3_R_Knee_Velocity_z;
B(9,52) = pVelocity_NW3_R_Hip_Velocity_x;
B(9,53) = pVelocity_NW3_R_Hip_Velocity_y;
B(9,54) = pVelocity_NW3_R_Hip_Velocity_z;

%End
```

APPENDIX C: PRE-PARTICIPATION QUESTIONNAIRE

Pre-Participation Questionnaire

Please respond truthfully to the following questions:

1. **Have you suffered an orthopedic injury that caused you to seek medical assistance within the past 6 months?**
    a. **Yes** **No**
2. **Have you had any previous lower extremity surgery?**
    a. **Yes** **No**
3. What is your age? _____
4. What is your height? _____ feet _____ inches
5. What is your weight? _____ lbs
6. What is your sex?
    a. Male
    b. Female
7. On average, how many days per week do you engage in physical activity that lasts at least 30 minutes?

    1 2 3 4 5 6 7

APPENDIX D: ADDITIONAL FIGURES

Additional figures below.

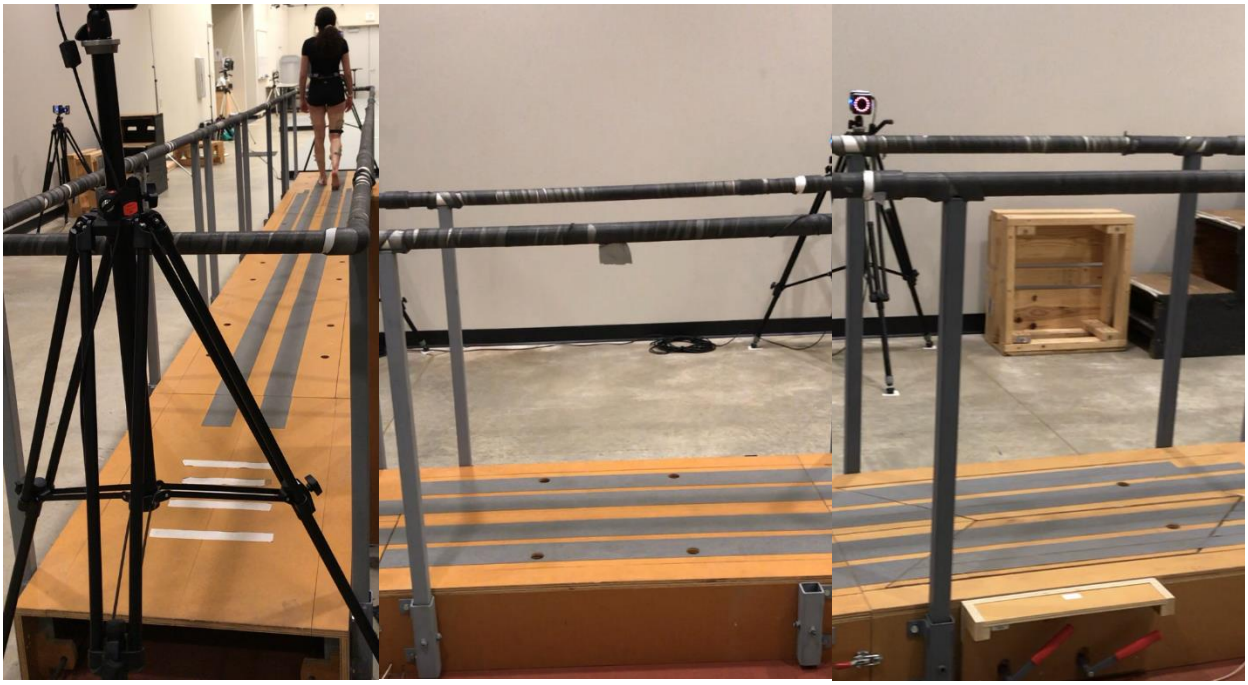

Figure 23. Custom Build Walkway. Left image shoes full walkway, middle image is the segment containing the inversion drop, right image is the segment containing the combined inversion/plantarflexion drop.

Above in Fig. 23 is the custom build walkway with both drop platforms.

Figure 24. Electrode placement.

Fig. 24 above shows the electrode placement and arrangement of the BIOPAC units on one subject.

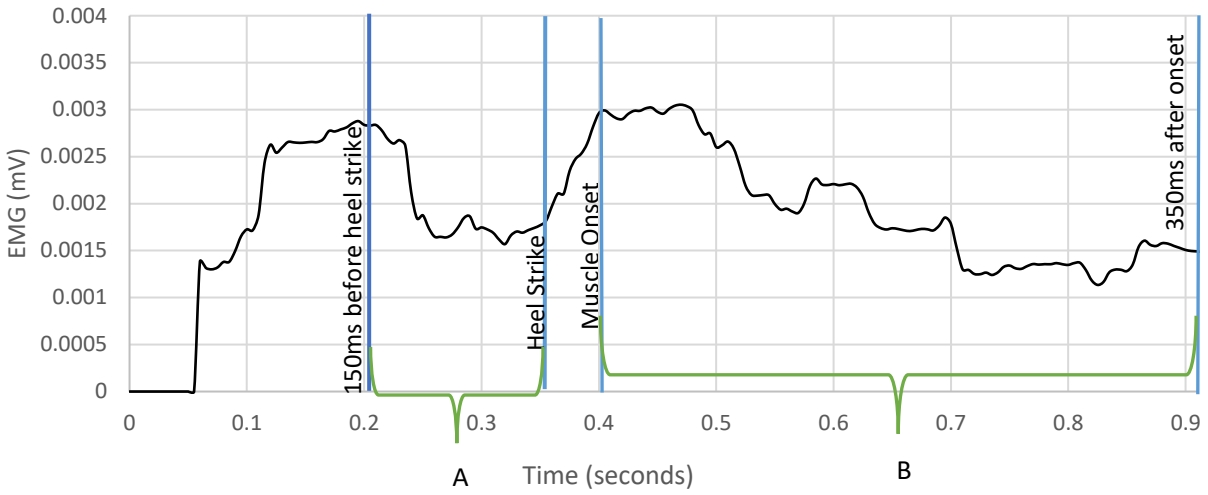

Figure 25. Sample EMG Time Series. A is the window of analysis 150ms prior to the heel strike in which the mean and standard deviation are calculated to then find muscle onset following heel strike when the signal is two standard deviations above that calculated mean. B is the window of analysis where peak and average EMG are calculated.

Fig. 25 above provides a visual of the EMG variables which were calculated as follows. Muscle onset calculated between heel strike and the end of the trial and recorded when the signal reached two standard deviations above the mean of the signal 150ms prior to the corresponding event. Reaction time calculated as the difference between onset time and the time of the corresponding true drop or heel strike event. Peak and average EMG was calculated between muscle onset and the 350ms following. Time to peak EMG was computed as the time between peak EMG and muscle onset.

Figure 26. Marker placement.

Fig. 26 shows the kinematic marker placement on one subject. Clusters were used for the shank and thigh.

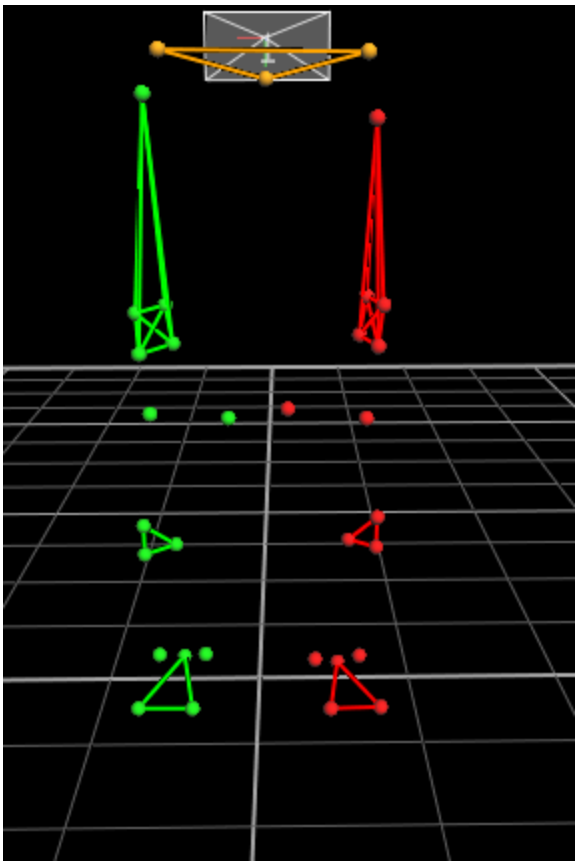

Figure 27. Lower extremity model created in Vicon.

Above in Fig. 27 is the model created in Vicon from the static calibration trial. The color red designates the left side, and the color green designates the right side. The yellow indicates the markers used for the pelvis including the sacrum and both iliac crests. For the dynamic trials, the medial and lateral ankle and knee markers were removed.

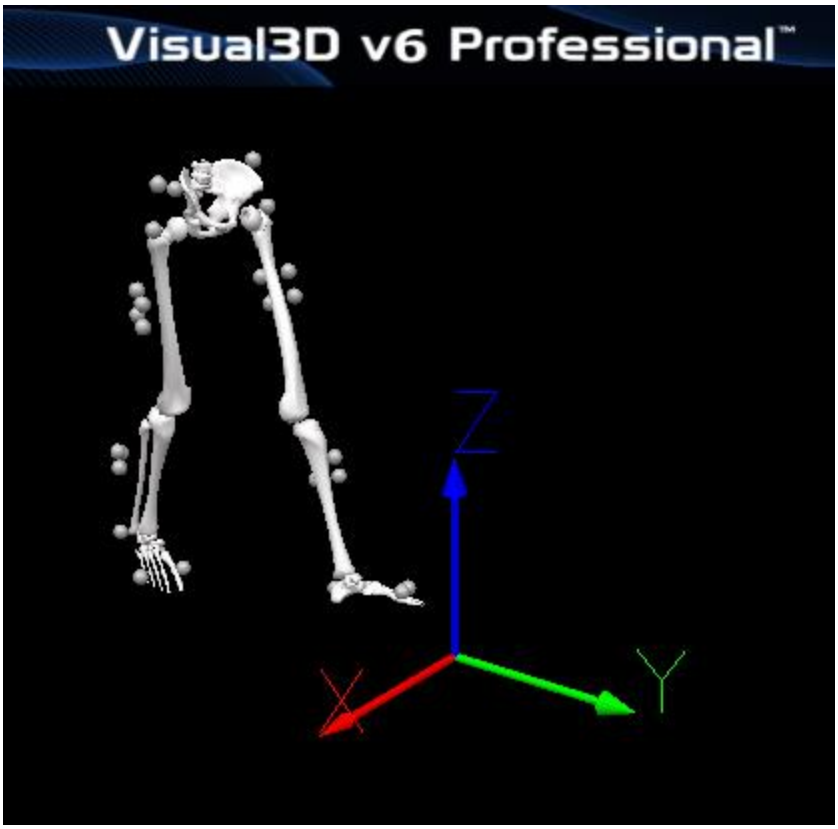

Figure 28. Lower extremity model compiled in Visual 3D.

Above in Fig. 28 is the lower extremity model that was created and compiled in Visual 3D based on the processed exported data file from Vicon and the assigned Kettlebell model file with subject specific anthropometrics.