1998-06-01

An integrated INS/GPS navigation system for small AUVs using an asynchronous Kalman filter

Hernandez, Glenn C.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INTEGRATED INS/GPS NAVIGATION SYSTEM FOR SMALL AUVS USING AN ASYNCHRONOUS KALMAN FILTER

by

Glenn C. Hernandez

June 1998

Thesis Advisor: Xiaoping Yun
Second Reader: Eric R. Bachmann

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1998	Master's Thesis

4. TITLE AND SUBTITLE
An Integrated INS/GPS Navigation System for Small AUVs Using An Asynchronous Kalman Filter

5. FUNDING NUMBERS

6. AUTHOR(S)
Hernandez, Glenn C.

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

13. ABSTRACT *(maximum 200 words)*

A Small AUV Navigation System (SANS) is being developed at the Naval Postgraduate School. The SANS is an integrated INS/GPS navigation system composed of low-cost, small-size components. It is designed to demonstrate the feasibility of using a low-cost Inertial Measurement Unit (IMU) to navigate between intermittent GPS fixes. This thesis presents recent improvements to the SANS hardware and software. The 486-based ESP computer used in the previous version of SANS is now replaced by an AMD 586DX133 based PC/104 computer to provide more computing power, reliability and compatibility with PC/104 industrial standards. The previous SANS navigation filter consisting of a complementary constant gain filter is now aided by an asynchronous Kalman filter. This navigation filter has six states for orientation estimation (constant gain) and eight states for position estimation (Kalman filtered). Low-frequency DGPS noise is explicitly modeled based on an experimentally obtained autocorrelation function. Ocean currents are also modeled as a low-frequency random process. The asynchronous nature of DGPS measurements resulting from AUV submergence or wave splash on the DGPS antennas is also taken into account by adopting an asynchronous Kalman filter as the basis for the SANS software. Matlab simulation studies of the asynchronous filter have been conducted and results documented in this thesis.

14. SUBJECT TERMS	15. NUMBER OF PAGES
INS, GPS, AUV, Navigation, Kalman Filter	56
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

AN INTEGRATED INS/GPS NAVIGATION SYSTEM FOR SMALL AUVS USING AN ASYNCHRONOUS KALMAN FILTER

Glenn C. Hernandez
Lieutenant, United States Coast Guard
B.S., U. S. Coast Guard Academy, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1998

ABSTRACT

A Small AUV Navigation System (SANS) is being developed at the Naval Postgraduate School. The SANS is an integrated INS/GPS navigation system composed of low-cost, small-size components. It is designed to demonstrate the feasibility of using a low-cost Inertial Measurement Unit (IMU) to navigate between intermittent GPS fixes.

This thesis presents recent improvements to the SANS hardware and software. The 486-based ESP computer used in the previous version of SANS is now replaced by an AMD 586DX133 based PC/104 computer to provide more computing power, reliability and compatibility with PC/104 industrial standards. The previous SANS navigation filter consisting of a complementary constant gain filter is now aided by an asynchronous Kalman filter. This navigation filter has six states for orientation estimation (constant gain) and eight states for position estimation (Kalman filtered). Low-frequency DGPS noise is explicitly modeled based on an experimentally obtained autocorrelation function. Ocean currents are also modeled as a low-frequency random process. The asynchronous nature of DGPS measurements resulting from AUV submergence or wave splash on the DGPS antennas is also taken into account by adopting an asynchronous Kalman filter as the basis for the SANS software. Matlab simulation studies of the asynchronous filter have been conducted and results documented in this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr. Xiaoping Yun for guiding me through this enlightening experience. His patience and understanding gave me the motivation to keep moving forward and use common sense. I am much appreciative of all the support he has given to my efforts on the SANS project. Secondly, many thanks go to Dr. Robert McGhee and Eric Bachmann for their many insights during my research. Their breaths of knowledge never cease to amaze me. Finally, I would like to express my gratitude to Russ Whalen for trying to show me six years worth of knowledge and work on SANS hardware within a span of two weeks. His presence was sorely missed, but I appreciate the bay view office you left me, Thanks Russ.

This research was supported in part by Grant CDA-9729814 from the National Science Foundation to the Naval Postgraduate School.

DEDICATION

For Mom and Dad

I. INTRODUCTION

A. GENERAL

In vehicle navigation systems, integrated inertial measurement devices are commonly used to compensate for the loss of Global Positioning Systems (GPS) signals due to environmental blockages. These devices have been in use for some time in aircraft, ships, and large land vehicles. In the past, inertial information has been derived from expensive gyros that required large spaces relative to the size of the vehicle. As the size of the vehicles gets smaller, implementing these kinds of integrated devices can be quite expensive. Thus, research is being conducted on ways to compensate for this GPS signal loss with low cost off-the-shelf components. Current chip technology involving micromachined silicon has allowed progress in this particular area of navigation systems. The goal of this thesis is to show that a low cost integrated (Inertial Navigiation System) INS/GPS navigation system can be developed using commercially available components.

B. BACKGROUND

At the Naval Postgraduate School (NPS), a small autonomous underwater vehicle (AUV) known as Phoenix requires accurate navigation to report the position of shallow water mines or monitor the coastal environment [Ref. 3]. In the course of its missions, the vehicle periodically surfaces to obtain a Differential Global Positioning System (DGPS) fix and then submerges. While underwater, the Phoenix uses gyros, a depth sensor, a speed sensor, and compass heading to estimate its position. Once a shallow water mine has been detected with sonars, the location of the mine can be estimated based on the position of the AUV. Previous hardware and software were tested with

promising results. The current Phoenix navigation system has two faults. One, it does not account for water current, and two, its gyros are costly, bulky, power hungry, and prone to failure.

Over the last seven years, research has gone into solving these problems and the Small AUV Navigation System (SANS) is the outcome of this work. SANS, now in its third generation of development, has undergone a number changes [Ref. 3].

SANS technology is also becoming a more attractive solution to human motion tracking. Since the size of its components is getting smaller with each generation, improved portability is making it possible to track a human carrying a SANS device. For example, a small law enforcement team carrying SANS might enter a building with their location being sent back to a remote location outside the building so tacticians could better evaluate situations.

As suggested in the previous examples, the application of a device such as SANS is limitless. In this thesis, work has been narrowed to AUV navigation, at the same time knowing that applications beyond Phoenix are possible.

C. DESIGN TASKS

- Replace SANS computer;
- Replace Inertial Measurement Unit;
- Replace DGPS unit;
- Replace magnetometer unit;
- Replace speed sensor;
- Develop an asynchronous Kalman filter for SANS;
- Conduct simulation of the asynchronous Kalman filter for SANS;

- Test and evaluate any new components.

D. SCOPE, LIMITATIONS, AND ASSUMPTIONS

This thesis reports part of the findings of the seventh year of research in an ongoing research project. A new hardware configuration for SANS and a simulation of the SANS asynchronous Kalman filter are the main outcomes of this work. The scope of this thesis is to develop a new navigation system (SANS) for eventual installation as a replacement for the expensive and older technology gyros now used onboard the NPS Phoenix AUV.

E. ORGANIZATION OF THESIS

The purpose of this thesis is to present a prototype hardware platform and the filter theory required to meet the mission requirements of SANS. In this thesis, the term AUV is understood to include any small underwater vehicle (including humans) which can easily carry such a compact device (SANS). The major thrust of this thesis is to describe the latest developments of the SANS hardware and offer an optimal filter as a replacement to the original complementary filter.

Chapter II presents an overview of the SANS theory of operation and an explanation of the navigation filter. Chapter III presents the previous SANS hardware configuration and its current state. Chapter IV reviews Kalman filter basics, describes the SANS asynchronous Kalman filter equations and presents simulation results. Chapter V presents the thesis conclusions and provides recommendations for future research.

II. SANS THEORY OF OPERATION

A. INTRODUCTION

The objective of the SANS is to produce real time navigation information by integrating an Inertial Navigational System with Differential Global Positioning System (DGPS). Previous work on the SANS filter discussed in [Ref. 3,5 and 6] consists of a 14-state complementary filter designed to estimate north-east position. This chapter explains the theory of inertial navigation, real world problems involved with this type of navigation, and how SANS uses this theory to deliver navigation information.

B. INERTIAL NAVIGATION THEORY

To navigate between GPS fixes, SANS uses inertial sensors that measure linear accelerations in three orthogonal axes and angular rates about three orthogonal axes. In a perfect world where errors do not occur in sensors, angular rate could be integrated to give angular position or orientation. Integrating the acceleration twice would give position. Thus, position could be obtained in three dimensions given acceleration in three orthogonal axes. Theoretically, with no noise in sensor measurement, orientation and position could be found using only an Inertial Measurement Unit (IMU) given an initial attitude and position.

C. REAL WORLD PROBLEMS WITH INERTIAL NAVIGIATION

In practice, low cost angular rate sensors and linear accelerometers are prone to drift. Estimating position and orientation with an IMU requires knowledge of the drift characteristics of the sensors and how to correct them. Using sensors that are more accurate drives costs into the tens of thousands of dollars. Thus, a method is needed to

account for sensor drift. Angular rate sensor data are used as high frequency data and integrated once to obtain orientation information in a short period of time. Accelerometer data are used as low frequency data to correct drift in orientaion estimation. [Ref. 2]

Integrating linear acceleration twice gives linear position. Integration causes errors in position estimation to grow quadratically. DGPS and waterspeed sensors are used as low frequency devices to correct drift in position estimation. Once acceleration is integrated to give velocity, errors within this integration is corrected with waterspeed. When velocity is integrated to give position, DGPS is used correct position errors. Differences between DGPS position and IMU position are attributed to ocean current.

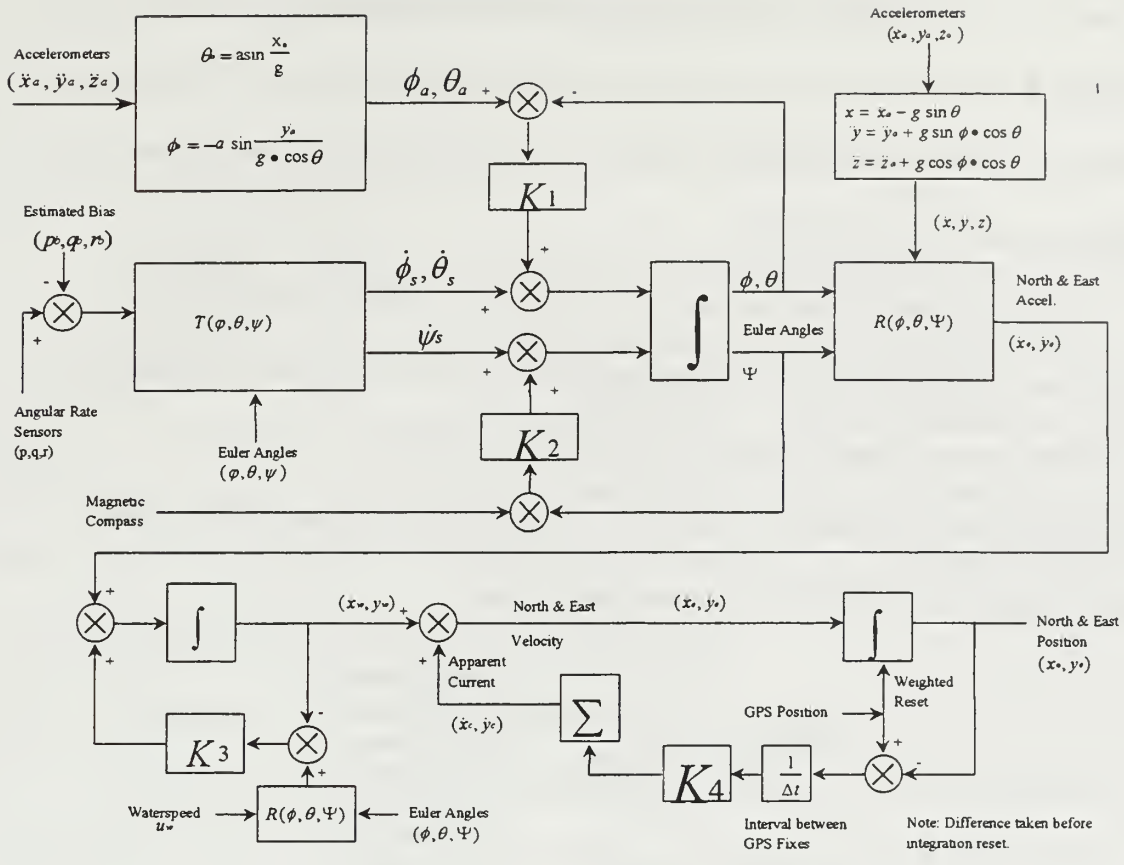


Figure 2.1: Previous SANS Filter

D. SANS NAVIGATION

SANS uses the principles of inertial navigation to estimate attitude. GPS and waterspeed data aid in position estimation. The previous SANS filter (Figure 2.1) is a 14-state complementary filter. Attitude estimation transforms all the body rates and angles of the AUV to earth-based coordinates, namely, north-east-down. The T transformation matrix in Figure 2.1 converts body angular rates to Euler rates and the R tranformation matrix transforms body-based linear accelerations and velocities to earth-based angles. Six states account for attitude, three for angular rate and three for

acceleration. The other eight states are north-east position, velocity, ocean current, and GPS bias. Position in the down axis is obtained using a pressure sensor. This thesis only focuses on the north-east coordinate estimation. The eight states were determined using constant gains for error correction.

SANS obtains accurate navigation information by integrating IMU data and DGPS data. While IMU data is sampled periodically, DGPS is available aperiodically due to asynchronous reacquisition time of satellite signals and asynchronous submergence-surfacing duration of the AUV. To optimally integrate IMU and DGPS data, an asynchronous Kalman filter is an ideal method.

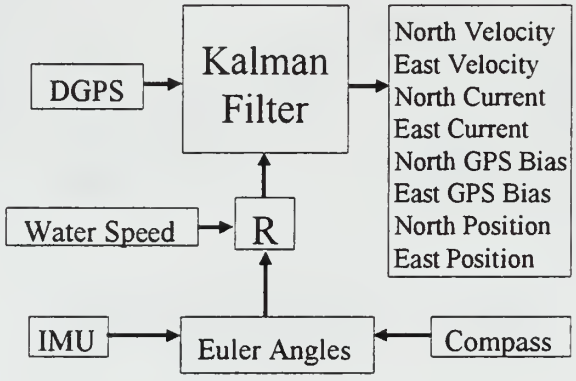


Figure 2.2: Current SANS Filter

Figure 2.2 shows the how eight states would be used in an asynchronous Kalman filter. The IMU and compass give the necessary Euler angles to break waterspeed into north-east vector using the AUV body to earth based coordinate R rotation matrix. Chapter IV goes into more detail as to how the filter was developed.

III. SANS SYSTEM CONFIGURATION

A. INTRODUCTION

The objective of SANS is to develop a low cost way to navigate between DGPS fixes using an IMU. The system should be small enough to fit into the Phoenix AUV. References [2,3,5,6, and 7] described the previous SANS. This thesis presents the current SANS. Most parts of the previous unit have been replaced with cheaper, smaller, and more powerful components. This chapter briefly describes the previous SANS and then details the current SANS configuration and the operation.

B. PREVIOUS SANS HARDWARE

The previous SANS (Figure 3.1) consisted of a 486SLC DX2 CPU module, an A/D converter, Systron Donner Inertial Measurement Unit (IMU), a waterspeed sensor , a Motorola ONCORE 8-channel receiver with an imbedded DGPS capability, and a Precision Navigation TCM2 Electronic Compass . These components were installed in a box measuring 6 x 17 x 3 inches (306 cubic inches). The previous version of SANS sampled at 40 Hz and required 12 VDC power. The electronic compass and power supply were located external to the box to reduce magnetic interference.

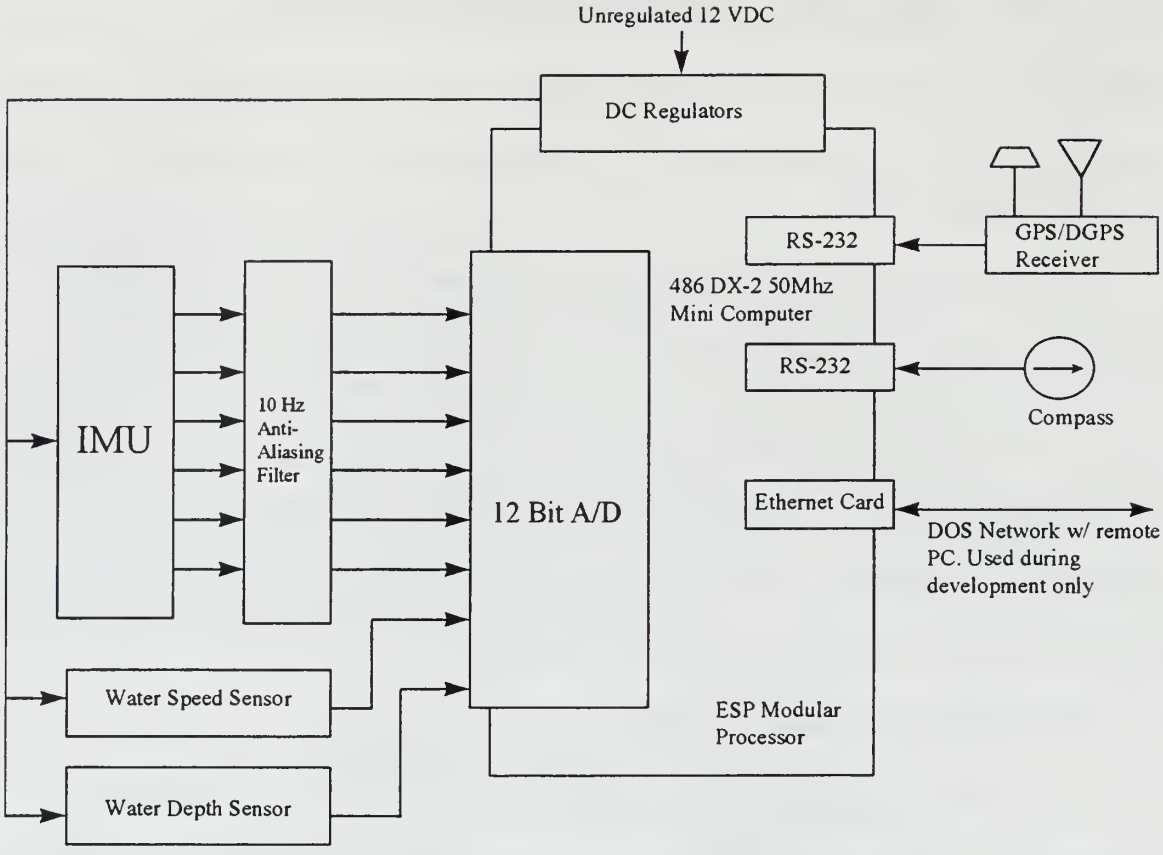


Figure 3.1: Previous SANS Configuration

C. CURRENT SANS HARDWARE

The current SANS configuration (Figure 3.2) is smaller, faster, and more flexible. The DGPS and electronic compass were kept from the old system. All other components were upgraded. This version of SANS is capable of sampling at an estimated rate of 100 Hz and requires only 5 VDC to operate. The current SANS configuration is approximately 150 cubic inches excluding the electronic compass and the power supply. The new components are a Real Time Devices 133 MHz AMD 586 PC/104, a Sealevel four port RS-232 module, a Crossbow six axis inertial measurement unit, and a SonTek Hydra water speed sensor.

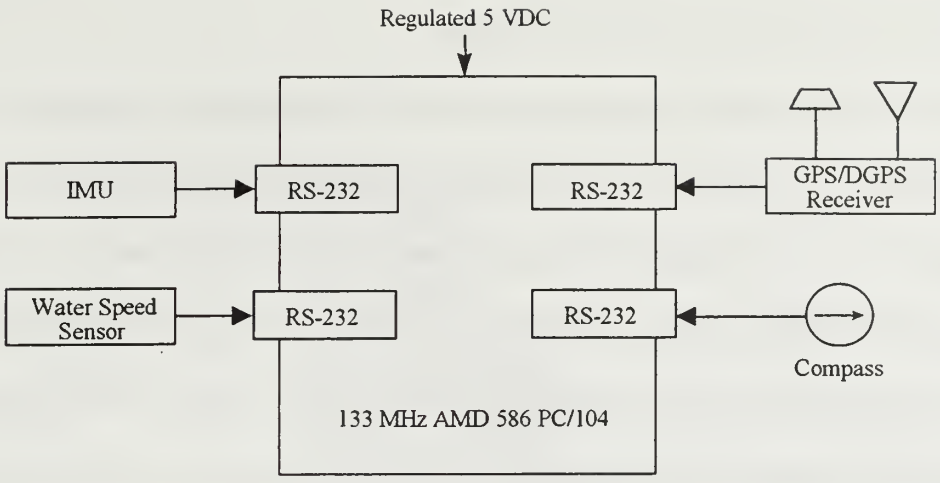


Figure 3.2: Current SANS Hardware Configuration

1. Real Time Devices DSV586DX133 PC/104

The Real Time Devices PC/104 module will process all data received from the sensors. Utilizing to the PC/104 standard adds flexibility to the system. The PC/104 CPU is small, relatively cheap at $600 per CPU and easy to expand with stacking modules. Over the past decade, this PC architecture has become an accepted platform for dedicated and embedded applications. PC/104 offers the full architecture, hardware and software compatibilities of the standard PC bus, but in compact (3.6 inches x 3.8 inches) stackable modules. [Refs. 8 and 14]

The Real Time Devices PC/104 has powerful features for its size. It operates on 5 VDC, as do all the other SANS components. Also, this PC/104 has a 12 MB disk on a chip (expandable to 72 MB) that will store the SANS code and any data it accumulates. A Integral Viper 170 MB hard drive is also available for the PC/104 if more storage is required, i.e. navigation data for post processing. [Ref. 8]

2. Four RS-232 Port Serial I/O Module

Two serial ports usually come standard on PC's. However four serial ports were needed in SANS. The Sealevel C4-104 serial I/O module provides SANS with four serial ports each with its own memory addresses and interrupt request assignments. All the sensors, the IMU, DGPS, water speed sensor, and electronic compass output to a RS-232 format. By using this module, an A/D board is not necessary to convert analog sensor data to a digital representation. Most importantly, removing the A/D board from SANS reduces its size by approximately 60 cubic inches. [Ref. 9]

3. Crossbow DMU-VG Six Axis Inertial Measurement Unit

The Crossbow DMU-VG six axis IMU consists of three rate gyros, three accelerometers, and a 12 bit A/D board all in a 3 x 3.375 x 3.250 inch box. It operates in one of two output modes, a scaled sensor mode and a voltage mode. In sensor mode, roll and pitch can be obtained directly from an on-board processor. This capability reduces the amount of computing for the PC/104. The original SANS code calculated the roll and pitch from the angular rates and acceleration readings of the Systron Donner IMU [Ref. 2]. The Crossbow unit produces a data packet (Table 3.1) which can be used to replace this portion in the SANS code. [Ref. 10]

Note: Most Significant Bit (MSB) Least Significant Bit (LSB)

0	Header (255)
1	Roll (MSB)
2	Roll (LSB)
3	Pitch (MSB)
4	Pitch (LSB)
5	Roll Rate X (MSB)
6	Roll Rate X (LSB)
7	Pitch Rate (MSB)
8	Pitch Rate Y (LSB)
9	Yaw Rate Z (MSB)
10	Yaw Rate (LSB)
11	Acceleration X (MSB)
12	Acceleration X (LSB)
13	Acceleration Y (MSB)
14	Acceleration Y (LSB)
15	Acceleration Z (MSB)
16	Acceleration Z (LSB)
17	Temp Sensor Voltage (MSB)
18	Temp Sensor Voltage (LSB)
19	Time (MSB)
20	Time (LSB)
21	Checksum

Table 3.1: Crossbow IMU Data Packet Format [Ref. 9]

4. SonTek Hydra Water Speed Sensor

The SonTek Hydra water speed sensor is a single point, high resolution, 3D Doppler current meter. This device uses one transmitter and 3 acoustic receivers which are aligned to intersect with the transmit beam pattern at a common sampling volume. The velocity measured by each receiver is referred to as the bistatic velocity, and is the projection of the 3D velocity vector on the bistatic axis of the acoustic receiver. An RS-

232 output makes intergration with the PC/104 simple. What makes this water sensor unique is its ability to report velocity data in the Earth (North-East-Down) coordinate system with the aid of an internal compass and tilt sensor. The velocity in Earth coordinates is fed directly into the asynchronous Kalman filter described in the next chapter. [Ref. 11]

5. McKinney Technology DGPS Receiver

This GPS/DGPS receiver package is specially designed to operate in the Monterey Bay area. The contained Motorola Oncore receiver is capable of tracking up to eight satellites simultaneously. It can provide position accuracy of better than 24 meters Spherical Error Probable (SEP) without Selective Availability (SA) and 10 meters (SEP) with SA. Typical-Time-To-First Fix (TTFF) is 18 seconds with a reacquisition time of 2.5 seconds. [Ref. 12]

6. Precision Navigation TCM2 Electronic Compass Sensor Module

The TCM2 consists of a three-axis magnetometer and a two-axis tilt sensor with a small A/D board to output roll, pitch, heading, and a three dimensional magnetic field measurement. It is accurate to within one half of a degree in level operation. Reference [5] describes calibration of the compass and its error characteristics. Its size is 2.5 x 2 1.1 inches. It requires 5 VDC and 15-22 mA. [Ref. 13]

D. FUTURE SANS HARDWARE

As of this writing research is proceeding regarding other possible components of SANS. A method to transmit data from SANS to a remote location is desired. This has led to the purchase of the Proxim RangeLAN2 7402 PC card. The RangeLAN2 is capable of transmitting data at 1.6 Mbps through a PCMCIA type II card format at

distances up to 1000 feet. The SANS PC/104 already comes with a PCMCIA module that fit 2 cards. With this feature, data received from the sensors or processed navigation information can be observed remotely. [Ref. 15]

Research into a smaller IMU showed that a device with a three axis accelerometer, three axis angular rate sensor, and three axis magnetometer all in a package the size of wrist watch is approximately two years away. Demand for such a device is slowly increasing, but not fast enough for the commercial industry to mass-produce them at this time. Finally, the current SANS DGPS module is three years old. Enough advances have been made in the GPS technology that would warrant changing out the DGPS package for a system that is smaller and more accurate. Thus far, research has offered two possible products. One is Rockwell Semiconductor's NAVCARD LP, which comes in a PCMCIA card format [Ref. 16]. The other is Trimble's Pathfinder with ASPEN software also in a PCMCIA card format [Ref. 17]. No information was found on the DGPS radio receivers integrated with GPS on PCMCIA cards. User defined settings of the receiver frequency are set for the area of GPS operation.

As with any hardware device, its effectiveness is usually determined by the quality of its software. The SANS code [Ref. 2] is currently written in C++ and utilizes the constant gain complementary filter method found in Chapter II. Using the configuration described in this chapter, the next task is to implement an asynchronous Kalman filter to integrate the sensor data. The result of this filter work is described in Chapter IV.

IV. SANS ASYNCHRONOUS KALMAN FILTER

A. INTRODUCTION

This chapter discusses the development of a discrete time asynchronous Kalman filter that estimates the eight states of the SANS model after attitude estimation. A brief discussion will be provided on Kalman filter basics leading into a derivation of the SANS Kalman filter equations. Finally, results of the SANS asynchronous Kalman filter Matlab simulation will be presented.

B. KALMAN FILTER BASICS

The discrete Kalman filter is a recursive predictive update technique used to estimate the states of a process model. This state-space filter method has two main features (1) vector modeling of the random processes under consideration and (2) recursive processing of the noisy measurement (input) data [Ref.1]. Given some initial estimates, it allows the states of a model to be predicted and adjusted with each new measurement, providing an estimate of error at each update. It incorporates the effects of measurement noise and modeling noise in its computational structure.

A model of a random process can be described as:

$$\mathbf{x}_{k+1} = \phi \mathbf{x}_k + \mathbf{w}_k \tag{4.1}$$

In cases where the relationship between model and its measurements is linear, the observation (measurement) of the random process model (Eq. 4.1) becomes:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{4.2}$$

Throughout the remainder of this chapter, various terms are used to better explain the Kalman filter. The following notations are consistent with Reference [1]:

$\mathbf{x}_k$ (n x 1) Vector containing actual states of a process model at time t_k.

$\hat{\mathbf{x}}_k$ (n x 1) Vector containing the current estimated states at t_k.

$\hat{\mathbf{x}}_k^-$ (n x 1) Vector containing the estimated states at time *t* before updating with measurement.

$\hat{\mathbf{x}}_{k+1}^-$ (n x 1) Vector containing the estimated states at the next time sample.

$\mathbf{P}_k$ (n x n) Matrix containing the error covariance for $\hat{\mathbf{x}}_k$.

$\mathbf{P}_k^-$ (n x n) Matrix containing the error covariance for $\hat{\mathbf{x}}_k^-$.

$\mathbf{P}_{k+1}^-$ (n x n) Matrix containing the error covariance for $\hat{\mathbf{x}}_{k+1}^-$.

$\mathbf{z}_k$ (m x 1) Vector containing the measurement at time t_k.

$\mathbf{H}_k$ (m x n) Matrix giving the ideal (noiseless) connection between the measurement and the state vector at time t_k.

$\mathbf{R}_k$ (m x n) Matrix denoting the measurement error covariance, which must be known or estimated *a priori.*

ϕ_k (n x n) Matrix containing the model relating $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k+1}^-$ in the absence of a forcing function (if $\hat{\mathbf{x}}_k$ is a sample of a continuous process, ϕ_k is the usual state transition matrix).

$\mathbf{w}_k$ (n x 1) Vector whose elements are white sequences with known covariance structure to be used with $\hat{\mathbf{x}}_k$.

$\mathbf{v}_k$ (m x 1) Vector whose elements are the measurement errors associated with $\mathbf{z}_k$ - assumed to be a white sequence with known covariance structure and having zero cross-correlation with the $\mathbf{w}_k$ sequence.

$\mathbf{Q}_k$ (n x n) Matrix containing the covariance of $\mathbf{w}_k$.

$\mathbf{K}_k$ (n x n) Kalman Gain matrix that relates the amount of influence that the error between $\hat{\mathbf{x}}_k^-$ and $\mathbf{z}_k$ has in deriving $\hat{\mathbf{x}}_k$.

q white noise variable with zero mean and variance of element being modeled.

The discrete Kalman filter basically contains five recursive equations. Beginning with a prior estimate, the noisy measurement $\mathbf{z}_k$ is used wth a blending factor $\mathbf{K}_k$ (yet to be determined) to improve the the estimate as follows;

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \qquad (4.3)$$

$\mathbf{K}_k$ (known as the Kalman gain) is now needed to find the optimal estimate $\hat{\mathbf{x}}_k$. It takes the error covariance (mean-square error) between the current state $\mathbf{x}_k$ and the estimated state $\hat{\mathbf{x}}_k$ and applies it with $\mathbf{H}_k$ and $\mathbf{R}_k$ resulting in

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{4.4}$$

Once the Kalman gain $\mathbf{K}_k$ minimizes the mean-square estimation error, a new covariance matrix can be computed for $\hat{\mathbf{x}}_k$ (Eq. 4.3) with

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- \tag{4.5}$$

The updated estimate $\hat{\mathbf{x}}_k$ is projected ahead using the state transition matrix ϕ_k. Unlike Eq. (4.1) the noise vector $\mathbf{w}_k$ does not affect the projected states because it has zero mean and is white (zero correlation with any previous $\mathbf{w}_k$). Thus,

$$\hat{\mathbf{x}}_{k+1}^- = \phi_k \hat{\mathbf{x}}_k \tag{4.6}$$

Finally, the projected error covariance for $\hat{\mathbf{x}}_{k+1}^-$ uses the updated error covariance $\mathbf{P}_k$ from Eq. (4.5), the covariance of $\mathbf{w}_k$ in Eq. (4.1), denoted as $\mathbf{Q}_k$, and ϕ_k the state transition matrix to form the following:

$$\mathbf{P}_{k+1}^- = \phi_k \mathbf{P}_k \phi_k^T + \mathbf{Q}_k \tag{4.7}$$

Equations (4.3 – 4.7) can be placed into an algorithm that can loop infinitely. This Kalman filter loop is shown in Figure (4.1).

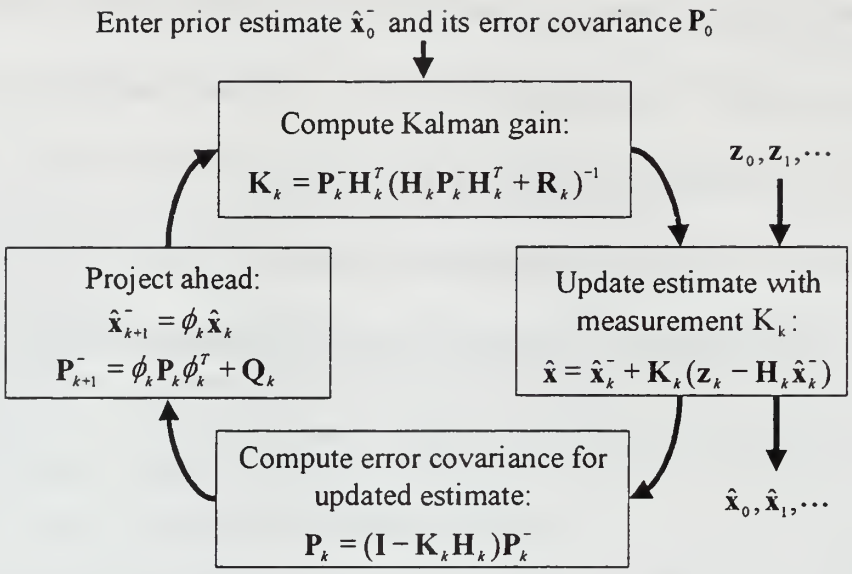


Figure 4.1: Kalman filter loop from Ref. [1]

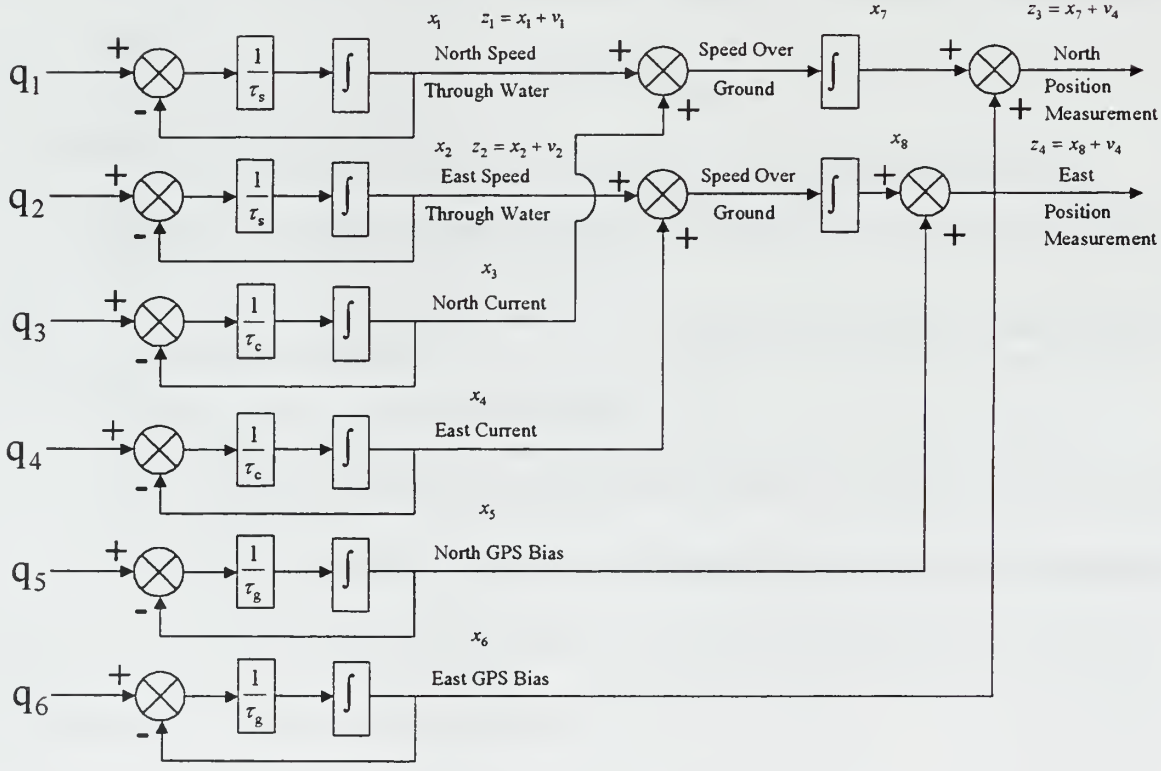


Figure 4.2: SANS Process Model (after attitude estimation)

C. DERIVATION OF SANS KALMAN FILTER EQUATIONS

The SANS asynchronous Kalman filter consists of eight states, north position, east position, north velocity, east velocity, north current, east current, north GPS bias, and east GPS bias. Figure 4.2 shows the process model developed by the SANS team [Ref. 4]. This model contains the eight states known as the vector $\mathbf{x}_k$. Equations 4.8 through 4.15 show the derivation of the state equations for the SANS process model.

$$\dot{x}_1 = -\frac{1}{\tau_1}x_1 + \frac{1}{\tau_1}q_1 \tag{4.8}$$

$$\dot{x}_2 = -\frac{1}{\tau_1}x_1 + \frac{1}{\tau_1}q_2 \tag{4.9}$$

$$\dot{x}_3 = -\frac{1}{\tau_2}x_3 + \frac{1}{\tau_2}q_3 \tag{4.10}$$

$$\dot{x}_4 = -\frac{1}{\tau_2}x_4 + \frac{1}{\tau_2}q_4 \tag{4.11}$$

$$\dot{x}_5 = -\frac{1}{\tau_3}x_5 + \frac{1}{\tau_3}q_5 \tag{4.12}$$

$$\dot{x}_6 = -\frac{1}{\tau_3}x_6 + \frac{1}{\tau_3}q_6 \tag{4.13}$$

$$\dot{x}_7 = x_1 + x_3 \tag{4.14}$$

$$\dot{x}_8 = x_2 + x_4 \tag{4.15}$$

From these equations, ϕ (state transition matrix) was developed.

$$\phi = \begin{bmatrix} e^{\frac{-2|\Delta|}{\tau_1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{\frac{-2|\Delta|}{\tau_1}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{\frac{-2|\Delta|}{\tau_2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{\frac{-2|\Delta|}{\tau_2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{\frac{-2|\Delta|}{\tau_3}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{\frac{-2|\Delta|}{\tau_3}} & 0 & 0 \\ \tau_1\left(1-e^{\frac{-2|\Delta|}{\tau_1}}\right) & 0 & \tau_2\left(1-e^{\frac{-2|\Delta|}{\tau_2}}\right) & 0 & 0 & 0 & 0 & 0 \\ 0 & \tau_1\left(1-e^{\frac{-2|\Delta|}{\tau_1}}\right) & 0 & \tau_2\left(1-e^{\frac{-2|\Delta|}{\tau_2}}\right) & 0 & 0 & 0 & 0 \end{bmatrix}$$

The next item needed for the Kalman filter was the $\mathbf{Q}_k$ matrix. Equation 4.16 is the equation for $\mathbf{Q}_k$, the mean of model noise squared.

$\mathbf{w}_k$ is the plant noise with covariance $\mathbf{Q}_k$,

$$\mathbf{Q}_k = E[\mathbf{w}_k \mathbf{w}_k^T] \tag{4.16}$$

$$\mathbf{Q}_k = \begin{bmatrix}
\frac{1}{2\tau_1}\left(1-e^{-\frac{2|\Delta t|}{\tau_1}}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2\tau_1}\left(1-e^{-\frac{2|\Delta t|}{\tau_1}}\right) & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2\tau_2}\left(1-e^{-\frac{2|\Delta t|}{\tau_2}}\right) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2\tau_2}\left(1-e^{-\frac{2|\Delta t|}{\tau_2}}\right) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2\tau_3}\left(1-e^{-\frac{2|\Delta t|}{\tau_3}}\right) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2\tau_3}\left(1-e^{-\frac{2|\Delta t|}{\tau_3}}\right) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

and $\mathbf{v}_k$ is measurement noise with covariance $\mathbf{R}$,

$$\mathbf{R} = \begin{bmatrix} .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ with DGPS signal}$$

$$\mathbf{R} = \begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix} \text{ without DGPS signal}$$

The **H** matrix is the noiseless connection between measurement and the state vector a time *t*. For the SANS asynchronous Kalman filter, two **H** matrices describe this connection, one for samples with DGPS the other for samples without DGPS.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \text{ with DGPS signal}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ without DGPS signal}$$

Using the main components of the Kalman filter in the previous section, the filter was tested in Matlab.

D. SANS KALMAN FILTER SIMULATION

The SANS asynchronous Kalman filter was coded in MATLAB® [Ref. 18] by applying the matrices developed in section C to the prediction and correction equations as outlined in Equations 4.1 through 4.7. Simulation results of the SANS Kalman filter are presented in the following section. The source code for the SANS Kalman filter is presented in Appendix A.

The model is assumed stationary throughout the simulation. Noise is generated from the process model and measurement errors. The following figures show the results of simulation over a six-minute period sampling at a rate of 100 Hz.

Figure 4.3 shows the north position versus time result in the six-minute simulation. The Δ symbols represent the colored noise sampling of DGPS position. The continuous line represents the actual model position, and the dotted line represents the estimated Kalman filter position of the model.

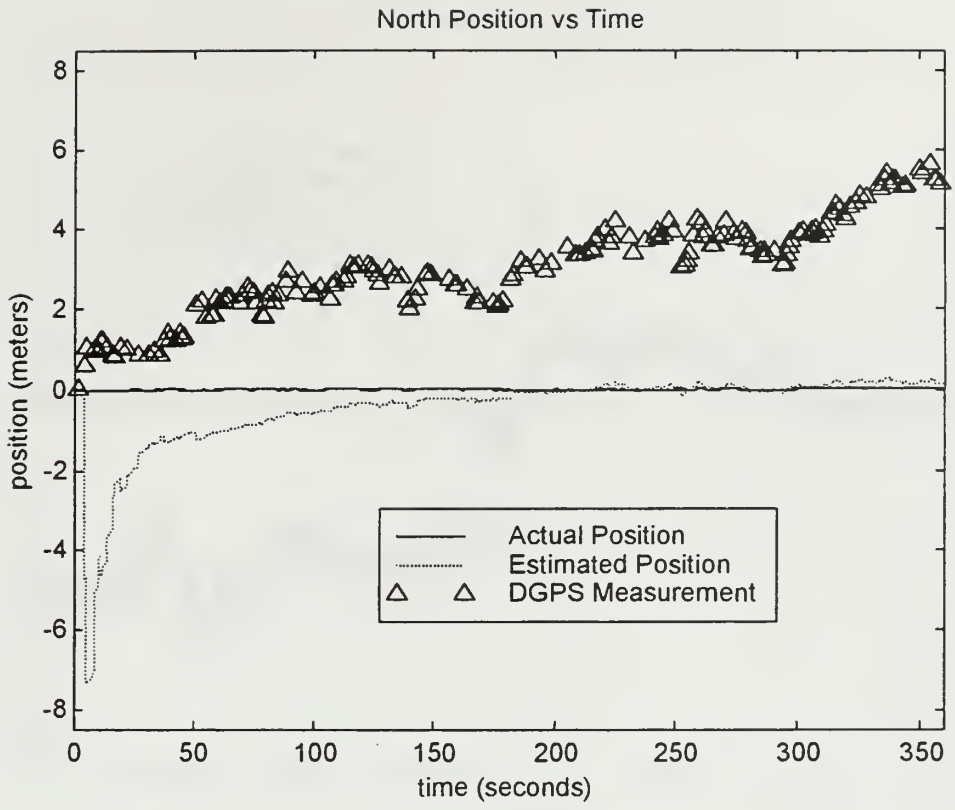


Figure 4.3: Plot of North Position vs. Time

Notice the initial jumps in the estimated position (dotted line) and how the estimate approaches the actual model state within the first 200 seconds although GPS information is drifting off. This behavior shows that the Kalman gain for GPS is decreasing and more reliance is being given to velocity measurement.

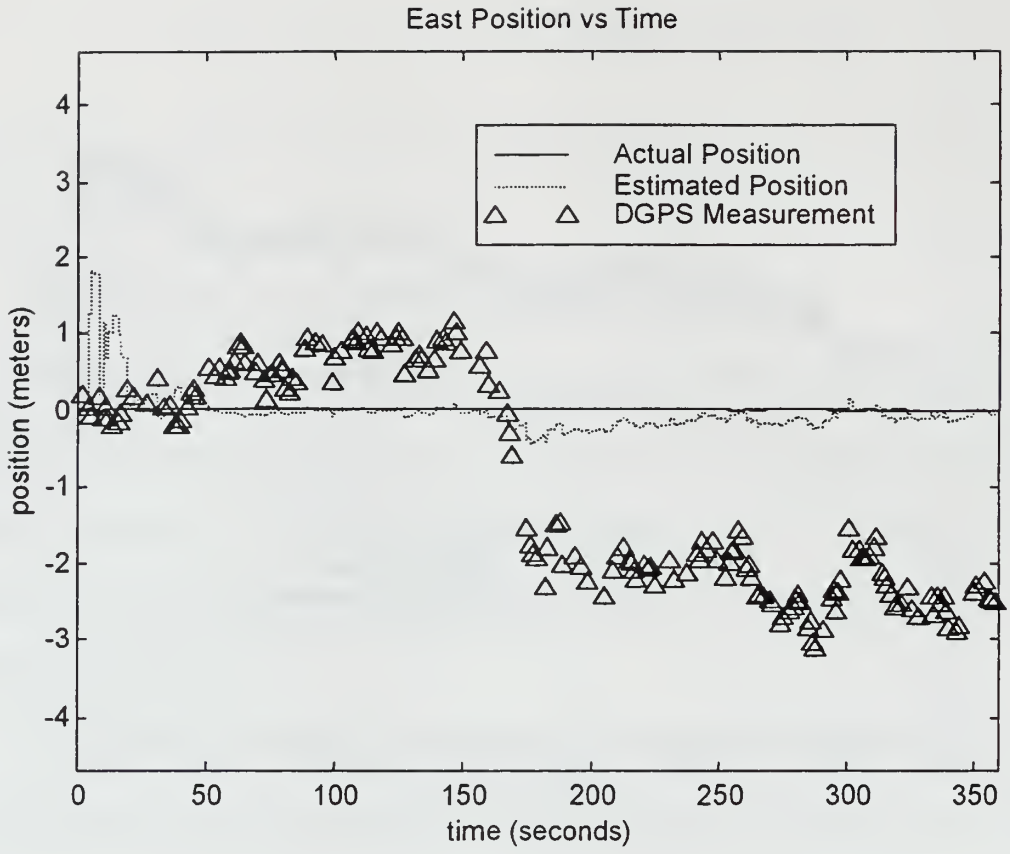


Figure 4.4: Plot of East Position vs. Time

Similar to Figure 4.3, Figure 4.4 displays estimated east position against time. This plot demonstrates GPS fluctuating about zero but the Kalman filter is able to maintain a good estimate of position given a zero velocity measurement. In addition to plotting position versus time, the error covariance matrix $\mathbf{P}_k$ is presented in Appendix B for examination of error covariance and correlation among each of the states in the Kalman filter.

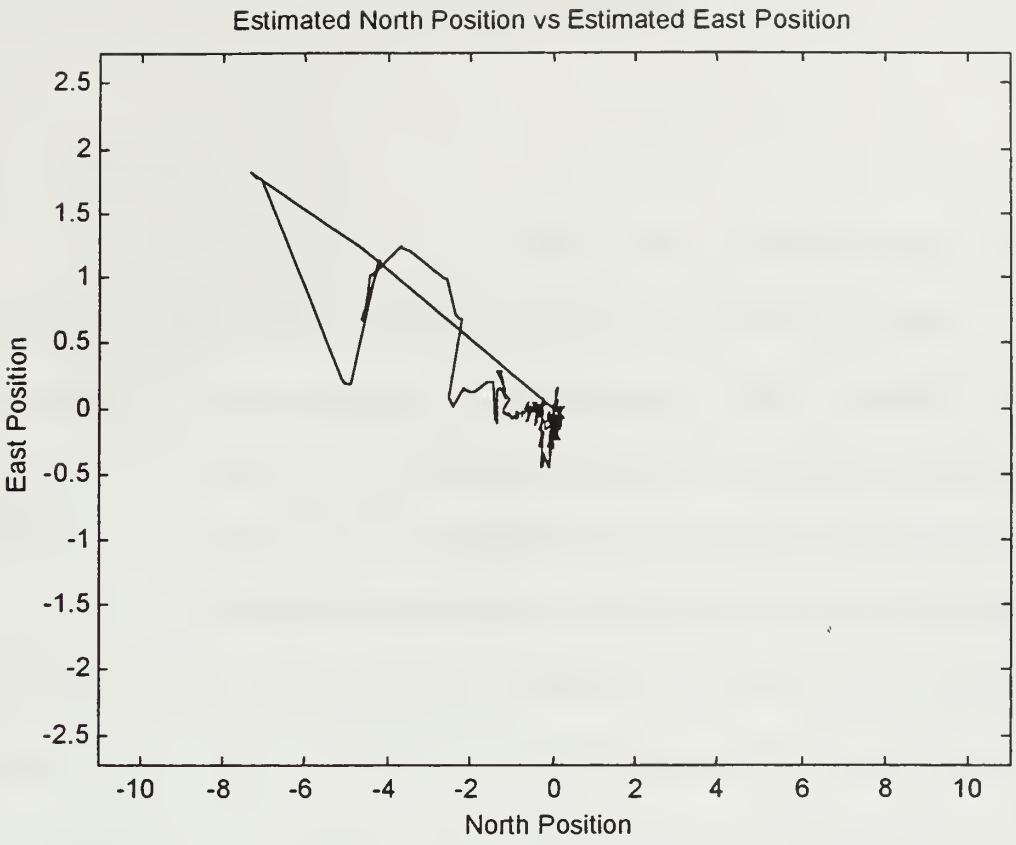


Figure 4.5: Plot of Estimated North Position vs Estimated East Position

Figure 4.5 represents data from Figures 4.3 and 4.4 and plots them together to give a more geographic plot of the simulation model. The long jumps in position were due to initial estimates within the first 30 seconds. As time went on, the position differential became minimal. It can be seen that an asynchronous Kalman filter is a good way to optimally estimate the states of SANS. This case only presents the position information, but all the other states can be easily represented. Much more can be done with this simulation, such as including all the states of the SANS filter and getting better models for measurement data (GPS and waterspeed).

V. CONCLUSIONS

A. SUMMARY

The purpose of this thesis was to develop a low cost method to navigate a small AUV by integrating an inertial navigation system and differential GPS. Major modifications have been made to SANS and the navigation filter now optimally estimates position. SANS was reduced by 52% in size and capable of running at an estimated 100 Hz sampling rate. The components in SANS were selected based on cost, size, and ease of operation. An asynchronous Kalman filter was developed and tested in simulation with promising results. What makes this navigation filter unique from any other AUV navigation system is that it accounts for current. The Kalman filter explained in this thesis estimates the AUV position, velocity, ocean current, and DGPS bias.

B. FUTURE RESEARCH

The future of SANS has many avenues predicated upon technology advances, software development, and the amount of research put into testing and evaluation. As micromachined chip technology evolves in the inertial navigation field, improvements in IMUs will most certainly come about. Reducing the size of SANS should continually be a goal while maintaining performance and low cost. Using the PCMCIA Card format for GPS receivers could potentially reduce the size of SANS by another 30 cubic inches. The limiting factor with PCMCIA GPS receivers is the added size of a differential signal receiver required for DGPS. No DGPS receivers have been implemented onto a single board.

Addition of the Proxim RangeLAN2 PC card [Ref. 15] gives SANS another dimension of usability. This card could give SANS more portability and potentially changes the way the sensors are integrated and utilized for applications other than AUVs. For example, data coming from the IMU, compass, DGPS, and water speed sensor could be transmitted to a remote computer for post processing rather than taking up CPU time running the navigation software.

Now that the asynchronous Kalman filter has been developed, more tests are needed to further test its reliability. A better DGPS model could be used for simulations that are more realistic. In addition, simulated velocities and currents could provide further assurance that the states are being estimated correctly. Another area needing work is the navigation software. The original SANS code is in C++ and operated on a Maxus ESP 50 MHz PC. A new SANS code needs to work on the PC/104 platform using the new IMU and waterspeed data format. The code could be in C++ or Java. Using Java gives SANS more flexibility within the network of computers imbedded within the Phoenix AUV.

Currently, research is being done on using quaternions vice Euler angles for attitude estimation. Use of quaternions would be necessary for SANS to track the motions of human limbs through the vertical. Accomplishing this task would also mean reducing the size of SANS to that of a wristwatch, which would be strapped onto various body segments. For more information see Reference [19]. Tilt table tests must be performed to examine the IMU data. Compass calibration has already been examined [Ref. 6]. When SANS reaches the stage when hardware and software are fully integrated ground tests and sea trials will be needed to prove its operation.

APPENDIX A: SANS ASYNCHRONOUS KALMAN FILTER MATLAB CODES

```
clear all

% SANS Kalman Filter
% By Glenn Hernandez
% 8 May 98

% Number of minutes for simulated run
minutes = 6;

% Define the resolution of the simulation here
delta_t = .01;  % For 100 Hz resolution

% Number of samples
samples = 1/delta_t * 60 * minutes;  % Gives 60000 samples at delta_t =
.01 seconds

% Time Constants
tau_1 = 60;   % seconds for velocity
tau_2 = 60;   % seconds for GPS
tau_3 = 3600; % seconds for ocean current

% Process Noise Vector
w1 = randn(1,samples);
w2 = randn(1,samples);
w3 = randn(1,samples);
w4 = randn(1,samples);
w5 = 600 * randn(1,samples);  % Gives a GPS standard deviation of 3 m
w6 = 600 * randn(1,samples);  % Gives a GPS standard deviation of 3 m
w7 = zeros(1,samples);        % No white noise input for x7
w8 = zeros(1,samples);        % No white noise input for x8
w = [w1;w2;w3;w4;w5;w6;w7;w8];

% Measurement Noise Vector
v1 = randn(1,samples);
v2 = randn(1,samples);
v3 = randn(1,samples);
v4 = randn(1,samples);
v_0 = [v1;v2];        % Noise vector without GPS input
v_1 = [v1;v2;v3;v4];  % Noise vector with GPS input

% Generate GPS Sampling
gps_flag = zeros(samples,1);

for g = 1:1/delta_t:samples  % Need at least .01 seconds between GPS
fixes

j = rand;  % Normalized random generator between 0 and 1
  if j < .5
    gps_flag(g) = 0; % No GPS Signal Available
```

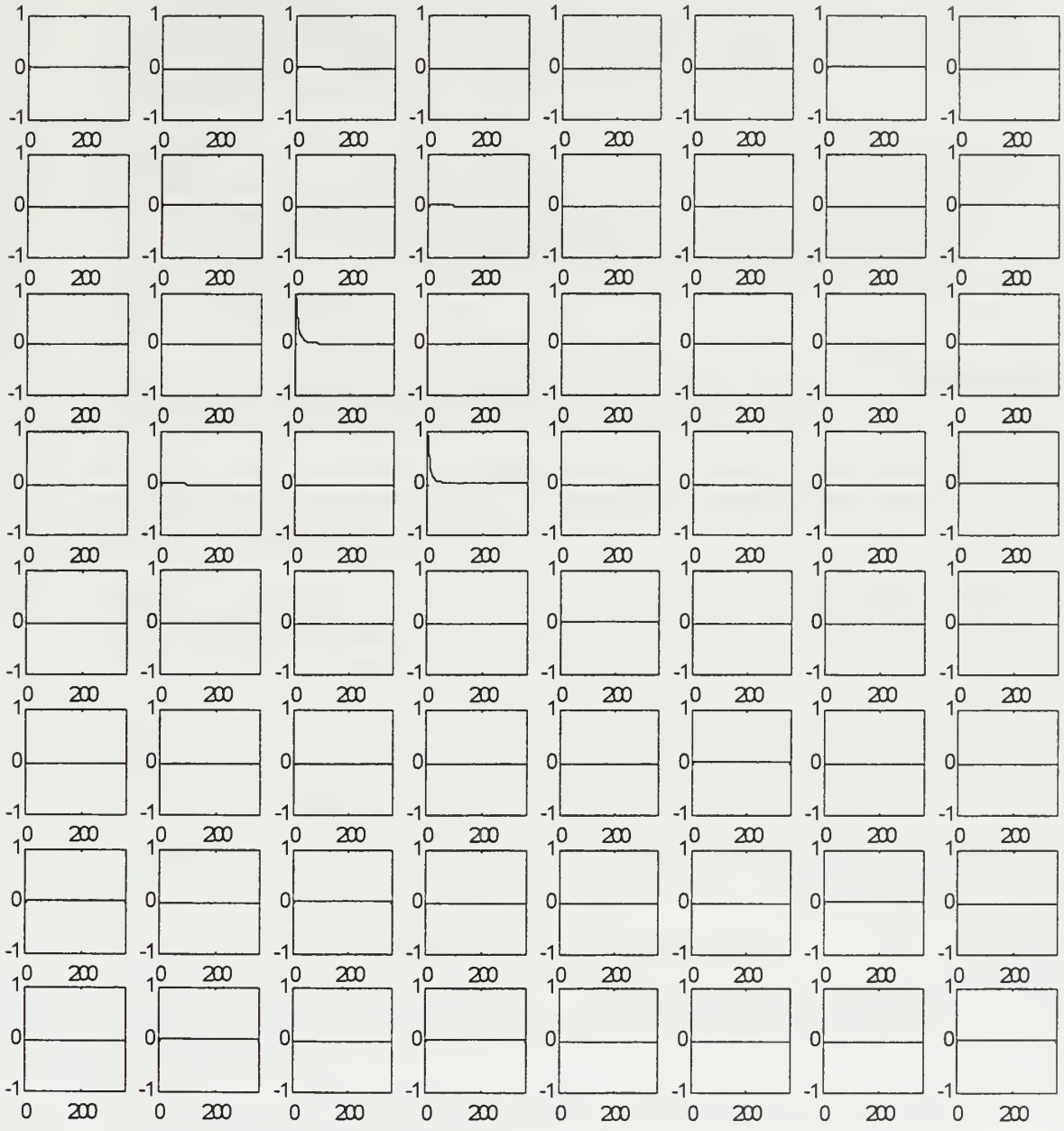
```
  else
    gps_flag(g) = 1; % GPS Signal Available
  end
end

'gps flags generated'

% System Matrix
A = [1/tau_1     0       0       0       0       0       0       0 ;
       0     1/tau_1     0       0       0       0       0       0 ;
       0       0     1/tau_2     0       0       0       0       0 ;
       0       0       0     1/tau_2     0       0       0       0 ;
       0       0       0       0     1/tau_3     0       0       0 ;
       0       0       0       0       0     1/tau_3     0       0 ;
     1/tau_1   0     1/tau_2     0       0       0       0       0 ;
       0     1/tau_1     0     1/tau_2     0       0       0       0];

% Input Noise Matrix
B = [w1;w2;w3;w4;w5;w6;w7;w8];

% Output Matrix
C = eye(8);

D = zeros(8,samples);

% State Transition Matrix
phi = [exp(-delta_t/tau_1)     0 0 0 0 0 0 0 ;
          0 exp(-delta_t/tau_1) 0 0 0 0 0 0 ;
          0 0 exp(-delta_t/tau_2) 0 0 0 0 0 ;
          0 0 0 exp(-delta_t/tau_2) 0 0 0 0 ;
          0 0 0 0 exp(-delta_t/tau_3) 0 0 0 ;
          0 0 0 0 0 exp(-delta_t/tau_3) 0 0 ;
          tau_1*(1-exp(-delta_t/tau_1)) 0
          tau_2*(1-exp(-delta_t/tau_2)) 0 0 0 0 0 ;
          0 tau_1*(1-exp(-delta_t/tau_1))
          0 tau_2*(1-exp(-delta_t/tau_2)) 0 0 0 0];

Q = [((1/(2*tau_1))*(1-exp((-2*delta_t)/tau_1)))     0 0 0 0 0 0 0 ;
        0 ((1/(2*tau_1))*(1-exp((-2*delta_t)/tau_1))) 0 0 0 0 0 0 ;
        0 0 ((1/(2*tau_2))*(1-exp((-2*delta_t)/tau_2))) 0 0 0 0 0 ;
        0 0 0 ((1/(2*tau_2))*(1-exp((-2*delta_t)/tau_2))) 0 0 0 0 ;
        0 0 0 0 ((1/(2*tau_3))*(1-exp((-2*delta_t)/tau_3))) 0 0 0 ;
        0 0 0 0 0 ((1/(2*tau_3))*(1-exp((-2*delta_t)/tau_3))) 0 0 ;
        0 0 0 0 0                          0                   0 0 ;
        0 0 0 0 0                          0                   0 0];


% Generate Process Noise Vectors
process_noise = sqrt(Q)*w;

% Error Covariance Matrix
R_0 = diag([.5 .5]);      % Without GPS signal
R_1 = diag([.5 .5 0 0]);  % With GPS signal
```

```
% Generate Measurement Noise Vectors
sensor_noise_0 = sqrt(R_0) * v_0;  % Without GPS signal
sensor_noise_1 = sqrt(R_1) * v_1;  % With GPS signal

% Initial x_hat_minus
x_hat_minus(8,samples) = zeros;

% Initial x_hat_plus
x_hat_plus(8,samples) = zeros;

% Initial State Vector
x(8,samples) = zeros;

% Initial Error Covariance Matrix
P_minus = [.5  0  0  0  0  0  0  0 ;
           0 .5  0  0  0  0  0  0 ;
           0  0  1  0  0  0  0  0 ;
           0  0  0  1  0  0  0  0 ;
           0  0  0  0  3  0  0  0 ;
           0  0  0  0  0  3  0  0 ;
           0  0  0  0  0  0  5  0 ;
           0  0  0  0  0  0  0  5];

time_index_a = 1;  % Initial Index for Measurement Vector without GPS
time_index_b = 1;  % Initial Index for Measurement Vector with GPS

'Beginning Kalman Loops'

% Begin Simulation
for k = 2:samples

% Generate State Vectors and Measurement Vectors
x(:,k) = phi * x(:,k-1) + process_noise(:,k-1);

  % Kalman loop with out GPS signal
  if gps_flag(k) == 0

        H = [1 0 0 0 0 0 0 0 ;
             0 1 0 0 0 0 0 0];

        R = diag([.5 .5]);

        sensor_noise_0(:,k) = sqrt(R) * v_0(:,k);

        z_vel1(time_index_a) = H(1,:) * x(:,k) + sensor_noise_0(1,k);
        z_vel2(time_index_a) = H(2,:) * x(:,k) + sensor_noise_0(2,k);

        z_vel_time(time_index_a) = k * delta_t;

        z_vel = [z_vel1 ; z_vel2];

        % Compute Kalman Gain
        K = P_minus * H' * inv(H * P_minus * H' + R);
```

```
        % Update Estimate
        x_hat_plus(:,k) = x_hat_minus(:,k-1) + K *
                          (z_vel(:,time_index_a) - H *
                          x_hat_minus(:,k-1));

        % Compute Error Covariance for Updated Estimate
        P_plus = ( eye(8) - K * H ) * P_minus;
        P_plus = ( P_plus + P_plus' ) / 2;

        time_index_a = time_index_a + 1;  % Increase the measurement
        vector index by 1

    else

        H = [1 0 0 0 0 0 0 0 ;
             0 1 0 0 0 0 0 0 ;
             0 0 0 0 1 0 1 0 ;
             0 0 0 0 0 1 0 1];

        R = diag([.5 .5 0 0]);

        z_gps1(time_index_b) = H(1,:) * x(:,k) + sensor_noise_1(1,k);
        z_gps2(time_index_b) = H(2,:) * x(:,k) + sensor_noise_1(2,k);
        z_gps3(time_index_b) = H(3,:) * x(:,k) + sensor_noise_1(3,k);
        z_gps4(time_index_b) = H(4,:) * x(:,k) + sensor_noise_1(4,k);

        z_gps = [z_gps1 ; z_gps2 ; z_gps3 ; z_gps4];

        z_gps_time(time_index_b) = k * delta_t;

        % Compute Kalman Gain
        K = P_minus * H' * inv(H * P_minus * H' + R);

        % Update Estimate
        x_hat_plus(:,k) = x_hat_minus(:,k-1) + K *
                          (z_gps(:,time_index_b) - H *
                          x_hat_minus(:,k-1));

        % Compute Error Covariance for Updated Estimate
        P_plus = ( eye(8) - K * H ) * P_minus;
        P_plus = ( P_plus + P_plus' ) / 2;

        time_index_b = time_index_b + 1;  % Increase the measurement
        vector index by 1

    end

pmin(:,:,k) = P_plus;  % Save Matrix for future analysis
```

```
  % Project Ahead
  x_hat_minus(:,k) = phi * x_hat_plus(:,k);
  P_minus = phi * P_plus * phi' + Q;
  P_minus = ( P_minus + P_minus' ) / 2;

  time(k) = k * delta_t;  %  Memorize time index

end


figure(1)
subplot(3,1,1)
plot(time, x(7,:), 'b-',time, x_hat_minus(7,:), 'r-.',z_gps_time,
z_gps3, 'g^')
xlabel('time (seconds)')
ylabel('position (meters)')
title('North Position vs Time')
axis([0 max(time) -1.5*max(abs(z_gps3)) 1.5*max(abs(z_gps3))])

subplot(3,1,2)
plot(time, x(8,:), 'b-',time, x_hat_minus(8,:), 'r-.',z_gps_time,
z_gps4, 'g^')
xlabel('time (seconds)')
ylabel('position (meters)')
title('East Position vs Time')
axis([0 max(time) -1.5*max(abs(z_gps4)) 1.5*max(abs(z_gps4))])

subplot(3,1,3)
plot(x_hat_minus(7,:),x_hat_minus(8,:),'b-')
xlabel('North Position')
ylabel('East Position')
title('North-East Position Plot')
axis([-1.5*max(abs(x_hat_minus(7,:))) 1.5*max(abs(x_hat_minus(7,:)))...
      -1.5*max(abs(x_hat_minus(8,:))) 1.5*max(abs(x_hat_minus(8,:)))])

orient tall
```

APPENDIX B: ERROR COVARIANCE MATRIX FOR SANS ASYNCHRONOUS KALMAN FILTER SIMULATION



8 x 8 Matrix for Error Covariance P in SANS Kalman Filter Simulation
Each box represents one element of the matrix vs time

LIST OF REFERENCES

1. Brown, R. and Hwang, P., *Introduction to Random Signals and Applied Kalman Filtering, Third Edition*, John Wiley and Sons, New York, 1997.

2. Bachmann, E. R. and Gay, D., "Design and Evaluation of an Integrated GPS/INS System for Shallow-water AUV Navigation (SANS)," Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

3. McGhee, R.B., Clynch, J. R.., Healey, S. H., Kwak, S.H., Brutzman, D. P., Yun, X.P., Horton, N. A., Whalen, R. H., Bachamnn, E. R., Gay, D. L. and Shubert, W. R., "An Experimental Study of an Integrated GPS/INS System for Shallow-Water AUV Navigation (SANS)," *Proceedings of the Ninth International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, New Hampshire, September 25-27, 1995.

4. McGhee, R.B., "SANS Process Model (After Attitude Estimation)," SANS Meeting Notes, Naval Postgraduate School, California, February 11, 1998.

5. Knapp, R., "Design and Calibration of a Water Speed Sensor for a Small AUV Navigation System (SANS)," Master's Thesis, Naval Postgraduate School, Monterey, California, December 1997.

6. Walker, R. G., "Design and Evaluation of an Integrated, Self-contained GPS/INS Shallow-water AUV Navigation System (SANS)," Master's Thesis, Naval Postgraduate School, Monterey, California, June 1996.

7. Roberts, R. L., "Experimental Evaluation, and Software Upgrade for Attitude Estimation by the Shallow-Water AUV Navigation System (SANS)," Master's Thesis, Naval Postgraduate School, Monterey, California, March 1997.

8. *CMV586DX133 cpuModule User's Manual*, Real Time Devices, Inc., September 1997.

9. *C4-104 User Manual*, Sealevel Systems Inc., January 1997.

10. *Crossbow Six Axis Dynamic Measurement Unit Specifications*, Crossbow Technology, January 1998.

11. The Internet, [http://www.sontek.com/products/hydra/apps/application.html].

12. *Oncore User's Guide*, Motorola Inc., August 1995.

13. *TCM2 Electronic Compass Module User's Manual*, Precision Navigation Inc., June 1995.

14. The Internet, [http://www.pc104-embedded-solns.com].

15. The Internet, [http://www.proxim.com/proxim/products/rnglan2/7400.htm].

16. The Internet, [http://www.tdc.co.uk/navcard.htm].

16. The Internet, [http://www.trimble.com/products/catalog/pd_gi001.htm].

17. The Internet, [http://www.mathworks.com].

18. The Internet, [http://www-npsnet.cs.nps.navy.mil/Inertial.html].

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code EC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Professor Xiaoping Yun, Code EC/Yx 2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Eric R. Bachmann, Code CS/Bc 2
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Professor Robert B. McGhee, Code CS/Mz 1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. Superintendent 1
   U.S. Coast Guard Academy
   New London, CT 06320-4195

8. Department of Electrical Engineering 1
   U.S. Coast Guard Academy
   New London, CT 06320-4195

9. LT Glenn C. Hernandez 2
   610 S. Oakland Street
   Arlington, VA 22204