Changes in the Horizontal Angular Vestibulo-ocular Reflex of SLS-2 Space Shuttle Astronauts Due to Weightlessness.

by

Christopher Francis Pouliot

S.B., Massachusetts Institute of Technology, 1993

Submitted to the Department of Aeronautics and Astronautics
in Partial Fulfillment of
the Requirements for the Degree of

MASTER OF SCIENCE

in

AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge, Massachusetts
February, 1995

Signature of Author ______________________________
Department of Aeronautics and Astronautics

Certified by ______________________________
Dr. Charles M. Oman
Thesis Supervisor

Accepted by ______________________________
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

Changes in the Horizontal Angular Vestibulo-ocular Reflex of SLS-2 Space Shuttle Astronauts Due to Weightlessness.

By Christopher Francis Pouliot

Submitted to the Department of Aeronautics and Astronautics
on October 12, 1994 in partial fulfillment of the
requirements for the Degree Master of Science in
Aeronautics and Astronautics

ABSTRACT

Vestibulo-Ocular Reflex (VOR) data was recorded in 4 astronauts (3m, 1f; age 36-51) before, during, and after a 14 day space shuttle mission (November, 1993). 129 days before launch subjects were tested in parabolic flight. Subjects were then tested pre-flight, 122/110/88/18 days before launch; in-flight, 4/10 days after launch; post-flight, 0/1/2/6/9/11 days after landing. Horizontal eye position (bi-temporal EOG) was sampled by computer. Subjects were blindfolded eyes open and instructed to keep head erect, and to gaze straight ahead. Alertness was maintained with conversation. To study "head erect" yaw axis dynamics, subjects were accelerated to a angular velocity of 120 deg/sec within 0.5 seconds with their heads erect. After 1 minute, the chair stopped within 0.5 seconds, and the subjects continued to keep their heads erect. Post-rotatory recording of the VOR continued for 1 minute. This procedure was repeated in the opposite direction. To study "dumping" yaw axis dynamics, the same procedure was used, except that after chair stop, subjects immediately pitched their head forward 90 degrees, and maintained face down position for 1 minute. All 4 head erect and dumping procedures were then repeated. Horizontal SPV was calculated using order statistic fast phase filtering (Engelken & Stevens, 1990). Momentary SPV dropouts were detected and removed using an iterative model fitting method (Balkwill, 1992) A first order exponential model was fit to the remaining data to yield a per- and post-rotatory gain and time constant. The data was also ensembled averaged. Techniques for near real time statistical data analysis were further developed.

In parabolic tests, the mean head erect time constant of all three subjects tested shortened (2/3 changes were significant, $p < 0.05$), confirming for the first time for individual subjects the finding of DiZio and Lackner (1988). After several days in orbit, the head erect time constant of two of the four subjects ("slow adapters") was reduced (58%; 72%) compared to preflight. (1/2 significant, $p < 0.05$). The time constants of the other two subjects ("fast adapters") were virtually unchanged (99%) or increased (123%) significantly ($p < 0.05$). Per-rotatory data showed similar trends. In all four subjects, in-flight dumping time constants were statistically indistinguishable from in-flight head erect. The "fast adapters" showed a significant increase of the in-flight dumping time constant when compared to pre-flight, whereas the "slow adapters" did not.

Changes in VOR time constant are thought to result from the effects of otolith cue conflict on central oculomotor velocity storage mechanisms. It is hypothesized that the "slow adapters" never fully recovered velocity storage which was lost in parabolic flight and initial entry into orbit, due to otolith conflict.

This would explain why, after 4-10 days in orbit , a) their in-flight head erect time constants were shorter than pre-flight, and b) their in-flight dumping time constants were similar to their in-flight head erect time constants, and c) their in-flight dumping constants were similar to pre-flight.

Similarly, "fast adapters" lose velocity storage in both parabolic flight and initial entry into orbit. However after 4-10 days in orbit, it is hypothesized that "fast adapters" regain velocity storage. This would explain why a) their in-flight head erect time constants were not shorter than pre-flight, b) their in-flight dumping time constants were similar to their in-flight head erect time constants, and c) their in-flight dumping time constants were significantly larger than pre-flight.

No statistically significant VOR gain changes were seen in parabolic or orbital flight. Dumping was present in all subjects early post-flight. Near real time data processing facilitated science decision making during pre-, in-, and post-flight data collection.

Supported by NASA Contract NAS9-16523.

Thesis Supervisor: Dr. Charles M. Oman
Title: Director
M.I.T. Man-Vehicle Laboratory
Senior Research Engineer
Department of Aeronautics and Astronautics

Acknowledgments

Many thanks to my parents, who have always fully supported me throughout my life, including the good and the bad.

Thanks to all my friends, who when needed,. have helped me forget that I was a student at a place like MIT.

Thanks to my advisor, who has given me the rare opportunity to be able to perform a experiment flown on the space shuttle, and who has helped me improve my English grammar.

Thanks to NASA Contract NAS9-16523, which has paid my tuition for the past 1 1/2 years.

Table of Contents

Abstract 2
Acknowledgments 4
Table of Contents 5

Chapter 1: Introduction & Background

1.1 Introduction 7
1.2 Previous Science Results 9
1.3 Previous Engineering Results 11
1.4 Goals of Thesis Research 14

Chapter 2: Experimental Methods

2.1 Subjects 15
2.2 Ground Equipment 16
2.3 Ground Experimental Protocol 17
2.4 Differences in Experimental Methods for In-flight and KC-135 Testing 19

Chapter 3: Data Analysis

3.1 Calibration of Eye Position Data 21
3.2 Calculation of Slow Phase Velocity
3.2.1 PFMH Filter 22
3.2.2 Calculation of Eye Velocity 22
3.2.3 AATM Filter 23
3.3 Tachometer Analysis 23
3.4 Outlier Detection and Removal 23
3.5 Decimation 24
3.6 Model Fitting 24
3.7 Run Segment Rejection Criteria 25
3.8 Near real-time statistical analysis 28
3.9 Statistical Comparisons of the Data
3.9.1 Kolmogorov-Smirnov Test 31
3.9.2 Two-Sample t-Test for Independent Samples 34
3.9.3 Sum of T-square statistic 34
3.9.4 Actual use of the statistical tests 35

Chapter 4: Results

4.1 Completion status of runs 36
4.2 Run rejection results 36
4.3 Near Real-Time Analysis Experience 38
4.4 Directional Asymmetry Analysis 39
4.5 Comparisons
4.5.1 The Effect of an Acute Exposure to Zero-gravity on Slow-Phase Velocity 44

4.5.2 The Effect of Prolonged 0-G Exposure on Head erect Slow-Phase Velocity 47
4.5.3 The Effect of Prolonged 0-G Exposure on Per-rotatory Slow-Phase Velocity 50
4.5.4 Changes in early post-flight head-erect responses 53
4.5.5 Changes in early post-flight per-rotatory responses 56
4.5.6 The Effect of Pre-Flight Dumping on Slow-Phase Velocity 59
4.5.7 The Effect of Dumping on In-flight Slow-Phase Velocity 62
4.5.8 The Effect of Prolonged 0-G Exposure on Dumping Slow Phase Velocity 65
4.5.9 The Effect of Dumping on Early Post-flight Slow-Phase Velocity 68
4.5.10 Changes in early post-flight dumping responses 71

Chapter 5: Discussion and Conclusion

5.1 "How did subject's pre-flight head erect VOR responses compare with results from previous studies ?" 74
5.2 "How did the head-erect SPV response change in parabolic flight ?" 75
5.3 "How did the head-erect SPV response change in prolonged weightlessness ?" 76
5.4 "How did the head erect SPV response change post-flight ?" 79
5.5 "Was the pre-flight dumping response consistent with previous studies ?" 80
5.6 "How did the dumping SPV response change in prolonged weightlessness ?" 81
5.7 "How did the dumping SPV response change post-flight ?" 82
5.8 "Did near-real time SPV analysis facilitate decision making during testing ?" 82
5.9 "Are there any recommendations for future analysis and future studies ?" 83

Bibliography 89

Appendix A: Ground data analysis Matlab scripts 92
Appendix B: In-flight specific data analysis Matlab scripts 148
Appendix C: Parabolic specific data analysis Matlab scripts 169

Chapter 1: Introduction & Background

1.1 Introduction

Space flight causes many potential medical problems for astronauts. Calcium is lost in the bones, muscle strength is reduced, immunity is decreased, and vestibular function is changed. The changes in the vestibular system are believed to be the cause of space motion sickness (SMS), which affects over 70% of astronauts. For shuttle flights of short duration, this can affect the performance of the astronaut for an appreciable part of the mission. Before we can establish any long term presence in space, we must fully understand the above mentioned medical problems. In this experiment the effect of zero-gravity on the horizontal vestibular ocular reflex (VOR) is studied.

If a person is subject to a step change in head angular velocity, the slow phase velocity (SPV) of the eyes first increases suddenly in the direction opposite to that of the rotation and then decays in a quasi-exponential manner. Upon returning the subject to rest, after a prolonged spin, the post-rotatory VOR is similar to the per-rotatory VOR, except the direction of the SPV is the same as the direction of the initial rotation (Figure 1.1).

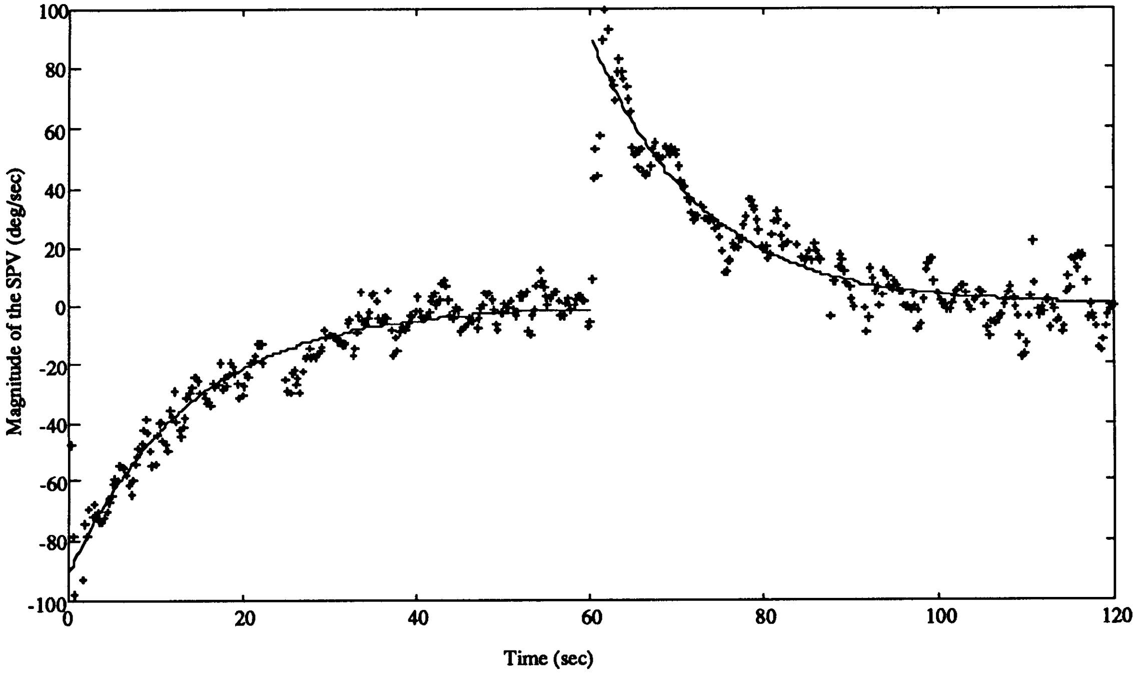


Figure 1.1: Example of the quasi-exponential behavior of the SPV (The black lines are model fits)

A common way to model this quasi-exponential behavior (Groen, 1962) is:

$$\frac{spv(s)}{\omega(s)} = \frac{-Ks}{Ts+1} \tag{1.1}$$

where K is the gain and T is the apparent time constant. The time response of this model to a step input, Dw is:

$$spv(t) = \Delta\omega K e^{-t/T} \tag{1.2}$$

The value of K and T can be found by finding a decaying exponential which minimizes the squared difference between the exponential curve and the actual data. The value of K depends on gaze instructions and with the subject told to "stare straight ahead and do not spot imaginary targets of the periphery", the value of K is typically 0.6. (Collins, 1962)

The problem now is that the firing frequency of the primary semi-circular canal afferents exhibit the following dynamics:

$$\frac{spv(s)}{\omega(s)} = \frac{-kT_1T_2T_a(T_Ls+1)s^2}{(T_2s+1)(T_as+1)} \tag{1.3}$$

where T_1 = short time constant of the quasi steady flow development, T_2 = long time constant of cupola return, T_a = peripheral neuron adaptation, and T_L = rate sensitivity time constant. (Fernandez & Goldberg,1972) However these are the dynamics of the primary afferents, not the VOR. It was hypothesized that there is a neural process going on at the vestibular nucleus, which is often referred to as "velocity storage" (Raphan, Matsuo & Cohen, 1979). The vestibular nucleus processes the information from the semi-circular canals (SSC) and other information regarding motion received from other sources, such as tactile cues, and sends it to the oculomotor nuclei, where the information is used by the oculo-motor system to complete the vestibulo-ocular reflex.

A model that included velocity storage and took into account the above mentioned dynamics of the primary afferents is shown in eqn 1.4 (Oman & Calkins, 1993) . It said that the VOR is determined from a direct pathway from the SSC plus an indirect pathway which includes velocity storage, which is modeled as a leaky integrator. The dynamics are:

$$\frac{spv(s)}{\omega(s)} = \frac{-kT_1T_2T_a(T_Ls+1)s^2}{(T_2s+1)(T_as+1)} \frac{(1+K_vT_v)(\frac{T_v}{1+K_vT_v}s+1)}{(T_vs+1)} \tag{1.4}$$

where K_V and T_V are the gain and time constant of the leaky integrator. The zero associated with the indirect pathway is believed to cancel the canal pole. If the peripheral adaptation is neglected, and terms multiplied by T_1 are neglected since T_1 is small (@ 0.005 seconds) then the model dynamics can be reduced to:

$$\frac{spv(s)}{\omega(s)} = \frac{kT_1s}{T_vs+1} \tag{1.5}$$

which is the transfer function identical to equation 1.1 in form.

It has been seen that the apparent time constant of the SPV profile changes depending on the subjects orientation to the gravity vector. For example if the subject performs a "dumping" maneuver in which the subject pitches his head 90 deg forward from an earth vertical plane to an earth horizontal plane at the start of the post-rotatory period, the time constant is reduced to 41% of the head-erect post-rotatory time constant. (Benson & Bodin, 1966a) Another example is that in parabolic flight, the time constant is reduced to 69% of the 1-G value. (DiZio & Lackner, 1988). It was hypothesized that when there is vestibular conflict the brain relies more on the SSC information and less on the velocity storage information, hence causing the time constant to decrease since the time constant of the SSC is smaller than the velocity storage time constant. The vestibular conflict during a dumping maneuver occurs because the otoliths signal that the head is pitched forward 90 deg, and the SSC signal rotation in an off vertical axis. The vestibular conflict in orbital and parabolic flight occurs because in zero-g the otoliths do not respond in a familiar way and the SSC are telling you that you are spinning. It is believed that this vestibular conflict may cause SMS. (Oman & Calkins, 1993)

1.2 Previous Science Results

There are two categories of rotation experiments that can be performed to test the VOR: These are active rotation, in which the subject is stationary and rotates his head, and passive rotation in which the subject is seated in a rotating chair and spun. In active rotation experiments it is hard to determine the time constant. Active movements may not elicit a "true" reflex since eye movements may be influenced by feed forward, "efference copy" signals in the central nervous system (CNS). One study using active movements showed a decrease in gain (Clement, 1985), another an increase in gain (Kornilova, et. al, 1993), and three other experiments showed no significant change. (Watt, 1985 & Benson, 1986 & Thorton, 1989).

Passive rotation experiments were performed before and after 4 previous missions, and in orbit on 2 earlier shuttle flights. The general form of the experiments were a one minute 120 deg/sec rotation, with collection of data one minute after rotation. Some experiments had some runs with dumping, others did not. One problem when testing for significant changes in the VOR, is that the standard deviation of the VOR parameters is approximately 20 - 30% of the mean. This means that large changes must occur in order to significantly determine if any changes occurred. The post-flight results for the previous missions are in Table 1.1, and in-flight in Table 1.2.

Mission	Results	Reference
SL-1:	Time constant decreases	(Oman, Kulbaski, 1988)
	Time constant decreases	(Liefield, 1993)
	Gain increase trend	
	No significant change in dumping time constants	
	Strength of motion sickness related to adaptation	
D-1:	No significant change in gain	(Oman, Weigl, 1989)
	Time constant decreases	
SLS-1:	Post-rotatory dumping inconsistent	(Oman, Balkwill, 1993)
	Post-rotatory head erect VOR were inconsistent	
IML-1:	No significant change in time constant or gain	
	but trend was towards increased gain and	
	decreased time constant	(Oman, Calkins, 1993)

Table 1.1: Post-flight results for passive testing on previous missions

Mission	Results	Reference
SL-1:	No usable in-flight data obtained	
D-1:	Experiment not performed	
SLS-1:	3/4 subjects head-erect and dumping time constants increased compared to pre-flight head erect data No evidence for change in gain	(Oman, Balkwill, 1993)
IML-1:	Gain increases and Time constant decreases and were correlated to SMS	(Oman, Calkins, 1993)

Table 1.2: In-flight results for passive testing on previous missions

1.3 Previous Engineering Results

On earlier flights (SL-1, SLS-1), the strategy of the SPV data analysis has been to first to take raw eye position and convert it to slow phase velocity. Then two paths have been chosen to further analyze the SPV data. The first path is to fit the SPV data with a model (henceforth called "parametric analysis") and then average and normalize the pre-flight model parameters and the post-flight model parameters, and use a t-test or analysis of variance to test to see if there are statistically significant changes in the model parameters. The second path (henceforth called "ensemble averaging") involved averaging of the SPV time series for two different groups of data. Then statistical tests are performed to determine whether there are significant differences between comparison groups.

In the SL-1 analysis done by Oman and Kulbaski, an acceleration based algorithm was implemented (Massoumnia, 1983), in which the fast phase of nystagmus was detected and then removed from differentiated eye velocity. A first order exponential model was fit to the data by taking the log of the SPV data and performing a linear regression. The data was also ensemble averaged and a chi-square statistical method was used to find statistical differences between selected subsets. A chi-square test is a way to tell if two ensemble averaged time series are significantly different from each other. To do this a chi-squared value is calculated.

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - x_i)^2}{\sigma_p^2} \qquad (1.6)$$

where χ^2 is the chi-square statistic, σ_p^2 is the pooled variance, N is the number of degrees of freedom, and x_i and y_i are the ensemble averaged time series being compared at time i. Then this value is compared to a chi-square distribution. If the calculated chi-square value is much greater than N then the average squared difference between the two time series is greater than the pooled variance. Since the distribution of chi-square is known, it is possible to test whether the two time series are significantly different. The assumptions that the chi-square test makes are that the number of degrees of freedom is large, and the successive samples are not correlated. (Kulbaski, 1986)

In the D-1 analysis, the eye position was obtained be electronic differentiation of low pass filtered horizontal EOG data. Then the average SPV was found by manual sampling from the eye velocity chart records at 0.5 second intervals. The differentiator gain was calibrated by reference to the EOG position signal. Gain and time constants were not calculated. For the statistical tests, the SPV time series were ensembled averaged compared using the same chi-square method. (Oman & Weigl, 1990)

In the SLS-1 analysis a new method to determine the slow phase velocity was used. The problem with the Massoumnia acceleration based algorithm was that 10-50% of the fast phase beats were missed, and the data had to be manually edited to remove these artifacts. A completely automated algorithm - order statistic filtering (Engelken, Stevens, 1990) was adopted. First the position data is filtered using a predictive median hybrid filter, which removes much of the high frequency noise. Second the data is digitally differentiated. Lastly the data is filtered using an asymmetrically trimmed mean statistical filter, which extracts the SPV envelope by assuming that the eye spends more time in the slow phase than in the fast phase. It was found that this algorithm worked well. It was found, however, that dropouts, which were when the eye instantaneously goes to zero velocity for reasons such as drowsiness and lack of attention, still remained. An automatic dropout detection algorithm based on a logarithmic linear regression was developed. (Balkwill, 1992)

The remaining data was ensembled averaged, but it was determined that the chi-square test may not have been sufficiently conservative. For a low number of runs, which is inherent in space physiology, the sum of the differences squared will not be distributed like a chi-square distribution. It will be distributed like a sum of t^2 distribution. The problem with this is that sum of t^2 distribution had not been calculated, so a monte carlo simulation was set up to determine the distribution. (Pouliot, 1991)

A Raphen-Cohen model was fit to the data, and averaged, with statistical significant differences found using a t-test. (Oman & Balkwill, 1993)

After the SLS-1 analysis it was desired to re-analyze the D-1 and SL-1 data sets using OS-filters. The SL-1 data set was selected to be analyzed first. D-1 re-analysis is planned for the future. OS filters were used to calculate slow phase velocity, and the automatic dropout detection algorithm was used. Ensemble averaging and testing using a sum of t^2 test was utilized. Models that were fit to the data included a first order exponential model, velocity storage/adaptation model, and a fractional exponent adaptation model (Landolt and Correia, 1980). The analysis pipeline from eye position to model fitting was quasi-automated (Liefield, 1993)

The IML-1 analysis was somewhat similar to the SLS-1 analysis. OS filters were used to calculate slow phase velocity, and the same automatic dropout detection algorithm was used. Instead of using more complicated models like the Raphen-Cohen model, only a first order exponential model was employed, using a log-linear regression method. The rationale behind this was that in the SL-1 re-analysis, it was found that the more free parameters a model had, the more variable the parameters, and the more difficult to determine statistical significance (Liefield, 1993). So since the VOR response is quasi-exponential it was believed that a simple exponential should capture the major trends of the data. The gain and time constant from this model were normalized with respect to the subject, run direction, and per or post rotatory phases, based on the pre-flight mean value, in order to combine subjects to increase the apparent number of runs. Then a two-way ANOVA with replications was used to find statistically significant differences. (Oman, Calkins, 1993)

In all the previous missions to date, the amount of time to process the data was enormous. This was in part due to the limited power of the computers at the time, and part to lack of automation of the analysis pipeline. It would take many months after the mission in order to even have a preliminary idea of what the results would be. In this analysis it was desired to make the analysis pipeline work near real-time, so that if the astronauts or investigators wanted to, they could look at the data after testing to see if responses were back to normal. Also if any decisions needed to be made about how to spend the limited available post-flight testing time, an informed decision, based on the data, could be made. For example if in the early post-flight period, the NASA administrators tell experimenters that they have only a certain amount of astronaut testing time, time could be better allocated to subjects, and or testing conditions, which would optimize the testing, since the experimenters would have some sense of how the subjects were performing in the experiment.

1.4 Goals of Thesis Research

The goals of this thesis were to analyze and draw conclusions of a very large database of EOG data taken from the SLS-2 mission of the space shuttle, and to continue the evolution of the analysis and statistical algorithms so that the analysis and preliminary statistical testing can be completed in near real time. The important science related question were:

1) How did subject's pre-flight head erect VOR responses compare with results from previous studies ?
2) How did the head-erect SPV response change in parabolic flight ?
3) How did the head-erect SPV response change in prolonged weightlessness ?
4) How did the head erect SPV response change post-flight ?
5) Was the pre-flight dumping response consistent with previous studies ?
6) How did the dumping SPV response change in prolonged weightlessness ?
7) How did the dumping SPV response change post-flight ?
8) Did near-real time SPV analysis facilitate decision making during testing ?
9) Are there any recommendations for future analysis and future studies ?"

Chapter 2: Experimental Methods

2.1 Subjects

Subjects used in this experiment were five crew members of the STS-58 Space Life Sciences - 2 mission of the Space Shuttle Columbia. There were 3 males and 2 females ranging in age between 36 and 51. The subjects were assigned a random subject code to maintain confidentiality. Their codes were T, V, W, X, and Y. All subjects were free of any vestibular disease. However subject T had esophoria, reduced acuity in the right eye, and a history of childhood strabismus surgery in that eye. For two of the subjects this was their first time in space.

Pre-flight testing occurred on four separate days. They were L - 122 (BDC1, meaning base-line data collection session #1), L - 110 (BDC2), L - 88 (BDC3), and L- 18 (BDC4), where for example "L - 122" means 122 days before launch. "Early post-flight testing" occurred on R+0 (BDC5), R+1 (BDC6), R+2 (BDC7), where for example "R+2" means 2 days after landing. "Late post-flight testing" was performed on R+6 (BDC8), R+9 (BDC9), R+11 (BDC10). All pre- and post-flight testing took place at Johnson Space Center in Houston Texas. Not every subject was tested on each testing day and table 2.1 shows which days each subject was tested.

One of the subjects who had flown previously had been tested on the earlier mission. For this subject, the preflight database included two sessions (L-122, L-18) from this flight, and four preflight sessions from the previous flight.

In-flight testing occurred on two different days. They were flight-days 4 and 10. The testing occurred in the Spacelab.

Parabolic flight testing occurred on L-129. The testing occurred on a NASA KC-135 aircraft.

Subject W was not available for in-flight and parabolic flight testing, and this subjects' preflight and post-flight VOR responses were highly variable, with a high proportion of dropouts, because the subject often found the testing provocative. Because of this high variability, the likelihood of a statistically significant pre/post-flight comparison was deemed small. Since the focus of this thesis was the comparison between preflight, parabolic flight, in-flight, and post-flight results, Subject W's data was not included in this analysis.

Sub.	L-122	L-110	L-88	L-18	R+0	R+1	R+2	R+6	R+9	R+11
T	x			x	x	xx		x	x	x
V	x	x	x	x		x	x	x	x	x
W	x	x	x	x	x	x	x	x	x	x
X		x	x	xx		x	x			
Y	x	x	x	x	x	x	x	x	x	x

Table 2.1: Ground Test Matrix (x = single test session, xx = double test session)

Subject	KC-135	Flight Day 4	Flight Day 10
T	x	x	x
V	x	x	x
X		x	x
Y	x	x	x

Table 2.2: KC-135 and Flight Test Matrix (x = single test session)

2.2 Ground Equipment

Ground testing was conducted using a motorized, computer-controlled rotating chair. The chair could accelerate/decelerate a Δw of 120 deg/sec within 0.5 seconds. The setup was capable of maintaining a chair speed of 120 deg/sec within +/- 2.5 deg/sec.

Eye movements were recorded by an electro-oculography (EOG) system which included neo-natal electrodes, an EOG amplifier (10 x), a low-pass filter (3rd order, 30 Hz cutoff), and a Macintosh IIci computer, running the Labview software package developed for data acquisition (Liefield, 1993), which sampled the data (120 Hz).

A tape recorder was used to record voice conversations for each testing session. For in-flight, parabolic flight, and early post-flight sessions a video-record of the session was made.

2.3 Ground Experimental Protocol

The procedure used was similar to that on SLS-1. Subjects were fitted with 5 electrodes at least fifteen minutes prior to testing. This allowed ample time for the electrodes to settle. The horizontal EOG signal was measured by two electrodes, one placed on each outer canthus. The vertical EOG signal was measured by two electrodes, one placed below the right eye and one above the right eye. Both the horizontal and vertical EOG signals were referenced to an electrode placed on the neck.

The subject was then seated in the rotating chair and given red goggles to wear, whose purpose was to stabilize the corneo-retinal potential. The subject was also given headphones to wear that would allow the researchers to communicate with him while minimizing ambient sound cues.

Gaze instructions were then given to the subjects. They were told to keep their eyes open and try to stare straight ahead and try not to spot imaginary targets on the walls of the room.

There were four testing options. They were calibrations, spontaneous nystagmus checks, head-erect runs, and dumping runs.

- A calibration consisted of the subject facing a wall. The wall mounted targets subtended +/- 10 degrees both horizontally and vertically. The subject was then told to look at the calibration targets in the following order: Center - Right - Center - Left - Center - Up - Center - Down - Center - Right - Left - Right - Left -Right - Left - Right - Left - Right - Left - Right - Left.

- A spontaneous nystagmus check consisted of having the subject place covers over the goggles and then having the subject stare straight ahead for ten seconds and then close his eyes and stare straight ahead with his eyes open for ten seconds.

- A head-erect run consisted of the chair spinning at 120 deg/sec for 60 seconds and then the chair would stop and data collection would continue for the next 60 seconds (Figure 2.1). The subject would hold his head so that his neck was straight and the head was in-line with the torso of the body for the entire 120 seconds. The subject was instructed to say "start" when he felt the chair start, "gone" when the direction of spinning sensation was ambiguous, "stop" when he felt the chair stop, and "gone" when the sensation of spinning was ambiguous.

- A dumping run was like the head-erect run except when the subject feels the chair stop he would say "stop" and the operator would say "down" and the subject would pitch his head 90 degrees forward, and hold it there for the 60 second post-rotatory period. (Figure 2.1)

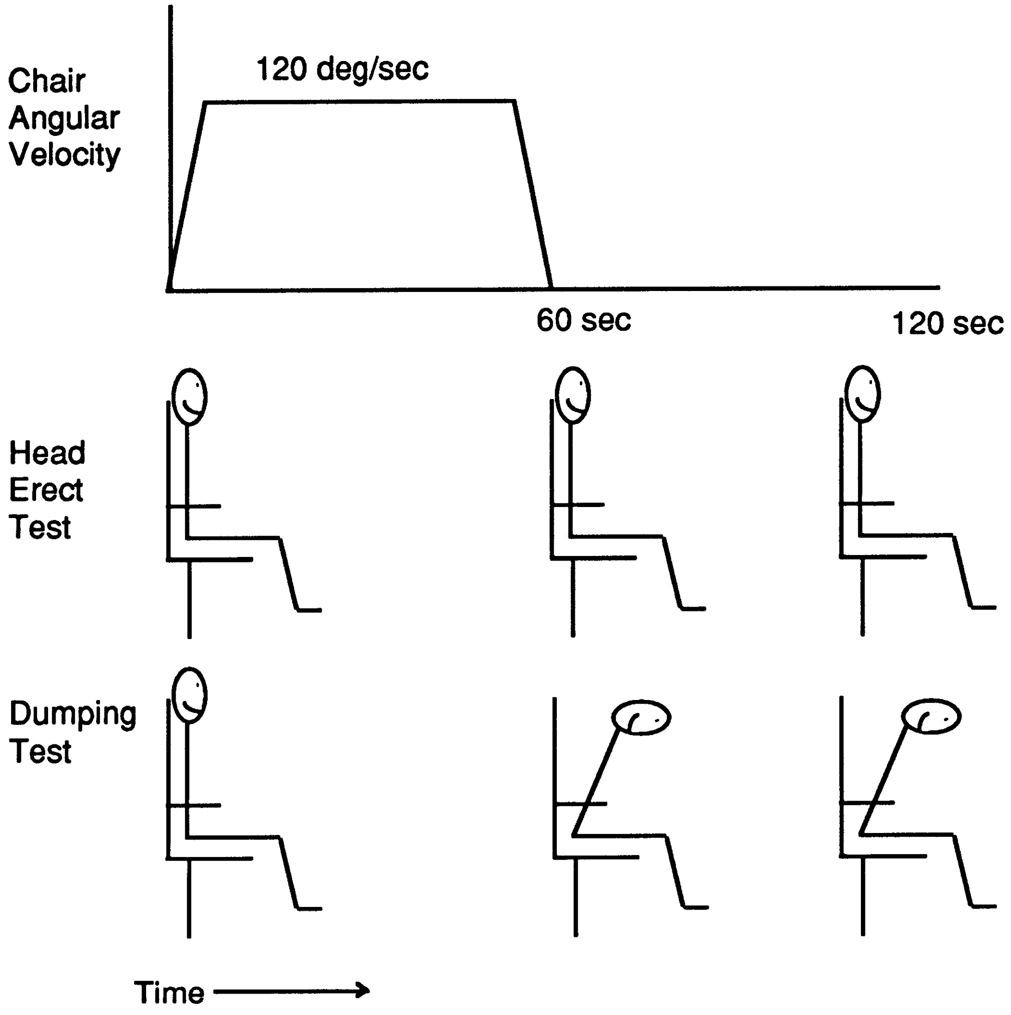


Figure 2.1: Schematic of Head Erect and Dumping Test Procedures

Each time the subject would say "gone" the operator would press a button which would add a pulse to the tachometer signal which would enable the researchers latter to know when the subject said "gone". The delay between the subject saying "gone" and the operator pressing the button was on the order of 0.5 seconds

Both clock-wise and counter-clockwise runs were performed. The direction of the run was alternated between runs in order to balance residual

effects of the previous run on the current run since the long time constant of semicircular canal neural adaptation is longer than 60 seconds.
The nominal experimental protocol was as follows:

Run	01:	Calibration
	02:	Spontaneous Nystagmus Check
	03:	CW head-erect run
	04:	CCW head-erect run
	05:	CW dumping run
	06:	CCW dumping run
	07:	Calibration
	08:	CW head-erect run
	09:	CCW head-erect run
	10:	CW dumping run
	11:	CCW dumping run
	12:	Calibration

Run codes were created for computer data files by the following file specification format: SBRR, where S = subject code, B = sequential BDC number, and RR = run number. For example T204, would represent one of subject T's CCW head-erect runs on BDC2 (L-45). KC-135 data had the file specification format: SPBRR, where P indicates that the data was taken in parabolic flight. In-flight data had the file specification format: SFBRR, where F indicates flight data.

2.4 Differences in Experimental Methods for In-flight and KC-135 Testing

There were several differences between the methods of the in-flight and KC-135 testing vs. the ground testing.

For the in-flight testing the differences were:

a) A manually rotated chair used. The chair was a lightweight Body Restraint System, originally built for 1983 European Vestibular Experiments on Spacelab-1 (Kass, et al, 1986) which was manually spun to accelerate to the desired speed of 120 deg/sec within 1 second and maintain that speed usually within 10 %. The subjects sat with their legs crossed. A belt mounted digital tape recorder (the Cassette Data Tape Recorder) was used for data collection instead of a Macintosh. 2 and 4 K EOG amplifiers were used instead of 10 K. The data was sampled at 100 Hz instead of 120 Hz. There was no middle EOG calibration. There was no tachometer. The angle between the target and the subjects eyes was 9 deg instead of 11.7 deg

For the KC-135 testing the differences were:

a) A lightweight, motorized, servo controlled chair was used. The chair had been developed for similar experiments on Spacelab IML-1 (Oman and Calkins, 1993). The subject's head was restrained by a helmet, which necessitated monocular (right eye) rather than binocular EOG calibrations, and prevented testing using "dumping" head movements. The angle between EOG calibration targets was 10 deg instead of 11.7 deg. The EOG amplifiers had 2 K gain instead of 10 K gain.

b) Parabolas were flown sequentially, providing alternating periods consisting of approximately 20-25 seconds of 0-G, preceded by a variable interval of 1.0 - 1.8 G. For operational reasons, it was not possible to precisely control the duration of the hypergravic interval. During this time, the chair was smoothly accelerated to 120 deg. sec in 2 seconds. Immediately after 0-G was established, the chair was decelerated in 2 seconds to a stop, and EOG recording continued for the remaining interval of 0-G. The per-rotatory interval was 46.0 sec (± 5.9 sec SE), and post-rotatory data was recorded for 22 seconds beginning at the onset of 0-G as determined by an accelerometer. Model fitting was based upon 19 seconds of data beginning 3 seconds after the onset of 0-G. The three second delay was incorporated to allow time for the operator to bring the chair to a halt. 1-G control runs were performed with the aircraft on the ground, just prior to flight, using a similar per-rotatory interval (51.4 sec. ± 1.6 sec SE) and an identical 22 sec. post-rotatory recording period. All subjects took scop/dex to reduce their susceptibility to motion sickness and thus maintain alertness.

Chapter 3: Data Analysis

3.1 Calibration of Eye Position Data

To determine the calibration factors from A/D units to degrees of eye movements, an interactive algorithm was used. (Balkwill, 1992) The horizontal eye position data from the calibrations was shown on the computer screen. Then the expert-user marked 5 regions where the subject was fixating on the right target and then regions where the subject was focusing on the left target. An average value in A/D units was calculated for the right target and then for the left target. The known angle ABC (20 deg),where A = left target, B = subject's head, C = right target was divided by the difference in values between the left and right targets in A/D units This value was the calibration factor for the calibration run. Usually three calibration runs were performed per session. To get a more accurate calibration factor for each individual run a linear interpolation was performed between calibration runs. This was an estimate of the calibration factor for the run. (Balkwill, 1992)

3.2 Calculation of Slow Phase Velocity

Two order statistic filters and one linear filter was used to calculate the slow phase velocity from the eye position data. An order statistic filter is a non-linear digital filter which uses a sliding window of odd-length of the input data samples. The input data samples are rank-ordered and the ordered samples are linearly weighted. A linear combination of the ordered statistics constitutes the filter output. The first ordered statistic filter is called a predictive finite-impulse response median hybrid filter (PFMH) which smoothes the input data while preserving saccadic transitions. The linear filter is differentiating filter whose purpose is to calculate eye velocity from eye position. The second order statistic filter is called an adaptive asymmetrically trimmed mean filter (AATM) which extracts the slow-phase velocity from the total eye velocity (Figure 3.1) (Engelken & Stevens, 1990)

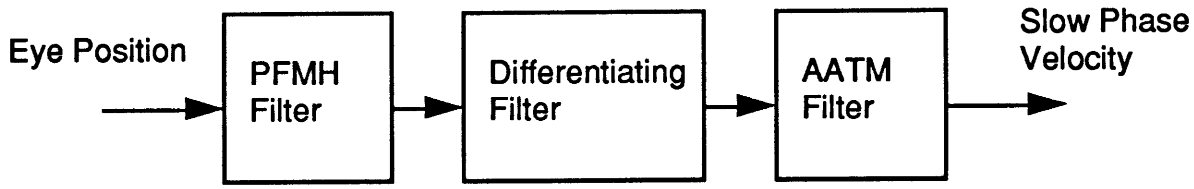


Figure 3.1 Block Diagram for calculation of slow phase velocity from eye position

3.2.1 PFMH Filter

The PFMH filter uses an odd length sliding window of length 2N + 1. The first N samples are used to calculate the forward predictor, X_f. The last N samples are used to calculate the backward predictor, X_b. For a first order FIR predictor the N samples are given weights that linearly depend on its position, with the sample farthest from the N+1st sample given the least weight, and the sample closest to the N+1st sample given the greatest weight. The median of the forward predictor, backward predictor, and the N+1st sample is the output of the filter.

This filter assumes the data is a combination of a root signal and some undesired noise. Repeated use of the PFMH filter will remove the noise, however for data-sets with a high noise/signal ratio the root signal obtained may not be the desired root signal. (Engelken, 1990)

For this experiment two filters were used of lengths N=6 and N=10, and each filter made two passes on the data. The values for N were the ones used on SLS-1 and were based upon the duration of the fast phases being around 50-100 ms.

3.2.2 Calculation of Eye Velocity.

Eye velocity was calculated from PFMH filtered eye position by using a linear nine point FIR velocity filter consisting of a three point differentiating filter convolved with a seven point low pass filter with a 10 Hz corner frequency (Massoumnia, 1983).

3.2.3 AATM Filter

The key assumption in the AATM filter is that the eye spends more time in the slow-phase than in the fast-phase. The filter uses this fact to extract the slow-phase velocity from the total eye velocity. A frequency histogram is calculated in the window and the dominant mode is assumed to indicate the SPV. The output of the filter is an average of the velocity samples in the neighborhood of the dominant mode. (Engelken, 1990)

3.3 Tachometer Analysis

The tachometer signal needed to be analyzed in order to extract some necessary information with respect to rotation of the chair. This information was the delay, spin length, run length, and spin velocity. An algorithm was developed to find these important parameters. (Pouliot, 1992) The algorithm basically looked for large changes in the tachometer signal which will occur at the chair start and stop. The first large change will be assigned the chair start and the second will be assigned the chair stop. From this information, the rest of the parameters can be calculated.

For the in-flight data collection, there was no tachometer signal, so the start and stop points were determined manually by looking at the record of the chair yaw accelerometers, and looking at the EOG data and looking for the first saccades.

3.4 Outlier Detection and Removal

The length of each per-rotatory and post-rotatory section had to be normalized to 60 seconds. If the run segment was less than 60 seconds then the median of the previous 5 samples was added to make the run segment 60 seconds. If the run segment was greater than 60 seconds then data beyond 60 seconds was truncated. In order to remove dropouts, an automated algorithm was developed. (Balkwill, 1992) The basic idea in this method is to look at the natural logarithm of the data and then perform a least squares linear fit. The root-mean square error (RMS) about the line was calculated, and any data point greater than 3*RMS or below 7.4 deg/sec, along with all points within 1/2 second, were marked as bad. A new linear fit was performed based upon the data with the previous bad sections ignored. However the RMS is calculated based upon the whole data-set, and then the process is repeated. When the RMS of the current fit converges to within 20% of the previous RMS value then the bad sections are designated as dropouts.

3.5 Decimation

It was desired to decimate the data from a sampling rate of 120 Hz down to 4 Hz in order to save computation time and disk space. In order to do this the 120 seconds of 120 Hz data was broken down into 30 equal segments and then the mean and variance of each segment was calculated. This information was saved as the decimated data. (Balkwill, 1992) In a similar manor, the in-flight 100 Hz data was decimated down to 4 Hz.

3.6 Model Fitting

It was desired to fit a first order exponential to the decimated data. To do this constrained optimization was used in order to obtain the fit that minimized the RMS between the data and the model fit. (Liefield, 1993) The per and post rotatory data were fit separately since the per and post rotatory segments should be asymmetric due to the long adaptation time constant being around 80 seconds. The model thus yielded a gain and time constant for both the per and post rotatory phases

3.7 Run Segment Rejection Criteria

In some cases, dropouts can be prolonged, and the model fit is based on a relatively small sample of SPV data. It is probably inappropriate to average the model fits based on small amounts of data, runs containing large electrode motion related SPV artifacts, or responses where SPV response was virtually absent due to fatigue with other data where this was not the case. A quantitative run-rejection criterion was developed, and applied uniformly across all subjects and all conditions. For some of the previous missions a quantitative run rejection rule was not used. For example in SLS-1 the run rejection criterion was "the SPV envelopes were examined visually, and discarded if appropriate." (Balkwill, 1992) This could conceivably have lead to an added bias due to experimenter prejudice.

In determining the criterion, histograms of the parametric data were examined. It was determined that atypical run segments with extreme outliers or a virtually absent per- or post-rotatory response could usually be identified by using the following criteria:

1) The model-fitting routine hit a lower bound. ($K < .06$) or ($T < .15$)
2) Percent good of a run segment was less than 60%.
3) Mean-square error was greater than 200.

The rationale behind the first criterion was that if the model fitting routine hit a lower bound then the run parameters were not physiological realistic for a 120 deg/sec step response. This might occur for example if the subject showed little response for most of the run. (figure 3.2)

The rationale behind the second criterion was that if over 40% of the first 25 seconds of the run segment had been discarded by the outlier detection algorithm than not enough data is there to make any further conclusions based upon this run segment. (figure 3.3)

The rationale behind the third criterion was that if the mean-square error was greater than 200 than the data was extremely noisy and should not be used. (figure 3.4)

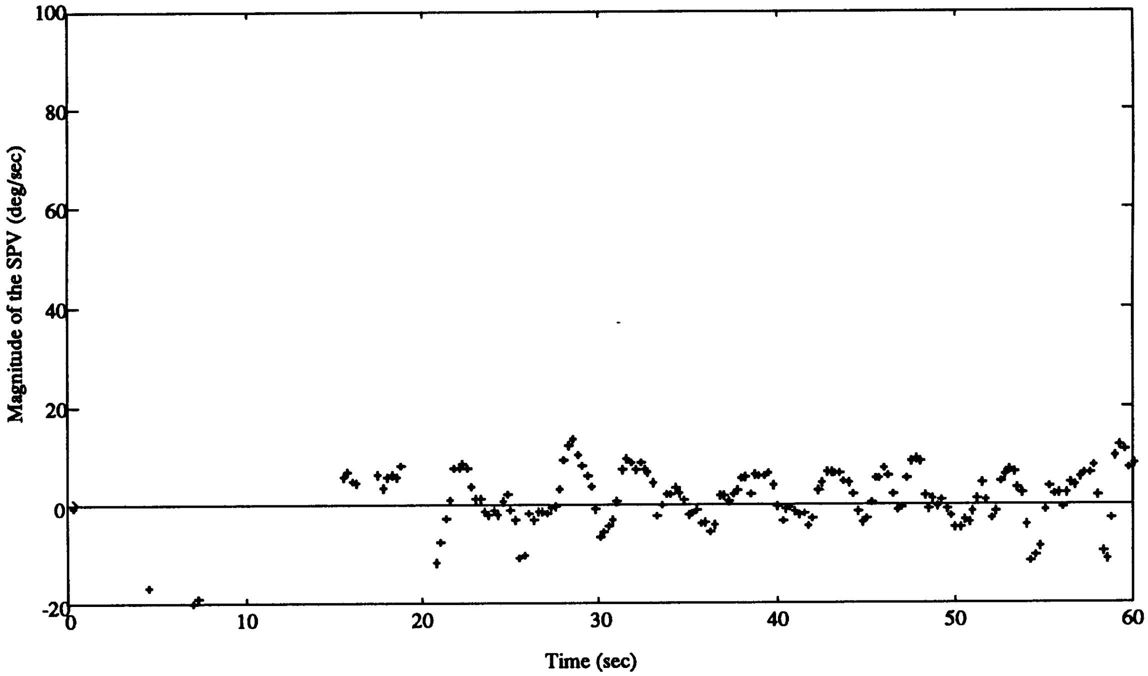


Figure 3.2: Example of model-fitting routine hitting a lower bound (Model fit is the black line). Low frequency SPV oscillations are typical of a sleepy subject.

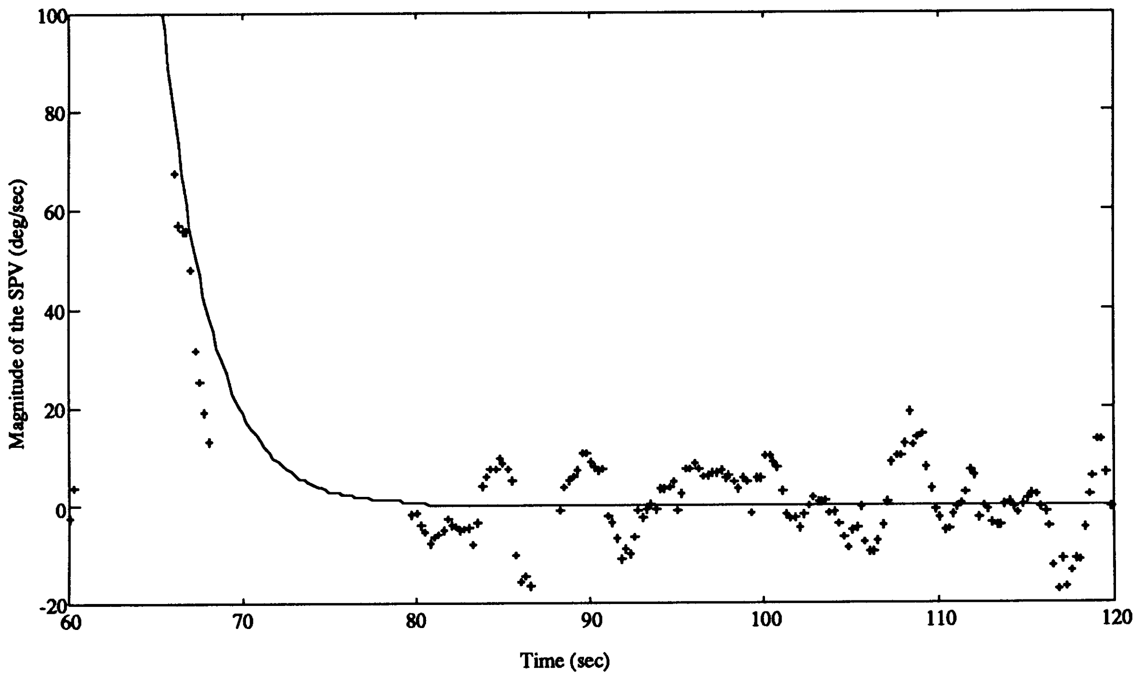


Figure 3.3: Example of % good < 60 % (Actual % good = 35 %)(Model fit is the black line)

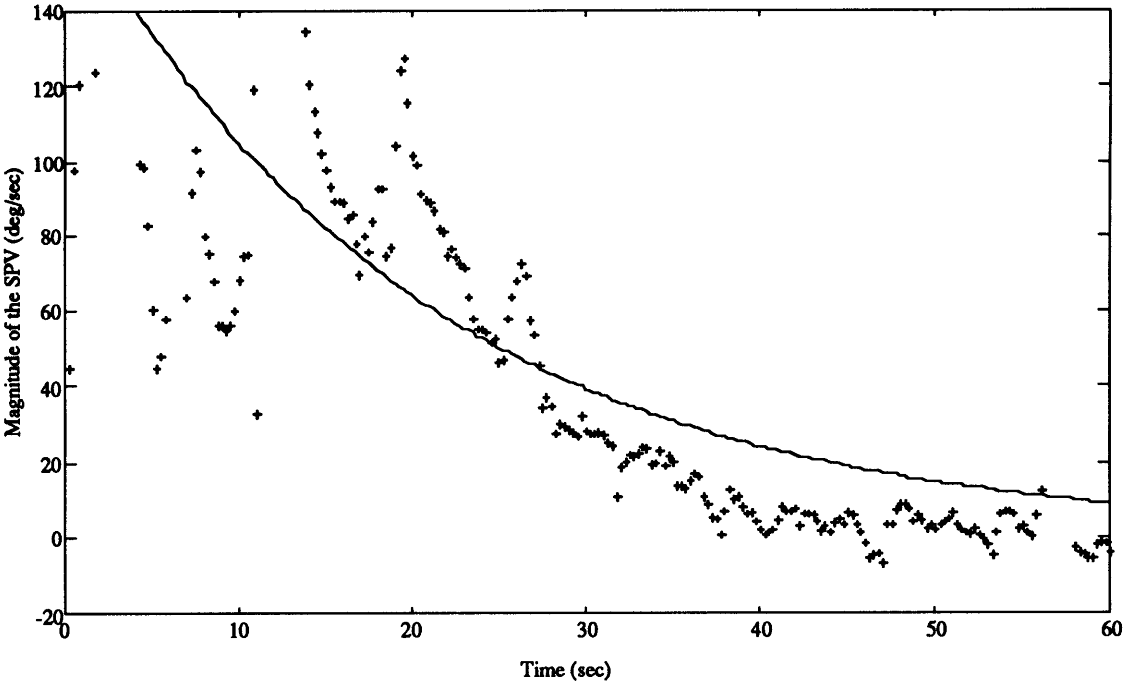


Figure 3.4: Example of MSE > 200 (Actual MSE = 1370)(Model fit is the black line)

It was decided that if one of the run-rejection criteria was met, than both the gain and the time-constant of the run-segment should be thrown out due to the direct relationship between the gain and time-constant.

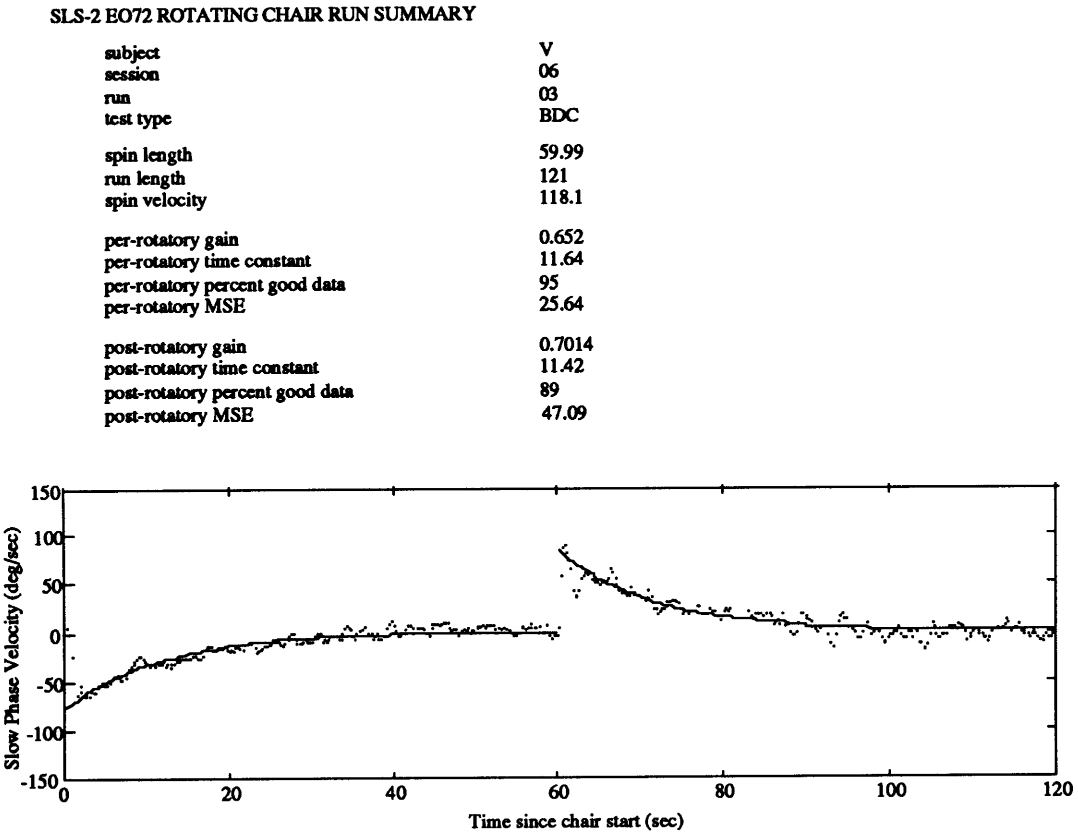
These criterion differed a little from the previous missions in which a quantitative run rejection criterion was used. In IML-1 (Oman & Calkins, 1993), the run rejection criterion was to remove all outliers in the histogram distribution greater than 2 SD's. But in looking at the individual runs it was found that some runs that were in the middle of the distribution were unreliable since for example, if there were a lot of dropouts, the model fitting routine could give unreliable results. In the SL-1 re-analysis (Liefield, 1993), the criterion was the following:

1) more than 10 seconds worth of data was removed from a run segment.
2) The model fitting routine hit a constraint (.18 < K < 1.8) or (5 < T < 45)
3) Peak SPV response was below 25 deg/sec.

The first criteria is the same the SLS-2 criterion #2, since 40% of 25 seconds (the number of seconds the % good statistic was based on) is 10 seconds. The second criteria is similar to the SLS-2 criteria #1, however it was decided to reduce the lower bound in order to not throw away low gain data, and .06 is very much below what is considered to be physiologically reasonable. The third criterion was not used, for the same reason why the lower bound was reduced, for it was not desired dismiss low gain responses, and in Liefield's thesis, as an afterthought he questioned the justification of this rule.

3.8 Near real-time statistical analysis

It was desired to calculate some statistical parameters in near real time during the post-flight BDC sessions in order to obtain a basic understanding of how the subjects were performing in the experiment. Two different ways of looking at the data in near real-time were developed. The first was to look at individual run reports (Figure 3.5), as was done by Oman and Calkins (1993), except that the results were fit by a model (Eqn 1.5), superimposed on the data, and the mean square error was also calculated. The second way was to calculate histograms (Figure 3.6), means, and variances, of the parametric data.



SLS-2 EO72 ROTATING CHAIR RUN SUMMARY

subject	V
session	06
run	03
test type	BDC
spin length	59.99
run length	121
spin velocity	118.1
per-rotatory gain	0.652
per-rotatory time constant	11.64
per-rotatory percent good data	95
per-rotatory MSE	25.64
post-rotatory gain	0.7014
post-rotatory time constant	11.42
post-rotatory percent good data	89
post-rotatory MSE	47.09

Figure 3.5: Example of a run report for one of subject V's early post-flight runs.

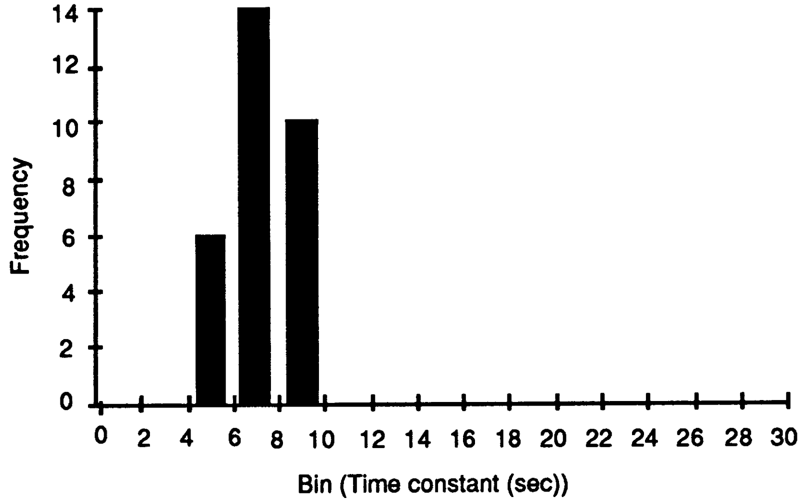
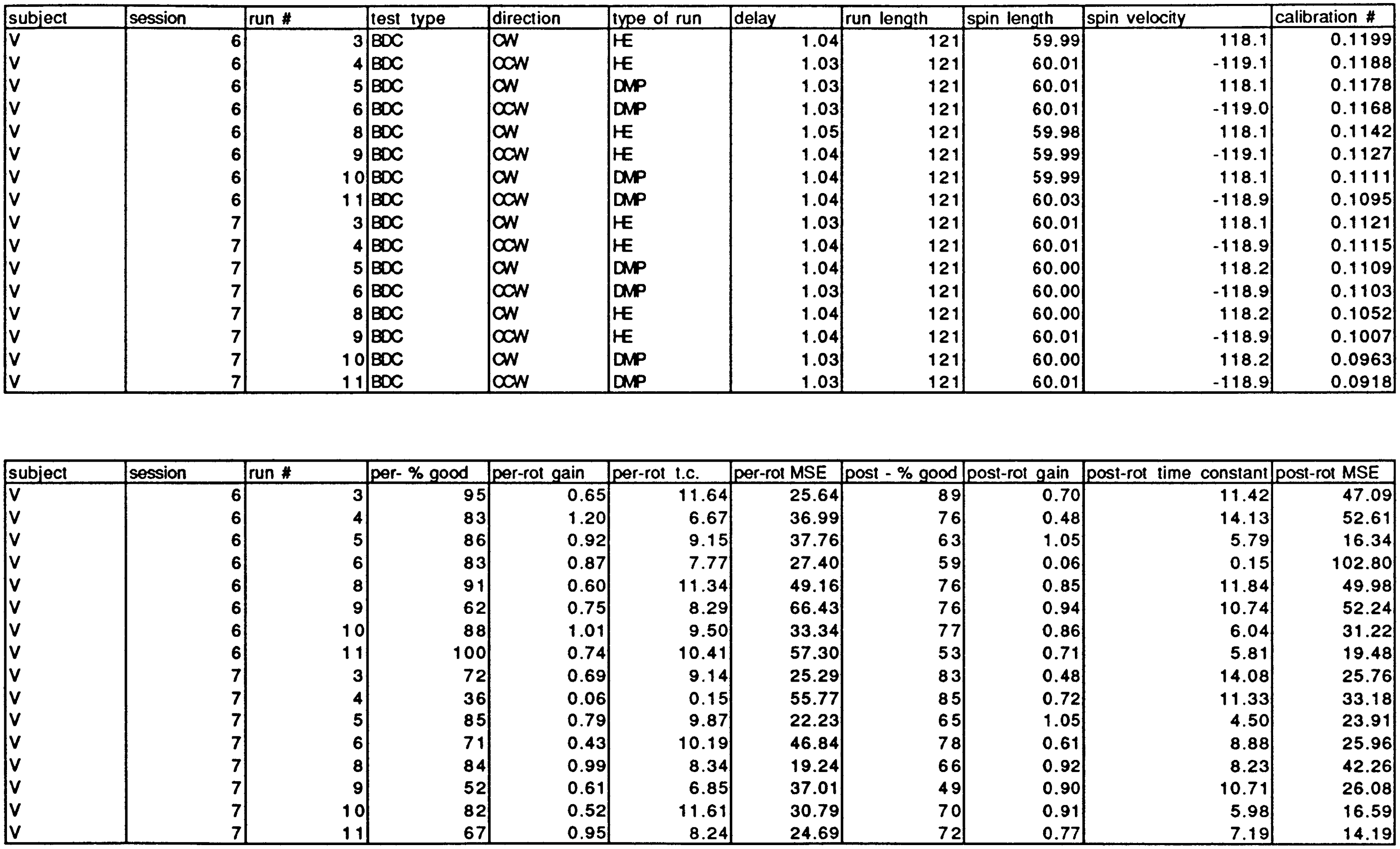


Figure 3.6: Example of a histogram for Subject V's early post-flight per-rotatory time constants.

The calculation of the histograms, means, and variances was performed by exporting the parametric information describing each run from Matlab to



subject	session	run #	test type	direction	type of run	delay	run length	spin length	spin velocity	calibration #
V	6	3	BDC	CW	HE	1.04	121	59.99	118.1	0.1199
V	6	4	BDC	CCW	HE	1.03	121	60.01	-119.1	0.1188
V	6	5	BDC	CW	DMP	1.03	121	60.01	118.1	0.1178
V	6	6	BDC	CCW	DMP	1.03	121	60.01	-119.0	0.1168
V	6	8	BDC	CW	HE	1.05	121	59.98	118.1	0.1142
V	6	9	BDC	CCW	HE	1.04	121	59.99	-119.1	0.1127
V	6	10	BDC	CW	DMP	1.04	121	59.99	118.1	0.1111
V	6	11	BDC	CCW	DMP	1.04	121	60.03	-118.9	0.1095
V	7	3	BDC	CW	HE	1.03	121	60.01	118.1	0.1121
V	7	4	BDC	CCW	HE	1.04	121	60.01	-118.9	0.1115
V	7	5	BDC	CW	DMP	1.04	121	60.00	118.2	0.1109
V	7	6	BDC	CCW	DMP	1.03	121	60.00	-118.9	0.1103
V	7	8	BDC	CW	HE	1.04	121	60.00	118.2	0.1052
V	7	9	BDC	CCW	HE	1.04	121	60.01	-118.9	0.1007
V	7	10	BDC	CW	DMP	1.03	121	60.00	118.2	0.0963
V	7	11	BDC	CCW	DMP	1.03	121	60.01	-118.9	0.0918

subject	session	run #	per- % good	per-rot gain	per-rot t.c.	per-rot MSE	post - % good	post-rot gain	post-rot time constant	post-rot MSE
V	6	3	95	0.65	11.64	25.64	89	0.70	11.42	47.09
V	6	4	83	1.20	6.67	36.99	76	0.48	14.13	52.61
V	6	5	86	0.92	9.15	37.76	63	1.05	5.79	16.34
V	6	6	83	0.87	7.77	27.40	59	0.06	0.15	102.80
V	6	8	91	0.60	11.34	49.16	76	0.85	11.84	49.98
V	6	9	62	0.75	8.29	66.43	76	0.94	10.74	52.24
V	6	10	88	1.01	9.50	33.34	77	0.86	6.04	31.22
V	6	11	100	0.74	10.41	57.30	53	0.71	5.81	19.48
V	7	3	72	0.69	9.14	25.29	83	0.48	14.08	25.76
V	7	4	36	0.06	0.15	55.77	85	0.72	11.33	33.18
V	7	5	85	0.79	9.87	22.23	65	1.05	4.50	23.91
V	7	6	71	0.43	10.19	46.84	78	0.61	8.88	25.96
V	7	8	84	0.99	8.34	19.24	66	0.92	8.23	42.26
V	7	9	52	0.61	6.85	37.01	49	0.90	10.71	26.08
V	7	10	82	0.52	11.61	30.79	70	0.91	5.98	16.59
V	7	11	67	0.95	8.24	24.69	72	0.77	7.19	14.19

Figure 3.7: Example of an Excel spreadsheet

Microsoft Excel. This information included for both the per and post-rotatory segments: subject (T, V, X, or Y), BDC # (1-10), chair direction (cw or ccw), post-rotatory head condition (he or dmp), percent good, gain, time constant, and mean square error. (Figure 3.7)

Then templates were made for each subject in Excel so that histograms, means, and variances could be calculated. A template is a Excel worksheet with all the columns and rows labeled, the formulas already calculated, and the graphs set so when the data is imported all the statistical calculations and histograms could be performed. The templates for each subject was made for pre-flight, early post-flight, and late post flight in order to calculate histograms, means, and variances for different subsets of the data. Preflight included BDC's 1-4, early post-flight BDC's 5-7, and late post-flight BDC's 8-10. The subsets were per-rotatory gain, per-rotatory time-constant, head-erect post-rotatory gain, head-erect post-rotatory time-constant, dumping post-rotatory gain, dumping post-rotatory time constant.

Using the macro programming language in Excel was studied but it was determined to be not feasible to use since it did not allow any flexibility of a real programming language but rather it was a string of pre-determined commands. In the future when a more suitable macro programming language becomes available it would be better to use this rather than the templates.

3.9 Statistical Comparisons of the Data

3.9.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) non-parametric test determines if two cumulative frequency distributions (A, B) came from the same population. It asks how probable it is that the greatest observed difference in their respective cumulative frequency distributions (CFD):

$$ks = \max\left|CFD(B) - CFD(A)\right| \qquad (3.1)$$

would occur if the distributions A and B, were really the same. (Siegel, 1956) The KS test was used to test pairs of sequences of parametric data, and pairs of SPV time series. The KS test can (in part) determine if one curve decays faster than another if both start from the same value, and assuming both decay exponentially. The example below (Natapoff, 1994, personal communication) shows the time histories of two hypothetical SPV curves (Figure 3.8) and their corresponding cumulative frequency distributions (Figure 3.9). The maximum difference (13 x 0.05 = 0.65) in the CFDs occurs at x = 112. A total of (4, 17) measurements of (A, B) lie below x = 112. Thus the difference at x = 112 is 13 out of the 20 points in each set. Since the two-tailed KS test has a critical value of 11 at the $\alpha = 0.01$ confidence level, the CFDs are significantly different.

This use of the KS test can not detect certain types of differences. It will not, for example, distinguish one permutation of a set of numbers from another, and therefore is not sensitive to the time sequence of these numbers. For example, the same sequence occurring in inverted time order would be indistinguishable from the original, and long sequences that converge to the same value. In this experiment, the KS test was used to compare ensemble average SPV responses under different conditions. 60 second long sequences were used, except that a shorter interval was used for both when the comparison involved flight data.

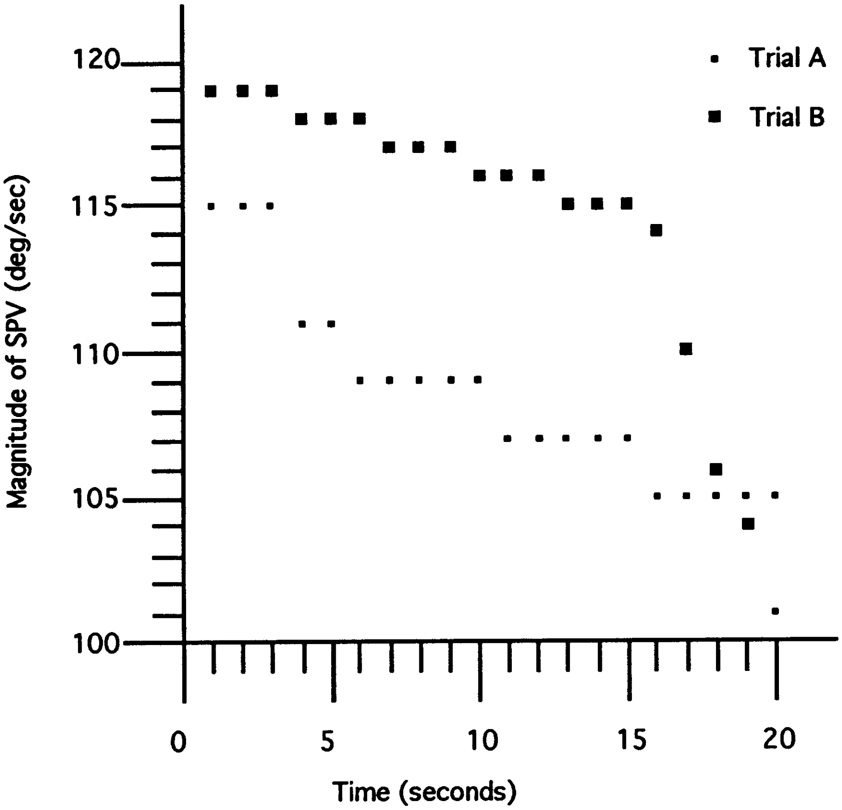


Figure 3.8: Time histories of two sample trials

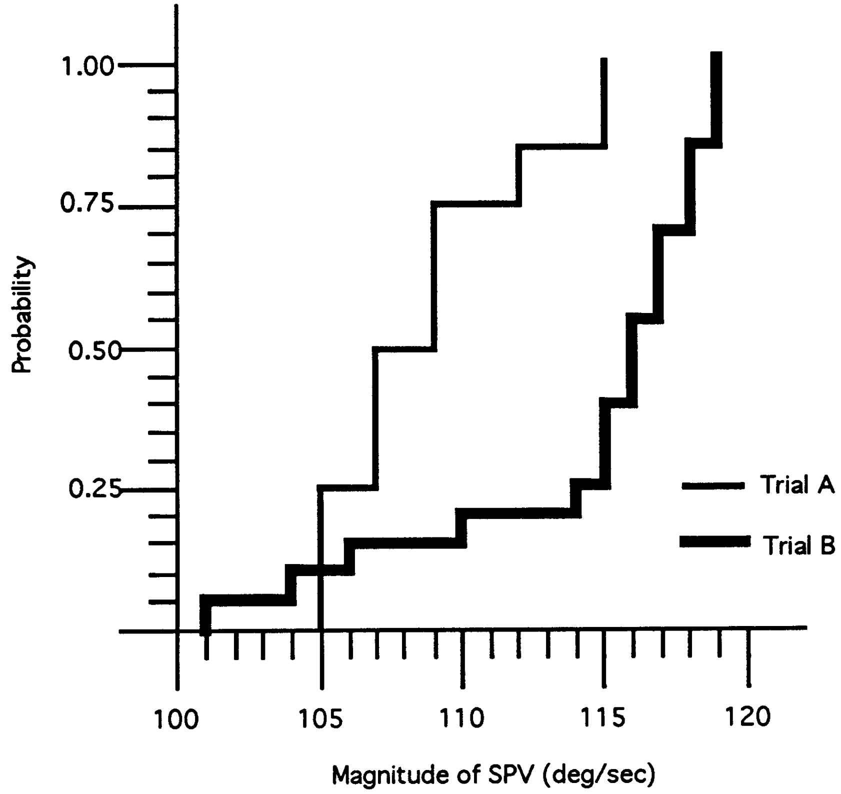


Figure 3.9: Cumulative Frequency Distribution

3.9.2 Two-Sample t-Test for Independent Samples

The parametric data was compared by referring the t-statistic to the two tailed student t-distribution for $n_1 + n_2 - 2$ degrees of freedom.

$$t = \frac{\overline{x_1} - \overline{x_2}}{(s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{3.2}$$

where

$$s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}} \tag{3.3}$$

3.9.3 Sum of T-square statistic

The statistic below would be distributed as χ^2 if x_i, y_i are based on averages of large number of measurements and if the measurements for different values of i were independent. These assumptions were made in the analysis of earlier missions, D-1 and SL-1. However for the AATM algorithm, which uses a 1 second moving averager, adjacent samples at 4 Hz are correlated. A new sampling frequency at which adjacent samples to be independent was found by looking at the auto-correlation of one subject's data. The auto-correlation with a delay of 1.5 seconds was found to be small if a new sampling rate of 2/3 Hz was used. Then the sum of t^2 statistic was calculated:

$$\sum t^2 = \sum_{i=1}^{N} \frac{(y_i - x_i)^2}{\sigma_p^2} \tag{3.4}$$

Since the distribution of this statistic (as opposed to $\chi^{2)}$ is not readily available, it was used qualitatively. A very large sum of t^2 would suggest a significant result, while a low sum of t^2 would suggest a non-significant result. By looking at a large number of t^2 values and comparing them to the plots of the actual data, and other statistical tests, it was determined that a sum of t^2 of 1000 would be the breakpoint between the two above mentioned trends.

3.9.4: Actual use of the statistical tests

Responses under two different conditions were compared using six statistical tests. The t-test and KS test were both performed on the gain and time constant parametric data. Both the sum of t^2 and KS tests were performed on the ensemble averaged data. A rule of thumb was adopted to determine if the results of the two comparisons taken together indicated a real difference. If either the gains or time constants were found to be significantly different by the t-test, and if the KS test for the ensemble average data also indicated a significant difference, then the two cases was determined to be different. The other two tests, i.e. KS test for the parametric data, and the sum of t^2 data for the ensemble data were used as secondary indicators.

Chapter 4: Results

4.1 Completion status of runs

The number of runs performed per subject varied. Reasons why some runs were not performed included time constraints, nausea of the subject, operator error, and equipment malfunctions. Table 4.1 lists the number of runs performed for each subject during the pre-flight, in-flight, early post-flight, late post-flight, and KC-135 testing periods.

Subject	Pre-flight	In-flight	Early post	Late post	KC-135
T	16	8	16	24	16
V	32	13	16	24	14
X	32	16	16	0	0
Y	32	9	18	24	13

Table 4.1: Number of runs performed for pre-flight, In-flight, Early Post-flight, and the KC-135

4.2 Run rejection results

As outlined in section 3.7, three run segment rejection criteria were used. The results of these criteria are listed in tables 4.2, 4.3, 4.4, and 4.5 for the pre-flight, in-flight, early post-flight, and KC-135 testing periods. The first column is the percent rejected category which is the total number of run segments rejected divided by the total possible number of run segments. A run segment differs from a run since a run is composed of both a per-rotatory and post-rotatory segment, while a run segment segment is either a per-rotatory segment or a post-rotatory segment. The next three columns are the categories constraint hit, % good < 60, and MSE > 200, which are the three run rejection criteria outlined in section 3.7. The percentage listed for these three categories is the percentage of the rejected run segments that this run segment rejection criteria was used. There is some overlap in these categories, e.g. one run segment might hit a constraint, and have a % good < 60. However the run rejection routine first used the constraint hit criteria, then the % good on the remaining segments, and finally the MSE criteria. Hence the relatively low percent rejected using the MSE criteria was in part due to the order in which the criteria were applied.

Subject	Percent Rejected	Constraint Hit	% good < 60	MSE > 200
T	66 %	52 %	48 %	0 %
V	19 %	17 %	83 %	0 %
X	77 %	65 %	35 %	0 %
Y	27 %	18 %	70 %	12 %

Table 4.2: Results of outlier detection for pre-flight

Subject	Percent Rejected	Constraint Hit	% good < 60	MSE > 200
T	19 %	100 %	0 %	0 %
V	15 %	100 %	0 %	0 %
X	13 %	100 %	0 %	0 %
Y	0 %	0%	0 %	0 %

Table 4.3: Results of outlier detection for in-flight

Subject	Percent Rejected	Constraint Hit	% good < 60	MSE > 200
T	47 %	67 %	33 %	0 %
V	13 %	50 %	50 %	0 %
X	84 %	85 %	15 %	0 %
Y	33 %	25 %	67 %	8 %

Table 4.4: Results of outlier detection for early post-flight

Subject	Percent Rejected	Constraint Hit	% good < 60	MSE > 200
T	25 %	75 %	25 %	0 %
V	7 %	100 %	0 %	0 %
Y	15 %	50 %	50 %	0 %

Table 4.5: Results of outlier detection for KC-135

One interesting conclusion that can be drawn from the run rejection results is that for all subjects and especially subject X, the percent rejected during in-flight and in the KC-135 was less than the pre-flight.

4.3 Near Real-Time Analysis Experience

The purpose of the near real-time analysis was so the experimenters could have an idea of how the data looked. Complete analysis of one session on four subjects typically required 4-7 hours of computing on a Macintosh Quadra. User interaction was required for EOG calibration analysis. However most of the subsequent processing which was automated in batch mode, using the method developed in Matlab by Liefeld (1993). Run reports were automatically generated. Data was then manually exported to Excel spreadsheets for rapid histogram generation and statistical analysis. On prior missions, this analysis took several days of interactive work. During preflight testing, analysis was performed at MIT between test sessions. During post-flight testing, data was analyzed at Johnson Space Center, and results from a given day's session were generally available the next day. The histograms, means, and variances were useful to see if the subjects data appeared normally distributed, and to see if there were any trends in the data as compared to pre-flight norms. The run reports which showed both the SPV envelope and the model parameters proved useful in keeping the experimenters in touch with the actual data from each run, to see if any runs were abnormal, and in planning for subsequent preflight test sessions. The use of the near real-time analysis was apparent preflight , in-flight and early post-flight.

In preflight testing, one subject asked if they could be excused from two test sessions, since test data for this subject was available from a previous mission. Test data obtained was compared to the earlier results, and fell within the previously established range.

During the mission, one Subject (Y) reported abnormally long rotation sensation durations, and asked whether the one minute rotation protocol should be extended. Subject Y's preflight data base was examined, and it was useful to know that the durations reported did indeed lie beyond the range of recorded preflight durations. A decision was made to maintain a consistent 1 minute rotation period.

During early post-flight testing, one subject found dumping runs extremely provocative, and asked what the impact on experiment science would be if these runs were omitted in subsequent testing. The preflight and early post-flight data base of this subject was examined, and based on the large variance of this particular subject's data, a decision was made not to ask this subject to perform further dumping runs, since the likelihood of a positive statistical finding was minimal.

4.4 Directional Asymmetry Analysis

Asymmetries are not unusual. They can arise from differences between right and left in oculo motor performance, end organ response, or the neural pathway between the end organ and the oculo motor system. The question is whether the asymmetry relates more to the direction of the stimulus, or to the direction of the response.

Performing an ANOVA individually on the gain and time constant data of each subject with rotation direction and condition being factors, showed no subjects having rotational direction being significant. (Table 4.6)

Subject	Parameter	dF	F (Dir)	Prob (Dir)
T	Gain	1	0.041	0.841
	T.C.	1	1.615	0.212
V	Gain	1	0.555	0.460
	T.C.	1	2.692	0.108
X	Gain	1	0.048	0.829
	T.C.	1	0.044	0.837
Y	Gain	1	0.312	0.580
	T.C.	1	0.002	0.968

Table 4.6: Pre-flight Stimulus Direction Asymmetry ANOVA Results

Asymmetries are also found that depress the SPV when grouped by the direction of the SPV response, not by the direction of rotation stimulus. So for example CW per-rotatory is compared to CCW head erect and CCW dumping runs because all these produce slow phase nystagmus to the left. These response asymmetries were noted in several subjects in the D-1 (Oman & Weigl, 1988) and SLS-1 analysis (Balkwill, 1992). Subject T had a history of strabismus surgery and esophoria in the right eye, an oculomotor abnormality. An ANOVA (Table 4.7) on the gain and time constant data of each subject with response direction and condition being factors showed subject T's time constant with a significant response direction asymmetry.

Subject	Parameter	dF	F (Dir)	Prob (Dir)
T	Gain	1	0.140	0.710
	T.C.	1	21.146	**0.000**
V	Gain	1	0.225	0.637
	T.C.	1	0.808	0.373
X	Gain	1	0.005	0.943
	T.C.	1	0.694	0.419
Y	Gain	1	2.706	0.108
	T.C.	1	2.618	0.114

Table 4.7: Pre-flight Response Asymmetry ANOVA Results

Subject T's time constants were consistently reduced when SPV was to the left, as shown in Table 4.8:

Direction	per	he	dmp
cw	16.75	31.91	15.15
ccw	22.56	23.05	13.76
average	19.65	28.69	14.65

Table 4.8 Pre-flight Mean Time Constants for Subject T

Although the ANOVA did not show a significant reduction in gain, the ensemble average SPV envelopes for Subject T were consistently reduced in magnitude when the SPV response was in a leftward direction, as shown in Figures 4.1 and 4.2:

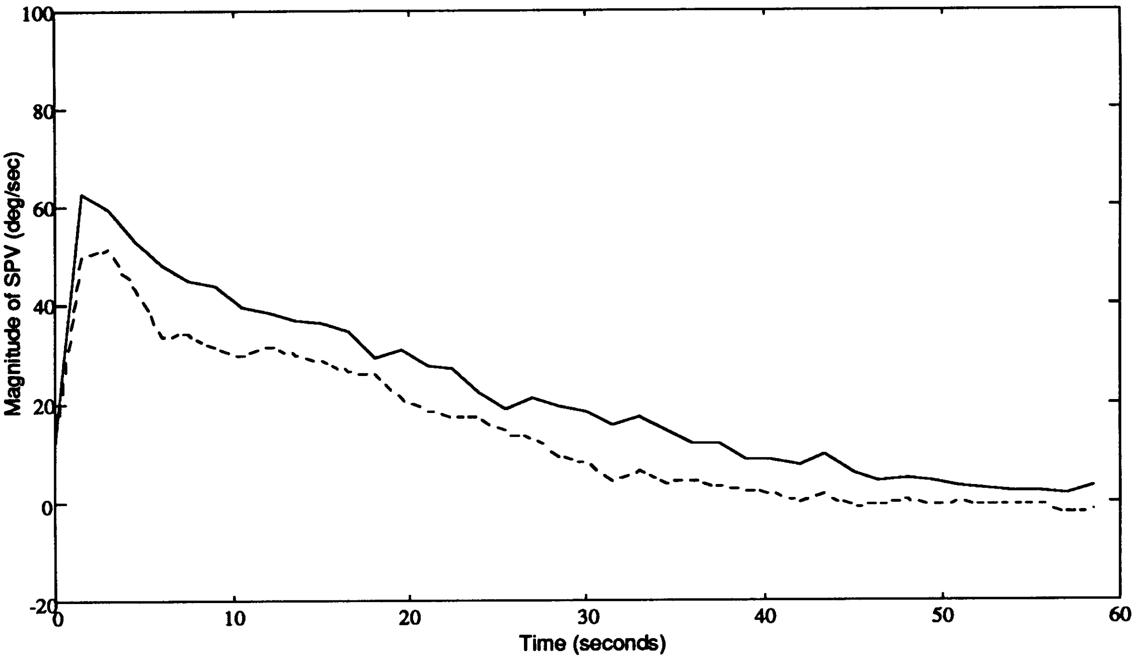


Figure 4.1: Subject T, pre-flight CCW per-rotatory (solid line) and. negative of pre-flight CW per-rotatory (dotted line) SPV response

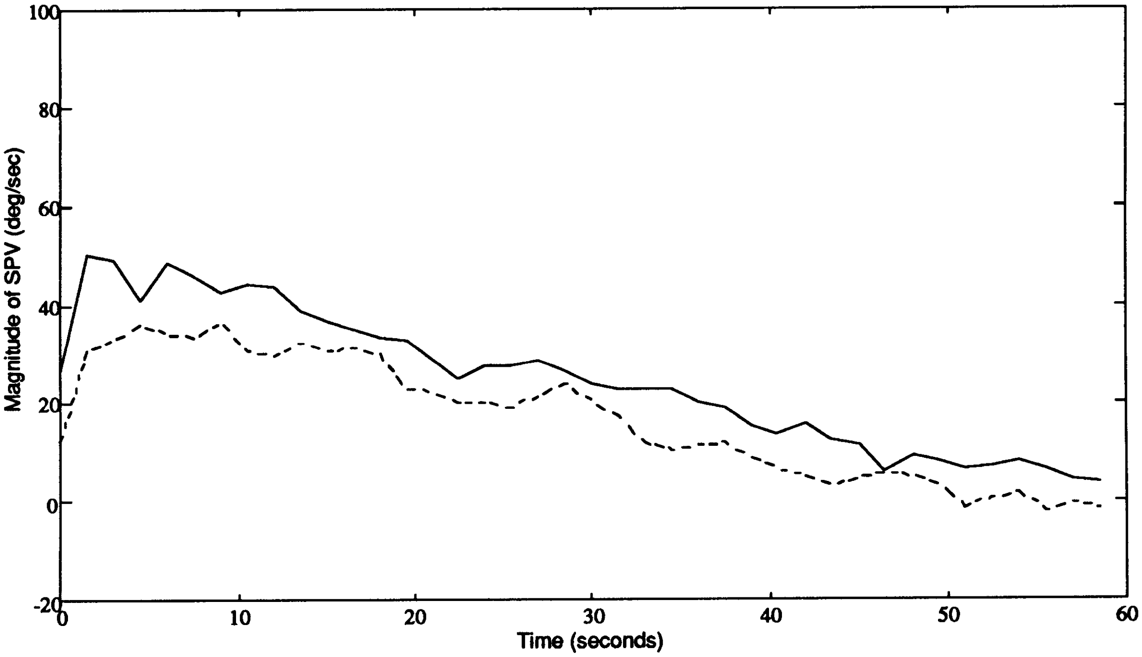


Figure 4.2: Subject T, pre-flight CW head erect (solid line) and negative of pre-flight CCW head erect (dotted line) SPV response

The effect of head tilt on Subject's T's SPV responses was present in both the CW and CCW directions (Figures 4.3-4.4), and therefore in the response when averaged across both directions (Figure 4.5). Because the goal of this experiment was to find the effect of zero-gravity on SPV, and since it was expected that - like dumping - the zero-G effect would be present in both the CW and CCW responses, for further analysis it was considered justified to group Subject T's CW and CCW cases together, rather than treating them separately.

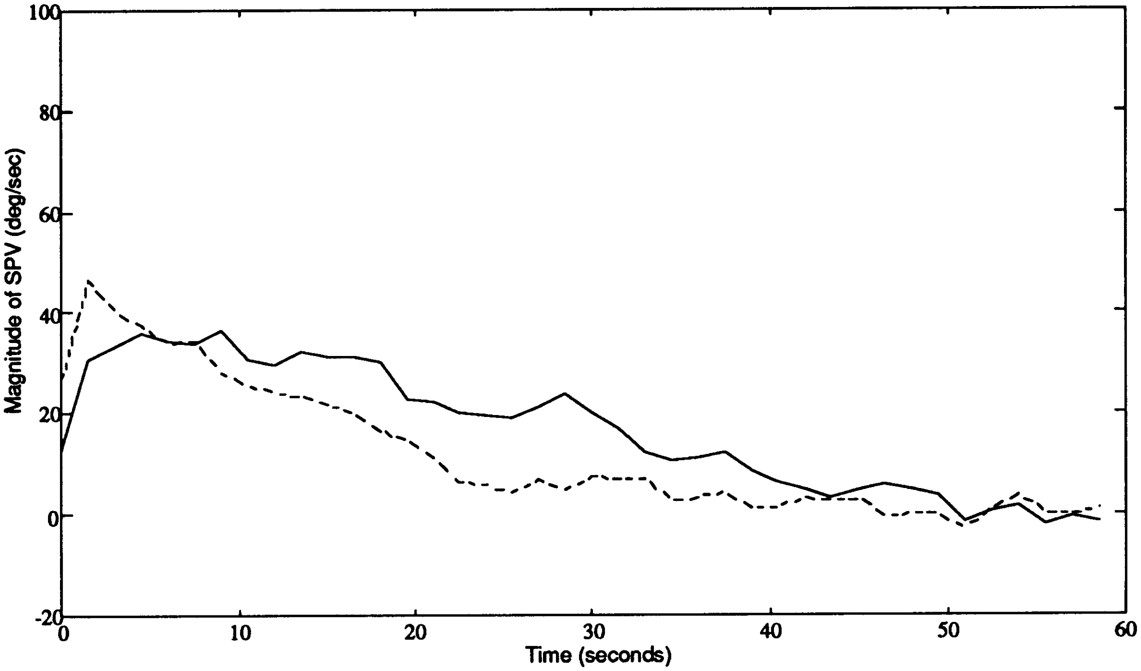


Figure 4.3: Subject T, pre-flight CW head erect (solid line) and pre-flight CW dumping (dotted line) SPV response

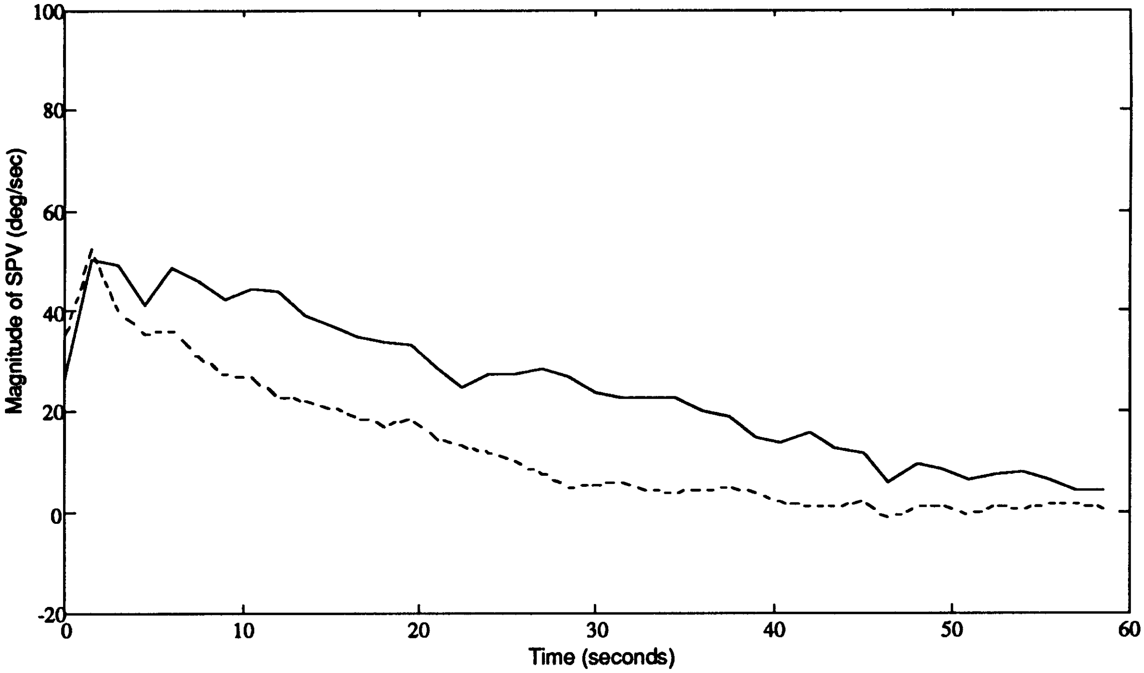


Figure 4.4: Subject T, pre-flight CCW head erect (solid line) and pre-flight CCW dumping (dotted line) SPV response

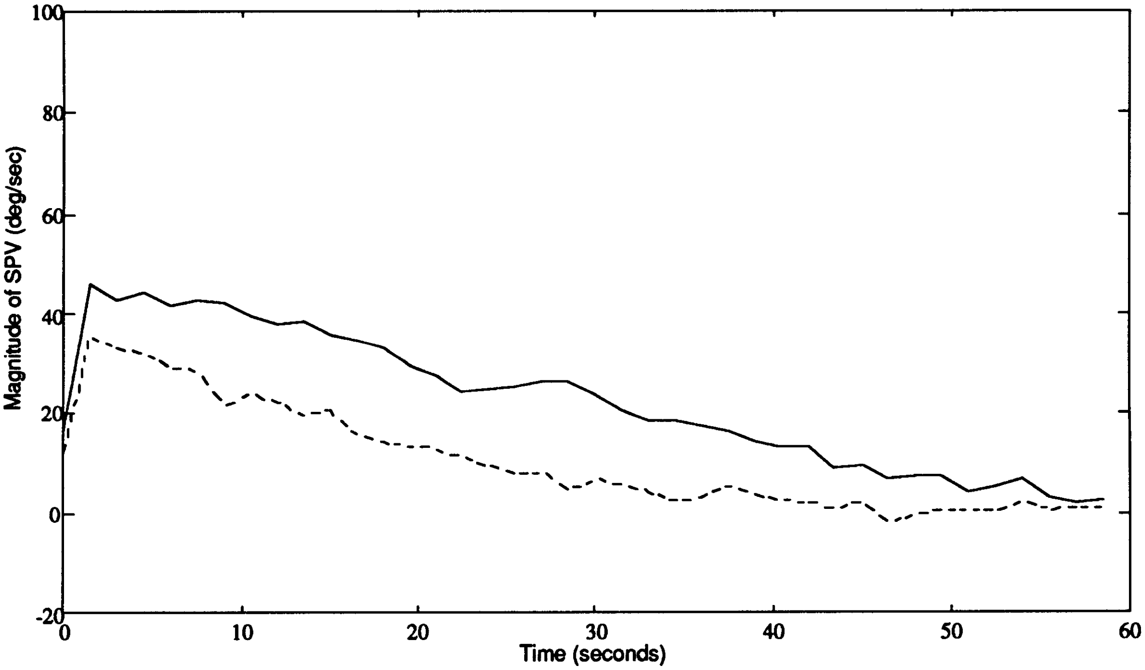


Figure 4.5: Subject T, pre-flight head erect (solid line) and pre-flight dumping (dotted line) SPV response

4.5 Comparisons

4.5.1 The Effect of an Acute Exposure to Zero-gravity on Slow-Phase Velocity

It was found that when a subject was exposed to an acute exposure to zero gravity in parabolic flight that the SPV response was significantly different than in 1-G in 2/3 subjects (Subjects V and Y), and in these two subjects the time constants were shorter. Subject T's time constant was also shorter, but the difference was not significant . Subject X was not tested due to technical problems. (Table 4.9, Figures 4.6-4.8)

				Parametric		Ensemble	
Subject		KC-1G	KC-0G	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					150.55	0.111
	Gain	0.39	0.34				
	T.C.	23.44	22.78				
	Parametric						
	Gain	0.39	0.33	0.389	0.125		
	T.C.	30.98	30.28	0.937	0.500		
V	Ensemble					**1901.90**	**0.006**
	Gain	0.48	0.42				
	T.C.	14.79	7.95				
	Parametric						
	Gain	0.45	0.43	0.706	1.000		
	T.C.	16.85	8.64	**0.000**	0.125		
Y	Ensemble					**3140.79**	**0.038**
	Gain	0.78	0.73				
	T.C.	20.01	12.57				
	Parametric						
	Gain	0.72	0.74	0.728	1.000		
	T.C.	26.75	13.52	**0.007**	0.080		

Table 4.9: Results of Comparison Between KC-135 1-G and KC-135 0-G (Results when $p <= 0.05$ or $\Sigma t^2 > 1000$ indicated in **bold**)

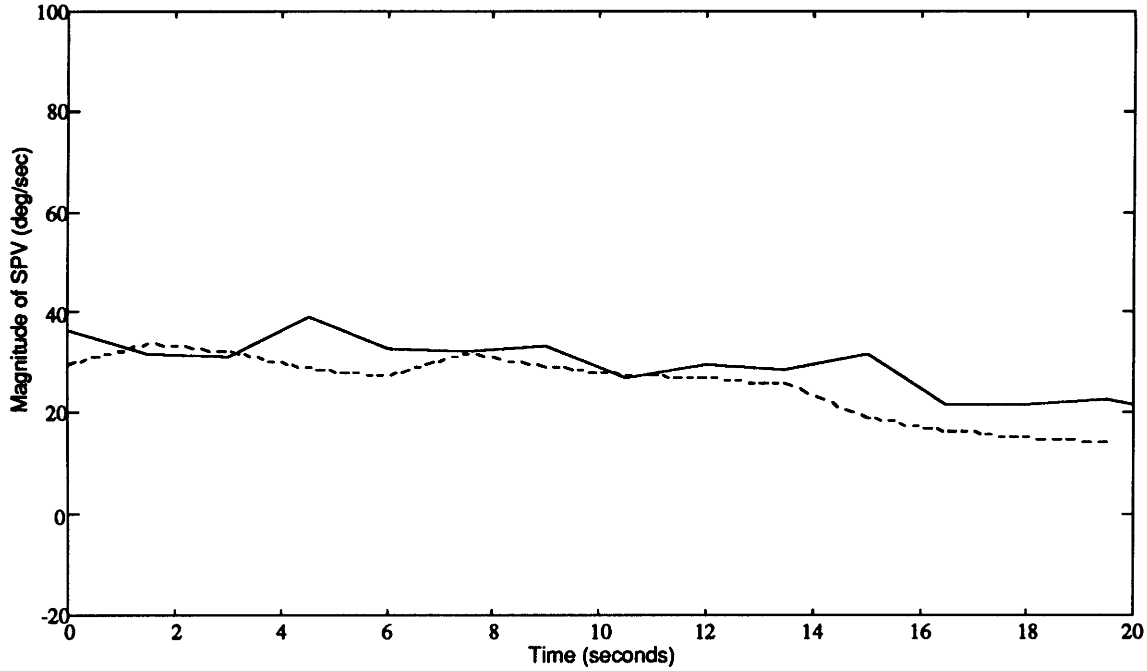


Figure 4.6: Subject T, KC-135 1-G (solid line) and KC-135 0-G (dotted line) SPV response

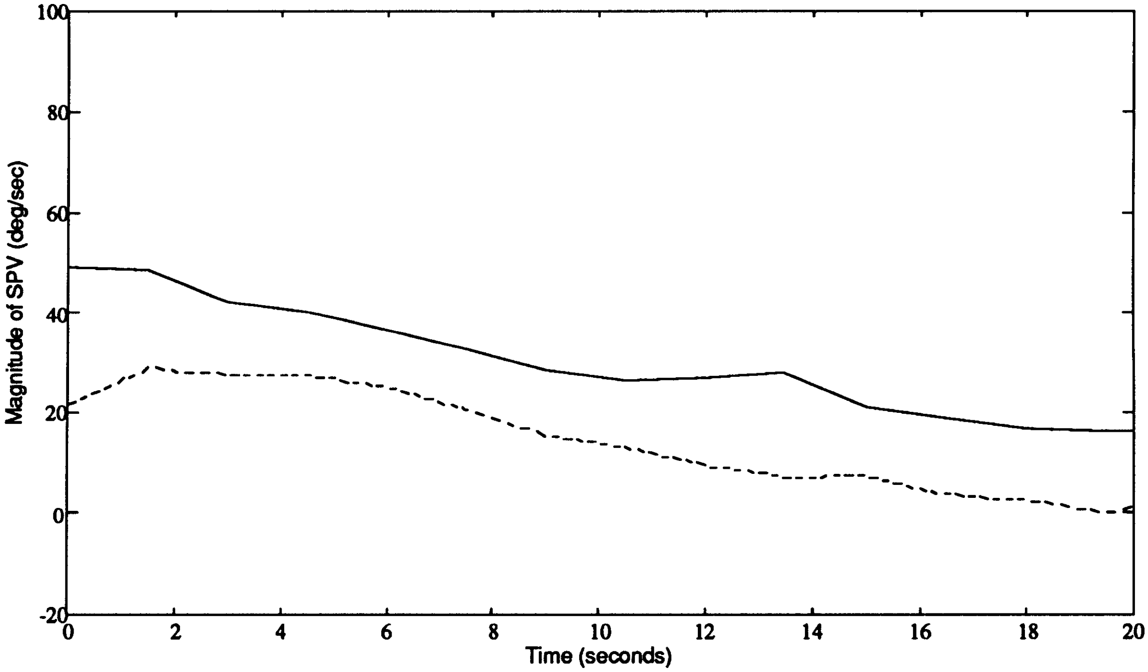


Figure 4.7: Subject V, KC-135 1-G (solid line) and KC-135 0-G (dotted line) SPV response

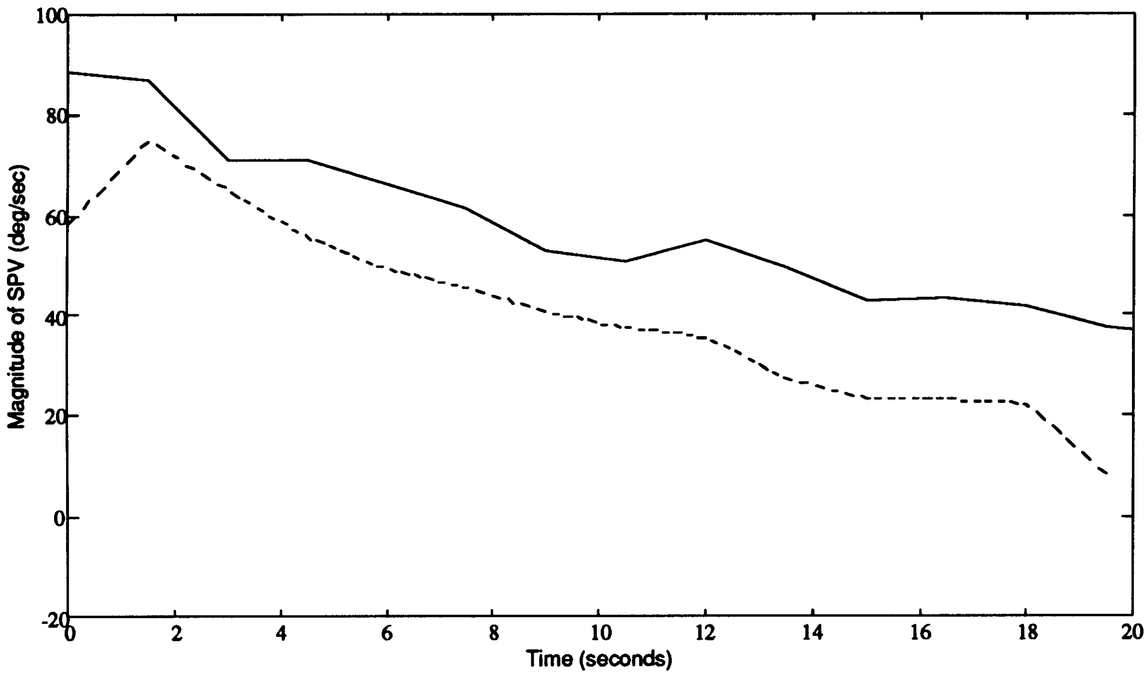


Figure 4.8: Subject Y, KC-135 1-G (solid line) and KC-135 0-G (dotted line) SPV response

4.5.2 The Effect of Prolonged 0-G Exposure on Head erect Slow-Phase Velocity

It was found that in-flight head erect runs were significantly different than pre-flight head erect runs for 2 subjects (Subjects T and Y). Subject T's in-flight head erect time constant was found to be shorter. Subject Y's in-flight head erect time constant was found to be larger. For subjects V and X, the in-flight head erect runs could not be proved to be significantly different. (Table 4.10, Figures 4.9-12)

				Parametric		Ensemble	
Subject		Pre-HE	FLIGHT-HE	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					**3895.06**	**0.000**
	Gain	0.48	0.31				
	T.C.	28.31	15.39				
	Parametric						
	Gain	0.47	0.31	0.056	0.667		
	T.C.	28.69	16.63	**0.026**	0.667		
V	Ensemble					247.82	0.976
	Gain	0.61	0.64				
	T.C.	13.69	13.13				
	Parametric						
	Gain	0.62	0.63	0.896	0.320		
	T.C.	13.74	13.45	0.828	0.820		
X	Ensemble					773.42	0.227
	Gain	0.46	0.35				
	T.C.	19.39	15.78				
	Parametric						
	Gain	0.46	0.35	0.287	0.500		
	T.C.	20.33	14.63	0.212	0.500		
Y	Ensemble					**1651.50**	**0.022**
	Gain	0.87	0.84				
	T.C.	18.49	22.49				
	Parametric						
	Gain	0.88	0.86	0.868	0.500		
	T.C.	18.14	22.19	**0.016**	0.125		

Table 4.10: Results of Comparison Between Pre-flight Head erect and In-flight Head erect (Results when $p <= 0.05$ or $\Sigma t^2 > 1000$ indicated in **bold**)

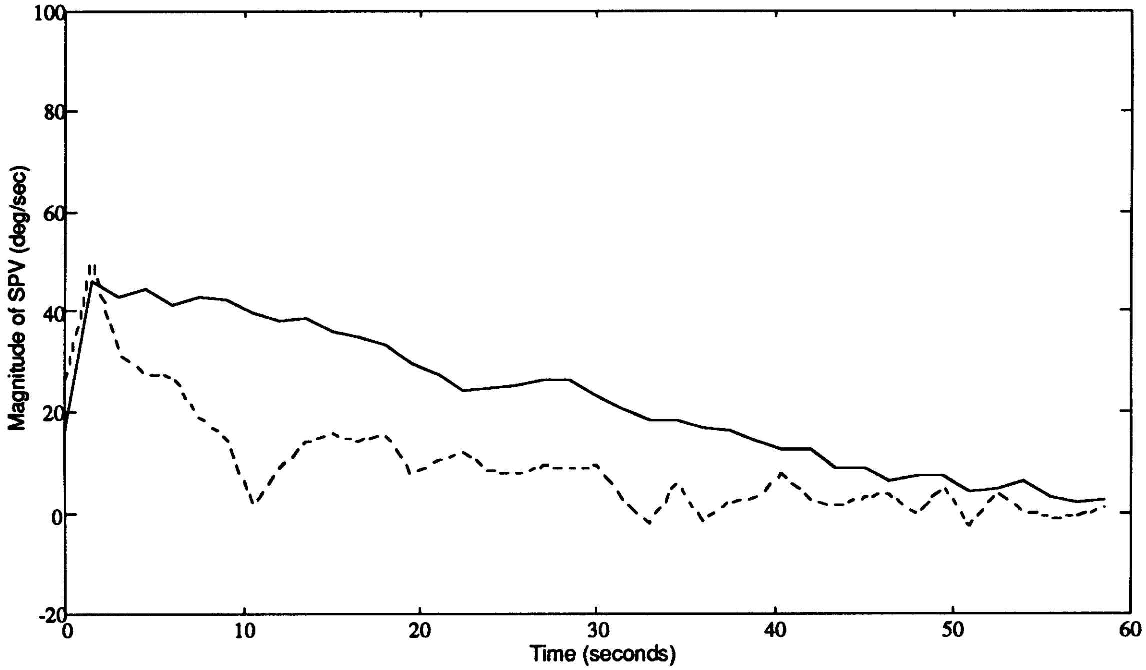


Figure 4.9: Subject T, Pre-flight Head erect (solid line) and In-flight Head erect (dotted line) SPV response

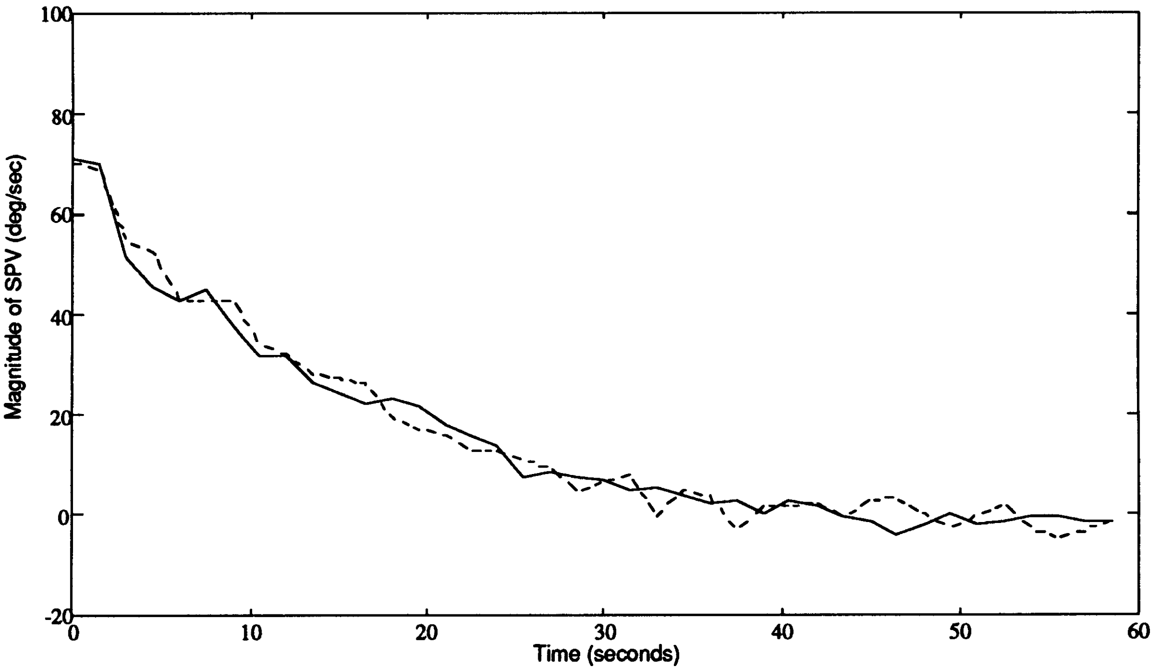


Figure 4.10: Subject V, Pre-flight Head erect (solid line) and In-flight Head erect (dotted line) SPV response

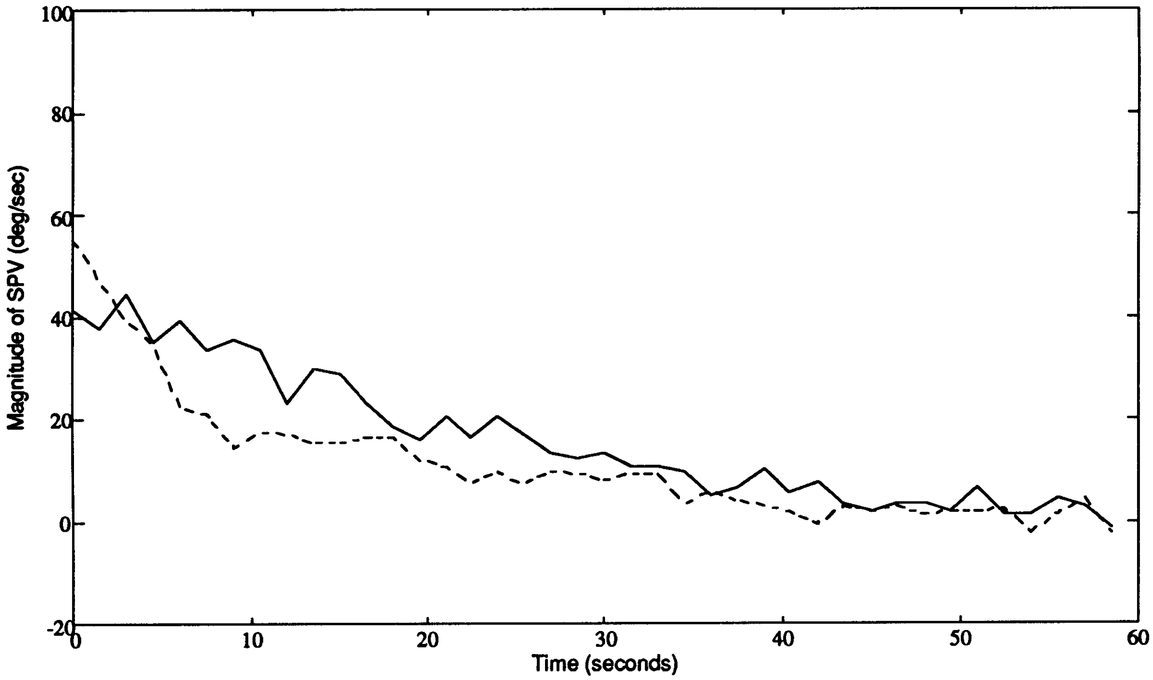


Figure 4.11: Subject X, Pre-flight Head erect (solid line) and In-flight Head erect (dotted line) SPV response

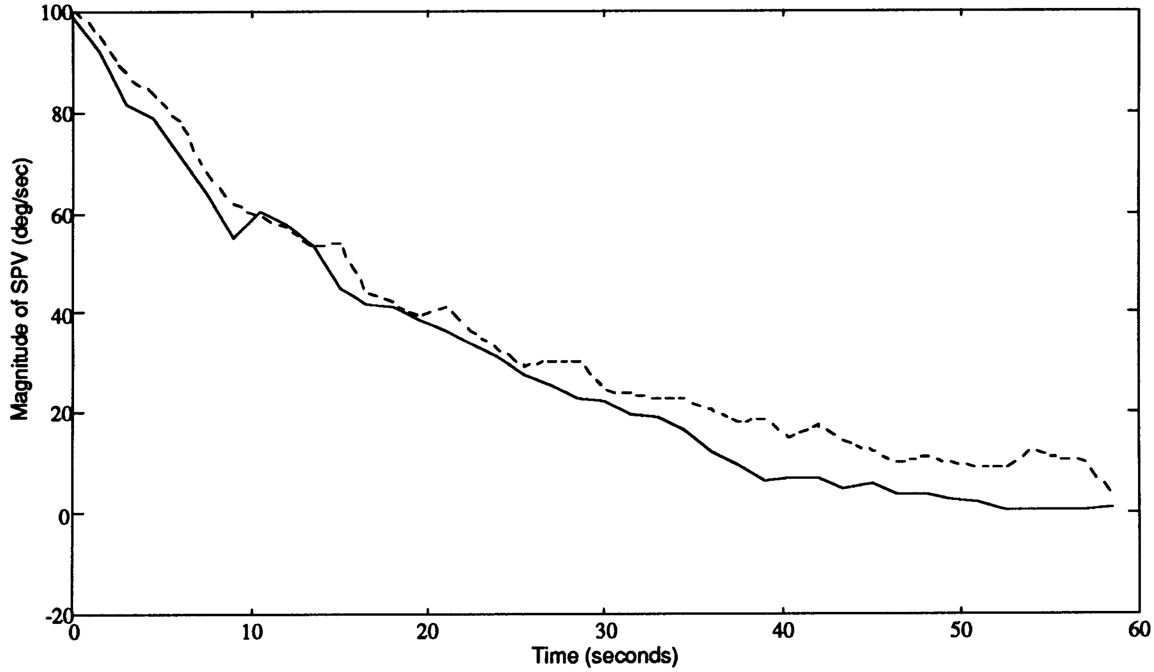


Figure 4.12: Subject Y, Pre-flight Head erect (solid line) and In-flight Head erect (dotted line) SPV response

4.5.3 The Effect of Prolonged 0-G Exposure on Per-rotatory Slow-Phase Velocity

It was found that in-flight per-rotatory runs were significantly different than pre-flight per-rotatory runs for 2 subjects (Subjects T and X). In these two subjects the in-flight per-rotatory time constants were shorter. For subjects V and Y the in-flight per-rotatory runs could not be proved significantly different than pre-flight per-rotatory runs. Subject V's in-flight per-rotatory time constant was found to be larger. (Table 4.11, Figures 4.13-16)

				Parametric		Ensemble	
Subject		PRE-PER	FLIGHT-PER	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					**12594.00**	**0.004**
	Gain	0.56	0.31				
	T.C.	18.6	11.41				
	Parametric						
	Gain	0.53	0.28	**0.001**	**0.009**		
	T.C.	19.65	12.37	**0.021**	0.074		
V	Ensemble					270.00	0.876
	Gain	0.76	0.67				
	T.C.	9.10	11.21				
	Parametric						
	Gain	0.81	0.70	0.143	0.112		
	T.C.	9.38	11.87	**0.001**	**0.006**		
X	Ensemble					**1779.14**	0.079
	Gain	0.51	0.20				
	T.C.	13.17	11.23				
	Parametric						
	Gain	0.50	0.39	0.444	**0.005**		
	T.C.	11.08	8.24	0.078	**0.028**		
Y	Ensemble					710.16	0.260
	Gain	1.05	0.97				
	T.C.	13.13	13.47				
	Parametric						
	Gain	1.04	0.92	0.197	0.307		
	T.C.	12.75	13.61	0.409	1.000		

Table 4.11: Results of Comparison Between Pre-flight Per-rotatory and In-flight Per-rotatory (Results when p <= 0.05 or $\Sigma t^2 > 1000$ indicated in **bold**)

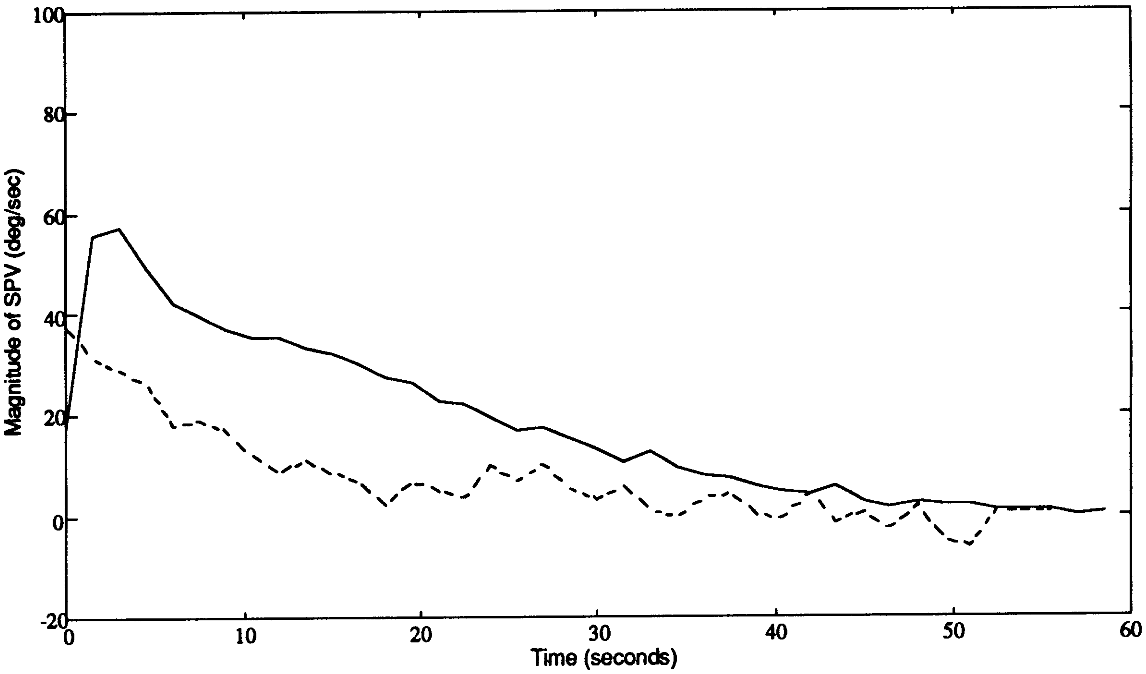


Figure 4.13: Subject T, Pre-flight Per-rotatory (solid line) and In-flight Per-rotatory (dotted line) SPV response

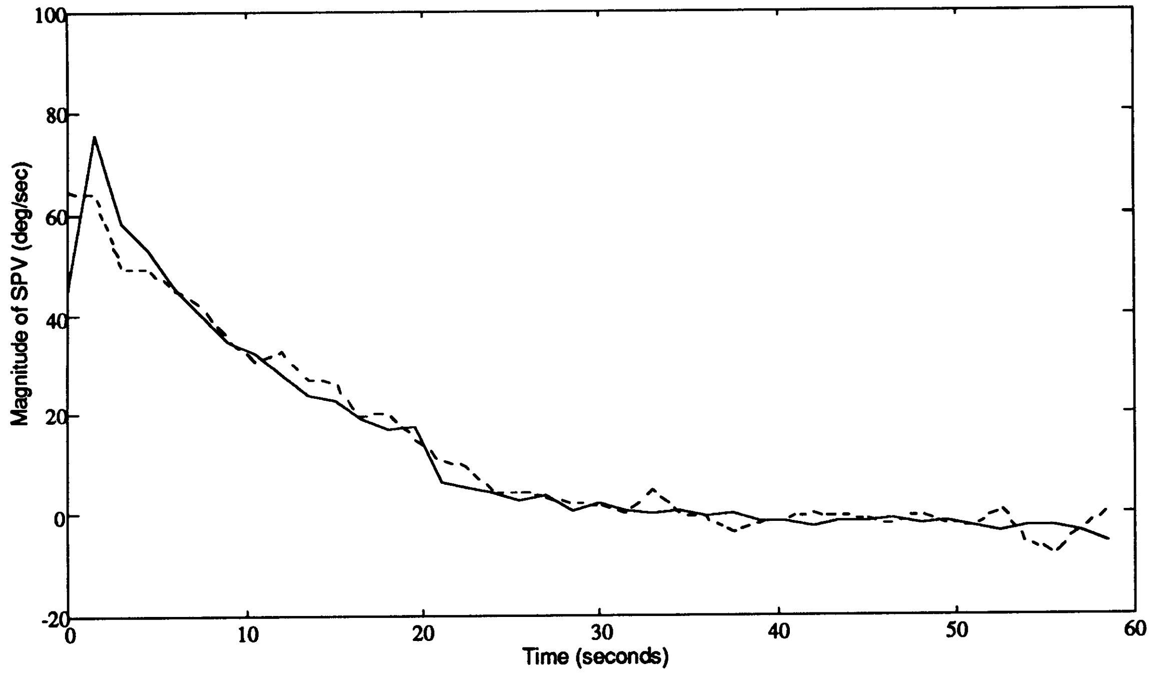


Figure 4.14: Subject V, Pre-flight per-rotatory (solid line) and In-flight per-rotatory (dotted line) SPV response

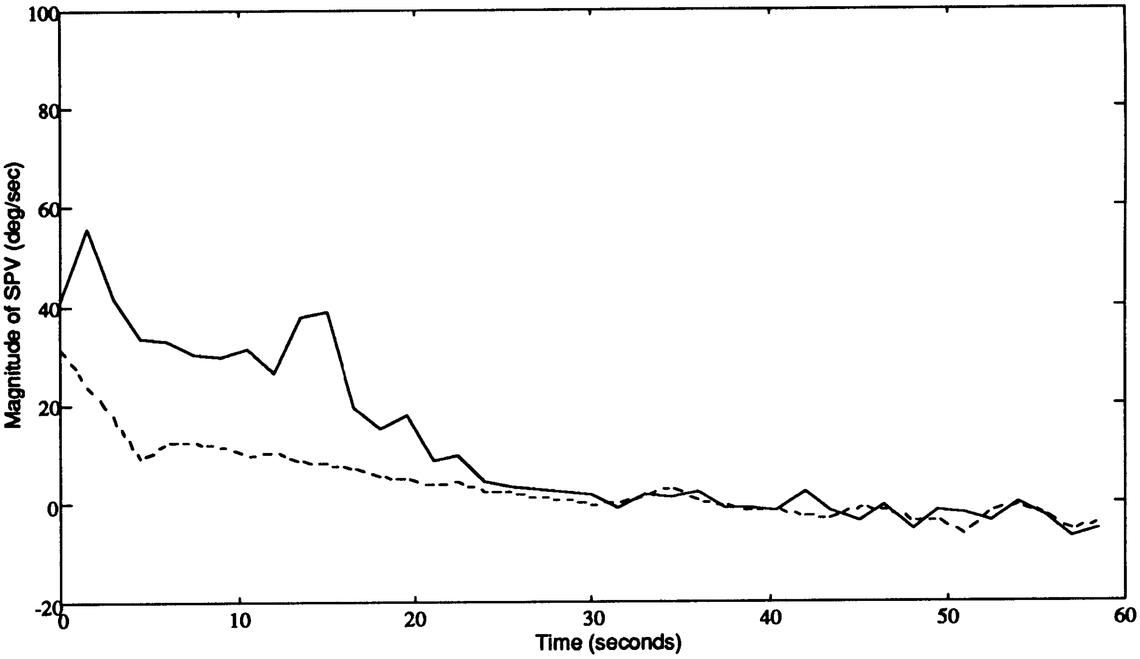


Figure 4.15: Subject X, Pre-flight per-rotatory (solid line) and In-flight per-rotatory (dotted line) SPV response

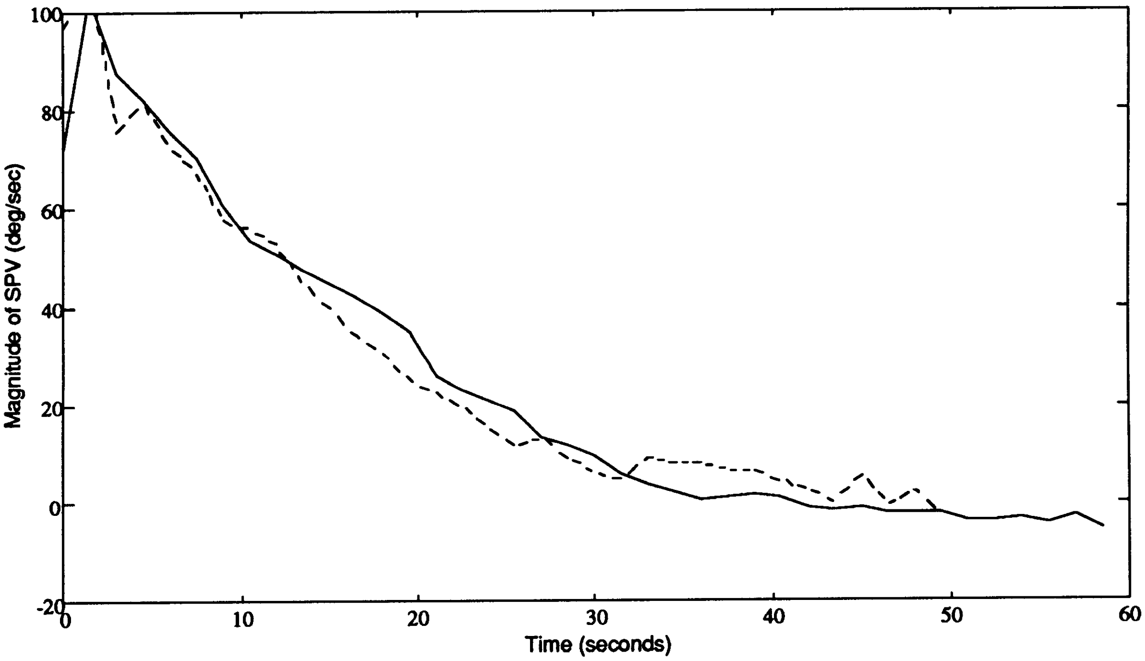


Figure 4.16: Subject Y, Pre-flight per-rotatory (solid line) and In-flight per-rotatory (dotted line) SPV response

4.5.4 Changes in early post-flight head-erect responses

The difference between early post-flight head erect and pre-flight head erect SPV responses could not be shown to be statistically significant for three subjects (T, V, Y). Subject Y's early post-flight head erect time constant was found to be significantly larger. (Table 4.12, Figures 4.17-4.19) Subject X's early post-flight data was not assessed, because only a single data point remained after application of the run segment rejection criteria.

				Parametric		Ensemble	
Subject		Pre-HE	Early-HE	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					**2404.07**	0.517
	Gain	0.48	0.45				
	T.C.	28.31	24.33				
	Parametric						
	Gain	0.47	0.43	0.500	1.000		
	T.C.	28.69	23.27	0.105	0.820		
V	Ensemble					177.436	0.876
	Gain	0.61	0.71				
	T.C.	13.69	11.76				
	Parametric						
	Gain	0.62	0.73	0.239	0.919		
	T.C.	13.74	11.68	0.081	0.189		
X	Ensemble						
	Gain	0.46	0.86				
	T.C.	19.39	8.13				
	Parametric						
	Gain	0.46	0.86				
	T.C.	20.33	8.13				
Y	Ensemble					**1309.96**	0.704
	Gain	0.87	0.83				
	T.C.	18.49	21.22				
	Parametric						
	Gain	0.88	0.82	0.420	0.516		
	T.C.	18.14	21.73	**0.009**	**0.006**		

Table 4.12: Results of Comparison Between Pre-flight Head erect and Early Head erect (Results when p <= 0.05 or $\Sigma t^2 > 1000$ indicated in **bold**)

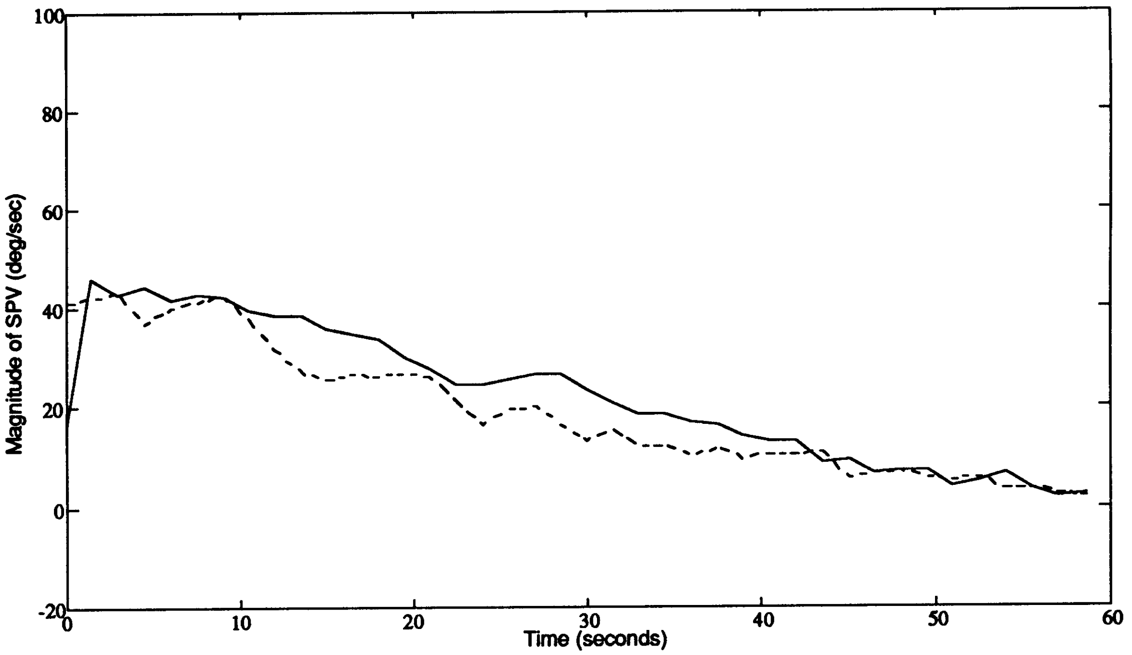


Figure 4.17: Subject T, Pre-flight Head erect (solid line) and Early Head erect (dotted line) SPV response

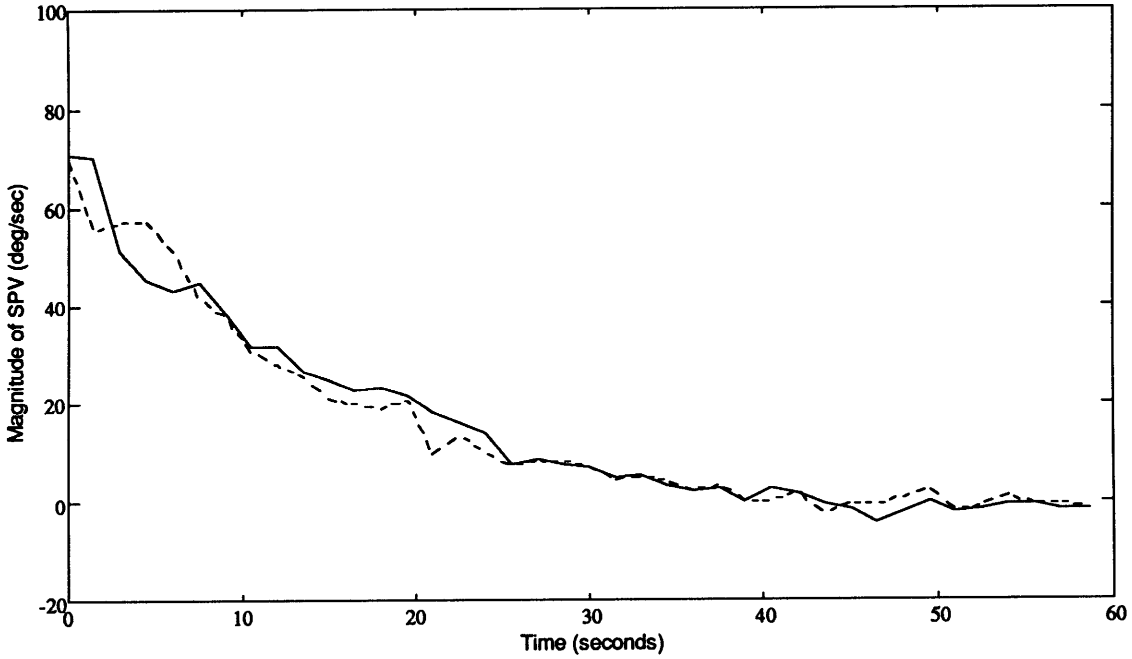


Figure 4.18: Subject V, Pre-flight Head erect (solid line) and Early Head erect (dotted line) SPV response

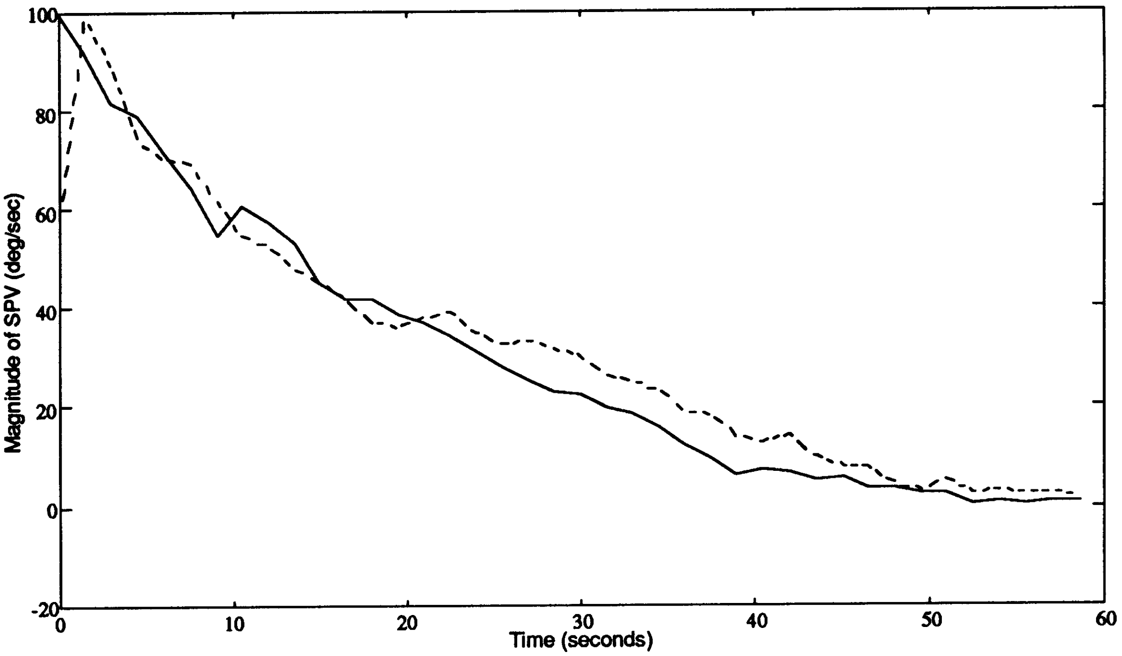


Figure 4.19: Subject Y, Pre-flight Head erect (solid line) and Early Head erect (dotted line) SPV response

4.5.5 Changes in early post-flight per-rotatory responses

The difference between early post-flight per-rotatory and pre-flight per-rotatory SPV responses could not be shown to be statistically significant for three subjects. (V, X, and Y). Subject T showed the early post-flight per-rotatory time constant to be significantly shorter. (Table 4.13, Figures 4.20-23)

				Parametric		Ensemble	
Subject		Pre-PER	Early-PER	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					**2932.84**	0.079
	Gain	0.56	0.53				
	T.C.	18.60	14.51				
	Parametric						
	Gain	0.53	0.65	0.061	0.320		
	T.C.	19.65	13.21	**0.008**	**0.080**		
V	Ensemble					197.006	0.704
	Gain	0.76	0.78				
	T.C.	9.10	9.50				
	Parametric						
	Gain	0.81	0.79	0.838	0.844		
	T.C.	9.38	9.44	0.913	0.264		
X	Ensemble					274.587	0.516
	Gain	0.51	0.56				
	T.C.	13.17	9.58				
	Parametric						
	Gain	0.50	0.56	0.357	0.667		
	T.C.	11.08	8.78	**0.044**	**0.000**		
Y	Ensemble					510.39	0.704
	Gain	1.05	0.92				
	T.C.	13.13	14.94				
	Parametric						
	Gain	1.04	1.01	0.674	0.664		
	T.C.	12.75	13.29	0.598	0.962		

Table 4.13: Results of Comparison Between Pre-flight per-rotatory and Early Per-rotatory (Results when p <= 0.05 or Σt^2 > 1000 indicated in **bold**)

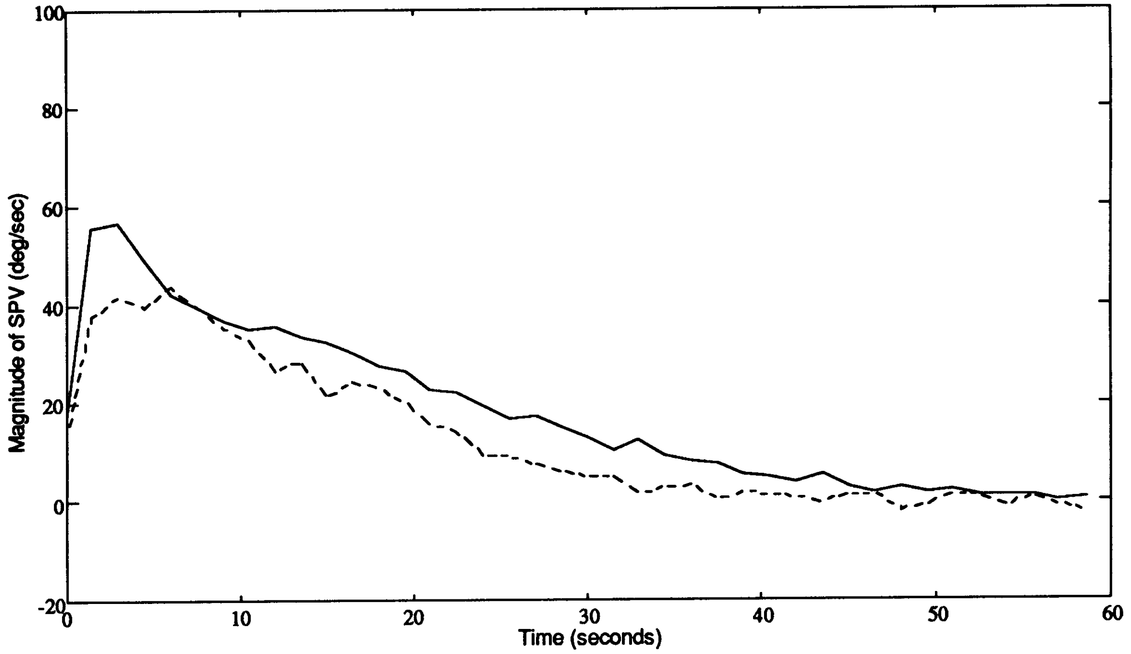


Figure 4.20: Subject T, Pre-flight Per-rotatory (solid line) and Early Per-rotatory (dotted line) SPV response

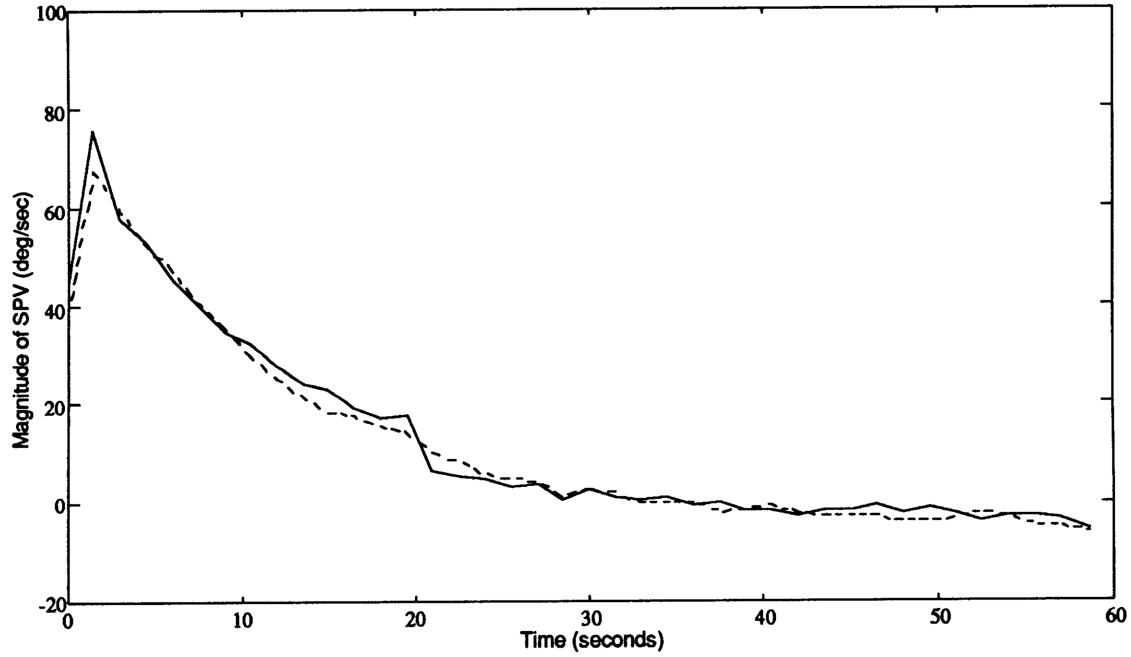


Figure 4.21: Subject V, Pre-flight Per-rotatory (solid line) and Early Per-rotatory (dotted line) SPV response

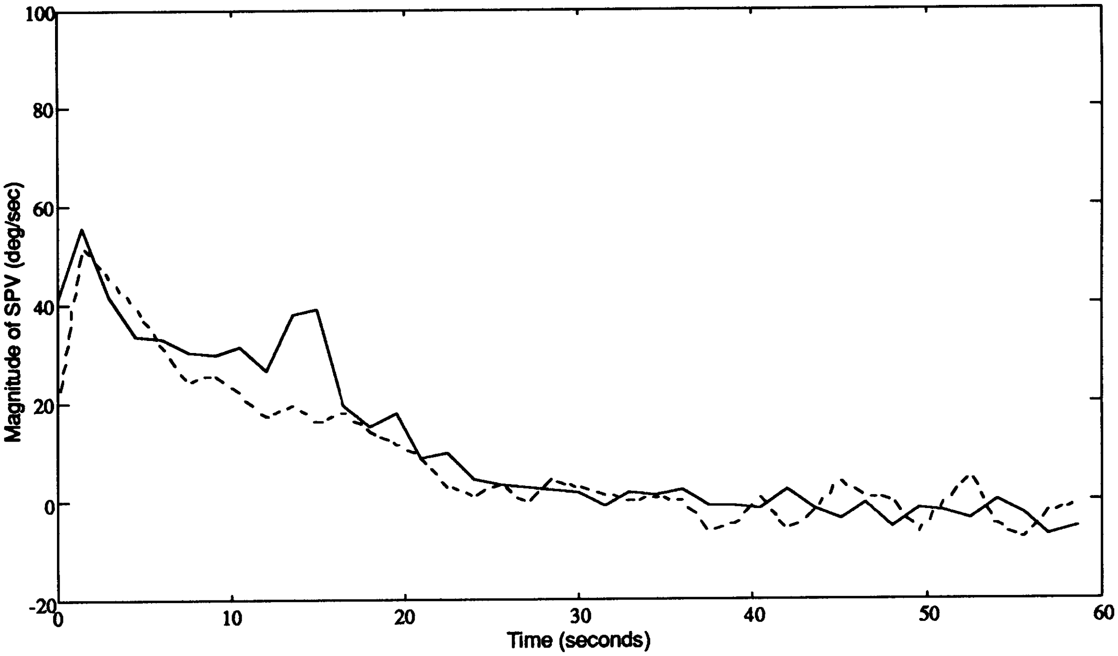


Figure 4.22: Subject X, Pre-flight Per-rotatory (solid line) and Early Per-rotatory (dotted line) SPV response

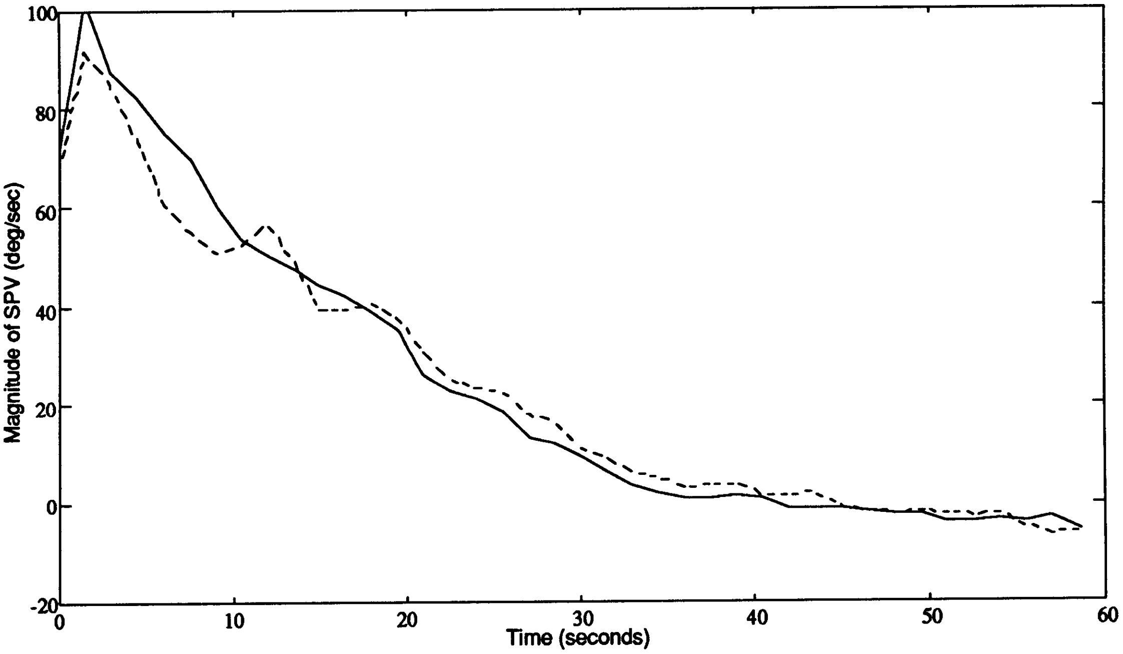


Figure 4.23: Subject Y, Pre-flight Per-rotatory (solid line) and Early Per-rotatory (dotted line) SPV response

4.5.6 The Effect of Pre-Flight Dumping on Slow-Phase Velocity

It was found that the pre-flight dumping runs were significantly different than the pre-flight head erect runs for 3/4 subjects (Subjects T, V, and Y). In these three subjects the pre-flight dumping time constants were shorter than the head erect time constants. Subject Y's dumping gain was found to be shorter. Subject X's dumping time constant did show a trend towards shortening, but was not significant different. (Table 4.14, Figures 4.24-4.27)

				Parametric		Ensemble	
Subject		Pre-HE	Pre-DMP	t-test p <=	KS p < =	sum t2	KS p < =
T	Ensemble					**10720.7**	**0.001**
	Gain	0.48	0.38				
	T.C.	28.31	15.67				
	Parametric						
	Gain	0.47	0.46	0.811	0.768		
	T.C.	28.69	14.65	**0.000**	**0.000**		
V	Ensemble					**1598.28**	0.079
	Gain	0.61	0.64				
	T.C.	13.69	7.97				
	Parametric						
	Gain	0.62	0.69	0.460	0.662		
	T.C.	13.74	7.85	**0.000**	**0.000**		
X	Ensemble					287.67	0.353
	Gain	0.46	0.71				
	T.C.	19.39	12.42				
	Parametric						
	Gain	0.46	0.43	0.773	1.000		
	T.C.	20.33	16.70	0.471	0.500		
Y	Ensemble					**14822.2**	**0.022**
	Gain	0.87	0.67				
	T.C.	18.49	10.96				
	Parametric						
	Gain	0.88	0.70	**0.041**	**0.028**		
	T.C.	18.14	11.19	**0.000**	**0.000**		

Table 4.14: Results of Comparison Between Pre-flight Head Erect and Pre-flight Dumping. (Results when $p <= 0.05$ or $\Sigma t^2 > 1000$ indicated in **bold**)

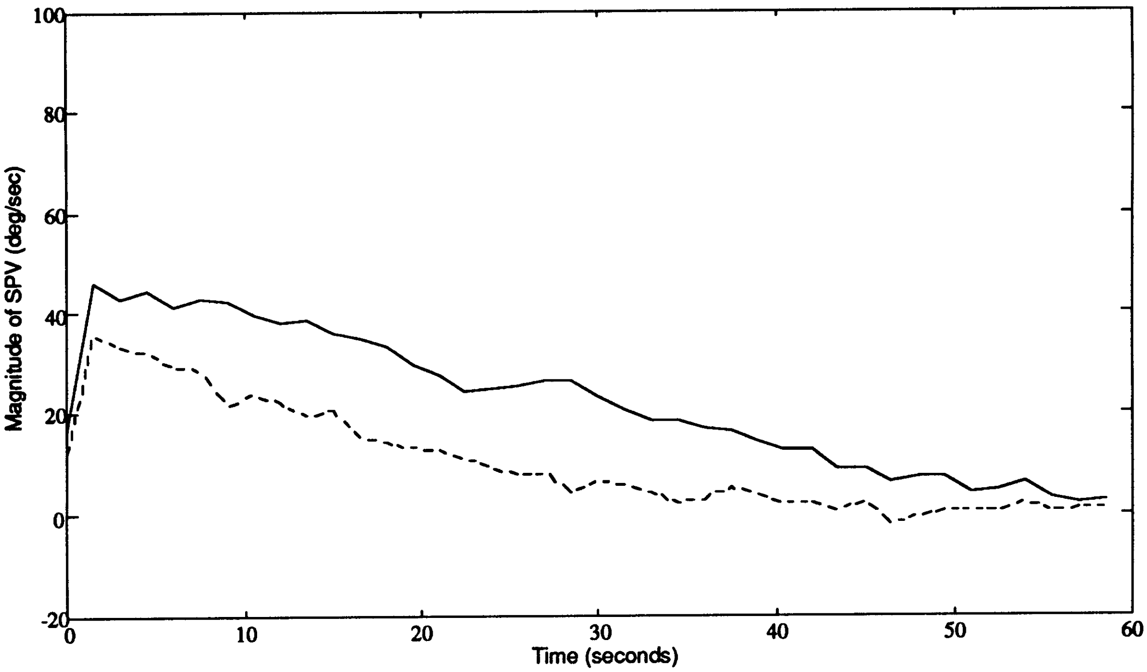


Figure 4.24: Subject T, Pre-flight head erect (solid line) and Pre-flight dumping (dotted line) SPV response

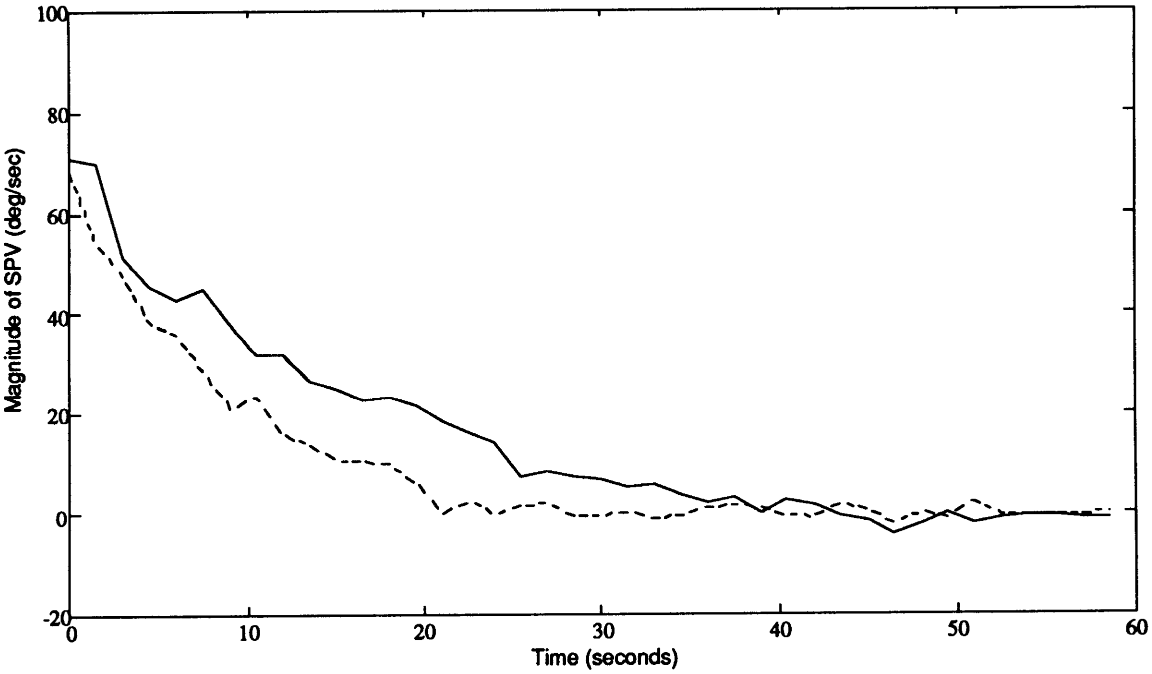


Figure 4.25: Subject V, Pre-flight head erect (solid line) and Pre-flight dumping (dotted line) SPV response

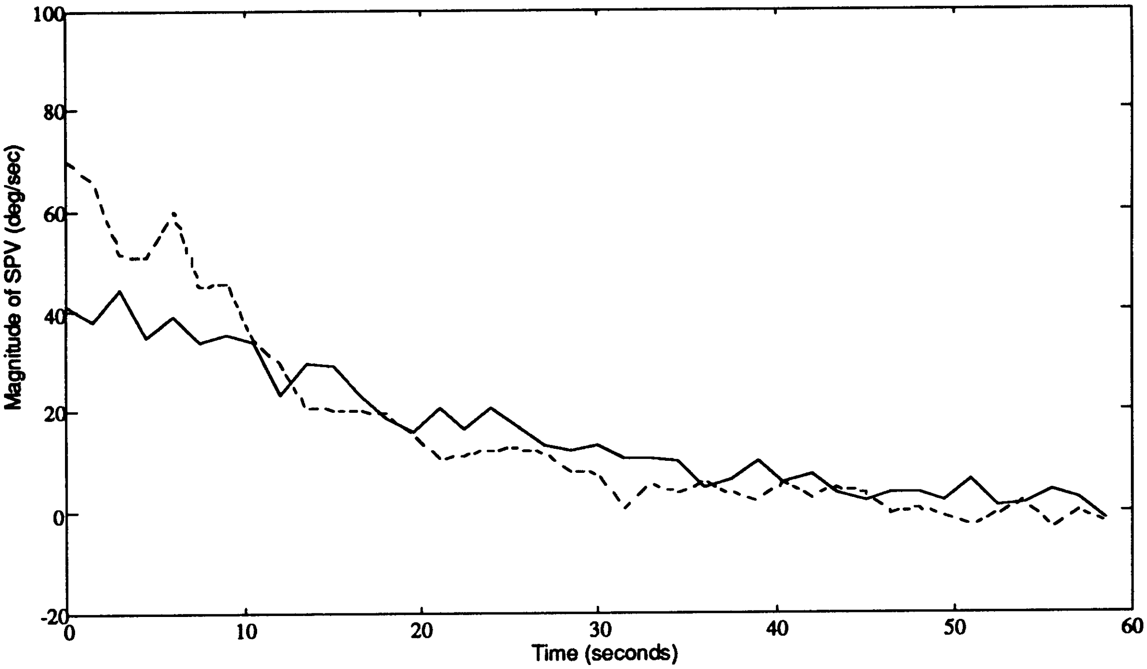


Figure 4.26: Subject X, Pre-flight head erect (solid line) and Pre-flight dumping (dotted line) SPV response

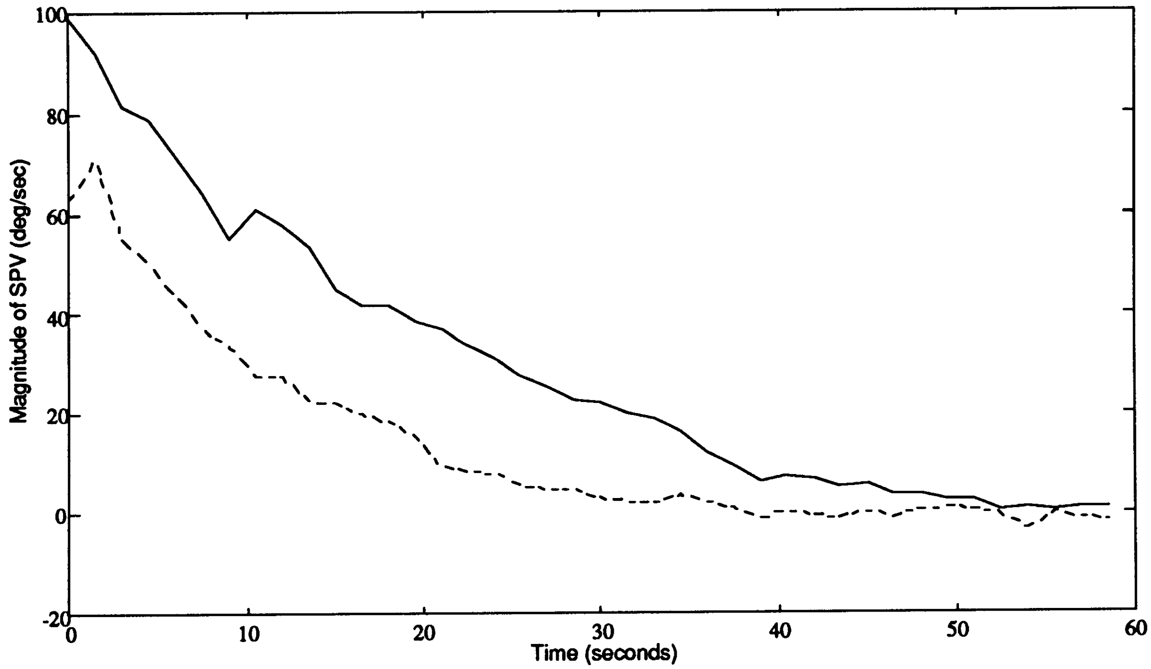


Figure 4.27: Subject Y, Pre-flight head erect (solid line) and Pre-flight dumping (dotted line) SPV response

4.5.7 The Effect of Dumping on In-flight Slow-Phase Velocity

Did the SPV time constant shorten when a dumping head movement was made in-flight ? We were unable to show a significant difference between the in-flight dumping and head erect time constants of any of the subjects. Subject T's in-flight dumping gain was found to be larger, and Subject Y's dumping gain was found to be shorter. (Table 4.15, Figures 4.28-31)

				Parametric		Ensemble	
Subject		FLIGHT-HE	FLIGHT-DMP	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					255.41	0.856
	Gain	0.31	0.45				
	T.C.	15.39	12.35				
	Parametric						
	Gain	0.31	0.46	**0.014**	**0.000**		
	T.C.	16.63	12.15	0.442	1.000		
V	Ensemble					425.57	0.981
	Gain	0.64	0.59				
	T.C.	13.13	14.09				
	Parametric						
	Gain	0.63	0.63	0.993	0.820		
	T.C.	13.45	13.52	0.966	0.820		
X	Ensemble					413.30	0.227
	Gain	0.35	0.37				
	T.C.	14.63	11.37				
	Parametric						
	Gain	0.35	0.45	0.184	0.516		
	T.C.	14.63	11.02	0.261	0.516		
Y	Ensemble					**2961.45**	0.732
	Gain	0.84	0.69				
	T.C.	22.49	22.79				
	Parametric						
	Gain	0.86	0.77	**0.006**	**0.000**		
	T.C.	22.19	23.33	0.778	0.500		

Table 4.15: Results of Comparison Between In-flight Head erect and In-Flight Dumping (Results when p <= 0.05 or Σt^2 > 1000 indicated in **bold**)

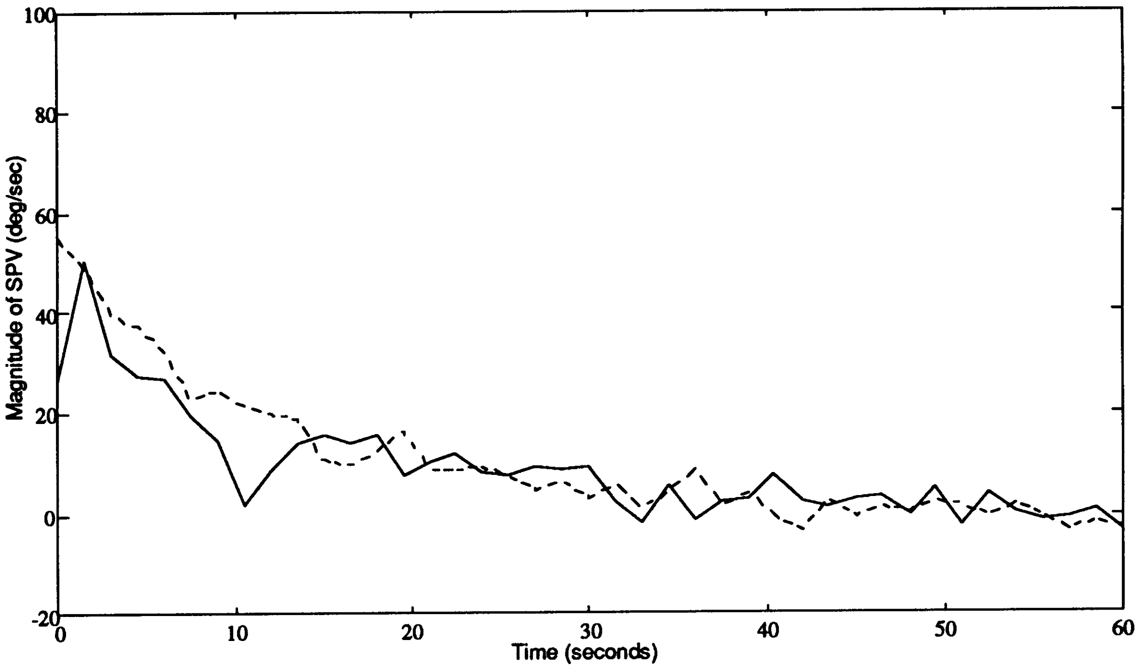


Figure 4.28: Subject T, In-flight Head erect (solid line) and In-flight Dumping (dotted line) SPV response

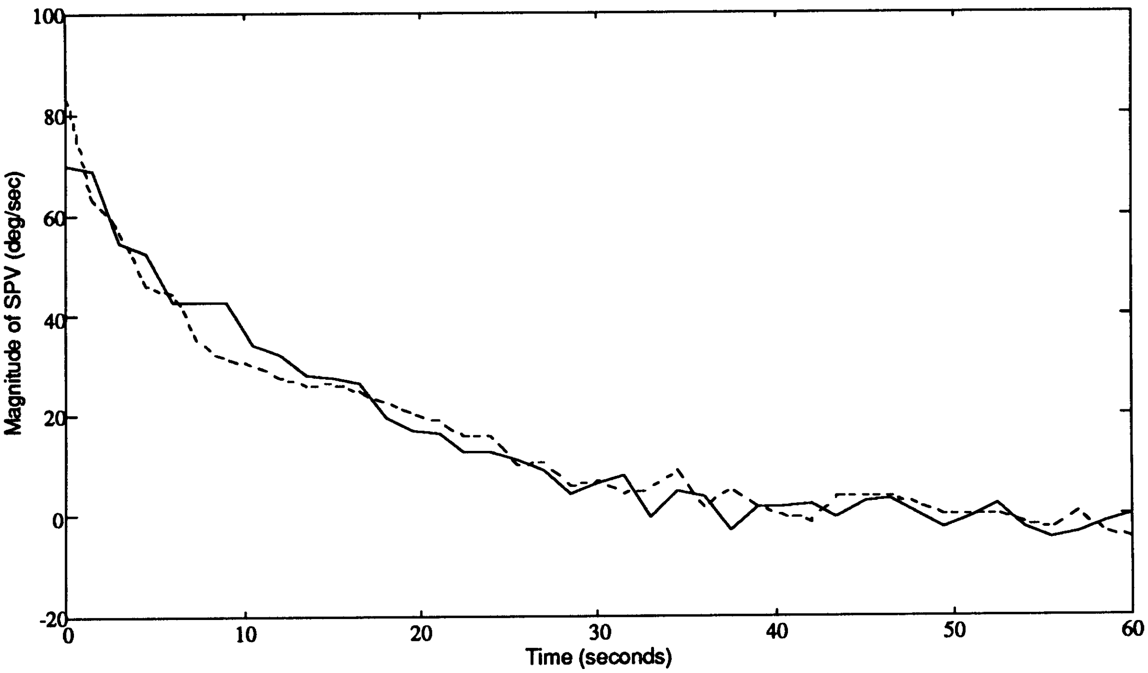


Figure 4.29: Subject V, In-flight Head erect (solid line) and In-flight Dumping (dotted line) SPV response

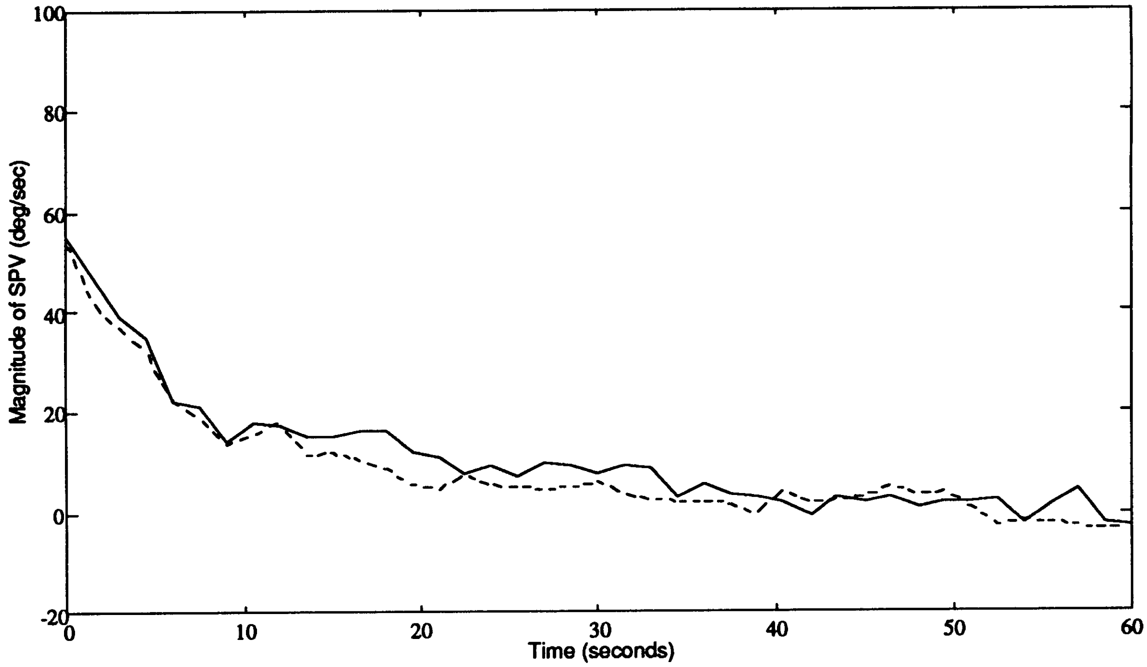


Figure 4.30: Subject X, In-flight Head erect (solid line) and In-flight Dumping (dotted line) SPV response

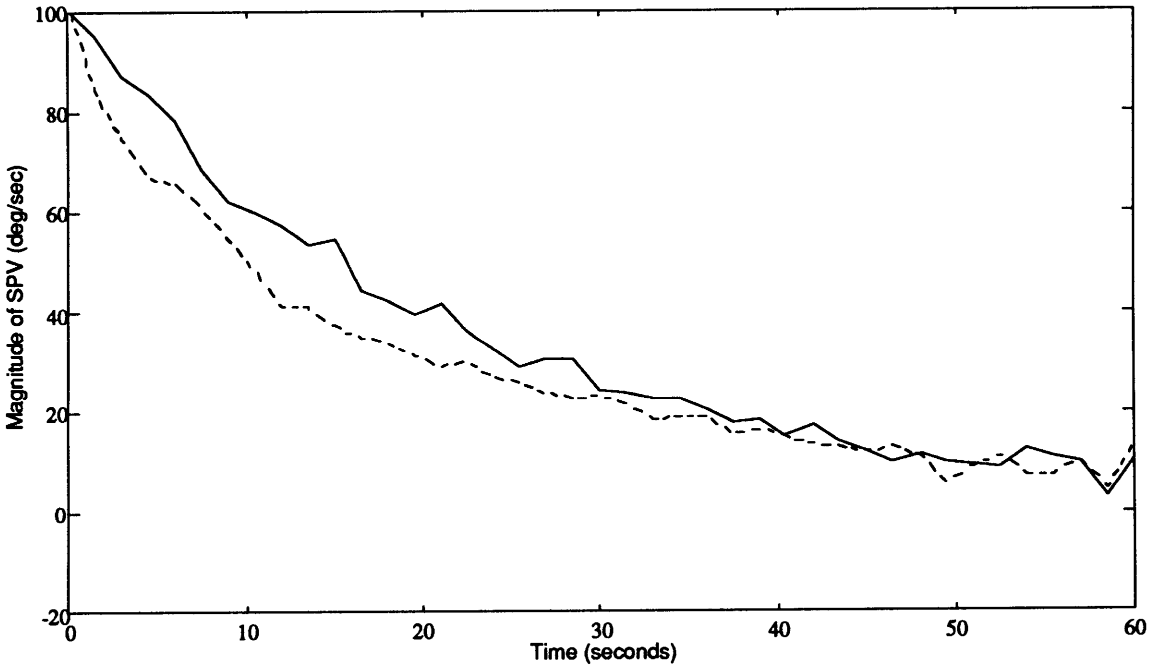


Figure 4.31: Subject Y, In-flight Head erect (solid line) and In-flight Dumping (dotted line) SPV response

4.5.8 The Effect of Prolonged 0-G Exposure on Dumping Slow Phase Velocity

It was found that in-flight dumping runs were significantly different than pre-flight dumping runs for 2 subjects (Subjects V and Y). For these two subjects the in-flight dumping time constants were larger. For subjects T and X, in-flight dumping runs were not found to be significantly different than pre-flight dumping runs. (Table 4.16, Figures 4.32-35)

				Parametric		Ensemble	
Subject		Pre-DMP	FLIGHT-DMP	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					316.40	0.970
	Gain	0.38	0.45				
	T.C.	15.67	12.35				
	Parametric						
	Gain	0.46	0.46	0.830	0.500		
	T.C.	14.65	12.15	0.084	0.125		
V	Ensemble					**1276.80**	**0.011**
	Gain	0.64	0.59				
	T.C.	7.97	14.09				
	Parametric						
	Gain	0.69	0.63	0.570	0.919		
	T.C.	7.85	13.52	**0.000**	**0.047**		
X	Ensemble					**1103.57**	0.353
	Gain	0.71	0.38				
	T.C.	12.42	11.37				
	Parametric						
	Gain	0.43	0.45	0.756	0.500		
	T.C.	16.70	11.02	0.127	0.125		
Y	Ensemble					**11377.9**	**0.000**
	Gain	0.67	0.69				
	T.C.	10.96	22.79				
	Parametric						
	Gain	0.70	0.77	0.515	0.320		
	T.C.	11.19	23.33	**0.000**	**0.000**		

Table 4.16: Results of Comparison Between Pre-flight Dumping and In-flight Dumping (Results when p <= 0.05 or $\Sigma t^2 > 1000$ indicated in **bold**)

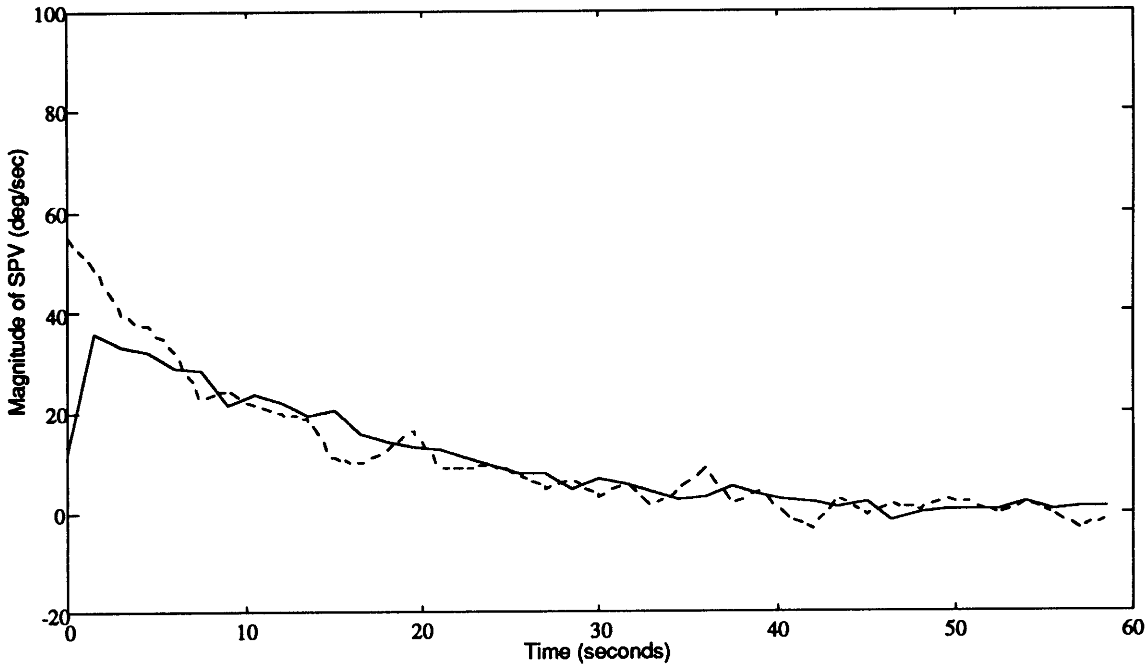


Figure 4.32: Subject T, Pre-flight Dumping (solid line) and In-flight Dumping (dotted line) SPV response

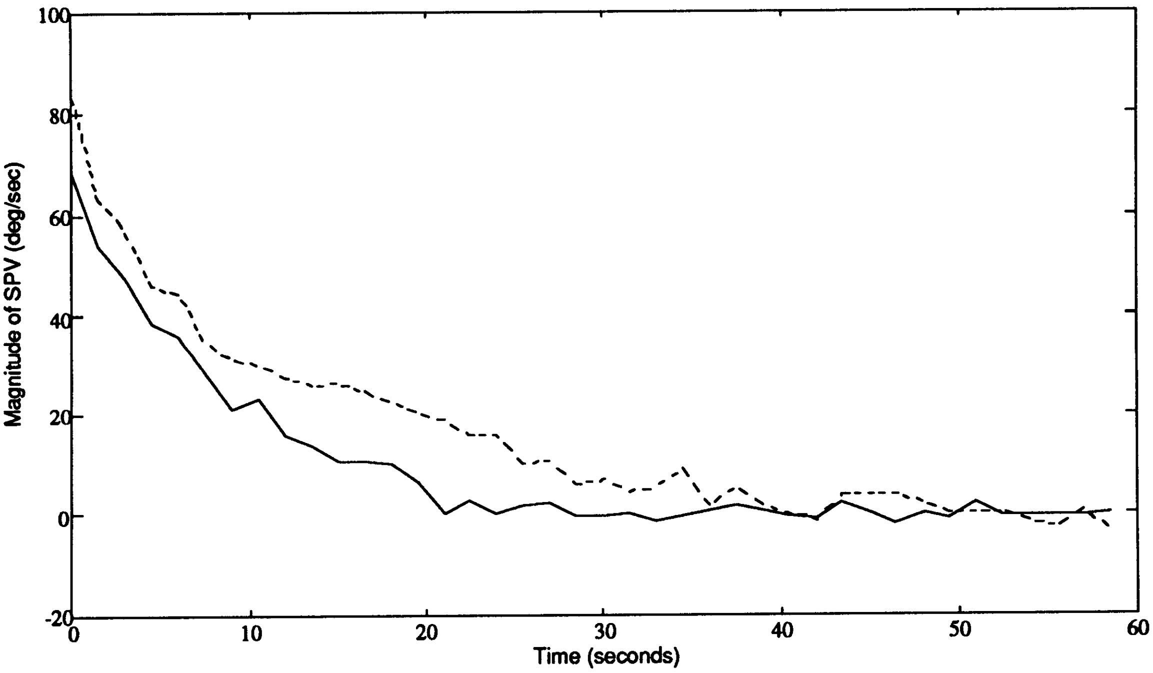


Figure 4.33: Subject V, Pre-flight Dumping (solid line) and In-flight Dumping (dotted line) SPV response

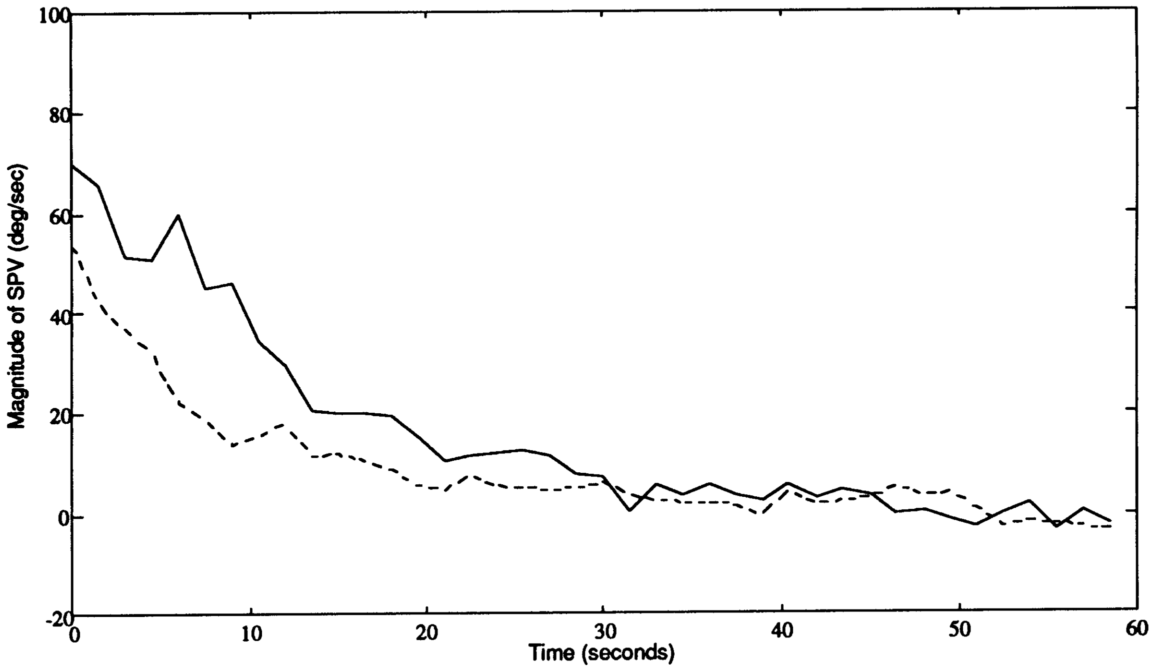


Figure 4.34: Subject X, Pre-flight Dumping (solid line) and In-flight Dumping (dotted line) SPV response

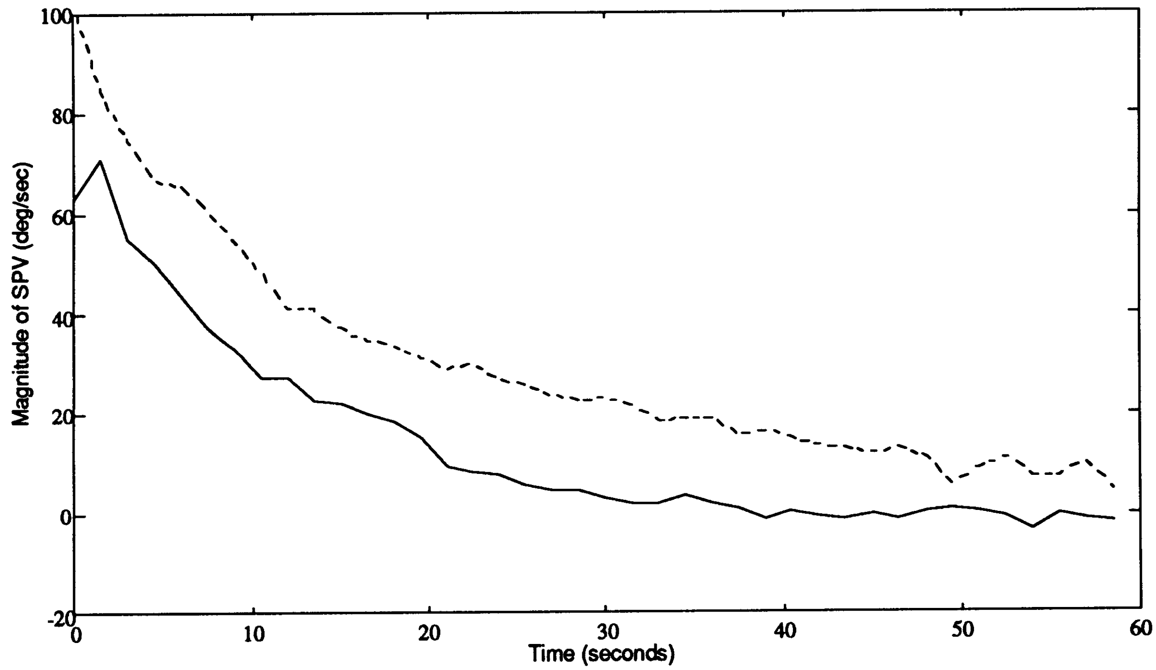


Figure 4.35: Subject Y, Pre-flight Dumping (solid line) and In-flight Dumping (dotted line) SPV response

4.5.9 The Effect of Dumping on Early Post-flight Slow-Phase Velocity

It was found that early post-flight dumping runs were significantly different than early post-flight head erect runs for 3/3 subjects (Subjects T, V, and Y). In these three subjects the dumping time constants were shorter for all three subjects. (Table 4.17, Figures 4.36-38) Subject X's early post-flight data was not assessed, because only a single data point remained after application of the run segment rejection criteria.

				Parametric		Ensemble	
Subject		Early-HE	Early-DMP	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					**3696.65**	**0.004**
	Gain	0.45	0.46				
	T.C.	24.33	14.36				
	Parametric						
	Gain	0.43	0.47	0.589	1.000		
	T.C.	23.27	13.82	**0.000**	**0.000**		
V	Ensemble					**2256.02**	**0.011**
	Gain	0.70	0.81				
	T.C.	11.76	6.63				
	Parametric						
	Gain	0.73	0.87	0.172	0.333		
	T.C.	11.68	6.40	**0.000**	**0.000**		
Y	Ensemble					**6588.71**	**0.011**
	Gain	0.83	0.77				
	T.C.	21.21	11.84				
	Parametric						
	Gain	0.82	0.77	0.279	0.333		
	T.C.	21.73	11.84	**0.000**	**0.000**		

Table 4.17: Results of Comparison Between Early Head erect and Early Dumping (Results when $p <= 0.05$ or $\Sigma t^2 > 1000$ indicated in **bold**)

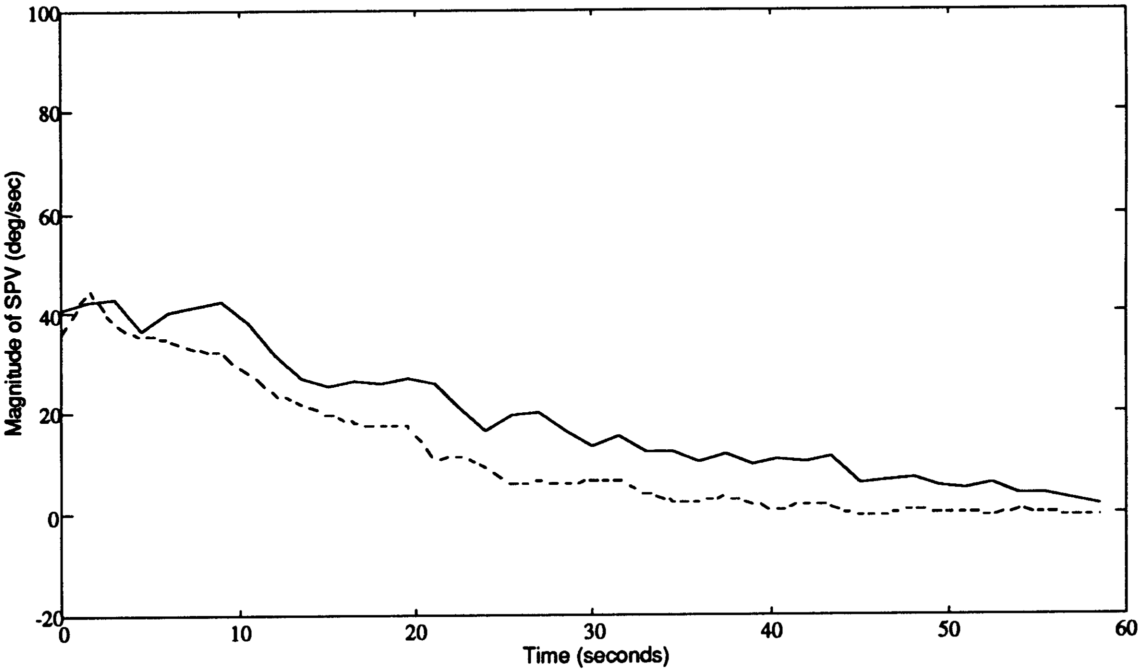


Figure 4.36: Subject T, Early Head erect (solid line) and Early Dumping (dotted line) SPV response

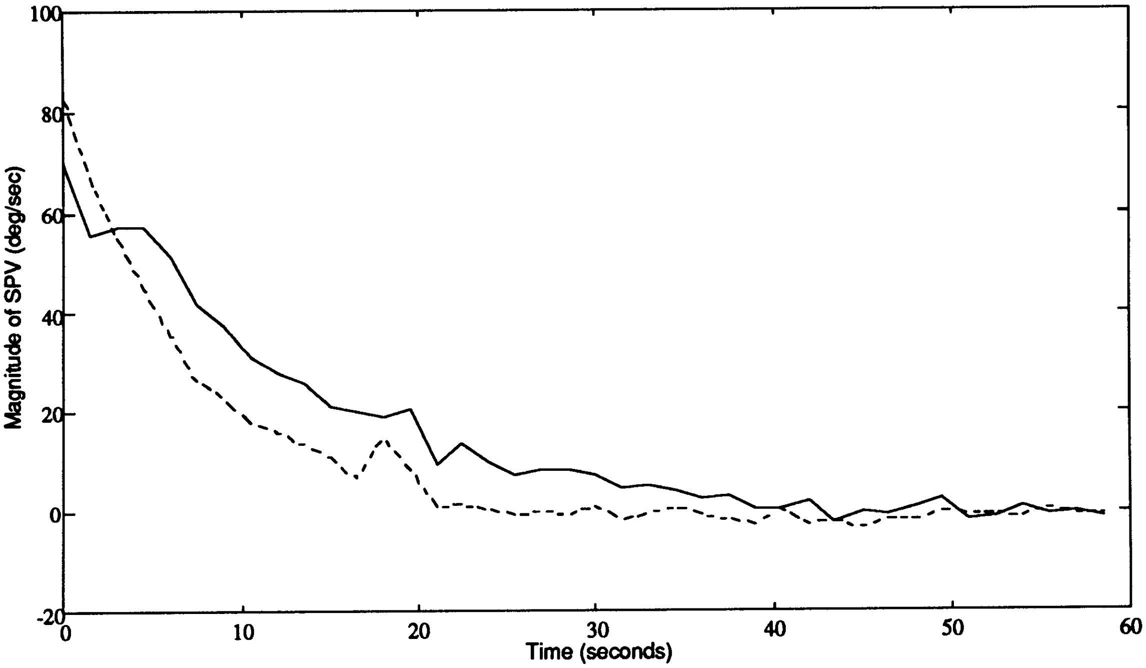


Figure 4.37: Subject V, Early Head erect (solid line) and Early Dumping (dotted line) SPV response

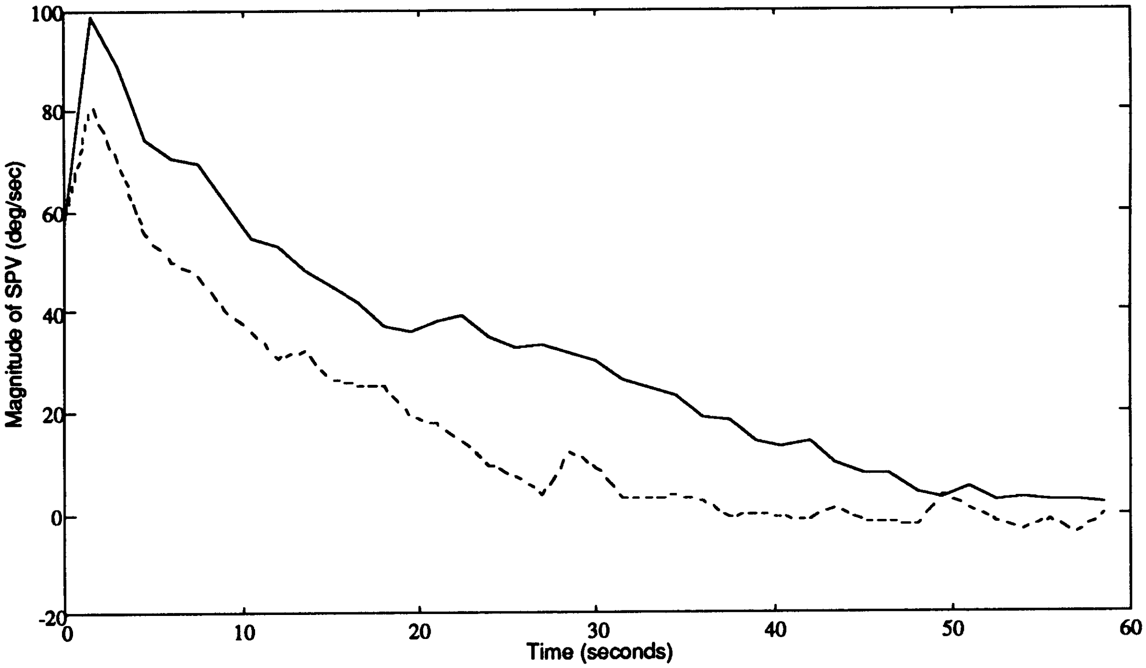


Figure 4.38: Subject Y, Early Head erect (solid line) and Early Dumping (dotted line) SPV response

4.5.10 Changes in early post-flight dumping responses

The difference between early post-flight dumping and pre-flight dumping SPV responses could not be shown to be statistically significant for three subjects (T, V, Y). (Table 4.18, Figures 4.39-41) Subject X data was not analyzed due to an extensive amount of outliers.

				Parametric		Ensemble	
Subject		Pre-DMP	Early-DMP	t-test p <=	KS p <=	sum t2	KS p <=
T	Ensemble					247.475	0.517
	Gain	0.38	0.46				
	T.C.	15.67	14.36				
	Parametric						
	Gain	0.46	0.47	0.987	0.250		
	T.C.	14.65	13.82	0.459	0.500		
V	Ensemble					228.048	0.138
	Gain	0.64	0.81				
	T.C.	7.97	6.63				
	Parametric						
	Gain	0.69	0.87	0.112	0.750		
	T.C.	7.85	6.40	0.104	0.750		
Y	Ensemble					**1297.88**	0.976
	Gain	0.67	0.77				
	T.C.	10.96	11.83				
	Parametric						
	Gain	0.70	0.77	0.467	0.600		
	T.C.	11.19	11.84	0.487	0.400		

Table 4.18: Results of Comparison Between Pre-flight Dumping and Early Dumping (Results when p <= 0.05 or Σt^2 > 1000 indicated in **bold**)

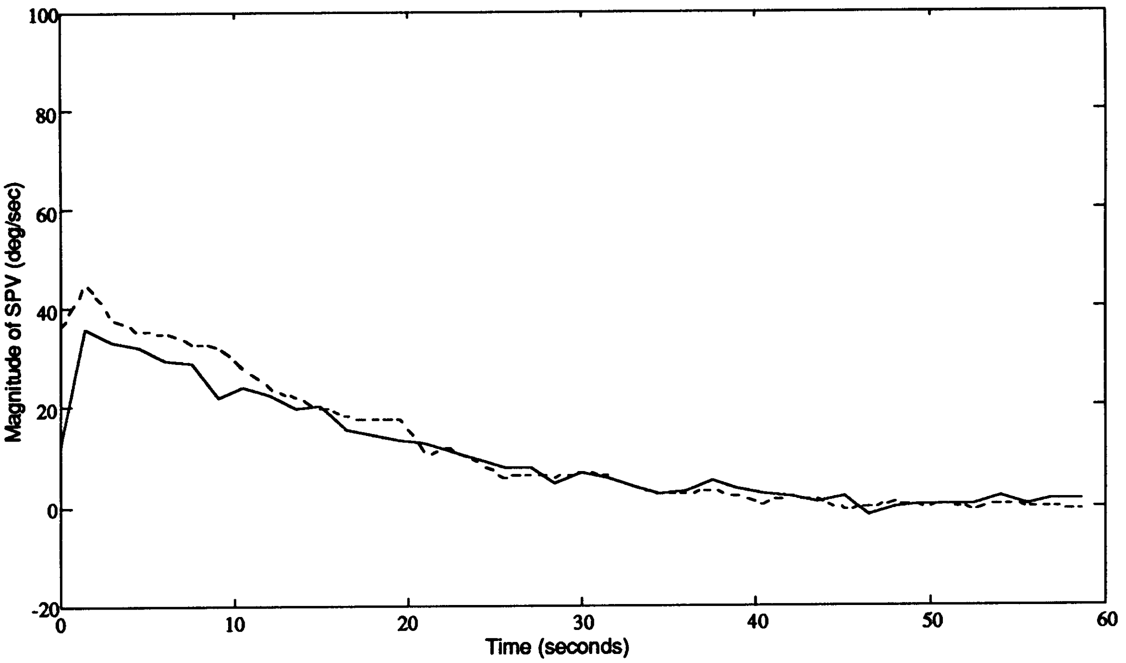


Figure 4.39: Subject T, Pre-flight Dumping (solid line) and Early Dumping (dotted line) SPV response

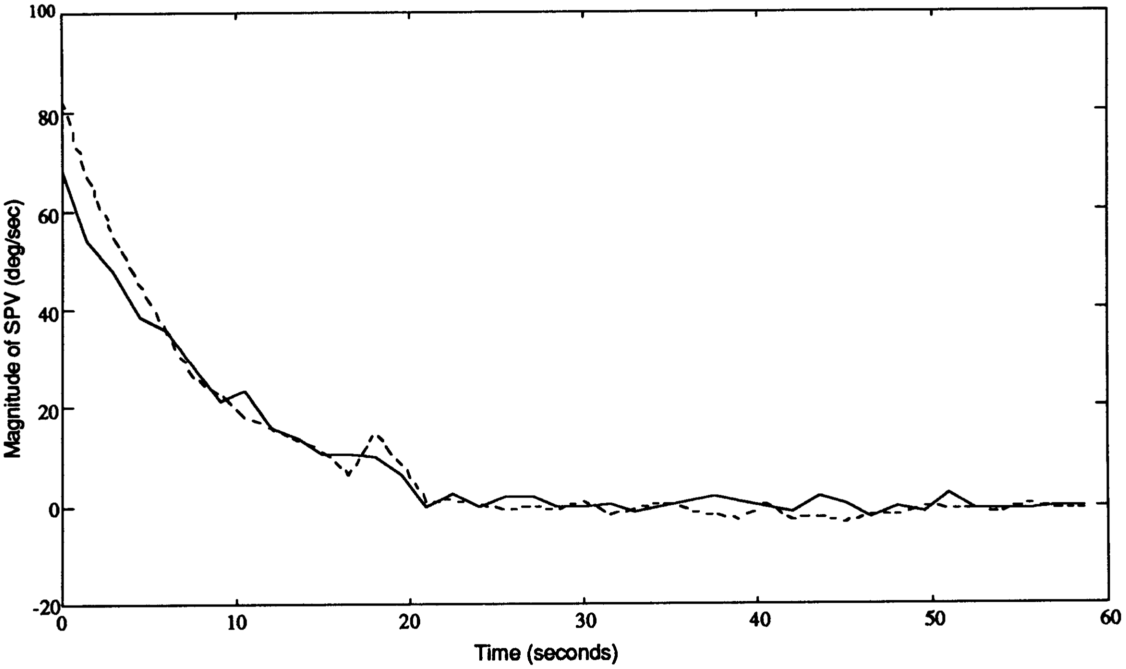


Figure 4.40: Subject V, Pre-flight Dumping (solid line) and Early Dumping (dotted line) SPV response

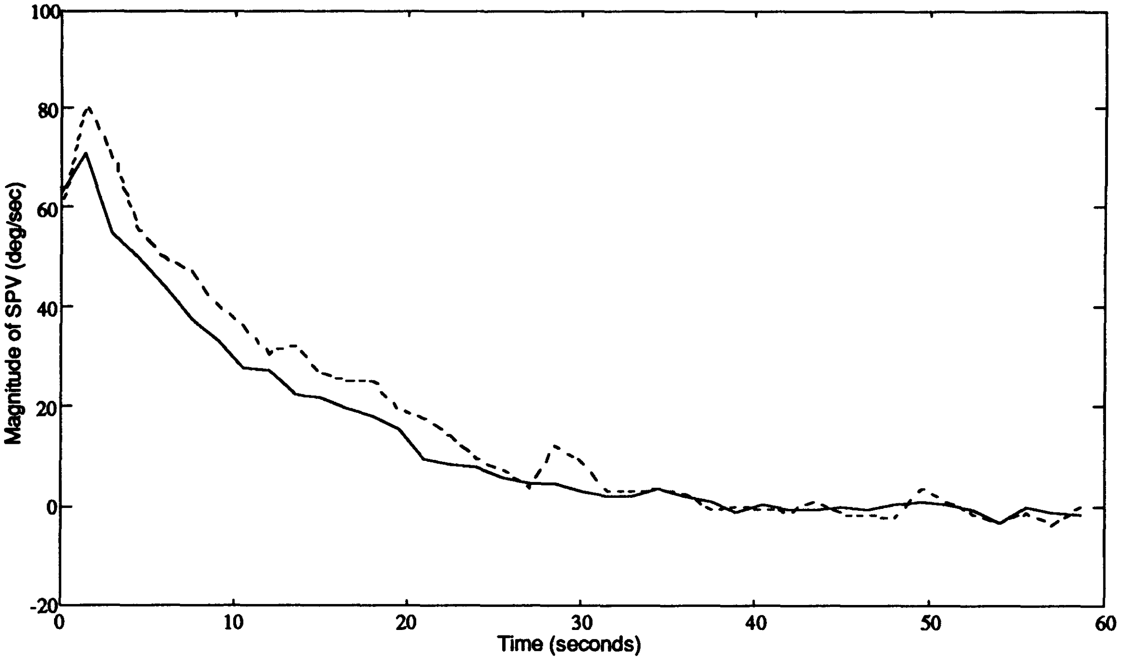


Figure 4.41: Subject Y, Pre-flight Dumping (solid line) and Early Dumping (dotted line) SPV response

Chapter 5: Discussion and Conclusion

This chapter reviews the experimental results in the context of the eight major scientific and engineering questions associated with this study.

5.1: "How did subject's pre-flight head erect VOR responses compare with results from previous studies ?"

SPV responses of all subjects showed intervals of response "dropouts", as noted by Balkwill (1992) and Liefeld (1993) on previous missions. On average, the outlier detection and removal algorithm removed 34% of the original SPV data (T:34%; V:29%; X:47%; Y:27%) . Subjects V and Y had the most consistent VOR. Their percentage of runs rejected pre-flight (Table 4.2) was 19 and 27 % respectively. Subjects T and X results were less consistent. Their percentage of run segments rejected were 66 and 77 % respectively. (Table 4.2). Even after outliers were removed in this way, the standard deviation of the remaining gain and time constant data was 20-40% of the mean, as shown in Figures 5.1-5.4 below. This variability is consistent with the results of Oman and Calkins (1993) when IML-1 astronauts were repeatedly tested, and somewhat different run segment rejection rules were used. The large coefficient of variation of parametric values in this apparently "normal" population of astronauts demonstrates the intrinsic variability of the human VOR response, a problem often ignored in other studies.

Mean post-rotatory apparent time constants were 28.7, 13.7, 20.3, and 18.1 sec for Subjects T, V, X, and Y respectively. In the earlier SL-1 study, Kulbaski (1986) found time constants in the range between 7.4 and 16.7 seconds. On IML-1, time constants between 11.5 and 15.9 seconds were reported (Oman and Calkins, 1992). However, time constants in the latter two studies were computed using log-linear regression. Liefeld's (1993) re-analysis of the same SL-1 data using a linear, first order model similar to that used in this study showed longer time constants (11.3-17.5 sec). It was concluded that the apparent time constants were likely within the range expected based on previous work, when differences in model fitting techniques were taken into account.

As expected based on previous studies (e.g. Balkwill, 1992, p.148), 3/4 subjects showed an eventual reversal of per-rotatory SPV just prior to chair stop. The mean per-rotatory apparent time constants of all 4 subjects were shorter than the corresponding post rotatory time constants by an average of 7.0 sec. This difference in apparent time constant when the SPV data is fit with a first order model (Eqn. 1.5) can be attributed to the presence of neural adaptation in the actual responses (corresponding to T_a in Eqn 1.4).

Preflight responses were examined for trends in gains (Figs. 5.3 and 5.4) and time constants. As on previous missions, no consistent trends were seen. Pre-flight responses were also examined for asymmetries associated either with the direction of rotation stimulation, or the direction of SPV response. A small amount of response asymmetry is not unusual. Subject T's SPV response was reduced for leftward SPV(Figs. 4.1-4.2). ANOVA (Table 4.7) demonstrated a statistically significant reduction in time constant in the leftward SPV direction. Based on this subject's medical history, the asymmetry was thought to be oculomotor in origin. Since effect of gravitational exposure is likely similar for both response directions for all subjects, including Subject T, CW and CCW responses under each condition were averaged together for further analysis (Section 4.4).

5.2: "How did the head-erect SPV response change in parabolic flight ?"

On SLS-1, it was not possible to test the subjects in parabolic flight to evaluate their head erect VOR response after an brief exposure to weightlessness. However, on SLS-2, three of the four subjects were tested in the NASA KC-135 in 0-G parabolic flight, and in the aircraft on the ground in 1-G. The mean head erect time constants of all three subjects shortened in weightlessness. Subjects V and Y showed a significant shortening of the head erect time constant (Section 4.5.1). The ratio of the 0-G/1-G KC-135 time constants was 60% (98 %, 51 %, and 32 % for subjects T, V, and Y respectively). The average time constant ratio (60 %) compares closely to the value of 69% found by DiZio and Lackner (1988) in a population of 15 subjects. In the DiZio and Lackner study individual subject mean responses and their standard deviations were not reported. Hence, the present study has shown for the first time that the population trend demonstrated by DiZio and Lackner is true for the majority of subjects tested as individuals.

Subject T's results were notable because of the large variation in the 0-G time constant data. Four of the data points clustered in the 20-25 second range, suggesting that the time constant shortened on these runs. Three other runs produced more scattered data with longer time constants, including one (not plotted in Fig. 5.1) of 60.8 sec. It may be that T's average time constant was skewed by one or two aberrant runs, and that if the KC-135 testing was performed again, this subject would probably show a shorter mean time constant.

The percentage of run segments rejected in parabolic flight due to dropouts and outlier parameter values (T:25%, V:7%, and Y:15%). These percentages were lower than in preflight testing, possibly due to the excitement of the weightless experience, or subjects use of scop-dex.

It was not possible to demonstrate a significant difference between 0-G and 1-G gain parameters (Figs 5.3, 5.4 and Table 4.9). Absence of a significant change in gain is consistent with previous studies by Jackson and Sears (1965), Vesterhauge, et al (1984), and DiZio and Lackner (1988).

As reviewed in Section 1.1, it has been hypothesized that the reduction in post-rotatory time constant observed in zero-gravity parabolic flight is due to vestibular conflict. The semicircular canals presumably function normally, while the otoliths do not respond in a familiar way during rotation. It was hypothesized that when this vestibular conflict occurs, velocity storage is lost, causing a shorter apparent time constant of the post-rotatory VOR.

5.3: "How did the head-erect SPV response change in prolonged weightlessness ?"

Although technical difficulties were encountered with in-flight EOG recording, the SLS-2 in-flight data set was somewhat larger than that obtained on SLS-1. The SLS-2 analysis differed from SLS-1 in that first order model fits were performed on individual runs using a method identical to ground testing, and both per- and post-rotatory in-flight data was analyzed. On SLS-2, this permitted assessment of inter subject differences and the statistical significance of the changes seen.

Had data from Subjects Y and T on FD10 been available, it would have been possible to test for in-flight adaptive trends in all four subjects. Data was available for Subjects V and X (Fig. 5.5) on both flight days. From the limited available data, no clear adaptive trends in the per-rotatory or head erect time constant data could be discerned. Therefore, data from FD4 and FD10 were considered together for subsequent analysis.

Had it been possible to test the subjects immediately after reaching orbit, presumably data similar to that recorded in parabolic flight would have been obtained.

As compared to preflight control data, after 4-10 days in orbit, subjects V and Y did not show a significant decrease in their head-erect and per-rotatory time constants (Fig. 5.2; Sections 4.5.2 and 4.5.3) In fact, Subject Y's per-rotatory and head erect VOR time constant actually increased somewhat in flight, and so apparently did Subject Y's subjective sensation. Subject Y commented during (Flight Day 4) testing that "comparing to 1-G..it felt like I was turning longer. The time constants seem longer up here in space." Subject V's head erect time constant was 13.7 sec on the ground, and 13.5 sec. in orbit. Subject Y's head erect time constant was 18.1 sec. on the ground, and 22.2 sec. in orbit. These results differed from 0-G parabolic flight, where these same two subjects showed a significant decrease in head-erect time constant (51%, 32% respectively). These two subjects thus showed a mean response similar to three of the four subjects in Oman and Balkwill's (1993) SLS-1 study.

Subjects T and X responded differently than the other two subjects in orbit. Their mean head erect and per-rotatory time constants decreased in-flight. For example, Subject T's head erect time constant was 58% of the preflight value (28.7 sec on the ground and 16.6 in orbit). Subject X's time constant was 72% of the preflight value (20.3 sec. on the ground and 14.6 sec. in orbit). Subject T's per-rotatory and head erect time constant decreases were statistically significant while subject X's were not, perhaps due to the small number of in-flight runs and large number of dropouts. (Sections 4.5.2 and 4.5.3).

As shown in Figs. 5.3 and 5.4, only Subjects T and X showed noticeable decreases in gain in orbit. However, when interpreted against the background normal variation in gain parameters, these changes were not statistically significant (Table 4.10-11). As shown in Figs. 4.9-4.12, the initial peak of the ensemble average SPV envelopes appear almost identical for all 4 subjects, suggesting that the VOR gain was unchanged. This result is similar to that found by Oman and Balkwill (1993) on SLS-1.

In contrast, on IML-1, Oman and Calkins (1993) found that the in-flight VOR gain had increased, at least when the 4 subjects were considered as a group. They reported a rank correlation between in-flight SMS intensity, gain increase, and time constant decrease (Oman and Calkins, 1993) in-flight. However, all 4 IML-1 subjects had experienced some symptoms of space motion sickness while in orbit. Liefeld (1993) noted a trend toward increased post-flight VOR gain and decreased time constant among those who had been sick in flight. The SLS-2 crew reported virtually no in-flight symptoms of SMS, so their in-flight sickness intensity could not be ranked. The change in head erect time constant of our SLS-2 subjects showed an approximate rank correlation with change in gain. Ranked in terms of time constant change, the order of the subjects was T, X, V, Y, and in terms of gain change, the order was T, X, Y, V. Subject V and Y's gain scores were within 4% of each other, and had the rank order been reversed, the rank correlation would have been statistically significant. If there is a causal relationship between in-flight gain increase and space motion sickness, and our SLS-2 subjects had little or no gain increase, it would be surprising that they experienced little space sickness.

As noted in Section 4.2, the percentage of run segments rejected due to dropouts and outlier parameter values (T:19%, V:15%, X:13%, Y:0%) was lower in flight than in preflight testing. Perhaps this was the result of a generally higher level of attentiveness associated with performing the experiment in flight after two years of training.

The SPV response asymmetry in mean post-rotatory time constants observed in Subject T preflight (Sect. 4.4) was also present in flight, but the corresponding asymmetry was not seen in per-rotatory responses. Since the number of runs was relatively small, the effect could not be reliably observed in-flight. It is interesting that Subject T was the one subject who showed only a small mean time constant decrease in parabolic flight, but did show a larger decrease when tested in orbit.

The principal conclusion, then, is that both per-rotatory and head erect time constants increased so they were equal to or larger than pre-flight values in half the subjects (V and Y) studied. A reason (Oman and Balkwill, 1993) why velocity storage might return is that as the CNS adapts to weightlessness, the otolith cue during rotation becomes familiar and sensory conflict disappears. Presumably vestibular conflict remained in Subjects T and X, making their apparent time constants shorter than preflight values.

5.4: "How did the head erect SPV response change post-flight ?"

Post-flight changes in the VOR may be difficult to measure experimentally, since it appears that most of the re-adaptation occurs within several hours after landing, prior to the time post-flight VOR testing becomes possible. Residual post-flight changes may be small, and difficult to demonstrate statistically against the normal background variability in VOR parameters. To see changes, results from several test sessions were averaged together. On SLS-2, R+0 VOR testing did not begin until at least 12 hours after landing in California, since the crew had to return by air to Houston.

Some changes can be demonstrated in the pooled "early post-flight" data (R+0, 1 and 2). Subjects T and X's early post-flight per-rotatory time constants were significantly shorter (Figure 5.1 and Table 4.13). Subject T's mean time constant was 67% of the preflight value, and Subject X's time constant was 79% of preflight. Subject T and X's early post-flight mean head erect time constants were also shortened (81% and 40% respectively) from preflight values, though the changes were not significant. Both subject's' ensemble average SPV profiles lay below their pre-flight profiles (Table 4.12, 4.13, Fig. 4.17, 4.20, and Fig. 4.22), though the changes were not demonstrably significant with KS tests.

Subjects V and Y's early post-flight per-rotatory and head erect (Fig. 5.2) time constants were apparently unchanged from their pre-flight values and their SPV profiles did not so consistently lie below their pre-flight profiles. (Tables 4.12, 4.13, and Figures 4.18, 4.19, 4.21, and 4.23).

By late post-flight, the subjects per-rotatory and head erect responses returned to values statistically indistinguishable with pre-flight. (Figures 5.1 - 5.4). Late per-rotatory mean time constants were 79%, 87%, and 101% of preflight values for Subjects T, Y, and V respectively. Similarly, late head erect mean time constants were 104%, 91%, and 88% of preflight values.

The outlier detection and removal algorithm removed 36% of the original early post-flight SPV data (T:39%, V:26%, X:55%; Y:24%). We had expected that post-mission "let down" might increase the frequency of occurrence of dropouts, but if this happened, the effect was no greater than preflight As in preflight testing, Subject V and Y had the most consistent VOR as compared to T and X. The percentage of early post-flight runs rejected were T:47% ; V:13%; X:84%; Y:33%. Thus, no consistent change was seen in the run rejection rate comparing preflight and post-flight results.

The original SL-1 (Oman and Kulbaski, 1988), and the D-1 analysis (Oman and Weigl, 1989) showed that the post-flight head-erect responses were reduced. In the in the SL-1 and D-1 analysis, the automated outlier detection algorithm was not used. There was some concern that if dropouts occurred with greater frequency post-flight, and had been missed, it might have caused the post-flight envelope to fall below pre-flight. Subsequent SLS-1 and SL-1 analysis with automated outlier detection (Balkwill, 1992; Liefield, 1993), showed that some subjects had reduced post-flight responses, while some subjects did not, which is consistent with the present SLS-2 experiment results. Since the dropout and run rejection rates on SLS-2 were similar both pre and post-flight, one can infer that at least a portion of SPV changes seen were not a fatigue artifact.

If the preflight, parabolic flight, in-flight, and early post-flight results are considered together, the combined data suggest the interpretation that subjects V and Y are "rapid adapters", since they showed some loss of velocity storage in parabolic flight, but a return of velocity storage when tested in orbit, and a return of velocity storage when tested during the first three days post-flight. In contrast, Subjects X and T are "slow adapters" since they continued to show loss of velocity storage after several days in orbit, and also in early post-flight testing.

5.5: "Was the pre-flight dumping response consistent with previous studies ?"

During the pre-flight, subjects T, V, and Y showed that the post-rotatory dumping time constant was significantly shorter than the post-rotatory head erect time constant. (Section 4.5.6) The fourth subject, subject X, showed a trend towards shortening, but this was not significant. The ratios of the dumping to head erect time constants were T: 51 %; V:57 %, X:72 %; and Y:62 %, and averaged 61 %. By comparison, Benson and Bodin (1966a) found the ratio of dumping to head erect time constants was 41 % for a population of 8 subjects tested with 60 deg/sec rotational stimuli. When subjects are rotated about an earth horizontal axis, the time constant is 48% of the head erect value (Benson and Bodin, 1966b). Thus, the time constant changes seen were qualitatively similar but on average quantitatively somewhat smaller than those reported by Benson and Bodin. The quantitative difference might be due to the different rotational speed used, the different SPV analysis method employed, and the fact that the whole body rather than just the head was tilted in the Benson and Bodin studies. As noted earlier (Section 1.1), the shortening of the apparent time constant has been attributed to sensory conflict and loss of velocity storage. With the head horizontal, the otoliths signal that the head is stationary, but pitched forward 90 deg, whereas the post-rotatory semicircular canal SSC cue indicates continuous rotation about an off vertical axis.

5.6: "How did the dumping SPV response change in prolonged weightlessness ?"

The dumping time constants of Subjects V and Y were significantly larger in orbit than on the ground (Table 4.16 and Fig. 5.2). The data from Subjects X and T showed an opposite trend (Table 4.16 and Fig. 5.1). Comparing the in - flight head erect and dumping time constants, however, it was not possible to show statistically significant differences (Table 4.15). Nonetheless, the data showed interesting trends: The mean dumping time constants of Subjects T and X were shorter than their head erect time constants by 4.5 sec and 3.6 sec, respectively. The in-flight head erect and dumping time constants of the other two subjects were essentially similar.

If one hypothesizes that Subjects V and Y were fast adapters, and that they no longer expected static otolith cues to change in the presence of semicircular canal cues, then one would expect that their flight dumping and head erect time constants would be identical, and that their head erect time constants preflight and in flight would also be the same. Taken together with the result that their preflight head erect time constant was greater than their preflight dumping time constant, one would expect that their in-flight dumping time constant would be greater than their preflight dumping time constant. This was in fact observed (Table 4.16, and Figure 4.33 and 4.35).

Since it was thought that subjects T and X were slow adapters, one would expect that velocity storage was already lost in 0-G due to the unfamiliar otolith cues when the head was in the erect position, and that in-flight dumping time would cause no further reduction in time constant. However, if these subjects were attending to tactile cues when making a dumping head movement, it might explain the trend towards a shorter dumping time constant as was actually observed in flight. For these subjects, the dumping observed in flight was greater than that produced on the ground (Table 4.16 and Figures 4.32, 4.34), perhaps due to the combined tactile and otolith sensory conflict.

When tested on FD4, Subject T remarked that subjective rotation sensation disappeared quickly. On FD10, post dumping sensations were more persistent, but VOR data from this session was not recorded. On FD4 Subject Y reported that sensation did not disappear immediately after head movement, and Subject V noted that post-rotatory dumping sensation continued, about an axis that moved with the head. Subject X felt quite disoriented by the dumping head movement. These comments are qualitatively consistent with the VOR results obtained, but apparently differ in some respects from SLS-1 reports. The four SLS-1 subjects described the duration of dumping sensation as much shorter than on the ground. In most cases, rotation sensation disappeared virtually instantaneously, even though VOR responses were prolonged. The SLS-2 subjective reports suggest that dumping is not always instantaneous, and may more closely relate to SPV.

5.7: "How did the dumping SPV response change post-flight ?"

During the early post-flight testing, subjects T, V, and Y showed that their early post-flight dumping time constant was significantly shorter than the early post-flight head-erect time-constant (section 4.5.9), meaning that the dumping response had returned. The early and late post-flight dumping time constants were statistically indistinguishable from pre-flight values. (section 4.5.10 and Figures 5.1 and 5.2). Early dumping mean gains showed a slight but insignificant increase early post-flight.

5.8 "Did near-real time SPV analysis facilitate decision making during testing ?"

Yes, near-real time analysis proved very helpful in maintaining data quality control. Science decision making was enhanced preflight, in-flight, and post-flight on the occasions described in section 4.3. The completion of a near-real time data analysis software pipeline, begun by Balkwill (1992) and Liefeld (1993), and finished by the present author, provided virtually completely automatic analysis of an entire session's worth of data in 4-7 hours. This proved to be a major advance over previous missions.

5.9 "Are there any recommendations for future analysis and future studies ?"

Yes, there are:

1) Automate the EOG calibration process, in order to make the analysis pipeline from EOG collection to model fitting totally automated.

2) Instead of exporting the data to Excel or Systat, see if the new version of Matlab (5.0), can perform the same functions, like calculation of histograms, t-tests, and Kolmogorov-Smirnov tests.

3) Improve on the statistical tests on the ensemble data. Perform a monte-carlo simulation, like the one performed before (Pouliot, 1991), except it would include all the relevant Matlab scripts like AATM and decimation.

4) Perform a giant ANOVA with replications on all the parametric data at once, with subject, run condition, and gravity conditions as factors.

5) Study to see if there is an effect of using the anti-motion sickness drug, scop-dex, on the frequency of dropouts. If there is an effect, scop-dex could be given to subjects for ground testing.

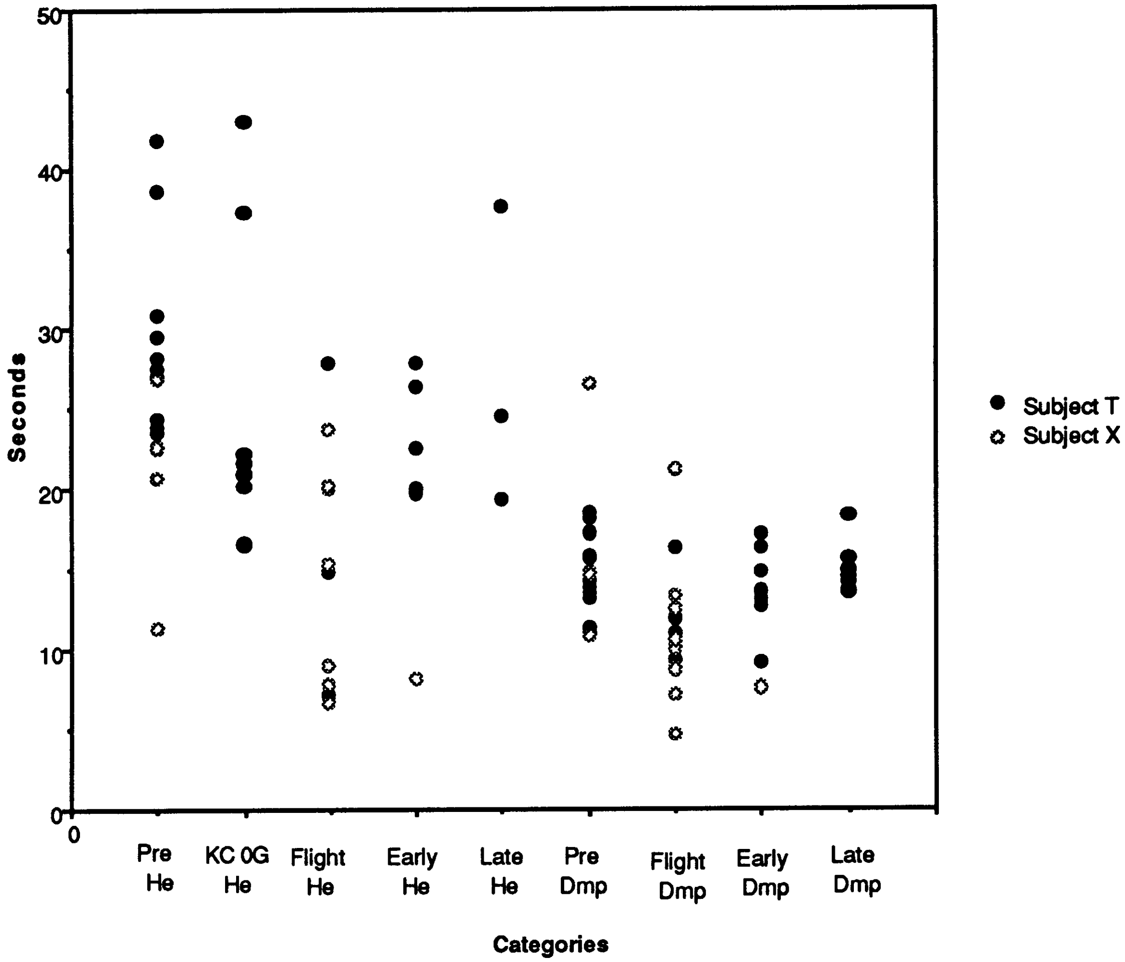


Figure 5.1: Summary of the post-rotatory first order model fit time constant parameters for subjects T and X

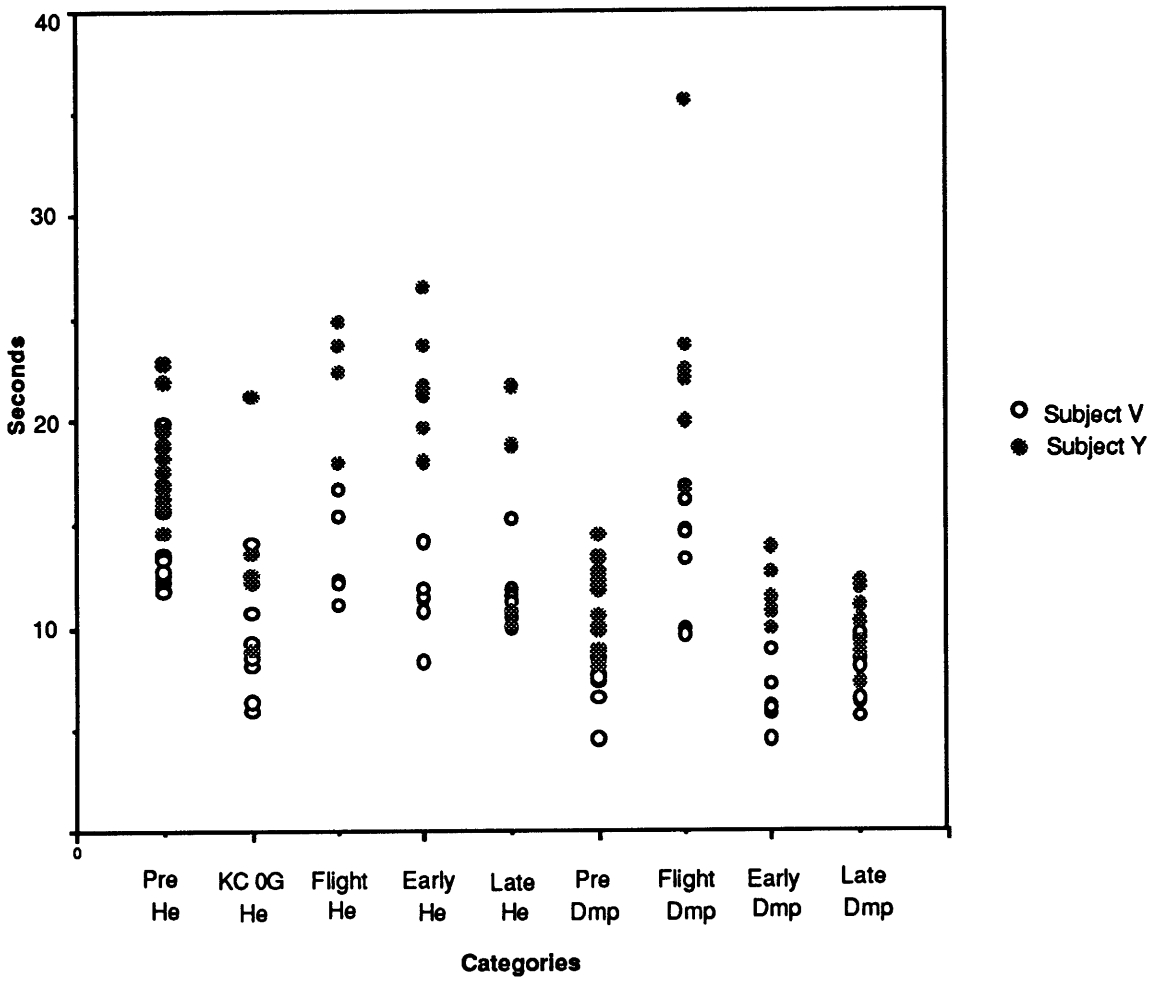


Figure 5.2: Summary of the post-rotatory first order model fit time constant parameters for subjects V and Y.

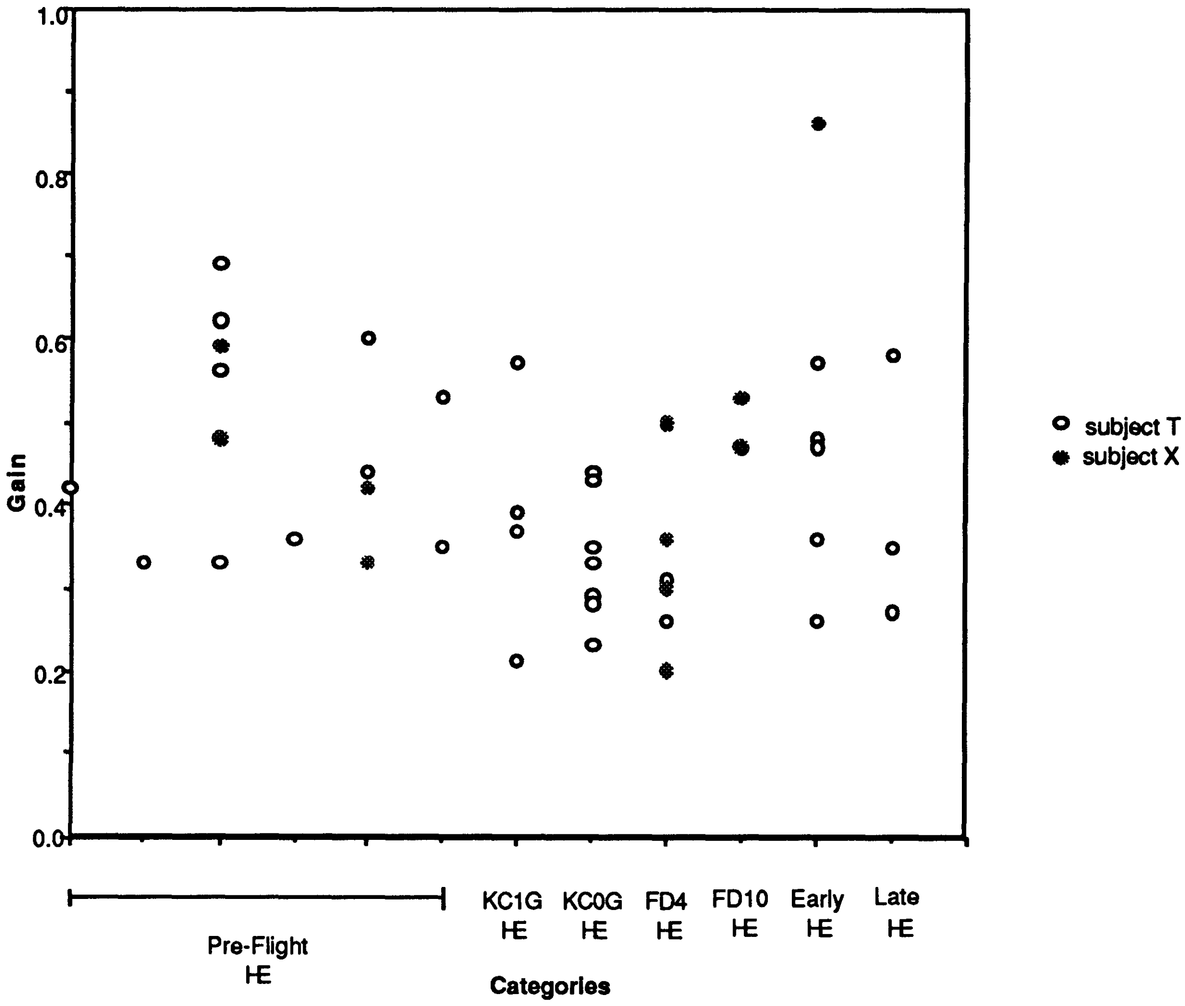


Figure 5.3 Summary of the post-rotatory first order model fit gain parameters for subjects T and X

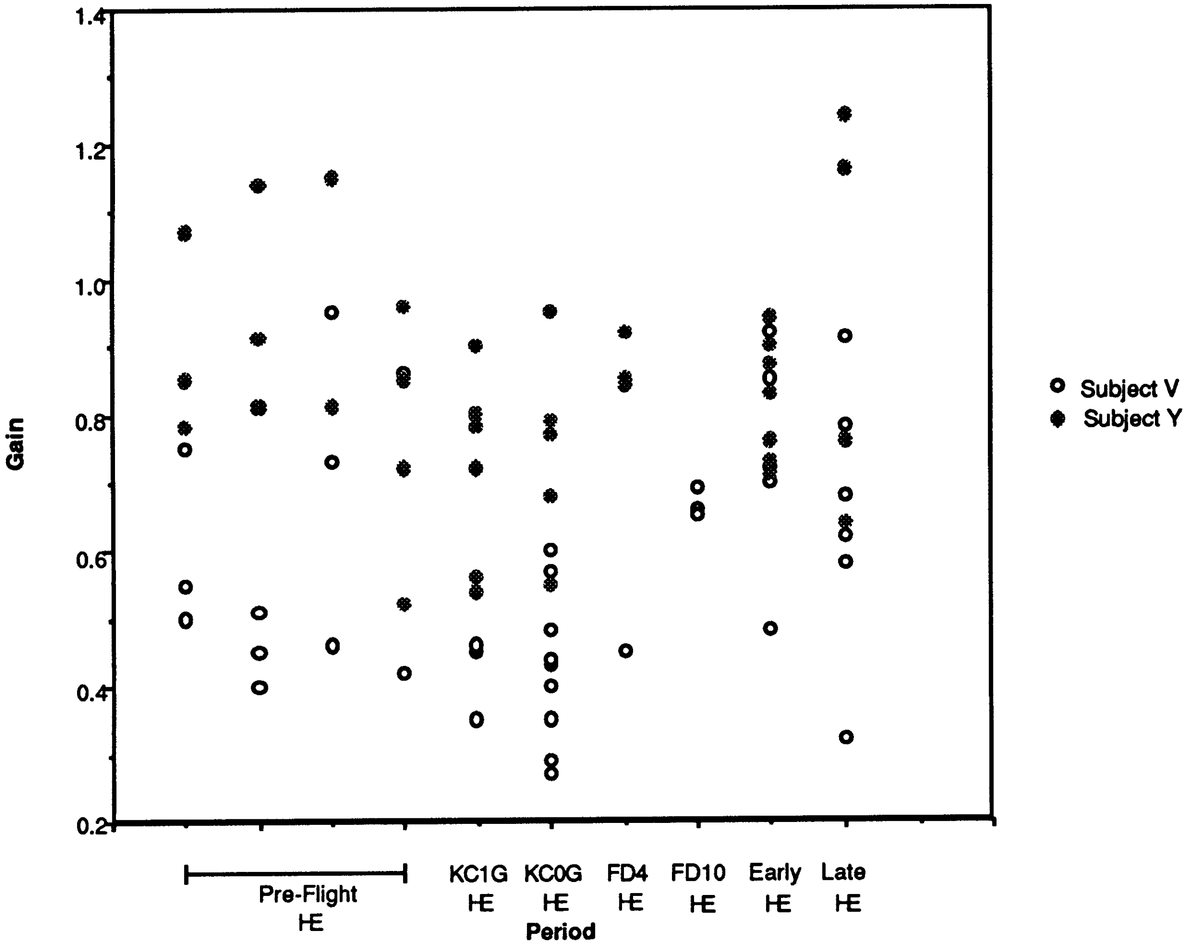


Figure 5.4 Summary of the post-rotatory first order model fit gain parameters for subjects V and Y.

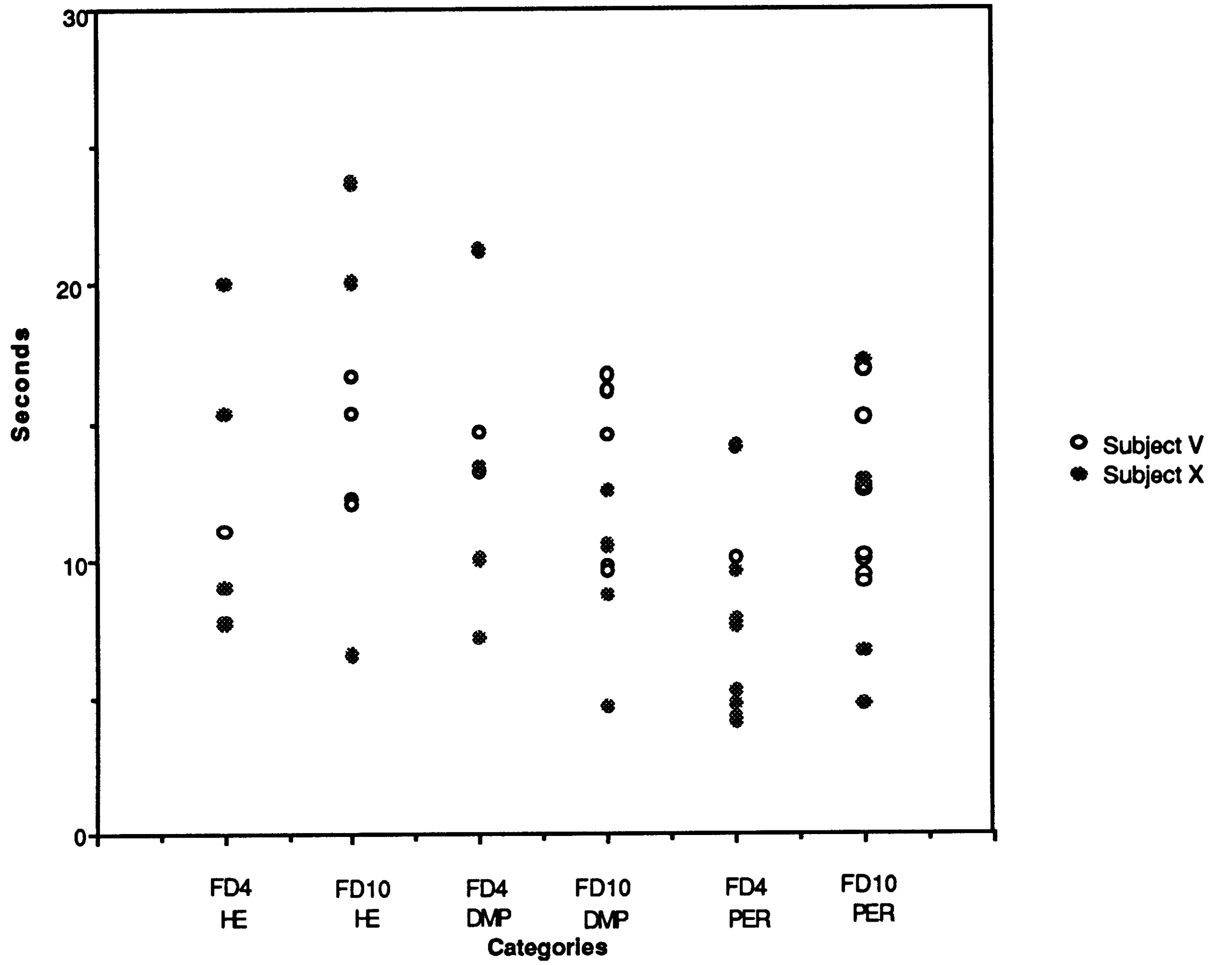


Figure 5.5 Summary of in-flight first order model time constant parameters for Subject V and X.

Bibliography

Balkwill, M., 1992, "Changes in Human Horizontal Angular VOR after the Spacelab SLS-1 Mission," S.M. Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA.

Benson, A., Bodin, M., 1966a, "Comparison of the Effect of the Direction of Gravitational Acceleration on Post-Rotational Responses in Yaw, Pitch, and Roll," Aerospace Medicine, 37:889-897.

Benson, A., Bodin, M., 1966b, "Effect of Orientation to the Gravitational Vertical on Nystagmous Following Rotation About a Horizontal Axis," Acta Otolaryngologica, 61:517-526.

Clement, G., Vieville, T., Lestienne, F., Berthoz, A., 1985, Preliminary results of "Equilibrium and Vertigo Experiment Performed During STS 51-G Shuttle Flight," 2nd International Conference on Space Physiology, Toulouse, France.

Collins, W., 1962, "Effects of Mental Set Upon Vestibular Nystagmus," Journal of Experimental Psychology, 63:191.

DiZio, P., Lackner, J., 1988, "The Effects of Gravitoinertial Force Level and Head Movements on Post-Rotational Nystagmous and Illusory After-Rotation," Experimental Brain Research, 70:485-495.

Engelken, E., Stevens, K., 1990, "A New Approach to the Analysis of Nystagmous: an Applacation for Order Statistic Filters," Aviation, Space, and Environmental Medicine," 61(9):859-864.

Fernandez, C., Goldberg, J., 1976, "Physiology of Peripheral Neurons Innervating Otolith Organs of the Squirrel Monkey. I. Response to Static Tilts and to Long Duration Centrifugal Force," Journal of Neurophysiology, 39:970-984.

Groen, J. , 1962, "Inhibitory Mechanism of the Vestibular System in Man in Comparison with Hearing," J. Acoust. Soc. Am. 34(9):1497-1503

Jackson, M., Sears, C., 1965, "The Effect of Weightlessness Upon the Normal Nystagmus Reaction," Aerospace Medicine,37:719-721.

Kass, J.R., Bruzek, W., Probst, T, Thumler, R., Vieville, T, Vogel H., 1986, "European vestibular experiments on the Spacelab-1 mission: 2. Experimental equipment and methods," Experimental Brain Research, 64:247-254.

Kornilova, L, Grigorova, V, Bodo, G., 1993, "Vestibular Function and Sensory Interaction in Space Flight," Journal of Vestibular Research, 3:219-230.

Kulbaski, M., 1983, "Effects of Weightlessness on the Vestibulo-Ocular Reflex in the Crew of the Spacelab 1," S.B. Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA.

Liefield, T., 1993, "Changes in Human Horizontal Angular VOR after the Spacelab SL-1 Mission," S.M. Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA.

Massoumnia, M., 1983, "Detection of Fast Phase of Nystagmous using Digital Filtering," S.M. Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA.

Oman, C., Balkwill, M., 1993, "Horizontal Angular VOR, Nystagmous Dumping, and Sensation Duration in Spacelab SLS-1 Crewmembers," Journal of Vestibular Research.

Oman, C., Calkins, D., 1993, "Effect of Orbital Flight on the Human Horizontal Vestibular Vestibulo-Ocular Reflex Response to 120 deg/sec Step Stimuli," Final Report, Microgravity Vestibular Investigations, pp. 33-54.

Oman, C., Kulbaski, M., 1988, "Spaceflight Affects the 1-g Post-rotatory Vestibulo-Ocular Reflex," Advanced Oto-Rhino-Laryng, 42:5-8.

Oman, C., Weigl, H., 1989, "Post-flight Vestibulo-Ocular Reflex Changes in Space Shuttle/Spacelab D-1 Crew," abstract 21, Aerospace Medical Association 60th Annual Scientific Meeting, Washington, D.C., May 7-11.

Pouliot, C., 1991, "Summary of the Sigma T-squared Simulation", Massachusetts Institute of Technology Man Vehicle Laboratory Internal Report.

Raphan, T., Matsuo, V., Cohen, B., 1979, "Velocity Storage in the Vestibulo-Ocular Reflex Arc (VOR)," Experimental Brain Research, 35:229-248.

Siegel, S., 1956, "Non-parametric Statistics for the Behavioral Sciences," McGraw Hill Book Company, New York.

Thornton, W., Uri, J., Moore, T., Pool, S., 1989, "Studies of the Horizontal Vestibulo-Ocular Reflex in Spaceflight," Arch Otolaryngol, Head and Neck Surgery, 115(8):943-949.

Vesterhauge, S., Mansson, A., Johansen, T., 1984, "Vestibular and oculomotor function during Gz variations,"Motion sickness: mechanisms, prediction, prevention, and treatment in Williamsburg, VA, NATO AGARD CP-372, 24.1-24.4.

Watt, D, Money, K., Bondar, R., Thirsk, R., Garneau, M., Scully-Power, P., 1985, "Canadian Medical Experiments on Shuttle Flight 41-G," Canadian Aeronautics and Space Journal, 31:215-226.

Appendix A: Ground data analysis Matlab scripts.

Batch_analyse
Cal_factor_gen
Cal_from_file
Calibrate
Calibration_calc
CODES
*** Collect
*** Combine
*** Declow
*** Dec_mean
*** Dec_mean_report
*** Dec_report
Delta_tach
Dec30
*** Graph
Ind_model_fit_exp
Lin_fit
Log_outlier
Mag_outlier
Mean_sep_exp_fit
Model_err_exp
Multiple_AATM
*** New_tsq
Pack_true
*** Parse
Pick_regions
Searchb
*** Stat
Stat_prep_batch
*** Tachan_batch
Three_point
Tsq_pvalues
*** T_square

*** indicates that the present author has written or has made major modifications to these scripts.

```
% Batch_analyse

% written by T. Liefeld throughout spring 93
% given a folder of runs from a BDC,  this functions as a
% superscript that will prompt the user for all analysis
% from data collection through to model fitting.
%
% slight organizational organizational modifacations by C. Pouliot 10/94

clear
hold off

hard_disk = 'SLS_HD';

data_path=input('Enter Data Path           >> ','s');
sub_code =input('Enter Subject Code   >> ','s');
number =  input('Enter Number of Runs >> ');

convert = input('Do you want to convert the data to MatLab format? >> ','s');
if ((convert == 'y') | (convert == 'Y'))
        code = [data_path,sub_code];
        mexchair_convert(code,number);
end


q1 = input('Do you want to do a calibration? >> ','s');
if ((q1=='y') | (q1=='Y'))
        q = input('Calibration factors from file or new? (f/n) >> ','s');
        if ((q == 'f') | (q == 'F'))
                cal_from_file
        end
        if ((q == 'n') | (q == 'N'))
                cal_factor_gen
        end
end;

q2 = input('Do you want to perform AATM? >> ','s');
if ((q2 == 'y') | (q2 == 'Y'))
        multiple_AATM;
end

% create the run_code matrix, codes
CODES
number = number-n_cals;

q3 = input('Do you want to perform Tachan >> ','s');

if ((q3 == 'y') | (q3 == 'Y'))
        for i = 1:number
                run_code = codes(i,:);
                fprintf(['\nRun code = ',run_code,'\n']);
                tachan_batch;
        end
end

q4 = input('Do you want to perform stat prep >> ','s');

if ((q4 == 'y') | (q4 == 'Y'))
```

```
    for i = 1:number
        run_code = codes(i,:);
        fprintf(['\nRun code = ',run_code,'\n']);
        stat_prep_batch;
    end
end

q5 = input('Do you want to fit a Model? >> ','s');
if ((q5=='y') | (q5=='Y')),
    for i = 1:number,
        run_code = codes(i,:);
        ind_model_fit_exp;
    end
end
```

```
% Cal_factor_gen
%
% calls calibrate for a number of runs and generates the cal factors
% for the PRN and dumping runs
%
% by T. Liefield

run = ones(1,number);
n_cals = input('How many cals >> ');
dim = 1;

if (n_cals >= 0)
        fprintf('\n Enter the run number for the cals in order')
        fprintf('\n from lowest to highest')
        for i=1:n_cals
                calnum(i) = input('cal #  >> ');
                run(calnum(i)) = 0;
                if (calnum(i) < 10)
                        n = num2str(calnum(i));
                        cal_code = [data_path,sub_code,'0',n]
                        calibrate
                        hcal(i) = scale1;
                        g(i) = input('Was this cal good enough to use >> ','s');
                else
                        if (calnum(i)>=10)
                                n = num2str((calnum(i)));
                                cal_code = [data_path,sub_code,n]
                                calibrate
                                hcal(i) = scale1;
                                g(i) = input('Was this cal good enough to use >> ','s');
                        end
                end
        end
end

% calculate calibration factors for the runs
% based only on the good calibrations
j = 1;
for i = 1:n_cals
  if ((g(i)== 'y') | (g(i) == 'Y'))    % g = chr array good or bad
        g_cal(j) = calnum(i);           % calnum all run# which are cals
        calh(j) = hcal(i);
        j = j + 1;
        end
end

hcal = calh;
calibration_calc

q = input('Would you like to save measured cal values? >>','s');
if ((q == 'y') | (q == 'Y'))
        save_name = input('Save File Name : ','s');
        save_name = [data_path,save_name];
        eval(['save ',save_name,' sub_code number dim g_cal hcal run n_cals hor_cal'])

end
```

```
% Cal_from_file
%
% written by T. Liefeld,  4/93
%
% Takes previously generated calibrations from a file
% specified by the user and generates the vector of
% calibration factors for use with mexAATM4

filename = input(' Enter the filename containing the calibration factors >> ','s');

eval(['load ', data_path filename]);
calibration_calc;
clear filename;
```

```
% Calibrate
%
% This script allows the user to obtain a calibration factor
% in degrees/unit. It assumes a three point calibration in
% each direction, although the zero is optional. For each axis,
% the user must specify the angular deviations and select the
% "flat" regions of the trace which correspond to fixations on
% the targets.
%
% D. Balkwill 11/90


sample = 120;
colour = 'y';

code = cal_code;

fprintf('\nCalibrating Axis#1...\n');
eval(['load ',cal_code,'.eogh']);
pos = eogh;
clear eogh
t = (([1:length(pos)] - 1)/sample)';

[scale1,noise1,offset1] = three_point(t,pos,colour);
fprintf('\nAxis#1 scale factor = %6.4f deg/unit\n',scale1);
```

```
% Calibration_calc
%
% written by T. Liefeld, 22/4/93
% interpolates or extrapolates as necessary to generate the
% calibration factors for runs,  given the number of runs,
% the position of the calibration runs in the series, and the
% calibration factors

if (length(g_cal) == 0)
        fprintf(' No good cals?')
elseif (length(g_cal) == 1)
        for i = 1:number
                hor_cal(i) = hcal(1);
        end
elseif (length(g_cal) >1)
        for i = 1:number
                for j = 1:(length(g_cal)-1)
                        if (i<g_cal(1))  % runs before first cal, extrapolate
                                delta = g_cal(1) - i;
                                del2 = g_cal(2) - g_cal(1);
                                hor_cal(i) = hcal(1)-(hcal(2)-hcal(1))*delta/del2;
                        end
                        if (i == g_cal(1))  % first cal run
                                hor_cal(i) = hcal(1);
                        end
                        if ((g_cal(j)<i) & (i<g_cal(j+1))) % interpolate between
                                delta = i - g_cal(j);
                                del2 = g_cal(j+1) - g_cal(j);
                                hor_cal(i) = hcal(j)+(hcal(j+1)-hcal(j))*delta/del2;
                        end
                        if (i == g_cal(j))
                                hor_cal(i) = hcal(j);
                        end
                        if (i == g_cal(j+1))
                                hor_cal(i) = hcal(j+1);
                        end
                        if (i> g_cal(j+1))
                                delta = i - g_cal(j+1);
                                del2 = g_cal(j+1) - g_cal(j);
                                hor_cal(i) = hcal(j+1)+(hcal(j+1)-hcal(j))*delta/del2;
                        end
                end
        end
end
```

```
% CODES
%
% written by T. Liefeld, 21/4/93
% creates a matrix, called codes,  containing all the run codes
% of non-calibration runs

j = 0;
for i = 1:number
        if (run(i) == 1 )

                stln = length([data_path,sub_code])+2;
                if (i < 10)
                        n = [num2str(0),num2str(i)];
                else
                        n = num2str(i);
                end
                codes(i-j,1:stln) = [data_path,sub_code,n];
        elseif (run(i) == 0)
                j = j+1;
        end
end
```

```
% Collect
% written by Christopher Pouliot on 8/4/93
% to collect all relevent parmeters for a subject and write them
% to an external file so that they can be exported to Excel

ret = '\n';

data_path = input('Enter data path: ','s');
subj_code = input('Enter subject code : ', 's');
cal_file = [subj_code '_cal'];
path = ['TR1:', subj_code, '_all_parms'];

runs = ['02';'03';'04';'05'];

L = length(runs);
for i = 1:L,
        run_code = [subj_code, runs(i,:)];
        parse;

        eval (['load ',data_path,cal_file])
        clear dim g_cal hcal n_cals number run sub_code
        if run_num == '02',
                index = 2;
        elseif run_num == '03',
                index = 3;
        elseif run_num == '04',
                index = 4;
        elseif run_num == '05',
                index = 5;
        elseif run_num == '06',
                index = 6;
        elseif run_num == '07',
                index = 7;
        elseif run_num == '08',
                index = 8;
        elseif run_num == '09',
                index = 9;
        elseif run_num == '10',
                index = 10;
        elseif run_num == '11',
                index = 11;
        end

        cal = num2str(hor_cal(index));
        clear index hor_cal

        eval (['load ',data_path,run_code, '.parms'])
        clear tau1 tau2 gain1 gain2

        delay1 = delay*4;
        spinl1 = spinl*4;

        eval (['load ',data_path, run_code, '.dec_good'])
        perc_good1 = 100*mean(dec_good(delay1:(delay1+100)));
        perc_good2 = 100*mean(dec_good((delay1+spinl1):(delay1+spinl1+100)));
        perc_good1 = num2str(perc_good1);
        perc_good2 = num2str(perc_good2);
        clear dec_good
        eval (['load ',data_path, run_code, '.eperfit'])
        K1 = num2str(model_parms(1));
        T1 = num2str(model_parms(2));
        MSE1 = num2str(options(8));
        clear model_parms options

        eval (['load ',data_path, run_code, '.epostfit'])
        K2 = num2str(model_parms(1));
        T2 = num2str(model_parms(2));
        MSE2 = num2str(options(8));
        clear model_parms options run_code

        delay = num2str(delay);
        runlen = num2str(runlen);
        spinl = num2str(spinl);
        spinv = num2str(spinv);

        all_parms = [subject '    ' session '        ' run_num '       ' test_type '      ' direction '
        '                    rtype ' ' delay '          ' runlen '         ' spinl ' ' spinv '           ' cal '
        ' perc_good1 '  ' K1 '   ''       ' T1 '   ''       ' MSE1 '        ' perc_good2 '  ' K2 '
        ''      ' T2 '    ''      ' MSE2 '        ' ret];

        fprintf (path, all_parms);

        clear subject session run_num test_type direction rtype delay
        clear runlen spinl spinv cal perc_good K1 MSE1 K2 MSE2
end
```

```
% Combine
%
% Program which averages the mean BDC responses.
% by Chris Pouliot

clear

disk = 'SLS_HD:T:';
sub = ['T2';'T3';'T4';'T5'];
cond = '.dmp';

data_path1 = [disk,sub(1,:),cond];
eval(['load ', data_path1])
mean_spv1 = mean_spv;
total1 = total;
var_spv1 = var_spv;
clear mean_spv total var_spv

if ~(length(sub(:,2)) == 1),
        for i = 2:length(sub(:,2)),
                data_path2 = [disk,sub(i,:),cond];
                eval(['load ', data_path2])
                mean_spv2 = mean_spv;
                total2 = total;
                var_spv2 = var_spv;
                clear mean_spv total var_spv
                the_total = total1 + total2;
                mean_spv1 = (total1 .* mean_spv1 + total2 .* mean_spv2) ./ the_total;
                mean_spv1(pack_true(isnan(mean_spv1))) = zeros(1:sum(isnan(mean_spv1)));
                mean_spv1(pack_true(~finite(mean_spv1))) =
zeros(1:sum(~finite(mean_spv1)));
                var_spv1 = (((total1-1) .* var_spv1)+((total2-1) .* var_spv2)) ./ (the_total-2);
                var_spv1(pack_true(isnan(var_spv1))) = zeros(1:sum(isnan(var_spv1)));
                var_spv1(pack_true(~finite(var_spv1))) = zeros(1:sum(~finite(var_spv1)));
                total1 = the_total;
        end
end

mean_spv = mean_spv1;
var_spv=var_spv1;
total = total1;
eval(['save SLS_HD:',sub(1,1),'results',cond,' mean_spv var_spv total'])
graph
```

```
% Declow
%
% Function which decimates the 4 Hz data down to 2/3 Hz
% by Chris Pouliot

function [tmean, tvar, tnum] = declow(sub,cond,period)

n = 6;
eval(['chdir TR1:'])
eval(['load ',sub,'results.',cond])

mean_var = mean(var_spv);
std_var = std(var_spv);
mm2s = mean_var-2*std_var;
mm1s = mean_var-1*std_var;
mp1s = mean_var+1*std_var;
mp2s = mean_var+2*std_var;

for i = 1:length(var_spv),
        bool2m = (var_spv(i) < mm2s);
        bool1m = (var_spv(i) < mm1s);
        bool1p = (var_spv(i) < mp1s);
        bool2p = (var_spv(i) < mp2s);

        if bool2m,
                nvar(i) = mm2s;
        elseif (bool1m & (~bool2m)),
                nvar(i) = mm1s;
        elseif (bool1p & (~bool1m)),
                nvar(i) = mean_var;
        elseif (bool2p & (~bool1p)),
                nvar(i) = mp1s;
        elseif (~bool2p),
                nvar(i) = mp2s;
        end
end
weight = total ./ nvar;

new_length = length(mean_spv)/n;
for i = 1:new_length,
        start = 1 + (i-1)*n;
        finish = n*i;
        x = [start:finish];
        y = mean_spv(start:finish);
        sqweight = weight(start:finish);
        smweight = sum(sqweight);
        xbar = sum(sqweight .* x) / smweight;
        ybar = sum(sqweight .* y) / smweight;
        ssqx2 = sum(sqweight .* (x - xbar).^2);
        ssqy2 = sum(sqweight .* (y - ybar).^2);
        ssqxy = sum(sqweight .* (x - xbar) .* (y - ybar));
        r2 = ssqxy^2/(ssqx2*ssqy2);
        s2 = (1-r2)*ssqy2/(smweight);
        b = ssqxy/ssqx2;
        a = ybar - b*xbar;
        tnum(i) = sum(total(start:finish))/n;
        tvar(i) = s2;
        tmean(i) = ybar;
```

```
end
tmean = tmean';
name = [sub cond period];
eval(['save SLS_HD:T:',name,' tmean /tabs /ascii'])
tmean = tmean';
```

```
% Dec_mean
%
% Program which averages runs together to net a mean repsponse
% by Chris Pouliot

clear

disk = 'TR1';
sub = 'A1';
data_path = [disk,':',sub,':'];

stat_code = ['TR1:', sub, '.he'];
runs = ['02';'03'];
[mean_spv, var_spv, total] = stat(sub,data_path,runs,241,480);
graph
eval(['save ',stat_code,' mean_spv var_spv total runs']);
clear mean_spv var_spv total runs stat_code

stat_code = ['TR1:', sub, '.dmp'];
runs = ['04';'05'];
[mean_spv, var_spv, total] = stat(sub,data_path,runs,241,480);
graph
eval(['save ',stat_code,' mean_spv var_spv total runs']);
clear mean_spv var_spv total runs stat_code
```

```
% Dec_mean_report
%
% Program which produces a run report for the mean responses
% by Chris Pouliot

clear

data_path = input('Enter Data Path : ','s');
stat_code = input('Enter Stats Code: ', 's');

eval(['load ',data_path, stat_code,'.stats'])
eval(['load ',data_path,stat_code,'.perexpfit'])

K(1) = model_parms(1);
T(1) = model_parms(2);
clear model_parms

eval(['load ',data_path,stat_code,'.postexpfit'])

K(2) = model_parms(1);
T(2) = model_parms(2);
clear model_parms

sample = 4;
minute_size = 60 * sample;
t = [1:(2*minute_size)] / sample;

hold off
clg

subplot (211);

text (.05,.90,'SLS-2 EO72 ROTATING CHAIR RUN SUMMARY', 'sc');

subplot (212);
axis([0 120 -150 150]);
xlabel('Time since chair start (sec)');
ylabel('Slow Phase Velocity (deg/sec)');
plot(t,mean_spv,'.');
hold on

x=-120*K(1)*exp(-1*t/T(1));
plot(t(1:240),x(1:240));

y=120*K(2)*exp(-1*t/T(2));
plot(t(241:480),y(1:240));

prtsc
hold off
subplot (111)
```

```
% Dec_report
% Program which produces run reports for the individual runs
% By Christopher Pouliot

data_path = input('Enter Data Path: ','s');
run_code = input('Enter Run Code: ', 's');
parse;

eval(['load ',data_path, run_code,'.dec_spv']);
eval(['load ',data_path,run_code,'.dec_good']);
eval(['load ',data_path,run_code,'.parms']);

eval(['load ',data_path,run_code,'.eperfit']);
K(1) = model_parms(1);
T(1) = model_parms(2);
E(1) = options(8);
clear model_parms options

eval(['load ',data_path,run_code,'.epostfit']);
K(2) = model_parms(1);
T(2) = model_parms(2);
E(2) = options(8);
clear model_parms options

percent1 = 100*mean(dec_good(1:100));
percent2 = 100*mean(dec_good(241:340));

A(1) = 120;
A(2) = -120;
if sum(dec_spv(1:240)) < 0,
          A(1) = -120;
          A(2) = 120;
end

sample = 4;
minute_size = 60 * sample;
t = [1:(2*minute_size)] / sample;

hold off
clg

subplot (211);

text (.05,.98,'SLS-2 EO72 ROTATING CHAIR RUN SUMMARY', 'sc');

text (.1,.93,'subject', 'sc');
text (.5,.93, subject, 'sc');

text (.1,.90,'session', 'sc');
text (.5,.90, session, 'sc');

text (.1,.87,'run', 'sc');
text (.5,.87,run_num, 'sc');

text (.1,.84,'test type', 'sc');
text (.5,.84,test_type, 'sc');

text (.1,.79,'spin length', 'sc');
text (.5,.79, num2str(spinl), 'sc');

text (.1,.76,'run length', 'sc');
text (.5,.76, num2str(runlen), 'sc');

text (.1,.73,'spin velocity', 'sc');
text (.5,.73, num2str(spinv), 'sc');

text (.1,.68,'per-rotatory gain', 'sc');
text (.5,.68, num2str(K(1)), 'sc');
if K(1) < .1,
            text (.8,.68, '***', 'sc');
end

text (.1,.65,'per-rotatory time constant', 'sc');
text (.5,.65, num2str(T(1)), 'sc');
if T(1) < 1,
                              text(.8,.65, '***', 'sc');
end

text (.1,.62,'per-rotatory percent good data', 'sc');
text (.5,.62, num2str(percent1), 'sc');

text (.1,.59,'per-rotatory MSE', 'sc');
text (.5,.59, num2str(E(1)), 'sc');

text (.1,.54,'post-rotatory gain', 'sc');
text (.5,.54, num2str(K(2)), 'sc');
if K(2) < .1,
            text (.8,.54, '***', 'sc');
end

text (.1,.51,'post-rotatory time constant', 'sc');
text (.5,.51, num2str(T(2)), 'sc');
if T(2) < 1,
          text(.8,.51, '***', 'sc');
end

text (.1,.48,'post-rotatory percent good data', 'sc');
text (.5,.48, num2str(percent2), 'sc');

text (.1,.45,'post-rotatory MSE', 'sc');
text (.5,.45,num2str(E(2)), 'sc');

subplot (212);
axis([0 120 -150 150]);
xlabel('Time since chair start (sec)');
ylabel('Slow Phase Velocity (deg/sec)');
plot(t(dec_good),dec_spv(dec_good),'.');
hold on

x=A(1)*K(1)*exp(-1*t/T(1));
plot(t(1:240),x(1:240));

y=A(2)*K(2)*exp(-1*t/T(2));
plot(t(241:480),y(1:240));
```

```
% Delta_tach
%
% Function which looks for large changes in the tach signal
% by D. Balkwill

function n = delta_tach(tach,init)

x = abs(tach);

false  = 0;
true = 1;

final = init + 100;

themean = mean(x(init:final));
thestd = 4*std(x(init:final));

thresholdup = themean + thestd;
thresholddown = themean - thestd;

i = init;
condition = false;
while (condition == false)
        if i == (length(x) - 5)
                condition = true;
        end
        if (x(i) >= thresholdup)
                if ((x(i+1) >= thresholdup) & (x(i+2) >= thresholdup))
                        condition = true;
                end
        elseif (x(i) <= thresholddown)
                if ((x(i+1) <= thresholddown) & (x(i+2) <= thresholddown))
                        condition = true;
                end
        end
        i = i + 1;
end

n = i - 2;
if n == (length(x) - 6)
        n = NaN;
end
```

```
% Dec30
%
% Program which decimates the data from 120 Hz down to 4 Hz.
% By T. Liefield

% set bad data to zero for summation purposes
norm_spv = norm_spv .* good_data;

l = length(good_data);

new_l = (l - 1)/30;          %new sampling frequency

y = zeros(30,new_l);
g = zeros(30,new_l);
n = zeros(1,new_l);
d = zeros(1:new_l);
x = zeros(1,new_l);
z = 1:1:30;

for t=1:(new_l-1)
        for i = 1:30
                y(i,t) = norm_spv(30*t+i);
                g(i,t) = good_data(30*t+i);
        end;
        x = sum(y);
        n = sum(g);
        [a] = polyfit(z,y(:,t)',1);  %linear least squares fit to each bin
        for i = 1:30                              % calc variance about the linear fit
                d(t) = d(t) +((y(i,t) - (a(2)+a(1)*i)).*g(i,t)).^2;
        end;
end;

dec_good = (n>0);        % good data flag

% Decimated SPV is box-car average across row, with n=number of
% good samples.  Correction in denominator to prevent division
% by zero for an entire bin of bad data;  dec_spv=0 in this case.
dec_spv = x ./ (n + (~dec_good));

% Variance within trace is variance of each bin around a linear
% polynomial fit to each bin. A correction in case of
% good samples in bin is <= 1;  variance within=0 in this case.
within = d ./ (n -1 + 2*(n<=1) );


% remove outliers in the variance, replace with means of each
% of three equal sized regions
[y,i] = sort(within);
mean1 = mean(y(1:160));
mean2 = mean(y(161:320));
mean3 = mean(y(321:480));
var(i(1:160)) = mean1*ones(1,160);
var(i(161:320)) = mean2*ones(1,160);
var(i(321:480)) = mean3*ones(1,160);
within = var;

% calculate weights as the number of samples divided by the variance of each
% sample.
```

```
dec_weight = n ./ within;
bad_weight = pack_true(isnan(dec_weight));
dec_weight(bad_weight) = zeros(1,max(size(bad_weight)));
zero_var = pack_true((within<=1e-7));
dec_weight(zero_var) = zeros(1,max(size(zero_var)));

%save data, having departed from 'file_specs' by now
eval(['save ',run_code,'.dec_spv dec_spv']);
eval(['save ',run_code,'.dec_weight dec_weight']);
eval(['save ',run_code,'.within within']);
eval(['save ',run_code,'.dec_good dec_good']);
%clear dec_good dec_spv i l     new_l  within  x out_weight zero_var dec_weight
```

```
% Graph
%
% Program to graph the SPV data
% by Chris Pouliot

clg
hold off
axis([0 240 -50 130])
plot(mean_spv);
xlabel('Sample Number');
ylabel('SPV (deg/sec)');
hold on
std_spv = sqrt(var_spv);
plot(mean_spv-std_spv)
plot(mean_spv+std_spv)
hold off
```

```
% Ind_model_fit_exp
% Program to fit a first order exponential model fit to the SPV data
% by T. Liefield


% load data
%

eval(['chdir ',data_path]);
eval(['load ',run_code,'.dec_spv']);
eval(['load ',run_code,'.dec_good']);
eval(['load ',run_code,'.parms']);

save_good = dec_good;
good_indices = pack_true(dec_good);
if (spinv < 0)
        dec_spv = -dec_spv;
end

%
% Initialize time vector, assuming 4 Hz decimated frequency
%
l = length(dec_spv);
t = ([1:l] - 0.5) / 4;
t = t';

%
% shape tach signal with exponential (0.17 sec time constant)
% ramp to a steady state level at 'spinv'
%
Tv = 0.17;
if (rem(l,2) == 1)
        u = [ones(1,((l+1)/2)) zeros(1,((l-1)/2))];
else
        u = [ones(1,l/2) zeros(1,l/2)];
end
u = u';

%
% overall control input (tach)
%
u = lsim( spinv/Tv, [1, 1/Tv], u, t);

%
% Nominal model parameters.  The parameters to be fitted are the
% non-dimensional ratios of the physical parameters to the
% nominal model parameters here.  This places equal emphasis
% on each model parameter, even though they may be orders of
% magnitude apart.

K = .6;  % gain constant
T = 15;  %  time constant
A = -120;         % alpha_m  amplitude of step input         -- fixed
norm_parms = [K ; T ;  A];

options = [0 ; 0.001 ; 0.001];     %error tolerances -- see "help foptions"
vlb = [.1; .01; 1];  %lower bounds
vub = [10; 10; 1]; %upper bounds
```

```
plot(t(good_indices),dec_spv(good_indices))

%
% Fit the per-rotatory portion first
%

fprintf(['\n\n\nFitting ',run_code,' per-rotatory\n']);
dec_good = save_good;
dec_good(1:12) = zeros(1,12);          % do not fit first 3 seconds of data
dec_good(241:480) = zeros(1,240);      % do not fit post-rotatory data

if (sum(dec_good) < 10)
        fprintf('Not enough data points to determine a curve fit.\n');
        return;
end

good_indices = pack_true(dec_good);

model_parms = [1; 1; 1];
[model_parms, options] = constr('model_err_exp', model_parms, options, vlb, vub, [], t, u,
dec_spv, good_indices, norm_parms);

model_parms = model_parms .* norm_parms;
eval(['save ',run_code,'.eperfit model_parms options'])

fprintf('*** Model fit: initial model parameters = 1.0\n');
fprintf('Number of iterations = %5.0f\n',options(10));
fprintf('Mean square error = %7.4f\n',options(8));

fprintf('K = %f\n',model_parms(1));
fprintf('T = %f\n',model_parms(2));
fprintf('A = %f\n',model_parms(3));

%
% Fit the post-rotatory portion now
%

fprintf(['\n\n\nFitting ',run_code,' post-rotatory\n']);

dec_good = save_good(241:480);
dec_good(1:12) = zeros(1,12); % do not fit first 3 seconds of data

if (sum(dec_good) < 10)
        fprintf('Not enough data points to determine a curve fit.\n');
        return;
end
t=t(1:240);
dec_spv_p = dec_spv(241:480);
good_indices = pack_true(dec_good);
norm_parms = [K ; T ; -1*A];
model_parms = [1; 1; 1];
[model_parms, options] = constr('model_err_exp', model_parms, options, vlb, vub, [], t, u,
dec_spv_p, good_indices,norm_parms);

model_parms = model_parms .* norm_parms;
eval(['save ',run_code,'.epostfit model_parms options'])
```

```
fprintf('*** Second fit: initial model parameters = 1.0\n');
fprintf('Number of iterations = %5.0f\n',options(10));
fprintf('Mean square error = %7.4f\n',options(8));

fprintf('K = %f\n',model_parms(1));
fprintf('T = %f\n',model_parms(2));
fprintf('A = %f\n',model_parms(3));
```

```
% Lin_fit
%
% D. Balkwill 5/21/91
% Program which determines a linear curve fit, y = mt + b, to a segment, x

function [m,b] = lin_fit(t,x)

[n1,n2] = size(t);
[n3,n4] = size(x);

if (n1 ~= n3)
        t = t';
        [n1,n2] = size(t);
end
l1 = n1 * n2;
l2 = n3 * n4;
if (l1 >= l2)
        l = l2;                 % sum{ 1 }
        t = t(1:l);
else
        l = l1;                 % sum{ 1 }
        x = x(1:l);
end

A = sum(t .* t);   % sum{ t^2 }
B = sum(t);                     % sum{ t }
C = sum(x);                     % sum{ x }
D = sum(x .* t);   % sum{ xt }

m = (D * l - B * C) / (A * l - B * B);
b = (A * C - D * B) / (A * l - B * B);

return;
```

```
% Log_outlier

function [diff,underflow,overflow,m,b] = log_outlier(t,x)

% Function to perform outlier detection based upon the natural
% logarithm of the SPV data.  The data is log-transformed, an
% optimal linear fit is calculated, and the RMS of the data
% about this line calculated.  Any difference exceeding 6 RMS
% values is flagged as bad data (outlying artifact).  Any
% difference exceeding 3 RMS values, at a point where the
% data point is less than 2 log units (7.4 deg/s^2) is flagged
% as bad data (dropout).  This only works on data which is
% greater than 8 deg/sec^2.  Any point within half a second of
% a bad data point is marked as bad, to account for transient
% behaviour.
%
% A new linear fit is calculated for the good data, new RMS
% calculated, and bad data flagging is repeated.  This process
% reiterates until RMS converges to within 20% on subsequent
% iterations.
%
% The log-transformed data is displayed, along with each linear
% fit, and the 3 RMS error bar lines.
%
% D. Balkwill 8/8/91

sample = 100;
filt_length = 1 + (sample/2);

[m1,m2] = size(x);
if (m1 > 1)
        x = x';
end
l = max(m1,m2);             %get number of samples

s = sign(mean(x));          % predominant direction of SPV
bool = ((s * x) > 0);       % 1 iff SPV is in predominant direction
filler = 1e-10 * ones(1,l);

% x is log(SPV) if x in predominant direction, or 1e-10 otherwise
x = log( (s * bool .* x) + ((~bool) .* filler) );

%do a first pass on the outlier detection
pass = 1;
[m,b] = lin_fit(t,x);
y = m * t + b;
d = abs(y - x);
RMS = sqrt(mean(d .* d));
fprintf('Pass %2.0f:  slope = %7.4f,  ',pass,m);
fprintf('intercept = %6.3f,  RMS = %6.3f\n',b,RMS);

hold off
clg
axis([t(1) t(l) -5 5]);
plot(t,x)
hold on
plot(t,y,'g')
plot(t,y+(3*RMS),'b')
plot(t,y-(3*RMS),'b')

bool = ((d > min(10,(3 * RMS))) & (x < 2)) | (d > (6 * RMS));
diff = filtfilt(ones(1,filt_length),1,bool);
diff = (diff > 0);   % diff is 1 iff there is an outlier

oldRMS = 1e50;           %dummy assignment
while (abs((oldRMS - RMS)/RMS) > 0.2)

        oldRMS = RMS;
        pass = pass + 1;

        good = pack_true(~diff);

        % next curve fit on good data only

        [m,b] = lin_fit(t(good),x(good));

        clear y
        y = m * t + b;
        RMS = sqrt(mean(d(good) .* d(good)));
        fprintf('Pass %2.0f:  slope = %7.4f,  ',pass,m);
        fprintf('intercept = %6.3f,  RMS = %6.3f\n',b,RMS);

        plot(t,y,'g')
        plot(t,y+(3*RMS),'b')
        plot(t,y-(3*RMS),'b')

        % remark bad data points based on new values
        % Note: all points, even previously good ones, are checked
        clear d bool diff
        d = abs(y - x);
        bool = ((d > min(10,(3 * RMS))) & (x < 2)) | (d > (6 * RMS));
        diff = filtfilt(ones(1,filt_length),1,bool);
        diff = (diff > 0);   % diff is 1 iff there is an outlier

end
hold off

% look for outlier in previous half second of data
for i = 1:filt_length
        if (bool(i) > 0)
                break;
        end
end
% number of extra outlying data points before this section of data
underflow = filt_length - i;

% look for outlier in last half second of data
for i = 1:filt_length
        if (bool(l-i+1) > 0)
                break;
        end
end
% number of extra outlying data points after this section of data
overflow = filt_length - i;

return;
```

```
% Mag_outlier

function [diff,underflow,overflow] = mag_outlier(t,x,thresh)

% Function to perform outlier detection based upon a specified
% magnitude threshold.  Any data point exceeding this threshold
% is marked as bad data (artifact).  Any point within half a
% second of a bad data point is marked as bad, to account for
% transient behaviour.
%
% D. Balkwill 8/8/91

sample = round( 1 / (t(2) - t(1)) );
filt_length = 1 + (sample/2);

[m1,m2] = size(x);
if (m1 > 1)
        x = x';
end
l = max(m1,m2);

x = abs(x);
bool = (x > thresh);
diff = filtfilt(ones(1,filt_length),1,bool);
diff = (diff > 0);

for i = 1:filt_length
        if (bool(i) > 0)
                break;
        end
end
underflow = filt_length - i;

for i = 1:filt_length
        if (bool(l-i+1) > 0)
                break;
        end
end
overflow = filt_length - i;

return;
```

```
% Mean_sep_exp_fit
% Program to fit a first order exponential model to the mean data.
% by T. Liefield

data_path = input('Enter data_path: ','s');
stat_code = input('Enter stats code: ','s');

%load data
eval(['chdir ',data_path]);
eval(['load ',stat_code]);

% time vector
l = length(mean_spv);
t = ([1:l] - 0.5) / 4;
t = t';

% shaped tach signal, with steady state level at 'spinv'
Tv = 0.17;
u = [ones(1,480/2) zeros(1,l/2)];
u = u';
spinv = 120;
u = lsim( spinv/Tv, [1, 1/Tv], u, t);

%initialize model parameters
K = 0.4;
T = 8;
A = 120;
norm_parms = [K;T;A];
model_parms = [1;1;1];

options = [0 ; 0.001 ; 0.001];     %error tolerances -- see "help foptions"
vlb = [.1;.01;1];  %lower bounds
vub = [10;10;1]; %upper bounds

dec_good = (total > 0);
dec_good(1:12) = zeros(1,12);
good_indices = pack_true(dec_good);

plot(t(good_indices),mean_spv(good_indices))
[model_parms, options] = constr('model_err_exp', model_parms, options, vlb, vub, [], t, u,
mean_spv, good_indices, norm_parms);

model_parms = model_parms .* norm_parms;
eval(['save ',stat_code,'.expfit model_parms options'])

fprintf(['\nStats code = ',stat_code,'\n']);
fprintf('Exponential fit:\n');
fprintf('Number of iterations = %5.0f\n',options(10));
fprintf('K = %f\n',model_parms(1));
fprintf('Tau = %f\n',model_parms(2));
fprintf('Mean square error = %7.4f\n',options(8));

clear dec_good good indices
xlabel('Time (seconds)')
ylabel('Magnitude of SPV (deg/sec)')
```

```
% Model_err_exp
%
% Error function for model fitting.  Constrained optimization
% minimizes the output of this function, which is currently set
% as the mean square error between the SPV data and the model
% SPV data
% by T. Liefield

function [f,g] = model_err_exp(model_parms,t,u,dec_spv,good_indices, norm_parms)

model_parms = model_parms .* norm_parms;
K = model_parms(1);
T = model_parms(2);
A = model_parms(3);

y=A*K*exp(-1*t/T);

[m2,n2] = size(dec_spv);
[m1,n1] = size(y);
if (m1 > n1)
        if (m2 < n2)
                y = y';
        end
else
        if (m2 > n2)
                y = y';
        end
end

d = y(good_indices) - dec_spv(good_indices);
f = sum(d.*d)/length(d);

plot(t(good_indices),dec_spv(good_indices))
hold on
plot(t(good_indices),y(good_indices), 'g');
hold off

g = -1;

return;
```

```
% Multiple_AATM
%
% written by T. Liefeld, 21/4/93
% prepares a batch file for AATM processing using the
% appropriate calibration factors and names, and a
% predefined batch file name, batch_factors, for use
% with mexAATM4.

% create file of ones for the vertical calibration if no vertical
% calibrations were performed

ver_cal = ones(1,number);

% remove earlier files
eval(['chdir ',hard_disk,':users:Chris:Matlab/Chair:'])
delete batch_factors

for i = 1:number
        if ((i < 10) & (run(i) > 0)),
                n = num2str(i);
                run_code = [data_path,sub_code,'0',n];
                batch_save(run_code,hor_cal(i),ver_cal(i));
        else
                if (i >=10 & (run(i) > 0)),
                        n = num2str(i);
                        run_code = [data_path,sub_code,n];
                        batch_save(run_code,hor_cal(i),ver_cal(i));
                end
        end
end
% send the EOF marker that mexAATM4 looks for
batch_save('&',0,0);

mexAATM4;
```

```
% New_tsq
%
% This routine performs a Monte Carlo simulation to determine
% the sum-of-t-squares distribution.  The simulation is done for
% a grid of values for N1 ranging from 2 to 20 in increments of
% 2, and integer N2 ranging from 1 to 10.  The number of degrees
% of freedom is 100, and it is repeated for 5000 iterations.
% Note: The time required on a Mac II for this simulation is
% approximately 8 seconds per iteration.
%
% C. Pouliot and D. Balkwill 12/6/91

N1 = 10;
N2 = 10;

n1 = 2 * [1:N1]';
n2 = [1:N2];
Mn1 = n1 * ones(1,N2);
Mn2 = ones(N1,1) * n2;

for i=1:N1
        for j=1:N2
                eval(['dist',int2str(n1(i)),'_',int2str(n2(j)),' = zeros(1000,1);']);
        end
end
rand('normal');

eps1 = zeros(N1,1);
eps1sq = zeros(N1,1);
ssq1 = zeros(N1,1);
eps2 = zeros(1,N2);
eps2sq = zeros(1,N2);
ssq2 = zeros(1,N2);

for total = 1:5000

        time = fix(clock);
        fprintf('%2.0f:',time(4));
        fprintf('%2.0f:',time(5));
        fprintf('%2.0f     ',time(6));
        fprintf('total = %5.0f\n',total);

        tsq_sum = zeros(N1,N2);

        for number = 1:40

                r1 = rand(2 * N1);
                for i=1:N1
                        rand1 = r1(2*i,(1:(2*i)));
                        rand1sq = rand1 .* rand1;
                        eps1(i) = sum(rand1);
                        eps1sq(i) = sum(rand1sq);
                end

                r2 = rand(N2);
                for j=1:N2
                        rand2 = r2(1:j,j);
                        rand2sq = rand2 .* rand2;
                        eps2(j) = sum(rand2);
                        eps2sq(j) = sum(rand2sq);
                end

                mn1 = eps1 ./ n1;
                mn2 = eps2 ./ n2;
                ssq1 = eps1sq - (eps1 .* mn1);
                ssq2 = eps2sq - (eps2 .* mn2);

                Mmn1 = mn1 * ones(1,N2);
                Mssq1 = ssq1 * ones(1,N2);

                Mmn2 = ones(N1,1) * mn2;
                Mssq2 = ones(N1,1) * ssq2;

                vardif = ((Mssq1 + Mssq2) ./ (Mn1 + Mn2 - 2)) .*
((ones(N1,N2)./Mn1)+(ones(N1,N2)./Mn2));
                tsq = (((Mmn1) - (Mmn2)) .^ 2) ./ vardif;

                tsq_sum = tsq_sum + tsq;
        end

        val = round(tsq_sum);
        bool = (val > 1000);
        val = (val .* (~bool)) + (bool * 1000);

        for i=1:N1
                for j=1:N2
                        eval(['dist',int2str(n1(i)),'_',int2str(n2(j)),'(val(i,j)) =
dist',int2str(n1(i)),'_',int2str(n2(j)),'(val(i,j)) + 1;']);
                end
        end

end

eval(['save tsq_dist']);
```

```
% Pack_true

function indices = pack_true(bool)

% returns vector of indices corresponding to samples which are
% true (1), given a boolean time series, bool.
%
% D. Balkwill 8/8/91

l = length(bool);

i = searchb(bool,0,1);    %find start of first false section
if (i == 1)
        indices = [];
elseif (i == NaN)
        indices = [1:l];
else
        indices = [1:i-1];
end

while (i ~= NaN)
        i = searchb(bool,1,i);    %look for beginning of true section
        if (i ~= NaN)
                beg = i;
                i = searchb(bool,0,i);  %look for beginning of false section
                if (i ~= NaN)
                        indices = [indices, (beg:i-1)];     %add true section
                else
                        indices = [indices, (beg:l)];        %add true section
                end
        end
end

return;
```

```
% Parse
% written on 7/7/93 by Chris Pouliot
% to parse the run code used by the rotating chair experiment performed
% on SLS-2

if run_code(2) == 'P'                          % is it taken in parabolic flight ??
        if length(run_code) == 5,              % session < 10
                subject = run_code(1);
                session(1) = '0';
                session(2) = run_code(3);
                run_num = run_code(4:5);
        else                                   % session > 10
                subject = run_code(1);
                session = run_code(3:4);
                run_num = run_code(5:6);
        end
        test_type = 'parabolic';
elseif run_code(2) == 'F'                      % is it taken in-flight ??
        if length(run_code) == 5,              % session < 10
                subject = run_code(1);
                session(1) = '0';
                session(2) = run_code(3);
                run_num = run_code(4:5);
        else                                   % session > 10
                subject = run_code(1);
                session = run_code(3:4);
                run_num = run_code(5:6);
        end
        test_type = 'flight';
else                                           % if not parabolic and flight then
        if length(run_code) == 4,              % it must be a BDC, session < 10
                subject = run_code(1);
                session(1) = '0';
                session(2) = run_code(2);
                run_num = run_code(3:4);
        else                                   % session > 10
                subject = run_code(1);
                session = run_code(2:3);
                run_num = run_code(4:5);
        end
        test_type = 'BDC';
end

if ((run_num == '03') | (run_num == '05') | (run_num == '07') | (run_num == '9') | (run_num == '11')),
        direction = 'CW';
else
        direction = 'CCW';
end

if ((run_num == '03') | (run_num == '04') | (run_num == '07') | (run_num == '08')),
        rtype = 'HE';
else
        rtype = 'DMP';
end
```

```
% Pick_regions
%

function regions = pick_regions(t,pos,colour)

% This is the main algorithm for the manual picking of
% calibration regions
%       t          = time coordinate, equally spaced at sampling period
%       pos      = eye position vector
%       colour = flag for colour monitor
%
% A region is selected by picking its beginning and end with
% the mouse.  A selected region is highlighted by picking within
% that region.  A highlighted region is un-highlighted by
% picking again withing that region.  A selected region is
% de-selected by highlighting it, and then pressing the delete
% key.
%
% The user has complete control over pan and zoom features, as
% well as selection and de-selection of regions.  The plotting
% makes use of different colours and line types, as available.
%
% D. Balkwill  11/27/90

l = length(pos);
sample = round(1/(t(2) - t(1)));   % assumes t is periodic

key = 0;
FINISHED = 27;                     % escape
PAN_LEFT = 28;                     % left arrow
PAN_RIGHT = 29;                    % right arrow
SCROLL_LEFT = 11;                  % page down
SCROLL_RIGHT = 12;                  % page up
DELETE_1 = 8;                      % backspace
DELETE_2 = 127;                    % delete
ZOOM_IN = 30;                      % up arrow
ZOOM_OUT = 31;                     % down arrow
FAST_ZOOM_IN = 46;                 % decimal
FAST_ZOOM_OUT = 48;                % zero
COMPLETE_PLOT_1 = 97;              % 'a' key
COMPLETE_PLOT_2 = 65;              % 'A' key
                                   % note: 1, 2, and 3 are reserved for mouse button(s)

num_pick = 0;                      % number of points picked
num_regions = 0;                   % number of regions picked
num_highs = 0;                     % number of regions highlighted
os = 1;                            % offset of start of current trace, in samples
w = l - 1;                         % width of trace, in samples
redraw = 1;                        % flag for plotting
mf = 1;                            % magnification factor

while (key ~= FINISHED)

        if (redraw == 1)
                df = floor(w/2000);
                if (df < 1)
                        df = 1;
                end
                tr = t(os:df:os+w);
                pr = pos(os:df:os+w);

% leave some blank space above and below trace for aesthetics
                mx = max(pr);
                if (mx < 0)
                        mx = mx * 0.9;
                else
                        mx = mx * 1.1;
                end
                mn = min(pr);
                if (mn < 0)
                        mn = mn * 1.1;
                else
                        mn = mn * 0.9;
                end

                hold off
                axis([tr(1) tr(length(tr)) mn mx]);
                if (colour == 'y')

                        % plot eye position signal in black, solid
                        plot(tr,pr,'w')

                        % plot picked regions in blue, dash-dotted
                        hold on
                        for i=1:num_regions
                                t3 = (regions(i,1) - 1)/sample;
                                plot([t3,t3],[mn,mx],'b-.')
                                t4 = (regions(i,2) - 1)/sample;
                                plot([t4,t4],[mn,mx],'b-.')
                                plot([t3,t4],[mn,mx],'b-.')
                        end
                        % plot currently picked point in blue, dotted
                        if (num_pick == 1)
                                plot([t1,t1],[mn,mx],'b:')
                        end

                        % plot highlighted regions in green, solid
                        for i=1:num_highs
                                t3 = (highs(i,1) - 1)/sample;
                                plot([t3,t3],[mn,mx],'g')
                                t4 = (highs(i,2) - 1)/sample;
                                plot([t4,t4],[mn,mx],'g')
                                plot([t3,t4],[mn,mx],'g')
                        end

                else

                        % plot eye position signal in solid
                        plot(tr,pr,'-')

                        % plot picked regions in dash-dotted
                        hold on
                        for i=1:num_regions
                                t3 = (regions(i,1) - 1)/sample;
                                plot([t3,t3],[mn,mx],'-.')
                                t4 = (regions(i,2) - 1)/sample;
```

```
            plot([t4,t4],[mn,mx],'-.')
            plot([t3,t4],[mn,mx],'-.')
        end
        % plot currently picked point in dotted
        if (num_pick == 1)
            plot([t1,t1],[mn,mx],':')
        end

        % plot highlighted regions in dashed
        for i=1:num_highs
            t3 = (highs(i,1) - 1)/sample;
            plot([t3,t3],[mn,mx],'--')
            t4 = (highs(i,2) - 1)/sample;
            plot([t4,t4],[mn,mx],'--')
            plot([t3,t4],[mn,mx],'--')
        end
    end

    text(.7,.93,['magnification = ',int2str(round(mf)),' X'],'sc')
    redraw = 0;

end

[x,y,key] = ginput(1);

if (key==ZOOM_IN)          % increase magnification factor

    old=mf;
    mf=min(old*2,max(old,floor(l/100)));
    if mf==old          % maximum magnification of 100X
        redraw=0;
    else
        redraw=1;
        w=floor(l/mf);
    end

elseif (key == FAST_ZOOM_IN)          % fast two-point zoom

    % first point of region to zoom into
    [t3,y,key] = ginput(1);
    if ((key ~= DELETE_1) & (key ~= DELETE_2))

        % bounds check on first point of region
        if (t3 < tr(1))
            t3 = tr(1);
        elseif (t3 > tr(length(tr)))
            t3 = tr(length(tr));
        end
        x3 = 1 + round(t3 * sample);
        t3 = (x3 - 1)/sample;

        % display first point
        hold on
        if (colour == 'y')
            plot([t3,t3],[mn,mx],'r:');
        else
            plot([t3,t3],[mn,mx],':');
        end
        hold off
        redraw = 1;

        % second point of region to zoom into
        [t4,y,key] = ginput(1);

        % allow user to abort zoom via delete key
        if ((key ~= DELETE_1) & (key ~= DELETE_2))

            % bounds check on second point of region
            if (t4 < tr(1))
                t4 = tr(1);
            elseif (t4 > tr(length(tr)))
                t4 = tr(length(tr));
            end
            x4 = 1 + round(t4 * sample);
            t4 = (x4 - 1)/sample;

            % display second point
            hold on
            if (colour == 'y')
                plot([t4,t4],[mn,mx],'r:');
            else
                plot([t4,t4],[mn,mx],':');
            end
            hold off

            % swap order of points if needed
            if (x4 < x3)
                old = x4;
                x4 = x3;
                x3 = old;
            end

            % calculate new magnification parameters
            if (x3 ~= x4)
                os = x3;
                w = x4 - x3;
                mf = l/w;
            end
        end
    end

elseif (key==ZOOM_OUT)  % decrease magnification

    if (mf == 1)        % already completely zoomed out
        redraw = 0;
    else
        redraw=1;
        old=mf;
        mf=max(floor(old/2),1);
        w=floor(l/mf);
        if (w >= l)
            w = l - 1;
        end
        if ((os+w)>l)
            os=floor(max(1,l-w));
        end
```

```
                end

        elseif ((key == COMPLETE_PLOT_1) | (key == COMPLETE_PLOT_2) | (key ==
FAST_ZOOM_OUT))  % display entire plot

                os = 1;
                mf = 1;
                w = l - 1;
                redraw = 1;

        elseif (key==PAN_RIGHT)  % increase offset by quarter-screen

                old=os;
                os=floor(max(1,min(l-w,os+0.25*w)));
                if old==os                      % already panned to end
                        redraw=0;
                else
                        redraw=1;
                end

        elseif (key==PAN_LEFT)          % decrease offset by quarter-screen

                old=os;
                os=floor(max(1,os-0.25*w));
                if os==old                      % already panned to beginning
                        redraw=0;
                else
                        redraw=1;
                end

        elseif (key==SCROLL_RIGHT)   % jump display one screenful right

                old=os;
                os=floor(max(1,min(os+w,l-w)));
                if os==old                      % already panned to end
                        redraw=0;
                else
                        redraw=1;
                end

        elseif (key==SCROLL_LEFT)    % jump display one screenful left

                old=os;
                os=floor(max(1,os-w));
                if old==os                      % already panned to beginning
                        redraw=0;
                else
                        redraw=1;
                end

        elseif ((key==DELETE_1) | (key==DELETE_2))

                if (num_pick > 0)               % wipe out currently picked point
                        num_pick = 0;
                        redraw = 1;
                elseif (num_highs > 0)  % wipe out highlit regions
                        for i=1:num_highs
                                index = InList(highs(i,1),regions);
```

```
            regions = DeleteRow(index,regions);
        end
        num_regions = num_regions - num_highs;
        num_highs = 0;
        clear highs
        redraw = 1;
    end

elseif (key==1) | (key==2) | (key==3)      % up to three-button mouse input

    if (num_pick == 0)          % this is the first picked point

        % bounds check on picked point
        if (x < tr(1))
            x = tr(1);
        elseif (x > tr(length(tr)))
            x = tr(length(tr));
        end

        % convert time value to sample number
        x1 = 1 + round(x * sample);

        % see if point is in a selected region
        index1 = InList(x1,regions);
        if (index1 > 0)
            index2 = InList(x1,highs);
            if (index2 > 0)   % de-highlight region
                num_highs = num_highs - 1;
                highs = DeleteRow(index2,highs);
                redraw = 1;
            else    % highlight region for future deletion
                x1 = regions(index1,1);
                x2 = regions(index1,2);
                t1 = (x1 - 1)/sample;
                t2 = (x2 - 1)/sample;
                num_highs = num_highs + 1;
                highs(num_highs,:) = [x1 x2];
                if (colour == 'y')
                    hold on
                    plot([t1,t1],[mn,mx],'g');
                    plot([t2,t2],[mn,mx],'g');
                    plot([t1,t2],[mn,mx],'g');
                else
                    hold on
                    plot([t1,t1],[mn,mx],'-');
                    plot([t2,t2],[mn,mx],'-');
                    plot([t1,t2],[mn,mx],'-');
                end
            end
        else  % point is not already in a selected region
            % display as first point of region being selected
            t1 = (x1 - 1)/sample;
            num_pick = 1;
            hold on
            if (colour == 'y')
                plot([t1,t1],[mn,mx],'b:');
            else
                plot([t1,t1],[mn,mx],':');
                end
                hold off
            end
        elseif (num_pick == 1)   % second picked point

            % bounds check on picked point
            if (x < tr(1))
                x = tr(1);
            elseif (x > tr(length(tr)))
                x = tr(length(tr));
            end

            % convert time value to sample number
            x2 = 1 + round(x * sample);
            t2 = (x2 - 1)/sample;

            if (x2 == x1)       % cannot have interval of zero width
                num_pick = 0;
                redraw = 1;
            else
                hold on
                if (colour == 'y')
                    plot([t2,t2],[mn,mx],'b:');
                else
                    plot([t2,t2],[mn,mx],':');
                end
                if (x2 < x1)        % order picked points
                    old = x1;
                    x1 = x2;
                    x2 = old;
                    old = t1;
                    t1 = t2;
                    t2 = old;
                end
                num_pick = 0;
                if (colour == 'y')
                    plot([t1,t2],[mn,mx],'b:');
                else
                    plot([t2,t2],[mn,mx],':');
                end

                % add picked region to list
                num_regions = num_regions + 1;
                regions(num_regions,:) = [x1 x2];
                hold off
            end
        end
    end
end

clear i index index1 index2 t1 t2 x1 x2 x y mn mx mnv sample
clear highs num_highs num_regions old num_pick
clear key redraw os w mf l df pr tr
return;
```

```
% Searchb

function index = search(vec,val,start)

% Searches a vector (vec) for a value (val), starting at a given
% index (start), and returns the index at which the value was
% found.  The value NaN is returned if the value was not found.
%
% D. Balkwill 8/8/91

l = length(vec);
cont = 1;
index = start;
while (cont)
        if (vec(index) == val)
                cont = 0;
        elseif (index == l)
                cont = 0;
                index = NaN;
        else
                index = index + 1;
        end
end
return;
```

```
% Stat
%
% Function which calculates the mean and variance of a selected number of
% individual SPV responses
%
% by Chris Pouliot

function [mean_spv, var_spv, total] = stat(sub,data_path,runs,s,e)

sample = 4;
minute_size = 60 * sample;
sum_spv = zeros(1,2*minute_size);
sum_square = zeros(1,2*minute_size);
total = zeros(1,2*minute_size);

if s == 1,
        kind = '.eperfit';
elseif s == 241,
        kind = '.epostfit';
end

for j = 1:length(runs(:,2))
        run_code = [data_path,sub,runs(j,:)];
        eval(['load ',run_code,'.dec_spv'])
        eval(['load ',run_code,'.dec_good'])
        eval(['load ',run_code,'.parms'])
        eval(['load ',run_code,kind])
        perc_good = 100*mean(dec_good(s:(e-140)));
        fprintf('Run ');
        fprintf(runs(j,:));
        fprintf(' percent good    %g\n',perc_good);
        fprintf('               gain                 %g\n',model_parms(1));
        fprintf('               time constant %g\n',model_parms(2));
        fprintf('               MSE                          %g\n',options(8));
        fprintf('               spinv                %g\n\n',spinv);
        sum_spv = sum_spv + ((spinv/abs(spinv)) .* dec_spv .* dec_good);
        sum_square = sum_square + (dec_spv .* dec_spv .*dec_good);
        total = total + dec_good;
        clear dec_spv dec_good spinv
end

mean_spv = sum_spv ./ total;
mean_spv(pack_true(isnan(mean_spv))) = zeros(1:sum(isnan(mean_spv)));
var_spv = sum_square - (sum_spv .* mean_spv);
var_spv = var_spv ./ (total - 1);
var_spv(pack_true(isnan(var_spv))) = zeros(1:sum(isnan(var_spv)));
clear sum_spv sum_square

mean_spv = mean_spv(s:e);
var_spv = var_spv(s:e);
total = total(s:e);
```

```
%stat_prep_batch
%
% Prepares an SPV profile for statistical analysis.  The first
% step is time-shifting and stripping out extra data to leave
% one minute per-rotatory and one minute post-rotatory.  The
% second step is outlier detection.  The third step is decimation
% by a factor of 30 down to 4 Hz.
%
% D. Balkwill 8/8/91
%
% Modified by T Liefeld, 12/17/92    to use a longer per-
% rotatory period consistent with the later use of actual
% tach signals as the stimulus for per-rotatory model
% fitting scripts.
% Modified again for use with batch analysis scripts

sample = 120;
minute_size = 60 * sample;

%load data
eval(['load ',run_code,'.aspvh']);
eval(['load ',run_code,'.parms']);

x = aspvh;
clear aspvh

%
% Normalize SPV profile to one-minute per and post-rotatory
% On the longer per_rotatory file,  perform outlier detection
% only upon the normslized one minute section due to the
% difficulty in dealing with a different exponential on the
% initial rise.  Make all extended per-rot files 65 seconds
% long, truncating extra data or padding as before

y = zeros(1,2 * minute_size + 1);   %initialize to two minutes

delay = delay*sample ;
spinl = spinl*sample;

if (spinl >= minute_size)  %extract first minute of per-rotatory
        y(1:minute_size) = x(delay:(delay+minute_size-1));
else    %pad per-rotatory out to one minute
        y(1:spinl) = x(delay:(delay+spinl-1));
        y(spinl+1:minute_size) = zeros(1,minute_size-spinl);
end


%post-rotatory data

if ((max(size(x))-delay-spinl) >= minute_size)        %extract first minute of post-rotatory
        y(minute_size+1:2*minute_size+1) = x((delay+spinl):(delay+spinl+minute_size));
else                                                  %pad post-rotatory out to one minute
        y((minute_size+1):(minute_size+max(size(x))-delay-spinl)) =
x((delay+spinl+1):(max(size(x))));
        y((minute_size+1+max(size(x))-delay-round(spinl)):2*minute_size+1) =
        zeros(1,minute_size-max(size(x))+1+delay+round(spinl)) * median(x((max(size(x))-
5):(max(size(x))-1)));
end
```

```
%
% Determine sections to be excluded from statistical analysis
% (dropouts and outliers).
%

t = [1:(2*minute_size+1)] / sample;
good_data = ones(1,2*minute_size+1);

%
%find valid range for log outlier detection in per-rotatory section
%
fprintf('Per-rotatory:\n');
i = sample + 1;  %one second after start
j = i + 20 * sample;   %insist upon 20 seconds, minimum
while (abs(mean(y(j:j+5*sample))) > 10)  %look for mean spv under 10 deg/sec
        j = j + 2 * sample;
end
t1 = t(i:j);
y1 = y(i:j);

[logout1,under,over,m,b] = log_outlier(t1,y1);

%take care of under- or over-flow
good_data(i:j) = ~logout1;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
if (over > 0)
        good_data(j+1:j+over) = zeros(1,over);
end

%save final fit in parms file as first-order "model fit"
tau1 = -1/m;
gain1 = exp(b)/120;
fprintf('Time length of outliers from log fit is ');
fprintf('%5.2f seconds.\n',(sum(logout1)+under+over)/sample);

% do magnitude outlier detection on remainder of per-rotatory SPV

i = j + over + 1;
[magout1,under,over] = mag_outlier(t(i:minute_size),y(i:minute_size),30);
good_data(i:minute_size) = ~magout1;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
fprintf('Time length of outliers from magnitude threshold is');
fprintf(' %5.2f seconds.\n',(sum(magout1)+under)/sample);
% don't fill in any overflow, because this would be post-rotatory

delay = delay / sample;
spinl = spinl / sample;

%
%find valid range for outlier detection in post-rotatory section
%
fprintf('Post-rotatory:\n');
```

```
i = minute_size + sample + 1;  %one second after stop
j = i + 20 * sample;    %insist upon 20 seconds, minimum
while (abs(mean(y(j:j+5*sample))) > 10)
        j = j + 2 * sample;
end
t2 = t(i:j);
y2 = y(i:j);

[logout2,under,over,m,b] = log_outlier(t2,y2);

%take care of under- or over-flow
good_data(i:j) = ~logout2;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
if (over > 0)
        good_data(j+1:j+over) = zeros(1,over);
end

%save final fit in parms file as first-order "model fit"
tau2 = -1/m;
gain2 = exp(b + 60 * m)/120;
fprintf('Time length of outliers from log fit is ');
fprintf('%5.2f seconds.\n',(sum(logout2)+under+over)/sample);

% do magnitude outlier detection on remainder of per-rotatory SPV

i = j + over + 1;
[magout2,under,over] = mag_outlier(t(i:2*minute_size),y(i:2*minute_size),30);
good_data(i:2*minute_size) = ~magout2;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
fprintf('Time length of outliers from magnitude threshold');
fprintf(' is %5.2f seconds.\n',(sum(magout2)+under)/sample);
% don't fill in any overflow, because this would be past two minutes

fprintf('Overall percentage of good data is %6.2f\n',100*mean(good_data));


%plot data and outlying regions
hold off
plot(t1,log(abs(y1)))
hold on
plot(t1,log(abs(mean(y1))) * logout1,'b--')
grid
xlabel('Time (sec)')
ylabel('ln(SPV)')
title([run_code,' -- SPV and log outlier indicator'])

hold off
plot(t2,log(abs(y2)))
hold on
plot(t2,log(abs(mean(y2))) * logout2,'b--')
grid
xlabel('Time (sec)')
ylabel('ln(SPV)')
title([run_code,' -- SPV and log outlier indicator'])
```

```
hold off
plot(t,y)
hold on
plot(t,50*(~good_data),'b--')
plot(t,-50*(~good_data),'b--')
grid
xlabel('Time (sec)')
ylabel('SPV (deg/sec)')
title([run_code,' -- SPV and overall outlier indicator'])

hold off
plot(t(good_data),y(good_data),'.')
grid
xlabel('Time (sec)')
ylabel('SPV (deg/sec)')
title([run_code,' -- good SPV (outliers removed)']);

hold off

eval(['save ',run_code,'.parms  delay spinl spinv runlen tau1 gain1 tau2 gain2 T1 T2']);

clear t t1 t2 y1 y2 i j logout1 logout2 minute_size sample
clear delay spinl spinv T1 T2 runlen under over magout1 magout2
clear m b tau1 gain1 tau2 gain2

norm_spv = y;

clear y

% save normalized data, having departed from 'file_specs' at this point

eval(['save ',run_code,'.good good_data']);
eval(['save ',run_code,'.norm norm_spv']);

%decimate data to 4 Hz, and save decimated information
dec30

clear good_data norm_spv
clear t tau1 tau2 run_len spinl spinv
```

```
% Tachan_batch
%
% Program which analyzes the tach signal to determine a set number of parameters
% by D. Balkwill and C. Pouliot 12/91

%load tach data into standardized variable

eval(['load ',run_code,'.tach']);
TACH = tach;
clear tach;

%convert to deg/sec
TACH = TACH / 204.8;            % 10 V == 2048 units
TACH = TACH * 120 / 2.08;       % 120 deg/sec = 2.08 V

sample = 120;

l = length(TACH);
t = [0:l-1] / sample;

steady_vel = 120;       %ideal chair spin rate
but_thresh = 40;        %threshold above which a button push must be


%look for events (large changes) until start of motion is found
event = delta_tach(TACH,1);
while (abs(TACH(event + sample)) < steady_vel/2)
        %if start, then chair must be up to speed one second later
        event = delta_tach(TACH,event+1);
end
start = event;

% look for next event, and calculate average velocity as
% mean value between (start + 3 seconds) and this event
event = delta_tach(TACH,start + 3 * sample);     %T1 > 3 sec
avg = mean(TACH(start + 3 * sample:event - 5));
level = mean(TACH(event + 10:event + 20));

% if level too low for button push, keep scanning
while (abs(level - avg) < but_thresh)  % noise artifact
        event = delta_tach(TACH,event+1);
        level = mean(TACH(event + 10:event + 20));
end

if ((mean(TACH(event+3:event+8)) - avg) < but_thresh)  %not a button push
        stop = event;
        but1 = NaN;
else
        but1 = event;
        level = but_thresh + avg;  %threshold level

        % look for end of button push
        event = event + 10;
        while (mean(TACH(event:event+2)) > level)
                event = event + 10;
        end

        % look for chair stop, skipping button pushes if needed
        event = delta_tach(TACH,event+10);
        while (abs(TACH(event + sample)) > but_thresh)          % button push or noise
                event = event + 10;
                while (TACH(event) > level)     % skip to end of button push
                        event = event + 10;
                end
                event = delta_tach(TACH,event);
        end
        stop = event;

end

% assume button push is at least 1.5 seconds after stop
flag = 1;
event = delta_tach(TACH,stop + 1.5 * sample);
if (event == NaN)          % no button push
        flag = 0;
elseif (TACH(event + 5) > but_thresh)  % noise artifact
        flag = 0;
end
while (flag)
        event = delta_tach(TACH,event+1);
        if (event == NaN)          % no button push
                flag = 0;
        elseif (TACH(event + 5) > but_thresh)  % noise artifact
                flag = 0;
        end
end
but2 = event;

spinv = avg;
delay = (start - 1) / sample;
T1 = (but1 - start) / sample;
T2 = (but2 - stop) / sample;
spinl = (stop - start) / sample;
runlen = t(l);

fprintf('Delay = %6.2f seconds.\n',delay);
fprintf('Spin length = %6.2f seconds.\n',spinl);
fprintf('Spin velocity = %6.2f deg/sec.\n',spinv);
fprintf('Per-rot button = %6.2f seconds.\n',T1);
fprintf('Post-rot button = %6.2f seconds.\n',T2);
fprintf('Run length = %6.2f seconds.\n',runlen);

clear event start stop but1 but2 sample avg but_thresh steady_vel flag level

eval(['save ',run_code,'.parms spinv delay T1 T2 spinl runlen']);

hold off
plot(t,TACH)
hold on
plot([delay,delay],[-500,500],'w')
plot([T1+delay,T1+delay],[-500,500],'b')
plot([spinl+delay,spinl+delay],[-500,500],'w')
plot([T2+spinl+delay,T2+spinl+delay],[-500,500],'b')
plot([delay,delay+spinl],[spinv,spinv],'g')
hold off
```

```
% Three_point

function [scale,noise,offset] = three_point(t,pos,colour)

% This script inputs the angular deviations from the user, and
% calls the 'pick_regions' script so that the fixation regions
% can be selected.  For each deviation, the average value over
% all of the regions is calculated.  The calibration factor for
% each axis is calculated as the angular difference between the
% positive and negative deviations (in degrees), divided by the
% difference between mean positive and negative deviations (in
% arbitrary units).  The calculated mean values are displayed
% to the user graphically as dotted lines, for inspection.
%
% If a zero fixation point is specified, it is used to
% determine the offset value.
%
% The noise is estimated by taking the root-mean-square value of
% the fluctuations about all selected regions.
%
% D. Balkwill 11/27/90

if isempty(pos_deg)
        pos_deg = 11.7;
end

if isempty(neg_deg)
        neg_deg = -11.7;
end

if (pos_deg == neg_deg)
        disp('Calibration range cannot be zero, Symmetrical calibration assumed.');
        neg_deg = -1 * pos_deg;
end

% select positive trace deflection regions
disp(");
disp('Use mouse to select flat-top regions of positive trace deflection.');
pos_regions = pick_regions(t,pos,colour);
[mp,np] = size(pos_regions);
if (mp == 0)
        error('No positive trace deflection flat-top regions, Cannot calibrate.');
end
for i=1:mp
        xpos = [xpos ; pos(pos_regions(i,1):pos_regions(i,2))];
end
mpos = mean(xpos);
hold on
plot([t(1),t(length(t))],[mpos,mpos],'r:');
hold off

% select negative trace deflection regions
disp('Now select flat-top regions of negative trace deflection.');
neg_regions = pick_regions(t,pos,colour);
[mn,nn] = size(neg_regions);
if (mn == 0)
        error('Cannot calibrate, no negative trace deflections selected.');
end
```

```
m = min([mp mn]);
if (mp<mn)
        fprintf('Fewer positive regions chosen,  fitting all paired regions.');
end
if (mn<mp)
        fprintf('Fewer negative regions chosen,  fitting all paired regions.');
end
for i=1:mn
        xneg = [xneg ; pos(neg_regions(i,1):neg_regions(i,2))];
end
mneg = mean(xneg);
hold on
if (colour == 'y')
        plot([t(1),t(length(t))],[mneg,mneg],'r:');
else
        plot([t(1),t(length(t))],[mneg,mneg],':');
end
hold off

% calculate scale factor in deg/unit
scale = (pos_deg - neg_deg) ./ (mpos - mneg);

% select optional zero trace deflection regions
if isempty(y)
        y='n';
end
xzero = [];
if y=='y' | y=='Y'
        disp('Select regions in which subject fixated on zero reference.');
        zero_regions = pick_regions(t,pos,colour);
        [m,n] = size(zero_regions);
        if (m == 0)
                offset = 0;
        else
                for i=1:m
                        xzero = [xzero ; pos(zero_regions(i,1):zero_regions(i,2))];
                end
                offset = mean(xzero);
                hold on
                if (colour == 'y')
                        plot([t(1),t(length(t))],[offset,offset],'r:');
                else
                        plot([t(1),t(length(t))],[offset,offset],':');
                end
                hold off
        end
else
        offset = 0;
end

% display positive and negative mean values
hold on
if (colour == 'y')
        plot([t(1),t(length(t))],[mneg,mneg],'r:');
        plot([t(1),t(length(t))],[mpos,mpos],'r:');
else
        plot([t(1),t(length(t))],[mneg,mneg],':');
        plot([t(1),t(length(t))],[mpos,mpos],':');
```

```
end
hold off

% estimate noise, in rms degrees
xpos = xpos - mean(xpos);
xneg = xneg - mean(xneg);
if (isempty(xzero) == 0)
        xzero = xzero - mean(xzero);
end
x = [xpos ; xneg ; xzero ];
noise = scale * sqrt(mean(x.*x))

clear pos_regions neg_regions zero_regions xpos xneg xzero x
clear i m n y neg_deg pos_deg
return;
```

```
% Tsq_pvalues
%
% This routine loads in the distributions which were simulated
% by "new_tsq", and calculates the precentiles for each (N1,N2)
% combination corresponding to p = 0.5, 0.975, 0.99, 0.995, and
% 0.999.  These percentiles are saved on the disk for future
% use and interpolation by "spv_stats".
%
% D. Balkwill 12/6/91

load SLS_HD:users:Chris:tsq_dist

L = length(dist8_4);
N = sum(dist8_4);

N1 = 10;
N2 = 10;

n1 = 2 * [1:N1]';
n2 = [1:N2];
Mn1 = n1 * ones(1,N2);
Mn2 = ones(N1,1) * n2;

p50 = L * ones(N1,N2);
p05 = L * ones(N1,N2);
p025 = L * ones(N1,N2);
p01 = L * ones(N1,N2);
p005 = L * ones(N1,N2);
p001 = L * ones(N1,N2);

thresh50 = .50 * N;
thresh05 = .95 * N;
thresh025 = .975 * N;
thresh01 = .99 * N;
thresh005 = .995 * N;
thresh001 = .999 * N;

for i=1:N1
        fprintf('\n%2.0f:',n1(i));
        for j=1:N2
                fprintf(' %2.0f',n2(j));
                eval(['dist = dist',int2str(n1(i)),'_',int2str(n2(j)),';']);
                plot(dist)
                total = 0;
                for k50=1:L
                        total = total + dist(k50);
                        if (total > thresh50)
                                p50(i,j) = k50;
                                break;
                        end
                end
                for k05=(k50+1):L
                        total = total + dist(k05);
                        if (total > thresh05)
                                p05(i,j) = k05;
                                break;
                        end
                end
        for k025=(k05+1):L
            total = total + dist(k025);
            if (total > thresh025)
                p025(i,j) = k025;
                break;
            end
        end
        for k01=(k025+1):L
            total = total + dist(k01);
            if (total > thresh01)
                p01(i,j) = k01;
                break;
            end
        end
        for k005=(k01+1):L
            total = total + dist(k005);
            if (total > thresh005)
                p005(i,j) = k005;
                break;
            end
        end
        for k001=(k005+1):L
            total = total + dist(k001);
            if (total > thresh001)
                p001(i,j) = k001;
                break;
            end
        end
    end
end
```

```
% T_square
%
% Program which calculates the sum of T square statistic

[meanA, varA, nA] = declow(subA, condA,periodA);
[meanB, varB, nB] = declow(subB, condB,periodB);

if length(meanA) > length(meanB),
        tlength = length(meanB);
        meanA = meanA(1:tlength);
        varA = varA(1:tlength);
        nA = nA(1:tlength);
elseif length(meanB) > length(meanA),
tlength = length(meanA);
meanB = meanB(1:tlength);
        varB = varB(1:tlength);
        nB = nB(1:tlength);
else
        tlength = length(meanA);
end

wons = ones(1,tlength);
den = varA + varB;
t2 = ((meanA-meanB).^2)./den;
t2(pack_true(isnan(t2))) = zeros(1:sum(isnan(t2)));
sumt2 = sum(t2);
n_sumt2 = length(t2);
meannA = mean(nA);
meannB = mean(nB);
fprintf('\nSubject ');
fprintf(subA);
fprintf(' (');
fprintf(periodA);
fprintf('-flight ');
fprintf(condA);
fprintf(') versus (');
fprintf(periodB);
fprintf('-flight ');
fprintf(condB);
fprintf(')\n\n');
fprintf('# dof      = %g\n',n_sumt2);
fprintf('N1                 = %g\n',meannA);
fprintf('N2                 = %g\n',meannB);
fprintf('Sum of t2 = %g\n\n',sumt2);

l = length(meanA);
t = ([0:l-1])*1.5;
t = t';

hold off
axis([0 60 -20 100])
xlabel('Time (seconds)')
ylabel('Magnitude of SPV (deg/sec)')
plot(t,meanA)
hold on
plot(t,meanB,'--')
hold off
%pause
```

%prtsc

Appendix B: In-flight specific data analysis Matlab scripts.

Some scripts were modified to account for the differences between the methods of the in-flight and gound testing. (Section 2.4).

The scripts that were modifed by the present author were:

In-Flight name	Equivalent ground name
Flight_analyse	Batch_analyse
Flight_cal_gen	Cal_factor_gen
Flight_calibrate	Calibrate
Flight_dec25	Dec30
Flight_exp_fit	Ind_model_fit_exp
Flight_mean	Dec_mean
Flight_report	Dec_report
Flight_stat_per	Stat
Flight_stat_post	Stat
Flight_stat_prep	Stat_prep_batch
Flight_three_point	Three_point

```
% Flight_analyse

% written by T. Liefeld throughout spring 93
% given a folder of runs from a BDC,  this functions as a
% superscript that will prompt the user for all analysis
% from data collection through to model fitting.
%
% Modified to flow through the in-flight analysis pipeline
% by Christopher Pouliot

clear
hold off

hard_disk = 'backup';

data_path=input('Enter Data Path      >> ','s');
sub_code =input('Enter Subject Code   >> ','s');
number =  input('Enter Number of Runs >> ');

q1 = input('Do you want to do a calibration? >> ','s');
if ((q1=='y') | (q1=='Y'))
        q = input('Calibration factors from file or new? (f/n) >> ','s');
        if ((q == 'f') | (q == 'F'))
                cal_from_file
        end
        if ((q == 'n') | (q == 'N'))
                flight_cal_gen
        end
end;

% create the run_code matrix, codes
CODES
number = number-n_cals;

q2 = input('Do you want to perform stat prep >> ','s');

if ((q2 == 'y') | (q2 == 'Y'))
        for i = 1:number
                run_code = codes(i,:);
                fprintf(['\nRun code = ',run_code,'\n']);
                flight_stat_prep;
        end
end

q3 = input('Do you want to decimate the data >> ','s');

if ((q3 == 'y') | (q3 == 'Y'))
        for i = 1:number
                run_code = codes(i,:);
                fprintf(['\nRun code = ',run_code,'\n']);
                flight_dec25;
        end
end

q4 = input('Do you want to fit a Model? >>','s');
if ((q4=='y') | (q4=='Y'))
        for i = 1:number
                run_code = codes(i,:);
        flight_exp_fit2;
    end
end;
```

```
% Flight_cal_gen
%
% calls calibrate for a number of runs and generates the cal factors
% for the PRN and dumping runs
% Ground version by T. Liefield
% Modified for in-flight analysis by C. Pouliot

run = ones(1,number);
n_cals = input('How many cals >> ');
dim = 1;

if (n_cals >= 0)
        fprintf('\n Enter the run number for the cals in order')
        fprintf('\n from lowest to highest')
        for i=1:n_cals
                calnum(i) = input('cal #  >> ');
                run(calnum(i)) = 0;
                if (calnum(i) < 10)
                        n = num2str(calnum(i));
                        cal_code = [data_path,sub_code,'0',n]
                        flight_calibrate
                        hcal(i) = scale1;
                        g(i) = input('Was this cal good enough to use >> ','s');
                else
                        if (calnum(i)>=10)
                                n = num2str((calnum(i)));
                                cal_code = [data_path,sub_code,n]
                                flight_calibrate
                                hcal(i) = scale1;
                                g(i) = input('Was this cal good enough to use >> ','s');
                        end
                end
        end
end

% calculate calibration factors for the runs
% based only on the good calibrations
j = 1;
for i = 1:n_cals
        if ((g(i)== 'y') | (g(i) == 'Y'))       % g = chr array good or bad
                g_cal(j) = calnum(i);           % calnum all run# which are cals
                calh(j) = hcal(i);
                j = j + 1;
        end
end

hcal = calh;
calibration_calc

q = input('Would you like to save measured cal values? >>','s');
if ((q == 'y') | (q == 'Y'))
        save_name = input('Save File Name : ','s');
        save_name = [data_path,save_name];
        eval(['save ',save_name,' sub_code number dim g_cal hcal run n_cals hor_cal'])
end
```

```
% Flight_calibrate
%
% This script allows the user to obtain a calibration factor
% in degrees/unit.  It assumes a three point calibration in
% each direction, although the zero is optional.  For each axis,
% the user must specify the angular deviations and select the
% "flat" regions of the trace which correspond to fixations on
% the targets.
%
% Ground version by D. Balkwill  11/27/90
% Modified for in-flight analysis by C. Pouliot

sample = 100;
code = cal_code;
colour = 'y';

fprintf('\nCalibrating Axis#1...\n');
eval(['load ',cal_code,'.eogh']);
pos = eogh;
clear eogh
t = (([1:length(pos)] - 1)/sample)';

[scale1,noise1,offset1] = flight_three_point(t,pos,colour);
fprintf('\nAxis#1 scale factor = %6.4f deg/unit\n',scale1);
```

```
% Flight_dec25
% Originally written to decimate ground data from 120 Hz down to 4 Hz by T. Liefield
% modified by C. Pouliot to decimate from 100 Hz down to 4 Hz.

eval(['load ',run_code,'.good']);
eval(['load ',run_code,'.norm']);

% set bad data to zero for summation purposes
norm_spv = norm_spv';
norm_spv = norm_spv .* good_data;

l = length(good_data);

new_l = (l - 1)/25;          %new sampling frequency

y = zeros(25,new_l);
g = zeros(25,new_l);
n = zeros(1,new_l);
d = zeros(1,new_l);
x = zeros(1,new_l);
var = zeros(1,new_l);
z = 1:1:25;

for t=1:(new_l-1)
        for i = 1:25
                y(i,t) = norm_spv(25*t+i);
                g(i,t) = good_data(25*t+i);
        end;
        x = sum(y);
        n = sum(g);
        [a] = polyfit(z,y(:,t)',1);  %linear least squares fit to each bin
        for i = 1:25                          % calc variance about the linear fit
                d(t) = d(t) +((y(i,t) - (a(2)+a(1)*i)).*g(i,t)).^2;
        end;
end

dec_good = (n>0);          % good data flag

% Decimated SPV is box-car average across row, with n=number of
% good samples.  Correction in denominator to prevent division
% by zero for an entire bin of bad data;  dec_spv=0 in this case.
dec_spv = x ./ (n + (~dec_good));

% Variance within trace is variance of each bin around a linear
% polynomial fit to each bin. A correction in case of
% good samples in bin is <= 1;  variance within=0 in this case.
within = d ./ (n -1 + 2*(n<=1) );


% remove outliers in the variance, replace with means of each
% of three equal sized regions
[y,i] = sort(within);
len3 = (length(y)/3) - rem(length(y)/3,1);
mean1 = mean(y(1:len3));
mean2 = mean(y((1+len3):(2*len3)));
mean3 = mean(y((1+2*len3):(3*len3)));
var(i(1:len3)) = mean1*ones(1,len3);
var(i((1+len3):(2*len3))) = mean2*ones(1,len3);
var(i((1+2*len3):(3*len3))) = mean3*ones(1,len3);
within = var;

% calculate weights as the number of samples divided by the variance of each
% sample.
dec_weight = n ./ within;
bad_weight = pack_true(isnan(dec_weight));
dec_weight(bad_weight) = zeros(1,max(size(bad_weight)));
zero_var = pack_true((within<=1e-7));
dec_weight(zero_var) = zeros(1,max(size(zero_var)));

eval(['save ',run_code,'.dec_spv dec_spv']);
eval(['save ',run_code,'.dec_weight dec_weight']);
eval(['save ',run_code,'.within within']);
eval(['save ',run_code,'.dec_good dec_good']);
```

```
% Flight_exp_fit
% Ground version by T. Liefeld 06/12/92
% to fit an exponential model to data
%
% modified C. Pouliot
% to fit  in-flight data

%
% load data
%

eval(['load ',run_code,'.dec_spv']);
eval(['load ',run_code,'.dec_good']);
eval(['load ',run_code,'.parms']);

delay = delay * 4 - rem(delay*4,1);
spinl = spinl * 4 - rem(spinl*4,1);
runlen = runlen * 4 - rem(runlen*4,1);

save_good = dec_good;
good_indices = pack_true(dec_good);
if (spinv < 0)
        dec_spv = -dec_spv;
end

%
% Initialize time vector, assuming 4 Hz decimated frequency
%
l = runlen;
t = ([1:l] - 0.5) / 4;
t = t';

%
% shape tach signal with exponential (0.17 sec time constant)
% ramp to a steady state level at 'spinv'
%
Tv = 0.17;
u = [zeros(1,delay) ones(1,spinl) zeros(1,runlen-spinl-delay)];
u = u';

%
% overall control input (tach)
%
u = lsim( spinv/Tv, [1, 1/Tv], u, t);

%
% Nominal model parameters.  The parameters to be fitted are the
% non-dimensional ratios of the physical parameters to the
% nominal model parameters here.  This places equal emphasis
% on each model parameter, even though they may be orders of
% magnitude apart.

K = .6;  % gain constant
T = 15;  %  time constant
A = -120;          % alpha_m  amplitude of step input        -- fixed
norm_parms = [K ; T ;  A];

options = [0 ; 0.001 ; 0.001];      %error tolerances -- see "help foptions"
```

```
vlb = [.1; .01; 1];  %lower bounds
vub = [10; 10; 1]; %upper bounds

plot(t(good_indices),dec_spv(good_indices))

%
% Fit the per-rotatory portion first
%

fprintf(['\n\n\nFitting ',run_code,' per-rotatory\n']);

dec_good = save_good(delay:(spinl+delay));
dec_good(1:delay) = zeros(1,delay); % do not fit the delay
dec_good((delay+spinl + 1):runlen) = zeros(1,(runlen - delay - spinl));

if (sum(dec_good) < 10)
        fprintf('Not enough data points to determine a curve fit.\n');
        return;
end

good_indices = pack_true(dec_good);

model_parms = [1; 1; 1];
[model_parms, options] = constr('model_err_exp', model_parms, options, vlb, vub, [], t, u,
dec_spv, good_indices, norm_parms);

model_parms = model_parms .* norm_parms;
eval(['save ',run_code,'.eperfit model_parms options'])

fprintf('*** Model fit: initial model parameters = 1.0\n');
fprintf('Number of iterations = %5.0f\n',options(10));
fprintf('Mean square error = %7.4f\n',options(8));

fprintf('K = %f\n',model_parms(1));
fprintf('T = %f\n',model_parms(2));
fprintf('A = %f\n',model_parms(3));

%
% Fit the post-rotatory portion now
%

fprintf(['\n\n\nFitting ',run_code,' post-rotatory\n']);

clear dec_good
the_end = spinl + delay + 240;
if the_end > runlen,
         the_end = runlen;
end

dec_good = save_good((spinl + delay + 1):the_end);

if (sum(dec_good) < 10)
        fprintf('Not enough data points to determine a curve fit.\n');
        return;
end
t=t(1:(the_end - spinl - delay));
dec_spv_p = dec_spv((spinl + delay + 1):the_end);
good_indices = pack_true(dec_good);
```

```
norm_parms = [K ; T ; -1*A];
model_parms = [1; 1; 1];
[model_parms, options] = constr('model_err_exp', model_parms, options, vlb, vub, [], t, u,
dec_spv_p, good_indices,norm_parms);

model_parms = model_parms .* norm_parms;
eval(['save ',run_code,'.newpost model_parms options'])

fprintf('*** Second fit: initial model parameters = 1.0\n');
fprintf('Number of iterations = %5.0f\n',options(10));
fprintf('Mean square error = %7.4f\n',options(8));

fprintf('K = %f\n',model_parms(1));
fprintf('T = %f\n',model_parms(2));
fprintf('A = %f\n',model_parms(3));
```

```
% Flight_mean
% Program which averages runs together to net a mean response.
% by Chris Pouliot

clear

disk = 'backup:FLIGHT_DATA:';
sub = 'FT4';
data_path = [disk,sub,':'];

stat_code = ['SLS_HD:T:', sub, '.per'];
runs = ['10';'04';'08'];
[mean_spv, var_spv, total] = flight_stat_per(sub,data_path,runs,1,250);
graph
eval(['save ',stat_code,' mean_spv var_spv total runs']);
clear mean_spv var_spv total runs stat_code

stat_code = ['SLS_HD:', sub, '.he'];
runs = ['04';'08'];
[mean_spv, var_spv, total] = flight_stat_post(sub,data_path,runs,1,250);
graph
eval(['save ',stat_code,' mean_spv var_spv total runs']);
clear mean_spv var_spv total runs stat_code
```

```
% Flight_report
% written by C. Pouliot
% to print in-flight decimated data

parse;

eval(['load ',data_path, run_code,'.dec_spv']);
eval(['load ',data_path,run_code,'.dec_good']);
eval(['load ',data_path,run_code,'.parms']);

eval(['load ',data_path,run_code,'.eperfit']);
K(1) = model_parms(1);
T(1) = model_parms(2);
E(1) = options(8);
clear model_parms options

eval(['load ',data_path,run_code,'.newpost']);
K(2) = model_parms(1);
T(2) = model_parms(2);
E(2) = options(8);
clear model_parms options

percent1 = 100*mean(dec_good(delay:(delay+100)));
percent2 = 100*mean(dec_good((delay+spinl):(delay+spinl+100)));

A(1) = -120;
A(2) = +120;

if spinv == -120,
        A(1) = 120;
        A(2) = -120;
end

t = linspace(1,round(runlen),(round(runlen*4)));

hold off
clg
subplot (211);

text (.05,.98,'SLS-2 EO72 ROTATING CHAIR RUN SUMMARY', 'sc');

text (.1,.93,'subject', 'sc');
text (.5,.93, subject, 'sc');

text (.1,.91,'session', 'sc');
text (.5,.91, session, 'sc');

text (.1,.89,'run', 'sc');
text (.5,.89,run_num, 'sc');

text (.1,.87,'test type', 'sc');
text (.5,.87,test_type, 'sc');

text (.1,.84,'spin length', 'sc');
text (.5,.84, num2str(spinl), 'sc');

text (.1,.82,'run length', 'sc');
text (.5,.82, num2str(runlen), 'sc');
text (.1,.80,'spin velocity', 'sc');
text (.5,.80, num2str(spinv), 'sc');

spinl = spinl * 4;
delay = delay * 4;
runlen = runlen * 4;
percent1 = 100*mean(dec_good(delay:(delay+100)));
percent2 = 100*mean(dec_good((delay+spinl):(delay+spinl+100)));

text (.1,.77,'per-rotatory gain', 'sc');
text (.5,.77, num2str(K(1)), 'sc');

text (.1,.75,'per-rotatory time constant', 'sc');
text (.5,.75, num2str(T(1)), 'sc');

text (.1,.73,'per-rotatory percent good data', 'sc');
text (.5,.73, num2str(percent1), 'sc');

text (.1,.71,'per-rotatory MSE', 'sc');
text (.5,.71, num2str(E(1)), 'sc');

text (.1,.68,'post-rotatory gain', 'sc');
text (.5,.68, num2str(K(2)), 'sc');

text (.1,.66,'post-rotatory time constant', 'sc');
text (.5,.66, num2str(T(2)), 'sc');

text (.1,.64,'post-rotatory percent good data', 'sc');
text (.5,.64, num2str(percent2), 'sc');

text (.1,.62,'post-rotatory MSE', 'sc');
text (.5,.62,num2str(E(2)), 'sc');

subplot (212);
time = runlen/4;
axis([0 time -150 150]);
% eval (['title (''', run_code, ''')']);
xlabel('Time since start of run (sec)');
ylabel('Slow Phase Velocity (deg/sec)');
plot(t(dec_good),dec_spv(dec_good),'--');
hold on

x=A(1)*K(1)*exp(-1*t/T(1));
plot(t((delay+1):(delay+spinl)),x(1:spinl));

y=A(2)*K(2)*exp(-1*t/T(2));
plot(t((spinl+delay+1):runlen),y(1:(runlen-spinl-delay)));
prtsc

hold off
subplot (111)
axis('normal')
```

```
% Flight_stat_per
% Function which calculates the mean and variance of a selected number of
% individual in-flight per-rotatory SPV responses
% by Chris Pouliot

function [mean_spv, var_spv, total] = flight_stat_per(sub,data_path,runs,s,e)

sample = 4;

minute_size = 60 * sample;
sum_spv = zeros(1,500);
sum_square = zeros(1,500);
total = zeros(1,500);

for j = 1:length(runs(:,2))
        run_code = [data_path,sub,runs(j,:)];
        eval(['load ',run_code,'.dec_spv'])
        eval(['load ',run_code,'.dec_good'])
        eval(['load ',run_code,'.parms'])
        eval(['load ',run_code,'.eperfit'])

        delay = round(delay*4);
        spinl = round(spinl*4);
        runlen = round(runlen*4);
        dec_spv1 = zeros(1,500);
        dec_good1 = zeros(1,500);
        dec_spv1(1:(spinl-delay)) = dec_spv((delay+1):spinl);
        dec_good1(1:(spinl-delay)) = dec_good((delay+1):spinl);
        dec_spv = dec_spv1;
        dec_good = dec_good1;

        perc_good = 100*mean(dec_good(1:(spinl-delay)));
        fprintf('Run ');
        fprintf(runs(j,:));
        fprintf(' percent good    %g\n',perc_good);
        fprintf('          gain                %g\n',model_parms(1));
        fprintf('          time constant %g\n',model_parms(2));
        fprintf('          MSE                  %g\n',options(8));
        fprintf('          spinv               %g\n\n',spinv);

        sum_spv = sum_spv + ((spinv/abs(spinv)) .* dec_spv .* dec_good);
        sum_square = sum_square + (dec_spv .* dec_spv .*dec_good);
        total = total + dec_good;
        clear dec_spv dec_good spinv
end

mean_spv = sum_spv ./ total;
mean_spv(pack_true(isnan(mean_spv))) = zeros(1:sum(isnan(mean_spv)));
var_spv = sum_square - (sum_spv .* mean_spv);
var_spv = var_spv ./ (total - 1);
var_spv(pack_true(isnan(var_spv))) = zeros(1:sum(isnan(var_spv)));
clear sum_spv sum_square

mean_spv = -mean_spv(s:e);
var_spv = var_spv(s:e);
total = total(s:e);
plot(mean_spv)
```

```
% flight_stat_post
% Function which calculates the mean and variance of a selected number of
% individual in-flight post-rotatory SPV responses
% by Chris Pouliot

function [mean_spv, var_spv, total] = kc_stat(sub,data_path,runs,s,e)

sample = 4;

minute_size = 60 * sample;
sum_spv = zeros(1,2200);
sum_square = zeros(1,2200);
total = zeros(1,2200);

for j = 1:length(runs(:,2))
        run_code = [data_path,sub,runs(j,:)];
        eval(['load ',run_code,'.dec_spv'])
        eval(['load ',run_code,'.dec_good'])
        eval(['load ',run_code,'.parms'])
        eval(['load ',run_code,'.newpost'])

        delay = round(delay*4);
        spinl = round(spinl*4);
        runlen = round(runlen*4);
        dec_spv1 = zeros(1,2200);
        dec_good1 = zeros(1,2200);
        dec_spv1(1:(runlen-delay-spinl)) = dec_spv((delay+spinl+1):runlen);
        dec_good1(1:(runlen-delay-spinl)) = dec_good((delay+spinl+1):runlen);
        dec_spv = dec_spv1;
        dec_good = dec_good1;

        perc_good = 100*mean(dec_good(1:(runlen-delay-spinl)));
        fprintf('Run ');
        fprintf(runs(j,:));
        fprintf(' percent good %g\n',perc_good);
        fprintf(' gain       %g\n',model_parms(1));
        fprintf(' time constant %g\n',model_parms(2));
        fprintf('MSE        %g\n',options(8));
        fprintf('spinv      %g\n\n',spinv);

        sum_spv = sum_spv + ((spinv/abs(spinv)) .* dec_spv .* dec_good);
        sum_square = sum_square + (dec_spv .* dec_spv .*dec_good);
        total = total + dec_good;
         clear dec_spv dec_good spinv
end

mean_spv = sum_spv ./ total;
mean_spv(pack_true(isnan(mean_spv))) = zeros(1:sum(isnan(mean_spv)));
var_spv = sum_square - (sum_spv .* mean_spv);
var_spv = var_spv ./ (total - 1);
var_spv(pack_true(isnan(var_spv))) = zeros(1:sum(isnan(var_spv)));
clear sum_spv sum_square

mean_spv = mean_spv(s:e);
var_spv = var_spv(s:e);
total = total(s:e);
plot(mean_spv)
```

```
% Flight_stat_prep
% Ground version by D. Balkwill 8/91
% Modified by Chris Pouliot to handle the inflight 100 Hz sampling rate.

sample = 100;

eval(['load ',run_code,'.aspvh']);
eval(['load ',run_code,'.parms']);

y = aspvh;
clear aspvh

runlen = runlen*sample;
delay = delay*sample ;
spinl = spinl*sample;

t = ([1:runlen] ./ sample);
good_data = ones(1,runlen);

%
%find valid range for log outlier detection in per-rotatory section
%

fprintf('Per-rotatory:\n');
i = delay;
j = i + 5*sample;
condition = 1;
while ((abs(mean(y(j:j+5*sample))) > 10) & (condition == 1))
        if ((j+7*sample) > (delay+spinl)),
                condition = 0;
        else
                j = j + 2 * sample;
        end
end
if condition == 0,
        j = j + 2 * sample;
end
t1 = t(i:j);
y1 = y(i:j);

[logout1,under,over,m,b] = log_outlier(t1,y1);

%take care of under- or over-flow
good_data(i:j) = ~logout1;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
if (over > 0)
        good_data(j+1:j+over) = zeros(1,over);
end

%save final fit in parms file as first-order "model fit"
tau1 = -1/m;
gain1 = exp(b)/120;
fprintf('Time length of outliers from log fit is ');
fprintf('%5.2f seconds.\n',(sum(logout1)+under+over)/sample);

% do magnitude outlier detection on remainder of per-rotatory SPV
```

```
i = j + over + 1;
[magout1,under,over] = mag_outlier(t((i:spinl+delay)),y(i:(spinl+delay)),30);
good_data(i:(spinl+delay)) = ~magout1;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
fprintf('Time length of outliers from magnitude threshold is');
fprintf(' %5.2f seconds.\n',(sum(magout1)+under)/sample);

fprintf('Post-rotatory:\n');
i = delay + spinl + sample + 1; %one second after stop
j = i + 5*sample;
condition = 1;
while ((abs(mean(y(j:j+5*sample))) > 10) & (condition == 1))
        if ((j+7*sample) > runlen),
                condition = 0;
        else
                j = j + 2 * sample;
        end
end
if condition == 0,
        j = j + 2 * sample;
end
t2 = t(i:j);
y2 = y(i:j);

[logout2,under,over,m,b] = log_outlier(t2,y2);

%take care of under- or over-flow
good_data(i:j) = ~logout2;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
if (over > 0)
        good_data(j+1:j+over) = zeros(1,over);
end

%save final fit in parms file as first-order "model fit"
tau2 = -1/m;
gain2 = exp(b + 60 * m)/120;
fprintf('Time length of outliers from log fit is ');
fprintf('%5.2f seconds.\n',(sum(logout2)+under+over)/sample);

% do magnitude outlier detection on remainder of per-rotatory SPV

i = j + over + 1;
[magout2,under,over] = mag_outlier(t(i:runlen),y(i:runlen),30);
good_data(i:runlen) = ~magout2;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
fprintf('Time length of outliers from magnitude threshold');
fprintf(' is %5.2f seconds.\n',(sum(magout2)+under)/sample);
% don't fill in any overflow, because this would be past two minutes

fprintf('Overall percentage of good data is %6.2f\n',100*mean(good_data));
```

```
%plot data and outlying regions
hold off
plot(t1,log(abs(y1)))
hold on
plot(t1,log(abs(mean(y1))) * logout1,'b--')
grid
xlabel('Time (sec)')
ylabel('ln(SPV)')
title([run_code,' -- SPV and log outlier indicator'])

hold off
plot(t2,log(abs(y2)))
hold on
plot(t2,log(abs(mean(y2))) * logout2,'b--')
grid
xlabel('Time (sec)')
ylabel('ln(SPV)')
title([run_code,' -- SPV and log outlier indicator'])

hold off
plot(t,y)
hold on
plot(t,50*(~good_data),'b--')
plot(t,-50*(~good_data),'b--')
grid
xlabel('Time (sec)')
ylabel('SPV (deg/sec)')
title([run_code,' -- SPV and overall outlier indicator'])

hold off
plot(t(good_data),y(good_data),'.')
grid
xlabel('Time (sec)')
ylabel('SPV (deg/sec)')
title([run_code,' -- good SPV (outliers removed)']);

hold off

clear t t1 t2 y1 y2 i j logout1 logout2 minute_size sample
clear delay spinl spinv T1 T2 runlen under over magout1 magout2
clear m b tau1 gain1 tau2 gain2

norm_spv = y;

clear y

% save normalized data, having departed from 'file_specs' at this point

eval(['save ',run_code,'.good good_data']);
eval(['save ',run_code,'.norm norm_spv']);

clear good_data norm_spv
clear t tau1 tau2 run_len spinl spinv
```

```
% Flight_three_point

function [scale,noise,offset] = three_point(t,pos,colour)

% This script inputs the angular deviations from the user, and
% calls the 'pick_regions' script so that the fixation regions
% can be selected.  For each deviation, the average value over
% all of the regions is calculated.  The calibration factor for
% each axis is calculated as the angular difference between the
% positive and negative deviations (in degrees), divided by the
% difference between mean positive and negative deviations (in
% arbitrary units).  The calculated mean values are displayed
% to the user graphically as dotted lines, for inspection.
%
% If a zero fixation point is specified, it is used to
% determine the offset value.
%
% The noise is estimated by taking the root-mean-square value of
% the fluctuations about all selected regions.
%
% Ground version by D. Balkwill 11/27/90
% Modified by Chris Pouliot to analyze in-flight data

if isempty(pos_deg)
        pos_deg = 9;
end

if isempty(neg_deg)
        neg_deg = -9;
end

if (pos_deg == neg_deg)
        disp('Calibration range cannot be zero, Symmetrical calibration assumed.');
        neg_deg = -1 * pos_deg;
end

% select positive trace deflection regions
disp('');
disp('Use mouse to select flat-top regions of positive trace deflection.');
pos_regions = pick_regions(t,pos,colour);
[mp,np] = size(pos_regions);
if (mp == 0)
        error('No positive trace deflection flat-top regions, Cannot calibrate.');
end
for i=1:mp
         xpos = [xpos ; pos(pos_regions(i,1):pos_regions(i,2))];
end
mpos = mean(xpos);
hold on
plot([t(1),t(length(t))],[mpos,mpos],'r:');
hold off

% select negative trace deflection regions
disp('Now select flat-top regions of negative trace deflection.');
neg_regions = pick_regions(t,pos,colour);
[mn,nn] = size(neg_regions);
if (mn == 0)
        error('Cannot calibrate, no negative trace deflections selected.');
```

```
end
m = min([mp mn]);
if (mp<mn)
        fprintf('Fewer positive regions chosen,  fitting all paired regions.');
end
if (mn<mp)
        fprintf('Fewer negative regions chosen,  fitting all paired regions.');
end
for i=1:mn
        xneg = [xneg ; pos(neg_regions(i,1):neg_regions(i,2))];
end
mneg = mean(xneg);
hold on
if (colour == 'y')
        plot([t(1),t(length(t))],[mneg,mneg],'r:');
else
        plot([t(1),t(length(t))],[mneg,mneg],':');
end
hold off

% calculate scale factor in deg/unit
scale = (pos_deg - neg_deg) ./ (mpos - mneg);

% select optional zero trace deflection regions
if isempty(y)
        y='n';
end
xzero = [];
if y=='y' | y=='Y'
        disp('Select regions in which subject fixated on zero reference.');
        zero_regions = pick_regions(t,pos,colour);
        [m,n] = size(zero_regions);
        if (m == 0)
                offset = 0;
        else
                for i=1:m
                        xzero = [xzero ; pos(zero_regions(i,1):zero_regions(i,2))];
                end
                offset = mean(xzero);
                hold on
                if (colour == 'y')
                        plot([t(1),t(length(t))],[offset,offset],'r:');
                else
                        plot([t(1),t(length(t))],[offset,offset],':');
                end
                hold off
        end
else
        offset = 0;
end

% display positive and negative mean values
hold on
if (colour == 'y')
        plot([t(1),t(length(t))],[mneg,mneg],'r:');
        plot([t(1),t(length(t))],[mpos,mpos],'r:');
else
        plot([t(1),t(length(t))],[mneg,mneg],':');
```

```
		plot([t(1),t(length(t))],[mpos,mpos],':');
end
hold off

% estimate noise, in rms degrees
xpos = xpos - mean(xpos);
xneg = xneg - mean(xneg);
if (isempty(xzero) == 0)
		xzero = xzero - mean(xzero);
end
x = [xpos ; xneg ; xzero ];
noise = scale * sqrt(mean(x.*x))

clear pos_regions neg_regions zero_regions xpos xneg xzero x
clear i m n y neg_deg pos_deg
return;
```

Appendix C: Parabolic specific data analysis Matlab scripts

Some scripts were modified to account for the differences between the methods of the KC-135 parabolic flight and ground testing. (Section 2.4).

The scripts that were modified by the present author were:

Parabolic-Flight name	Equivalent ground name
Delta_g	---
End_zero_g	---
KC_analyse	Batch_analyse
KC_calibrate	Calibrate
KC_cal_gen	Cal_factor_gen
KC_dec30	Dec30
KC_exp_fit	Ind_model_fit_exp
KC_mean	Dec_mean
KC_report	Dec_report
KC_stat	Stat
KC_stat_prep	Stat_prep_batch
KC_tachan	Tachan_batch
KC_three_point	Three_point

```
% Delta_g
% Function, based on the Delta_tach algorythm (Balkwill, 1992), which finds large
% jumps in the z-accelerometer data.
% by Chris Pouliot

function n = delta_g(z,init)

x = z;

false  = 0;
true = 1;

final = init + 1000;

themean = mean(x(init:final));
thestd = 5*std(x(init:final));

thresholdup = themean + thestd;
thresholddown = themean - thestd;

i = init;
condition = false;
while (condition == false)
        if i == (length(x) - 5)
                condition = true;
        end
        if (x(i) >= thresholdup)
                if ((x(i+1) >= thresholdup) & (x(i+2) >= thresholdup))
                        condition = true;
                end
        elseif (x(i) <= thresholddown)
                if ((x(i+1) <= thresholddown) & (x(i+2) <= thresholddown))
                        condition = true;
                end
        end
        i = i + 1;
end

n = i - 2;
if n == (length(x) - 6)
        n = NaN;
end
```

```
%End_zero_g
% Program which finds the period of zero-gravity by looking at the z-accelerometer data.
% by Chris Pouliot

clear
chdir KC&T:KC135:
other_path = 'KC&T:TP1:';

run_code = input('Enter run code:  ', 's');
eval (['load ', run_code])

z = AData(:,7);

condition = 0;
while condition == 0
        hold off
        clg

        plot(z)
        hold on

        start = input('Enter starting point:  ');
        stop = delta_g(z,start);
        plot([stop, stop], [-20000, 20000], 'b')

        quest = input('Is this the right ending point?? ', 's');
        if (quest(1) == 'Y') | (quest(1) == 'y')
                condition = 1;
        end
end

tach = AData(1:stop, 6);
eogh = AData(1:stop, 2);
eogv = AData(1:stop, 1);

sub_code = input('Enter subject code:  ', 's');
eval(['save ', other_path, sub_code, '.tach tach'])
eval(['save ', other_path, sub_code, '.eogh eogh'])
eval(['save ', other_path, sub_code, '.eogv eogv'])
```

```
% KC_analyse

% written by T. Liefeld throughout spring 93
% given a folder of runs from a BDC,  this functions as a
% superscript that will prompt the user for all analysis
% from data collection through to model fitting.
%
% Modified to flow through the KC-135 parabolic flight pipeline
% by Christopher Pouliot

clear
hold off

hard_disk = 'harddisk';

data_path=input('Enter Data Path       >> ','s');
sub_code =input('Enter Subject Code    >> ','s');
number =  input('Enter Number of Runs >> ');

q1 = input('Do you want to do a calibration? >> ','s');
if ((q1=='y') | (q1=='Y'))
        q = input('Calibration factors from file or new? (f/n) >> ','s');
        if ((q == 'f') | (q == 'F'))
                cal_from_file
        end
        if ((q == 'n') | (q == 'N'))
                kc_cal_gen
        end
end;

q2 = input('Do you want to perform AATM? >> ','s');
if ((q2 == 'y') | (q2 == 'Y'))
        multiple_AATM;
end;

% create the run_code matrix, codes
CODES
number = number-n_cals;

q3 = input('Do you want to perform Tachan >> ','s');

if ((q3 == 'y') | (q3 == 'Y'))
        for i = 1:number
                run_code = codes(i,:);
                fprintf(['\nRun code = ',run_code,'\n']);
        kc_tachan;
        end
end

q4 = input('Do you want to perform stat prep >> ','s');

if ((q4 == 'y') | (q4 == 'Y'))
        for i = 1:number
                run_code = codes(i,:);
                fprintf(['\nRun code = ',run_code,'\n']);
                kc_stat_prep;
        end
end
```

```
q5 = input('Do you want to fit a Model? >>','s');
if ((q5=='y') | (q5=='Y'))
        for i = 1:number
                run_code = codes(i,:);
                kc_exp_fit;
        end
end;
```

```
% KC_calibrate
%
% This script allows the user to obtain a calibration factor
% in degrees/unit.  It assumes a three point calibration in
% each direction, although the zero is optional.  For each axis,
% the user must specify the angular deviations and select the
% "flat" regions of the trace which correspond to fixations on
% the targets.
%
% Ground version by D. Balkwill
% Modified by Chris Pouliot to analyze KC-135 data

eval(['load vel_filter.mat']);
eval(['load dim']);
eval(['load colour']);

code = cal_code;

fprintf('\nCalibrating Axis#1...\n');
eval(['load ',cal_code,'.eogh']);
pos = eogh;
clear eogh
t = (([1:length(pos)] - 1)/sample)';

[scale1,noise1,offset1] = kc_three_point(t,pos,colour);
fprintf('\nAxis#1 scale factor = %6.4f deg/unit\n',scale1);
```

```
% KC_cal_gen
%
% calls calibrate for a number of runs and generates the cal factors
% for the PRN and dumping runs
% Ground analysis version by T. Liefield
% Modified for KC-135 analysis by Chris Pouliot

run = ones(1,number);
n_cals = input('How many cals >> ');
dim = 1;

if (n_cals >= 0)
        fprintf('\n Enter the run number for the cals in order')
        fprintf('\n from lowest to highest')
        for i=1:n_cals
                calnum(i) = input('cal #  >> ');
                run(calnum(i)) = 0;
                if (calnum(i) < 10)
                        n = num2str(calnum(i));
                        cal_code = [data_path,sub_code,'0',n]
                        kc_calibrate;
                        hcal(i) = scale1;
                        g(i) = input('Was this cal good enough to use >> ','s');
                else
                        if (calnum(i)>=10)
                                n = num2str((calnum(i)));
                                cal_code = [data_path,sub_code,n]
                                kc_calibrate
                                hcal(i) = scale1;
                                g(i) = input('Was this cal good enough to use >> ','s');
                        end
                end
        end
end

% calculate calibration factors for the runs
% based only on the good calibrations
j = 1;
for i = 1:n_cals
        if ((g(i)== 'y') | (g(i) == 'Y'))                 % g = chr array good or bad
                g_cal(j) = calnum(i);                     % calnum all run# which are cals
                calh(j) = hcal(i);
                j = j + 1;
        end
end

hcal = calh;
calibration_calc

q = input('Would you like to save measured cal values? >>','s');
if ((q == 'y') | (q == 'Y'))
        save_name = input('Save File Name : ','s');
        save_name = [data_path,save_name];
        eval(['save ',save_name,' sub_code number dim g_cal hcal run n_cals hor_cal'])
end
```

```
% KC_dec30
% Origanlly written to decimate ground data from 120 Hz down to 4 Hz by T. Liefield
% Modified for use in the KC-135 analysis by Chris Pouliot

% set bad data to zero for summation purposes
norm_spv = norm_spv';
norm_spv = norm_spv .* good_data;

l = length(good_data);

new_l = (l - 1)/30;          %new sampling frequency

y = zeros(30,new_l);
g = zeros(30,new_l);
n = zeros(1,new_l);
d = zeros(1,new_l);
x = zeros(1,new_l);
var = zeros(1,new_l);
z = 1:1:30;

for t=1:(new_l-1)
        for i = 1:30
                y(i,t) = norm_spv(30*t+i);
                g(i,t) = good_data(30*t+i);
        end;
        x = sum(y);
        n = sum(g);
        [a] = polyfit(z,y(:,t)',1);  %linear least squares fit to each bin
        for i = 1:30                                   % calc variance about the linear fit
                d(t) = d(t) +((y(i,t) - (a(2)+a(1)*i)).*g(i,t)).^2;
        end;
end;

dec_good = (n>0);          % good data flag

% Decimated SPV is box-car average across row, with n=number of
% good samples.  Correction in denominator to prevent division
% by zero for an entire bin of bad data;  dec_spv=0 in this case.
dec_spv = x ./ (n + (~dec_good));

% Variance within trace is variance of each bin around a linear
% polynomial fit to each bin. A correction in case of
% good samples in bin is <= 1;  variance within=0 in this case.
within = d ./ (n -1 + 2*(n<=1) );


% remove outliers in the variance, replace with means of each
% of three equal sized regions
[y,i] = sort(within);
len3 = (length(y)/3) - rem(length(y)/3,1);
mean1 = mean(y(1:len3));
mean2 = mean(y((1+len3):(2*len3)));
mean3 = mean(y((1+2*len3):(3*len3)));
var(i(1:len3)) = mean1*ones(1,len3);
var(i((1+len3):(2*len3))) = mean2*ones(1,len3);
var(i((1+2*len3):(3*len3))) = mean3*ones(1,len3);
within = var;
```

```
% calculate weights as the number of samples divided by the variance of each
% sample.
dec_weight = n ./ within;
bad_weight = pack_true(isnan(dec_weight));
dec_weight(bad_weight) = zeros(1,max(size(bad_weight)));
zero_var = pack_true((within<=1e-7));
dec_weight(zero_var) = zeros(1,max(size(zero_var)));

%save data, having departed from 'file_specs' by now
eval(['save ',run_code,'.dec_spv dec_spv']);
eval(['save ',run_code,'.dec_weight dec_weight']);
eval(['save ',run_code,'.within within']);
eval(['save ',run_code,'.dec_good dec_good']);
%clear dec_good dec_spv i l     new_l  within  x out_weight zero_var dec_weight
```

```
% KC_exp_fit
% modified T. Liefeld 06/12/92
% to fit an exponential model to data
%
% modified C. Pouliot 08/11/93
% to fit post-rotatory KC-135 data over a 22 second period

%
% load data
%
data_path = 'KC&T:TP2:';
run_code = input('Enter run code >> ', 's');

eval(['chdir ',data_path]);
eval(['load ',run_code,'.dec_spv']);
eval(['load ',run_code,'.dec_good']);
eval(['load ',run_code,'.parms']);

delay = delay * 4 - rem(delay*4,1);
spinl = spinl * 4 - rem(spinl*4,1);
runlen = runlen * 4 - rem(runlen*4,1);

save_good = dec_good;
good_indices = pack_true(dec_good);
if (spinv < 0)
        dec_spv = -dec_spv;
end

%
% Initialize time vector, assuming 4 Hz decimated frequency
%
l = runlen;
t = ([1:l] - 0.5) / 4;
t = t';

%
% shape tach signal with exponential (0.17 sec time constant)
% ramp to a steady state level at 'spinv'
%
Tv = 0.17;
u = [zeros(1,delay) ones(1,spinl) zeros(1,runlen-spinl-delay)];
u = u';

%
% overall control input (tach)
%
u = lsim( spinv/Tv, [1, 1/Tv], u, t);

%
% Nominal model parameters.  The parameters to be fitted are the
% non-dimensional ratios of the physical parameters to the
% nominal model parameters here.  This places equal emphasis
% on each model parameter, even though they may be orders of
% magnitude apart.

K = .4;  % gain constant
T = 10;  %  time constant
A = -120;          % alpha_m  amplitude of step input         -- fixed
```

```
norm_parms = [K ; T ;  A];

options = [0 ; 0.001 ; 0.001];
vlb = [.1; .01; 1];  %lower bounds
vub = [10; 10; 1]; %upper bounds

plot(t(good_indices),dec_spv(good_indices))

%
% Fit the post-rotatory portion now
%

fprintf(['\n\n\nFitting ',run_code,' post-rotatory\n']);

clear dec_good
the_end = spinl + delay + 88;
if the_end > runlen,
        the_end = runlen;
end

dec_good = save_good((spinl + delay + 1):the_end);
dec_good(1:12) = zeros(1,12); % do not fit first 3 seconds of data

if (sum(dec_good) < 10)
        fprintf('Not enough data points to determine a curve fit.\n');
        return;
end
t=t(1:(the_end - spinl - delay));
dec_spv_p = dec_spv((spinl + delay + 1):the_end);
good_indices = pack_true(dec_good);
norm_parms = [K ; T ; -1*A];
model_parms = [1; 1; 1];
[model_parms, options] = constr('model_err_exp', model_parms, options, vlb, vub, [], t, u,
dec_spv_p, good_indices,norm_parms);

model_parms = model_parms .* norm_parms;
eval(['save ',run_code,'.newpost model_parms options'])

fprintf('*** Second fit: initial model parameters = 1.0\n');
fprintf('Number of iterations = %5.0f\n',options(10));
fprintf('Mean square error = %7.4f\n',options(8));

fprintf('K = %f\n',model_parms(1));
fprintf('T = %f\n',model_parms(2));
fprintf('A = %f\n',model_parms(3));
```

```
% KC_mean
% Program which averages runs together to net a net mean response.
% by Chris Pouliot

clear

disk = 'KC&T:';
sub = 'TP2';
data_path = [disk,sub,':'];

stat_code = ['KC&T:', sub, '.kc1ccw'];
runs = ['02';'04'];
[mean_spv, var_spv, total] = kc_stat(sub,data_path,runs,1,250);
eval(['save ',stat_code,' mean_spv var_spv total runs']);
```

```
% KC_report
% written by C. Pouliot
% to print KC-135 decimated data

clear

data_path = 'KC&T:TP1:';
run_code = input('Enter Run Code: ', 's');
parse;

eval(['load ',data_path, run_code,'.dec_spv']);
eval(['load ',data_path,run_code,'.dec_good']);
eval(['load ',data_path,run_code,'.parms']);

eval(['load ',data_path,run_code,'.newpost']);
K(1) = model_parms(1);
T(1) = model_parms(2);
E(1) = options(8);
clear model_parms options

A(1) = -120;
if sum(dec_spv(20:120)) < 0,
        A(1) = 120;
end

len = length(dec_spv);
t = ([1:len] - 0.5) / 4;
t = t';
len_120 = (len/4)*120;

hold off
clg

text (.05,.98,'SLS-2 EO72 ROTATING CHAIR RUN SUMMARY', 'sc');

text (.1,.93,'subject', 'sc');
text (.5,.93, subject, 'sc');

text (.1,.90,'session', 'sc');
text (.5,.90, session, 'sc');

text (.1,.87,'run', 'sc');
text (.5,.87,run_num, 'sc');

text (.1,.84,'test type', 'sc');
text (.5,.84,test_type, 'sc');

text (.1,.79,'spin length', 'sc');
text (.5,.79, num2str(spinl), 'sc');

text (.1,.76,'run length', 'sc');
text (.5,.76, num2str(runlen), 'sc');

text (.1,.73,'spin velocity', 'sc');
text (.5,.73, num2str(spinv), 'sc');

text (.1,.68,'post-rotatory gain', 'sc');
text (.5,.68, num2str(K(1)), 'sc');
```

```
if (K(1) == 0.18) | (K(1) == 1.2)
        text (.8,.68, '***', 'sc');
end

text (.1,.65,'post-rotatory time constant', 'sc');
text (.5,.65, num2str(T(1)), 'sc');
if (T(1) == 4.5) | (T(1) == 30)
        text(.8,.65, '***', 'sc');
end

text (.1,.62,'post-rotatory MSE', 'sc');
text (.5,.62, num2str(E(1)), 'sc');

spinl = spinl * 120;
delay = delay * 120;
runlen = runlen * 120;

eval(['load ',data_path,run_code,'.good']);
percent = 100*mean(good_data((delay+spinl):(runlen)));
clear good_data

text (.1,.57,'percent good data', 'sc');
text (.5,.57, num2str(percent), 'sc');
pause
clg

spinl =  (spinl/120)  * 4;
delay =  (delay/120)  * 4;
runlen = (runlen/120) * 4;

subplot (211);
axis([0 120 -100 100]);
eval (['title ('', run_code, '')']);
xlabel('Time since start of run (sec)');
ylabel('Slow Phase Velocity (deg/sec)');
plot(t(dec_good),dec_spv(dec_good),'.');
hold on

y=A(1)*K(1)*exp(-1*t/T(1));
plot(t((spinl+delay+1):runlen),y(1:(runlen-spinl-delay)));

hold off
eval(['load ', data_path, run_code, '.z'])
l = length(z);
t = ([1:l] - 0.5) / 120;
t = t';
subplot(212)
axis ([0 120 -2 1])
z = (z+4811)/7480;
if len_120 > length(z),
        len_120 = length(z);
end
plot (t(1:len_120), z(1:len_120))
xlabel('Time since start of run (sec)');

hold off
subplot (111)
axis('normal')
```

```
% KC_stat
%
% Function which calculates the mean and variance of a selected number of
% individual KC-135 SPV responses.
%
% by Chris Pouliot

function [mean_spv, var_spv, total] = kc_stat(sub,data_path,runs,s,e)

sample = 4;

minute_size = 60 * sample;
sum_spv = zeros(1,500);
sum_square = zeros(1,500);
total = zeros(1,500);

for j = 1:length(runs(:,2))
        run_code = [data_path,sub,runs(j,:)];
        eval(['load ',run_code,'.dec_spv'])
        eval(['load ',run_code,'.dec_good'])
                                        eval(['load ',run_code,'.parms'])
        eval(['load ',run_code,'.newpost'])

        delay = round(delay*4);
        spinl = round(spinl*4);
        runlen = round(runlen*4);
        dec_spv1 = zeros(1,500);
        dec_good1 = zeros(1,500);
        dec_spv1(1:(runlen-delay-spinl)) = dec_spv((delay+spinl+1):runlen);
        dec_good1(1:(runlen-delay-spinl)) = dec_good((delay+spinl+1):runlen);
        dec_spv = dec_spv1;
        dec_good = dec_good1;

        perc_good = 100*mean(dec_good(1:(runlen-delay-spinl)));
        fprintf('Run ');
        fprintf(runs(j,:));
        fprintf(' percent good    %g\n',perc_good);
        fprintf('          gain              %g\n',model_parms(1));
        fprintf('          time constant %g\n',model_parms(2));
        fprintf('          MSE                      %g\n',options(8));
        fprintf('          spinv            %g\n\n',spinv);

        a = ((spinv/abs(spinv)) .* dec_spv(1) .* dec_good(1))
        sum_spv = sum_spv + ((spinv/abs(spinv)) .* dec_spv .* dec_good);
        sum_square = sum_square + (dec_spv .* dec_spv .*dec_good);
        total = total + dec_good;
        clear dec_spv dec_good spinv
end

mean_spv = sum_spv ./ total;
mean_spv(pack_true(isnan(mean_spv))) = zeros(1:sum(isnan(mean_spv)));
var_spv = sum_square - (sum_spv .* mean_spv);
var_spv = var_spv ./ (total - 1);
var_spv(pack_true(isnan(var_spv))) = zeros(1:sum(isnan(var_spv)));
clear sum_spv sum_square

mean_spv = mean_spv(s:e);
var_spv = var_spv(s:e);
```

```
total = total(s:e);
plot(mean_spv)
```

```
% KC_stat_prep
%
% Prepares an SPV profile for statistical analysis.  The first
% step is time-shifting and stripping out extra data to leave
% one minute per-rotatory and one minute post-rotatory.  The
% second step is outlier detection.  The third step is decimation
% by a factor of 30 down to 4 Hz.
%
% Ground version by D. Balkwill
% Modified for analyzing the KC-135 data by Chris Pouliot

sample = 120;

eval(['load ',run_code,'.aspvh']);
eval(['load ',run_code,'.parms']);
eval(['load ',run_code,'.tach']);

tachom = tach;
clear tach;

tachom = tachom * 0.03;

y = aspvh;
clear aspvh

runlen = runlen*sample;
delay = delay*sample ;
spinl = spinl*sample;

t = ([1:(runlen+1)] ./ sample);
good_data = ones(1,runlen+1);

%
%find valid range for log outlier detection in per-rotatory section
%

fprintf('Per-rotatory:\n');
i = delay;
j = i + 1*sample;
condition = 1;
while ((abs(mean(y(j:j+5*sample))) > 10) & (condition == 1))
        if ((j+7*sample) >  (delay+spinl)),
                condition = 0;
        else
                j = j + 2 * sample;
        end
end
if condition == 0,
        j = j + 2 * sample;
end
t1 = t(i:j);
y1 = y(i:j);

[logout1,under,over,m,b] = log_outlier(t1,y1);

%take care of under- or over-flow
good_data(i:j) = ~logout1;
if (under > 0)
```

```
        good_data(i-under:i-1) = zeros(1,under);
end
if (over > 0)
        good_data(j+1:j+over) = zeros(1,over);
end

%save final fit in parms file as first-order "model fit"
tau1 = -1/m;
gain1 = exp(b)/120;
fprintf('Time length of outliers from log fit is ');
fprintf('%5.2f seconds.\n',(sum(logout1)+under+over)/sample);

% do magnitude outlier detection on remainder of per-rotatory SPV

i = j + over + 1;
[magout1,under,over] = mag_outlier(t((i:spinl+delay)),y(i:(spinl+delay)),30);
good_data(i:(spinl+delay)) = ~magout1;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
fprintf('Time length of outliers from magnitude threshold is');
fprintf(' %5.2f seconds.\n',(sum(magout1)+under)/sample);

fprintf('Post-rotatory:\n');
i = delay + spinl + sample + 1; %one second after stop
j = i + 1*sample;
condition = 1;
while ((abs(mean(y(j:j+5*sample))) > 10) & (condition == 1))
        if ((j+7*sample) >  runlen),
                condition = 0;
        else
                j = j + 2 * sample;
        end
end
if condition == 0,
        j = j + 2 * sample;
end
t2 = t(i:j);
y2 = y(i:j);

[logout2,under,over,m,b] = log_outlier(t2,y2);

%take care of under- or over-flow
good_data(i:j) = ~logout2;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
if (over > 0)
        good_data(j+1:j+over) = zeros(1,over);
end

%save final fit in parms file as first-order "model fit"
tau2 = -1/m;
gain2 = exp(b + 60 * m)/120;
fprintf('Time length of outliers from log fit is ');
fprintf('%5.2f seconds.\n',(sum(logout2)+under+over)/sample);

% do magnitude outlier detection on remainder of per-rotatory SPV
```

```
i = j + over + 1;
[magout2,under,over] = mag_outlier(t(i:runlen),y(i:runlen),30);
good_data(i:runlen) = ~magout2;
if (under > 0)
        good_data(i-under:i-1) = zeros(1,under);
end
fprintf('Time length of outliers from magnitude threshold');
fprintf(' is %5.2f seconds.\n',(sum(magout2)+under)/sample);
% don't fill in any overflow, because this would be past two minutes

fprintf('Overall percentage of good data is %6.2f\n',100*mean(good_data));

%plot data and outlying regions
hold off
plot(t1,log(abs(y1)))
hold on
plot(t1,log(abs(mean(y1))) * logout1,'b--')
grid
xlabel('Time (sec)')
ylabel('ln(SPV)')
title([run_code,' -- SPV and log outlier indicator'])

hold off
plot(t2,log(abs(y2)))
hold on
plot(t2,log(abs(mean(y2))) * logout2,'b--')
grid
xlabel('Time (sec)')
ylabel('ln(SPV)')
title([run_code,' -- SPV and log outlier indicator'])

hold off
plot(t,y)
hold on
plot(t,50*(~good_data),'b--')
plot(t,-50*(~good_data),'b--')
grid
xlabel('Time (sec)')
ylabel('SPV (deg/sec)')
title([run_code,' -- SPV and overall outlier indicator'])

hold off
plot(t(good_data),y(good_data),'.')
grid
xlabel('Time (sec)')
ylabel('SPV (deg/sec)')
title([run_code,' -- good SPV (outliers removed)']);

hold off

clear t t1 t2 y1 y2 i j logout1 logout2 minute_size sample
clear delay spinl spinv T1 T2 runlen under over magout1 magout2
clear m b tau1 gain1 tau2 gain2

norm_spv = y;

clear y
```

```
% save normalized data, having departed from 'file_specs' at this point

eval(['save ',run_code,'.good good_data']);
eval(['save ',run_code,'.norm norm_spv']);

%decimate data to 4 Hz, and save decimated information
kc_dec30

clear good_data norm_spv
clear t tau1 tau2 run_len spinl spinv
```

```
%KC_tachan
% Program which analyzes the tach signal to determine a set number of parameters
% Ground version by D. Balkwill and C. Pouliot
% Modifed by C. Pouliot to analyze KC-135 data

%load tach data into standardized variable

eval(['load ',run_code,'.tach']);
TACH = tach;
clear tach;

TACH = TACH * .03;

sample = 120;
steady_vel = 120;

l = length(TACH);
t = [0:l-1] / sample;

event = delta_tach(TACH,1);
while (abs(TACH(event+120)) < steady_vel/2)
        event = delta_tach(TACH,event+1);
end
start = event;

event = delta_tach(TACH,start + 500);
while (abs(TACH(event+120)) > steady_vel/2)
         event = delta_tach(TACH,event+10);
end
stop = event;
avg = mean(TACH(start + 3 * sample:stop - 5));

spinv = avg;
delay = (start - 1) / sample;
spinl = (stop - start) / sample;
runlen = t(l);

fprintf('Delay = %6.2f seconds.\n',delay);
fprintf('Spin length = %6.2f seconds.\n',spinl);
fprintf('Spin velocity = %6.2f deg/sec.\n',spinv);
fprintf('Run length = %6.2f seconds.\n',runlen);

clear event start stopsample avg steady_vel flag level TACH

eval(['save ',run_code,'.parms spinv delay spinl runlen']);

hold off
plot(t,TACH)
hold on
plot([delay,delay],[-1500,1500],'w')
plot([spinl+delay,spinl+delay],[-1500,1500],'w')
plot([delay,delay+spinl],[spinv,spinv],'g')
hold off
```

```
% KC_three_point

function [scale,noise,offset] = KC_three_point(t,pos,colour)

% This script inputs the angular deviations from the user, and
% calls the 'pick_regions' script so that the fixation regions
% can be selected.  For each deviation, the average value over
% all of the regions is calculated.  The calibration factor for
% each axis is calculated as the angular difference between the
% positive and negative deviations (in degrees), divided by the
% difference between mean positive and negative deviations (in
% arbitrary units).  The calculated mean values are displayed
% to the user graphically as dotted lines, for inspection.
%
% If a zero fixation point is specified, it is used to
% determine the offset value.
%
% The noise is estimated by taking the root-mean-square value of
% the fluctuations about all selected regions.
%
% Ground version by D. Balkwill 11/27/90
% Modified by Chris Pouliot to analyze KC-135 data

if isempty(pos_deg)
        pos_deg = 10;
end

if isempty(neg_deg)
        neg_deg = -10;
end

if (pos_deg == neg_deg)
        disp('Calibration range cannot be zero, Symmetrical calibration assumed.');
        neg_deg = -1 * pos_deg;
end

% select positive trace deflection regions
disp('');
disp('Use mouse to select flat-top regions of positive trace deflection.');
pos_regions = pick_regions(t,pos,colour);
[mp,np] = size(pos_regions);
if (mp == 0)
        error('No positive trace deflection flat-top regions, Cannot calibrate.');
end
for i=1:mp
        xpos = [xpos ; pos(pos_regions(i,1):pos_regions(i,2))];
end
mpos = mean(xpos);
hold on
plot([t(1),t(length(t))],[mpos,mpos],'r:');
hold off

% select negative trace deflection regions
disp('Now select flat-top regions of negative trace deflection.');
neg_regions = pick_regions(t,pos,colour);
[mn,nn] = size(neg_regions);
if (mn == 0)
        error('Cannot calibrate, no negative trace deflections selected.');
```

```
end
m = min([mp mn]);
if (mp<mn)
        fprintf('Fewer positive regions chosen,  fitting all paired regions.');
end
if (mn<mp)
        fprintf('Fewer negative regions chosen,  fitting all paired regions.');
end
for i=1:mn
        xneg = [xneg ; pos(neg_regions(i,1):neg_regions(i,2))];
end
mneg = mean(xneg);
hold on
if (colour == 'y')
        plot([t(1),t(length(t))],[mneg,mneg],'r:');
else
        plot([t(1),t(length(t))],[mneg,mneg],':');
end
hold off

% calculate scale factor in deg/unit
scale = (pos_deg - neg_deg) ./ (mpos - mneg);

% select optional zero trace deflection regions
if isempty(y)
        y='n';
end
xzero = [];
if y=='y' | y=='Y'
        disp('Select regions in which subject fixated on zero reference.');
        zero_regions = pick_regions(t,pos,colour);
        [m,n] = size(zero_regions);
        if (m == 0)
                offset = 0;
        else
                for i=1:m
                        xzero = [xzero ; pos(zero_regions(i,1):zero_regions(i,2))];
                end
                offset = mean(xzero);
                hold on
                if (colour == 'y')
                        plot([t(1),t(length(t))],[offset,offset],'r:');
                else
                        plot([t(1),t(length(t))],[offset,offset],':');
                end
                hold off
        end
else
        offset = 0;
end

% display positive and negative mean values
hold on
if (colour == 'y')
        plot([t(1),t(length(t))],[mneg,mneg],'r:');
        plot([t(1),t(length(t))],[mpos,mpos],'r:');
else
        plot([t(1),t(length(t))],[mneg,mneg],':');
```

```
          plot([t(1),t(length(t))],[mpos,mpos],':');
end
hold off

% estimate noise, in rms degrees
xpos = xpos - mean(xpos);
xneg = xneg - mean(xneg);
if (isempty(xzero) == 0)
          xzero = xzero - mean(xzero);
end
x = [xpos ; xneg ; xzero ];
noise = scale * sqrt(mean(x.*x))

clear pos_regions neg_regions zero_regions xpos xneg xzero x
clear i m n y neg_deg pos_deg
return;
```