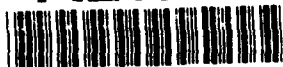


AD-A282 909



①

**NAVAL POSTGRADUATE SCHOOL**  
**Monterey, California**



**DTIC**  
**ELECTE**  
**AUG 03 1994**  
**S B D**

**THESIS**

**NPS STATE VECTOR ANALYSIS AND RELATIVE MOTION  
PLOTING SOFTWARE FOR STS-51**

by  
**Lieutenant Lee A. Barker**

**March, 1994**

**Thesis Advisor:**

**Dr. Rudolf Panholzer**

**Approved for public release; distribution is unlimited.**

**94-24395**



13918

**DTIC QUALITY INSPECTED 1.**

**94 8 02 070**

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1994.	3. REPORT TYPE AND DATES COVERED Engineer's Thesis		
4. TITLE AND SUBTITLE NPS STATE VECTOR ANALYSIS AND RELATIVE MOTION PLOTTING SOFTWARE FOR STS-51		5. FUNDING NUMBERS		
6. AUTHOR(S) Lee A. Barker				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE A		
13. ABSTRACT (maximum 200 words) This thesis outlines the objectives, structure, and operation of the software designed to meet requirements set forth by Development Test Objective 700-6 for the space shuttle Discovery mission STS-51. The primary goals were the comparison of state vector information produced by GPS sources and Discovery's inertial navigation computer, and the real-time display of relative position and rendezvous information between Discovery and a retrievable shuttle pallet satellite. In-flight and post-flight examination of GPS and inertial state vectors provided the first step in the development of GPS as an on-orbit navigation system. Analysis of the Orbiter and target satellite state vectors produced real-time graphical displays of operationally significant data to the Discovery's flight crew during rendezvous and proximity operations.				
14. SUBJECT TERMS STS-51 DTO 700-6, State Vector Comparison, Relative Motion Plotting, Spacecraft Rendezvous and Proximity Operations			15. NUMBER OF PAGES 634	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

NPS STATE VECTOR ANALYSIS AND RELATIVE MOTION  
PLOTING SOFTWARE FOR STS-51

by

Lee A Barker  
Lieutenant , United States Navy  
B.S., U. S. Naval Academy, 1984

Submitted in partial fulfillment  
of the requirements for the degree of

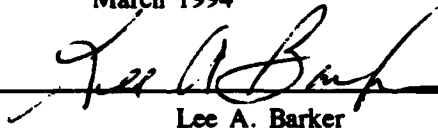
MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING  
and  
AERONAUTICAL AND ASTRONAUTICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL

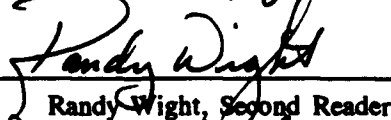
March 1994

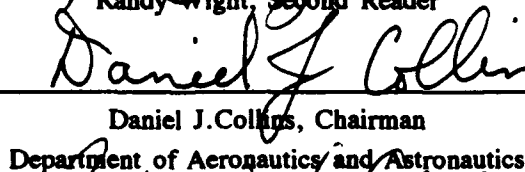
Author:

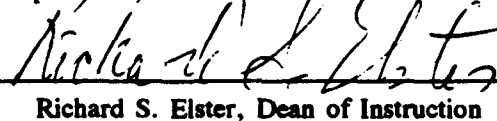
  
Lee A. Barker

Approved by:

  
Rudolf Panholzer, Thesis Advisor

  
Randy Wight, Second Reader

  
Daniel J. Collins, Chairman  
Department of Aeronautics and Astronautics

  
Richard S. Elster, Dean of Instruction

### ABSTRACT

This thesis outlines the objectives, structure, and operation of the software designed to meet requirements set forth by Development Test Objective 700-6 for the space shuttle Discovery mission STS-51. The primary goals were the comparison of state vector information produced by GPS sources and Discovery's inertial navigation computer, and the real-time display of relative position and rendezvous information between Discovery and a retrievable shuttle pallet satellite. In-flight and post-flight examination of GPS and inertial state vectors provided the first step in the development of GPS as an on-orbit navigation system. Analysis of the Orbiter and target satellite state vectors produced real-time graphical displays of operationally significant data to the Discovery's flight crew during rendezvous and proximity operations.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist.	Avail and/or Special
A-1	

## TABLE OF CONTENTS

<b>I. INTRODUCTION</b>	<b>1</b>
A. STS-51 DTO 700-6 MISSION OVERVIEW	1
B. PURPOSE OF DEVELOPMENT TEST OBJECTIVE 700-6	3
C. NPS SOFTWARE OBJECTIVES	4
D. SUMMARY	5
 <b>II. DEVELOPMENT TEST OBJECTIVE 700-6 DESCRIPTION</b>	 <b>7</b>
A. DEVELOPMENT TEST OBJECTIVE 700-6 OBJECTIVES	7
B. DTO 700-6 COMPONENT DESCRIPTIONS	8
1. Hardware Components	8
a. Pulse-Code Modulation Master Unit (PCMMU)	8
b. GRID 1535 Portable GRID Systems Computer (PGSC)	8
c. GRID 1530 Portable GRID Systems Computer (PGSC)	9
d. TANS Quadrex GPS Receiver	9
e. ORFEUS/SPAS	9
2. Software Components	10
a. PC Decommutator (PCDecom)	10
b. TANS Software	10
c. NPS Software	10
 <b>III. RELATIVE MOTION AND PLOTTING</b>	 <b>13</b>
A. RELATIVE MOTION PLOTS (R-BAR/V-BAR)	13

B.	SINGLE IMPULSE, IN PLANE, ORBITAL MANEUVERS . . . . .	15
1.	Posigrade Burn . . . . .	15
2.	Retrograde Burn . . . . .	16
3.	Radial Burn . . . . .	16
C.	RENDEZVOUS MANEUVER SEQUENCE . . . . .	18
1.	Phase Adjustment Maneuvers . . . . .	18
2.	Corrective Combination Maneuvers . . . . .	18
3.	Terminal Initiation . . . . .	19
4.	Midcourse Correction . . . . .	19
D.	PROXIMITY OPERATIONS . . . . .	19
1.	Station Keeping . . . . .	19
2.	Separation Maneuvers . . . . .	20
3.	Approach To Target . . . . .	20
IV.	MISSION DESCRIPTION . . . . .	23
A.	MISSION OVERVIEW . . . . .	23
B.	STS-51 MISSION TIME LINE . . . . .	23
V.	NPS SOFTWARE ARCHITECTURE . . . . .	27
A.	NPS SOFTWARE DESIGN PHILOSOPHY . . . . .	27
B.	DATA STRUCTURES . . . . .	27
1.	Vector Structure . . . . .	29
2.	State_vector Structure . . . . .	29
3.	RV_vector Structure . . . . .	29
4.	Diff_vector Structure . . . . .	29
5.	Quaternion Structure . . . . .	30

6.	Packet Structures . . . . .	30
7.	Nps_state_vector_info Structure . . . . .	31
8.	Nps_quaternion_info Structure . . . . .	31
9.	Gravity_data Structure . . . . .	31
10.	Gravity_model_data Structure . . . . .	32
11.	Drag_model_data Structure . . . . .	32
12.	Perturbations Structure . . . . .	32
13.	Nps_predictor_thrust Structure . . . . .	32
14.	Nps_predictor Structure . . . . .	33
15.	Nps_graphics_info Structure . . . . .	33
16.	Nps_plot_axis . . . . .	33
17.	Nps_plot_info Structure . . . . .	33
18.	Nps_rvplot_info . . . . .	34
19.	Nps_diffplot_info Structure . . . . .	34
20.	Nps_pyplot_info Structure . . . . .	35
21.	Nps_filter_info Structure . . . . .	35
22.	Time_dhms Structure . . . . .	35
23.	Edit_label Structure . . . . .	35
24.	Edit_field_info Structure . . . . .	35
25.	Edit_field Structure . . . . .	36
26.	Current_field Structure . . . . .	36
27.	Edit_screen Structure . . . . .	36
C.	PROGRAM ARCHITECTURE . . . . .	36
1.	The 'Main' Function . . . . .	37
2.	Keyboard Control . . . . .	38
D.	NPS SOFTWARE FILE DIRECTORY ORGANIZATION . . . . .	40

<b>VI. PROGRAM FUNCTIONS</b>	<b>43</b>
A. VECTOR/MATRIX FUNCTION LIBRARY	43
B. COWELL PROPAGATOR	45
1. M-50 to WGS-84/WGS-84 to M-50 Converters	46
2. GEM-9 Gravity Model	46
3. Jacchia Atmospheric Model	47
C. F AND G FUNCTION PROPAGATOR	48
D. PREDICTOR FUNCTIONS	49
E. PLOT/DATA FUNCTIONS	51
F. GPS FILTER	54
G. EDIT FUNCTIONS	57
<b>VII. SOFTWARE OPERATION</b>	<b>59</b>
A. PROGRAM INPUT	59
B. DATA OPERATIONS	60
C. PROGRAM OUTPUT	62
1. R-bar/V-bar Plots	62
2. Sawtooth Plots	63
3. Pitch/Yaw Displays	65
<b>VIII. DATA COLLECTION AND ANALYSIS</b>	<b>67</b>
A. GPS STATE VECTOR ANALYSIS	68
1. Orbiter INS vs TANS GPS	69
2. Orbiter Target vs SPAS GPS	70
B. GPS RELATIVE NAVIGATION	72
C. RELATIVE MOTION PLOTTING RENDEZVOUS AID	72



<b>IX. LESSONS LEARNED</b>	<b>77</b>
A. 'GIGO' (GARBAGE IN, GARBAGE OUT): GPS VELOCITY ACCURACY	77
B. MORE 'GIGO' OR 'NINO' (NOTHING IN, NOTHING OUT)	78
C. THE GPS FILTER	79
D. TDRSS COMMUNICATIONS GAPS	80
E. A POST-FLIGHT DEBRIEF/TRAINING TOOL	81
F. GROUND CONTROLLERS AID	81
 <b>X. CONCLUSIONS</b>	 <b>83</b>
A. Flight Crew Remarks	83
B. Future NPS Software Development	84
 <b>REFERENCES</b>	 <b>87</b>
 <b>APPENDIX A - DEFINITIONS AND ABBREVIATIONS</b>	 <b>89</b>
 <b>APPENDIX B - NPS SOFTWARE USER MANUAL</b>	 <b>94</b>
 <b>DISTRIBUTION LIST</b>	 <b>127</b>
 <b>APPENDIX C - NPS SOFTWARE SOURCE CODE (Bound Separately)</b>	 <b>130</b>

## ACKNOWLEDGEMENTS

This thesis and the experiment on which it is based is the result of untold hours of dedicated effort by many talented individuals. Project completion would not have been possible without the other members of the NPS "Cheap Labor" team, particularly the countless derivations of LT Les Makepeace, graphics programming research by LT Carolyn Tyler, and the never ending "search and destroy" research efforts of LT Steve Rewald. Thanks to NASA's Penny Saunders for providing us the requisite GPS hardware and software background. A special thanks to Tom Silva, 'C' programmer extraordinaire, who stayed up almost every night into the early morning hours for four weeks straight to guide us through the most intense 'C' programming party imaginable. Many thanks to NASA Mission Specialist Jim Newman who provided the opportunity to work on such an exciting project, to space shuttle Discovery mission commander Frank Culbertson, Space Shuttle pilot Bill Readdy, Mission Specialists Dan Bursch and Carl Waltz for their expertise, insight, and faith in the work of four graduate students, and to all the astronauts and staff at Johnson Space Center for their support and hospitality during the six months spent developing the flight software.

## **I. INTRODUCTION**

On September 12, 1993, space shuttle Discovery mission STS-51 successfully launched into orbit carrying several pieces of hardware and software in support of Development Test Objective (DTO) 700-6. The purpose of DTO 700-6 was to investigate the use of the Global Positioning System (GPS) for shuttle Orbiter navigation and Orbiter/satellite relative navigation.

Components of the experiment included (1) the shuttle Orbiter's navigation computer and pulse code modulation master unit (PCMMU) containing Orbiter navigation data and telemetry, (2) a Trimble Advanced Navigation Sensor (TANS) QUADREX GPS receiver for gathering GPS state vector information, (3) a GRID 1535 Portable GRID Systems Computer (PGSC) with software for stripping and sending desired data packets from the Orbiter's 128 Kbyte data stream used by the PCMMU to (4) a second GRID 1530 PGSC with Naval Postgraduate School (NPS) and TANS GPS software packages for analyzing and storing the data, and (5) the German ORFEUS/SPAS satellite with an onboard GPS receiver.

### **A. STS-51 DTO 700-6 MISSION OVERVIEW**

The STS-51 mission was the first space shuttle flight to successfully carry and operate a TANS GPS receiver in space.

This provided the unique science opportunity of collecting GPS state vector data and comparing this data with Orbiter state vector data generated by ground tracking and propagated by the Orbiters navigation computer to determine the reliability of GPS as an on-orbit navigation aid. Additionally, the space shuttle Discovery deployed and retrieved a shuttle pallet satellite (SPAS) also carrying a GPS receiver. This presented the opportunity to compare ground tracking state vectors with SPAS GPS state vectors as well as evaluating TANS GPS and SPAS GPS relative navigation during rendezvous and proximity operations.

The TANS QUADREX GPS state vector data was compared with the Orbiter inertial navigation system (INS) state vectors propagated by the Orbiter's navigation computer. A comparison of the states would demonstrate the feasibility of using GPS data to update the Orbiter navigation system. The Orbiter INS state vector is presently updated by data uplink from earth based tracking stations approximately every five to eight hours to correct for inertial system drift. Given reliable GPS state vector information, it was expected that by comparing the Orbiter INS state vector with the TANS GPS state vector, the Orbiter inertial position vector drift away from the Orbiter TANS GPS position vector over time would be observed. On an INS update, it was predicted that the inertial position would return to some position close to the TANS GPS position

vector, lending some credibility to the concept of using GPS for navigational updates.

The STS-51 mission profile called for releasing the ORFEUS/SPAS satellite on flight day two, maneuvering the Orbiter away from the satellite for several days to allow data collection, then a rendezvous to capture SPAS on flight day nine prior to de-orbiting. ORFEUS/SPAS state vector information was available via data link of it's onboard GPS receiver data, and by ground tracking and uplink of a target state vector. Orbiter and SPAS state vectors were then available on the 128 Kbyte data stream sent from the PCMMU to the GRID 1535. The NPS software tools contained in the PGSC received data packets from the GRID 1535 and the TANS GPS unit to provide real-time INS/GPS difference plots, several real-time relative motion displays, and relative position prediction displays that normally would require ground tracking information and manual plotting. These tools lay the foundation for the use of GPS as an aid to on orbit navigation and spacecraft rendezvous and docking. Lessons learned will be useful for future shuttle missions and essential to the development and operation of a manned space station.

#### **B. PURPOSE OF DEVELOPMENT TEST OBJECTIVE 700-6**

The DTO 700-6 objectives were defined by the DTO software plan published by the STS-51 DTO 700-6 manager. The DTO software was divided into four levels in order of priority.

The Level I DTO objectives as stated in the software plan were to operate the TANS GPS receiver, display data on the PGSC CRT, and store state vectors and other GPS data packets in files for post flight analysis (PFA).

The Level II DTO code was to visually display the relative difference of the Orbiter INS state vector and the Orbiter TANS GPS state vector in real time. If possible, Level II code was also to perform a similar comparison between the target vector for SPAS and the SPAS GPS. State vector inputs to the program were to be manually or automatically input. Level II DTO code is contained in the NPS software.

The Level III/IV DTO code objectives included using state vector information supplied by the Orbiter, GPS and ground tracking to provide real-time relative motion plotting of the ORFEUS/SPAS satellite and the Orbiter. Level III/IV DTO code is also contained in the NPS software.

### **C. NPS SOFTWARE OBJECTIVES**

The NPS software package written for STS-51 was designed to meet the requirements set forth in the DTO 700-6 software plan. This was accomplished through the use of state vector difference plots (Level II) and relative position plots (Level III/IV). In addition, the NPS software was designed to provide target locating data in the form of pitch/yaw/distance information, as well as predicted motion plotting to aide in rendezvous and proximity operations. These tools, while

inspired by the GPS DTO, do not rely on GPS to provide navigation solutions. The NPS software only requires target and Orbiter state vectors. Therefore, any state vector source (INS, GPS, ground targeting, approach radar, or a manual keyboard entry) may be used to provide an automated situational awareness tool to the flight crew.

#### **D. SUMMARY**

The scope of this thesis was to develop the Naval Postgraduate School State Vector Analysis and Relative Motion Plotting (NPS) software residing in the GRID 1530 PGSC, integrate that software with existing software and hardware to accomplish the DTO 700-6 objectives, support the STS-51 mission during training and actual flight, and perform post-flight analysis of the collected data.

Testing of the NPS software was accomplished on a desktop computer (PC) with software emulation of the flight inputs. Testing of the complete integrated flight software package (w/o TANS GPS data) and related DTO components was performed in the space shuttle integrated flight simulator at Johnson Space Center prior to flight aboard STS-51.

The NPS software was written in the 'C' programming language using Borland C++. Several proven 'off the shelf' software tools were integrated with tools developed as a part of this project. 'Off the shelf' tools included a proven Cowell orbit propagator, a GPS state vector filter, and

communications software. Developed software included interface routines, data processing and plotting routines, and rendezvous aids.

Details of the NPS software, its role in the STS-51 mission and DTO 700-6, code development, and post flight data analysis are contained within this thesis. Chapter II contains a description of DTO 700-6, including hardware and software components. Chapter III contains an introduction to spacecraft relative motion and plotting. Chapter IV contains an overview of the STS-51 mission as it relates to DTO 700-6. Chapter V contains a detailed description of the NPS software structure and organization. Chapter VI contains a detailed description of major software functions. Chapter VII contains a description of the NPS program flow and operation. Chapter VIII discusses data collection and analysis. Chapter IX discusses lessons learned. Chapter X contains results and conclusions. A glossary of terms, software user's manual, and the source code are contained as appendices.



## **II. DEVELOPMENT TEST OBJECTIVE 700-6 DESCRIPTION**

### **A. DEVELOPMENT TEST OBJECTIVE 700-6 OBJECTIVES**

Generally stated, the purpose of DTO 700-6 is the investigation of the use of GPS as a source for on orbit inertial navigation system update and relative motion determination between the space shuttle and a target satellite. The goals of DTO 700-6 are described in detail by the objectives of each of the four levels of the DTO software plan.

- Level I - Operate the TANS QUADREX GPS receiver. Display data on the CRT and store state vectors and engineering data in files.
- Level II - Input the Orbiter state vector by hand or automatically to perform TANS GPS vs Orbiter state vector comparison. If available, perform Orbiter target vector vs SPAS GPS state vector comparison.
- Level III - Use Orbiter GPS and SPAS GPS state vectors in a rendezvous display program to evaluate relative GPS solution with ground and onboard solution.
- Level IV - Integrate the Orbiter GPS and inertial state vectors, the SPAS GPS and target state vectors, and Orbiter attitude data into one program showing the entire rendezvous and proximity operations profile.

Level I objectives were met by TANS GPS software written by Mike Arnie of the Lockheed Engineering and Sciences Corporation for NASA. The TANS GPS software runs in the GRID 1530 PGSC and accepts input from the TANS GPS unit via an RS-422 cable and port.

Level II, III, and IV objectives were achieved by the NPS software running co-resident in the GRID 1530 PGSC. The NPS software accepted TANS GPS state vector information from the TANS GPS software and all other necessary information from the Orbiter's computer via the GRID 1535 PGSC and an RS-232 port.

## **B. DTO 700-6 COMPONENT DESCRIPTIONS**

### **1. Hardware Components**

#### ***a. Pulse-Code Modulation Master Unit (PCMMU)***

The PCMMU was the source for the 128 Kbyte data link information. This data stream contains up-link and down-link telemetry used by the Orbiter and NASA flight controllers. Information contained in the telemetry included Orbiter and SPAS target state vectors, Orbiter attitude information in the form of a quaternion, and SPAS telemetry including SPAS GPS state vectors.

#### ***b. GRID 1535 Portable GRID Systems Computer (PGSC)***

The GRID 1535 laptop computer was the interface between the PCMMU and the GRID 1530 PGSC. It was linked to the PCMMU through a 128 Kbyte data link cable. The GRID 1535 contained the PC Decommutator (PCDecom) software package. PCDecom transmitted data packets from the GRID 1535 PGSC to the GRID 1530 PGSC via the RS-232 communications port for use by the NPS software.

**c. GRID 1530 Portable GRID Systems Computer (PGSC)**

The GRID 1530 PGSC is a portable computer with a 10 MHz 80386 microprocessor, 8 Mbytes RAM, and 40 Mbyte internal hard drive. The GRID 1530 PGSC contained the NPS software and the TANS GPS software. It received state vector data from the GRID 1535 via the RS-232 port and from the TANS QUADREX GPS receiver via the RS-422 port.

**d. TANS Quadrex GPS Receiver**

The Trimble Advanced Navigation Sensor (TANS) is a six channel GPS receiver which provides position, velocity, time and other information to external data terminals. The TANS GPS receiver was connected to the GRID 1530 PGSC via the RS-422 port. Communications and control of the TANS unit was handled by the TANS GPS software in the GRID 1530 PGSC. The TANS GPS receiver has three antennae that were velcro mounted in the Orbiter windows on Flight Day one of the mission. This antenna arrangement made TANS GPS reception a function of the Orbiter's attitude. For further information on the TANS GPS receiver, see [Ref. 1].

**e. ORFEUS/SPAS**

The Orbiting Retrievable Far and Extreme Ultraviolet Spectrometer (ORFEUS) housed onboard the German built Shuttle Pallet Satellite (SPAS) contained a German made space qualified GPS receiver. The ORFEUS/SPAS GPS unit was used to provide state vector information in its telemetry. The

SPAS state vector information was then retrieved from the 128 Kbyte data link stream for use in the NPS software.

## **2. Software Components**

### **a. PC Deconvutator (PCDecom)**

The PC Deconvutator (PCDecom) software package written by Tom Silva of The Telemetry Workshop resided in the GRID 1535 PGSC. PCDecom is capable of reading data from the 128 Kbyte data stream, displaying selected data to the GRID 1535 PGSC display screen, and packetizing desired data for serial transmission to other users. For DTO 700-6, PCDecom transmitted data packets containing Orbiter, SPAS target, and SPAS GPS state vectors as well as the Orbiter's quaternion to the GRID 1530 via an RS-232 port for use by the NPS software.

### **b. TANS Software**

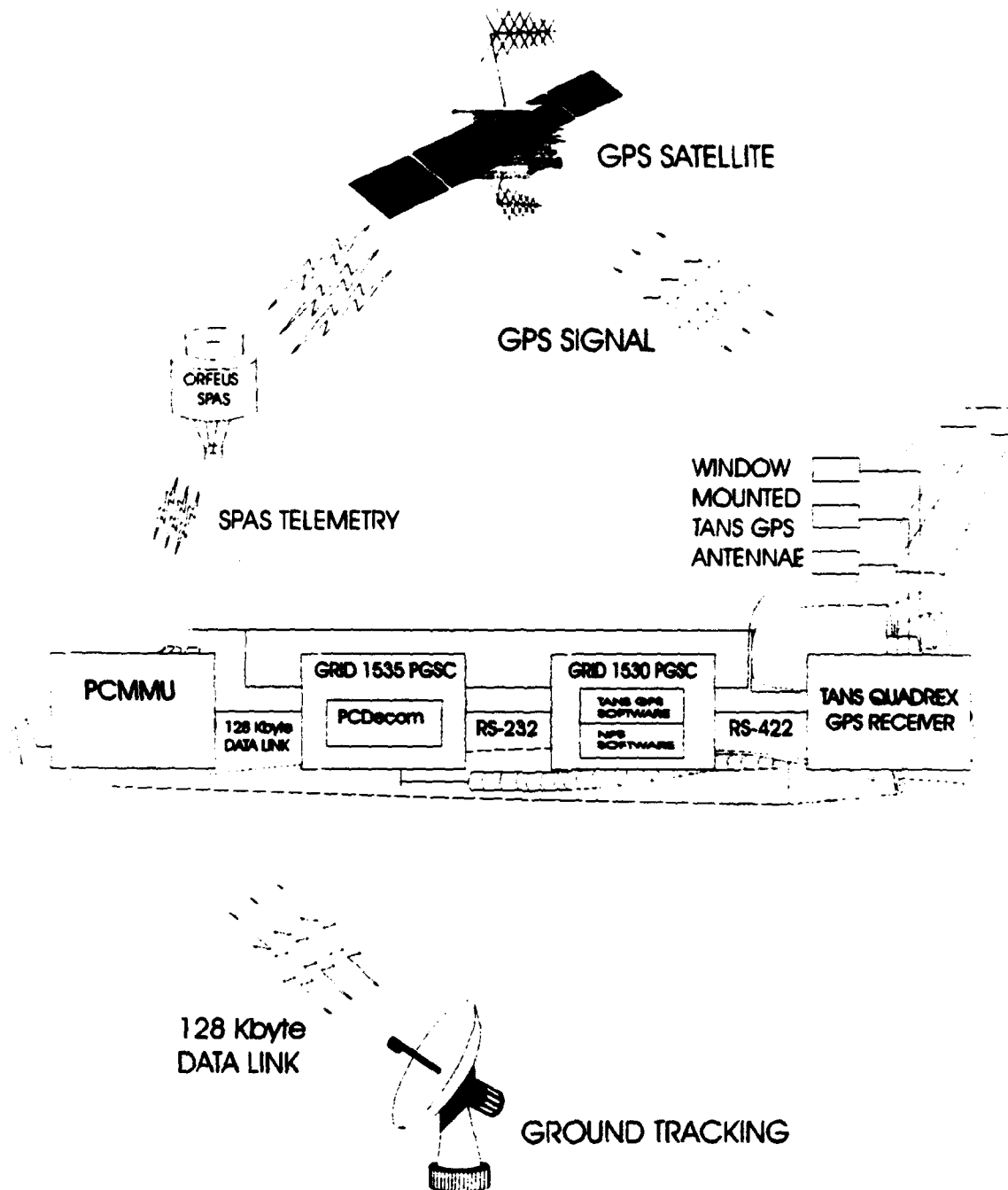
The TANS GPS software, or level I code, communicated with the TANS unit, displayed various TANS GPS data on the GRID 1530 PGSC display screen, converted the TANS GPS state vector from WGS-84 Earth-centered, Earth-fixed (ECEF) coordinates to M-50 inertial coordinates, and provided the M-50 GPS state vector data to the NPS software.

### **c. NPS Software**

The NPS software, or level II, III, and IV code, processed incoming state vectors for display on relative motion plots and state vector difference plots. The NPS software was run independently as a stand alone program when

TANS GPS data was unavailable during shuttle flight simulations prior to the launch of STS-51, and as a subordinate function to the TANS GPS level I code during the STS-51 mission. Details of the NPS software are contained in the Chapters V, VI, and VII. Further information is contained in the User's Guide in Appendix B.

The relationship between the hardware and software components of this experiment is illustrated in Figure 2-1.



**Figure 2-1. DTO Hardware/Software Components.**

### **III. RELATIVE MOTION AND PLOTTING**

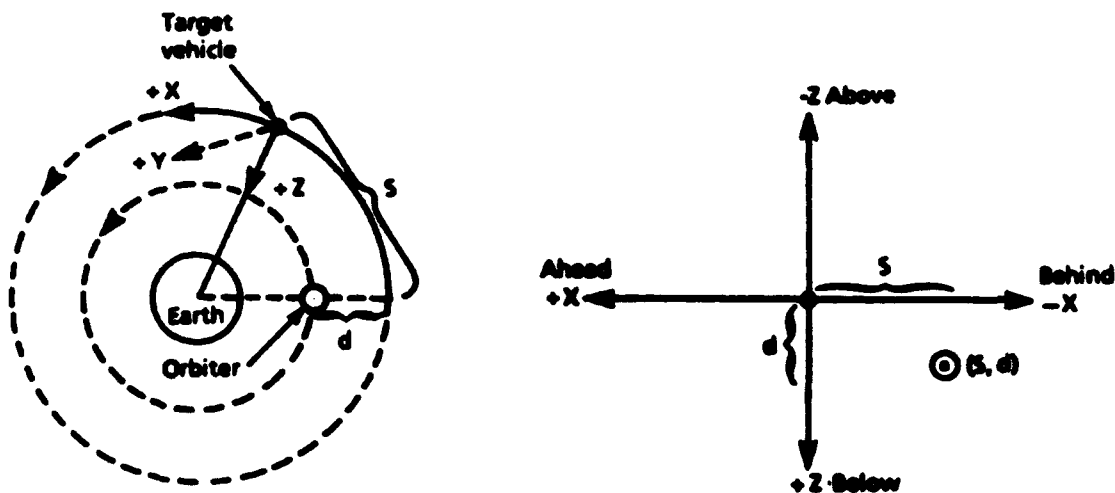
The STS-51 mission objectives included the use of the TANS GPS receiver to obtain GPS data on orbit, and the deployment and recovery of the ORFEUS/SPAS satellite. These events provided unique opportunities to test features of the NPS software relative motion plotting functions and fulfill the level III/IV objectives of DTO 700-6, specifically evaluating relative GPS navigation for satellite rendezvous and displaying in real time the rendezvous profile.

To appreciate the capabilities of the NPS software and its employment, a basic understanding of spacecraft relative motion, relative motion plotting, and Orbiter separation, rendezvous, and proximity operations is necessary. The following description, while by no means complete, serves to provide the background to understand the NPS software's use during the STS-51 mission.

#### **A. RELATIVE MOTION PLOTS (R-BAR/V-BAR)**

The concept of relative motion between a target vehicle and a chaser vehicle in space is complicated by the lack of a constant reference when viewed in an inertial frame. To simplify the presentation and explanation of relative motion, the relative motion plot uses a local vertical, circular (LVC) coordinate system. The LVC coordinate system is target-

centered. The Z-axis rotates with the target and is positive directed radially toward the Earth (R-bar). The X-axis is curvilinear and positive in the general direction of orbit motion (V-bar). The Y-axis is normal to the target's orbital plane and completes the right-hand coordinate system. The R-bar/V-bar plot displays the relative motion between a target and a chaser vehicle in LVC. This system is the one used by NASA in planning shuttle Orbiter rendezvous and proximity operations. (Figure 3-1.)



**Figure 3-1. Relative motion plot definition.**

R-bar displacement is measured positively towards the Earth and represents an altitude difference between the target and the chaser. V-bar displacement is measured positively in the target's direction of travel and represents the phase difference between the target and the chaser. Thus, if the



chaser were directly below the target on the radius vector to the Earth's center, it would appear on the +Z axis ( $\bar{R}$ ) of the relative motion plot. Similarly, if the chaser was ahead of the target on the target's orbital path and at the targets altitude, it would appear on the +X axis ( $\bar{V}$ ) on the relative motion plot.

## **B. SINGLE IMPULSE, IN PLANE, ORBITAL MANEUVERS**

The following maneuvers are the fundamental building blocks of rendezvous and proximity operations. The resulting motions described assume the Orbiter's initial position is co-located with a non-maneuvering target and in a co-planer orbit in order to provide a reference point to measure motion against.

### **1. Posigrade Burn**

A posigrade burn is one in which the thrust is in the direction of the velocity vector. A posigrade burn increases the energy of the orbit, increases the semi major axis and the angular momentum, and increases the period of the orbit. For a near circular orbit, a posigrade burn will raise every point of the orbit except the thrust point. For example, if the orbit was originally circular, a posigrade maneuver will create an elliptical orbit, with the thrust point becoming perigee and apogee occurring 180 degrees of orbit travel away. [Ref. 2: p. 1-11] Results of a posigrade burn will be separation from the target along the negative  $\bar{V}$ .

## **2. Retrograde Burn**

A retrograde burn is one in which the thrust is opposed to the direction of the velocity vector. A retrograde burn decreases the energy of the orbit, decreases the semi-major axis and the angular momentum, and decreases the period of the orbit. For a near circular orbit, a retrograde burn will lower every point of the orbit except the thrust point. For example, if the orbit was originally circular, a retrograde maneuver will create an elliptical orbit, with the thrust point becoming apogee and perigee occurring 180 degrees of orbit travel away. [Ref. 2: p. 1-15] Results of a retrograde burn will be separation from the target along the positive  $V$ -bar.

## **3. Radial Burn**

A radial burn is one in which the thrust is applied in a direction perpendicular to the spacecraft's velocity vector and in the orbital plane of the spacecraft. [Ref. 2: p. 1-18] Radial burns result in a 'fly-around' maneuver. That is, to an observer on the target, it appears as if the chaser vehicle flies in an ellipse whose path size is dependent on the magnitude of the burn and whose period is equal to the orbital period of the chaser vehicle.

Figure 3-2 illustrates the resultant relative motion of a chaser vehicle performing posigrade, retrograde, and radial burn maneuvers (1 ft/s burns) with respect to a target

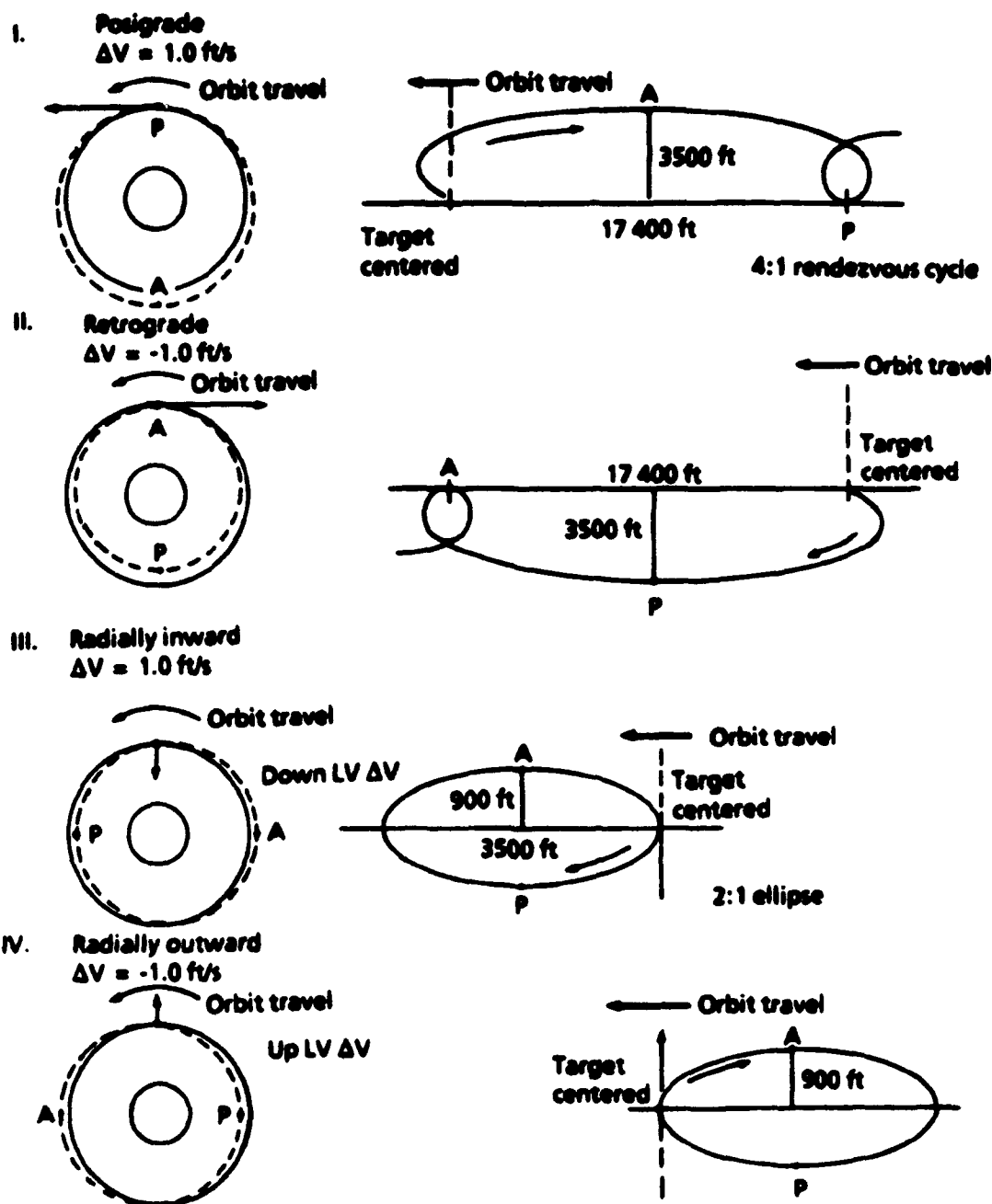


Figure 3-2. Summary of relative motion plots for 1 ft/s burns made in a circular orbit.

vehicle. Both vehicles are initially colocated and in circular orbits.

### **C. RENDEZVOUS MANEUVER SEQUENCE**

The rendezvous is the act of bringing two vehicles together. Considerations for rendezvous are numerous and include orbital parameters such as altitude, phasing, and inclination. The lighting conditions are also of importance as the rendezvous must be conducted in the light and star tracking of the target requires that it be illuminated. Depending on the mission, the rendezvous sequence can begin as early as launch time. For the STS-51 SPAS mission, rendezvous commencement was from the negative V-bar, hence only maneuvers relevant to rendezvous from near co-altitude, co-planer orbits will be discussed.

#### **1. Phase Adjustment Maneuvers**

The phase adjustment maneuver, or nominal correction (NC), is used to adjust the 'catchup' rate of the Orbiter to the target. This is performed by raising or lowering the semi-major axis through a posigrade or retrograde maneuver. The nominal correction maneuver is ground targeted.

#### **2. Corrective Combination Maneuvers**

The nominal corrective combination (NCC) maneuver is the first onboard targeted burn based on onboard sensor data such as the star tracker or rendezvous radar.

### **3. Terminal Initiation**

The terminal initiation (TI) burn alters the approach from one of phasing to one of direct intercept. TI occurs approximately one revolution prior to intercept.

### **4. Midcourse Correction**

The midcourse correction (MC) maneuvers are performed to ensure correct trajectory from TI to target intercept. Four MC burns were planned during the final revolution of the rendezvous.

The scheduled rendezvous maneuvers were designed to conclude with the Orbiter in a stable position 400 feet in front of the target on the V-bar. The final approach to the target is included in proximity operations maneuvering.

## **D. PROXIMITY OPERATIONS**

Proximity Operations (PROX OPS) consist of several types of maneuvers including close in station keeping, separation maneuvering, and final approaches to the target.

### **1. Station Keeping**

Station keeping is performed by matching the target orbital plane and altitude. When precise station keeping is not required, as is the case when the target is being held at some stand-off distance, local phasing or fly-around maneuvers can be executed to maintain station.

## **2. Separation Maneuvers**

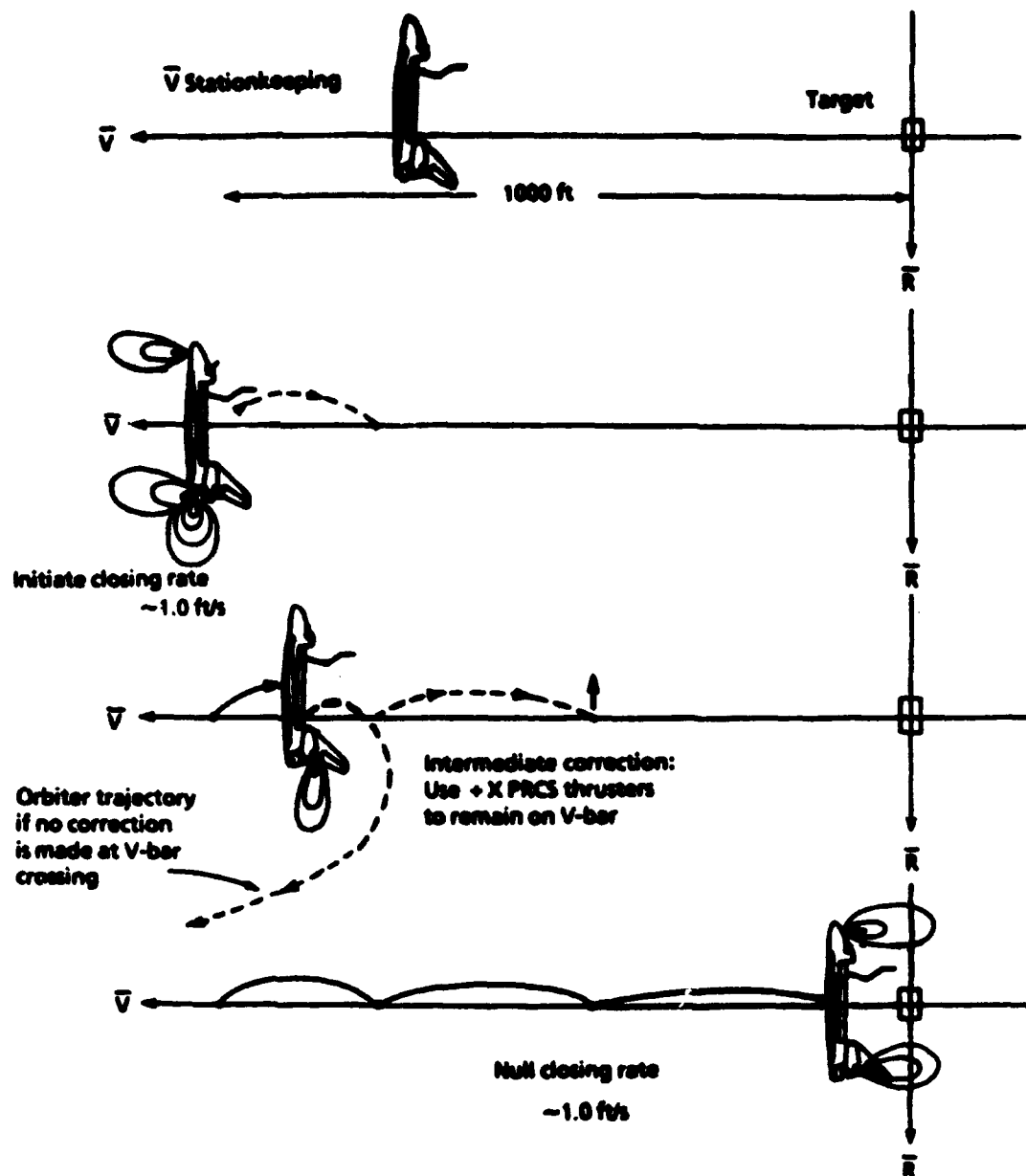
Separation from the target can be achieved by means of a posigrade or retrograde burn, a radial burn, or by a combination of these. A separation maneuver is referred to as a SEP burn.

## **3. Approach To Target**

While there is no standard rendezvous procedure applicable to all situations, three methods of approach technique have been studied with respect to operations feasibility, propellant consumption, and plume impingement. The techniques are (1) the direct approach, (2) the V-bar approach (from behind or ahead), and (3) the R-bar approach (from above or below). [Ref. 2: p. 4-10] The STS-51 mission profile called for a positive V-bar approach and an explanation of this method is in order.

The positive V-bar approach is initiated by establishing a closure rate toward the target via a combination retrograde/radial outward burn. This results in a 'hop' on the V-bar toward the target. With no further thruster inputs, the trajectory would fall below the V-bar and the Orbiter would move ahead and away from the target. As the Orbiter crosses the V-bar, a radially outward directed burn from the primary reaction control system (PRCS) raises the Orbiter slightly above the V-bar, allowing the Orbiter to slow down and close with the target further. This sequence

continues until the target is intercepted, where a braking maneuver nulls the closure rate and stabilizes the Orbiter with respect to the target. (Figure 3-3.)



**Figure 3-3. V-bar Approach.**



#### IV. MISSION DESCRIPTION

##### A. MISSION OVERVIEW

The STS-51 mission objectives included the use of the TANS GPS receiver to obtain Orbiter GPS data on orbit, and the deployment and recovery of the GPS equipped ORFEUS/SPAS satellite. These events provided unique opportunities to test features of the NPS software relative motion plotting functions and fulfill the objectives of DTO 700-6.

##### B. STS-51 MISSION TIME LINE

The STS-51 planned mission profile events pertaining to DTO 700-6 and related events such as SPAS deployment and retrieval are described in [Ref. 3] and were as follows:

**Flight day one** - The DTO hardware components (TANS GPS receiver, GRID 1530/1535 PGSC's, antennas, etc) were assembled in Discovery's flight cabin.

TANS GPS data recording was initiated.

**Flight day two** - SPAS deployment and separation was achieved. Through a series of three separation (SEP) burns,

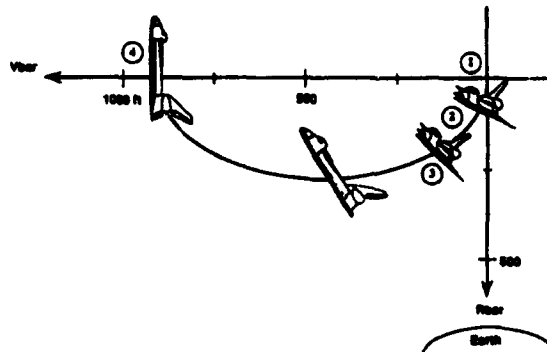


Figure 4-1. SEP 1 Maneuver

Discovery was moved away from the SPAS along the V-bar by means of posigrade and retrograde thrusts. Initial separation was accomplished at the SEP 1 (retrograde) burn. (Figure 4-1., Table 4-1.) The Orbiter arrived at the positive V-bar after one revolution. A posigrade burn was performed to null rates and allow V-bar station keeping. The SEP 2 (posigrade) carried Discovery over the top and behind SPAS. (Figure 4-2.) At V-bar crossing, the SEP 3 (retrograde) burn was performed and the Orbiter advanced to a station keeping position on the V-bar between +13 and +20 nmi. Nominal Correction (NC) 1 was performed at this time and Discovery maintained station

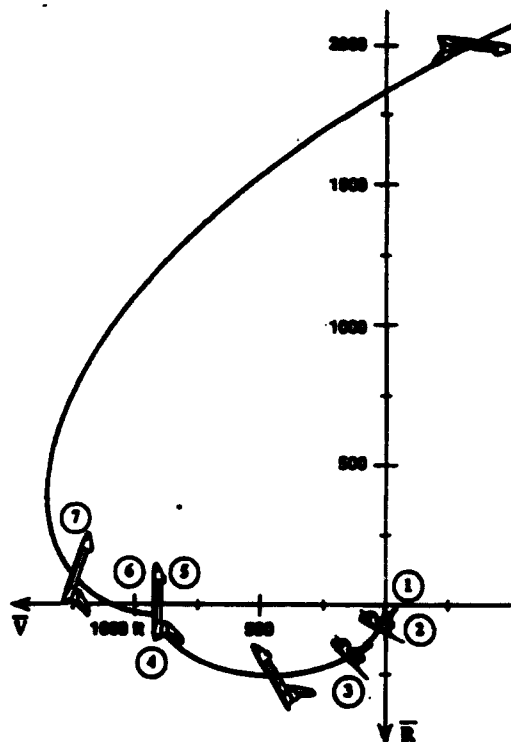
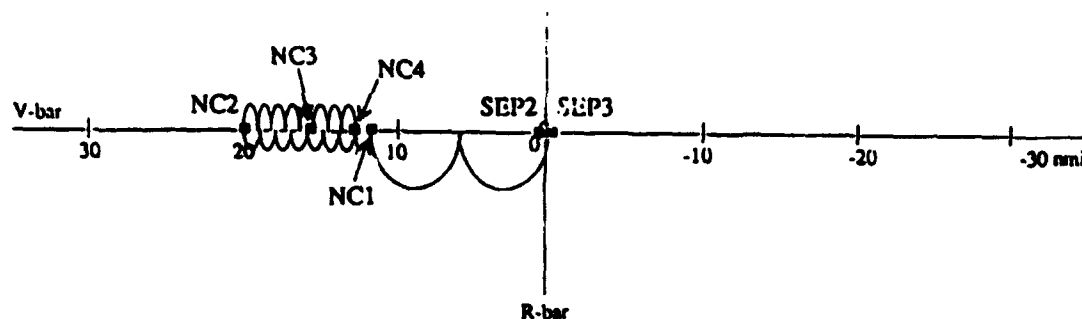


Figure 4-2. SEP 1 and SEP 2 burns

Time	Event
1	ORFEUS/SPAS Release
2	Sep 1 : 0.5 fps +Z
3	Initiate Orbiter -Z axis Target Track
4	V-bar Arrival, Null Rates
5	V-bar Stationkeeping
6	Sep 2 : 0.5 fps Posigrade

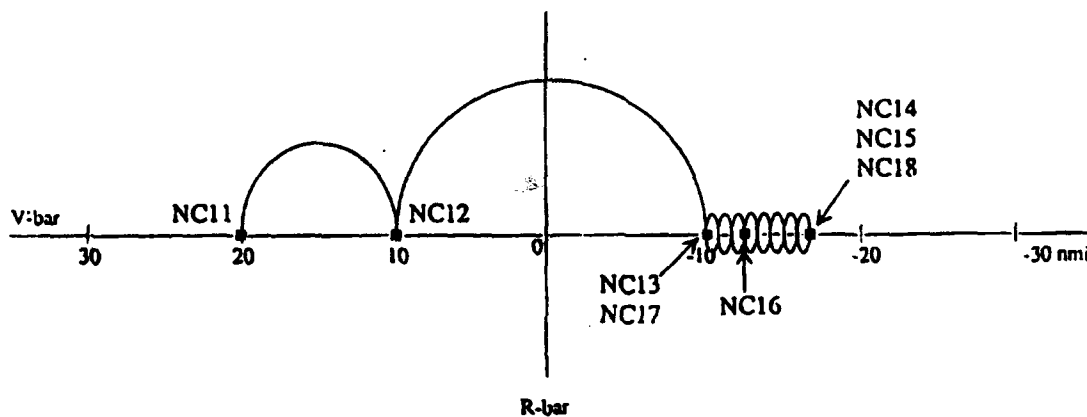
Table 4-1. ORFEUS/SPAS SEPARATION PROFILE

here through a series of phasing maneuvers until flight day six. (Figure 4-3.)



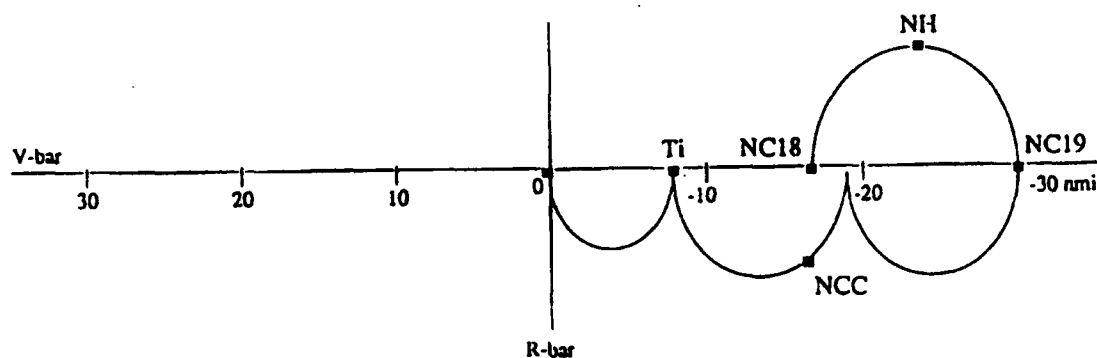
**Figure 4-3. SEP 3 maneuver to V-bar station keeping.**

**Flight day six** - NC 11 phasing maneuver (posigrade) was performed. Discovery maneuvered towards and over the top of the SPAS to take up a station keeping position on the V-bar at -10 to -17 nmi. (Figure 4-4.)



**Figure 4-4.**

**Flight day eight** - The rendezvous maneuver was performed. A posigrade phasing maneuver moved the Orbiter from station keeping to approximately -30 nmi on the V-bar. A second retrograde phasing maneuver initiated the three revolution approach to intercept. This was followed by target acquisition with the Orbiter's star tracker, a relative navigation update, and the NCC burn. The rendezvous radar acquired the target at approximately two revolutions out. At V-bar crossing, one revolution prior to intercept, the TI burn occurred. In the final orbit of the approach, four MC maneuvers were performed to ensure the Orbiter arrived slightly above the V-bar approximately 400 feet ahead of the target. From here a positive V-bar approach (discussed previously in chapter III) was planned to complete the intercept. (Figure 4-5.)



**Figure 4-5.**

**Flight day nine** - Disassembly and storage of DTO hardware was completed in preparation for return to Earth.

## **V. NPS SOFTWARE ARCHITECTURE**

### **A. NPS SOFTWARE DESIGN PHILOSOPHY**

The NPS software architecture design is an event driven architecture. The NPS software must collect data from multiple sources such as communications ports or files, process data, control the screen, and respond to keyboard inputs. The program cannot be dependent on any single event occurring such as might be required in a loop. The NPS software must poll the data source, then continue with its other tasks or return control to the calling routine. The program must handle keyboard interrupts without losing input data. It must be robust enough to handle partial data packets or suspension of data transfer without crashing.

The NPS software is organized on several levels. Data types used by the system are organized into a hierarchy of data structures for program efficiency and readability. The code is organized into a versatile toolbox of functions that allow for flexibility in programming and usage. The file system is organized for ease of editing functions, adding new functions or files, and compiling the code.

### **B. DATA STRUCTURES**

The NPS software was designed to run on the GRID 1530 PGSC, a 10 MHz 80386 portable computer. The slow clock speed,

combined with high data rates and computationally intensive orbit propagation, necessitated an efficient code in order to ensure no loss of data.

In order to minimize the time spent passing large data structures (such as a state vector which contains 56 bytes) around between functions, the NPS software maximizes the use of pointers to access data structures. Simply passing the pointer to a structure into a function allows access to the entire structure in the function by referring to the memory address of the structure.

Efficient use of structures internal to the NPS software allow single functions, such as plotting routines, to be used for multiple combinations of inputs, adding flexibility to the code and reducing the executable file size. For example, there is only one R-bar/V-bar plotting function. However, this one function is able to display six distinct plots simply by calling the function with pointers to structures containing the plot data, titles, scales, and ring buffer files for the individual plot desired.

Some defined structures are used for editing screens. An editing screen allows the operator to manually input or modify program variables such as state vectors or function toggle switches. Again, the use of pointers to structures containing edit information is a flexible and efficient way of programming.

Several types of structures have been created for use in programming. These include simple structures and complex structures. A simple structure, such as the **Vector** structure, consists of three **doubles** (double precision floating point number, 8 bytes each), **x**, **y**, and **z**. A complex structure, such as the **State\_vector** structure, contains two **Vector** structures and a **double** for time. The structure data types used by the NPS software are defined here.

#### 1. **Vector Structure**

A **Vector** structure consists of three **doubles** (8 bytes each), **x**, **y**, and **z**. **Vectors** are used in complex structures such as **State\_vectors** and independently throughout the code.

#### 2. **State\_vector Structure**

The **State\_vector** structure contains two **Vector** structures, representing position and velocity, and a **double** representing time. **State\_vectors** are the primary information source the NPS software was designed to examine.

#### 3. **RV\_vector Structure**

The **RV\_vector** structure contains a position **Vector** structure and a **double** representing time. An **RV\_vector** contains the coordinates of the Orbiter with respect to the target in the **R-bar/V-bar** coordinate system.

#### 4. **Diff\_vector Structure**

The **Diff\_vector** structure contains an array of two **doubles**, the difference of the magnitudes of two state

vector's position and velocity vectors, and a double containing time.

#### **5. Quaternion Structure**

A **Quaternion** structure contains five doubles, q1, q2, q3, q4, and time. A Quaternion is a four parameter representation of a transformation matrix. It provides a numerical relationship between coordinate frames. Quaternions are used due to their convenient small size; that is, four parameters, as opposed to nine parameters in a transformation matrix (or direction cosine matrix). [Ref. 4, p 1-1] The relationship between the Orbiter body and M-50 (Aries-mean-of-1950 Cartesian coordinate system) inertial coordinate frames comes into the NPS software as a quaternion. The quaternion is then converted into a direction cosine matrix for use in the program. Quaternion algebra is not used in the NPS software.

#### **6. Packet Structures**

State vectors and various other information are passed into the NPS system in the form of data packets. A data packet is a structure consisting of several other structures including a **Packet\_header** structure, packet specific information, and a **Packet\_trailer** structure. There are three types of packets, (1) the **Orb\_packet** (Orbiter packet), (2) the **Spas\_packet** (SPAS packet), and (3) the **Orb\_gps\_packet** (Orbiter GPS packet). The **Orb\_packet** contains an Orbiter **State\_vector** structure, Orbiter **Quaternion** structure, **KU\_info** structure,



and target **State\_vector** structure. The **Spas\_packet** contains the SPAS **State\_vector** structure. The **Orb\_gps\_packet** contains the Orbiter TANS GPS **State\_vector** structure.

The **KU\_info** structure was included to allow input of information relating to the KU-band radar used for approach and rendezvous. This structure was not utilized for STS-51. It was retained for possible future use.

#### **7. Nps\_state\_vector\_info Structure**

The **Nps\_state\_vector\_info** structure contains the input **state\_vector** structure, a second 'copy' **state\_vector** structure, and various administrative flags and time markers used by the program to organize the information known about a given state vector. The original input state vector is never altered in the NPS system. Only the 'copy' is manipulated or propagated. This preserves the original data should some other function or higher level system require it.

#### **8. Nps\_quaternion\_info Structure**

The **Nps\_quaternion\_info** structure serves the same function for the **Quaternion** structure as the **Nps\_state\_vector\_info** structure serves for the **state\_vector** structure.

#### **9. Gravity\_data Structure**

The **gravity\_data** structure consists of five doubles containing planetary constants such as Earth radius,  $\mu_{\text{earth}}$  (Earth gravitational constant), the Earth flattening

coefficient, the J2 zonal harmonic coefficient, and the earth rotation rate  $\omega$ . The **gravity\_data** structure also includes two **double** arrays containing the GEM 9 gravity model sectoral and tesseral harmonic coefficients.

For further information on gravity model data, see [Ref. 5: Chapter 9].

#### 10. Gravity\_model\_data Structure

The **gravity\_model\_data** structure contains a **gravity\_data** structure and three **integers**. Two of the **integers** determine the number of sectoral and tesseral harmonics to use in the gravity model. The third is an on/off mode switch.

#### 11. Drag\_model\_data Structure

The **drag\_model\_data** structure consists of four **doubles** defining the solar flux (F10.7), mean solar flux (F10.7<sub>AVE</sub>), the Earth's geomagnetic index, and a ballistic number for the object to which the **drag\_model\_data** structure is assigned.

#### 12. Perturbations Structure

The **Perturbations** structure contains a **gravity\_model\_data** structure and a **drag\_model\_data** structure. Each propagated object (ie Orbiter, ORFEUS/SPAS) has its own **Perturbation** structure.

#### 13. Nps\_predictor\_thrust Structure

The **Nps\_predictor\_thrust** structure contains three **integer** flags indicating the status (on/off) of the thrust predictor function and the delayed thrust option, and the

LVLH/body coordinate switch. This structure also contains a **double** with the delayed thruster firing time, and a **Vector** structure holding the delta-V components of the thrust.

#### **14. Nps\_predictor Structure**

The **Nps\_predictor** structure contains a pointer to the **Perturbations** structure, a pointer to the **Nps\_quaternion\_info** structure, the **Nps\_predictor\_thrust** structure, three **doubles** containing step size, delay time between predictor updates, and the time when the predictor will execute next. The structure also contains two **integers** declaring the number of steps to predict and whether the function is active or not.

#### **15. Nps\_graphics\_info Structure**

The **Nps\_graphics\_info** structure contains **integers** defining the video screen width and height, screen center, font size, page number (for text screens), and a text/graphics mode flag.

#### **16. Nps\_plot\_axis**

The **Nps\_plot\_axis** structure contains two **doubles** representing the axis origin and the axis range for a single axis on a given plot.

#### **17. Nps\_plot\_info Structure**

The **Nps\_plot\_info** structure contains pointers to plot title strings and **Nps\_plot\_axis** structures for the x and y axis.

### **18. Nps\_rvplot\_info**

Each R-bar/V-bar plot has a designated **Nps\_rvplot\_info** structure containing the **Nps\_plot\_info** structure, a pointer to the plot **RV\_vector** ring buffer, a pointer to the current menu bar list, pointers to the associated state vectors, pointers to the associated buffer storage file, a pointer to the **Nps\_predictor** structure, and various plot function flags. The plotting function is called with this structure as the calling argument. This permits a single relative motion plot function or difference plot function to be customized to display any one of the six possible combinations of desired state vector comparisons with appropriate titles, plot scale, and other features.

### **19. Nps\_diffplot\_info Structure**

Each 'Sawtooth' plot has a designated **Nps\_diffplot\_info** structure containing the **Nps\_plot\_info** structure, a pointer to the plot **Diff\_vector** ring buffer, a pointer to the current menu bar list, pointers to the associated state vectors, pointers to the associated buffer storage file, and various plot function flags. The plotting function is called with this structure as the calling argument. This permits a single difference plot function to be customized to display any one of the four possible combinations of desired state vector comparisons with appropriate titles, plot scale, and other features.

## **20. Nps\_pyplot\_info Structure**

Each Pitch/Yaw plot has its own **Nps\_pyplot\_info** structure containing title strings and a pointer to the menu structure. The Pitch/Yaw function is called with this structure as the calling argument. This permits a single Pitch/Yaw plot function to be customized to display pitch/yaw information based on any one of the three possible combinations of source state vectors with appropriate titles.

## **21. Nps\_filter\_info Structure**

The **Nps\_filter\_info** structure contains **integers**, **doubles**, and **Vectors** used for initializing and maintaining the GPS Filter function and its' ring buffer.

## **22. Time\_dhms Structure**

The **Time\_dhms** structure contains **unsigned short** integers for the day of the year, and the hours, minutes, seconds, and milli-seconds past midnight.

## **23. Edit\_label Structure**

The **Edit\_label** structure contains two **integers**, the text screen row and column location of the edit field title, and a pointer to the character string containing the edit field title.

## **24. Edit\_field\_info Structure**

The **Edit\_field\_info** structure defines the edit field string location, flags, size, and cursor offset.

## **25. Edit\_field Structure**

The **Edit\_field** structure defines screen location, data location, and field edit routines for a single edit field.

## **26. Current\_field Structure**

The **current\_field** structure contains pointers to the **Edit\_field** and **Edit\_field\_info** structures currently in use.

## **27. Edit\_screen Structure**

The **Edit\_screen** structure contains **Edit\_label** and **Edit\_field** structures that make up an individual edit screen and the **current\_field** structure. The **Edit\_screen** structure also contains pointers to data used by edit screen processing routines.

## **C. PROGRAM ARCHITECTURE**

The NPS software is not a single program. It is a collection of functions that can be mixed and matched to create specific applications using the NPS relative motion plotting and state vector analysis routines. One version of the code might read its input from multiple communications ports (ie the flight version) while another version acquires its input from data files (ie for post-flight analysis of recorded data). An all encompassing 'Do-it-all' version was undesirable due to size limitations on the executable size of the program.

The versions of most interest are the NPS 'stand-alone' flight version and the TANS GPS/NPS integrated version as

these were the versions flown on STS-51. Two other versions of interest are the 'simulator' version, which creates its own state vector inputs in realtime and is used for program testing, and the 'replay' version, which plays back recorded data (real or simulated). Despite the variety of possible combinations, all variants of the program contain a common architecture. An understanding of one yields an understanding of the others.

#### **1. The 'Main' Function**

All 'C' programs must have a 'main' function from which to begin. The NPS software is no different. An important feature of the NPS software is that it does not matter where the 'main' function is or what it does, only that it calls the appropriate NPS software subfunctions.

From the 'main' function the data source is initialized. The source could be a data port or other device, a file, or another simulator function. The NPS software does not care where the data comes from, only that it is in the appropriate format.

Following data source initialization, the NPS system is initialized. System initialization includes the following: (1) opening and reading configuration files containing memory setup data, IERS data, and plot history files, (2) memory setup, (3) plot ring buffer initialization, (4) graphics driver initialization, and (5) keyboard handler designation.

At this point, the 'main' function enters an infinite loop of updating the NPS data buffers and awaiting keyboard commands. The NPS system is running and processing incoming data regardless of whether or not the plotting screens are up. Control of the program resides with the controlling keyboard handler. The 'main' routine has a keyboard handler and the NPS program has a keyboard handler. When the 'main' program is initiated, program control is held by the 'main' keyboard handler. Control is passed to the NPS keyboard handler upon selection of the NPS system. Both systems, the calling 'main' program and NPS, are still running, only now the NPS keyboard handler controls the program and will only respond to NPS system assigned key strokes. Escaping from the NPS system passes keyboard control back to the calling program but does not terminate calls to the NPS software for data updates.

Upon 'main' program termination, the NPS system is gracefully exited. This includes storage of plot history files for future recall and the properly closing out of the data sources (ie files, ports, etc.).

## **2. Keyboard Control**

Control of the program resides with the keyboard handler. The main routine, which may be a program like the TANS GPS program in the integrated version, the simulator propagator in the simulator version, or simply a shell program that does nothing, has a keyboard handler. This 'main'



keyboard handler takes control of the keyboard upon start-up of the 'main' program. Control is passed to the NPS keyboard handler upon selection of the NPS system. Both the main program and the NPS software as a lower level system are still running, only now the NPS keyboard handler controls the keyboard and will only respond to recognized key strokes assigned by the NPS software. Escaping from the NPS system passes keyboard control back to the calling program but does not terminate calls to the NPS software for data updates.

As an example of program architecture, in the TANS GPS/NPS integrated flight version of the program, the 'main' function is the TANS GPS program. The TANS GPS program performs the functions of communicating via the RS-422 port with the TANS GPS unit, collecting and storing of TANS GPS data, and presentation of TANS GPS data, as well as the NPS input port and system initialization, and updating the NPS system with TANS GPS state vectors. When TANS.EXE is executed, the TANS GPS program is displayed and the TANS GPS keyboard handler is in control. The NPS system is running but is not observed until the NPS function key in the TANS GPS keyboard handler is selected. At this time the NPS software displays are shown, the NPS keyboard handler takes control, and the program responds to all NPS software system commands. The TANS GPS program is still collecting and storing TANS data and performing it's other necessary functions. Exiting the NPS system returns the display and keyboard control to the TANS

GPS program but does not interrupt the flow of data to the NPS system.

#### **D. NPS SOFTWARE FILE DIRECTORY ORGANIZATION**

The NPS software files are organized into a main directory with subdirectories. Files are located in subdirectories by subject. A Borland C++ 'makefile' is used for compiling the various versions of the code. The 'makefile' utilizes the filing system described here to optimize compiling time.

The main directory name is arbitrary and referred to as the 'NPS\_51' directory for the purpose of this report. Subdirectories included in the NPS\_51 directory are NPS, NPS\_SIM, TANS, TANS\_SIM, ORBMECH1, UTIL, INC, COM, TEST, and OBJ.

The NPS\_51 directory contains the program executable files, the current International Earth Rotation Service (IERS) data file, configuration files and graphics drivers required by the program, and the 'makefile' used for program compilation. The NPS\_51 directory will also include several '.nps' files created by the NPS software for storing plot information. Files contained in the NPS\_51 directory are the only files necessary to run the program. Files contained in all subdirectories are only required for source code examination, program compiling, or modification.

The **NPS** subdirectory contains source code directly related to the NPS Relative Motion Plotting and State Vector Analysis Program and subfunctions available to the NPS program.

The **NPS\_SIM** subdirectory contains source code for the shell programs of the various versions of the programs. These shell programs call the appropriate initialization, operation, and shutdown routines for the given version. These variations give the user the option to generate simulated state vectors and provide them to the program for program simulation, generate state vectors and send them to a file, communications port, or device, and to read simulated or actual state vectors from a file, communications port, or memory into the NPS program.

The **ORBMECH1** subdirectory contains source code for the Cowell propagator. This propagator includes M-50 to WGS-84 and WGS-84 to M-50 coordinate system transformations, Jacchia atmosphere model and data files, the WGS-84 gravity model and forcing functions, and the Runge-Kutta fourth order integrator. The Cowell propagator was provided as an 'off-the-shelf' tool by the MacDonnell Douglas Corporation for DTO 700-6. The internal structures and programming architecture of the propagator are not necessarily the same as those used in the NPS program.

The **UTIL** subdirectory contains source code for various '.c' and '.asm' utilities called throughout the program by other functions.

The **TANS** and **TANS\_SIM** subdirectories contain source code for the level I TANS GPS code.

The **INC** subdirectory contains source code for 'include' files used at compile time.

The **TEST** subdirectory contains source code for various routines written to develop or test segments of the NPS code.

The **COM** subdirectory contains source code for communications routines used by the program for sending and receiving data via the RS-232/422 port.

The **OBJS** subdirectory is required by the 'makefile' and becomes the repository for object files created in the compilation process.

## **VI. PROGRAM FUNCTIONS**

The NPS flight and test software was developed in a highly modular fashion. It was designed as a group of common software 'tools' that could be used as plug-in modules. The functions available include the standard 'C' library functions and an assortment of functions specifically designed for use in the NPS software. Because of the modularity built into the NPS code, the flexibility exists to replace individual components such as the propagator or plot routines, should more efficient routines be developed at a later date, without making modifications to the remainder of the code.

### **A. VECTOR/MATRIX FUNCTION LIBRARY**

A collection of vector operation tools is included. These tools are used extensively throughout the code. The following are vector and matrix operators written for use in the software.

The **vector\_magnitude** function returns the magnitude of the input vector. The protocol for **vector\_magnitude** is:

```
double vector_magnitude(Vector *v)
```

The **vector\_unit** function returns the unit vector of the input vector. The protocol for **vector\_unit** is:

```
int vector_unit(Vector *vin, Vector *vout)
```

The **vector\_dot** function returns the dot product of two input vectors. The protocol for **vector\_dot** is:

```
double vector_dot(Vector *v1, Vector *v2)
```

The **vector\_cross** function returns the cross product of two input vectors. The protocol for **vector\_cross** is:

```
void vector_cross(Vector *v1, Vector *v2, Vector *vout)
```

The **vector\_transform** function returns the transformed vector product of an input vector and a transform matrix. The protocol for **vector\_transform** is:

```
void vector_transform(Vector *vin, Vector *vout,  
double transform[3][3])
```

The **matrix\_transpose** function returns the transpose of the input 3x3 matrix. The protocol for **matrix\_transpose** is:

```
void matrix_transpose(double m_in[3][3],  
double m_out[3][3])
```

The **state\_vector\_\_itol\_matrix** function returns the M-50 inertial coordinate system to LVLH coordinate system matrix for the given input state vector. The protocol for **state\_vector\_\_itol\_matrix** is:

```
int state_vector__itol_matrix(State_vector *sv,  
double matrix[3][3])
```

In addition to these useful operators, the **vector\_rvbar** and **vector\_diff** functions are used to manipulate the output into the desired form for plotting.

The **vector\_rvbar** function produces the position vector from one state vector (target) to another state vector (chaser). The inputs are assumed to be in GCI, and the output

is returned in the LVLH of the principle state vector (target). The output vector is used by the relative motion plot. The protocol for **vector\_rvbar** is:

```
void vector_rvbar(Vector *target_pos,  
                  Vector *target_vel,  
                  Vector *chaser_pos,  
                  Vector *rv)
```

The **vector\_diff** function returns the difference vector of two input vectors. The difference vector is used by the 'Sawtooth' plot to compare state vector position and velocity. The protocol for **vector\_diff** is:

```
void vector_diff(Vector *v1, Vector *v2, Vector *vout)
```

#### B. COWELL PROPAGATOR

**BGProp** is a Cowell orbit propagator using a Runge-Kutta fourth order integrator. **BGprop** was originally written by Robert Gottlieb and Mike Fraietta of the McDonnell Douglas Corporation for NASA and was modified for use in the NPS software. (NOTE - The function **BGProp** is not written internally with the same system of data structures or modularity that exists in the remainder of the NPS software.)

**BGProp** provides high fidelity solutions for state vector propagation at the expense of heavy computation. The propagator consists of an M-50 to WGS-84 Earth centered Earth fixed (ECEF) coordinate converter, the GEM-9 Earth gravity model and a recursive forcing function routine, the Jacchia atmospheric model, a Runge-Kutta fourth order integrator, and

a WGS-84 to M-50 converter. While it is not the intent of this paper to explain the details of the Cowell propagator, it is necessary to provide some background of the components to understand its function in the NPS software.

#### **1. M-50 to WGS-84/WGS-84 to M-50 Converters**

Integrating the equations of motion requires knowledge of the forcing function acting on the body being propagated. The forcing function is determined by the geopotential model used. For an assumed spherical Earth (a two body problem), the force acting on an orbiting object is a function of altitude only. For an Earth with only zonal harmonics, the force is a function of altitude and latitude. Either of these cases can be integrated in inertial coordinates because rotation of the Earth does not enter into the solution. Tesseral harmonics, however, are dependent upon longitude and require knowledge of ones position over the Earth relative to the Earth's coordinate system. Use of these terms requires converting the state vector from M-50 inertial coordinates to an ECEF coordinate system. **BGProp** refers to this ECEF coordinate system as a WGS-84 coordinate system.

#### **2. GEM-9 Gravity Model**

The GEM-9 gravity model contains normalized zonal and tesseral harmonic coefficients empirically determined from actual satellite orbit tracking. These coefficients are applied to a recursive non-spherical Earth potential function



out to the specified number of harmonics (maximum of 30). This forcing function is used to solve for body accelerations which are integrated to obtain new velocities. Accuracy and computation time for a propagation increase as more harmonics are used. The gravity model selected during the flight included four zonal and four tesseral harmonics.

### 3. Jacchia Atmospheric Model

An atmospheric model is necessary for drag determination. There are many atmospheric models to choose from, each with merits and limitations, and all subject to the local uncertainty that is associated with weather forecasting. For a study requiring knowledge of atmospheric constituents at a specific place for a given amount of time, a highly accurate model is desired. For the purpose of low Earth orbit drag determination, where the body is moving rapidly through the upper atmosphere and the average drag over a time period is desired, a 'good' fast model will suffice.

The Jacchia '70' atmospheric model contained in this code considers factors including latitude, longitude, altitude, time of the year and local time of day, solar activity and the Earth's geomagnetic activity and uses this information to modify precomputed densities contained in tabular form. The result is a fast atmospheric density computation that approximates that density encountered by the orbiting body as it circles the globe.

The coded Jacchia 70 atmospheric model is limited to use between 90 and 500 km altitude. The shuttle typically flies between 200 and 350 km. The STS-51 mission flight profile was between 300 and 350 km.

**BGProp** is called directly using the protocol:

```
void bgprop(double state_vector_time,  
            Vector *position_in,  
            Vector *velocity_in,  
            Perturbations *P,  
            double delta_time,  
            double step_size,  
            Vector *position_out,  
            Vector *velocity_out)
```

or indirectly through the NPS interface function **nps\_bgprop** using the protocol:

```
int nps_bgprop(State_vector *sv,  
               double new_time,  
               int is_orbiter)
```

### C. F AND G FUNCTION PROPAGATOR

For propagations over relatively short time periods where Keplerian motion (planer elliptical orbit) is assumed, the *f* and *g* functions described in [Ref. 5] produce a good approximation of the orbit motion with considerably fewer computations than the Cowell propagator. These qualities make the *f* and *g* functions desirable for use in the predictor and future thrust functions, allowing fast calculation of predicted relative motion into the near future (two to three orbits) without consuming CPU time with the intensive calculations that are

associated with the Cowell propagator. The *f* and *g* functions as used in the NPS software are described in [Ref. 6].

The **getclas** function determines the classical orbital elements from a given state vector. These classical elements are used by the *f* and *g* functions. The protocol for **getclas** is:

```
void getclas(Vector *pos_o, Vector *vel_o,  
             double aeEoMon[5], Perturbations *P)
```

The **f\_and\_g** function propagates the input state vector to a given new time using the orbital elements derived in **getclas**. The protocol for **f\_and\_g** is:

```
void f_and_g(Vector *pos_o, Vector *vel_o,  
             Vector *pos_new, Vector *vel_new,  
             double clas_aeEoMon[5], double t_dif)
```

The **quick\_prop** function is an implementation of **getclas** and **f\_and\_g** and produces a new state vector for the given state vector and new state vector time. This function is an intermediate function that allows the use of any propagator to be substituted for **f\_and\_g** without changing the function call in the calling routine. The protocol for **quick\_prop** is:

```
void quick_prop(State_vector *old,  
               double new_time,  
               Perturbations *perts)
```

#### D. PREDICTOR FUNCTIONS

The NPS software utilizes the **quick\_prop** function described previously to predict where the Orbiter will be with respect to the target sometime in the future.

The **nps\_rvplot\_\_display\_future** function calls the **f\_and\_g** function for a user specified number of times and step size for a given R-bar/V-bar plot in order to display a predicted line of motion of the chaser with respect to the target. Predicted motion is displayed in steps ahead for the given step size. The first ten steps are displayed as digits. For display of predicted motion beyond ten steps, the "+" symbol is used for each step and digits are displayed every tenth step. If predicted motion based on user supplied thruster inputs is desired, the **nps\_predictor\_\_thrust** function (explained below) is called. The protocol for **nps\_rvplot\_\_display\_future** is:

```
void nps_rvplot__display_future(Nps_rvplot_info *this)
```

The **nps\_predictor\_\_thrust** function utilizes the **quick\_prop** function and arguments held in the **Nps\_predictor** structure to add thrust velocity to the Orbiter state vector and implement the **quick\_prop** propagator to solve for the predicted relative position of the Orbiter with respect to the target. Depending on flags contained in the **Nps\_predictor** structure, the solution may be based on current orbital elements of the target and chaser, or on the orbital motion following a programmed Orbiter burn at some given time. To distinguish between predicted motion without thrust and predicted motion with thrust, the "□" symbol is substituted for the "+" for predicted motion following thrusts. The protocol for **nps\_predictor\_\_thrust** is:

```
int nps_predictor__thrust(Nps_predictor *predictor,  
                          State_vector *chaser)
```

The `nps_predictor__rndv` function is an implementation of the two impulse time constrained rendezvous problem described in [Ref. 5, p 179]. The function determines the thrust required to intercept the target in a specified time. The protocol for `nps_predictor__rndv` is:

```
int nps_predictor__rndv(Nps_predictor *predictor,  
                        State_vector *target_in,  
                        State_vector *chaser_in,  
                        double time_to_rendezvous)
```

(NOTE - The `nps_predictor__rndv` function has been updated since the STS-51 flight. The new algorithm is described in [Ref. 6])

## **E. PLOT/DATA FUNCTIONS**

There are three types of plots: (1) six R-bar/V-bar plots for various combinations of available state vectors, (2) four 'Sawtooth' plots for position and velocity comparisons of Orbiter INS vs. TANS GPS and Orbiter Target vs. SPAS GPS sources, and (3) three Pitch/Yaw plots for target pointing information based on various input state vectors. Each plot has common data handling functions and plotting functions.

The data handling functions for R-bar/V-bar and 'Sawtooth' plots include a data initialization function, a data update function, a display update function, a data save function, a data flush function, and a data stop function. (The Pitch/Yaw plot does not maintain a data history buffer, therefore it only requires a display update function.)

The data initialization functions, `nps_rvplot__data_init` and `nps_diffplot__data_init`, set up memory for the plot data history ring buffer and load previously stored plot history files into the buffer if applicable. Protocols for `nps_rvplot__data_init` and `nps_diffplot__data_init` are:

```
int nps_rvplot__data_init(Nps_rvplot_info *this,
                          ushort max_mem, ushort config_mem,
                          Eem_info *eem)
```

```
int nps_diffplot__data_init(Nps_diffplot_info *this,
                             ushort max_mem, ushort config_mem,
                             Eem_info *eem)
```

(`ushort` is a type definition for an unsigned short integer.)

The plot display initialization functions, `nps_rvplot__display_init`, `nps_diffplot__display_init`, and `nps_pyplot__display_init`, call several subfunctions responsible for displaying the initial plot graphics (axis, menu bars, etc.) and plotting the historic data points stored in the ring buffer for that display. Protocols for `nps_rvplot__display_init`, `nps_diffplot__display_init`, and `nps_pyplot__display_init` are:

```
void nps_rvplot__display_init(Nps_rvplot_info *this)
```

```
void nps_diffplot__display_init(Nps_diffplot_info *this)
```

```
void nps_pyplot__display_init(Nps_pyplot_info *this)
```

The data update functions, `nps_rvplot__data_update` and `nps_diffplot__data_update`, take the latest appropriate state vectors, processes plot information with either `vector_rvbar` or `vector_diff` and load the processed plot data into the plot ring

buffer. Protocols for `nps_rvplot_data_update` and `nps_diffplot_data_update` are:

```
void nps_rvplot_data_update(Nps_rvplot_info *this)
void nps_diffplot_data_update(Nps_diffplot_info *this)
```

The display update functions, `nps_rvplot_display_update`, `nps_diffplot_display_update`, and `nps_pyplot_display_update`, take the latest appropriate state vectors, processes plot information with either `vector_rvbar`, `vector_diff`, or `nps_pitch_yaw`, and display the processed plot data to the screen. Protocols for `nps_rvplot_display_update`, `nps_diffplot_display_update`, and `nps_pyplot_display_update` are:

```
void nps_rvplot_display_update(Nps_rvplot_info *this,
                               double target_time,
                               double chaser_time)
— void nps_diffplot_display_update(Nps_diffplot_info *this,
                                   double target_time,
                                   double chaser_time)

void nps_pyplot_display_update(Nps_pyplot_info *this,
                               State_vector *target,
                               State_vector *orbiter,
                               Quaternion *quat,
                               double target_time,
                               double orbiter_time)
```

The data save functions, `nps_rvplot_data_save` and `nps_diffplot_data_save`, store the current plot history ring buffer to a file. Protocols for `nps_rvplot_data_save` and `nps_diffplot_data_save` are:

```
void nps_rvplot__data_save(Nps_rvplot_info *this)
```

```
void nps_diffplot__data_save(Nps_diffplot_info *this)
```

The data flush functions, `nps_rvplot__data_flush` and `nps_diffplot__data_flush`, clear the current plot history ring buffer. Protocols for `nps_rvplot__data_flush` and `nps_diffplot__data_flush` are:

```
void nps_rvplot__data_flush(Nps_rvplot_info *this)
```

```
void nps_diffplot__data_flush(Nps_diffplot_info *this)
```

The data stop functions, `nps_rvplot__data_stop` and `nps_diffplot__data_stop`, return the ring buffer memory reserved in the data initialization process to the operating system. Protocols for `nps_rvplot__data_stop` and `nps_diffplot__data_stop` are:

```
void nps_rvplot__data_stop(Nps_rvplot_info *this)
```

```
void nps_diffplot__data_stop(Nps_diffplot_info *this)
```

The plotting functions for each plot include a plot initialization function, various functions for displaying state vector times and other computed values such as distance to target, and functions for displaying plot points (or lines in the case of the Pitch/Yaw plot.)

#### **F. GPS FILTER**

The program contains a least squares fit GPS state vector filter originally written in Fortran by Dr. Lubomyr Zyla of the McDonnell Douglas Corporation for NASA. TANS GPS state vectors provide high position accuracy on the order of a few hundred



feet. TANS velocity vectors are not so accurate with errors on the order of one meter per second. For state vector propagation over short periods of time this is manageable. For longer propagation, the velocity error can cause significant errors in the propagated state. The purpose of a filter is to 'smooth' the errors in position and velocity so that long propagations may be used in the event the TANS unit is switched off or stops functioning.

The GPS filter is a 'batch' filter, meaning all the operations required to reach a solution are performed each and every time a filtered state vector is desired. The filter maintains a state vector buffer of the latest sixty TANS GPS state vectors. When activated, the GPS filter propagates backwards the latest seven TANS GPS state vectors through the last sixty collected TANS GPS state vectors with a high fidelity propagator (**bgprop**), and compares the outcomes with the recorded state vectors. It uses the results to determine a new 'smoothed' state vector. The large number of high fidelity propagations make this function very CPU intensive.

The architecture of the filter code is similar to that of the plot data functions. This allows for the inclusion of alternate, faster and more accurate filters should they become available in the future. The filter includes functions for data update, data saving, data flushing, and filter execution.

The `nps_filter_data_save` function saves a state vectors position and time in the filter data ring buffer. The protocol for `nps_filter_data_save` is :

```
void nps_filter_data_save(Nps_filter_info *this)
```

The `nps_filter_data_update` function checks an incoming GPS state vector against the last incoming GPS state vector to determine if it is a new state vector. If it is new, `nps_filter_data_save` is called. The protocol for `nps_filter_data_update` is :

```
void nps_filter_data_update(Nps_filter_info *this,  
                           State_vector *new_vector)
```

The `nps_filter_data_flush` function clears the filter data ring buffer. The protocol for `nps_filter_data_flush` is:

```
void nps_filter_data_flush(Nps_filter_info *this)
```

The `ls_filter` function is a 'C' implementation of the least squares fit GPS state vector filter. The protocol for `ls_filter` is:

```
int ls_filter(State_vector *tans_in,  
             State_vector *tans_out,  
             double tans_buf[4][60],  
             int fbi, int point_num,  
             Perturbations *P)
```

The `nps_filter_run` function uses data stored in the filter buffer and calls `ls_filter` to process this data and produce a smoothed GPS state vector. The protocol for `nps_filter_run` is:

```
void nps_filter_run(Nps_filter_info *this,  
                  State_vector *out_vector, Perturbations *P)
```

## G. EDIT FUNCTIONS

The program consists of several edit screens. Each screen is composed of edit fields and labels. An edit field has a screen position (x and y) and contains a converted character string representing some binary variable value. An edit label has a screen position and contains characters to be printed on the screen.

The edit screens allow the user to manually input or edit certain data for various functions. The data being edited may be data that is currently in use by the program. It may even be data that is changing such as a state vector. The program can not stop to wait for data that it needs for processing to be updated by a user. For this reason, real-time editing screens are employed. The real-time edit screen holds the data being edited in a temporary variable until the editing is complete. The new or edited data is reinserted into the original variable upon leaving the current edit screen.

Four types of edit fields exist: (1) integer edit fields, (2) double edit fields, (3) "yesno" edit fields ("yes" is an integer equal to one, "no" is an integer equal to zero), and (4) day/hour/minute/second (dhms) edit fields.

For each type of edit field there are three edit field call back functions, "get", "put" and "key". The "get" function converts its data into a string. The "put" function converts the string back to the appropriate data type. The "key" function behaves like a mini key handler for the data

field in use, allowing keyboard entry of the new data string.

## VII. SOFTWARE OPERATION

Like any program, the NPS software accepts input, operates on that input, and produces output. These processes are the subjects of this chapter.

### A. PROGRAM INPUT

The NPS software uses data 'packets' containing state vectors and a quaternion as input. These inputs are used to produce the various plots for graphical display of the information.

The state vector describes the motion of an orbiting body in a rectangular coordinate system. It consists of a position vector and a velocity vector each containing x, y, and z components, and a time. All state vector comparisons contained in the NPS software are performed in M-50 (Aries-mean-of-1950 Cartesian coordinate system) inertial coordinates. State vector sources used in the NPS software include:

- Orbiter INS State Vector - defines the shuttle Orbiter state. Initial states and updates are provided via a ground tracking solution or the orbiters star trackers. The states are propagated in the orbiter computer by the 'Super-G' propagator.
- Target INS State Vector - defines the state of the ORPHEUS/SPAS or other satellite that the Orbiter is conducting proximity operations with. Initial state is provided by ground tracking.
- Orbiter GPS State Vector - defines the Orbiter state as described by the onboard TANS GPS unit.

- SPAS GPS State Vector - defines the ORFEUS/SPAS satellite state vector as described by the German GPS unit contained onboard the ORFEUS/SPAS.

The Orbiter state vector operates in one of two modes, an inertial navigation mode and a relative navigation mode. In the inertial navigation mode, the Orbiter state is updated via ground tracking. This mode provides a state vector representing the state of the Orbiter in inertial space. In the relative navigation mode, a target state vector is up-linked to the Orbiter. The target state vector is considered to be a 'truth' vector. The Orbiter's star tracker is trained on the target to smooth the Orbiter state vector relative to the target. At closer ranges (<30 miles) this relative state vector can be further smoothed using sensor data from the Orbiter's KU-band radar to get an accurate real-time relative tracking solution.

The quaternion relates the Orbiter body coordinate system to the M-50 inertial coordinate system. It is calculated from angular rates determined by the inertial measurement unit (IMU) and updated by the Orbiter's star tracker. The quaternion is required by the NPS software for computing attitude related information such as pointing information to the target.

## **B. DATA OPERATIONS**

The NPS software receives M-50 inertial state vectors from various sources and processes these to provide relative motion

and vector difference plots. To accomplish this, the incoming state vectors must be propagated to the same point in time, then transformed into the appropriate form for display. The following is a simplified explanation of how the code works by way of an example.

A valid Orbiter INS state vector is received by NPS with a time tag of 0:00:00. At time 0:00:04, four seconds later, a valid Orbiter GPS state vector is received. The NPS software recognizes that it has two different state vectors and that it would like to compare them. The time stamp of the older state vector is compared to the time stamp of the newer state vector and the time difference,  $\Delta t$ , is computed (four seconds in this case.) The older state vector and the  $\Delta t$  are arguments passed to the propagator.

The propagator is responsible for matching the time stamps of two state vectors. In the propagator, the passed M-50 inertial state vector is transformed into Earth Centered Earth Fixed (ECEF) coordinates. In these coordinates, the Earth gravity model and atmospheric model are applied to determine the geopotential forcing function and drag. The resulting forcing function is integrated over time  $\Delta t$  using the Runge-Kutta integrator. The propagated ECEF state vector is transformed back into M-50 inertial coordinates and returned to the NPS main routine. The two state vectors are now matched in time and comparison of position and velocity is performed.

### C. PROGRAM OUTPUT

The NPS software presents state vector comparisons in three different forms: (1) the R-bar/V-bar plot, (2) the state vector difference or 'Sawtooth' plot, and (3) the Pitch/Yaw display.

#### 1. R-bar/V-bar Plots

The R-bar/V-bar plot displays the relative motion between a target and a chaser vehicle and is of particular importance during rendezvous and proximity operations. The relative motion plot uses a target-centered coordinate system

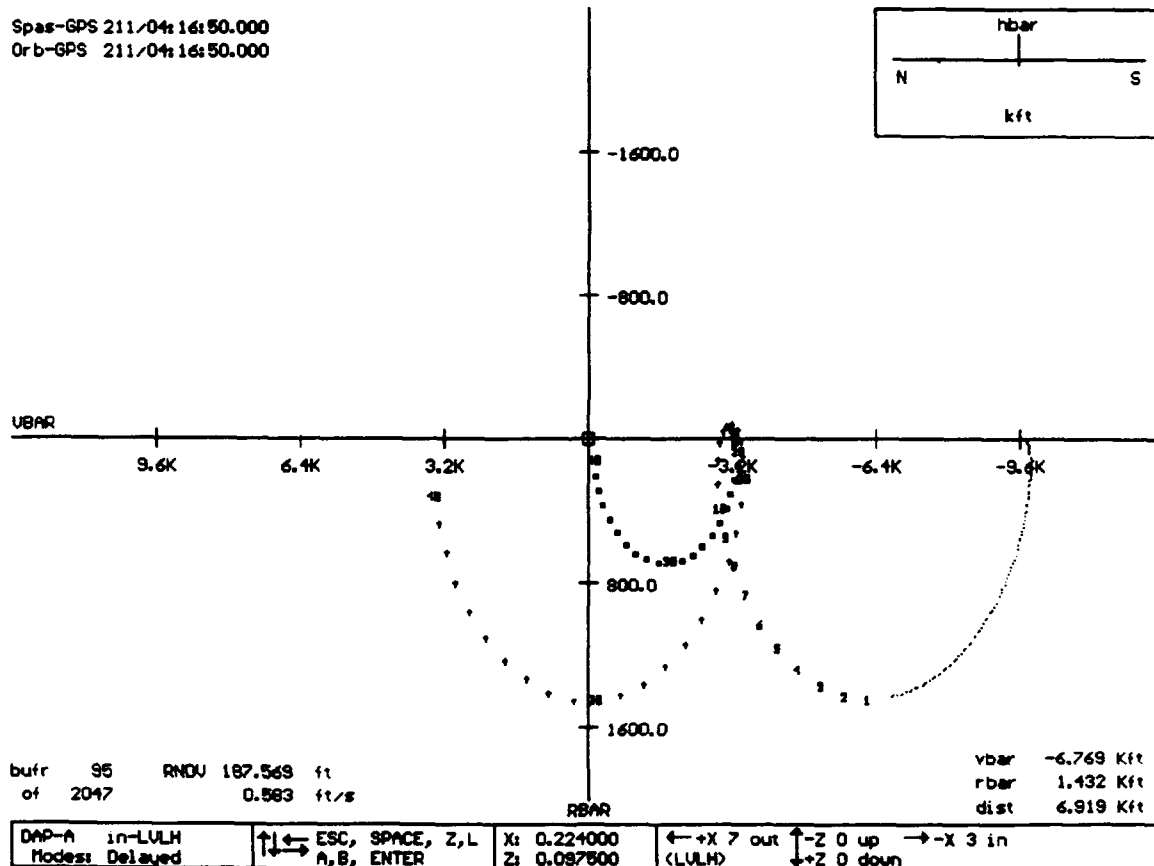


Figure 7-1. Sample R-bar/V-bar plot.



in which the Z-axis rotates with the target and is positive directed radially toward the Earth, the X-axis is curvilinear and positive in the direction of orbit motion, and the Y-axis is out-of-plane and completes the right-hand coordinate system.

R-bar/V-bar plots are available for the following target/chaser combinations of state vectors:

- SPAS GPS/Orbiter GPS
- SPAS GPS/Orbiter INS
- Orbiter GPS/Orbiter INS
- SPAS GPS/Orbiter target
- Orbiter target/Orbiter GPS
- Orbiter target/Orbiter INS

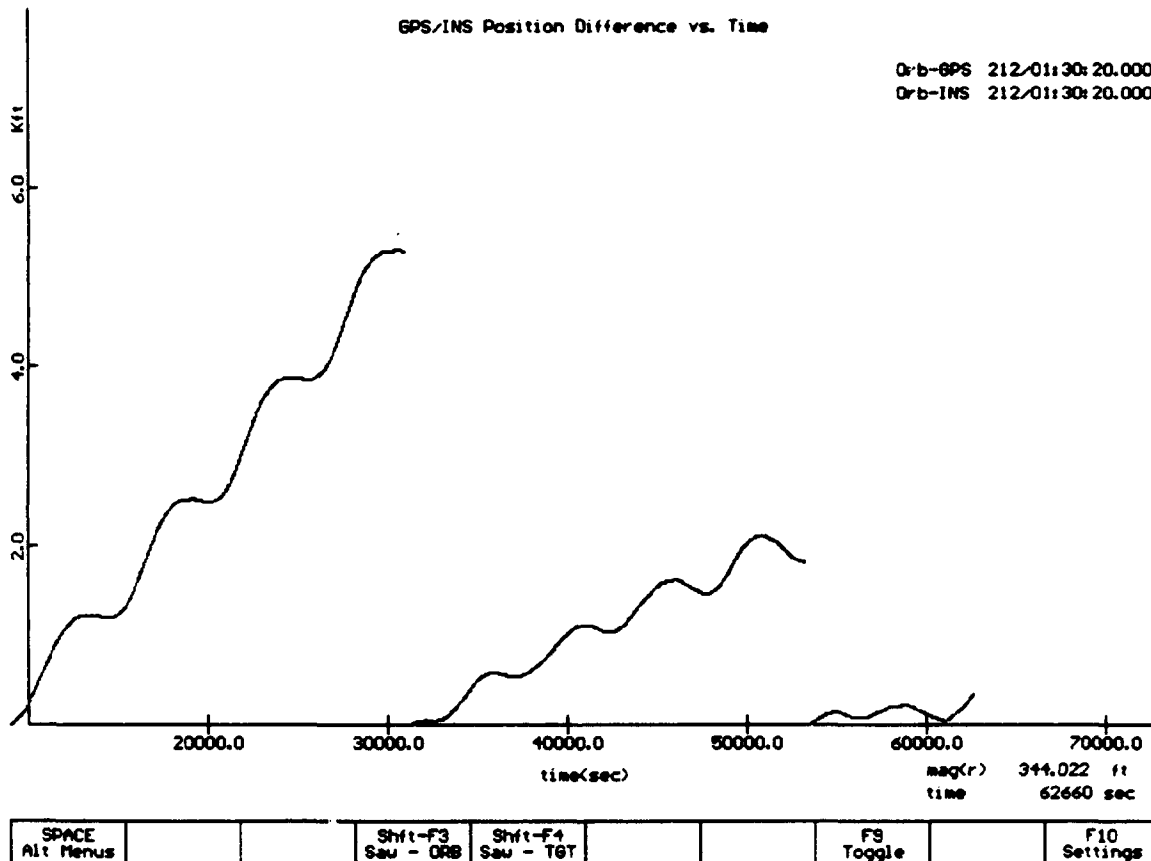
Figure 7-1 is an example of an R-bar/V-bar plot.

Orbiter GPS/Orbiter INS and SPAS GPS/Orbiter target plots are not used as relative motion plots. They are used as another method of viewing the difference vector and show on the R-bar/V-bar display where the inertial navigation system thinks the Orbiter or SPAS is located relative to where GPS believes it to be.

## **2. Sawtooth Plots**

The relative difference or 'Sawtooth' plots display the difference in the magnitudes of the position or velocity vectors of two source state vectors. This is the tool used to quantify the error between the inertial navigation system and

GPS. The term 'Sawtooth' was derived from the expected output display of the plot. It is known that the inertial system will drift over time. Assuming the GPS state vector error to be small and the GPS position to be close to the true value, it



**Figure 7-2. Sample 'Sawtooth' plot.**

was expected that the difference vector magnitude would grow over time. On each occasion of an INS update from ground tracking or onboard star trackers, it was expected that the difference vector would be reduced to a small value as the INS position and velocity were returned to values inside the GPS

error. The results from these plots serve to validate or invalidate the use of GPS for Orbiter state vector updating. Two state vector source combinations for 'Sawtooth' plots are available:

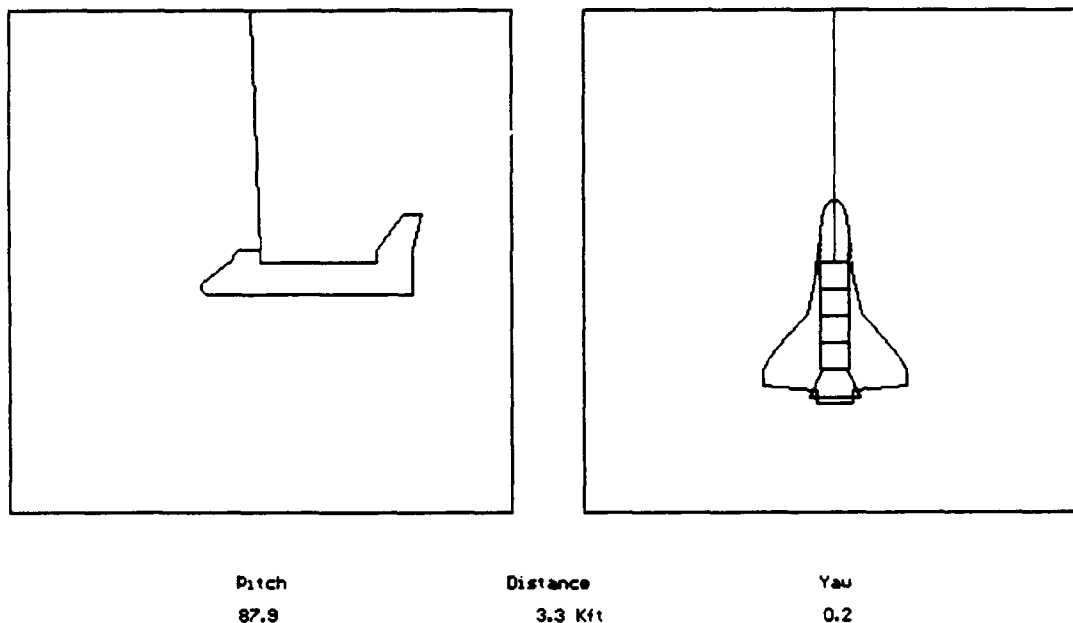
- Orbiter GPS/Orbiter INS
- SPAS GPS/Orbiter target

Figure 7-2 is an example of a 'Sawtooth' plot.

### 3. Pitch/Yaw Displays

The Pitch/Yaw display gives a graphical presentation of the Orbiter (TOP and SIDE views) and a target pointing

Spas-GPS 212/05:57:00.000  
Orb-GPS 212/05:57:30.000



ESC prev scrn	F1 Spas/O-GPS	F2 Spas/O-INS	F3 O-GPS/INS	F4 Spas/TGT	F5 TGT/O-GPS	F6 TGT/O-INS		F9 Toggle	F10 Settings
------------------	------------------	------------------	-----------------	----------------	-----------------	-----------------	--	--------------	-----------------

Figure 7-3. Sample Pitch/Yaw plot.

vector measured in pitch up from Orbiter body X-axis and yaw about the rotated Orbiter body Z-axis. These angles are computed from the state vectors of the Orbiter and target, and the Orbiter quaternion. Target distance is also shown. Three Target/chaser state vector source combinations for the Pitch/Yaw plot can be selected:

- SPAS GPS/Orbiter GPS
- Orbiter target/Orbiter INS
- SPAS GPS/Orbiter INS

Figure 7-3 is an example of a Pitch/Yaw plot.

### VIII. DATA COLLECTION AND ANALYSIS

During Discovery's ten day mission, hardware equivalent to the flight hardware was assembled in Mission Control at the Johnson Space Center (Figure 8.1). The 128 Kbyte downlink data stream was wired into a desktop computer and a GRID 1535 laptop computer, each loaded with the "PCDecom" software. The desktop "PCDecom" unit was used to record the downlink telemetry for the entire flight on digital magnetic tape. This data was used for playback and post-flight analysis. The laptop "PCDecom" unit was connected via an RS-232 port to another generic laptop computer loaded with the NPS software (stand-alone flight version). This allowed ground personnel to view the program in realtime exactly as the flight crew saw it. All state vector information for the Orbiter inertial system and the ORFEUS/SPAS were available for plotting. The TANS GPS data was not available for viewing from Mission Control during the flight.

Post-flight data analysis included preparation of the data for replay and merging of the PCMMU recorded data and the TANS GPS data stored separately in flight. Subsequent sections of this



**Figure 8.1 DTO equipment in Mission Control during flight.**

chapter summarize the flight data collected and analyzed at the time of this writing with respect to the objectives of the DTO stated in Chapters I and II.

#### A. GPS STATE VECTOR ANALYSIS

The level II DTO objective, comparison of GPS state vector data with corresponding inertial or ground tracking state vector solutions, was met yielding very useful results. Comparisons of both Orbiter INS vs TANS GPS and Orbiter target vs SPAS GPS were obtained.

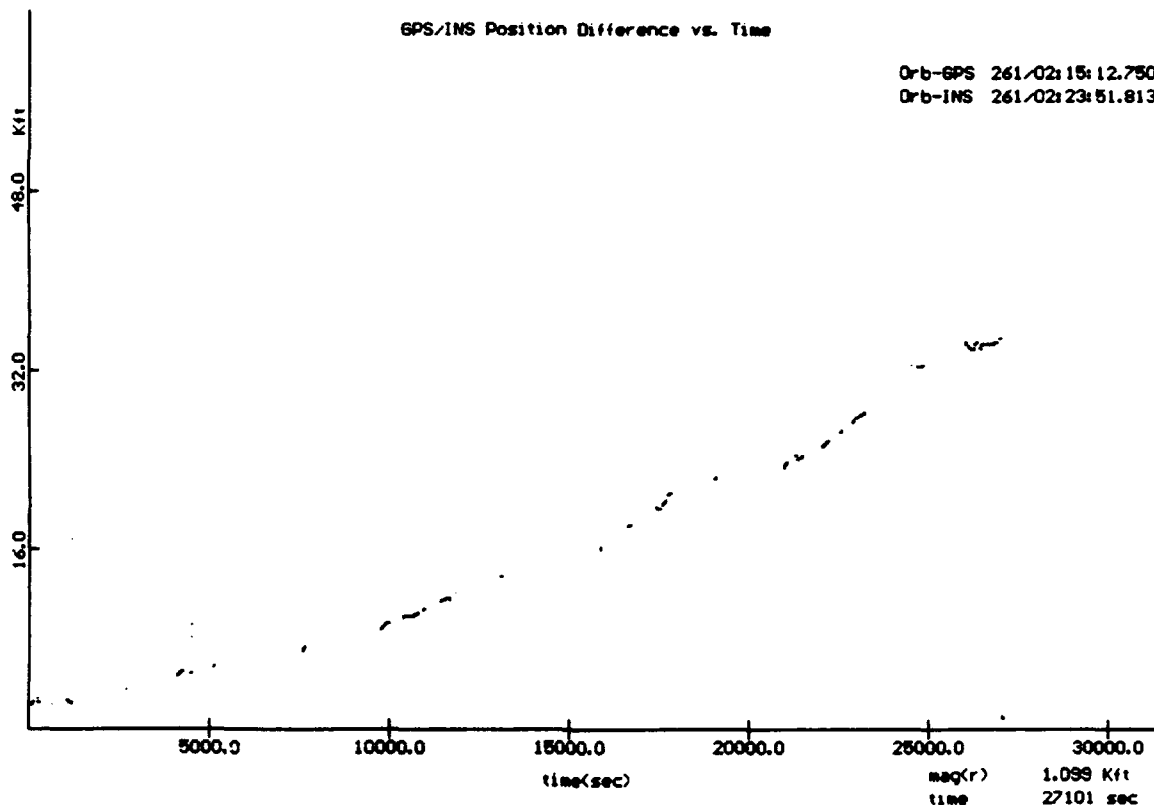


Figure 8.2 Orbiter INS vs TANS GPS "Sawtooth" plot

## 1. Orbiter INS vs TANS GPS

A comparison of the inertial solution for the Orbiter state and the TANS GPS solution was performed during flight when the Orbiter attitude was permitted four satellite GPS reception. While continuous GPS state vector information was not available throughout the flight, there were several periods of continuous reception exceeding several hours in duration. This allowed observation of the expected 'sawtooth' predicted in the Orbiter INS vs TANS GPS difference plot.

Figure 8.2 shows the state vector differences for GPS input states vs inertial states during the longest continuous period of GPS state vector reception during the mission. Over the eight hour period shown, the difference in the INS and GPS position is observed to grow to approximately 35,000 feet before a navigational update from the ground removes inertial system drift error and reduces the error to approximately 1000 feet. GPS state vector reception is lost shortly after the update.

Figure 8.2 shows the first step in validating the use of GPS for on orbit navigation. While the inertial system position accrues a significant error over time, the GPS position error remains bounded by the limits of the GPS receiver and the dithering of the incoming GPS signal caused by selected availability (SA).

It is important to note that the data shown in Figure 8.2 includes only comparison of the incoming GPS state

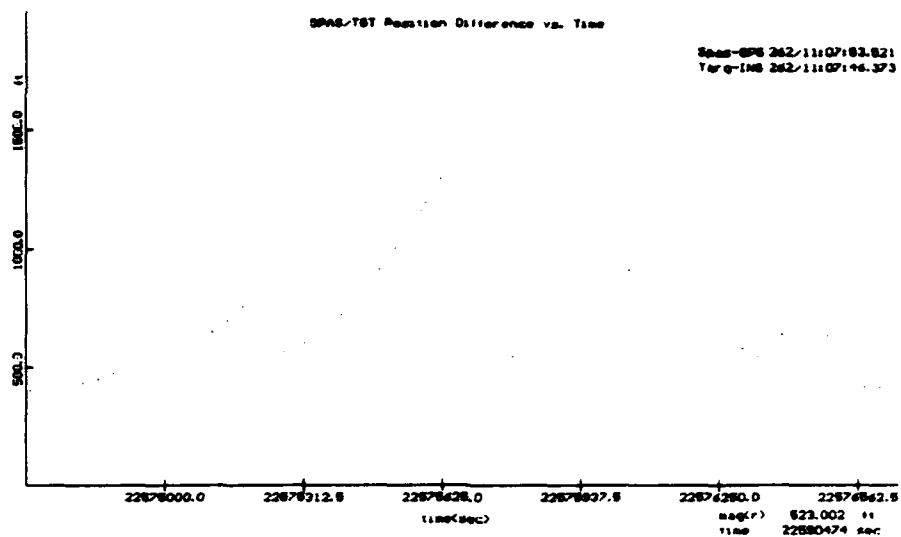
vectors, not the propagated GPS state. Propagation of the GPS state vector yields a position difference plot with erratic spikes between updated states due to errors in GPS velocity on the order of one meter per second. Such errors are probably due to the high Position Dilution of Precision (PDOP) of the GPS signal observed during flight and concur with expected errors for the observed PDOP. [Ref. 7]

## **2. Orbiter Target vs SPAS GPS**

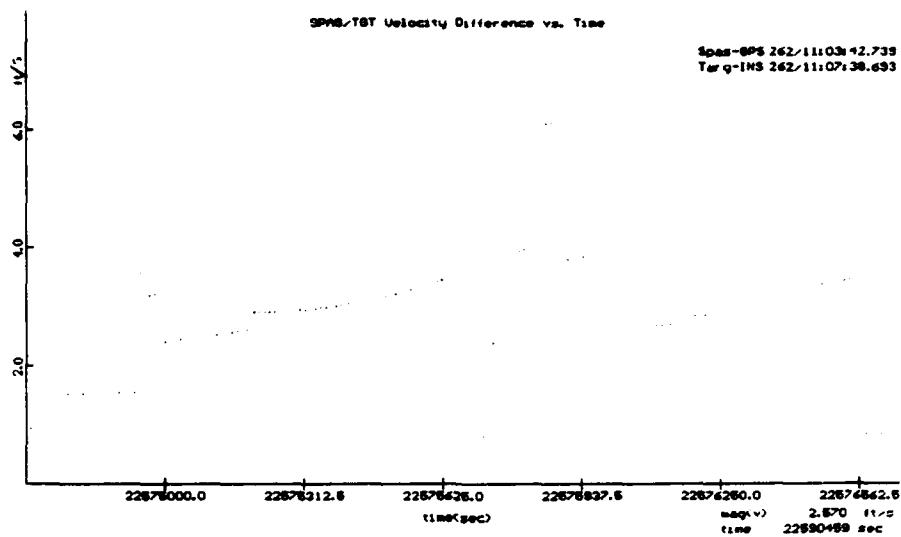
Comparisons of Orbiter target and SPAS GPS state vectors were also made through clever manipulation of the post flight data. During flight the target state vector was updated only periodically when necessary to perform certain tasks such as rendezvous commencement. During the rendezvous itself, it was not necessary to continually update the target state vector because, once the rendezvous commenced, the target state was assumed to be a truth state and the Orbiter state was computed relative to the target. Following the rendezvous and capture of the SPAS, the SPAS GPS data could be compared to the Orbiter inertial system since the Orbiter state and the target state coincided.

Figures 8.3 and 8.4 show a comparison of position and velocity between SPAS GPS and the coincident Orbiter INS/target states for a sample time period following capture of the SPAS. These plots show both the incoming state and its NPS propagated value. State vector updates can be identified





**Figure 8.3 Representative data of comparison of SPAS GPS and Orbiter target state vector position.**



**Figure 8.4 Representative data of comparison of SPAS GPS and Orbiter target state vector velocity.**

by the beginning of the discontinuities shown. These plots illustrate the danger of propagating an orbit with a bad state vector velocity. Position errors accumulate rapidly. The updated position fixes, however, are quite consistent with position errors around 1000 feet.

#### **B. GPS RELATIVE NAVIGATION**

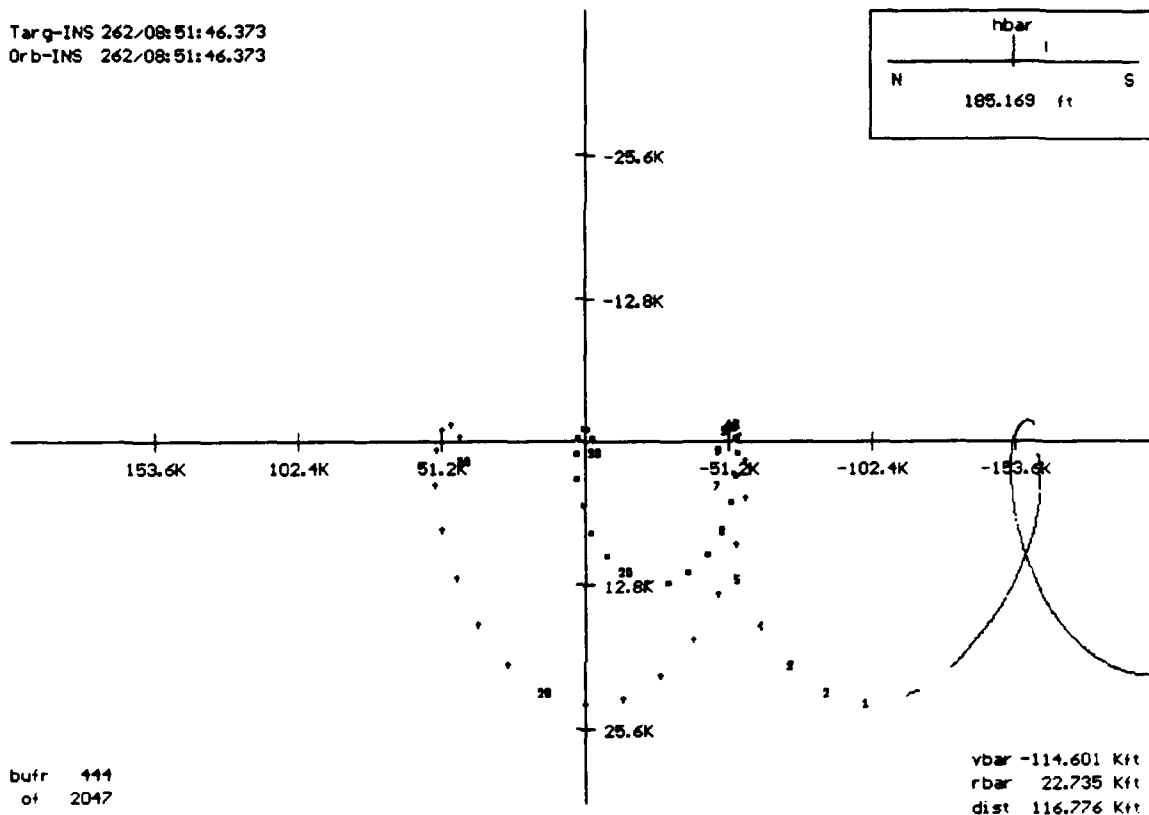
The level III DTO objective of plotting the rendezvous profile with relative GPS was a partial success. Relative GPS navigation was not possible during the rendezvous due to the lack of TANS GPS state vectors during this period. The four satellite reception required for three dimensional GPS position determination was not achieved by the onboard TANS GPS unit during the rendezvous event. However, SPAS GPS reception was good during this period and relative positioning of the SPAS GPS with respect to the Orbiter inertial state vector was possible. The relative motion plots produced were similar in appearance to the relative motion plots produced using inertial state vector sources but lacked the accuracy and relative smoothing required for use during rendezvous operations.

#### **C. RELATIVE MOTION PLOTTING RENDEZVOUS AID**

The level IV DTO objectives provided the most exciting results of the project as the program demonstrated itself to be a powerful and useful tool during the rendezvous with the

ORFEUS/SPAS providing operationally useful information to the flight crew in realtime.

Targ-INS 262/08:51:46.373  
Orb-INS 262/08:51:46.373



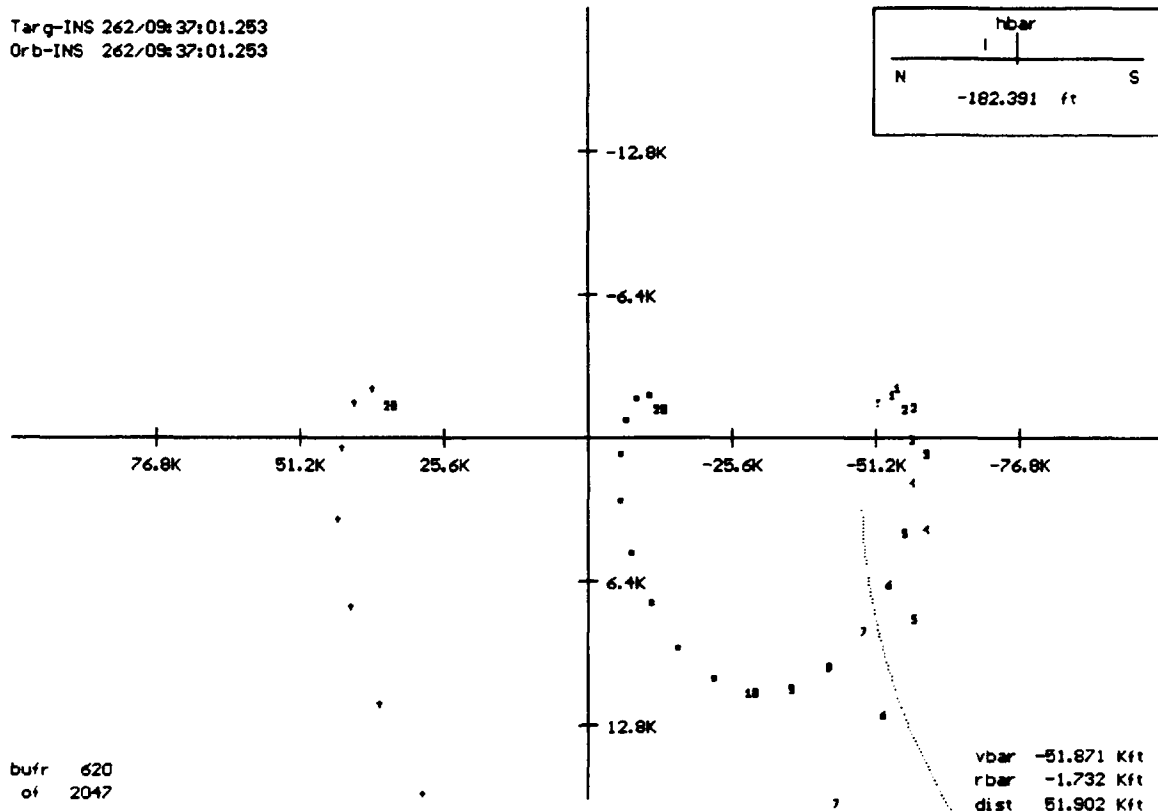
SPACE	F1	F2	F3	F4	F5	F6	F7	F8	F10
Alt Menus	Spas/O-GPS	Spas/O-INS	O-GPS/INS	Spas/TGT	TGT/O-GPS	TGT/O-INS	Singl Pred	Auto Pred	Settings

**Figure 8.5 SPAS Rendezvous. TI Burn (initial look)**

Figure 8.5 shows the rendezvous profile up to one and one half revolution prior to intercept with the pre/post-TI burn predictors displayed. Using the uplinked TI burn rates given by mission control, the crew was able to view the predicted results of the upcoming maneuver.

As the Orbiter closed the distance to the SPAS, the relative state vector solution is 'sweetened' by the

rendezvous radar. As figure 8.6 illustrates, the NPS software predicted the fact that the planned TI burn might cause the Orbiter to fall short of the targeted intercept unless larger than planned MC burns were performed following TI. The actual TI burn did, in fact, target the Orbiter short and the crew was well prepared when the larger MC burns required for rendezvous were calculated.

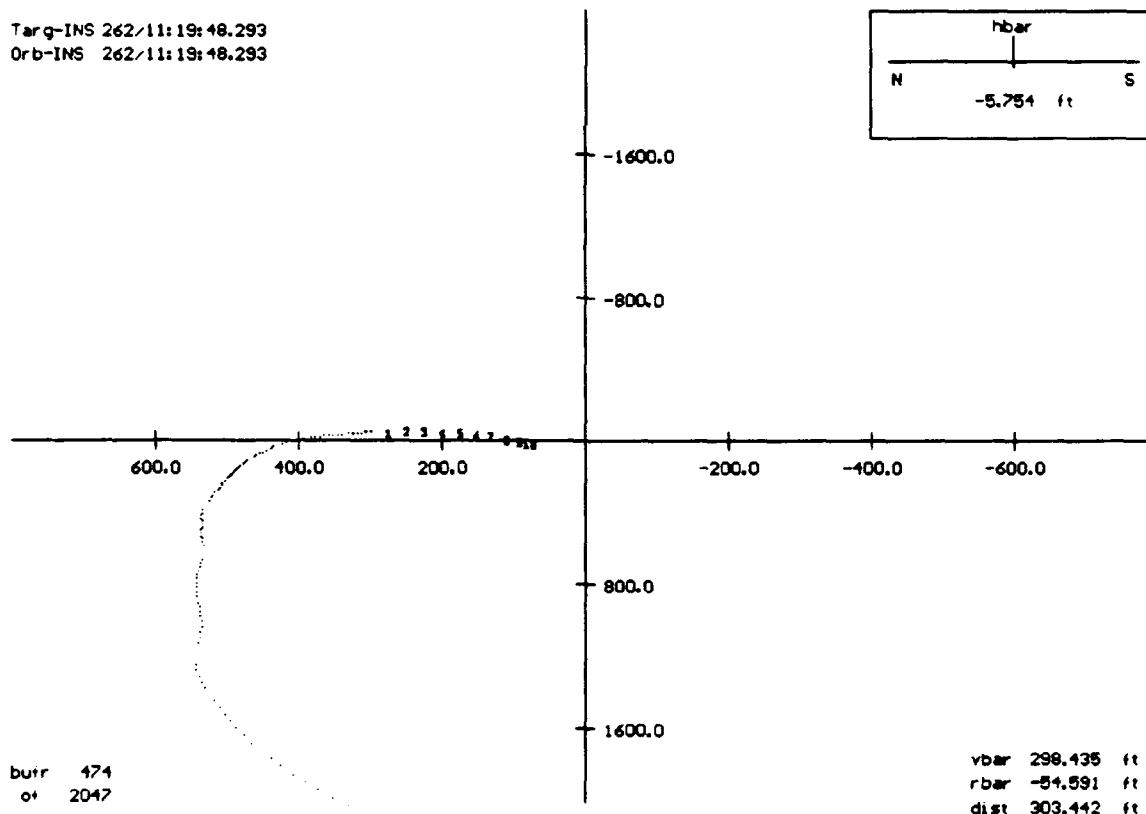


**Figure 8.6 SPAS Rendezvous TI Burn. Predictor falls short of target.**

Given a similar situation in the future, using a certified code with the NPS features, an operator could conceivably use

the Predictor/Future Thrust option of the software to 'walk' the post-TI predicted trajectory to the desired intercept point and suggest modifications to burns as appropriate. The value of this tool is even more evident in a situation where realtime communications with mission control is impractical, such as a manned mission to Mars, or when communications are not possible due to communications loss such as that occurring during the TDRSS gap.

Figure 8.7 shows the final minutes of the rendezvous. The serpentine trajectory recorded is the result of several



**Figure 8.7 SPAS Rendezvous. Final Approach.**

braking and elevating (negative R-bar) maneuvers used to achieve V-bar crossing 400 feet ahead of the target. Review of the cockpit video tape taken during the rendezvous and debrief discussions with Discovery pilot Bill Readdy revealed that the NPS software confirmed information from 'approved' flight aids as well as providing that information faster and more reliably than previous programs used for rendezvous operations. Significantly, the inertial system was able to sense the firing of maneuvering thrusters allowing the NPS software predictors to provide instantaneous feedback of a maneuver's effectiveness to the crew. As a result of the improved situational awareness provided by the NPS software and other rendezvous software flown on STS-51, Discovery's mission commander, Frank Culbertson, skillfully executed the most fuel efficient rendezvous to date in the space shuttle program. [Ref. 8] In fact, data provided in [Ref. 9] confirmed that the actual fuel consumed during the manual flight phase of Discovery's rendezvous was about 66 percent of budgeted fuel usage for the same period.

## **IX. LESSONS LEARNED**

While the Development Test Objective was an unqualified success, as with any project, there were many lessons learned and areas for improvement found. The following is a summary of items noted for both improvements to the software and to the manner in which it is employed.

### **A. 'GIGO' (GARBAGE IN, GARBAGE OUT): GPS VELOCITY ACCURACY**

The NPS relative motion plots are only as good as the state vectors provided to the program. This introduces the problem of poor accuracy of velocity derived from GPS data. Assume that the GPS position information is good. If the velocity error is significant, then any propagation of the state vector will integrate that error over the time of propagation. Unfortunately, GPS derived velocity errors are on the order of a meter per second. This is a significant error in terms of orbital motion. Propagation of state vectors with velocity errors of this magnitude leads to significant position errors in short time spans.

Solutions to this problem may include incorporation of a Precise Position Service (Military) GPS receiver to reduce the errors caused by selected availability, the use of recursive Kalman filtering of the GPS state vector to smooth the velocity solution, or a combination of both. The use of the

least squares fit GPS filter currently used by the program is an inadequate solution due to the time it requires to produce a smoothed state vector.

**B. MORE 'GIGO' OR 'NINO' (NOTHING IN, NOTHING OUT)**

Related to the GIGO problem noted above is the problem of discontinuous data. Again, the program output is only as good as its input. Of course, if no target state vector is available, then relative motion plotting is impossible. Such was the case for significant portions of the flight.

The target state vector was not maintained by ground tracking except (and fortunately) during separation, rendezvous, and proximity operations with the ORFEUS/SPAS. The SPAS GPS state vector was only available during limited time periods during the flight due to the high volume of telemetry from the ORFEUS/SPAS's onboard experiments. Segments of the flight when neither the target state vector nor the SPAS GPS state vector were available severely reduced the capabilities of the relative motion plotting functions. Relative motion plotting is still available if the target state vector is known and hand keyed into the program. However, this information is still dependent on the ground providing good target state vector information to the crew. The lack of a good target vector during periods of SPAS GPS reception also hampered comparison of these two state vector sources.



The TANS GPS data is not continuous as the GPS reception was dependent upon the Orbiter's flight attitude. In addition, while good TANS GPS reception was the rule during the flight, the four satellite coverage required for three dimensional position and velocity determination was the exception. As a result, the actual percentage of flight time with good TANS GPS state vectors was small. Possible solutions for this include the use of exterior mounted GPS antennae in future flights to increase the range of good Orbiter attitudes for reception.

#### **C. THE GPS FILTER**

The GPS filter written by McDonnell Douglas for the software, while functional, was inadequate for the required task. This filter requires high fidelity propagation of the latest seven GPS state vectors back through the last sixty collected GPS state vectors (420 high fidelity propagations), and comparison of the outcomes with the recorded state vectors in order to solve for one 'smoothed' state vector. This 'batch' process takes on the order of one minute of CPU time on the computers currently in use. This dedication of CPU time to a single task was not an affordable luxury for the mission. The need for continual GPS data reception, processing, and storage to disk, as well as plotting functions of the NPS code precluded use of the filter.

A recommended replacement for the least squares fit GPS filter might be a recursive filter that stores a solution and updates that solution progressively rather than performing a 'batch' operation upon selection.

#### **D. TDRSS COMMUNICATIONS GAPS**

Once every revolution the Orbiter experiences a total loss of communications with the ground due to the gap in TDRSS coverage over the Indian Ocean. This loss of signal, or LOS, lasts between seven to eight minutes and includes a loss of the down-link telemetry stream which is stored on digital magnetic tape for post-flight analysis. The crew does not experience this loss of data in flight. In fact, the telemetry is stored onboard and down-linked after communications are restored. The data can then be recovered and integrated with the recorded data. This process requires additional data link hardware, some clever programming, and considerable coordination with NASA. An alternative approach might be to store the NPS data packets on board on the laptop hard disk during LOS in the same manner that the TANS GPS data was stored throughout the flight. Integration of the 'gapped' data collected on the ground and the NPS data packets stored on board during LOS could be accomplished by much simpler means than the process required to recover the LOS data from NASA.

## **E. A POST-FLIGHT DEBRIEF/TRAINING TOOL**

The NPS software has a significant value as a post-flight debriefing tool. Once the flight data is replayed from the digital tape and the NPS packets are stored to disk, rendezvous and proximity operations from the flight can be reviewed by the flight crew and by crew members preparing for future missions on the 'replay' version of the program. The simulator version of the NPS software can also be used for crews to practice their own maneuvers or demonstrate hypothetical scenarios. Either replay or simulation require only a desktop or laptop computer. While not a substitute for current trainers and simulators, this capability is an invaluable supplement to both flight crews and to support personnel who otherwise might not receive expensive simulator time.

## **F. GROUND CONTROLLERS AID**

In a similar fashion to that described above, the NPS software is also a valuable situational tool to mission control/payload support personnel, providing a real-time picture of the rendezvous when video down-link is not available. (Use of the KU-band radar during rendezvous precludes the down-link of video due to limited bandwidth.) It is possible for ground personnel to plug into the data link at mission control regardless of whether or not the Orbiter is equipped with the appropriate hardware and software. This was

the case in December 1993 when crew members of STS-60, practicing for the Wake Shield Facility experiment in 1994, and Hubble Space Telescope representatives used the NPS software at mission control during the Hubble Space Telescope rendezvous.

## **X. CONCLUSIONS**

The NPS software positively contributed to the outcome of the STS-51 mission and continues to be a springboard for research and experimentation in the use of real-time data for situational awareness to assist the flight crew on orbit. The best subjective review of the software comes from the flight crew of STS-51. Their inputs concerning the project are recorded in the *STS-51 FLIGHT CREW REPORT* [Ref. 10] and appropriate remarks follow.

For reference, the certified rendezvous software used by NASA is "Payload Bay" (PLBAY). PLBAY is not automated, requiring the operator to manually input most parameters. The program does not offer many of the options available in the NPS software. It also does not offer the flexibility for growth that the NPS software contains. The "Rendezvous/Prox Ops Program" (RPOP), is an automated version of PLBAY, offering reduced need for user interface, but no new functionality over PLBAY. These programs are referred to in the remarks regarding the NPS software.

### **A. Flight Crew Remarks**

The most convincing argument for continued use of programs like NPS on future missions comes directly from feedback provided by the flight crew. Mission Specialist Dr. James

Newman comments on the use of the NPS software in the *STS-51 FLIGHT CREW REPORT*. Referring to the `nps_rvplot_display_future` and `nps_predictor_thrust` functions, he writes;

...These programs (PLBAY, RPOP, and NPS) ran from well prior to TI through the entire rndz and prox ops. The NPS code was able to perform future burns as well as "what if" thruster firings and predicted that the TI burn would result in a short rndz case. This was born out and MCI through 4 all worked to correct this.

Dr. Newman compared the outputs and functionality of the NPS software to PLBAY and RPOP, stating;

...RPOP and the NPS plots were evaluated against the certified version of PLBAY and contributed to the overall situational awareness. The NPS code had a number of features desirable in operational versions of RPOP, including the ability to select predictors more than 9 minutes in the future. It was also able to maintain the no-thrust predicted trajectory and the "what-if" trajectory at the same time, making comparisons of desired thrust inputs easier to do. And NPS kept track of the number of "what-if" firings in the various directions and the net delta-v in the orbiter axes.

Dr. Newman's recommendations to NASA were clearly stated;

Incorporate desirable features from the NPS code into RPOP to improve situational awareness during the rendezvous and proximity operations.

Concerning the GPS on-orbit demonstration, Dr. Newman reported;

...The NPS software produced a delta plot of the GPS position and on-board position. The first time this was observed, the delta was ~18kft. A short while later it was observed to be about 300 ft! This was just before and after MCC unlinked a new orbiter state vector and showed dramatically that the GPS was working as expected.

Post-flight analysis of the GPS data, Orbiter onboard position, and reference trajectory have since confirmed the real-time data observed in flight. [Ref. 11]

#### **B. Future NPS Software Development**

The NPS software is not fixed. It is a continually evolving experiment with the goals of continued improvement in providing the flight crew with added situational awareness. Since the return of STS-51, the software has been modified to incorporate features suggested by astronauts and mission planners. New features include the ability to view the relative motion plot from side, top, and end on views, display of closure rates, orbital plane crossing rate ( $\dot{Y}$ ) and the plane crossing time, and improvements to display options such as scaling.

A modified version known as "NPS lite", processing only the Orbiter and target state vectors, and displaying Orbiter pitch, roll, and yaw angles (to observe Orbiter dynamics), was prepared for use by STS-60 in 1994 for providing display of the rendezvous and proximity operations of Discovery with the Wake Shield Facility experiment.

Currently, STS-66, scheduled to fly late in 1994, is evaluating the use of NPS for another GPS relative navigation test. On this flight the GPS equipped SPAS will again be deployed and retrieved. The TANS GPS receiver will be located in the payload bay to improve GPS signal reception.

Still other flight opportunities exist if the NPS software continues to offer useful features not currently available in 'approved' software. Some of these desirable features include the incorporation of the payload bay laser data (this system is being brought on line for use in MIR and Space Station rendezvous), raw radar data solutions, and a 'Windows' version of the program.

The NPS software serves as a working, "proven" test and evaluation program for new uses of real-time data available via "PCDecom". This includes evaluation of new algorithms for rendezvous and relative motion display, comparison of future test navigational sources, and visual display of non-navigational data such as the shuttle's remote arm position. The opportunities for future research and development are unlimited. It is the hope of the author that others will continue to use and develop the NPS software, offering improvements that will make it a still more valuable product in our nation's space program.



## REFERENCES

1. TANS Trimble Advanced Navigation Sensor 6-Channel GPS Receiver Specification and User's Manual, Trimble Navigation Limited, Sunnyvale, CA, March 1991.
2. Rendezvous/Proximity Operations Workbook, RNDZ 2102, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, February, 1985.
3. Rendezvous - STS-51 Flight Supplement, Mission Operations Directorate, Flight Design and Dynamics Division, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, January, 1993.
4. Quaternions Supplemental Workbook, QUAT-S 2102, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, November, 1984.
5. Chobotov, Vladimir A., *Orbital Mechanics*, American Institute of Aeronautics and Astronautics, Inc., Washington D.C., 1991.
6. Makepeace, L. B., *Theoretical Basis for State Vector Comparison, Relative Position Display, and Relative Position/Rendezvous Prediction*, Masters Thesis, Naval Postgraduate School, Monterey, California, December 1993.
7. Clynch, James R., *Error Characteristics of GPS Differential Positions and Velocities*, Proceedings of ION GPS-92, Albuquerque, New Mexico, 16-18 September 1992.
8. Conversation between Captain William Readdy, Astronaut Office (Code CB), NASA, Johnson Space Center, and the author, 2 December 1993.

9. Facsimile of STS-51 Actual Propellant Usage Data and Propellant Budget Summary, Rendezvous Guidance Procedures Section (Code DM43), NASA, Johnson Space Center, Houston, Texas, 20 January 1994.
10. STS-51 FLIGHT CREW REPORT, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, in preparation.
11. Saunders, Penny E., *The First Flight Tests of GPS on the Space Shuttle*, Proceedings of ION Technical Meeting, San Diego, California, January 1994.

## **APPENDIX A**

### **DEFINITIONS**

#### **Body Axis Coordinate System**

A Cartesian right-handed coordinate system with origin at the Orbiter center of mass. The X-axis is parallel to the Orbiter structural body X-axis, positive towards the nose. The Z-axis is parallel to the Orbiter plane of symmetry, positive down with respect to the Orbiter fuselage. The Y-axis completes the right-handed system. (Ref 2: p. A-2)

#### **Earth-centered, Earth-fixed (ECEF) Coordinate System**

A Cartesian coordinate system. The center is at the Earth's center, The X-axis points toward the Greenwich meridian at the equator, the Z-axis points toward the North Pole, and the Y-axis completes the right-handed coordinate system. The coordinate system rotates with Earth.

#### **Local-vertical, Circular (LVC) Coordinate System**

The LVC coordinate system is a target-centered coordinate system. The Z-axis rotates with the target and is positive directed radially toward the Earth. The X-axis is curvilinear and positive in the general direction of orbit motion. The Y-axis is normal to the target's orbital plane and completes the right-hand coordinate system.

#### **Local-vertical, Local-horizontal (LVLH) Coordinate System**

The LVLH coordinate system is a target-centered coordinate system. The Z-axis rotates with the target and is positive directed radially toward the Earth. The X-axis is positive in the general direction of orbit motion. The Y-axis is normal to the target's orbital plane and completes the right-hand coordinate system.

## Midcourse Correction (MC) Maneuver

Rendezvous maneuver performed as necessary to ensure a correct intercept from Terminal Initiation (TI) to the target. (Ref 2: p. 3-8)

## M-50

Aries-mean-of-1950 Cartesian coordinate system. Inertial coordinate system with the origin at the center of the Earth. The epoch is the beginning of the Besselian year 1950 or Julian ephemeris date 2433282.423357. The X-axis points toward the mean vernal equinox of epoch, the Z-axis points toward the Earth's mean rotational axis of epoch and is positive north, and the Y-axis completes the right-handed system. (Ref 2: p. A-1)

## Remote Manipulator System

Mechanical arm on the payload bay longeron. It is controlled from the Orbiter aft flight deck to deploy, retrieve, or move payloads.

## Orbiter

Manned orbital flight vehicle of the Space Shuttle system.

## Orbiter Star Tracker

The Orbiter star tracker (STRK) is an image dissector electro-optical tracking device used to obtain precise angular measurements of selected stars and sun illuminated orbiting objects (targets). These measurements are used to determine the Orbiter's attitude in inertial space, and this data is used to align the inertial measurement units (IMUs). The STRK also provides angular data from the Orbiter to a target being tracked. (Ref 2: p. 2-1)

## Posigrade Burn

A posigrade burn is one which increases the speed but does not change the direction of the spacecraft at the point it is applied. A posigrade burn increases the energy, semi-

major axis, and period of the orbit. For example, if the orbit was originally circular, a posigrade maneuver will create an elliptical orbit, with the thrust point becoming perigee and apogee occurring 180 degrees of orbit travel away. (Ref 2: p. 1-11)

#### Radial Burn

A radial burn is one in which the thrust is applied in a direction perpendicular to the spacecraft's velocity vector and in the orbital plane of the spacecraft. (Ref 2: p. 1-18)

#### Retrograde Burn

A retrograde burn is one which decreases the speed but does not change the direction of the spacecraft at the thrust point. A retrograde burn decreases the energy, semi-major axis, and period of the orbit. For example, if the orbit was originally circular, a retrograde maneuver will create an elliptical orbit, with the thrust point becoming apogee and perigee occurring 180 degrees of orbit travel away. (Ref 2: p. 1-11)

#### RS-232

Standard serial port interface.

#### RS-422

Standard serial port interface.

#### Selected Availability

This is the name of the policy and implementation scheme by which users of GPS will have their accuracy limited to 100 meters 2dRMS horizontal and 156 meters 2dRMS vertical. (Ref 1: p. D-3)

#### Space Shuttle

Orbiter, external tanks, and solid rocket boosters.

## State Vector

The state vector defines the state of an orbiting body. The state vector consists of a position vector, a velocity vector, and a time.

## Terminal Initiation (TI)

The Terminal Initiation burn is designed to change the rendezvous approach from one of phasing to a direct intercept. The burn occurs approximately 5 minutes before orbital noon at the apogee of the post NCC orbit. (Ref 2: p. 3-7)

## Quaternion

A four parameter representation of a transformation matrix. It provides a numerical relationship between coordinate frames. Quaternions are used due to their convenient small size; that is, four parameters, as opposed to nine parameters in a transformation matrix (or direction cosine matrix). This greatly reduces computer computation time when numerous coordinate frames are involved in a transformation. (Ref 4: p. 1-1)

## WGS-84

World Geodetic System (1984), a mathematical reference ellipsoid used by GPS, having a semi-major axis of 6378.137 km and a flattening of  $1/298.257223563$ . (Ref 3: p. D-4) This model is used for orbit perturbation harmonic prediction.

WGS-84 is also used to refer to an earth-centered, earth-fixed (ECEF) coordinate system.

## ABBREVIATIONS

CCTV	Closed Circuit Television
COAS	Crew Optical Alignment Sight
DTO	Development Test Objective
ECEF	Earth-centered, earth-fixed coordinate system
GPS	Global Positioning System
LVC	Local Vertical Circular Coordinate System
LVLH	Local Vertical/Local Horizontal Coordinate System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KU-BAND	15.250 to 17.250 GHz
MC1	Midcourse Correction Maneuver 1
NC1	Nominal Correction 1 (Phasing Maneuver)
NCC	Nominal Corrective Combination (Maneuver)
NH	Nominal Height (Adjustment Maneuver)
ORFEUS	Orbiting Retrievable Far and Extreme Ultraviolet Spectrometer
PCMMU	Pulse Code Modulation Master Unit
PGSC	Portable GRID Systems Computer
PRCS	Primary Reaction Control System
PROX OPS	Proximity operations phase
R-bar ( $\bar{R}$ )	Radius vector axis
SA	Selected Availability
SEP	Separation
SPAS	Shuttle Pallet Satellite (German)
STRK	Orbiter Star Tracker
STS	Shuttle Transportation System
TI	Terminal Initiation
V-bar ( $\bar{V}$ )	Velocity vector axis

**APPENDIX B**  
**NPS SOFTWARE USER MANUAL**

**A. Organization**

This user manual is divided into two main parts.

- **Getting Started** - This first section describes how to install and start the NPS State Vector Comparison and Relative Motion Plotting program (referred to as the NPS software).
- **Tutorial** - The Tutorial is an introduction to the NPS software. The basic functions are described. For a detailed reference of functions, see Chapters V, VI, and VII of the Naval Postgraduate School thesis *NPS State Vector Analysis and Relative Motion Plotting Software for STS-51* by Lt. Lee Barker, USN, and Chapters III, IV, and V of the Naval Postgraduate School thesis *Theoretical Basis for State Vector Comparison, Relative Position Display, and Relative Position/Rendezvous Prediction* by Lt. Lester Makepeace, USN.

**B. Getting Started**

**1. NPS Software System Requirements**

The NPS software written for STS-51 DTO 700-6 was designed to run on the GRID 1530 Portable Grid Systems Computer (PGSC). The GRID 1530 PGSC contains an 80386 10 MHz CPU with an 80387 coprocessor, 4 Mbytes RAM, 200 Mbyte internal hard disk, 1.44 Mbyte 3.5" floppy drive, and RS-232/422 ports. The basic NPS software requires:

- IBM PC/XT or compatible MS-DOS computer, 10 MHz 80386 or faster recommended.
- MS-DOS Version 3.0 or later.



- 640 Kbytes of memory.
- Expanded Memory System recommended.
- A hard disk with 3 Mbytes of free space.
- A 1.44 Mbyte 3.5" floppy disk drive.
- Color Graphics Adapter (CGA), Enhanced Graphics Adapter (EGA), Video Graphics Array card (VGA), AT&T Graphics card (ATT), or compatible.
- Borland C++ Version 3.1 is required for executable file compiling. Borland C++ is not required to run the program.

In addition, certain variants of the NPS software may require additional hardware:

- An RS-232 port for software executable files utilizing the communications port for data reception or transmission.
- An RS-422 port or RS-232 port with RS-422/RS-232 converter is required for actual TANS GPS data collection.

## 2. Disk Contents

The NPS software is in compressed format on one or more 1.44 M byte 3.5" diskettes. The disk should include the following files:

nps\_32.zip      npsdat32.zip      nps\_exe.zip      pkunzip.exe

After installation, your \STS51NPS directory will contain the following subdirectories:

NPS	NPS_SIM	TANS	TANS_SIM	ORBMECH1
UTIL	INC	COM	TEST	OBJS

The STS51NPS directory contains the executable files, the current International Earth Rotation Service (IERS) data

file, configuration files and graphics drivers required by the program, and the 'makefile' used for program compilation.

STS51NPS files include:

att.bgi	cga.bgi	egavga.bgi	gpsalm.dat
iers.dat	litt.chr	makefile	nps.cfg
ans.chr	save_dat.bat	save_src.bat	tans.ild
tans.osc	turboc.cfg	worldmap.dat	worldmap.raw

The STS51NPS subdirectory contains executable files included in 'nps\_exe.zip' or additional files when created by the 'makefile' and Borland C++ Version 3.1. These include:

tans.exe	TANS, with NPS *** STS-51 Flight Version ***
tans_spd.exe	TANS, no NPS, with SPDRIVE
tans51sm.exe	TANS, with NPS, Sim from file
tans51fn.exe	TANS, no NPS, testing
tsim.exe	TANS device simulator
test_com.exe	Com-port tester (via built-in interrupts)
test_dev.exe	Com-port tester (via DOS device driver)
com_txrx.exe	Com-port two-way tester (via interrupts)
vmode.exe	Video-mode switcher
scom.exe	NPS Simulator, input com-port (built-in), no output
sdev.exe	NPS Simulator, input com-port (DOS driver), no output
sdev_dly.exe	NPS Simulator, **post-flight data replay**, input disk, no output
sprp.exe	NPS Simulator, input propagator, no output
sdsk.exe	NPS Simulator, input disk, no output

smem.exe	NPS Simulator, input memory (from disk), no output
prp_dsk.exe	NPS Generator, input propagator, output disk
prp_dev.exe	NPS Generator, input propagator, output com-port (DOS driver)
prp_com.exe	NPS Generator, input propagator, output com-port (built-in)
nps.exe	NPS Stand-alone, ** STS-51 Flight Version **
test_vec.exe	Vector functions tester
map_raw.exe	Map-file converter, text to raw
emm_test.exe	EMM functions tester
tansdump.exe	TANS data-file dumper
tans_vec.exe	TANS data-file vector dumper

The STS51NPS subdirectory will also include several '.nps' files created by the NPS software for storing plot information. These files can be deleted using the Dos command 'del \*.nps' prior to program start-up if the user does not want to see the plots from an earlier run.

The NPS subdirectory contains source code for the NPS program shell and plotting routines along with several other available functions. NPS files include:

dif_data.c	dif_plot.c	f_and_g.c	lsfilter.c
menuplot.c	nps.c	nps.h	nps_dap.c
nps_edit.c	nps_filt.c	nps_m50.c	nps_m50.h
nps_none.c	nps_plot.c	nps_pred.c	plane.c
py_data.c	py_plot.c	rv_data.c	rv_plot.c

The NPS\_SIM subdirectory contains source code for the program shells of the various versions of the program. These variants give the user the option to generate simulated state vectors and provide them to the program for program testing and simulation, generate state vectors and send them to a file or communications port, or read simulated or actual state vectors from a file, communications port, or memory. NPS\_SIM files include:

gcom_com.c	gcom_dev.c	gdisk.c	gnone.c
nps_main.c	scom.c	scom_dly.c	scom_old.c
sdisk.c	sdisk_2.c	sim.c	sim.h
smem.c	sprop.c	spropdsk.c	

The ORBMECH1 subdirectory contains source code for the Cowell propagator. The propagator includes M50 to WGS-84 and WGS-84 to M50 conversions, Jacchia atmosphere model and data files, the GEM-9 gravity model and forcing functions, and the Runge-Kutta fourth order integrator. ORBMECH1 files include:

amatrix.c	bgprop.c	cowell.c	der.c
gem9.c	getiers.c	gotpot.c	iers.h
itowgs84.c	jacatm.c	jacdat.c	m50rnp.c
rk4.c	rnpd.c	rnpm50.c	util.c
util.h			

The UTIL subdirectory contains source code for '.c' and '.asm' utilities used by the program. UTIL files include:

edit.c	edit_fld.c	edit_rt.c
emm_cons.c	emm_lowa.asm	emm_map.c
emm_mapm.c	key_get.asm	key_int9.asm
swap_d.asm	swap_f.asm	swap_s.asm
tdbl_hms.asm	tdbl_str.asm	tgps_utc.c
thms_dbl.asm	thms_str.asm	timer.asm
v_j2000.c	v_m50.c	v_matrix.c
v_quat.c	v_rvbar.c	v_vector.c

The TANS and TANS\_SIM subdirectories contain source code for the DTO 700-6 level I TANS GPS code written by Mike Arnie for NASA. This code is required for compiling the integrated program and is included for completeness but will not be discussed in this report. TANS files include:

almanac.c	almanac.h	constant.h	display.c
display.h	exit.c	log.c	log.h
m50_eph.c	m50_eph.h	main_dis.c	map_dis.c
map_dis.h	misc_dis.c	misc_dis.h	misc_fun.c
misc_fun.h	nps_fake.c	nps_if.c	para_dis.c
para_dis.h	prc_tpkt.c	prc_tpkt.h	spc_edit.c
spc_edit.h	tans_sys.c	tans51.c	tans51.h
tansfile.h	tanstime.c	tanstime.h	tglbtype.h
tns_io_f.c	tns_io_f.h	tnsprtfn.c	tnsprtfn.h
tnsprtsm.c			

TANS\_SIM files include:

tsim.c	tsim.h	tsim_2x.c	tsim_3x.c
tsim_5x.c	tsim_ex.c	tsim_pkt.c	

The INC subdirectory contains source code for 'include' files used at compile time. INC files include:

com_port.h	com_port.inc	const.h	edit.h
edit_rt.h	emm.h	keys.h	nps_if.h
orbmech1.h	packet.h	pkt_if.h	swap.h
tans_pkt.h	times.h	times.inc	types.h
vector.h			

The TEST subdirectory contains source code for various routines written to develop or test segments of the NPS code. TEST files include:

com_txrx.c	emm_test.c	map_raw.c	pkt_cln.c
pkt_show.c	tans_v2.c	tans_vec.c	tansdump.c
test.c	test_com.c	test_dev.c	test_vec.c
test1.c	test2.c	vect_old.c	vmode.c

The COM subdirectory contains source code for communications routines used by the program for sending and receiving data via the RS-232/422 port. COM files include:

com_inta.asm	com_intc.c	orb_file.c	orb_pkt.c
orb_port.c	tans_pkt.c	tansfile.c	tansport.c
tanspkt2.c			

The OBJS subdirectory is required by the 'makefile' and becomes the repository for object files created in the compilation process.

Updates to the NPS software may contain additional files not listed above.

### **3. Program Installation**

The following instructions assume drive A is a 1.44 M byte 3.5" floppy disk drive and drive C is the hard disk. If this is not the case, substitute the correct drive designation when following the instructions.

1. Create a directory on your hard disk called \STS51NPS and set it to the current directory:

```
C> C:
```

```
C> MKDIR \STS51NPS
```

```
C> CD \STS51NPS
```

2. Insert the floppy disk labeled STS\_51 NPS SOFTWARE into drive A and copy the contents into C:\STS51NPS:

```
C> A:
```

```
A> copy *.* C:
```

```
A> C:
```

Remove the disk from drive A.

3. Decompress the files on the hard disk using the included PKUNZIP.EXE:

```
C> pkunzip -d -o nps_32.zip
```

```
C> pkunzip -d -o npsdat32.zip
```

```
C> pkunzip -d -o nps_exe.zip
```

The subdirectories will be created and the files will automatically be decompressed in the appropriate directories.

4. Make an OBJS subdirectory in the STS51NPS directory:

```
C> MKDIR \OBJS
```

5. If using Borland C++ Version 3.1 to compile the NPS software, edit the 'BCC\_PATH', 'LIBPATH', and 'INCPATH' lines in the 'makefile' in the STS51NPS directory to reflect the correct path of 'bcc.exe', and the Borland C++ library and INCLUDE files.

This completes program installation.

#### 4. Configuration

The NPS software utilizes the memory configuration file, 'nps.cfg', located in the 'STS51NPS' directory, to reserve RAM for program use. The amount of memory available is system dependent. The amount of memory required by the program includes the executable file and the sum of the eight plot history buffers. Most systems will be limited to a total of 640 Kbytes. Each plot history buffer can hold a maximum of 64 Kbytes of data. The individual plot buffer sizes are set in the configuration file to the desired amount using any ASCII editor. The expanded memory option is also available by editing this file.

In addition to the 'nps.cfg' file, the graphics driver files (\*.bgi), character files (\*.chr), and the 'iers.dat' file must be present in the STS51NPS directory.

The 'iers.dat' file contains daily variations in the Earth's rotation vector from the Mean of 1950 (M-50) rotation



vector. The file contains data for a 90 day window. The program has been coded to fail if propagation is attempted outside the 90 day window. This hard coded failure can be removed but the propagations would assume no variation in the Earth's rotation vector and produce erroneous results. The 'iers.dat' file included with the software includes the STS-51 flight window. For current IERS data, contact the US Naval Observatory.

### C. Program Compiling

The included makefile and Borland C++ version 3.1 are used to compile the executable files. The source code contains pre-processor commands that allow customizing the compilation process for a particular program. If optimal rendezvous prediction functions are desired, the modifier -DRNDZ is used. If GPS data is available, the modifier -DGPS is used.

For example, to compile the post-flight analysis version of the software, "sdev\_dly.exe", including optimal rendezvous prediction functions and GPS data, the command line input would be:

```
C> make -DRNDZ -DGPS sdev_dly.exe
```

The "makefile" was created this way in order to allow compilation of the smallest possible executable file and still meet the operators requirements. **NOTE** - Versions compiled without the full capabilities of the NPS software will lack some of the features discussed in this manual.

#### **D. Starting the NPS Program**

Once the desired program version has been compiled using the makefile provided, the program can be executed. Program initiation procedures depends upon the version being run. For the integrated TANS GPS/NPS version, the NPS program is called from the TANS GPS program via a function key. For stand alone versions, typing the executable file name at the DOS prompt starts the program. Some versions prompt the user for information at start up (ie communications port number, data file paths/names, etc.). These are self explanatory.

The executable file "sdev\_dly.exe" included is used to play back post-flight data and is a good version to start with for learning the program operation. The program needs the "PCDecom" data packet file and the (optional) TANS GPS data file for the desired time period of the flight.

After executing "sdev\_dly" at the command prompt, the user is asked to provide the data source. The default is com-port two. Over-write the default with the data file name (ie 51\_r.dat) and press <ENTER>. The "SIM MENU" appears. Press <F1> to change the delay time between data points in the playback. <ESC> always takes the user back to the previous menu. From the "SIM MENU" press <alt S> to start or stop the data playback. <F6> is used to enter the NPS program.

## **E. A Program Tour**

Upon entry into the NPS program, the CRT displays a program title, four M-50 state vectors in raw received form and in post-propagated form, and an options menu containing a choice of plot displays and program settings menus. From the start-up display, the user can immediately see if data is being received from any of the four input state vector sources: (1) Orbiter INS, (2) Orbiter GPS, (3) Orbiter target, and (4) SPAS GPS.

### **1. Plot Menu Options**

The NPS software presents state vector comparisons in three different forms: (1) the  $\bar{R}/\bar{V}$  plot, (2) the state vector difference or 'Sawtooth' plot, and (3) the Pitch/Yaw display. These are selectable by use of the <FUNCTION> keys as noted on the menu bar.

#### **a. $\bar{R}/\bar{V}$ Plots**

The  $\bar{R}/\bar{V}$  plot displays the relative motion between a target and a chaser vehicle and is of particular importance during rendezvous and proximity operations. The relative motion plot uses a target-centered coordinate system in which the Z-axis rotates with the target and is positive directed radially toward the Earth, the X-axis is curvilinear and positive in the direction of orbit motion, and the Y-axis is out-of-plane and completes the right-hand coordinate system. (Figure B-1.)

Targ-INS 211/03:03:00.000  
 Orb-INS 211/03:03:30.000

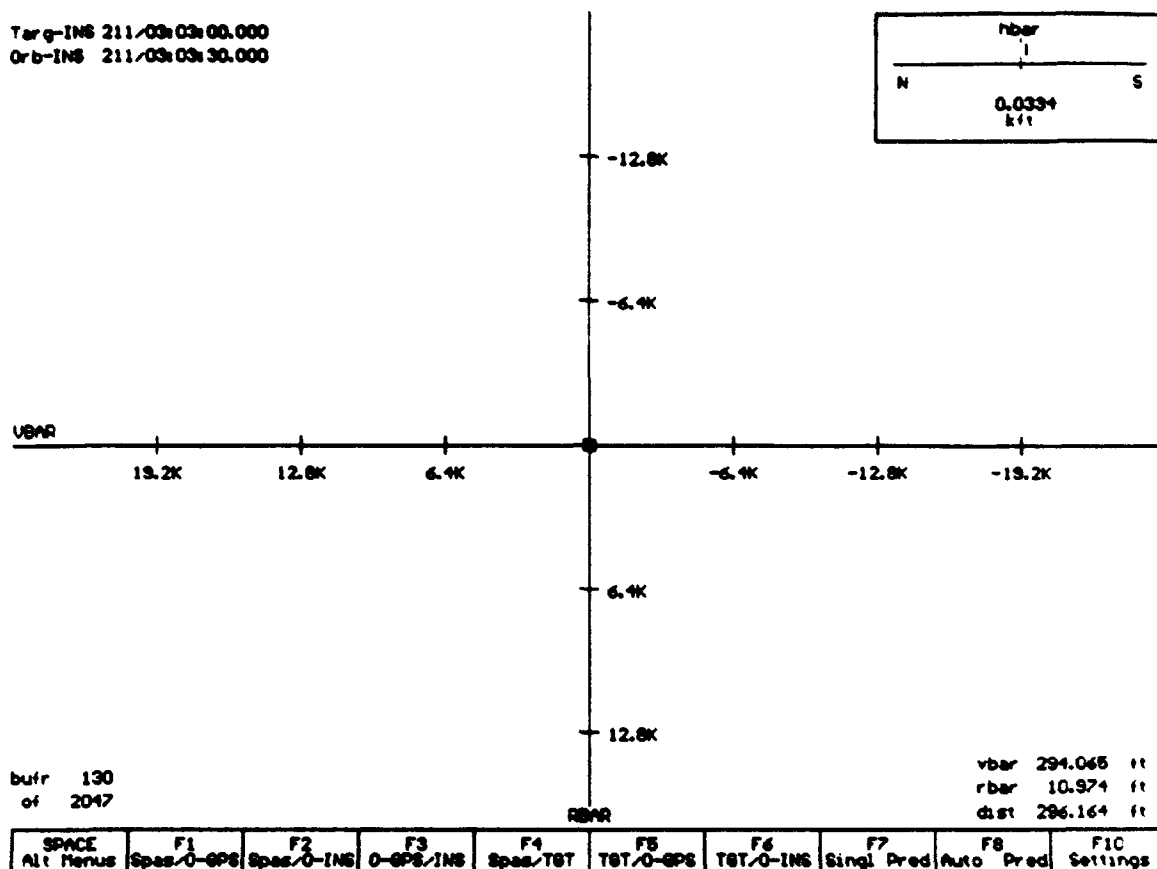


Figure B-1. R-bar/V-bar plot.

As a relative motion plot is normally used when the phase angle between the target and chaser is small, the coordinate systems used in a relative motion plot can be thought of as a local vertical/local horizontal (LVLH) coordinate system centered at the target. Thus, if the chaser were directly below the target on the radius vector to the Earth's center (R-bar), it would appear on the +Z axis of the relative motion plot. Similarly, if the chaser was ahead of

the target on the target's velocity vector ( $\bar{V}$ ), it would appear on the +X axis on the relative motion plot.

R-bar/V-bar plots are selectively available through the use of <FUNCTION> keys for the following target/chaser combinations of state vectors:

- <F1> SPAS GPS/Orbiter GPS
- <F2> SPAS GPS/Orbiter INS
- <F3> Orbiter GPS/Orbiter INS
- <F4> SPAS GPS/Orbiter target
- <F5> Orbiter target/Orbiter GPS
- <F6> Orbiter target/Orbiter INS

Plots <F3> and <F4> are not used as relative motion plots. These are used as another method of viewing the difference vector and show on the R-bar/V-bar display where the inertial navigation system thinks the Orbiter or SPAS is located relative to where GPS believes it is.

R-bar/V-bar plots can be selected from the main menu or from any other plot.

### ***b. Sawtooth Plots***

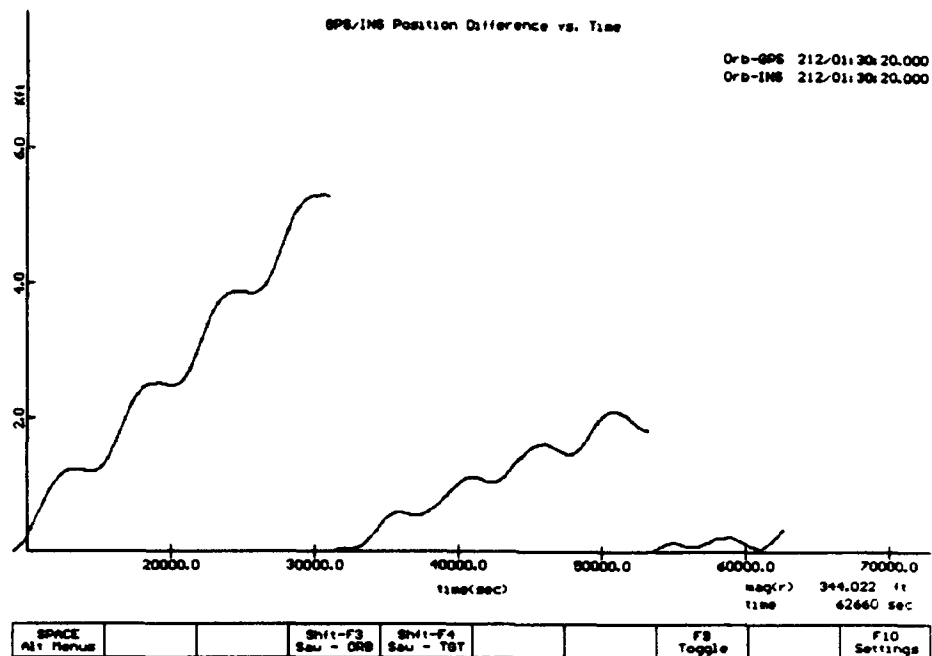
The relative difference or 'Sawtooth' plots display the difference in the magnitudes of the position or velocity vectors of two source vectors. (Figures B-2., B-3.) This is the tool used to quantify the error between the inertial navigation system and GPS. The results from these plots serve to validate or invalidate the use of GPS for Orbiter state vector updating. Two 'Sawtooth' plots are selectively available through the use of <SHIFT> <FUNCTION> keys:

- <SHIFT> <F3> Orbiter GPS/Orbiter INS
- <SHIFT> <F4> SPAS GPS/Orbiter target

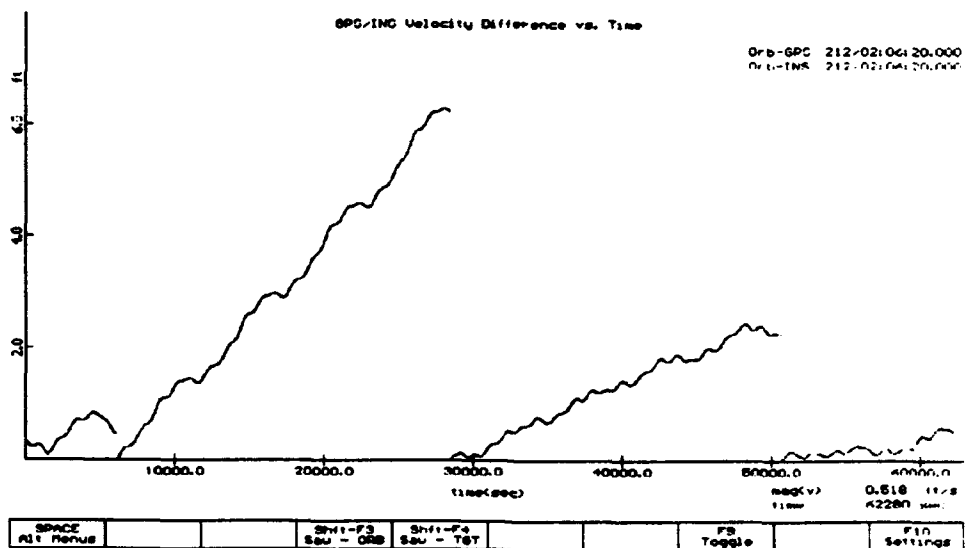
Once a 'Sawtooth' plot has been selected, the <F9> key toggles the plot between position and velocity vector comparison.

'Sawtooth' plots can be selected from the main menu or from any other plot.

'Sawtooth' plots are only available on program versions compiled with the -DGPS modifier.



**Figure B-2. Sample Position Difference Sawtooth plot.**



**Figure B-3. Sample Velocity Difference Sawtooth plot.**

### ***c. Pitch/Yaw Displays***

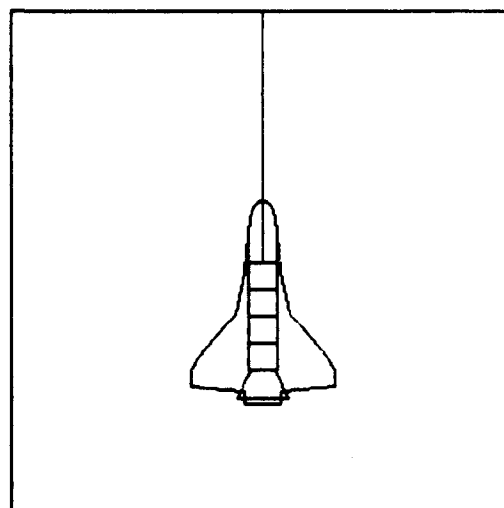
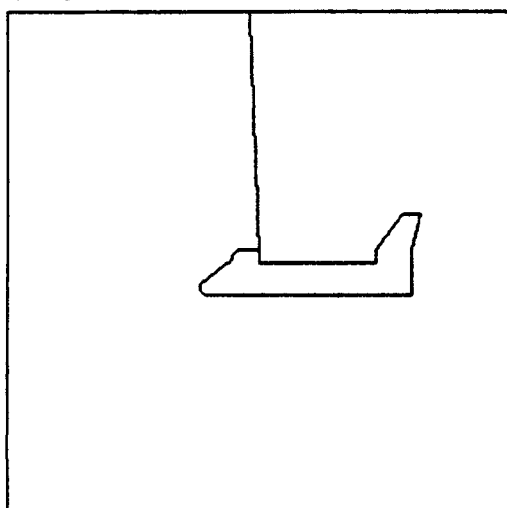
The Pitch/Yaw display gives a graphical presentation of the Orbiter (TOP and SIDE views) and a target pointing vector measured in pitch up from Orbiter body X-axis and yaw about the rotated Orbiter body Z-axis. (Figure B-4.) These angles are computed from the state vectors of the Orbiter and target, and the Orbiter quaternion. The Pitch/Yaw display is selected with the <ALT><F5> keys. Target/chaser state vector source combinations for the Pitch/Yaw plot can be selected with the <F9> key. Choices include:

- SPAS GPS/Orbiter GPS
- Orbiter target/Orbiter INS
- SPAS GPS/Orbiter INS

The Pitch/Yaw display can be selected from the main menu or from any other plot.



Spas-GPS 212/05:57:00.000  
 Orb-GPS 212/05:57:30.000



Pitch  
87.9

Distance  
3.3 Kft

Yaw  
0.2

ESC prev scrn	F1 Spas/O-GPS	F2 Spas/O-INS	F3 O-GPS/INS	F4 Spas/TGT	F5 TGT/O-GPS	F6 TGT/O-INS		F9 Toggle	F10 Settings
------------------	------------------	------------------	-----------------	----------------	-----------------	-----------------	--	--------------	-----------------

**Figure B-4. Sample Pitch/Yaw plot.**

#### ***d. Common Plot Features***

##### ***(1) Plot Labels***

All plots display the source state vector labels and the day/time of the most recent state vector from that source in the upper left hand corner of the display.

##### ***(2) Plot History Buffers***

The R-bar/V-bar and 'Sawtooth' plots each have a ring buffer for storing screen displays. This permits the user to leave a display and return to it later and see the historical motion or difference plot. The buffer sizes are set in the 'nps.cfg' configuration file. The maximum buffer size for each plot is 64 kbytes. The type of plot determines the number of bytes per data point required. The number of stored data points and the total number of points available for each plot is presented in the lower left hand corner of the display.

##### ***(3) Menu Bars***

Menu bars are used to assist the operator by providing a quick reference of available functions at the bottom of the display screen. The multiple menu bars are cycled through by pressing the <SPACE> bar on the keyboard.

## 2. Change Settings Menu Option

The NPS software package includes an interactive settings display that allows the user to manually input state vector information, alter default settings and data used by the propagator, change default screen and buffer settings, and adjust parameters in the thrust/predictor mode (discussed later). The 'Change Settings' displays are entered by pressing <F10>. <Page Up> and <Page Down> are used to display the desired settings page. There are seven settings pages:

- Sample and Step Rates
- Propagation Perturbations
- Orbiter Inertial State Vector
- Orbiter Target State Vector
- Orbiter GPS State Vector
- SPAS GPS State Vector
- Thrust Information

To change a setting on a display, the user must advance the curser to the desired setting using <TAB> or <ENTER>, overwrite the setting, and press <ENTER> prior to exiting the settings menu.

### Sample and Step Rates

Sample rates (in secs, 0 = fast)

plots show: 0

buffer save: 10

Orb-GPS filter save: 9

auto-rerun: N rerun delay: 300

Step size for propagation (in secs): 10

Maximum propagation time (in secs): 1800

Position predictor settings

Points to display: 40

Step size (secs): 60

Auto-mode delay: 8

#### *a. Sample and Step Rates Page*

The Sample and Step Rates page displays for editing plot show rate, buffer save rate, Orbiter GPS filter save rate, filter auto-rerun and rerun delay rate, propagator step size, maximum valid propagation time, and predictor settings. These parameters are defined below.

- Plot Show Rate - minimum time, in seconds, between screen writes of relative motion data.
- Buffer Save Rate - minimum time, in seconds, between plot data points recorded in the plot buffer files.
- Orbiter GPS Filter Save Rate - minimum time, in seconds, between TANS GPS state vectors stored in the GPS filter buffer.
- Orbiter GPS Filter Auto-Rerun Switch - ON/OFF switch for the filter auto-rerun feature. When active, the filter will execute and solve for a new filtered state vector to propagate from every rerun delay rate.

- Propagator Step Size - Time increment, in seconds, for the orbit propagator to step forward with.
- Maximum Valid Propagation Time - The maximum time, in seconds, that the propagator will propagate forward to.
- Position Predictor Points to Display - Number of Predictor points (defaulted to minutes) to display.
- Position Predictor Step Size - Step size, in seconds between predictor points.
- Predictor Auto-mode Delay - Delay, in seconds, between execution of automatic predictions when in AUTO PREDICT mode.

All times/rates are in seconds.

### Propagation Perturbations

Drag Mode on: Y

Ballistic numbers

Orbiter: 64

Payload: 64

Drag Information

Solar Flux: 185.959340379

Mean Solar Flux: 178.644769868

Geomagnetic Activity Index: 178.644769868

Propagator Information

Zonal Harmonics: 4

Tesseral Harmonics: 4

#### *b. Propagation Perturbations Page*

The Propagation Perturbation page allows editing of the current propagator parameters. These include:

- Drag Mode Status - ON/OFF switch for use of drag forces in the propagator. Default is ON.
- Orbiter/target Ballistic Numbers - Ballistic coefficients ( $m/C_D A$ ) for Orbiter and target used in determining drag forces.
- Solar Flux - The predicted F-10.7 solar activity index. This parameter affects the height of the atmosphere and, hence, the atmospheric drag.
- Mean Solar Flux - The 90 day average solar activity index. This parameter affects the height of the atmosphere and, hence, the atmospheric drag.
- Geomagnetic Activity Index - Measurement of the Earth's magnetic activity.
- Zonal/Tesseral Harmonics - The number of harmonic terms of the geopotential function to include in the gravity model (up to 30). Using more harmonics increases the accuracy of the propagator at the cost of increased computation time. Default is 4/4.

#### Orbiter GPS State Vector

Position x: 15432.6498524  
Kft y: -15172.9904659  
z: 0.971861685925

Velocity x: 17.9049415653  
Kft/s y: 18.1705472718  
z: -0.00226579476339

Time (ddd/hh:mm:ss): 211/06:30:00.000

Use Realtime Vector: Y

Apply time correction: N (delta in secs): 0

Source for GPS vector (0 GPS-M50, 1 GPS-WGS84): 0

Converter for WGS84-M50 (0 iload file, 1 computed): 0

#### c. State Vector Pages (4)

The State Vector Pages present and allow keyboard entry of each of the four state vector inputs. Definable parameters include:

- State Vector Position - M-50 inertial coordinates X, Y, and Z in kilofeet.
- State Vector Velocity - M-50 inertial velocity Vx, Vy, and Vz in kilofeet/sec.
- Time - The universal time stamp of the given state vector in day of the year, hour, minute, and second.
- Use Realtime Vector - An ON/OFF flag, determines whether or not to accept new state vectors from the TANS or the Orbiter's Navigation computer. This option would be turned off in the event of erroneous data from the online sources. Propagation would then take place from the last given or hand keyed state vector shown on the page. Default is ON.

- Apply Time Correction - An ON/OFF flag. OFF accepts the time tag of a state vector as is. ON modifies the time tag by a specified number of seconds. Default is OFF.

GPS source State Vector Pages include the following additional options:

- Source for GPS State Vector - Flag for determining source state vector coordinate system, M-50 or WGS-84. Default is M-50.
- Converter for WGS-84 to M-50 - Flag for determining conversion scheme of WGS-84 state vector to M-50 state vector. The 'iload file' option uses a data file with a conversion matrix for a specified recent date and the converter uses the matrix modified for the passage of time since the file date. The 'computed' option performs the complete WGS-84 to M-50 conversion for each state vector. The 'computed' option costs more computation time. Default is 'iload file'.



# Thrust information (ft/s)

		X-comp	Z-comp	
DAP-A	+x up	0.056000	0.009870	10 deg pitch
	-x down	-0.056000	0.009470	9.6 deg pitch
	+z out		0.055000	
	+z in		-0.076000	
(low-z)	+z out		0.052000	
	+z in		-0.076000	

DAP-B	+x up	0.016000	0.002820
	-x down	-0.016000	0.002700
	+z out		0.024000
	+z in		-0.034000
(low-z)	+z out		0.013000
	+z in		-0.034000

Future thrust: N      Note: future thrusts use the current  
 Now + 000/00:00:00.000      attitude, so answers are valid  
 or at 211/04:27:00.000      only during inertial hold, the  
                                  alternative is to apply thrust in  
                                  LVLH

Thrust in LVLH: N  
 Thrust on: N    Total thrust (XYZ):      0.0000      0.0000      0.0000

TTR (Do NOT pick whole orbit intervals!): 000/01:00:00.000

#### **d. Thrust Information Page**

The Thrust Information Page presents and allows keyboard entry of thrust vector data to be used by the predictor for displaying predicted relative motion based upon planned thruster firings. Parameters include:

- DAP-A/DAP-B Single Impulse Thrust Settings - Orbiter impulse thrust components for the Digital Auto-Pilot (DAP) A and B modes in feet per second. Default settings shown are actual values for the Orbiter thruster system.
- Future Thrust - An ON/OFF flag for future thrust relative motion predictor function. This function, when activated, allows the programming of thruster firings at a specified time in the future and displays the predicted relative motion on the R-bar/V-bar display along side the predicted motion of the Orbiter without thruster inputs.
- Future Thrust Activation Time - Time (ddd/hh/mm/ss) of desired thrust inputs relative to present time or in absolute UTC.
- Thrust in LVLH - An ON/OFF flag. ON indicates thruster inputs are in the LVLH coordinate system. OFF indicates thruster inputs are in Orbiter body coordinates. Default is OFF.
- Thrust On - ON/OFF flag indicating if the thrust function is active. Default is OFF.
- Total Thrust - Current thruster settings (XYZ) in feet per second.
- Time to Rendezvous (TTR) - Input time used by the program Rendezvous function for determining the optimal two impulse rendezvous thrusts. This time cannot be a multiple of the orbital period due to singularities in the function. Default is one hour.

#### **3. Additional Functions and Program Features**

An assortment of tools and features were included in the NPS software to provide additional information to the user. These features can be found on the menu bar at the

bottom of each plot. Multiple menu bars are available and can be cycled through by pressing the <SPACE> bar.

**a. Plot Scaling/Shifting**

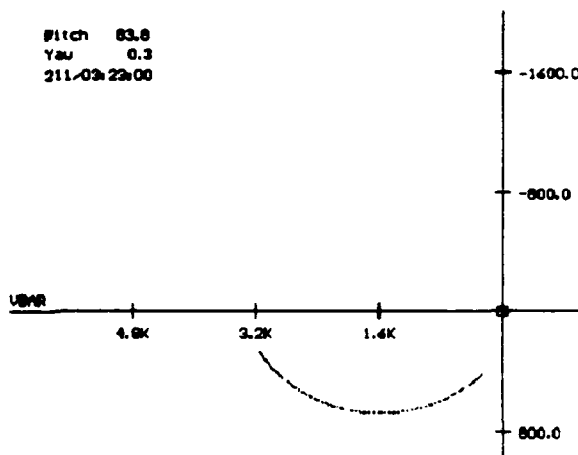
All R-bar/V-bar plots are independently scalable in the V-bar and R-bar direction. V-bar scaling is accomplished through the use of left and right <arrow> keys. Similarly, R-bar scaling is accomplished with the up and down <arrow> keys.

All 'Sawtooth' plots are scalable in magnitude (Y) and time (X) with the same arrow key scheme.

Plot shifting (left,right,up,down) is available on all R-bar/V-bar and 'Sawtooth' plots in a similar manner to plot scaling using the <CTRL><arrow> keys.

**b. Fast Pitch/Yaw**

If the program operator desires current pitch/yaw information while remaining in an R-bar/V-bar plot, the Fast Pitch/Yaw option, <ALT><F6>, is available. This option displays the target's current pitch and yaw angles in degrees and range in the



**Figure B-5. Sample Fast P/Y display.**

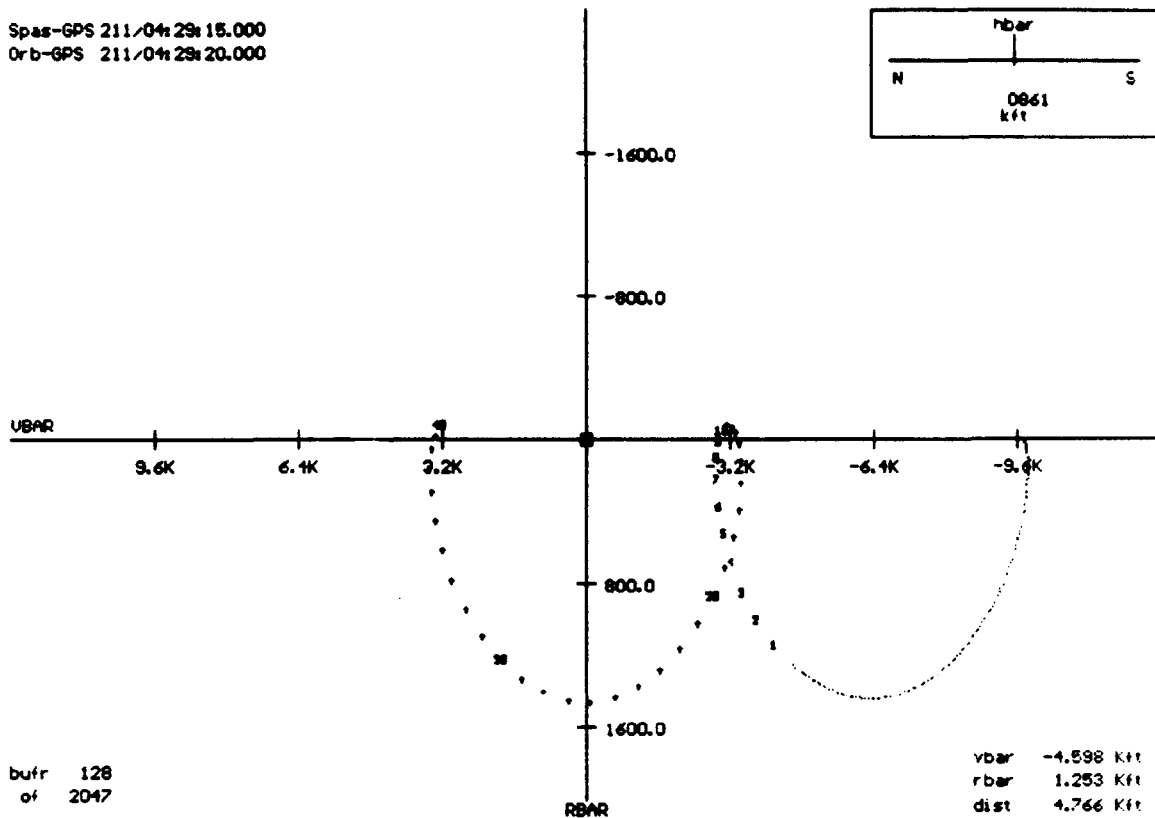
upper left hand corner of the current plot. (Figure B-5.) The Fast Pitch/Yaw function is a one time calculation at time of

execution, freeing up CPU time that is used for continuous pitch/yaw plotting in the <ALT><F5> Pitch/Yaw display.

### **c. Predictors**

Relative motion prediction is available by propagating the orbits ahead in time, allowing the user to see where the Orbiter will be in the future. The predictor function can be singularly executed for a one-time prediction, <F7>, or placed in an automatic update mode, <F8>. Initiation of the predictor displays the future position of the Orbiter on the screen in increments set by the user. The default settings are for ten points at one minute intervals. Both the number of points and the interval are adjustable by changing settings in the 'Sample and Step Rates' Page of the Change Settings <F10> function.

Spas-GPS 211/04:29:15.000  
 Orb-GPS 211/04:29:20.000



SPACE	F1	F2	F3	F4	F5	F6	F7	F8	F10
Alt Menu	Spas/O-GPS	Spas/O-INS	O-GPS/INS	Spas/TGT	TGT/O-GPS	TGT/O-INS	Singl Pred	Auto Pred	Settings

Figure B-6. Sample R-bar/V-bar plot with Predictor active.

#### **d. H-bar Display**

The R-bar/V-bar plot does not include out-of-plane component information. To provide this information visually, the H-bar, or momentum vector component of displacement can be displayed in the upper right hand corner of the screen. This display is toggled On or Off by the <ALT><F9> key. Default setting is On.

#### **e. GPS Filter**

The program contains a least squares fit GPS state vector filter developed by the McDonnell Douglas Corporation for NASA. TANS GPS state vectors provide high position accuracy on the order of a few hundred feet. TANS velocity vectors are not so accurate with errors on the order of one meter per second. For state vector propagations over short periods of time this is manageable. For longer propagations, the velocity error can cause significant errors in the propagated state. The purpose of a filter is to smooth the errors in position and velocity so that long propagations may be used in the event the TANS unit is switched off or stops functioning.

The filter is implemented by the <CTRL><F1> keys. This filter maintains a state vector buffer of the latest sixty Orbiter GPS state vectors. In the event that the Orbiter GPS state vector is lost due to a TANS failure, the GPS filter

function solves for a smoothed state vector to continue propagation with.

**f. DAP mode**

The Digital Autopilot, or DAP, mode allows the user to input thruster firings at present or future times to examine the resulting relative motion on the R-bar/V-bar plot. Activation of the DAP mode key <CTRL><F7> displays the DAP menu bar. The DAP mode menu consists of the following options:

- DAP-A/DAP-B Mode <a>/<b> - Toggles the DAP mode between DAP-A and DAP-B. This is essentially a change in the magnitudes of thruster impulses.
- Low Z Mode <z> - Toggles the DAP mode between normal mode where maximum efficiency of thruster is desired, and Low Z mode where plume impingement on a nearby object located above the Orbiter payload is the primary concern.
- LVLH/Body Mode <L> - Toggles the DAP mode to recognize thruster inputs in Orbiter body or Local Vertical Local Horizontal coordinates.
- Left/Right/Up/Down <arrow> keys - manually adjust the input thrust one impulse per key depression.
- Reset <SPACE> - Turns DAP mode display off.
- Redisplay <ENTER> - Refreshes display with current DAP mode thruster settings.

Thruster firings can be entered in LVLH or Orbiter body coordinates by the arrow keys in this mode, or alternatively from the Change Settings <F10> Thrust Information page. From the Thrust Information page, the time of thrust implementation may be set allowing for viewing of future Orbiter motion based on planned maneuvers.

NOTE - Upon entering the DAP mode, keyboard control is turned over to the DAP mode keyboard handler. NPS keyboard functions such as scaling or plot selection are not available in this mode. Keyboard control is returned to the NPS keyboard handler upon exiting DAP mode (<ESC> key).

***g. Rendezvous Predictor***

The rendezvous predictor function, <CTRL><F6>, is an application of the two impulse optimal rendezvous algorithm. When activated, the rendezvous predictor solves for the thrust required to complete the rendezvous and applies this thrust to the predictor. This function is available only when the program is compiled with the -DRNDZ modifier.



# INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2. Library, Code 052 Naval Postgraduate School Monterey CA 93943-5002	2
3. Chairman, (Code AA) Department of Aeronautics and Astronautics Naval Postgraduate School Monterey CA 93943-5000	1
4. Chairman, (Code SP) Space Systems Academic Group Naval Postgraduate School Monterey CA 93943-5000	1
5. Randy Wigley, (Code SP) Space Systems Academic Group Naval Postgraduate School Monterey CA 93943-5000	1
6. Dr. James H. Newman Astronaut Office (Code CB) NASA, Johnson Space Center Houston TX 77058	1
7. Capt. Frank L. Culbertson, Jr. Astronaut Office (Code CB) NASA, Johnson Space Center Houston TX 77058	1
8. Cdr. William F. Readdy Astronaut Office (Code CB) NASA, Johnson Space Center Houston TX 77058	1
9. Lt. Cdr. Daniel W. Bursch Astronaut Office (Code CB) NASA, Johnson Space Center Houston TX 77058	1

10. Maj. Carl E. Walz 1  
Astronaut Office (Code CB)  
NASA, Johnson Space Center  
Houston TX 77058
11. Col. Charles F. Bolden, Jr. 1  
Astronaut Office (Code CB)  
NASA, Johnson Space Center  
Houston TX 77058
12. Capt. Kenneth S. Reightler, Jr. 1  
Astronaut Office (Code CB)  
NASA, Johnson Space Center  
Houston TX 77058
13. Capt. Robert L. Gibson 1  
Astronaut Office (Code CB)  
NASA, Johnson Space Center  
Houston TX 77058
14. Lt. Cdr. Brent W. Jett 1  
Astronaut Office (Code CB)  
NASA, Johnson Space Center  
Houston TX 77058
15. Jack Brazzel 1  
McDonnell Douglas Space Systems  
Houston Division  
13100 Space Center Blvd  
Houston TX 77059-3556
16. Tom Silva 1  
The Telemetry Workshop Inc.  
17629 El Camino, Suite 206  
Houston TX 77058
17. Commanding Officer, Carrier Group Eight 1  
Attn: Lt. Lester B. Makepeace  
Unit 60104  
FPO AE 09501-4308
18. Code OC\CL 1  
Department of Oceanography  
Naval Postgraduate School  
Monterey CA 93943-5000
19. Code 53/SD 1  
Department of Mathematics  
Naval Postgraduate School  
Monterey CA 93943-5000

- |     |                                                                                                                               |   |
|-----|-------------------------------------------------------------------------------------------------------------------------------|---|
| 20. | NASA, Johnson Space Center<br>EE6/Penny Saunders<br>Houston TX 77058                                                          | 1 |
| 21. | Bill Ober (Code DM43)<br>NASA, Johnson Space Center<br>Houston TX 77058                                                       | 1 |
| 22. | Lt. Lee Barker (Code 31)<br>Department of Aeronautics and Astronautics<br>Naval Postgraduate School<br>Monterey CA 93943-5000 | 1 |