Self-Alignment and Navigation Algorithms for DREO Navigation Laboratory Heading Reference Unit

by

Carole R.M. Bolduc, B.Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Engineering

Ottawa-Carleton Institute for
Mechanical and Aerospace Engineering

Department of
Mechanical and Aerospace Engineering
Carleton University
Ottawa, Ontario

April 10, 1995

ISBN 0-612-03014-8

Name Carole Rita Marie Beldue

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Aerospace Engineering — SUBJECT TERM

0538 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

The undersigned recommend to

The Faculty of Graduate Studies and Research

acceptance of the thesis

Self-Alignment and Navigation Algorithms

for DREO Navigation Laboratory

Heading Reference Unit

submitted by

Carole R M. Bolduc, B.Eng.

in partial fulfilment of the requirements for

the degree of Master of Engineering

Dr. D.A Staley

Thesis Supervisor

Dr. R. Bell

Chair, Department of Mechanical and Aerospace Engineering

Carleton University

April 1995

Abstract

An Attitude and Heading Reference System (AHRS), under development at the Defence Research Establishment Ottawa (DREO), consists of two, Canadian Strapdown Gyroscopes (CSG-2), and three QA1400, Q-Flex pendulous accelerometers. In this thesis, the self-alignment and strapdown navigation algorithms for the DREO AHRS are formulated.

The proposed alignment scheme is divided into two phases, namely, coarse and fine alignment. The coarse alignment stage is designed to function in a static environment and relies entirely on the inertial sensor outputs. Subsequently, an analytic fine leveling and gyrocompassing stage is used to refine the initial computation by damping any residual tilt and heading errors Vehicle motion is not restricted during fine alignment. A damped local-level wander-azimuth mechanization is employed since it enables true worldwide navigation

The validity of the algorithms and their software implementation is verified using simulated and real data.

Acknowledgements

My deepest thanks is extended to my supervisor Dr. Douglas A. Staley for his invaluable advice and supervision throughout the course of this research.

I would also like to express my thanks to Michael Vinnins and Lloyd Gallop for their valuable assistance and advice during my thesis research, and for making this project possible. Also, I would like to thank DREO for their financial support.

I am grateful to Dr. Peter R. Frise for his encouragement and valuable advice.

Many thanks also to the DREO Navigation Group for their contributions. In particular, I would like to acknowledge my friends Derek Hartl and Thomas Fejes for scholarly discussions and their enthusiasm throughout the course of this thesis.

The author also wishes to thank the Mechanical and Aerospace Engineering office staff for their assistance.

Finally, I would like to express my deepest appreciation to my family; this thesis would not have been completed without their endless encouragement. I am grateful to my Mom, Rollande and my Dad, Réjean for supporting my education. A special thanks to my brother, Stéphane for his refreshing humour and wit.

Thank You Jesus for giving me strength and guidance.

I dedicate this work to my parents and my brother.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A/D	Analog to Digital
AHRS	Attitude and Heading Reference System
ATP	Acceptance Test Procedure
AX,AY,AZ	X, Y and Z-axis accelerometers, respectively.
BTC	Bias Temperature Coefficient
C/A	Coarse Acquisition
CDU	Control and Display Unit
CEP, SEP	Circular, Spherical Error Probable
CFAV	Canadian Forces Auxiliary Vessel
CMT	Contraves Motion Table
CPU	Central Processing Unit
CSG-2	Canadian Strapdown Gyroscope (TDF)
D.E.	Differential Equation
DREO	Defence Research Establishment Ottawa
DCM	Direction Cosine Matrix
DIINS	Dual Inertial Integrated Navigation System
ESG	Electrostatic Gyroscope
FOG	Fiber Optic Gyroscope
GMT	Greenwich Mean Time
GPS	Global Positioning System
HRU	Heading Reference Unit
IA	Input Axis
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ISA	Instrument Sensor Assembly
LINS	Laser Inertial Navigation System
LPF	Low Pass Filter
OA	Output Axis
PA	Pendulous Axis
ppm	Parts per million
RHP	Right Half Plane
RK	Runge-Kutta
RLG	Ring Laser Gyroscope
RMS	Root Mean Square
SDF, TDF	Single degree of freedom, Two degrees of freedom
SS	Steady-state
TANS	Trimble Advanced Navigation Sensor
TINORU	The Inertial Navigation Organized Research Unit
WGS	World Geodetic System

Nomenclature

a	Heading error damping parameter (Chapter 5, 7)
a_i	Accelerometer outputs along subscript axis, for example, a_{Gy} is the acceleration along the Y Gyro axis.
a_x, a_y, a_z	Accelerometer outputs along X, Y, Z system axes
a_{xr}, a_{yr}, a_{yr}	Raw accelerometer outputs
$\bar{g}^i$	Gravity vector in 'i' frame of reference (superscript), (Chapter 5)
g_m	Mass gravitational accceleration
h	Altitude
q_i	Quaternion parameters
s	Laplace transform complex variable
t_{acc}, t_{gyr}	Student's t-distribution (§5.4.2)
t	time (w.r.t. differential equations)
AB_i	Accelerometer bias, for 'i' axis
A_i	Pure acceleration terms, along 'i' axis
A_{ij}	Accelerometer misalignment angles, the first subscript refers to the body axis and the second refers to the system axis
AK_i	Accelerometer scale factor, for 'i' axis
B	Body Align Matrix (Coarse Alignment)
C_i^j	Coordinate Transformation matrix, from coordinate frame delineated by subscript 'i' to coordinate frame in superscript 'j'
CI	Confidence Interval
D	Transformation Matrix relating the earth frame to the computational frame of reference
E, N, U	East, North, Up coordinates
G	Geographic Align Matrix (Coarse Alignment)
GB_i	Gyroscope bias drift, for 'i' axis
G_{ij}	Gyroscope misalignment angles, the first subscript refers to the body axis and the second refers to the system axis

GK_i	Gyroscope scale factor, for 'i' axis
M	Misalignment matrix relating true navigation frame of reference with the estimated computational frame of reference
MU_i	Gyroscope mass unbalance drift term, for 'i' axis
N	Number of samples
P	Vehicle position
Q	Quaternion matrix
Q_i	Gyroscope quadrature drift term, for 'i' axis
R	Adjustable gain parameter in damped fine alignment control scheme. It is less than Earth's radius.
R_o	Earth's radius
S	South
$\bar{V}$	Velocity
W	West
$\bar{X}$	Mean Value of X
X, Y, Z	Orthogonal set of coordinates
α	Wander angle
$\delta(R)_i$	Apparent gravity vector component
ϕ	Roll angle
ϕ_L	Latitude
λ	Longitude
μ_i	True angular velocity of the vehicle (Chapter 6)
v	Low pass filter output for input accelerometer data
θ	Pitch angle
θ_i	Small angle misalignments for 'i' axis, (Chapter 5,7)
ρ_i	Angular rate of the computational frame of reference relative to the earth (Chapter 6)
$\bar{\rho}^g_{c/e}$	Angular rate of the computational frame of reference relative to the earth in geographic coordinates, (Chapter 5)
σ_N	Standard deviation

τ	Low pass filter time constant
τ_s	Schuler period
υ	Angular velocity of rotating axes
$\vec{\omega}^{b}_{i/e}$	Angular rate of the earth relative to inertial space in body frame of reference
ω_s	Schuler frequency
$\omega_x, \omega_y, \omega_y$	Angular rates along X, Y, Z system axes
$\omega_{xr}, \omega_{yr}, \omega_{yr}$	Angular rate raw data outputs from gyroscopes
ψ	Yaw angle
ψ_T	True heading
ΔB_1	Bias instability
ΔB_2	Bias instability plus inexact earth rate compensation
$\vec{\vartheta}$	Instantaneous rotation detected by the vehicle
$\vec{\Omega}^{g}_{i/e}$	Angular rate of the earth relative to inertial space in geographic reference (superscript), (Chapter 5).
Ω	Full earth rate (Chapter 6)
Ω_i	Local earth rate about 'i' axis (Chapter 6)
$\vec{\Omega}_b$	Angular rate of the body (subscript) relative to inertial space
$\vec{\Omega}_g$	Angular rate of the geographic coordinates (subscript) relative to inertial space

<u>Subscripts:</u>

acc	accelerometer
b	body frame
c	computational frame (navigational frame)
e	earth-fixed frame
g	geographic frame
gd	ground

gyr	gyroscope
i	inertial frame
N	N^{th} sample
n	navigational frame
r	raw data
rot	rotating
ss	steady-state
stat	stationary
trans	transient

Chapter 1

Introduction

1.1 Motivation and Problem Statement

One of the research and development projects at the Defence Research Establishment Ottawa (DREO) Navigation Laboratory (NavLab) is an Attitude and Heading Reference System (AHRS) for sea and land-based applications. The AHRS comprises (Q-Flex) pendulous accelerometers and two-degree of freedom (Canadian Strapdown Gyroscopes (CSG-2)) as the inertial sensors used for detecting accelerations and angular velocities relative to inertial space. The motivation behind this thesis is to design and develop the self-alignment and strapdown navigation algorithms specifically for the DREO AHRS, alternately designated the Heading Reference Unit (HRU).

The problem to be solved is essentially one of dead-reckoning since an inertial navigation system must first be initialized before it can generate continuous attitude, heading and position information. The HRU is intended to operate as a fully autonomous navigation system. Unlike radio navigation systems in existence today, such

as the Global Positioning System (GPS), Loran C, and Doppler radar, Inertial Navigation Systems (INS) are inherently non-jammable. The ability of an INS to function in all-weather conditions, independent of external references and equipment, leaves this type of system unrivaled in the area of resilient, dependable navigation information. Furthermore, recent advances in micro-processor technology, ultra-precision Analog to Digital (A/D) converters and high accuracy inertial grade sensors have paved the way for the progression towards analytic strapdown systems as opposed to mechanical, gimballed systems. In a strapdown system the inertial sensors are fixed to the vehicle, without gimbals, to sense the vehicle's linear and angular motions. The consequence of having the inertial sensors fixed to the vehicle is that the navigation equations must be implemented using a fully analytical software oriented approach.

The proposed optimum alignment scheme is intended to align the system axes of the inertial package with a local-level frame of reference through the use of the inherent outputs available from the HRU sensors. This is accomplished with only minimum operator-supplied initialization data. This alignment algorithm does not take advantage of external sources of reference since doing so would compromise the self-contained nature of the HRU.

The other incentive for this thesis is to develop the strapdown navigation algorithms that derive the latitude, longitude, velocity, acceleration, roll, pitch, and heading information of the vehicle based on instantaneous outputs from the inertial sensors in a three-dimensional frame of reference. For instance, roll angles refer to rotations about a ship's keel or longitudinal axis, and pitch angles are rotations about the lateral axis. The

heading of a ship corresponds to the line-of-sight vector and it is positive when measured East from true North. For a local-level wander-azimuth frame of reference the heading of the ship or terrestrial vehicle is defined in terms of the azimuth (or yaw) angle and the wander angle. The azimuth angle relates the vehicle's orientation to the local-level wander-azimuth frame of reference and the wander angle is the angle by which the level axes deviate from a geographic north-pointing system. This concept is addressed in more detail in Chapter 6.

The focus of this thesis is the development of the coarse and fine alignment mechanizations and the navigation algorithms. Fine alignment is further subdivided into fine leveling and gyrocompassing modes of operation. This thesis also includes the software implementation of these algorithms.

1.2 Objectives and Overview of Results

The primary design objectives for the alignment procedure are: a scheme that executes rapidly and tolerates vehicle motion. There are two stages to the alignment process, namely, coarse and fine alignment. This thesis describes a method for obtaining the transformation matrix that relates the vehicle's body coordinates to the computational frame of reference, namely the local-level, wander-azimuth frame of reference.

An analysis of the linearized error equations describing misalignments between the ideal computational frame of reference and the estimate obtained during coarse alignment is then undertaken. The purpose of this analysis is to find the optimum filter parameters for the East, North and Azimuth loops of the fine leveling and

gyrocompassing equations, that ensure damping of the transient response of the misalignment angles. The effects of manoeuvre accelerations and bias instabilities on the misalignment angles are considered in a detailed tradeoff study.

Error analysis is an integral part of the process since sensor internal errors and installation errors affect the overall system performance. Procedures dedicated specifically to perform the compensation of the known, constant error terms are also needed.

Conclusions drawn from the fine alignment tradeoff study are then incorporated in the damped navigation equations. The purpose of the navigation algorithms is to efficiently and accurately calculate the vehicle's orientation and position on a continuous basis and display the updates at regular time intervals

The last objective of this thesis is to effectively prove that the concepts are valid by implementing the programs to execute the alignment and navigation algorithms. Results from simulated track profiles, different initial condition errors and real dynamic data collected from the HRU are presented.

1.3 Thesis Outline

This thesis is organized in sections that progressively cover background information, a survey of the relevant literature, theoretical formulations, the final implementation, results and finally conclusions. A description of the thesis outline follows.

Chapter 2 gives background information on the features of an Attitude and Heading Reference System. Differences between strapdown and gimballed systems are identified

and the underlying principles that govern the operation of various inertial sensors are explained.

Chapter 3 reviews the evolution of alignment and navigation techniques. This enables a parallel to be drawn between approaches already in use in existing systems and the HRU algorithms. The distinguishing features of each method are emphasized.

Chapter 4 describes the present state of the DREO HRU. The goals of this thesis in relation to the HRU are stated.

Chapter 5 develops the inertial sensor data compensation, coarse alignment, fine leveling and gyrocompassing equations. A detailed study of several steady-state and transient level error responses is undertaken to ensure damping of the misalignment error angles. A condition for heading error stability is also established.

Chapter 6 develops the local-level wander-azimuth navigation equations for a strapdown system. Topics such as the generation of the Euler angles, the velocity equations and position information are covered.

Chapter 7 describes the program design approach for the implementation of the Coarse Alignment and Navigation software. The results of a static multiposition test are presented to validate the Coarse Alignment software. A series of initial error test cases is used to verify the fine alignment procedure. A simulated test track and HRU data from the Dual Inertial Integrated Navigation System (DIINS) 1994 sea trial are used to test the navigation equations.

Chapter 8 lists the conclusions of this thesis. Suggestions for future research endeavours and a summary of contributions are given.

1.4 Variable Notation

The convention used to denote the various types of mathematical variables used in this thesis is summarized below:

- Column vectors are denoted with an arrow "$\rightarrow$"and the variables are written in normal typeface mode.
- Matrices are expressed in bold letters.
- Matrix subscripts refer to the system from which the transformation originates and the superscripts refer to the coordinate system to which the vector is being transformed.
- Derivatives are denoted with dot notation.
- Scalar quantities are written in italics
- Measured or inexact quantities are denoted with a tilde "~".
- Absolute values are written inside the "| |" symbols.
- Exact spatial rates are denoted with a circumflex " ^ " to distinguish them from the damped spatial rate parameters

Chapter 2

Inertial Navigation Systems

2.1 Introduction

Inertial navigation systems are used in a wide range of applications including marine, aeronautical, aerospace, terrestrial, surveying, and guidance mechanisms, to name a few. An inertial navigation system consists of inertial sensors, namely accelerometers and gyroscopes, that sense linear accelerations and angular rates, respectively and a navigation computer. The gyroscopes are used to maintain the chosen orientation of the computational reference frame with respect to the earth. The accelerometer outputs are then translated to the local-level (or computational) frame of reference and integrated to resolve velocity and doubly integrated to obtain position information. The gyroscopes provide attitude and heading information.

In this chapter, different types of gyroscopes and accelerometers are described. Furthermore, a brief comparison of gimballed and strapdown inertial systems is given to preface the choice of software mechanization. The physical hardware details and the

system's mechanical construction are addressed to clarify the subsequent alignment and navigation algorithms.

Important features associated with the outputs generated by an INS are its resolution and accuracy. Resolution refers to the preciseness of the results and it is a function of the number of bits used to represent the solution. In contrast, accuracy refers to the exactness or correctness of the solution compared to the true solution. Furthermore, a navigation system is characterized by its dynamic and environmental capabilities. The properties of the sensors used dictate the harshness of the operating environment the INS can withstand. Therefore, the overall system performance depends on the instruments, support electronics and the software implementation.

A summary of general performance capabilities of existing navigation systems is compiled in Table 2.1, on the following page. Table 2.1 lists only the published specifications available for the systems of interest.

Attitude and heading angles are specified in degrees root mean square (rms) The angular rates of the body, in deg/sec, are known to fall below the threshold given in Table 2.1. Position accuracy is specified in terms of nautical miles per hour (nmi/hr) (Circular Error Probable (CEP)), for pure inertial navigation systems, but in metres (Spherical Error Probable (SEP)), based on the GPS solution. CEP refers to the radius of a circle with centre at the mean point of a group of measurements that encloses 50% of the measurements. The SEP is essentially the previous concept extended to three dimensions. Applications that resolve position based on orbiting satellites, such as GPS, have performance specified in three dimensional spherical coordinates [3].

Navigation Systems	Pitch, Roll Accuracy (deg rms)	Heading Accuracy (deg rms)	Body Rates (deg/sec)	Position (m)	Velocity (m/s rms)
Honeywell H-423 (LINS) [1]	0.05	0.10	0.043	0.8 nmi/hr (CEP)	0.76
Honeywell H764-C3 (INS/GPS) [2]	0.01	0.02	---	16 m (SEP)	0.03
Typical unaided AHRS [3]	0.20	0.50	0.1	---	2.06
RLG AHRS [4]	0.15	0.25	0.02	---	---
HRU AHRS [5]	0.10	0.10	0.10	< 1 nmi/hr (CEP)	---

Table 2.1: Performance Characteristics of Navigation Systems. Comparison between pure inertial and integrated inertial and GPS navigation systems.

The Ring Laser Gyro (RLG) AHRS is a system designed for the specific purpose of enhancing the performance of a Doppler navigator. The typical unaided AHRS listed refers to a system which uses conventional spinning gyroscopes. This performance statistic is compiled based on systems such as the Lear Siegler Inc. Model 6000 Series, Sperry SRS-1000 and Litton LR-80 [4]. Anticipated performance levels for the DREO Heading Reference Unit are also provided in Table 2.1.

2.2 Attitude and Heading Reference System

Attitude and Heading Reference System is the terminology used to describe an INS that

specifically provides information on the orientation of the vehicle relative to the horizontal plane and its heading relative to a known reference. Other distinctive AHRS outputs are the angular rates experienced by the body. In an aircraft the altitude rates, angle of attack, normal acceleration and airspeed also need to be computed. In this thesis, the terms INS and AHRS are used interchangeably since the desired outputs from the system of interest are position, velocity, acceleration, flight dynamics, attitude and heading. However, an unaided AHRS provides low-accuracy outputs with unbounded errors, whereas modern inertial navigation systems typically combine AHRS data with GPS or other external references thereby limiting long-term error growth [3]. Another important feature of an AHRS is that it is conventionally a strapdown system that must support a high dynamic range. A requirement for the AHRS is that it not be susceptible to gimbal lock. Gimbal lock occurs, for instance, when two gimbal axes become aligned in a two-dimensional gyro thereby causing loss of one degree of freedom.

Besides the Instrument Sensor Assembly (ISA) and its associated electronics, the AHRS includes a navigation Central Processing Unit (CPU). The navigation processor encompasses the microprocessor that executes the navigation and alignment algorithms. It consists of an arithmetic logic unit, memory, and a means for supervising system inputs and outputs. Temperature controllers, converters to digitize the analog sensor outputs and clock synchronization are some additional essential components of a full-fledged AHRS.

The purpose of the navigation processor is to compute and display present attitude, position and velocity information to several subsystems of the vehicle. A transformation

matrix translates the accelerometer data from platform coordinates to inertial earth coordinates taking into account sensor biases, earth angular rates and the rate of the computational frame. The accelerations are integrated to obtain position information. It is imperative that these calculations be done in real-time. Data supplied at a 200 Hz rate effectively approximates a continuous data flow. To ensure that all data is homogeneous, the gyro and accelerometer outputs must be sampled simultaneously.

A navigation system can encompass external references inserted in the navigation processor as well as the Inertial Measurement Unit (IMU) measurements. External references such as from a radar altimeter or Global Positioning System are sometimes used to enhance the performance of an INS. However, the IMU is sufficient for providing all the necessary navigation information. The DREO Heading Reference Unit for which the alignment and navigation algorithms are designed is intended to operate as an entirely self-contained unit independent of any external references.

2.3 INS Mechanization

There exist three possible approaches for obtaining accelerometer data with respect to fixed local-level coordinates. The gimbalied system approach uses the gyroscopic information to detect the vehicle's rotation and keep the platform on which the accelerometers are mounted, fixed in the earth's inertial axes. In contrast, in a strapdown system the accelerometers and gyroscopes are fixed to the vehicle and the gyro rate information is supplied to a navigation processor that uses it to computationally resolve the accelerometer data and determine the vehicle's accelerations

and velocities in the three local-level coordinates. A strapdown system effectively consists of a mathematical platform as opposed to a physical platform. It does not physically instrument inertial nor navigation frames of reference. For this reason, a strapdown system is also referred to as an analytic or gimballess INS.

A strapdown system has the advantage of reduced mechanical complexity since it does not require mechanical gimballing. However, this is at the expense of a high-speed digital computer required to do the complex navigation computations. With the advent of faster microprocessors the technology now exists for developing accurate and improved strapdown navigation systems. The sensors in a strapdown system have higher performance requirements than those in a gimballed system since they are subjected to more severe motions. Strapdown instruments experience the rotation rates of the vehicle, whereas in a gimballed system smaller, shorter duration platform angular rates are experienced. Also, the gyro torquing requirements are different in a strapdown mechanization because larger output angle rotational displacements must be supported or larger torquers are needed for rebalancing [6]. Technology has progressed in the area of inertial sensor development to the point that strapdown systems are a feasible alternative. Also, with the emergence of high-speed computers to do the needed real-time computations with high accuracy gyroscopes and digitizers means that the high performance needed for a strapdown system can now be achieved. Furthermore, a strapdown system is more easily maintained because it has fewer rotating parts. Thus, it is often the favoured approach because it can be more reliable and less costly.

A third configuration is referred to as the space-stabilized mechanization. This is a

semi-analytic system characterized by an instrumented, earth-centered non-rotating inertial frame of reference and position computations being done in the computer. Unlike geometric systems that mechanize both an inertial non-rotating frame and a navigation frame and require five gimbals, the space-stabilized system physically instruments the former frame only.

This thesis deals with a strapdown system but it was deemed necessary to explain gimballed systems to adequately illustrate the properties of a strapdown system. Furthermore, a good understanding of both types of systems is useful when describing such concepts as gyrocompassing, and platform torquing; in the case of gimballed systems, the torque is applied mechanically and in strapdown systems it is applied mathematically.

2.4 Inertial Measurement Unit

The IMU consists of an assembly of inertial sensors, namely gyroscopes and accelerometers. The instruments are oriented such that full three-dimensional acceleration and angular rate data is available. The AHRS processes data received from the IMU. A good understanding of the principles governing the operation of these sensors is needed before the equations of motion and the corresponding data compensation algorithms can be developed.

2.4.1 The Gyroscope

The term 'gyroscope' was coined by French scientist Leon Foucault (1851), after the

Greek words 'gyros' meaning revolution and 'skopein' meaning to view [7]. There are several different kinds of gyroscopes. These include: dry-tuned mechanical gyroscopes, and optical gyroscopes, namely ring laser and fiber optic (FOG). RLG's detect angular rates by measuring the frequency difference between two contra-rotating laser beams. Three or four mirrors, arranged in a triangular or rectangular configuration, are used to reflect each beam around an enclosed area. When the gyro experiences an angular rate about an axis perpendicular to the path of the two beams, one beam has an apparently greater optical path length and the other beam a shorter path length. This difference in optical path length translates into a difference in resonant frequency. FOGs use a similar principle whereby a coil of optical fiber serves as the sensing path. The most significant advantage of FOGs is that they do not exhibit the lock-in effects for low-rotation rates observed in RLGs [7].

Optical gyroscopes operate well over a high dynamic range. Specifically, the GG1320 RLG can detect angular rates of up to 400 deg/sec and angular accelerations of up to 600 deg/sec^2 [2]. They also provide inherent digital output which is ideal for the navigation processor. Another advantage of optical gyroscopes is that they can be left unused without deteriorating; this is called dormancy. More detailed information on optical gyroscopes can be found in [7].

Another type of gyroscope is the Electrostatic Gyro (ESG). It consists of a spherical rotor suspended in a vacuum by an electrostatic field formed by case-fixed electrodes [8]. It is being considered for future use in strapdown applications.

A mechanical gyro consists of a spinning wheel or rotor. The assembly is supported

in bearings such that the rotor axis is free to rotate in one or two planes perpendicular to the plane of spin. If the rotor is supported in a single gimbal ring, the base can be turned relative to the rotor plane, and the rotor has two degrees of freedom. The rotor has freedom to rotate in three dimensions with the addition of an outer gimbal ring at right angles to both the inner gimbal ring and the rotor.

The two fundamental principles upon which gyroscopic theory is based are: gyroscopic inertia and precession. Gyroscopic inertia or rigidity refers to the tendency for a rotating mass to maintain the same plane of rotation unless acted on by an external force. This is, in effect, Newton's first law of motion which states that a body will stay at rest or remain in a state of uniform straight line motion unless compelled to change states by an external force. The particles of a spinning wheel are subjected to radial forces which keep the wheel axle pointing in the same direction unless another external force is applied. Thus, angular momentum is conserved if the gyro is subjected to internal mutual forces and no external forces.

The law of precession is another consequence of the conservation of angular momentum. If a couple is applied to the rotor about an axis perpendicular to the spinning axis, the wheel will precess about a third axis that is perpendicular to the planes of spin and of the applied couple. The direction of the precession is such as to bring the plane and direction of wheel spin axis towards the direction of the applied torque. The precession rate of the rotor is inversely proportional to the wheel speed and the moment of inertia of the rotor and directly proportional to the applied torque [9].

A more detailed description of two degree of freedom gyroscopes, ensues since

these are the type used in the HRU. In a two-degree-of-freedom gyro (TDF) the spin axis can rotate about two orthogonal axes. Gimbals, flotation, electrostatic-suspension and magnetic-suspension are examples of supports that can allow more than a single-degree-of-freedom (SDF). The orientation of the gyro spin axis relative to gimbals is a measure of the sensed angular velocity.

Each degree of freedom has an associated pickoff and torquer. The purpose of the pickoff output is to provide a measure of the angular displacement between the case and spin axes. An angular velocity about either input axis results in a rotation of the gyro spin axis that is an indication of vehicle motion or earth rate. This output signal is supplied to an analog rebalance loop which generates a direct rebalance current. The current then passes through the torquer coils and maintains the gyro at null. The pickoff signals serve a dual purpose. They are also fed to an analog to digital converter which digitizes the signal for use in the navigation computer. A small disturbance causes the dynamically tuned gyro to nutate. Nutation is the coning motion, or transient response that a two-dimensional gyroscope exhibits

The HRU ISA consists of two, TDF Canadian Strapdown Gyroscopes (CSG-2). The CSG-2 is an elastically-supported, dynamically tuned gyro that consists of a spinning rotor mounted on a shaft driven by the gyro motor. Inner torsional flexures give the inner gimbal some freedom to rotate about one axis. The torsion elements for the rotor allow it to have some angular freedom about an axis perpendicular to the shaft and the gimbal flexures. The rotor assembly rotates at a constant speed. If the placement of the rotor relative to the shaft deviates from orthogonality, the gimbal pulsates at the tuned

speed to balance torques generated by the flexures. The key feature of dynamically tuned gyroscopes is that they use dynamic inertia effects to cancel out the torsional flexure stiffness. The tuning equations for a dynamically tuned gyro are derived by Lawrence in Chapter 9 of [7].

The innovative use of electro-discharge machining techniques in the CSG-2 gyroscopes flexure mechanisms enhance their performance characteristics making these acceptable for use in a strapdown system. A description of the various inertial sensor error terms is compiled in §2.5. The principal performance criteria for these gyros include:

- random drift stability < 0.005 deg/hr
- non-g sensitive drift < 3.0 deg/hr
- non-g drift repeatability < 0.01 deg/hr
- g-sensitive drift < 10.0 deg/hr/g
- g-sensitive drift repeatability < 0.02 deg/hr/g

The above specifications are based on extensive testing and evaluation of the sensors undertaken at the DREO NavLab. Results from these tests are compiled in several reports, [10], [11], [12]. The CSG-2 general specification for parts and materials conforms to the MIL-E-5400 military standard. Detailed information on electrical and environmental requirements can be found in reference [13]. Figure 2.1 is a drawing of a tuned rotor gyro.

Unlike optical gyroscopes, mechanical gyros will not operate instantaneously

because they require a warm-up time during which the temperature must stabilize to within operating temperature limits (68 ± 0.2 deg Celsius for the CSG-2). The DREO HRU has dedicated temperature sensors and controllers to regulate any temperature variations.

2.4.2 The Accelerometer

The other type of sensor in the HRU ISA is the linear accelerometer. Accelerometers are devices used to measure accelerations. There are many accelerometer designs which accomplish this purpose. In its simplest form, a single-axis accelerometer consists of a suspended mass in a case which is moved away from null as the vehicle accelerates along a single axis. The mass is restrained by springs which obey Hooke's law. When a force incites the case to move, inertia properties of matter tend to keep the mass stationary. Consequently, the springs supply a force to move the mass, and it lags behind the case. The measure of acceleration is related to the displacement of the proof mass relative to the case. Therefore, the accelerometers measure the force exerted by the springs and this force is proportional to acceleration.

Alternatively, an accelerometer consisting of a pendulum has angular displacements proportional to acceleration. Accelerometers where a displacement is related to the acceleration fall in the category of open loop sensors. Electrical pickoffs are used to detect the minute changes in the position of the mass [7].

If the case and the suspended mass 'free fall" together under the isolated influence of gravitational acceleration along the unconstrained axis of the instrument, the

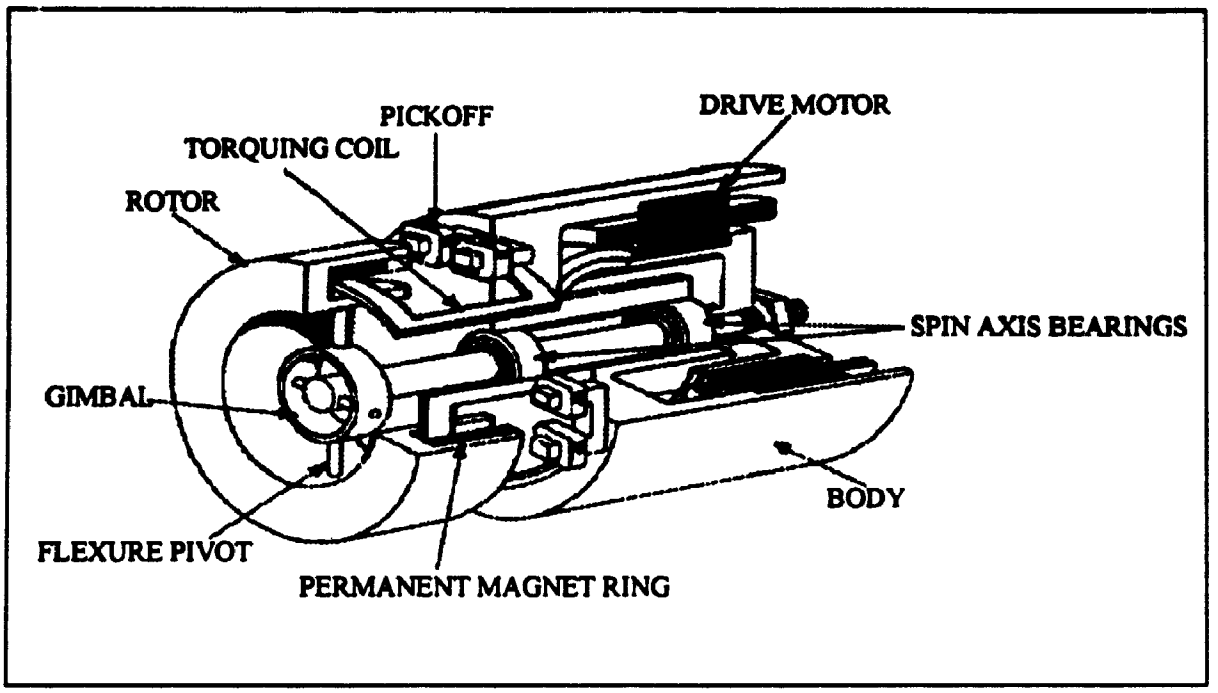


Figure 2.1: Tuned Rotor Gyroscope [10]

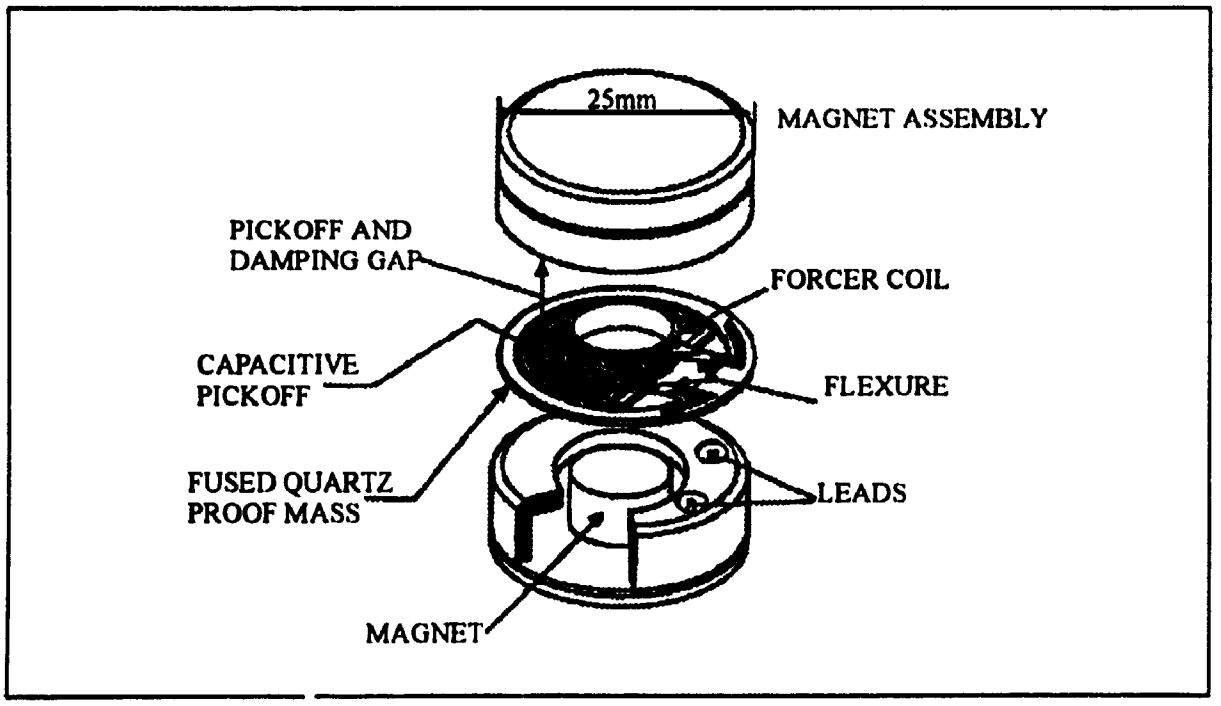


Figure 2.2: Q-Flex Accelerometer [7]

accelerometer output is zero. If the case is held stationary, the accelerometer output is a fraction of g depending on the orientation of the instrument, and this constitutes the acceleration along the accelerometer's sensitive axis.

. However, open 'oop sensors such as the spring-mass system described previously are subject to non-linearity errors. For this reason, closed loop accelerometers which rely on force-rebalancing principles to maintain the position of the mass constant are more often used in practice. Closed-loop accelerometers exhibit better performance characteristics in general. For example, closed-loop electrostatic sensors rely on an electronic servo loop to keep the mass equidistant from all surfaces, and the required applied voltage is an indication of acceleration. Similarly, the current required to return the mass to its null position in an electromagnetic sensor is also proportional to the input acceleration.

So far, the accelerometers described are of the non-integrating type, meaning their output signal is proportional to acceleration. There exist integrating accelerometers whose raw output consists of velocity increments. These integrating accelerometers provide variable-frequency outputs. They are sometimes referred to as velocity meters. A more detailed description of the different types of accelerometers can be found in Chapter 3 of Leondes [14].

Three SDF pendulous-type accelerometers are mounted in the HRU such that their input axes are mutually orthogonal and aligned with the vehicle system axes. The operation of a Q-Flex closed-loop pendulous accelerometer is based on a pendulum which, when displaced from null in response to an acceleration, alters the capacitances

on both sides of the pendulum. The one-piece hinge and pendulum structure is fabricated from fused quartz, a highly stable, non-conducting material. The subsequent process parallels the gyro operation described previously in that the accelerometer output signal is supplied to forcer coils that are part of a servo loop to return the pendulum to null. Specifically, the output is demodulated and amplified, and transmitted to electromagnets or torquers. The current output is then dropped across a resistance to yield an analog voltage proportional to the sensed acceleration.

The principal mutually orthogonal axes in a pendulous accelerometer are the input axis (IA), hinge output axis (OA), and pendulous axis (PA). Accelerations are sensed along the IA, whereas restoring torques are applied about the OA and the pendulum lies on the PA. The restraining torque is proportional to acceleration and pendulosity. The latter is a function of the proof mass and pendulum length [7]. A drawing of a typical Q-Flex accelerometer is shown in Figure 2.2.

The QA1400 accelerometers employed in the HRU are high long-term stability, minimum complexity, inertial grade accelerometers. One notable characteristic of the Q-Flex type is its inherent modelability. Polynomial models of the accelerometer's characteristics, specifically bias and alignments, over a wide temperature can be developed and ultimately included in the navigation software. Temperature modelling is viable since thermal gradients do not have a significant impact on the dry gas damped design [15]. Furthermore, the use of thermistors is no longer imperative. The bias temperature coefficient term is a measure of the thermal drift uncertainty. It is a low-cost accelerometer that exhibits exceptional turn-on to turn-on repeatability.

Typical performance characteristics for this sensor are:

- Scale Factor, 1.32 mA/g ± 10%
- Bias, ± 10 mg
- Bias Temperature Coefficient (BTC), < 10 μg/ °C
- Input axis misalignment , ± 2 mrad

The navigation processor converts the HRU Q-Flex accelerometer outputs from g-readings to acceleration in m/s^2 and performs the integrations to obtain velocity and position. Thus, the precision of the accelerometers affects the overall system performance.

2.4.3 Support Electronics

The inertial sensor outputs cannot be used directly in the navigation computer. Digital outputs are needed to interface with the microprocessor. Recent advances in analog to digital converter technology have improved the speed and resolution of conversion of the analog outputs from the gyros and accelerometers to digital form. A 22 bit digitizer is currently under development by The Inertial Navigation Organized Research Unit (TINORU) of Carleton University to give high resolution conversion of the gyroscope readings. The required minimum number of bits for digitizing the gyroscope information depends on the dynamic range of angular rates sensed by the gyroscopes and the minimum detectable change in latitude derived from local earth rate components. It remains to be determined whether a 16-bit converter is adequate for conversion of the

accelerometer outputs, or if additional TINORU converters will be needed [16].

Data rates of 400 Hz are anticipated when the system is in its final form. Real-time execution of the navigation algorithms relies on these minimum input data rates. A supplementary description of the digitizer technology and associated software can be found in reference [17]. A description of the current state of the Heading Reference Unit is given in Chapter 4.

2.5 Inertial Sensor Error Terms

System performance and accuracy depend on the digitization technology, block alignment, alignment and navigation algorithm implementation, and temperature compensation, to name a few key factors. Component imperfections and the mechanical assembly of the instruments also contribute to the degradation of the inertial navigation systems' performance and accuracy. Some of these errors can be measured and consequently corrected in the strapdown navigation software.

Errors associated with the CSG-2 mechanical gyros include. non g-sensitive drift rate, g-sensitive drift rate, random drift, axes misalignments, temperature sensitive drift rate, g^2-sensitive drift rate, scale factor non-linearity, frequency-sensitive anisoinertia terms and angular accelerations sensitivities. The data compensation algorithm addresses first-order, deterministic, repeatable errors only. Random drift, for instance, cannot be directly compensated because it is a measure of the standard deviation of the output. However, error growth in response to bias instabilities, for example, can be controlled via the algorithms.

Insufficient temperature data prevented thermal characteristics from being incorporated in the algorithms. Scale factor nonlinearity is one case where a known error term varies with temperature. Finally, anisoinertia and angular acceleration sensitivities are negligible quantities in comparison to the other error terms.

Accelerometers are subject to scale factor nonlinearity, IA misalignments, bias and anisoinertia effects. Anisoinertia and scale-factor non-linearities are neglected in the accelerometer error compensation for the same reason they are neglected in the gyroscope error compensation. Misalignments and bias are the two accelerometer errors that are corrected.

Rectification errors can result when accelerometer outputs are multiplied by the gyro-derived transformation matrix These errors are associated with the numerical methods used to implement the navigation and alignment algorithms. Orthogonalization conditions are enforced specifically for the purpose of diminishing these types of errors.

Chapter 3

Survey of Alignment and Navigation Methods

3.1 Alignment Techniques

The technique employed to align an inertial navigation system depends on many factors such as: constraints associated with the system mechanization, available data from the sensors, and the overall desired system performance goals. Although the mechanization of early gyrocompasses is very different from a strapdown inertial navigation system, the principles of the former method can still be applied to the latter. For this reason, early gyrocompassing is explained in the next section. Some modern analytic alignment methods are discussed in §3.1.2.

3.1.1 Early Gyrocompassing

The gyroscope is an inertial sensor affected by the earth's angular velocity. Specifically, the earth's rotation about the North Polar axis at the sidereal rate of one revolution in 23 hours and 56 minutes is significant. Secondly, the spherical, or to be more precise

ellipsoidal, shape of the earth implies that all terrestrial distances travelled map out a curve as opposed to a straight line. In a moving system, the gyroscope outputs include a component that is proportional to the vehicle velocity and inversely proportional to the Earth's radius. Another crucial fact employed in the original gyrocompass design is the knowledge that a gravitational force, acting normal to the earth's surface, causes a body in free-fall to accelerate at 9.806 m/sec^2.

The usefulness of early gyroscopic ship's compasses arises from the fact that they can sense the earth's rotation and distinguish it from any rolling and pitching motions of the ship. Furthermore, if the gyrocompass is disturbed, it gradually corrects itself and points again in the true North direction after a few hours.

The operation of a gyrocompass can be explained by considering the behaviour of a three degree of freedom gyroscope carrying a pendulous mass. Consider a gyroscope with spin axis initialized to point north with a small error deviation. As the earth rotates, the initial error cycles from a heading error to a tilt error and then the negative counterparts of these. For example, assume an initial positive counter-clockwise error about the Up(Z) axis such that the apparent North(Y) axis is tilted towards the West(-X) axis. After a 90 degree rotation of the earth, the North axis is now tilted slightly downwards. Considering quarter earth period increments, easterly and upward deviations follow. The pendulum does not have an effect when the error is entirely represented by a heading misalignment and the system is level. However, tilt errors cause the gravity pendulum to generate a torque.

Torque is created by a moment arm and force. In this case, the force is the

gravitational component sensed along the apparent North axis and the moment arm extends to the pendulum extremity and originates at the gyrocompass centre. But applied torque is equal to the rate of change of the angular momentum vector This torque tends to drive the angular momentum vector West and East for upward and downward errors, respectively. Thus, the pendulum produces a correcting torque that causes the angular momentum vector to trace out an ellipse. Without the addition of the pendulum, the point of the angular momentum vector maps out a circular path. For a more detailed explanation refer to Den Hartog's text [18].

For motion over the Earth, an effective pendulum length equal to the radius of the Earth is needed. Thus, the end point of the angular momentum vector maps out a full ellipse in 84.4 minutes, the Schuler period.

Therefore, in practice, these early gyrocompasses are coarsely initialized such that the rotor axis points towards the Polaris. The pendulum is used to sense any small initial errors. However, for accurate gyrocompassing these errors need to be damped. Introducing damping into this mechanism causes the angular momentum vector path to spiral inwards and the error angles to decay.

The physical gyrocompass consists of a gyroscopic disk mounted in three gimbal rings, a virtual pendulum, and a damping mechanism. Examples of actual constructions are the Sperry, Mark and Brown gyrocompasses. The Sperry marine compass achieves damping by precessing about the vertical axis using simple mercury ballistic instead of a pendulum. In contrast, the Brown gyrocompass uses oil forced by air pressure up to control 'bottles' and some counter-acting 'bottles' for damping. In both cases, the goal

is to have an upward error deviation cause a precession acting downward. Richardson's text is a good reference for detailed explanations of these gyrocompasses [9].

An analogy to a physical gyrocompass instrument can be drawn for the alignment of an inertial navigation system. In this case, the accelerometers perform the function of the pendulum and sense the initial errors. In a gimballed system, the servomechanism effectively drives the sensed rotations to zero thereby aligning the reference coordinate frame to North in the same way that a gyrocompass moves to point North. The sensed rotations are zero for the East Axis, so the system is aligned in azimuth by moving the gimbal such that the East gyro output is nulled. The platform is then slewed 180 degrees and the process is repeated.

In this thesis the principle of gyrocompassing is demonstrated as it applies to a strapdown INS. This process is referred to as analytical gyrocompassing since the East gyro is not physically driven to point East as in a gimballed system. The accelerometers are not used to apply a correcting torque, but instead to compute the required rate correction. This precession rate is applied to correct the orientation of the local-level wander-azimuth frame of reference. It is also important to note that although the reference frame is not a geographic north-pointing frame of reference, accurate heading alignment correction is still possible for a precisely defined wander angle rate.

The analytic gyrocompass is comprised of gyroscopes and accelerometers. Damping is injected by filtering the accelerometer outputs and using the filtered outputs to precess the local-level frame of reference.

3.1.2 Analytic Alignment Methods

Analytic alignment is the broad term used to refer to the process of leveling and gyrocompassing a strapdown system. A strapdown system cannot be physically precessed into a north-pointing orientation. Moreover, the platform is not physically erected into verticality. However, the role of the computational frame of reference in a strapdown system is equivalent to the physical inner gimbal in gimbaled system.

Priel describes a two-step gyrocompassing procedure that uses the inertial sensors available in an INS in his paper [19]. By successively placing the horizontal gyro axes along the North direction, through a 90 degree rotation about the azimuth axis, the horizontal gyro drift errors are measured Torquing rates can then be adjusted to correctly eliminate Earth rate components. Since only the North gyro drift component can be determined accurately, the East gyro drift component is found by rotating the platform effectively causing the East gyro to become the North gyro [20]. Slewing the platform is not possible in a strapdown system but the gyrocompassing errors can be modeled in a similar manner and reduction of the initial errors can be achieved through proper control of the analytic rates used to precess the mathematical platform.

Error modeling of inertial navigation systems is a research area which has received much attention in the form of many published papers. Error modeling of inertial navigation system outputs such as position, velocity, attitude and heading is an effective means of tracking failures. The modeling equations should be expressed in terms of the physical variables that best represent the error terms of interest. For example, the "psi-error angle model" [21] can be used to represent attitude, velocity and position errors

using a relatively simple dynamics matrix. In this thesis, an error model suited specifically for the gyrocompassing and leveling methods is developed. The primary goal of this error model is to improve the system accuracy by adjusting parameters that control how the initial attitude and heading errors decay with time.

As an alternative to the gyrocompassing technique, an iterative process can be used to align strapdown systems. The measurements in the navigation software are rotated to computational coordinates by a direction cosine matrix (DCM). This is repeated until the DCM has been adjusted to perform the transformation exactly [22]. The main drawback of this method is that quantization errors may hinder the convergence process.

The same approach can be used to re-calculate the sensor biases utilized in the data compensation algorithm at the beginning of the alignment mode. However, in the paper [23], the author states that it is preferable to determine the sensor biases as accurately as possible using an independent, precise calibration technique. One such calibration technique consists of placing the inertial package in a van and storing data for post-processing as it proceeds with a series of pre-defined manoeuvres. Alternatively, multiposition testing in a controlled laboratory environment affords more flexibility by placing the sensors in various orientations, which is not always possible at the start of every navigation mission.

The previous discussion focuses on gyrocompassing techniques used to initialize the heading of an INS. In conjunction with the gyrocompassing operation the INS platform must be leveled to remove any initial tilt errors Proportional and proportional plus integral "servo" controllers can be used to level the mathematical platform [24].

The accuracy of the analytic gyrocompassing technique governs the position accuracy that can be expected from the system. The Pinson error model is one means of assessing the efficacy of the alignment technique in terms of small angle misalignments. The Pinson method consists of formulating error equations from a system error block diagram of the alignment mechanization [25].

3.2 Navigation Considerations

The principal considerations when deriving an appropriate navigation mechanization are: choosing the best frame of reference for solving the navigation equations and determining the most efficient way of mechanizing the attitude algorithms.

The different coordinate frames that can be used to mechanize the navigation solution are related in terms of one or more transformations [26]. The mechanizations generally differ in terms of the corrective precession rates applied to the gyroscope outputs. North-slaved, unipolar, free azimuth and wander-azimuth systems are just a few examples of the diverse number of ways to implement the local-level navigation solution. The vertical precession rates are applied to compensate for any combination of the earth rate and vehicle transport rate components.

The navigation equations can also be solved in non local-level coordinate systems such as the inertial reference frame [3]. The most convenient reference frame in which to mechanize the navigation algorithms depends on the application for which it is intended. For instance, the inertial frame of reference is most suitable for space navigation, [27]. In this situation, the direction cosine matrix relates body coordinates to

the inertial frame of reference with origin at the sun.

The most efficient way to mechanize the attitude determination portion of the strapdown navigation algorithms is an important issue to consider. Grubin investigates the various tradeoffs between Euler angle, quaternion, and direction cosine methodologies in his paper [28]. Conclusions drawn from the extensive error analysis associated with this study are that the DCM scheme is the least accurate and the Euler method accuracy is superior to the quaternion scheme for larger coning angles used to test the algorithms. However, the Euler scheme places the highest demand on computational time. The results of this research are taken into account when deriving the HRU strapdown navigation algorithms.

There are various approaches devised for performing the navigation computations. With regards to the different navigation algorithms, general guidelines have been developed concerning the required speed at which the various navigational quantities must be computed. All navigation elements computed in a fast rotating frame of reference, such as the attitude matrix, should be calculated at a fast rate. In contrast, latitude and longitude calculations which rely on a transformation between the earth and local-level frames of reference can be performed at a slower rate. However, calculations performed at a rate that is too slow may give rise to noncommutativity errors. In the paper [29], Bar-Itzhack condones performing computations at slow rates in systems where the use of electrostatic gyroscopes has eliminated the need for attitude computations.

The navigation algorithms for the HRU are developed in Chapter 6. Any relevant findings in the literature are applied where possible.

Chapter 4

The Heading Reference Unit

4.1 Introduction

The DREO Heading Reference Unit is an in-house Attitude and Heading Reference System designed at the DREO NavLab. In its current state the inertial sensor assembly contains two, TDF, CSG-2 tuned rotor gyroscopes and three single-axis Q-Flex QA1400 pendulous accelerometers. The aim of this thesis is to devise alignment and navigation algorithms and software specifically for the HRU. For this reason, a brief description of the system is given prior to formulating the detailed mechanizations.

4.2 General System Description

The ISA case is a 6 inch cube and the inertial sensors and associated thermistors and heater controllers are mounted in an internal sensor block. The relation between HRU system and electronic components is illustrated in Figure 4.1. The support electronics consists of a DREO Heater Circuit board for temperature regulation of the ISA, a

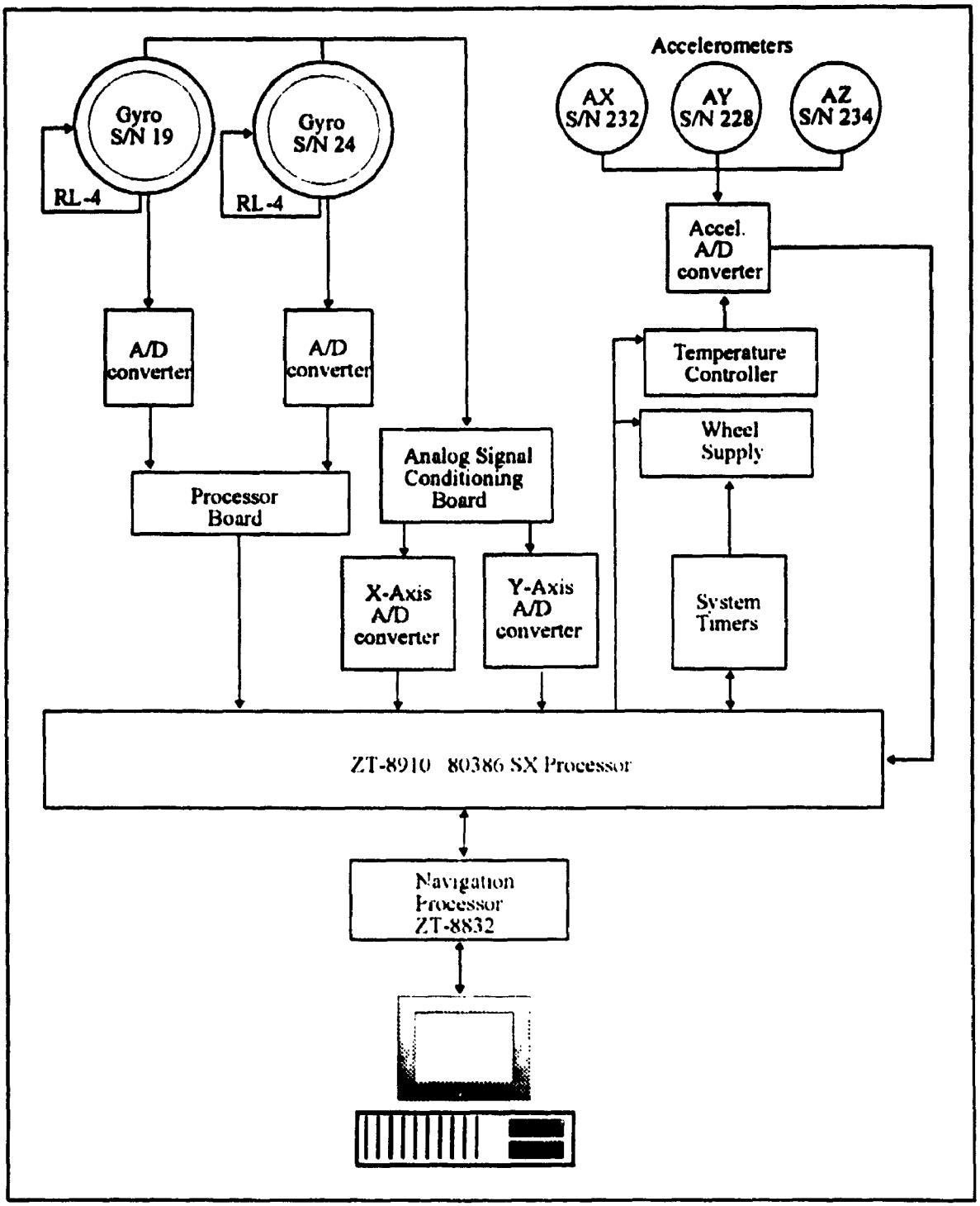


Figure 4.1: HRU Systems and Electronics Components. RL-4 is the analog rebalance loop used for rebalancing the two TDF gyroscopes. The CPU schedules I/O operations, interfaces with timers and processes interrupts. ZT-8832 is the designated navigation processor.

DREO Motor Circuit Board that generates the signal to drive the synchronous gyro motors and a separate pickoff reference. An ANCON Analog Signal Conditioning Board is dedicated to demodulating the gyro axes pickoff signals. The support electronics is well documented in [30]. An RL-4 analog rebalance loop controls the gyro over its dynamic range and provides precise angular rate measurements. The rebalance electronics was tested and evaluated to ensure it could sustain the rates experienced by a strapdown system.

The present data acquisition system encompasses a 10 channel, 24 bit Keithley 2001 multimeter for analog to digital conversion of the sensor data and software constructed to perform raw data collection. The DREO proprietary HRU CDU version 1.50 software stores the sensor outputs in separate files for the gyroscopes, accelerometers and temperature, respectively. It features a control display unit (CDU) for monitoring purposes and block averaging capabilities. The Keithley meter samples the channels sequentially, yielding discrete samples at a period of approximately 1.5 seconds per channel. The Keithley meter will be replaced with high resolution digitizers to achieve real-time data acquisition and simultaneous sampling of the data. The alignment and navigation algorithms are designed with the end configuration in mind but tested with the existing system assembly.

The current Central Processing Unit (CPU) is an Intel 80386SX, Ziatech ZT-8910 processor with clock speed of 16 MHz. An Intel 80387 co-processor is also available for floating-point operations. The CPU will perform such tasks as scheduling input/output (I/O) operations, interfacing with timers, and processing interrupts. A

ZT-8832, Intel 80186 compatible processor with 8087 math co-processor is to be used for calculating the navigation solution. System inputs and outputs are in IEEE-488 standard format.

In its final configuration the HRU will consist of a complete CDU for easy interfacing with an operator supplying initialization information. Monitoring of the system performance and navigation outputs will also be realized via the CDU.

The IMU yields rotation and acceleration information on all three orthogonal axes, hence the need for two gyroscopes and three accelerometers. The placement of the inertial sensors in the HRU is drawn in Figure 4.2. The corresponding photograph of the inertial sensors is shown in Figure 4.3. The orientation in which the inertial instruments are installed is important for defining polarities and axes definitions in the data error compensation equations.

The system axes, namely, roll, pitch and yaw, follow the convention of a right-handed, positive counter-clockwise, system. For example, in an aircraft positive roll angles constitute rotations that cause a right wing down movement, positive pitch angles correspond to a nose down manoeuvre and yaw rotations effectively rotate the heading causing a left turn. The Y-Case axes for both gyros are along the negative Z system axis. The case axes for both gyroscopes represent axes along which input angular velocities are sensed. The X-Case, Y-Case and Spin axes of a single TDF gyroscope also form an orthogonal right-handed triad. This accounts for the sign corrections applied to the raw inertial sensor outputs in Equations 5.1 and 5.2. Torquer axes are also denoted in Figure 4.2.

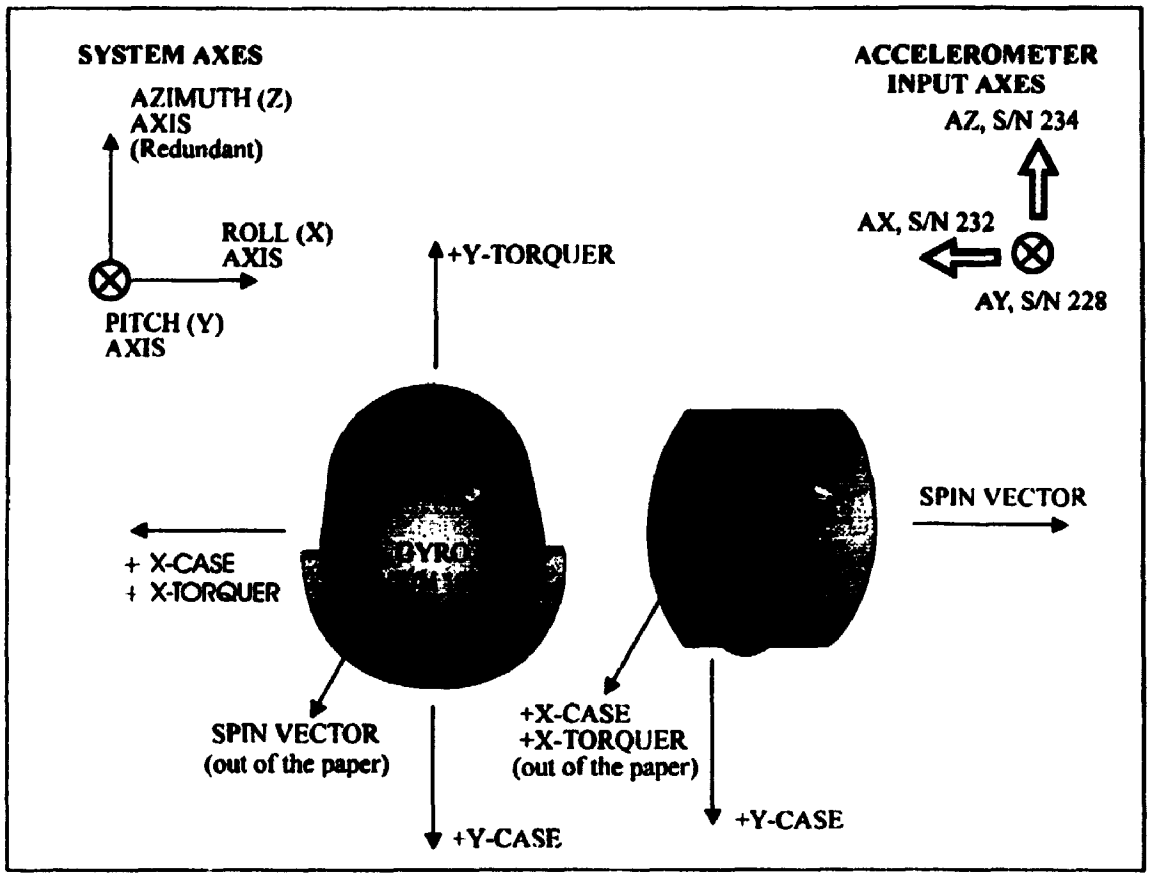


Figure 4.2: Placement of Inertial Sensors in the HRU. The system axes are the vehicle's frame of reference. Input angular rates are sensed about the gyro case axes. Linear accelerations are sensed along the accelerometer input axes.

For an individual gyroscope, a positive torque applied about the X-Torquer axis causes the rotor to precess in a positive direction about the Y-Case (or pickoff) axis. Similarly, a positive precession of the rotor about the X-Case axis occurs when a positive torque is applied about the Y-Torquer axis. This relation is important when implementing the software since the raw data collection is referenced to torquer coil currents whereas the alignment and navigation algorithms refer to gyroscope case axes. Furthermore, the polarity of Gyros S/N 19 and 24 X-Torquer coil outputs in the raw data collection

software is reversed according to the roll, pitch and yaw convention followed in this thesis and in the navigation software included in Appendix C.2.

Each accelerometer sensitive axis is mounted colinear with a gyro input axis. Hence the accelerometer axes are aligned along positive, or negative system axes.

Figure 4.3 is a photograph of the HRU inertial sensor block.

4.3 DREO Navigation Laboratory

One of the principal testing mechanisms used in the DREO NavLab is a two axis Contraves-Goerz motion table (CMT). The CMT, shown in a photograph in Figure 4.4, may be programmed to rotate to specific azimuth and tilt positions. The position commands may be entered manually via the Table console, or alternately, they may be automated by running a sequence of instructions programmed in the motion table language. Furthermore, precise rates may be applied to the tilt and azimuth axes of the table. One important characteristic of the CMT is that the azimuth axis and tilt axes rotate in negative directions according to the East-North-Up convention. For dynamic testing the different operational properties of the CMT's tilt and azimuth axes must be accounted for. A rotation about the azimuth axis al[illegible] stimulates the same instrument axis regardless of the position of the tilt axis, whereas the converse is not true. For static tests, this characteristic feature needs to be remembered when placing a package in different positions.

For the static coarse alignment tests, the HRU is mounted on the table top and it is precisely rotated to different attitude and heading orientations. The misalignments of the

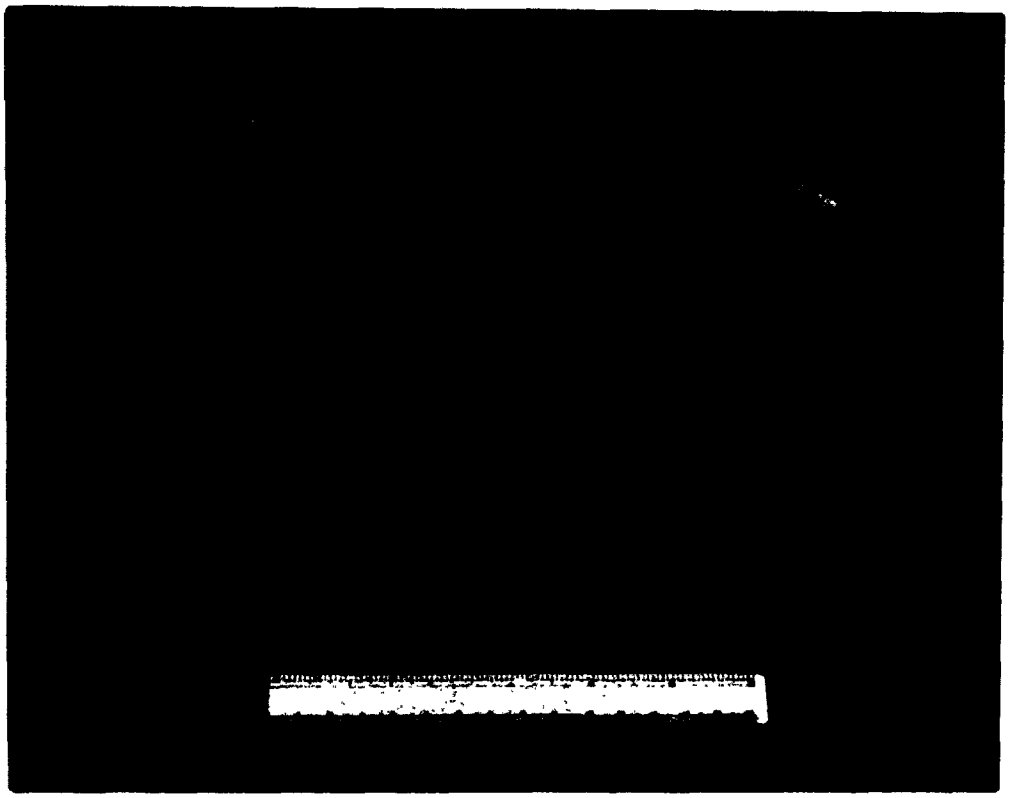


Figure 4.3: HRU Inertial Sensor Block. Note: Gyro S/N 19 is in foreground.

Figure 4.4: Contraves-Goerz Motion Table, DREO NavLab

motion table axes relative to each other can be neglected since these are on the order of three arc seconds. By properly selecting the order of the rotations, any sensor axis can be adjusted to point along any direction.

4.4 Thesis Justification

The intent of this thesis is to accomplish the following:

1. *Distinctive damped alignment scheme.* An innovative approach for injecting damping in a vertical control loop without the use of external aids is proposed by Staley in [31]. This concept is extended here in an analytic gyrocompassing and leveling method formulated for a complete AHRS. The object is to limit the growth of initial alignment errors and errors induced by instabilities in the system biases. The HRU is to operate as an entirely self-contained, unaided AHRS, thus the damped alignment scheme must be independent of external velocity references, such as from Doppler-radar.

2. *Alignment algorithms for strapdown AHRS.* The alignment algorithms are designed specifically for the DREO Heading Reference Unit. Special consideration is given to the types of sensors in the HRU and their associated error terms. Furthermore, by nature, a strapdown system lacks the flexibility of a gimballed system for performing alignment since vehicle relocation is restricted. A purely analytic approach to alignment is devised that does not depend on optical aids but instead takes advantage of the inherent capabilities of an inertial

navigation system.

3. *Minimal alignment time.* Alignment is typically broken up into two stages: coarse and fine alignment. Fine alignment is typically a time-consuming stage that refines the initial estimate. However, it can be inconvenient to wait for a system to finish aligning before departing. For this reason, the proposed fine alignment scheme functions in the presence of vehicle motion. Furthermore, the alignment procedure does not rely on the execution of specific manoeuvres to be effective. The vehicle can start navigating immediately after the coarse alignment stage. The alignment time before departure is minimal and the accuracy of the navigation solution improves as the in-motion alignment progresses.

4. *Differential angular velocity inputs.* The digitized gyroscope outputs are readings of the angular velocities, ω (deg/sec), the gyroscope input axes sense. Thus, the navigation algorithms are geared for input data that is in the form of a derivative, $d\theta/dt$, as opposed to the more common form consisting of incremental angles ($\Delta\theta$) over constant time intervals.

5. *Navigation algorithms for strapdown AHRS.* An optimum navigation mechanization, specifically for the DREO Heading Reference Unit is proposed. The requisite goal is to compute the desired attitude, heading, position and velocity outputs in an efficient, systematic manner.

6. *Practical software implementation.* The alignment and navigation algorithms are

coded in Borland Turbo C++ version 3.0. Simulated test tracks are generated for testing purposes. Also, the software is used to post-process raw data from the HRU.

The system algorithm in Figure 4.5 illustrates the two modes of operation for the HRU, namely: alignment and navigation. An operator is needed to select the mode of operation and enter the parameters for system initialization. The inputs to the navigation and alignment algorithms are angular velocity, ω, from the gyroscopes and acceleration, a, from the accelerometers. The inertial sensor data is compensated for static and dynamic errors before it is used to compute the system outputs.

At the output of the data error correction block the data is in the body-system frame of reference. The attitude matrix used to compute the Euler angles is updated by applying the necessary precession rates and integrating the quaternion parameters. The DCM is also used to transform the accelerometer data to a computational frame of reference. Accelerometer information is corrected for gravity and coriolis accelerations. It is then integrated to obtain navigation velocity and position outputs.

The block diagram in Figure 4.5 delineates the overall system inputs and outputs and gives a general overview of the process implemented in this thesis.

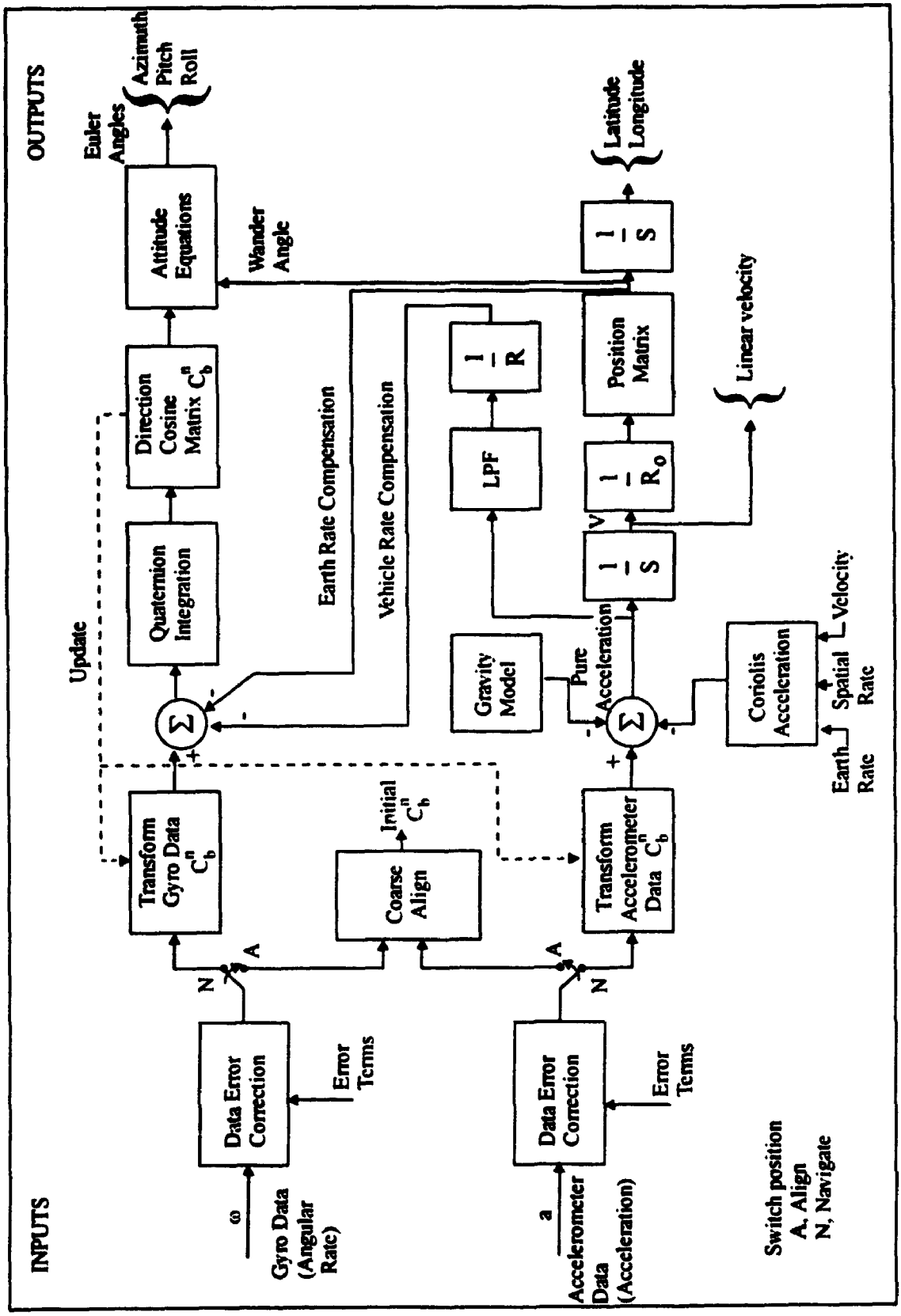


Figure 4.5: Overall HRU Alignment and Navigation System Algorithm

Chapter 5

Theoretical Formulation of Alignment Algorithms

5.1 General Overview

Inertial navigation is a process whereby the vehicle's attitude, heading and position is computed based on incremental differences from known previous values. This form of dead-reckoning depends on initial conditions supplied by an alignment procedure. Thus, before navigation mode can commence, the HRU must be aligned to a known frame of reference.

Alignment for the HRU is subdivided into coarse and fine alignment stages. The approach used does not require any external references and it is designed to take a minimum amount of time. Analytic alignment combined with an initialization phase to start the process, is the precursor to the navigation algorithms.

The coordinate frames of reference used in the development of the navigation and alignment algorithms are introduced in this chapter. The data compensation algorithms needed to correct the raw inertial sensor data for known error terms are then developed.

Finally, an in-depth description of the coarse alignment, fine leveling and gyrocompassing schemes is given.

5.2 Coordinate Frames of Reference

The derivation of strapdown inertial navigation equations relies on specific coordinate frames of reference. The frames of reference that have been studied in the development of the Heading Reference Unit equations are the inertial, local geographic, body, local-level wander-azimuth and earth-centered inertial. The coordinate frames of reference are depicted in Figure 5.1.

The *geographic frame* is a local-level frame of reference. It can have positive axes defined as either North, East, Down; or East, North, Up. The location of the HRU defines the origin of this frame of reference. The Up (U) axis is defined as the normal to the reference ellipsoid used to model the earth. The North (N) axis is oriented along the projection of the inertial angular velocity vector of the earth in the local horizontal plane. Finally, the East (E) axis completes the right-hand orthogonal triad. The convention followed in this report is East=X, North=Y and Up=Z. The azimuth analytical precession rate required to keep the reference frame north-pointing approaches infinity at high latitudes. A geographic local-level system is feasible for low and moderate latitudes but not near the poles.

The second frame of reference is the *body frame* or *system frame* of the vehicle. It consists of the roll, pitch and yaw axes of the vehicle, and its origin is typically at the center of mass of the vehicle. This does not imply that the origin of the body frame is

coincident with the location of the Attitude and Heading Reference System. The relation between gyroscope and accelerometer frames of reference and the body frame must also be determined for specific applications.

The *local-level wander-azimuth frame* is also referred to as the computational or navigational frame. Its origin is coincident with that of the geographic frame. The horizontal axes of this frame lie in a plane that is tangent to the local vertical. Eulerian rotations of longitude, geodetic latitude and wander angle are used to relate the local-level wander-azimuth frame to the earth. The convention is to define latitude as being positive in the northern hemisphere and the wander angle is positive west of true North. The wander angle is measured in the geodetic horizon plane. The local wander-azimuth and geographic frames are identical for a wander angle of zero [22]. The platform angular precession rate for the level axes is identical for wander-azimuth and geographic mechanizations. However, a north-pointing heading is not maintained thereby removing any singularities at the poles and the requirement for high azimuth angular rates. Only the earth rate components are removed from the azimuth axis. The wander angle is defined for low and moderate latitudes and the mechanization will not fail at the poles. Longitude and heading calculations are omitted at high latitudes.

The *inertial frame* is the only frame in which Newton's laws are directly valid. This is because its absolute motion is zero. This frame of reference is used only as an aid in visualizing the other frames of reference. Unlike the inertial frame the *earth-centered inertial frame*, accelerates with respect to inertial space. It moves with the earth and has its origin at the earth's center of mass [3].

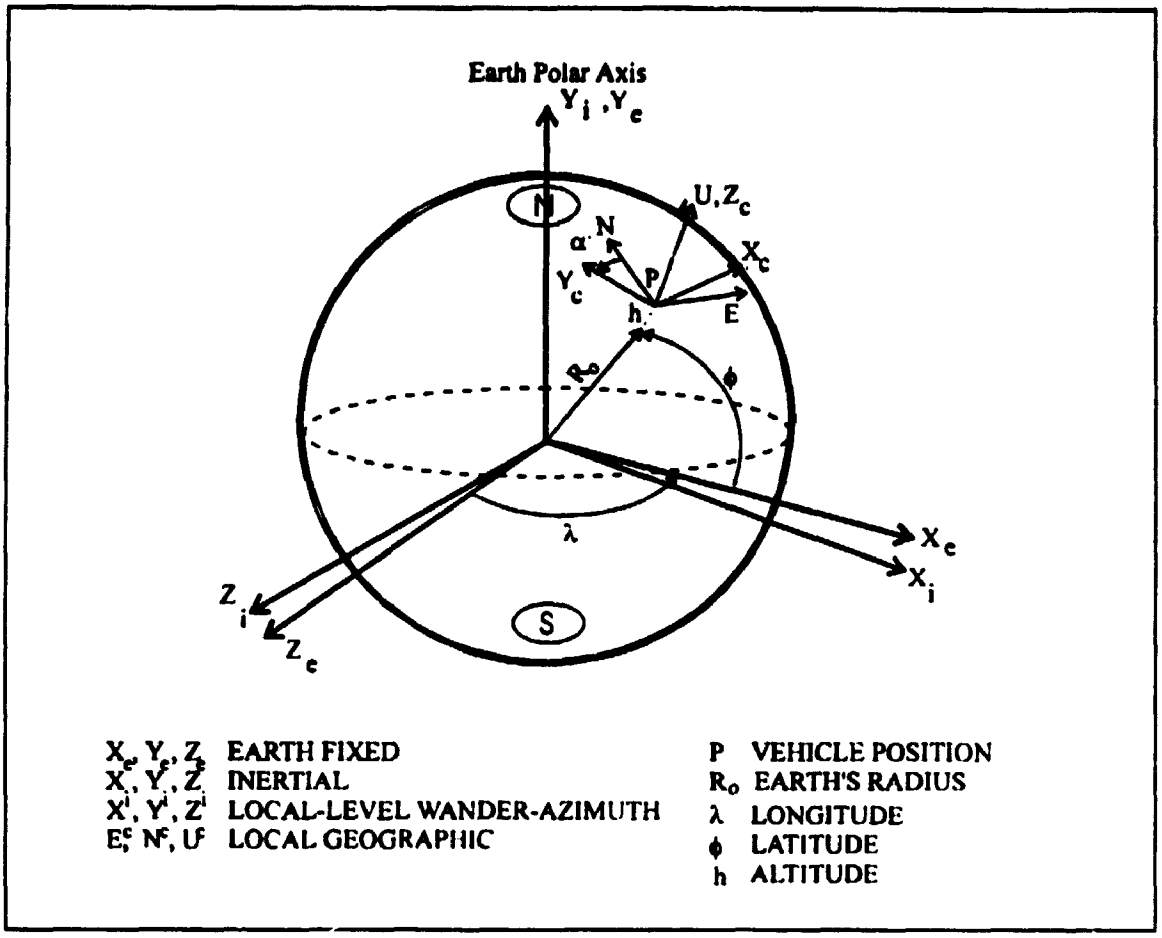


Figure 5.1: Coordinate Frames of Reference

5.3 Inertial Sensor Data Compensation

The first phase of the data analysis consists of compensating the raw sensor outputs for known error terms. This is necessary because the subsequent alignment and navigation algorithms require compensated angular velocity and acceleration measurements. Thus, the data compensation equations are applied at a minimum rate that is equal to the fastest loop in the navigation processor. Explicit equations that model the performance of the HRU inertial sensors are developed. The equations are specific to the type of sensor

used, for example, acceleration sensitivities would not need to be specified for laser gyros.

As described in Chapter 3, the HRU inertial sensors are: two TDF CSG-2s and three pendulous Q-Flex, QA1400 accelerometers. The dominant first-order error terms associated with these instruments, corrected in the data compensation algorithm are the following:

- *accelerometer scale factor*, the change in output is related to the change in input signal by the scale factor. It is typically defined as the slope of the best fit straight line over a specified range of input signals [7]. Additional thermal compensation may be needed to ensure scale factor stability and a higher order scale factor model may be warranted to compensate for non-linearity effects. It is expressed in units of mA/g.

- *accelerometer bias*, a constant offset in the output that exists even when there is no acceleration. Biases can be due to an interaction between the sensing element and a magnetic field or mechanical and electrical pickoff nulls not coinciding. It has units of mg.

- *gyroscope torquer scale factor*, is used to convert mA outputs to deg/hr. Higher order models can be used to account for temperature sensitivities and repeatability.

- *gyroscope bias drift*, the offset output when there is no input signal, that has

no correlation with acceleration, but is due to inaccurate null setting. It is expressed in deg/hr.

- *gyroscope mass unbalance(MU) and quadrature(Q) terms*, are g-sensitive drift terms. An acceleration along the X-axis of the gyroscope causes an inertial torque about the Y-axis which causes gyro drift (MU) about the X-axis. Similarly, the quadrature term is a g-sensitive drift about the opposite axis to that along which the acceleration acts [7]. These have units of deg/hr/g. They are a consequence of fabrication imperfections due to mechanical assembly tolerances.

- *misalignment angles*, these are the angles that relate accelerometer and gyroscope input axes to the system body axes of the unit, used in the navigation algorithms. The instruments are installed in the nominal system axes. However, assembly tolerances and surface imperfections result in small misalignments. Both sets of misalignment angles are related to a common frame of reference, therefore the angular displacement between any accelerometer and gyroscope axes is known. This information is needed to calculate the mass unbalance and quadrature components.

The scale factors, biases, MU and Q uncertainties depend on the physical properties of gyroscopes and accelerometers. In contrast, the misalignment angles are a result of machining imperfections on the sensor block and the mechanical tolerances during the installation of the instruments on the sensor block.

With the use of an automated laboratory procedure to accurately determine the instrument misalignments, it is no longer necessary to precisely align the instruments using optical references. The laboratory method is particularly well-suited for a strapdown navigation system because the known misalignment angles can be easily included in the navigation software. The system needs only to be calibrated once since the misalignments are internal to the instruments and do not change. Moreover, as long as the instruments remain fixed in the package, the angles relating separate gyroscopes or accelerometers do not change either

The laboratory calibration procedure consists of two independent tests, one for each type of sensor. Through the use of carefully chosen rotations and knowledge of the resulting orientation of the package, a set of equations which accurately models the position of the instruments and their characteristics can be developed. These equations, in conjunction with the measured instrument outputs, are used to resolve the relative orientation of the instruments axes to each other.

A static multiposition test is one way of determining the magnitude and signs of these error terms. Alternately, the error terms associated with the gyros can be solved for using dynamic rate testing. One type of multiposition test that can be used to determine the misalignments between the accelerometer axes consists of placing the AHRS in six positions, all of which can be specified in terms of constant rotations about the tilt and azimuth axes away from an arbitrarily assigned home position. The positions are chosen such that each accelerometer axis is oriented to sense the full force of gravity (g), in the positive and negative directions, as well as zero gravity. Equations are

derived which describe the accelerometer outputs in terms of their bias errors and the misalignment of the gravity vector from the East-North-Up coordinates. Accelerometer bias, linear scale factor, gravity's misalignment from geographic coordinates and accelerometer misalignment angles are the unknowns solved for with the 18 equations that model the accelerometer input vectors.

The gyroscope error terms can be found from a static multiposition test that successively places each gyroscope case axis such that it senses full positive, negative and zero earth rate. Additional positions that place each individual gyro axis such that it senses the force of gravity in the positive and negative directions are also needed to resolve the acceleration-sensitive drift errors. A second method for resolving the gyro error terms consists of subjecting the gyro input axis to positive and negative table rates. Applied rates on the order of 10 deg/sec are optimum since they make drift and earth rate terms negligible about the axis that is being rotated. Torquer heating effects would need to be considered for rates in excess of 30 deg/sec. The gyro characteristics can be resolved from a set of 24 simultaneous equations using the latter method. The error terms found from multiposition testing are used in the data compensation algorithm.

More detailed descriptions of optimal multiposition alignment techniques are given in [32] and [33]. Precise calibration is needed to remove navigation errors that are a result of inaccuracies in the scale factors and misalignments.

The quadrature, mass unbalance and scale factors used in the software are those supplied by Litton test data and Acceptance Test Procedure results for the gyroscopes and accelerometers, respectively. However, differences in the rebalance loop used for

testing and electrical biases, necessitate that the accelerometer and gyroscope biases be re-calculated. Thus, the biases were determined using a simplified static 6-position test. A more complex series of tests with more positions, as described previously, is needed to resolve misalignment angles. Thus, it is necessary to approximate these to zero. The values of the error terms used in the data compensation algorithm are specified in the software documentation.

Equations 5.1 to 5.3 below, illustrate how the compensated angular rate and acceleration outputs are calculated from raw inertial sensor outputs in the data error compensation module. By convention, ω_x refers to the angular rate sensed by Gyro S/N 19 X-Case axis, and similarly for the other axes.

$$\begin{bmatrix} -\omega_x \\ -\omega_y \\ -\omega_z \end{bmatrix} = \begin{bmatrix} 1 & G_{yx} & G_{zx} \\ G_{xy} & 1 & G_{zy} \\ G_{xz} & G_{yz} & 1 \end{bmatrix} \cdot \left(\begin{bmatrix} GK_x & 0 & 0 \\ 0 & GK_y & 0 \\ 0 & 0 & GK_z \end{bmatrix} \cdot \begin{bmatrix} \omega_{xr} \\ \omega_{yr} \\ \omega_{zr} \end{bmatrix} - \begin{bmatrix} GB_x \\ GB_y \\ GB_z \end{bmatrix} - \begin{bmatrix} Q_x \cdot a_{G_{z1}} \\ Q_y \cdot a_{G_{z2}} \\ Q_z \cdot a_{G_y} \end{bmatrix} - \begin{bmatrix} MU_x \cdot a_{G_{xz}} \\ MU_y \cdot a_{G_{xy}} \\ MU_z \cdot a_{G_{xz1}} \end{bmatrix} \right) \tag{5.1}$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & A_{yx} & A_{zx} \\ A_{xy} & 1 & A_{zy} \\ A_{xz} & A_{yz} & 1 \end{bmatrix} \cdot \left(\begin{bmatrix} AK_x & 0 & 0 \\ 0 & AK_y & 0 \\ 0 & 0 & AK_z \end{bmatrix} \cdot \begin{bmatrix} a_{xr} \\ a_{yr} \\ a_{zr} \end{bmatrix} - \begin{bmatrix} AB_x \\ AB_y \\ AB_z \end{bmatrix} \right) \tag{5.2}$$

$$\begin{bmatrix} a_{G_x} \\ a_{G_y} \\ a_{G_{z1}} \\ a_{G_{z2}} \end{bmatrix} = \begin{bmatrix} 1 & G_{xy} & G_{xz} & 0 \\ G_{yx} & 1 & G_{yz} & 0 \\ G_{z1x} & G_{z1y} & 1 & 0 \\ G_{z2x} & G_{z2y} & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \\ a_z \end{bmatrix} \tag{5.3}$$

Gyro #1 (S/N 19) Y-Case axis provides redundant information along the Z-system axis. Gyro #2 (S/N 24) Y-Case is chosen as the reference angular rate output along the Z-axis because it exhibits lower random drift characteristics along the vertical axis than Gyro #1

Y-Case. At present, it is preferable to use the data from the more accurate gyroscope, thereby reducing the amount of data collection. However, the redundant gyro axis should still be monitored for back-up purposes. It could be used as a replacement in the event of a failure of the primary azimuth gyro axis. System operation cannot be maintained if the failure affects both axes of the same gyro. Alternatively, the redundant information can be used to improve the system's accuracy.

The order in which the error corrections are applied is important. As shown in equation 5.2 the scale factor correction is applied first to the raw accelerometer outputs. Accelerometer biases are then removed and finally the measurements are resolved along the body system axes through multiplication by the matrix of misalignment angles. A small angle approximation is used to derive the misalignment angle matrix from a sequence of three rotations. The misalignment angles are positively defined for a counter-clockwise rotation from the sensor to system axes. A_{yx} refers to the misalignment angle between the Y-Body axis and the X-System axis, with similar definitions for the remaining misalignment matrix elements.

Similarly, scale factor correction is applied first to the raw gyroscope outputs. Secondly, bias drift is subtracted from the outputs. When an acceleration is sensed along a gyro axis, g-sensitive gyroscopic drifts result about the axis upon which the acceleration acts and about the opposite axis. These are the MU and Q terms, respectively. In a space-stabilized system, only the g-sensitive drift terms resulting from the gyroscope axis oriented along the vertical are significant. However, in a strapdown system the orientation of the AHRS follows the vehicle and it is possible for any gyro

axis to detect gravity. This is true even for terrestrial applications since the roll and pitch angles can reach up to 30 degrees. Thus, to compensate for these terms the accelerometers are used to determine the gravity components along all four gyroscope axes. The acceleration along the redundant gyro axis is needed for this calculation because it produces a quadrature g-sensitive drift term along the Gyro #2 X-Case axis. The system axis accelerations are transformed to accelerations along the gyroscope axes by multiplying through a matrix of misalignment angles that is the transpose of the misalignment angle matrix, refer to equation 5.3, in keeping with the convention that the first subscript of the misalignment angle refers to the body axis of the sensor and the second refers to the system axis. The exception in the previous statement is that the transpose matrix includes additional parameters for the misalignment between the redundant gyro axis and the system axes. This yields a four-parameter vector for the accelerations sensed along each gyroscope axis.

The last matrix multiplication required applies sign corrections to the gyroscope and accelerometer outputs. These depend on the physical orientation of the sensor input axes relative to the defined system axes. The placement of the sensors relative to HRU roll, pitch and yaw system axes are illustrated in Figure 4.2, Chapter 4.

For simplicity in presenting the equations, the required unit conversions have not been shown. The unit conversions are implemented in the software thereby yielding system level outputs in rad/sec and m/sec^2 for the gyros and accelerometers, respectively. All future references to body angular rates and linear accelerations imply compensated inertial sensor outputs along system axes.

Separate algorithms are needed to compensate for dynamic errors. Coning errors result when there is a net rotation about a third axis due to a time-varying angular velocity about the other two axes. Therefore, a coning algorithm is needed to correct the gyro output data. Sculling error is a dynamic error term that results when the accelerometer sensor data is transformed to the computational frame of reference through a direction cosine matrix. The data compensation algorithm only corrects errors directly related to the instrument characteristics. Errors that are a consequence of the digital processing techniques are considered in Chapter 7.

5.4 Coarse Alignment

The initial mode of operation for an INS is the alignment stage. An INS must first be initialized before it can be used for navigation. An external known reference can be used to transfer align the INS or alternatively optical means of aligning inertial systems exist. The self-alignment scheme used for the HRU is entirely analytical and does not require any external optical aids. However, if the vehicle is in motion then an external source of information is needed to acquire initial alignment. Coarse alignment is only the first phase of automatic alignment. The fine alignment scheme described in the next section refines this initial estimate by damping out any initial heading and level errors. Vehicle motion does not impair the fine alignment stage as will be shown in the next section.

When an INS is stationary the total angular rate sensed by the gyroscopes is equal to full earth rate, 15.0417 deg/hr. Similarly, the total linear accelerations sum up to 1g. A reasonably accurate coarse alignment can be achieved using solely the inertial sensor

outputs and operator supplied latitude coordinates. Effectively, the natural vectors, gravity and earth rate, are used to level and azimuth align the computational frame of reference. The analytic coarse alignment algorithm is based on ideas presented in Chapter 9 of Britting's book [22]. The measurements obtained from the sensors need to be aligned to a computational frame of reference. Thus, the alignment process of the HRU consists of defining the rotation matrices. The two principle rotation matrices are: the quaternion relating the body frame of reference to the computational frame of reference and the direction cosine matrix relating the computational and earth frames of reference.

The self-alignment procedure is more complex for a strapdown system than a gimballed system because gyroscope movement is restricted in a strapdown system. In a gimballed system, error terms such as bias drift are easily removed by changing the orientation of the inertial navigation system and taking measurements in each configuration. This method is called gyrocompassing [20]. However, strapdown systems are fixed to the vehicle and rotating the whole vehicle by 180 degrees is not always a practical alternative.

The derivation described herein is for a geographic frame of reference. During the initial alignment stage the wander-azimuth and geographic frames of reference can be considered coincident. This is true for a wander-azimuth mechanization that starts with an initial wander angle of zero. The wander angle rate is a measure of the control torque required to maintain a north-pointing reference frame. The angular rate of the coordinate frame relative to the earth is a function of vehicle velocity, thus the wander

angle does not change if the vehicle remains motionless. A more detailed explanation of the wander angle is given in Chapter 6 of this thesis. It is only mentioned here to demonstrate that wander angle correction is not needed during coarse alignment.

5.4.1 Open-Loop Alignment Algorithm

The coarse alignment is an open loop alignment scheme. This implies that the method is not recursive and there is no feedback of the outputs into the beginning of the loop. The result is an initial estimate of the transformation matrix which relates body and geographic coordinates. The coarse alignment method is an analytic procedure that does not physically align the inertial navigation system. Instead, the computed offset misalignment angle is maintained in the computer.

The initialization and coarse alignment procedures require an operator of the system to provide initial latitude and position information to begin the process. To resolve the gyroscope outputs into local geographic coordinates the latitude must be known. If a wander angle other than zero is used than it must also be input by the operator. For this analysis the vehicle is stationary and a wander angle of zero is assumed. In addition to the information supplied by the operator, the gyroscopes and accelerometers sense gravity and earth rate. Thus, the process of calculating the attitude and heading of the vehicle using the inertial sensor outputs and latitude is called coarse alignment. Also, in order to continuously update the position of the vehicle, initial longitude information must also be provided. This latter process is an initialization sequence and it consists of defining the direction cosine matrix relating earth-fixed coordinates to the local-level

wander-azimuth frame of reference, refer to Chapter 6 §6.5. The following discussion focuses specifically on the process for determining the vehicle's initial attitude and heading.

Single point measurements from the inertial sensors are inadequate because they do not coincide exactly with the actual parameter. An optimum averaging scheme which minimizes the duration of the coarse align stage and maximizes the accuracy of the results is also implemented. For the purpose of this analysis it can be assumed that the means of the raw outputs are normally distributed. The time required for coarse alignment is dictated by the number of measurements taken. Under normal environmental conditions, coarse alignment should be realized in under ten minutes.

The natural vectors, expressed in navigational coordinates, are transformed to body coordinates via a DCM, as shown in expressions 5.4.

$$\begin{aligned} \vec{\omega}^{b}_{ie} &= \mathbf{C}^{b}_{n} \cdot \vec{\omega}^{n}_{ie} \\ \vec{g}^{b} &= \mathbf{C}^{b}_{n} \cdot \vec{g}^{n} \end{aligned} \tag{5.4}$$

This concept can be extended to apply to data in local geographic **G** and body **B** reference matrices as shown in equation 5.5.

$$\mathbf{B} = \mathbf{C}^{b}_{n} \cdot \mathbf{G} \tag{5.5}$$

Hung and White in their paper [20] describe a zero-torquing platform self-alignment scheme for fixed base gimballed inertial measurement units. They emphasize the importance of using a simpler, less time-consuming special matrix inversion algorithm that makes use of matrix partitioning techniques. However, complex matrix inversion routines can be circumvented altogether by using a direction cosine matrix since it is

orthogonal, so that taking the transpose is equivalent to finding its inverse. A notation for the transpose is given in equation 5.6.

$$\mathbf{C}_b^n = \left(\mathbf{C}_n^b\right)^T \tag{5.6}$$

Similarly, **G** is an orthogonal matrix. Using the identity in equation 5.6 and the definition in equation 5.5 the DCM relating body coordinates to geographic coordinates can be solved for according to equation 5.7.

$$\mathbf{C}_b^n = \mathbf{G} \cdot (\mathbf{B})^T \tag{5.7}$$

The elements of the *Geographic align matrix*, **G** and *Body align matrix*, **B** matrices are all known values. The sensor measurements are appropriately manipulated and normalized to yield the transpose of **B**. Initial latitude information is all that is required to define the **G** matrix.

The orthonormal vectors $\vec{\omega}$, $\vec{\varsigma}$ and $\vec{\xi}$ are used to obtain **B**. In the body-referenced coordinate system, $\vec{\omega}$ is the vector of angular rate outputs from the gyroscopes, $\vec{\varsigma}$ is the cross product $\vec{\omega} \times \vec{g}$ and, $\vec{\xi}$ is the cross product $\vec{\omega} \times (\vec{\omega} \times \vec{g})$ where $\vec{g}$ is the gravity vector with components from the three accelerometers. The elements of **G** and **B** are substituted in the equation for the DCM resulting in equation 5.8.

$$\mathbf{C}_b^n = \begin{bmatrix} \omega_x^n & \varsigma_x^n & \xi_x^n \\ \omega_y^n & \varsigma_y^n & \xi_y^n \\ \omega_z^n & \varsigma_z^n & \xi_z^n \end{bmatrix} \cdot \begin{bmatrix} \omega_x^b & \varsigma_x^b & \xi_x^b \\ \omega_y^b & \varsigma_y^b & \xi_y^b \\ \omega_z^b & \varsigma_z^b & \xi_z^b \end{bmatrix}^T \tag{5.8}$$

The corresponding **G** is defined using the normalized earth rate vector $\vec{\omega}$ and the normalized gravity vector $\vec{g}$ in matrices 5.9 a, b, on the following page.

$$\vec{g} = \begin{bmatrix} 0 \\ 0 \\ \frac{g}{|g|} \end{bmatrix} \qquad \vec{\omega} = \begin{bmatrix} 0 \\ \frac{\omega_{i/e} \cdot \cos(\phi_L)}{|\omega_{i/e}|} \\ \frac{\omega_{i/e} \cdot \sin(\phi_L)}{|\omega_{i/e}|} \end{bmatrix}$$

$$\vec{\varsigma} = \begin{bmatrix} \frac{\omega_{i/e} \cdot \cos(\phi_L) \cdot g}{|\omega_{i/e} \cdot \cos(\phi_L) \cdot g|} \\ 0 \\ 0 \end{bmatrix} \qquad \vec{\xi} = \begin{bmatrix} 0 \\ \frac{\omega_{i/e}^2 \cdot \sin(\phi_L) \cdot \cos(\phi_L) \cdot g}{|\omega_{i/e}^2 \cdot \cos(\phi_L) \cdot g|} \\ \frac{-\omega_{i/e}^2 \cdot \cos(\phi_L) \cdot \cos(\phi_L) \cdot g}{|\omega_{i/e}^2 \cdot \cos(\phi_L) \cdot g|} \end{bmatrix} \tag{5.9a-d}$$

The local Geographic Align matrix is formed using equations 5.9 b, c, d as the columns of a 3×3 matrix. The **G** matrix is defined in terms of known quantities, namely: earth rate, latitude and acceleration due to gravity. The triad of orthonormal vectors specified above is chosen because it leads to the formation of an orthogonal matrix. This is easily verified since the dot product of any two columns is zero and has a determinant equal to ± 1.

Britting describes a similar procedure where a different set of non-normalized vectors are used to form the **G** matrix [22] The advantage of using equations 5.9 a, b, c to form the local geographic align matrix is that it eliminates all singularities at the poles. The chosen simplified **G** matrix is shown in equation 5.10.

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 \\ \cos(\phi_L) & 0 & \sin(\phi_L) \\ \sin(\phi_L) & 0 & -\cos(\phi_L) \end{bmatrix} \tag{5.10}$$

The second transformation matrix needed to initialize the system for navigation is the

DCM that relates computational frame of reference coordinates and earth frame coordinates. It is calculated using initial latitude and longitude positions. It is obtained at the start of the navigation loop used in the fine alignment and navigation modes of operation, accordingly it is explained in Chapter 6.

5.4.2 Statistical Data Analysis

The analytic coarse alignment scheme is designed to function provided there is no excessive base motion and the instrument output noise levels fall within acceptable margins. As stated in the previous section the average gyroscope and accelerometer outputs are used to calculate the **B** matrix. The static data is assumed to be normally distributed, meaning there is a 99.7% probability that the bias instability, for instance, will not exceed 3σ, where σ^2 is the variance. It is defined as the square of the standard deviation which can be measured.

The general form of the equation for calculating the mean of a large sample is given by equation 5.11 [34].

$$\overline{X} = \frac{1}{N} \cdot \sum_{i=1}^{N} X_i \tag{5.11}$$

However, calculating a cumulative sum and sample total is not the best approach for finding the mean of a large number of samples. This would require keeping two running totals that would rapidly exceed the bounds of double precision arithmetic thereby resulting in truncation and rounding errors. Consequently, sequentially calculating the mean as per equation 5.12 is preferable.

$$\bar{X}_N = \frac{N-1}{N} \cdot \bar{X}_{N-1} + \frac{1}{N} \cdot X_N \tag{5.12}$$

However, to ensure that the average is an accurate representation of the sensed accelerations and angular rates, the variance of the mean is also needed. The definition for sample variance, denoted by σ^2 is given by equation 5.13 [34].

$$\sigma_N^2 = \frac{1}{N-1} \cdot \sum_{i=1}^{N} (X_i - \bar{X})^2 \tag{5.13}$$

Using equations 5.11 to 5.13 an equation for sequentially calculating the variance is derived, refer to the result in equation 5.14. This is a more amenable way of calculating the variance for a large set of sequential samples. The original definition required storage space for all the individual samples to calculate the sum of products. In contrast equation 5.14 makes use of the previous estimates of the mean and variance.

$$\sigma_N^2 = \left(\frac{N-2}{N-1}\right) \cdot \sigma_{N-1}^2 + (\bar{X}_{N-1} - \bar{X}_N)^2 + \frac{(X_N - \bar{X}_N)^2}{N-1} \tag{5.14}$$

The mean of all the inertial instrument outputs is successively computed until one of two decision conditions is satisfied. Calculated confidence intervals are used to evaluate the decision condition. The mean of a normal distribution when the standard deviation is known lies within a confidence interval. For a 95% confidence interval the mean is expected to fall within the confidence limits an average of 95 times in 100. Thus, the confidence limits can be set according to the accuracy of the instrument outputs. A probability percentage is chosen that yields the best possible mean estimate that is neither too stringent nor relaxed. The mean lies within ± half the confidence interval (CI), where CI is defined as per equation 5.15.

$$\frac{CI}{2} = \frac{t \cdot \sigma_N}{\sqrt{N}} \tag{5.15}$$

The t-parameter is the 100α/2 percentage point of the t distribution with n-1 degrees of freedom. The value of 't' can be found in Statistical tables for a given number of samples and confidence coefficient. For a large number of samples, the 't' parameters for the accelerometers and gyroscopes can be estimated as per equation 5.16. Confidence coefficients of 0.99 and 0.95 are used for the accelerometer and gyroscope data, respectively. Thus, the gyro mean falls within confidence limits 95 times out of 100, whereas the accelerometer mean falls within confidence limits 99 times out of 100. A smaller confidence coefficient is assigned to the gyro data because it exhibited noisier characteristics.

$$\begin{aligned} t_{acc} &= 2.576 + \frac{4.92}{N-1} \\ t_{gyr} &= 1.960 + \frac{2.4}{N-1} \end{aligned} \tag{5.16}$$

The gyro output mean should not vary according to turn-on to turn-on bias instability, random drift and electrical noise. Thus, a confidence interval of 0.4 deg/hr was selected. Similarly, accelerometer sensitivities warrant a confidence interval of 0.00010 g . Note that these are the recommended default intervals. For a more detailed discussion on confidence interval estimation the reader is referred to Chapter 8 of [34].

The decision rule encompasses two conditions, the first describes the action to be taken when the confidence interval condition is satisfied, the second limits the sample size. The first decision condition compares all simultaneously calculated confidence intervals with standard limits. When the means of all the instrument outputs fall within

the confidence intervals, data collection is ended and the Geographic Align Matrix is calculated. If confidence interval estimates are not within range after 20 minutes of operation then the second decision condition is reached and alignment is deemed to be unsuccessful.

5.5 Fine Alignment

Alignment is not complete after the open loop coarse align scheme because this first stage is valid only for a benign environment where the vehicle is not subjected to angular disturbance vibrations or accelerations. However, it is impossible to completely isolate the INS from all disturbances. Thus, the effect of these disturbances on the gravity and earth rate measurements must be accounted for. Furthermore, the effect of instrument uncertainties such as turn-on to turn-on bias repeatability and instrument non-orthogonality contribute to form an estimated transformation matrix that is misaligned from the actual transformation matrix.

A Schuler-tuned fine align stage is needed to correct the small angle misalignments between the indicated computational frame of reference and the true frame of reference. Refer to Figure 5.2 for an illustration of the misalignment error angles.

The concept of Schuler-tuning can best be explained by considering a simple pendulum suspended over the Earth. When the vehicle is at rest, the pendulous axis of the pendulum coincides with the true vertical direction. However, when the pivot is accelerated the pendulum becomes aligned along the apparent vertical axis which is offset from the true vertical axis. Thus, the apparent vertical corresponds to the

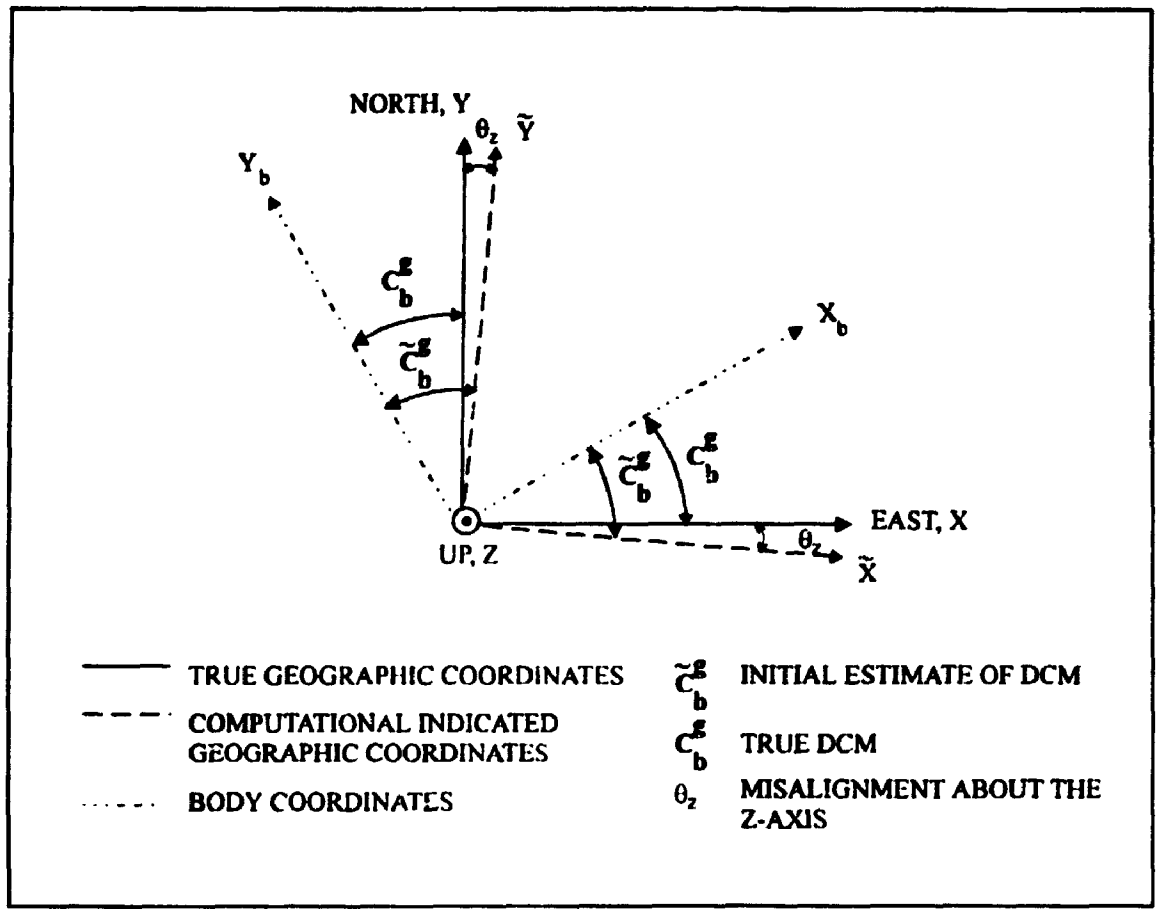


Figure 5.2: Misalignment Error Angles. These are the small angles relating the true geographic coordinates and the estimated geographic coordinates found during coarse alignment.

direction of the nonfield specific force, defined by equation 6.12. Therefore, a pendulum that remains aligned with the true vertical axis in the presence of accelerations would have its pivot at the Earth's surface and center of mass at the center of the Earth Such a pendulum is physically unrealizable, however, by tuning the pendulum at the Schuler frequency, derived in §6.2, the angular velocity of the pendulum equals the angular velocity of the vertical. Hence, the pendulum remains aligned with the vertical in the presence of accelerations, and initial misalignments oscillate about the vertical with an

undamped natural period of 84.4 minutes. The magnitude of the initial error dictates the amplitude of the undamped sinusoidal wave. The initial misalignments can be damped by altering the Schuler-tuning condition, and this is demonstrated in the subsequent sections of this chapter.

There are many different ways of refining the initial estimate of the DCM. *Linear Optimal Filtering* is one such method. A linear optimal filter can be implemented as a self-corrective recursive loop that continuously updates the error angles between the indicated and true frames of reference. Accelerometer and gyroscope measurements are used to solve for the error angles. The initial estimate of $\mathbf{C}_b^n$ is rotated by these error angles to obtain a new estimate of the DCM.

The iterative process consists of updating $\mathbf{C}_b^n$ until the vector of accelerometer measurements in the body frame consists of zero acceleration in the East and North directions and 1g along the Up direction. The transformed gyro measurements result in zero rate in the East direction and local earth rate components in the North-Up directions. That is, the iteration is repeated until the error angles are zero, or to within the specified tolerances [22]. The term 'error' is used to represent the difference between the true value of a variable as it exists physically and the apparent interpretation of that variable in the system computer. The disadvantage of using this method is that it can be a very lengthy process before the error angles converge to zero. Also, the behaviour of the control loop is sometimes unpredictable and difficult to stabilize.

A second approach that can be used is known as *Analytic Gyrocompassing and Leveling*. After an initial phase where the loops are opened, and consequently the

computational platform is not maintained locally level, the offset between the true and indicated computational frames is calculated. The computational platform is then leveled by closing the Schuler loops at the end of the initial alignment procedure. In a gimballed system, the misalignment matrix is physically driven to be the unit matrix by physically aligning the gyroscope input axes with the E, N, U geographic axes. In a strapdown system, the same result is achieved by damping out any residual initial errors. Note that in this stage the vehicle latitude and longitude are refined. If the vehicle is still stationary, any significant differences between initial lat/long and final lat/long indicate a base motion that is too excessive for proper alignment. Analytic gyrocompassing in a strapdown system is analogous to physical gyrocompassing of a gimballed system.

Analytic precession rates are applied to the gyroscope outputs to remove bias drift, earth rate and the angular rate of the computational frame relative to the earth. The remaining angular rate corresponds to the pure angular rate of the body relative to the earth. Bias is removed during inertial sensor data compensation and the remaining two system-level external precession rates are compensated for in the navigation processor. However, inaccuracies in the calculation of the applied analytic rate cause the computational platform to be overdriven past level. Bias instabilities are one source of error affecting the applied rates. Also, the calculation of earth rate components relies on an accurately known heading. Furthermore, an inexact heading initialization causes the accelerometer data to be transformed to a misaligned reference frame.

If additional sources of navigation data were available these could be used to prevent the growth of initial errors [6]. However, the HRU is an unaided AHRS and

consequently other means must be used to damp the heading and level errors.

By straying from a true Schuler-tuned mechanization, damping of initial errors can be achieved. In a Schuler-tuned system the accelerometer outputs are integrated and divided by the earth's radius to determine the command platform rate to correct for linear motions. The integrator can be replaced with a low pass filter (LPF) with gain 1/R, where R is not equal to the radius of the earth. The filter time constant and gain are adjusted to provide optimum damping.

There exists an optimum choice of parameters that will minimize errors due to gyro drift instabilities and those due to manoeuvre accelerations. If tilt errors are not damped, gravity components are sensed along the level axes and when integrated these give erroneous velocity readings. The damped platform alignment model for the North axis accelerometer is shown in Figure 5.3.

The second method is preferable primarily because it is not recursive in nature. From the analysis of the fine alignment error equations an error damping scheme is developed that does not require the vehicle to remain stationary. Unlike coarse alignment which is generally a rapid process, fine alignment can take hours. Thus, a fine alignment scheme that tolerates vehicle motion is advantageous because it does not require the user to wait for long periods of time before departure. Also, aligning aboard a ship at sea subjected to rolling and pitching motions is possible. The navigation solution is computed immediately after coarse alignment. Fine alignment and navigation mode occur concurrently whether or not the vehicle is in motion. This implies that the navigation solution gradually improves as the alignment procedure progresses.

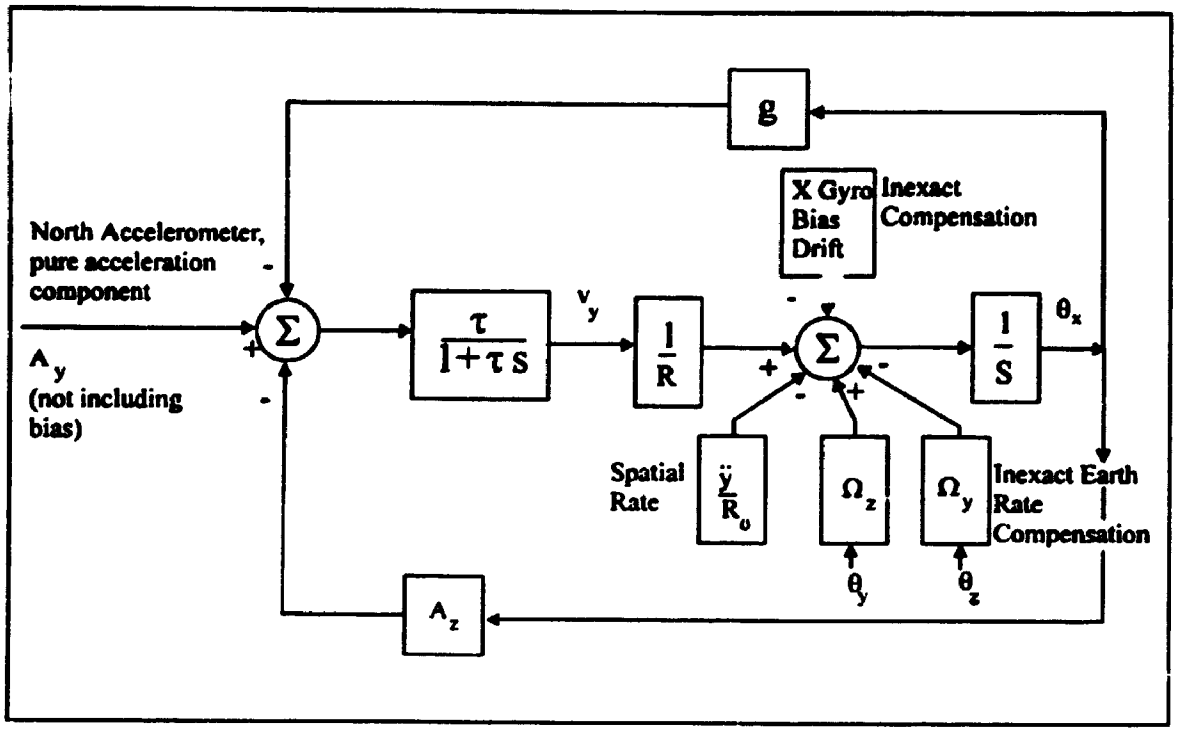


Figure 5.3: Damped Platform Alignment Model for the North-Axis Accelerometer. The factors affecting the rate of change of the East error angle are illustrated. A misalignment about the East axis results in a gravity component being sensed along the apparent North axis.

5.5.1 Differential Equations for Small Error Angles

A system of differential equations for the small error angles between the estimated geographic align matrix and the true align matrix is derived to demonstrate how damping can be injected to ensure that the errors are stable and do not grow with time. Timothy Ryan [32] uses a similar approach for determining the error angle rates, with a control scheme applicable only to a stationary vehicle. However, the concept is extended here to study the time response of the error angles due to bias instabilities, and misalignments in a moving vehicle.

Ideally, the computed DCM that transforms coordinates from body to geographic coordinates is the inverse of the true DCM that transforms geographic coordinates to body coordinates. However, component imperfections, calibration inaccuracies, quantization and truncation errors yield an estimated DCM that is offset from the actual DCM. Therefore, all subsequent references to the DCM computed during coarse alignment imply that it is an estimate , and all estimates are depicted with a '~' symbol.

When the estimated DCM and the transpose of the actual DCM are multiplied together the result is not the identity matrix. The resulting skewed identity matrix is referred to as the misalignment matrix **M**, equation 5.17.

$$\mathbf{M} = \tilde{\mathbf{C}}_b^g \cdot \mathbf{C}_g^b \tag{5.17}$$

The **M** matrix can also be expressed in terms of the error angles θ_x, θ_y, θ_z. That is, **M** can be defined by equation 5.18 where the error angles are written in the form of a skew-symmetric matrix, equation 5.19. The properties of skew-symmetric matrices are explained briefly here since they are used extensively in the derivation of the navigation equations also. A column matrix of elements along X, Y, Z axes can be written as a skew-symmetric square matrix. The defining property of a skew-symmetric matrix is that $\mathbf{A}^T = -\mathbf{A}$. Also, its main diagonal elements are all zero.

$$\mathbf{M} = \mathbf{I} - \boldsymbol{\theta} \tag{5.18}$$

$$\boldsymbol{\theta} = \begin{bmatrix} 0 & \theta_z & -\theta_y \\ -\theta_z & 0 & \theta_x \\ \theta_y & -\theta_x & 0 \end{bmatrix} \tag{5.19}$$

Thus, the misalignments are removed when the estimated coordinate system is rotated in

a counter-clockwise direction by the positively defined error angles. Conversely, when the transformed body coordinates are perfectly aligned with the actual computational coordinates, represented by the identity matrix, a rotation by the negative error angles yields the estimated coordinate frame.

The derivatives of the DCMs in expression 5.17 are given by equations 5.20 and 5.21, where $\mathbf{\Omega}_g$ is the angular rate of the geographic coordinates relative to inertial space in geographic coordinates and $\mathbf{\Omega}_b$ is the angular rate of the body relative to inertial space in body coordinates.

$$\dot{\tilde{\mathbf{C}}}_b^g = -\tilde{\mathbf{C}}_b^g \cdot \tilde{\mathbf{\Omega}}_b + \tilde{\mathbf{\Omega}}_g \cdot \tilde{\mathbf{C}}_b^g \tag{5.20}$$

$$\dot{\mathbf{C}}_g^b = -\mathbf{C}_g^b \cdot \mathbf{\Omega}_g + \mathbf{\Omega}_b \cdot \mathbf{C}_g^b \tag{5.21}$$

Using the first definition of $\mathbf{M}$ and the derivatives of the DCMs the derivative of the misalignment matrix can be found, refer to equation 5.22.

$$\dot{\mathbf{M}} = -\mathbf{M} \cdot \mathbf{\Omega}_g + \tilde{\mathbf{C}}_b^g \cdot (\mathbf{\Omega}_b - \tilde{\mathbf{\Omega}}_b) \cdot \mathbf{M} \cdot \tilde{\mathbf{C}}_g^b + \tilde{\mathbf{\Omega}}_g \cdot \mathbf{M} \tag{5.22}$$

The geographic and body angular rates relative to inertial space can be broken into two components each. The angular rate of the computational frame relative to inertial space consists of the angular rate of the computational frame relative to the earth and earth rate relative to inertial space, see equation 5.23.

$$\vec{\mathbf{\Omega}}_g = \vec{\rho}_{c/e}^g + \vec{\mathbf{\Omega}}_{i/e}^g \qquad \tilde{\vec{\mathbf{\Omega}}}_g = \tilde{\vec{\rho}}_{c/e}^g + \tilde{\vec{\mathbf{\Omega}}}_{i/e}^g \tag{5.23}$$

Earth rate relative to inertial space is sometimes referred to as the sidereal rate since it is the earth's rotation relative to the fixed stars and it has a value of 15.0417 deg/hr. Earth rate components sensed along the system axes are a function of latitude. Vehicle rate

compensation is needed to maintain the platform level as it travels along a curved path. The frame rate (or spatial rate) compensates for the earth's curvature. Thus, the gyro outputs transformed to the navigation coordinates include earth rate and vehicle rate compensation, refer to the illustration in Figure 5.4. The accelerometers can be used to calculate the frame rate using the LPF with gain equal to the reciprocal of R. The integrated accelerometer data with gain of $1/R_o$ is used to determine the earth rate components. Information acquired from the accelerometers are considered estimates.

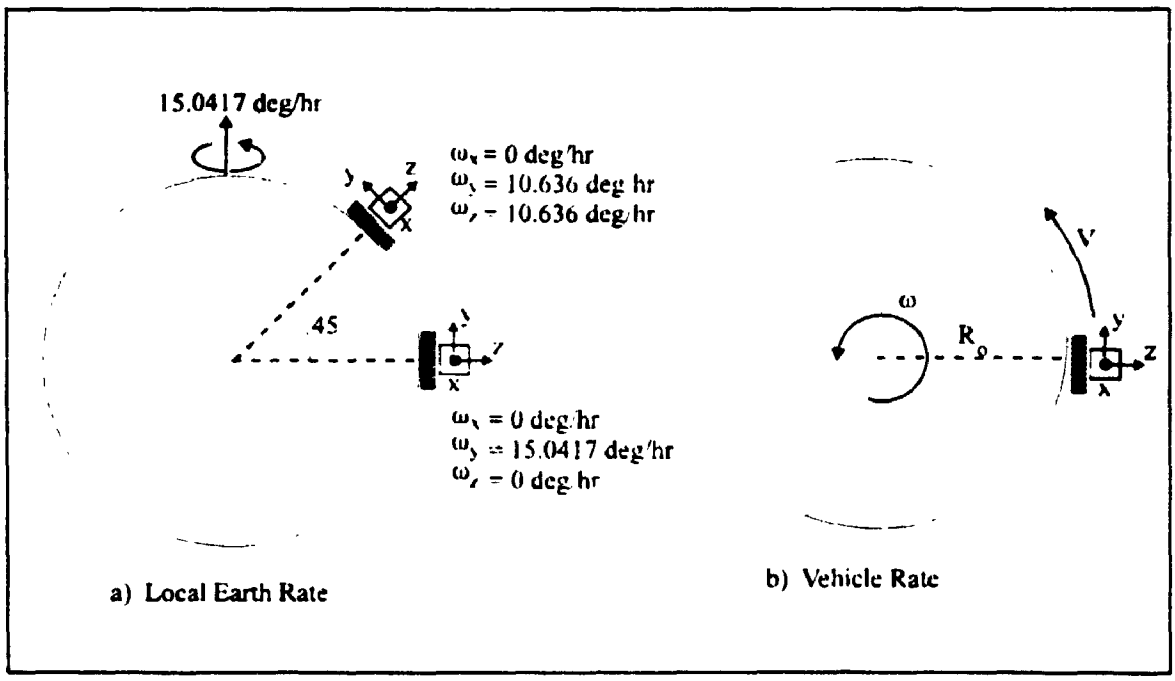


Figure 5.4: Earth Rate and Vehicle Motion Compensation. Compensation for the local earth rate is a function of the vehicle's latitude. Vehicle rate compensation is applied to keep the platform level as it travels along a curved path.

The angular rate of the body relative to inertial space is obtained by subtracting bias drift from the raw gyro outputs. If the correct bias is compensated for then the true Ω_b is found. However, in practice due to the uncertainties associated with the bias drift an

estimate of Ω_b is used for compensation, refer to equation 5.24.

$$\tilde{\Omega}_b = \tilde{\omega}_r - \bar{G}\bar{B} \qquad\qquad \tilde{\tilde{\Omega}}_b = \tilde{\omega}_r - G\tilde{\tilde{B}} \tag{5.24}$$

Equations 5.23 and 5.24 are substituted in 5.22. The resulting equation is then simplified using the similarity transformation, explained in Appendix A, on the middle term. The difference between the actual bias drift and the compensated bias drift is re-written as ΔB_1, a bias instability. The simplified equation for the derivative of **M** is given in equation 5.25.

$$\dot{\mathbf{M}} = -\mathbf{M}\cdot(\rho^{g}_{c/e} + \Omega^{g}_{i/e}) + (\Delta \mathbf{B}_1)\cdot\mathbf{M} + (\tilde{\rho}^{g}_{c/e} + \tilde{\Omega}^{g}_{i/e})\cdot\mathbf{M} \tag{5.25}$$

Typical inertial navigation systems experience gyro drift rate instabilities of approximately 0.01 deg/hr. When compared with other error terms such as scale factor errors around 50 parts per million (ppm) and the nonorthogonality between gyro axes which is generally a small quantity, the gyro drift term is the dominant error term [19]. Specifically, for the HRU CSG-2's the nonorthogonality between gyro axes is less than 7 mrad.

The rates of change of the error angles can be used to predict if the error angles will grow or if they will damp to small steady-state values. Equation 5.18 is substituted in equation 5.25 to find the derivatives of the error angles as shown in equation 5.26. Terms that have been cancelled include: the bias instability and coordinate frame rates multiplied by the small angle misalignments. These were neglected since they involve the multiplication of two small quantities. The variable ΔB_2, includes the difference between actual and estimated bias, as well as the difference between earth rate as sensed by the gyroscopes and that calculated using the accelerometers.

$$\dot{\boldsymbol{\theta}} = \boldsymbol{\rho}^{g}_{c/e} - \boldsymbol{\theta} \cdot \boldsymbol{\Omega}^{g}_{i/e} + \boldsymbol{\Omega}^{g}_{i/e} \cdot \boldsymbol{\theta} - \Delta \mathbf{B}_2 - \tilde{\boldsymbol{\rho}}^{g}_{c/e} \tag{5.26}$$

The general set of linear coupled differential equations for the error angles are given in equation 5.27 with the platform rate correction substitutions made. A wander-azimuth platform is not kept north-pointing, thus, the heading error depends on how accurately the wander angle rate is known.

$$\begin{aligned}
\dot{\theta}_x &= -\frac{V_y}{R_o} - \Omega_y \cdot \theta_z + \Omega_z \cdot \theta_y - \Delta \mathrm{B}_x + \frac{v_y}{R} \\
\dot{\theta}_y &= \frac{V_x}{R_o} - \Omega_z \cdot \theta_x - \Delta B_y - \frac{v_x}{\mathrm{R}} \\
\dot{\theta}_z &= \frac{(\cos(\alpha) \cdot V_x - \sin(\alpha) \cdot V_y)}{R_o} \cdot \tan(\phi_L) + \Omega_y \cdot \theta_x - \Delta B_z + \dot{\alpha}
\end{aligned} \tag{5.27}$$

Controlled precession of the mathematical platform is used to keep it locally level. The platform heading is allowed to diverge away from North, but the calculated system heading must be stable. An in-depth analysis of the steady-state and transient behaviour of the error angles is needed to determine the command control parameters, as will be completed in the following section.

5.5.2 Level Errors

The roll and pitch error angles are first considered independently from the heading error angle. Hence, the simplified set of coupled differential equations are listed in 5.28, below.

$$\begin{aligned}
\dot{\theta}_x &= -\frac{V_y}{R_o} + \Omega_z \cdot \theta_y - \Delta B_x + \frac{v_y}{R} \\
\dot{\theta}_y &= \frac{V_x}{R_o} - \Omega_z \cdot \theta_x - \Delta B_y - \frac{v_x}{R}
\end{aligned} \tag{5.28}$$

For navigation purposes, the acceleration of interest is the pure acceleration of the vehicle, namely, the acceleration in computational coordinates relative to the earth. The accelerometer output consists of inertial acceleration measurements from which the relative acceleration can be directly derived. The accelerometer output includes, pure vehicle, coriolis, centripetal and gravitational accelerations plus an offset bias. Coriolis, and gravity compensations are applied to the transformed accelerometer outputs. The coriolis misalignment terms cancel in the analysis. The resultant estimated relative acceleration consists of the actual relative acceleration plus a bias instability component multiplied by the misalignment matrix, a gravity component multiplied by **M**, minus the gravity vector model correction.

All bias instabilities become negligible since they are transformed by the small angle misalignments. However, initial tilt errors result in the gravitational acceleration components being sensed along the level axes. In an ideal, perfectly aligned system, zero g components are sensed along the East and North axes. The effect of a misalignment angle on the gravity vector is illustrated for an East axis error in Figure 5.5. Consequently, under static conditions a level error translates the accelerometer readings causing false apparent accelerations along the axes perpendicular to the vertical.

The level pure acceleration equations are given in equation 5.29, where V represents the estimated pure acceleration and A the true pure acceleration terms, not to be confused with the accelerometer outputs.

$$\begin{aligned} \dot{V}_x &= A_x + A_z \cdot \theta_y + g \cdot \theta_y \\ \dot{V}_y &= A_y - A_z \cdot \theta_x - g \cdot \theta_x \end{aligned} \tag{5.29}$$

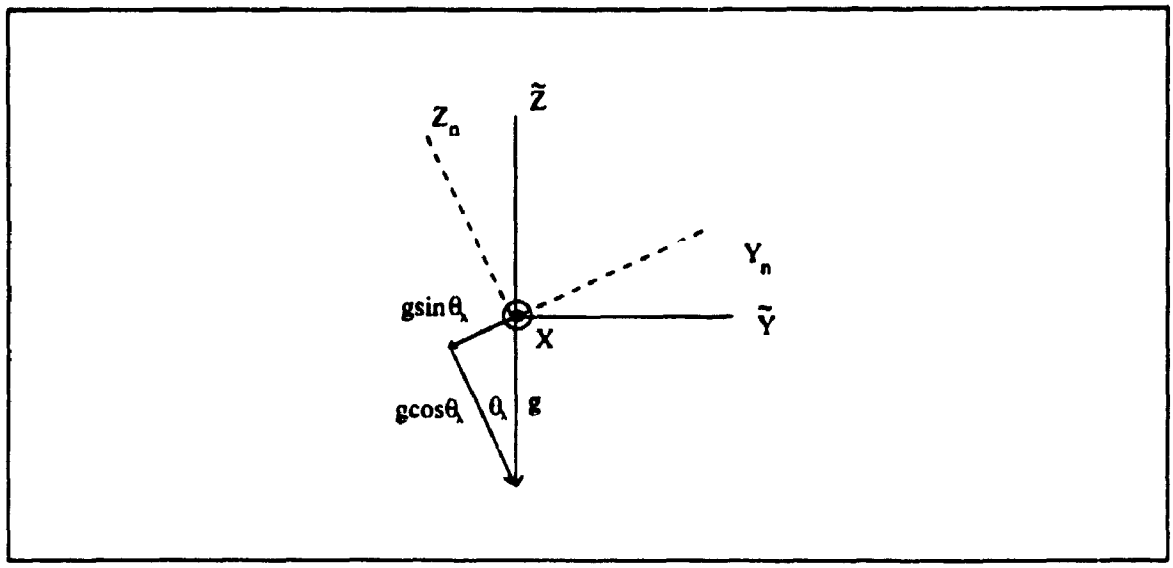


Figure 5.5: East Axis Error Angle. A misalignment about the East axis results in inexact gravity components being sensed along the apparent North and Up axes.

The filtered accelerations yield a velocity component used as a command rate control. This method is similar to an analog filter implementation described in a report by Staley for vertical control loop stability in a navigation system [31]. The output filtered velocities are obtained according to 5.30

$$\dot{v}_x + \frac{1}{\tau} \cdot v_x = \dot{V}_x \qquad\qquad \dot{v}_y + \frac{1}{\tau} \cdot v_y = \dot{V}_y \tag{5.30}$$

The time constant of a system is defined as the time required for the output to reach 63% of its final or steady-state value, in response to a step input. The filter time constant, τ, is different from the fine alignment leveling time constant. In fact, the filter time constant is on the order of minutes, whereas platform leveling and damping of the initial errors may take hours. For values of τ approaching infinity the LPF is equivalent to an integrator realization. Thus, the limits on acceptable minimum and maximum values for τ are considered in §5.5.2.5.

The East and North errors are combined to form one error term, the level error. Consider, complex variable θ with real component θ_x and imaginary component θ_y. The derivative of the complex variable is found by substituting the complex representation in the level error equations 5.28.

Similarly, the filtered acceleration is defined as a complex variable ν with real component ν_x and imaginary component ν_y. The filtered acceleration can then be re-written as complex variables in terms of the relative accelerations, misalignment angles and a time constant using equations 5.29, 5.30. The filtered acceleration is then used to obtain the second-order non-homogeneous differential equation for the level errors, see equations 5.31 and 5.32, after substitution in equation 5.28.

$$\ddot{\theta} + \dot{\theta}\cdot(\frac{1}{\tau} + i\Omega_z) + \theta(\frac{i\Omega_z}{\tau} + \frac{A_z}{R} + \frac{g}{R}) = RHS \tag{5.31}$$

$$RHS = -\frac{i}{R}\cdot(A_x + iA_y) - \frac{\dot{V}_y}{R_o} + i\frac{\dot{V}_x}{R_o} + \frac{1}{\tau}\cdot(\frac{-V_y}{R_o} - \Delta B_x - i\Delta B_y + i\frac{V_x}{R_o}) \tag{5.32}$$

5.5.2.1 SS Level Errors for Constant Velocity Case

The steady-state error can be found by applying the Final Value Theorem of the time function to the level error expression 5.31, that is, evaluating it in the limit as 't' approaches infinity. A stable time response is characterized by a constant final value [36]. The steady-state response is evaluated for different test cases to test for stability and to find the relationship between adjustable parameters and the long-term residue.

The first case study involves the long-term effect of the vehicle moving at a constant velocity. The steady-state error is derived by setting the first and second order time-

derivatives of the error angle in equations 5.31 and 5.32 to zero. The effect of a constant velocity is then found by setting the bias instabilities to zero and neglecting acceleration terms. The East and North level errors are solved for independently, then combined as a single root mean square (rms) level error θ. Similarly, the velocity V refers to the rms velocity of the East and North components. The steady-state level error under a constant velocity is given by equation 5.33.

$$|\theta_{ssl}| = \left(\frac{V}{R_o \cdot \tau}\right) \cdot \frac{1}{\sqrt{\frac{g^2}{R^2} + \frac{\Omega_e^2}{\tau^2}}} \tag{5.33}$$

The resultant error as a function of R and τ, assuming a cruise velocity of 20 kph, or 10.8 knots, is plotted in Figure 5.6.

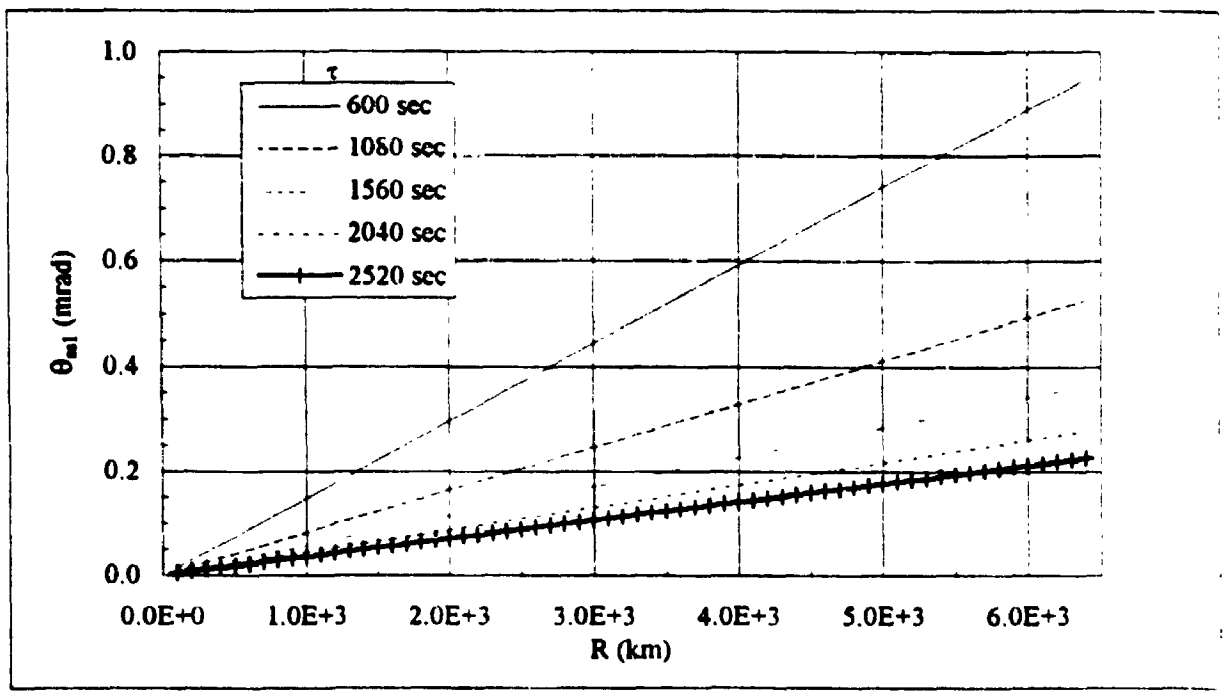
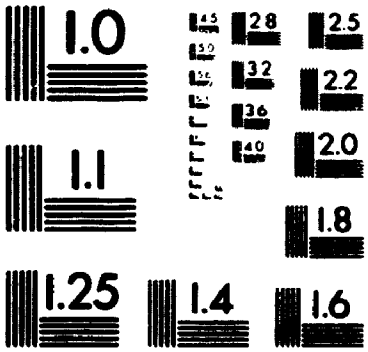


Figure 5.6: Steady-State Level Error for Constant Velocity Case. The LPF gain and time constant can be selected to minimize the magnitude of the steady-state level error when the vehicle is travelling at a cruise velocity of 20 kph.

2

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT



PRECISION℠ RESOLUTION TARGETS

The Navlab latitude of 45.349 degrees N is chosen as the vehicle's position. The constant velocity steady-state error grows as R approaches the radius of the earth, which is approximately 6378.14 km. The benefit of choosing a larger value for τ is that it reduces the final steady-state error.

For zero initial conditions, the magnitude of the level error is in the range of 0.2 to 0.9 milliradians for $R=R_o$. The slope of the error response increases by approximately a factor of 4 for a 34 minute increase in the filter time constant

5.5.2.2 SS Level Errors for Gyro Bias Instability Case

Similarly, the level error response in the steady-state can be determined for the case where the true bias and the compensated bias are unequal Under the influence of bias instabilities only, the rms level steady-state error equation is given by equation 5 34. The first and second order time-derivatives of the error angle in equations 5.31 and 5.32 are set to zero. In contrast to the previous case, conditions at the other extreme are observed by considering the effect of the bias terms and neglecting the velocity components in equation 5 32

$$|\theta_{ss2}| = \left(\frac{\Delta B}{\tau}\right) \cdot \sqrt{\frac{2}{\frac{g^2}{R^2} + \frac{\Omega_e^2}{\tau^2}}} \tag{5 34}$$

Even under ideal circumstances where the turn-on bias drifts for the level gyros are exactly equal to the bias drifts used in the data compensation algorithm, the bias drift will wander in response to temperature gradients, or disturbances within the sensor. This

uncertainty is referred to as the gyro random drift characteristic.

Assuming bias instabilities of 0.01 deg/hr for both the East and North gyro axes, equation 5.31 is converted to a system of four first-order linear differential equations and solved using a fifth order Runge-Kutta (RK-5) integration routine. For zero initial conditions, Figures 5.7 and 5.8 illustrate how the appropriate choice of gain and time constant, results in stable, damped error angle responses. The responses damp faster for τ=1080 sec than for τ=2040 sec but this is at the expense of a larger steady-state end result.

The effect of choosing a larger R is a right phase shift of the sinusoidal response, thereby generating larger steady-state values. Thus, the steady-state response is better for larger time constants and smaller values of R. The response parallels the constant velocity case and is used in the tradeoff analysis to find the optimum R and τ values in §5.5.2.5.

The effect of bias instabilities on the level error responses is also considered for the case where initial error angle offsets exist. Figures 5.9 and 5.10 depict the time responses for initial misalignment angles of 0.1 deg. In addition to the decaying oscillatory response seen in the previous test case, the final value approaches zero despite the initial errors. Initially, the amplitude of oscillation of the error response is equal to the misalignment error angle. In an undamped Schuler-tuned system the error response would oscillate at the Schuler period without decaying. By choosing a value of R not equal to the earth's radius, the period of oscillation of the error response is reduced and the time constant of the LPF injects damping in the system.

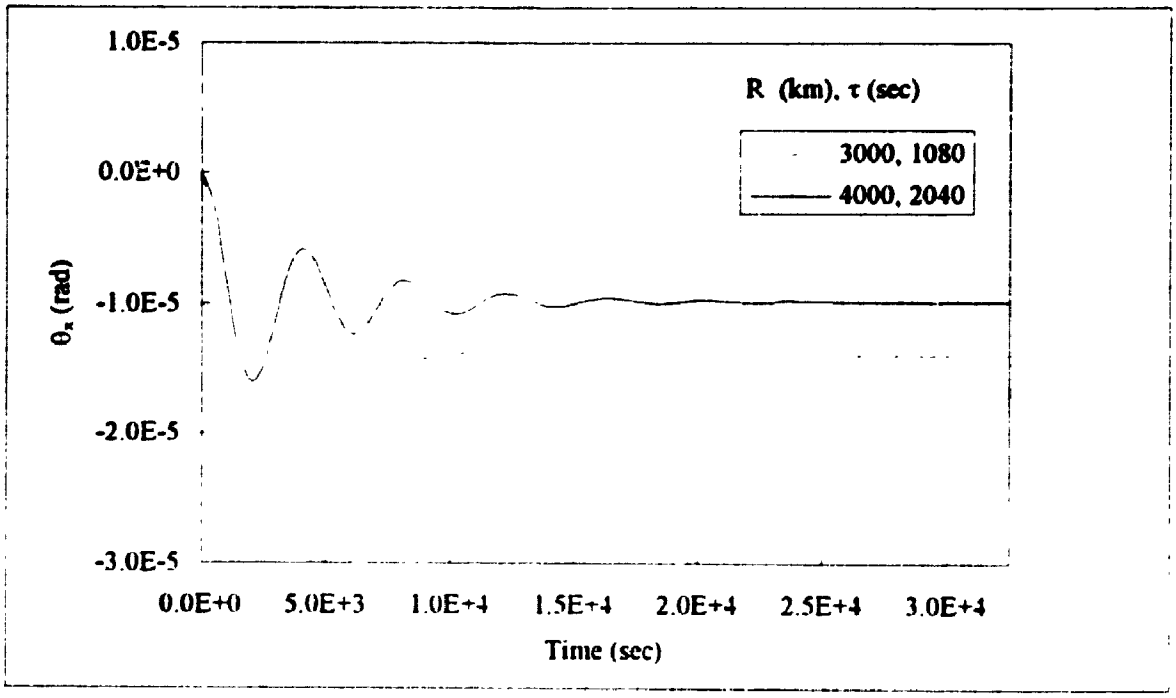


Figure 5.7: East Error Angle Damped Response for Bias Instability Input (0.01deg/hr). The initial condition misalignments are assumed to be zero.

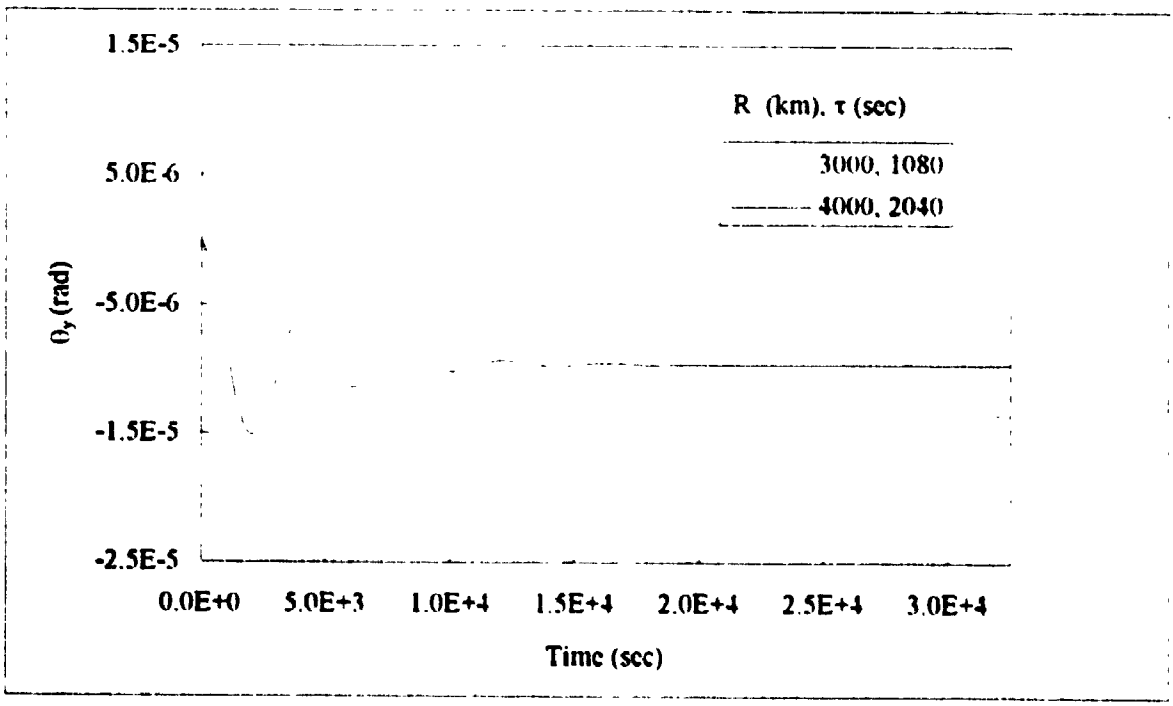


Figure 5.8: North Error Angle Damped Response for Bias Instability (0.01 deg/hr). The initial condition misalignments are assumed to be zero.

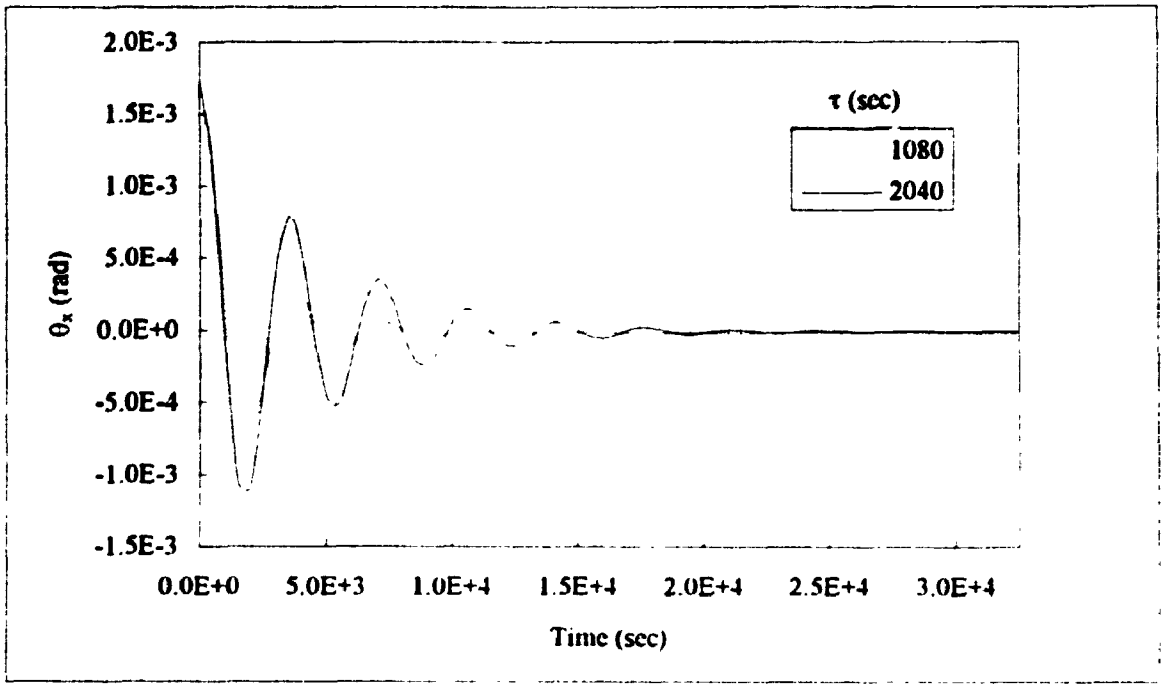


Figure 5.9: East Error Angle Damped Response. The test conditions are:
Initial Error 0.1 degree, Bias Instability 0.01 deg/hr and R=3000 km

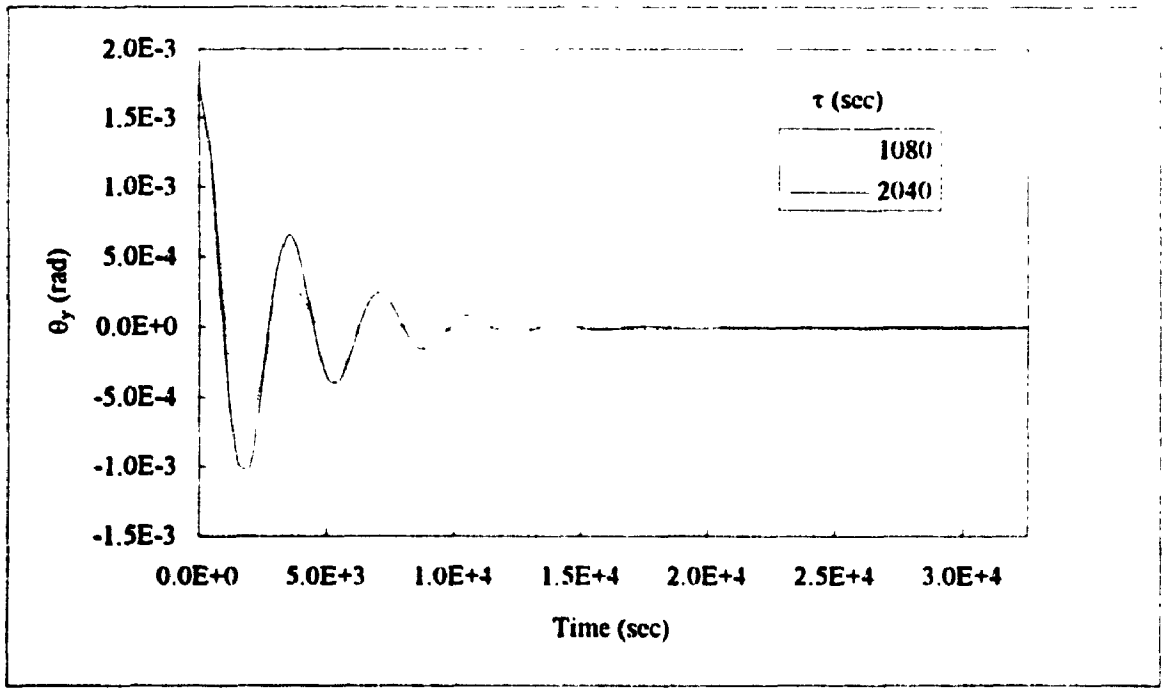


Figure 5.10: North Error Angle Damped Response. The test conditions are:
Initial Error 0.1 degree, Bias Instability 0.01 deg/hr and R= 3000 km.

5.5.2.3 Transient Response to a Constant Acceleration

It is also important to characterize the resulting error angle at the end of a transient constant acceleration. This case study depicts the situation where the vehicle accelerates from a stationary state to 20 kph in 30 sec. For this analysis, both the velocity and the acceleration for travel along a straight path are assumed to act entirely along the easterly direction. Equation 5.32 for the level error angles is modified accordingly and the new system of four linear first-order differential equations is solved using a RK-5 numerical integration routine.

The purpose of this exercise is to determine the level error at the end of the acceleration phase and view how it relates to the τ and R parameters. The rms level error that combines the East and North errors is plotted in Figure 5.11, as a function of τ and R. The magnitude of the error angle increases as the difference between R and the exact value for the radius of the earth becomes larger. Similarly, for larger values of τ, the rms level error augments

By the same token, inaccuracies in the earth model used in the navigation algorithms cause misalignment errors. In effect, these represent offsets between the true earth radius at a given position and the estimated value in the computer.

The results presented in Figure 5.11 indicate that conditions which are favourable for minimizing level errors due to a constant manoeuvre acceleration are different from those required to minimize the steady-state level errors discussed in §5.5.2.1 and §5.5.2.2.

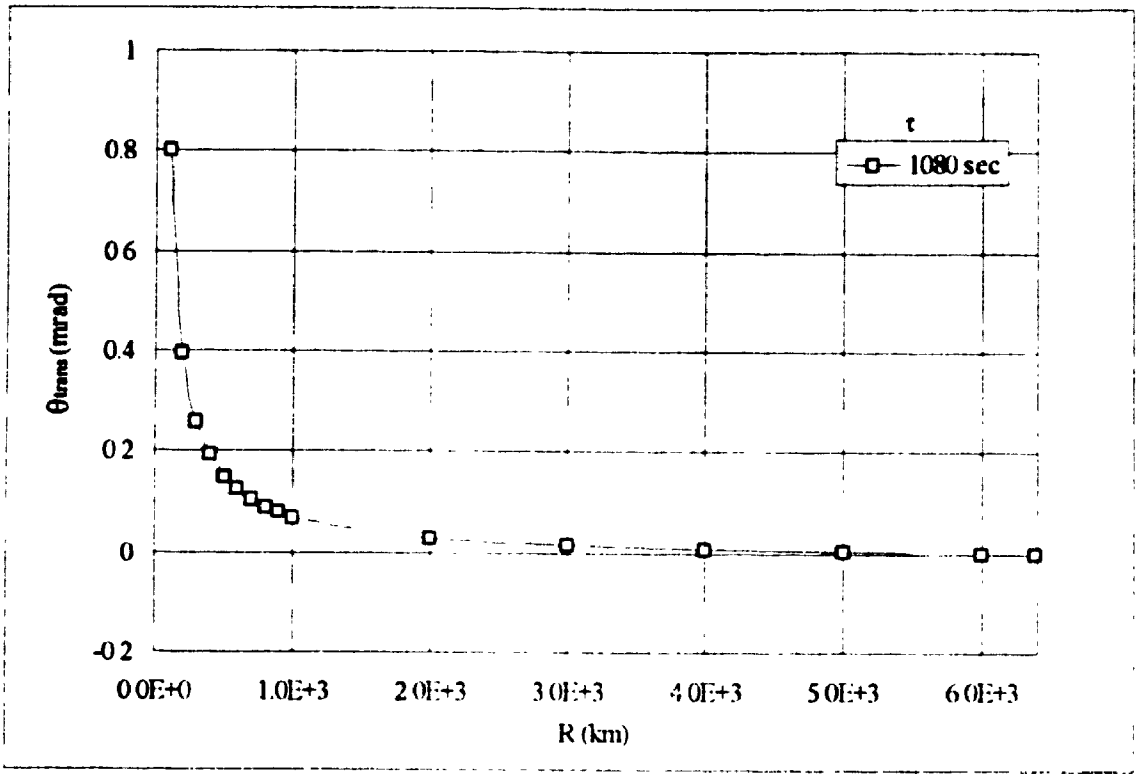


Figure 5.11: Level Error after 30 second Constant Acceleration (τ=1080 sec). Following the vehicle's acceleration from a state of rest to a speed of 20 kph in 30 seconds, the magnitude of the level error angle decreases for values of R approaching the earth's radius (R=6378.14 km).

5.5.2.4 Transient Response to a Manoeuvre Acceleration

The manoeuvre acceleration considered is based on the performance of a ship as it turns. The case study manoeuvre is a 90 degree turn executed in 30 seconds at a speed of 30 knots. If the vehicle and strapdown HRU travel at a constant velocity while executing a turn, a corresponding centripetal acceleration is sensed by the accelerometers. Again, the representative set of ordinary D.E.'s is solved. If the acceleration is concentrated along the East axis, the East error angle response is as shown in Figure 5.12.

At the manoeuvre's completion, the effect of the damping is to increase the magnitude of the error as R increases. In contrast, the orthogonal axis along which the

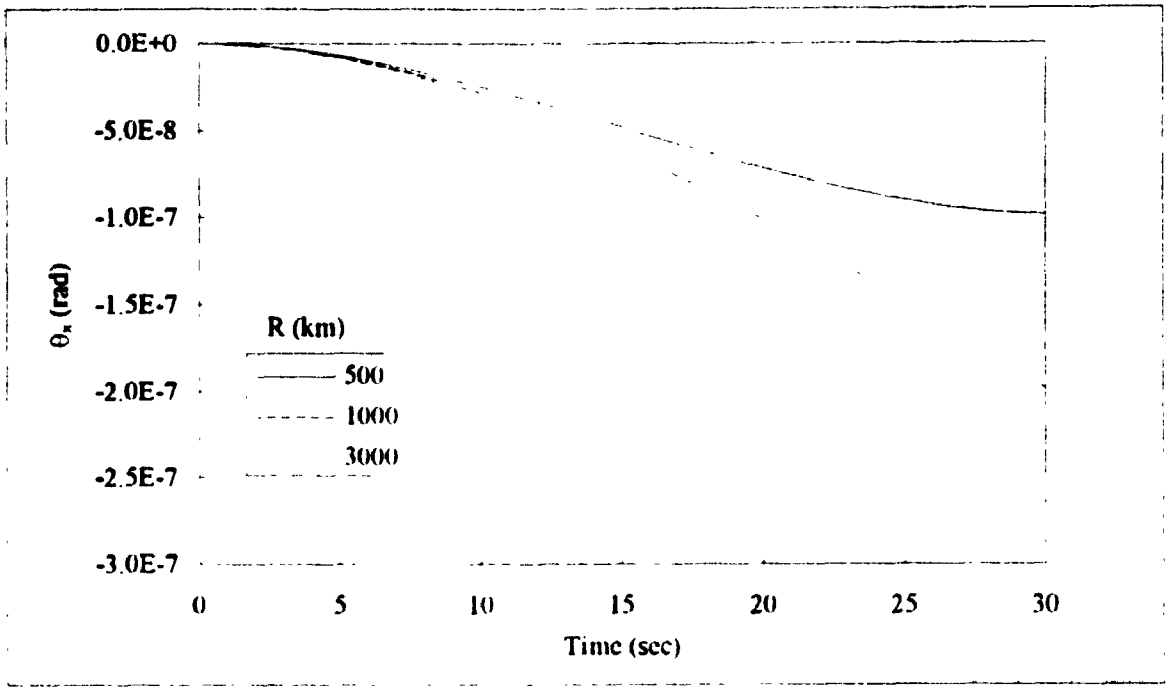


Figure 5.12: East Error Angle for 30 second Easterly Manoeuvre Acceleration. The manoeuvre corresponds to a ship executing a 90 degree turn at 30 knots.

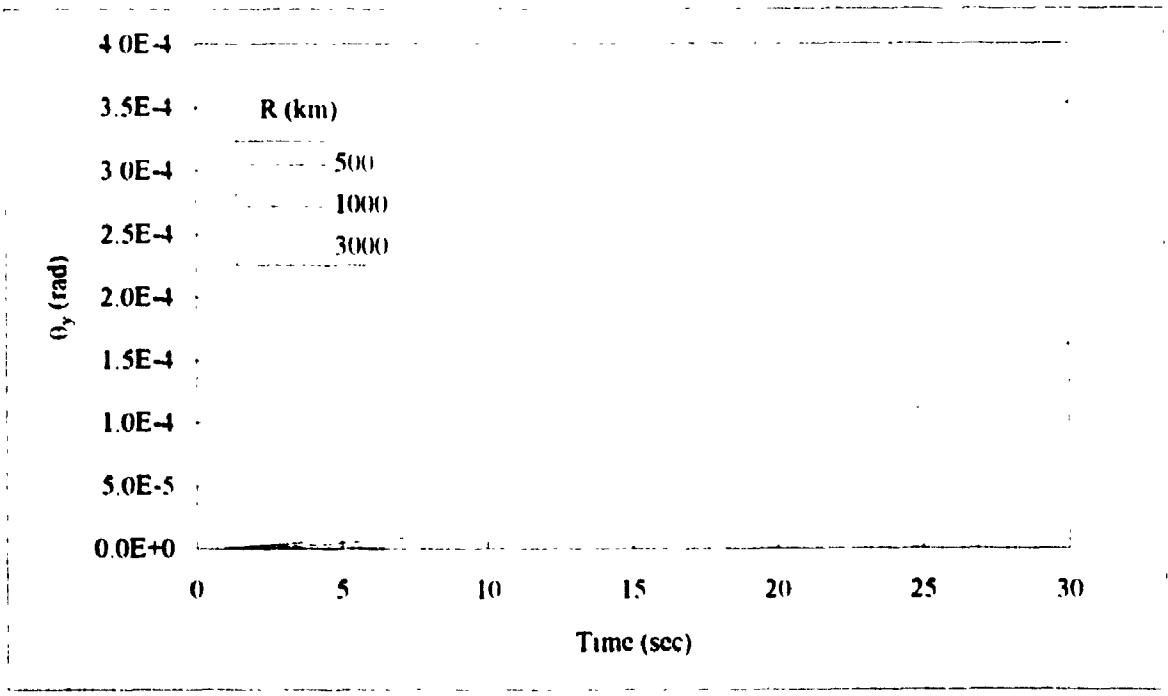


Figure 5.13: North Error Angle for 30 second Easterly Manoeuvre Acceleration. The manoeuvre corresponds to a ship executing a 90 degree turn at 30 knots.

constant velocity is sensed, exhibits a decreasing error angle for larger values of R, see Figure 5.13. The latter error is the dominant term since by comparison, the East error angle is roughly three orders of magnitude smaller than the North error angle for the R=1000 km series. Consequently, at the end of the manoeuvre, the residual level error is smaller for values of R approaching the earth's radius.

5.5.2.5 Optimum Parameter Selection for Level Error Damping

The previous results indicate that the choice of R and τ relies on a compromise between minimizing transient and steady-state level errors. The steady-state level errors due to gyro bias instabilities or a constant velocity exhibit similar trends in relation to the two parameters. Similarly, the transient error responses in the presence of a constant or manoeuvre acceleration are comparable. Therefore, one error of each type, namely, gyro bias instability and manoeuvre acceleration, is observed in the tradeoff plot in Figure 5.14.

Minimizing one error at the expense of the other is not acceptable, consequently concessions must be made to achieve optimum results. The parameter values selected are R=3000 km and τ=1080 sec. The steady-state errors due to bias instabilities of 0.01 deg/hr are in the tens of μrad range. In contrast, the magnitudes of the transient level errors are more significant, they are in the mrad range.

Values for τ between 10 and 42 minutes are considered in the tradeoff study. Results from several simulated test cases discussed in §7.3.4.1 indicate that increasing the time constant further significantly augments the time required to damp initial

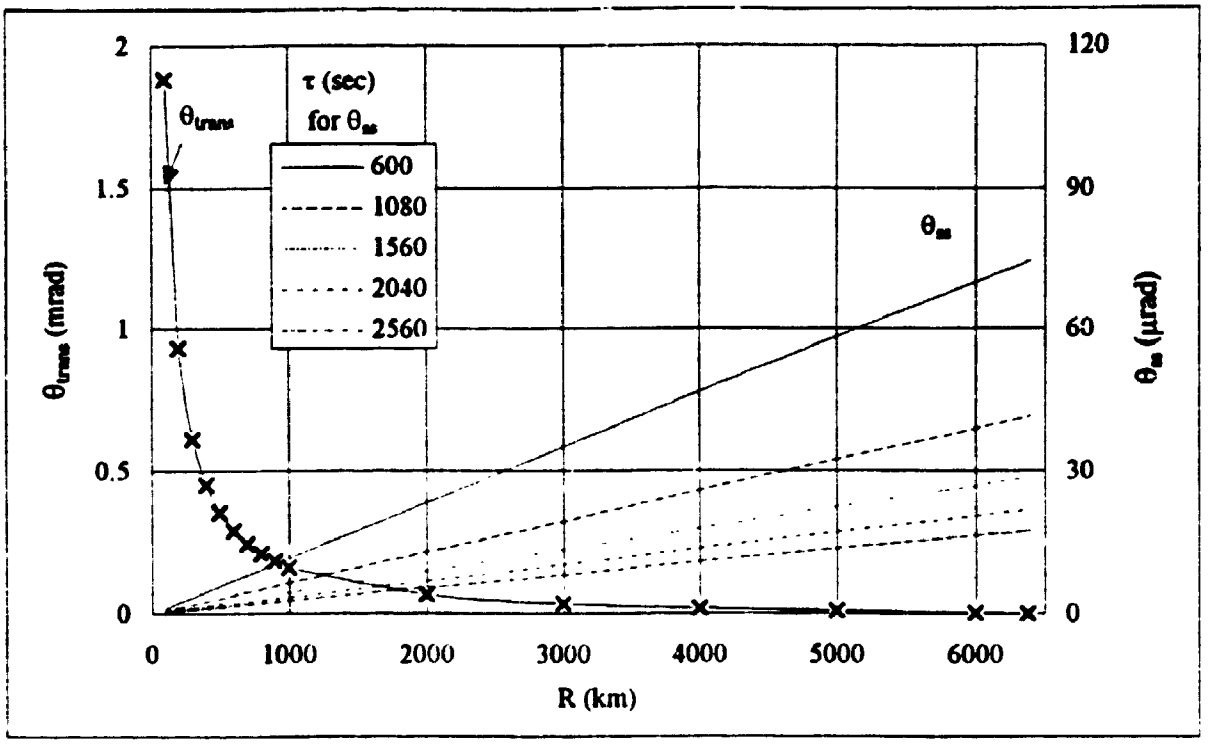


Figure 5.14: Transient and Steady-State Level Error Tradeoff. Optimum LPF filter parameters are selected to minimize the two types of errors.

condition errors. Furthermore, increasing the time constant by a factor of ten, yields a bias instability steady-state error on the order of 4.13 μrad at $R=R_o$ and $\tau=3$ hours. However, the results presented in Figure 5.14 indicate that the errors are sufficiently small for τ in the range of minutes. Consequently, it is preferable to increase the rate of decay of the error response. Hence, the maximum value of τ should be in the range of 1 hour. Also, as τ increases the magnitude of the transient errors increases.

Furthermore, it is observed that for each successive reduction in the time constant by equal time step intervals, the difference in magnitude between successive steady-state errors increases. For example, the steady-state errors at $R=R_o$, for $\tau=1560$, 1080 and 600 seconds are 74, 41 and 29 μrad, respectively. Therefore the minimum value of τ

should be in the range of 5 to 10 minutes to prevent rapid growth of the steady-state error.

5.5.3 Heading Errors

For stability purposes, heading errors must also be damped. For a local-level wander-azimuth mechanization the platform heading diverges from true North by an angle referred to as the wander angle. Unlike the level axes a precession rate is not applied to the azimuth channel to correct for the vehicle's motion over the spherical earth. The platform heading and the wander angle are both used to calculate the system heading. To ensure that heading errors are stable, a rate correction is applied to the azimuth channel to correct for an excessive platform divergence rate, inaccurate earth rate and bias drift compensation.

Bias drift instability errors translate directly to system initial attitude and heading errors. A bias drift represents an angular change over time of the transformation matrix, and thus any error in the bias drift correction causes a misalignment between the desired platform and the computational platform. For example, the consequence of having a bias instability about the Up system axis, is a heading error. This concept is illustrated in Figure 5.15.

To simplify the heading error analysis, bias instabilities are replaced with equivalent initial condition errors. The actual and effective platform heading divergence rates cancel. A rate correction is applied to the azimuth channel to prevent growth in the initial heading error. This damping term is proportional to the acceleration along the orthogonal level axis. The simplified coupled error equations with damping for the

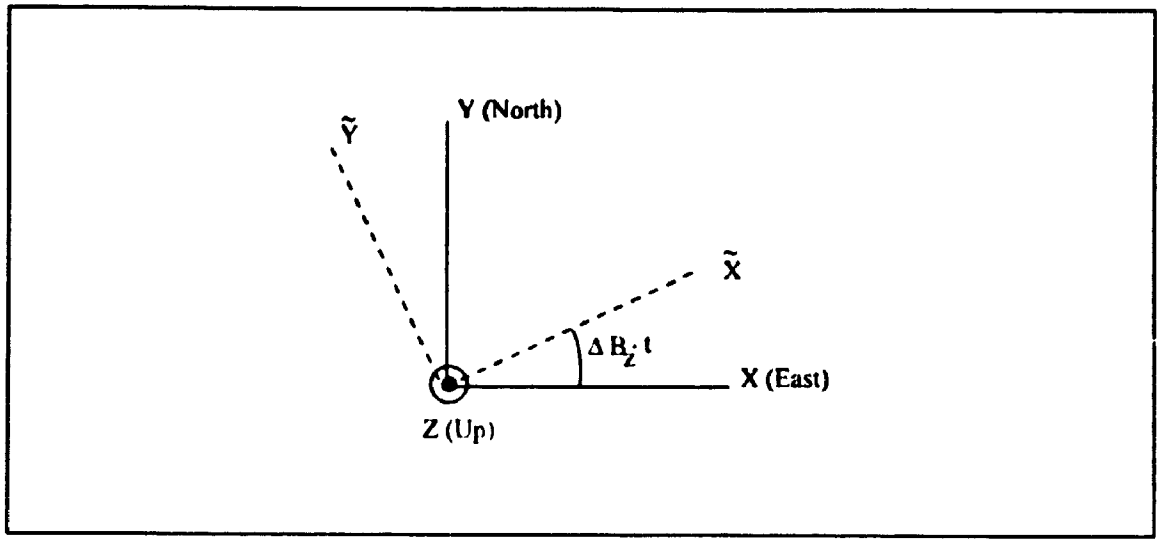


Figure 5.15: Bias Drift Instability Causing System Heading Misalignment.

azimuth and East channels are given in equation 5.35. This implies that the azimuth error is dependent on the day-to-day bias repeatability of the East gyro.

$$\begin{aligned} \dot{\theta}_x &= -\frac{\dot{V}_y}{R_o} - \Omega_y \cdot \theta_z + \frac{v_y}{R} \\ \dot{\theta}_z &= -\Omega_y \cdot \theta_x - a \cdot \dot{V}_y \end{aligned} \tag{5.35}$$

The next task consists of determining the optimum damping coefficient that will ensure stability of the linear system in response to an appropriate stimulus, in this case initial condition errors. Towards this end, the Laplace transform of the above coupled vertical and level differential equations and the expressions for $\dot{V}_y$ (5.29) and $\dot{v}_y$ (5.30), is taken. The system's characteristic equation is found and has been determined to satisfy the necessary condition for stability that all of its roots have negative real parts. Based on the characteristic equation 5.36, the first requirement on 'a' comes from setting all coefficients positive.

$$s^3 + s^2 \cdot \left(\frac{1}{\tau}\right) + s \cdot \left(\frac{-g}{R_o} + \frac{g}{R} + \Omega_y^2 + a \cdot g \cdot \Omega_y\right) + \frac{1}{\tau} \cdot \left(\frac{-g}{R_o} + \Omega_y^2 + a \cdot g \cdot \Omega_y\right) = f(s) \quad (5.36)$$

The above condition is not sufficient for ensuring system stability. As an additional measure for guaranteeing stability the Routh-Hurwitz Stability Criterion is applied [36]. By computing the Routh array for the polynomial the previously stated condition is repeated and all the other cofactors are in accordance with a system that does not have any right-half plane (RHP) roots. The threshold value for 'a' is 0.003 sec/m below which the system becomes unstable for a 45 degree latitude position. The threshold value does not include a margin of error, on that account the optimum value for 'a' is selected experimentally in Chapter 7. By deliberately injecting initial condition heading misalignments in the navigation software the damping of the heading error can be observed as a function of the damping parameters. The navigation software itself serves as an excellent tool for modelling the misalignment angles since the axes cross-coupling is inherent in its mechanization.

Chapter 6

Theoretical Formulation of Navigation Algorithms

6.1 Introduction

An East(X), North(Y), Up(Z) Local-Level Wander-Azimuth mechanization is used to derive the navigation equations for the Heading Reference Unit Measurements of specific force and angular velocity relative to inertial space over known time intervals are used to determine the instantaneous vehicle position, velocity and attitude information The governing navigation differential equations are to be integrated in real-time and matrix algebra properties are used to resolve the parameters of motion of the vehicle at the required time intervals

Accelerations sensed by the accelerometers are used to calculate position information and the angular rates provided by the gyroscopes are used to maintain control of the coordinate reference frame. The reference frame rotates to compensate for earth rate and vehicle motion over the earth, thereby keeping the frame locally level. However, the level axes rotate freely about the vertical axis, that is, they are not held in a

specific azimuth direction. This is because only the vertical component of earth rate is compensated and not the vertical component resulting from the vehicle's motion over the earth. The advantages of using a wander-azimuth system mechanization are relatively simple coordinate transformations and no singularities at the poles. Hence, true worldwide navigation is possible.

In this chapter, the damped wander-azimuth navigation equations are developed for a self-contained INS implementation. The navigation outputs found explicitly are: latitude, longitude, wander angle, ground velocity, roll, pitch, and heading.

6.2 Wander-Azimuth Mechanization, Updating the Quaternion

An integral part of the navigation mechanization is the transformation matrix that relates the system reference frame to the computational frame, namely the local-level wander-azimuth frame. The transformation can be defined in terms of three ordered angular rotations. The disadvantage of this approach is the mechanization is singular for any angle equal to ±90 degrees. Furthermore, a sequence of three matrix multiplications is required to execute the transformation.

Alternatively, the sequence of Euler rotations can be written as a single direction cosine matrix. The drawback of the latter approach is that nine linear differential equations, one for each element in the matrix, must be solved to update the DCM. Depending on the application for which the DCM is intended the number of linear differential equations to be solved can be reduced to six. For instance, this is the case if

the DCM is used uniquely to solve for variables and not to transform a vector to a different coordinate system.

Thirdly, the transformation matrix can be defined in terms of four quaternion parameters. Three parameters characterize the axis of orientation and the fourth signifies the angle of rotation. Quaternions are commonly used in aerospace applications since the four-parameter representation is the minimum redundancy method for representing a coordinate transformation matrix. Also, the quaternion representation is the preferred method since it is not singular and requires only four linear differential equations be solved. Quaternions are explained in more detail in the text by Siouris [3].

In a strapdown system the quaternion parameters are updated using a skew-symmetric matrix μ_c. The derivatives of the quaternions are equal to the angular velocity of the transformation matrix multiplied by the quaternion parameters. The angular velocity is given by the $\vec{\mu}_c$ vector, it includes the angular rates relative to inertial space measured by the gyroscopes, and the rate of the local-level frame relative to inertial space [26]. The angular velocity vector $\vec{\mu}_c$ is expressed in computational coordinates, as denoted by the subscript. The 3×1 vector $\vec{\mu}_c$ is calculated as shown in equation 6.1.

$$\vec{\mu}_c = \mathbf{Q} \cdot \vec{\omega} - \left(\vec{\rho}_c + \vec{\Omega}_c\right) \tag{6.1}$$

Thus, $\vec{\mu}_c$ is the true angular velocity of the vehicle relative to the local-level frame of reference. The term $(\vec{\rho}_c + \vec{\Omega}_c)$ represents the angular rate of the computational frame relative to inertial space in local-level wander-azimuth coordinates. Specifically, it is the summation of the angular rate of the computational frame relative to the earth, and the

rotation rate of the earth relative to inertial space, respectively. The angular rate of the vehicle measured by the gyroscopes in body coordinates and calibrated for constant error terms is represented by the $\vec{\omega}$ vector. The gyroscope outputs correspond to instantaneous measurements and not incremental changes over a pre-defined period of time. They are transformed to local-level coordinates by multiplying it by the transformation matrix **Q**. As stated previously, **Q** is the four-parameter transformation matrix that converts data in system coordinates to computational coordinates.

The rate corrections applied to the gyro outputs are a post-processed means of precessing the platform in a strapdown system that is analogous to torquing the platform in a gimballed mechanization. Precessional rates are applied to remove earth rate and to compensate for vehicle motion over the earth. A system is said to be Schuler-tuned when the latter compensation is implemented. This arises from the fact that accelerometers are used to measure the direction of the gravity vector. To maintain vertical alignment of the coordinate system, one must consider the effect of horizontal acceleration on the displacement of the vertical instrument axes. Thus, inertial navigation systems must be corrected such that they do not tilt when accelerated. This is achieved by Schuler-tuning the system with an 84.4 minute period. The equation for the Schuler period is shown in equation 6.2.

$$\tau_s = 2 \cdot \pi \cdot \sqrt{\frac{R_o}{g}} \tag{6.2}$$

where τ is the Schuler period, R_o is the radius of the earth and g is the acceleration due to gravity.

Hence, inertial navigation systems are subject to two types of errors, these oscillate at the Schuler rate and earth rate. For instance, misalignment errors not compensated for during the alignment process will propagate at the Schuler rate. The diurnal effect is not considered here. As demonstrated in Chapter 5, applied rate corrections at frequencies higher than the Schuler frequency are used to damp out INS errors. Optimum values for R and the filter time constant are chosen that result in a stable platform that is not perfectly Schuler-tuned. Furthermore, in the wander-azimuth mechanization only the vertical component of earth rate is corrected.

The column vector $\vec{\mu}_c$ as determined in equation 6.1 is written as a 4×4 skew-symmetric matrix and used to update the quaternion as shown in equation 6.3 [26].

$$\begin{bmatrix} \frac{d}{dt}(q_1) \\ \frac{d}{dt}(q_2) \\ \frac{d}{dt}(q_3) \\ \frac{d}{dt}(q_4) \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} 0 & \mu_z & -\mu_y & \mu_x \\ -\mu_z & 0 & \mu_x & \mu_y \\ \mu_y & -\mu_x & 0 & \mu_z \\ -\mu_x & -\mu_y & -\mu_z & 0 \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{6.3}$$

The quaternion parameters are defined in terms of a set of four coupled linear differential equations. Hence, the parameters can be solved for by integrating the differential equations provided initial conditions for the parameters exist. The initial quaternion parameters are found during the self-alignment process. A numerical method must be used to solve for the quaternions since the gyro outputs are inexact quantities.

The four quaternion parameters (q_1, q_2, q_3, q_4) obey equation 6.4, that is, they are not independent.

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{6.4}$$

This condition is periodically enforced when the system is in navigation mode, to minimize errors. In theory, the transformation matrices are orthogonal. However, inaccuracies in the measurements, quantization and truncation errors contribute to form a transformation matrix that does not effectuate a pure rotation of the vector but instead alters the vector length. Constraining the transformation matrix to be orthogonal effectively forces the pure rotation condition.

A substantial number of studies have been done to determine the best orthogonalization process. The underlying purpose of these consists of obtaining the pure rotation that best matches reality. One such method is the dual iterative algorithm proposed in the paper by Bar-Itzhack, Meyer and Fuhrman. This method consists of finding the square root of the matrix multiplied by its transpose and multiplying the result by the inverse of the transpose of the estimated matrix. An alternative orthogonalization algorithm by Bjorck and Bowie is also described in the same paper [37]. For this application, the quaternion parameters are orthogonalized after every integration solution. Consequently, a faster, simpler method has been chosen. Each quaternion parameter is normalized by dividing it by the square root of the sum of squares, as in equation 6.4. An optimal, more time-taxing, orthogonalization procedure could be utilized at intermediate, larger time intervals.

Other sources of concern are non-commutativity errors that are a consequence of the fact that the order in which the rotations occur is not commutative. These errors are minimized by increasing the number of computation updates and occasionally combining

current gyro outputs with previous outputs, as Yeon Fuh Jiang and Yu Ping Lin demonstrate in their paper [38]. Therefore, to minimize commutativity errors, the quaternion parameters are updated in the software at the maximum sampling rate of the HRU. The navigation computations can be categorized into groups that must be evaluated at fast, intermediate or slow rates. The quaternions are evaluated at the fast rate. System performance levels can be enhanced by combining the quaternion method with a rotation vector concept that further reduces non-commutativity errors. The rotation vector concept is a pre-processor coning algorithm that can use between one and four gyro samples to improve the strapdown attitude realization. Because the inertial sensors in a strapdown system are subjected to the high dynamics experienced by the vehicle, a supplementary attitude algorithm that accounts for high frequency base motion can be used to improve the quaternion attitude algorithm efficiency. Different feasible pre-processor algorithms, not currently implemented in the present HRU, are described in [38] and [39].

6.3 Euler Angle Generation

The instantaneous rotation detected by the vehicle is given by equation 6.5 where the ω's are the angular velocities and $\hat{i}$, $\hat{j}$, and $\hat{k}$ are the unit vectors along the X, Y, and Z gyroscope system input axes about which the rotation is detected.

$$\vec{\vartheta} = \omega_x \cdot \hat{i} + \omega_y \cdot \hat{j} + \omega_z \cdot \hat{k} \tag{6.5}$$

Since the transformation matrix is defined in terms of quaternion parameters, the Euler

angles are not directly available. However, the relation between the Euler angles and the transformation matrix is known, therefore they can be calculated subsequently. Equation 6.6 is the equation for the quaternion matrix [6].

$$\mathbf{Q} = \begin{bmatrix} (q_4^2 + q_1^2 - q_2^2 - q_3^2) & 2\cdot(q_1\cdot q_2 + q_3\cdot q_4) & 2\cdot(q_1\cdot q_3 - q_2\cdot q_4) \\ 2\cdot(q_1\cdot q_2 - q_3\cdot q_4) & (q_4^2 - q_1^2 + q_2^2 - q_3^2) & 2\cdot(q_2\cdot q_3 + q_1\cdot q_4) \\ 2\cdot(q_1\cdot q_3 + q_2\cdot q_4) & 2\cdot(q_2\cdot q_3 - q_1\cdot q_4) & (q_4^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{6.6}$$

The quaternion matrix is associated with a unique direction cosine matrix, equation 6.7. The transformation matrix transforms a vector in the system frame of reference to the computational frame of reference. Thus, the equivalent DCM **C** is obtained from a sequence of a roll rotation (ϕ), followed by pitch (θ) and finally by yaw (ψ). The convention adopted throughout this thesis for performing coordinate transformations is to multiply the rotation matrices in sequential order from right to left.

$$\mathbf{C} = \begin{bmatrix} C_\theta\cdot C_\psi & -C_\phi\cdot S_\psi + S_\phi\cdot S_\theta\cdot C_\psi & S_\phi\cdot S_\psi + C_\phi\cdot S_\theta\cdot C_\psi \\ C_\theta\cdot S_\psi & C_\phi\cdot C_\psi + S_\phi\cdot S_\theta\cdot S_\psi & -S_\phi\cdot C_\psi + C_\phi\cdot S_\theta\cdot S_\psi \\ -S_\theta & S_\phi\cdot C_\theta & C_\theta\cdot C_\phi \end{bmatrix} \tag{6.7}$$

Therefore, the Euler angles, roll, pitch, and yaw can be solved for by equating the DCM **C** and quaternion **Q** matrix elements. The vehicle tilt and attitude information is found as shown in equations 6.8.

$$\begin{aligned} \psi &= \arctan\left(\frac{C_{21}}{C_{11}}\right) \\ \theta &= \arcsin(-C_{31}) \\ \phi &= \arctan\left(\frac{C_{32}}{C_{33}}\right) \end{aligned} \tag{6.8}$$

The principal outputs of an AHRS are heading and attitude information. The wander-

azimuth mechanization is a local level system, consequently the roll and pitch angles can be used directly. This is not the case for heading information, which is by convention defined as an angular deviation from North. Unlike a geographic frame of reference, the wander-azimuth mechanization is not north-pointing. A wander angle correction must be applied to the azimuth angle to obtain the true heading of the system. The wander angle is the positive counter-clockwise divergence angle away from the north axis. The true system heading, ψ_T is found using expression 6.9.

$$\psi_T = \psi - \alpha \tag{6.9}$$

where ψ is the azimuth angle and α the wander angle.

The convention adopted for the ranges of the heading and attitude information is the following: true heading lies between ±180 degrees where positive angles are those counter-clockwise from North, roll is between ±180 degrees and pitch is between ±90 degrees. Positive angles are defined according to the right-hand rule.

The transformation matrix is used in ensuing calculations but not the attitude and heading information Thus, these navigation outputs are calculated at the user-prescribed rate. This update rate is typically on the order of one second, which is a slow rate in comparison to other navigational update standards, such as Schuler-tuning.

Attitude and tilt rates can also be determined from the Euler angles and the gyroscope angular velocity measurements The rates of change of the roll, pitch and yaw angles are needed in operating environments where the user wants to monitor high dynamic manoeuvres.

In Chapter 5, an alignment scheme is put forth that relies solely on the natural

vectors sensed by the inertial instruments. If attitude and heading information is available, such as when the vehicle was previously aligned in the same orientation, a fast alignment procedure can be performed. This initialization process uses the operator-supplied ϕ, θ, ψ angles to compute **C** and then solves for the quaternion parameters. The quaternions as functions of the direction cosines are given in equation 6.10 [3].

$$
\begin{aligned}
q_4 &= \pm\frac{1}{2}\cdot\sqrt{(1+tr(C)} \\
q_1 &= \frac{1}{4\cdot q_4}\cdot(C_{23}-C_{32}) \\
q_2 &= \frac{1}{4\cdot q_4}\cdot(C_{31}-C_{13}) \\
q_3 &= \frac{1}{4\cdot q_4}\cdot(C_{12}-C_{21})
\end{aligned}
\tag{6.10}
$$

where $\mathrm{tr}(\mathbf{C}) = C_{11} + C_{22} + C_{33}$, is the trace of a 3×3 square matrix.

6.4 Vehicle Relative System Velocity

The next task of the navigation algorithm is to determine the HRU or vehicle relative system velocity. The rate of change of the system velocity relative to the earth, with respect to the rotating local-level coordinate frame is given by $\dot{V}$. The system acceleration is expressed as shown in equation 6.11.

$$\vec{\dot{V}} = \dot{V}_x\cdot\hat{\mathrm{i}} + \dot{V}_y\cdot\hat{\mathrm{j}} + \dot{V}_z\cdot\hat{\mathrm{k}} \tag{6.11}$$

The inertial acceleration components measured by the accelerometers are the acceleration of the system relative to the earth and the acceleration due to gravity. From the point of view of a distant observer, the accelerometer output corresponds to the

second order time derivative of R with respect to inertial space less gravitational mass attraction, where R is the distance travelled [3]. Positive gravity is assumed to act downward. Although the accelerometers do not distinguish between gravitational and non-gravitational acceleration, these are separated in the mathematical analysis. For navigation purposes the second order derivative of the position vector is important. The vector $\vec{A}$ represents the accelerometer measurements along the body axes of the vehicle corrected for bias and other error terms, refer to equation 6.12.

$$\vec{A} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \ddot{\vec{R}}_I - \vec{g}_m \tag{6.12}$$

The quaternion matrix is used to transform the accelerometer measurements resolved in body coordinates to local-level computational coordinates, the result is depicted by vector $\vec{A}_c$ in equation 6.13. The inertial acceleration outputs from the accelerometers are expressed in the local-level frame of reference.

$$\vec{A}_c = \mathbf{Q} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{6.13}$$

The general form for the transformation of the time derivative of a vector $\vec{V}$ between rotating and non-rotating axes is shown in equation 6.14.

$$\dot{\vec{V}}_{stat} = \dot{\vec{V}}_{rot} + \vec{\upsilon} \times \vec{V} \tag{6.14}$$

where $\vec{\upsilon}$ is the angular velocity of the rotating axes [40].

Similarly, the first derivative of the position vector is defined in terms of $\vec{V}$, the velocity vector representing the rate of change of the position vector $\vec{R}$ relative to the

earth, the position vector itself, and $\vec{\Omega}$ the angular rate of the earth relative to inertial space. Refer to equation 6.15.

$$\dot{\vec{R}}_I = \vec{V} + \vec{\Omega} \times \vec{R} \tag{6.15}$$

Equations 6.12 to 6.15 are used to solve for the system velocity relative to the rotating computational frame, referenced to the earth. For the relative acceleration case as it pertains to an INS, replace $\dot{\vec{V}}_{rot}$ with $\dot{\vec{V}}$, the vehicle velocity relative to the earth-referenced computational coordinates. Also, the acceleration of the system relative to inertial space in the local-level frame should equal $\dot{\vec{V}}_{stat}$. Thus, $\dot{\vec{V}}$ is the vehicle's acceleration relative to the earth and it is defined in terms of: the rotation of the earth relative to inertial space, the rotation of the local-level wander-azimuth frame relative to the earth and, the inertial acceleration terms in the local-level frame.

By differentiating expression 6.15 one more time and substituting it and the modified equation 6.15 into expression 6.12, the relative acceleration equation is found. It is given by equation 6.16. It is implied that all the vectors are expressed in the computational frame of reference. The accelerometer outputs are transformed to local-level coordinates whereas the other vectors are already in the desired frame of reference.

$$\dot{\vec{V}} = -(2 \cdot \vec{\Omega} + \vec{\rho}) \times \vec{V} + \vec{A}_c + \vec{g}_m - \vec{\Omega} \times (\vec{\Omega} \times \vec{R}) \tag{6.16}$$

where $(2 \cdot \vec{\Omega} + \vec{\rho}) \times \vec{V}$ term is the Coriolis acceleration, $\vec{V}$ is the velocity vector, $\vec{A}_c$ the inertial acceleration outputs in the local-level frame of reference, $\vec{g}_m$ the gravitational mass attraction, $\vec{\Omega}$ the angular rate of the earth relative to inertial space and $\vec{R}$ is the position vector.

The Coriolis acceleration is the normal acceleration that results when a body is moving around a point with a fixed angular velocity, and also moving radially with a constant speed. In other words, the Coriolis acceleration depicts the difference in relative acceleration terms as measured from rotating and non-rotating axes which is the case when a vehicle is moving with some velocity with respect to a rotating coordinate frame [40].

The accelerometers measure specific force, and so by applying Newton's second law of motion, the output is found to be proportional to the sum of vehicle acceleration along the Up direction and gravity, where gravity acts downward. But the inertial acceleration sensed by the vehicle relative to inertial space includes the centripetal acceleration that acts towards the centre of the earth. The centripetal acceleration corresponds to the last term in equation 6.16.

To obtain the acceleration of the vehicle relative to the rotating frame of reference, a gravity model is needed to compensate for mass attraction due to gravity and centripetal acceleration. The expression to be defined with an appropriate gravity model is given by $\delta(R)$ in equation 6.17. The gravity model should depict the plumb-bob gravity vector [3]. This implies that gravity acts perpendicular to the surface of the earth and consequently at non-equatorial latitudes it does not pass through the centre of the earth. Gravitational forces due to other celestial bodies such as the moon have been neglected for the near-earth navigation problem. $\delta(R)$ represents the apparent gravity vector and is based on the definition of the Schuler frequency, ω_S, the former is related to the latter as per $(\omega_S^2)R$.

$$\vec{\delta}(R) = \vec{g}_m - \vec{\Omega} \times (\vec{\Omega} \times \vec{R}) \tag{6.17}$$

where $\vec{\Omega} \times (\vec{\Omega} \times \vec{R})$ is the centripetal acceleration.

It is important to have an accurate gravitational model of the earth if the vehicle acceleration components are to be correctly extracted from the accelerometer outputs. The gravity model is dependent on the parameter R, the radius of the earth, in two ways. The most obvious is the centripetal acceleration which is directly related to the direction and magnitude of the position vector. Secondly, the earth is not perfectly spherical, hence the mass gravitational potential derivation is more accurately based on an ellipsoidal model of the earth. The consequence of the equatorial bulge and flattening of the poles, is a gravity vector that is perpendicular to the reference ellipsoid but when extrapolated it does not point to the centre of the earth at non-equatorial latitudes. Hence, the parameter R varies according to the earth's geometry. Deflections of the vertical refer to the difference between the directions of the gravity vector and the normal to the reference ellipsoid.

Another consequence of the asymmetry in the earth's mass distribution is that the mass anomalies cause actual deviations of the gravity magnitude. Thus, an oblate spheroid earth model which accounts for the gravitational potential gradient, different radii of curvature, and effectively assigns the equatorial earth radius as the semi-major axis of an ellipse and the polar earth radius as the semi-minor axis is preferable [22]

A simplified earth model is implemented with the navigation algorithms. For marine and terrestrial applications, the altitude 'h' above the surface of the earth is not as critical as it is for an aircraft and consequently is neglected. The spherical earth model employed

assumes the gravitational force is constant and concentrated entirely along the vertical direction. This assumption is based on an earth model that is symmetric about the polar axis. For simplicity centripetal accelerations are neglected since as Siouris demonstrates in his text [3] the vertical component of centripetal acceleration is on the order of 0.003g. As an approximation a nominal value of 6378.14 km is used for the radii of curvature of the earth. Future developments with an emphasis on performance improvements should include an appropriate World Geodetic System based on reference ellipsoid navigational grids, such as the WGS-84 standard.

The relative acceleration vector equation rewritten in terms of the gravity vector is given by equation 6.18.

$$\dot{\vec{V}} = -(\vec{\rho} + 2 \cdot \vec{\Omega}) \times \vec{V} + \vec{A}_c + \vec{\delta}(R) \tag{6.18}$$

The general form of the linear relative acceleration components is shown equation 6.19.

$$\begin{aligned} \dot{V}_x &= A_{cx} - (\rho_y + 2 \cdot \Omega_y) \cdot V_z + (\rho_z + 2 \cdot \Omega_z) \cdot V_y + \delta(R)_x \\ \dot{V}_y &= A_{cy} - (\rho_z + 2 \cdot \Omega_z) \cdot V_x + (\rho_x + 2 \cdot \Omega_x) \cdot V_z + \delta(R)_y \\ \dot{V}_z &= A_{cz} - (\rho_x + 2 \cdot \Omega_x) \cdot V_y + (\rho_y + 2 \cdot \Omega_y) \cdot V_x + \delta(R)_z \end{aligned} \tag{6.19}$$

For constant level trajectories the vertical velocity can be assumed to equal zero and any small transient effects are neglected. This assumption is possible since there is minimal cross-coupling between the level and vertical channels as Huddle confirms in his paper [26]. For travel along the vertical axis the inherent instabilities in the vertical channel need to be addressed. Namely, error models of the vertical velocity and its integral the vehicle elevation equation must be developed to obtain precise elevation measurements.

Another navigation output of interest is the ground velocity. The level velocities are

found by numerically integrating the relative acceleration equations. The ground velocity is simply given by equation 6.20.

$$V_{gd} = \sqrt{V_x^2 + V_y^2} \tag{6.20}$$

The true angular rate of the local level frame relative to the earth, $\hat{\rho}$, is defined in terms of the level velocities, the radius of curvature of the earth and the wander angle, see equation 6.21. The notation used here does not imply that it is a unit vector, rather it is simply used to differentiate between true and damped parameters. The level spatial rates ensure that the platform remains horizontal and the azimuth precession is specific to the type of mechanization used [3]. The spatial rates are written in the wander-azimuth frame of reference and the approximations are a consequence of using a spherical earth model. That is, the radius of the earth along the North/South and East/West axes are considered to be equal. Since no compensation is made for the angular rate of the platform about the vertical, due to vehicle motion, it is inferred that the level axes will deviate from the geographic north-pointing system, hence the wander angle.

$$\begin{aligned}
\hat{\rho}_x &= \rho_E \cdot C_\alpha + \rho_N \cdot S_\alpha = -(\frac{V_N}{R_{N/S}}) \cdot C_\alpha + (\frac{V_E}{R_{E/W}}) \cdot S_\alpha \cong -\frac{V_y}{R_o} \\
\hat{\rho}_y &= -\rho_E \cdot S_\alpha + \rho_N \cdot C_\alpha = (\frac{V_N}{R_{N/S}}) \cdot S_\alpha + (\frac{V_E}{R_{E/W}}) \cdot C_\alpha \cong \frac{V_x}{R_o} \\
\hat{\rho}_z &= \left[\frac{\rho_y \cdot C_\alpha \cdot C_{\phi_L} + \rho_x \cdot S_\alpha \cdot C_{\phi_L}}{1 - S_{\phi_L}^2} \right] \cdot S_{\phi_L} + \dot{\alpha} = 0
\end{aligned} \tag{6.21}$$

The symbol used to represent the wander angle is the Greek symbol α and $\dot{\alpha}$ is the wander angle rate. The wander angle rate or divergence rate refers to the angular rate of the wander azimuth coordinate frame relative to the earth and it is computed using an

estimate of the velocity relative to the earth. The divergence rate is defined according to equation 6.22. Another interpretation of the divergence rate is that it is proportional to the rate of change of longitude times the secant of the latitude [3].

$$\dot{\alpha} = \tan(\phi_L) \cdot \left(\frac{C_a \cdot V_x - S_a \cdot V_y}{R_o} \right) \tag{6.22}$$

A damped set of spatial rate components is used to apply precession rates to the body referenced platform. It is different from the exact spatial rate components applied to the earth fixed platform used to obtain position information. The former is derived from raw gyro and accelerometer outputs, hence the requirement for damping discussed in Chapter 5. The damped spatial rates used for body platform control are given by equation 6.23. These are the parameters used in equation 6.1.

$$\begin{aligned} \rho_x &= \frac{-v_y}{R} \\ \rho_y &= \frac{v_x}{R} \\ \rho_z &= a \cdot \dot{V}_y \end{aligned} \tag{6.23}$$

6.5 Position Information

The direction cosine matrix **D** which transforms coordinates from the earth frame to the computational frame of reference is used to extract the position information. The earth-fixed axes form a right-hand orthogonal triad with the Y_e axis along the axis of rotation of the earth and the X_e and Z_e axes in the equatorial plane. **D** is defined by the following set of rotations: a positive rotation about the Y_e axis, by the angle λ, for longitude. It is

then followed by a rotation about the rotated X_e axis by an angle equal to $-\phi$, the negative latitude. Finally, the third rotation is the wander angle rotation, α , about the new $Z_{e\ axis}$. The wander angle is considered to be positive as the local-level axes are rotated in a counter-clockwise direction from North. The **D** matrix is given by equation 6.24.

$$\mathbf{D} = \begin{bmatrix} C_\alpha \cdot C_\lambda - S_\alpha \cdot S_{\phi_L} \cdot S_\lambda & S_\alpha \cdot C_\phi & -C_\alpha \cdot S_\lambda - S_\alpha \cdot S_{\phi_L} \cdot C_\lambda \\ -S_\alpha \cdot C_\lambda - S_{\phi_L} \cdot S_\lambda \cdot C_\alpha & C_\alpha \cdot C_{\phi_L} & S_\alpha \cdot S_\lambda - C_\alpha \cdot S_{\phi_L} \cdot C_\lambda \\ C_{\phi_L} \cdot S_\lambda & S_{\phi_L} & C_{\phi_L} \cdot C_\lambda \end{bmatrix} \tag{6.24}$$

Earth rate in the earth-fixed coordinates can be transformed to computational coordinates by the DCM **D**. It is useful to express earth rate in terms of the direction cosines themselves, as shown in equation 6.25. Note that the variable Ω represents full earth rate of 15.0417 deg/hr.

$$\begin{aligned} \Omega_x &= S_\alpha \cdot \Omega \cdot C_{\phi_L} = \Omega \cdot D_{12} \\ \Omega_y &= C_\alpha \cdot \Omega \cdot C_{\phi_L} = \Omega \cdot D_{22} \\ \Omega_z &= \Omega \cdot S_{\phi_L} = \Omega \cdot D_{32} \end{aligned} \tag{6.25}$$

The principal position navigation outputs, latitude and longitude are given by equation 6.26. The range for latitude is $\pm\pi/2$ with positive latitudes occurring in the northern hemisphere. Longitude and wander angle are defined in the range of $\pm\pi$. The Greenwich meridian is the zero longitude position and negative values correspond to locations West of Greenwich, England. The wander angle is not an essential navigation output but it is used to calculate the vehicle heading.

$$\begin{aligned}
\alpha &= \arctan\left(\frac{D_{12}}{D_{22}}\right) \\
\phi_L &= \arcsin(D_{32}) \\
\lambda &= \arcsin\left(\frac{D_{31}}{D_{11}\cdot D_{22} - D_{12}\cdot D_{21}}\right)
\end{aligned} \tag{6.26}$$

D can be updated using any one of the methods described in §6.2 for updating the quaternion matrix. The individual elements of the **D** matrix can be evaluated by solving a set of linear differential equations The rate of change of the direction cosines is equal to the angular rate of the local-level coordinates multiplied by the direction cosines, as shown in equation 6.27.

$$\dot{\vec{\mathbf{D}}} = \vec{\hat{\rho}} \cdot \vec{\mathbf{D}} \tag{6.27}$$

The undamped spatial rate components are used to precess the earth-fixed platform. Only six of the nine differential equations need be solved to find the required navigation outputs. Furthermore, since the vertical component of the spatial rate is zero the set of differential equations reduces to equation 6.28. The **D** matrix is updated at the same rate as the quaternions, however the new values of latitude, longitude and wander angle are displayed according to the user's requirements, which is typically a 1 Hz rate.

$$\begin{aligned}
\dot{D}_{11} &= -\hat{\rho}_y \cdot D_{31} \\
\dot{D}_{21} &= \hat{\rho}_x \cdot D_{31} \\
\dot{D}_{31} &= \hat{\rho}_y \cdot D_{11} - \hat{\rho}_x \cdot D_{21} \\
\dot{D}_{12} &= -\hat{\rho}_y \cdot D_{32} \\
\dot{D}_{22} &= \hat{\rho}_x \cdot D_{32} \\
\dot{D}_{32} &= \hat{\rho}_y \cdot D_{12} - \hat{\rho}_x \cdot D_{22}
\end{aligned} \tag{6.28}$$

The actual relative acceleration equations implemented in the navigation algorithms are

expressed in terms of the direction cosines in the **D** matrix, as per equation 6.29.

$$
\begin{aligned}
\dot{V}_x &= A_{cx} - (\rho_y + 2\cdot\Omega\cdot D_{22})\cdot V_z + (2\cdot\Omega\cdot D_{32})\cdot V_y + \delta(R)_x \\
\dot{V}_y &= A_{cy} - (2\cdot\Omega\cdot D_{32})\cdot V_z + (\rho_x + 2\cdot\Omega\cdot D_{12})\cdot V_z + \delta(R)_y \\
\dot{V}_z &= A_{cz} - (\rho_x + 2\cdot\Omega\cdot D_{12})\cdot V_y + (\rho_y + 2\cdot\Omega\cdot D_{22})\cdot V_x + \delta(R)_z
\end{aligned}
\tag{6.29}
$$

The Navigation software flowchart in Chapter 7 of this thesis illustrates how the navigation equations are implemented in the software.

Chapter 7

Software Implementation and Results

7.1 Programming Techniques

The software development language used to code the navigation and alignment algorithms is Borland's Turbo C^{++} Version 3.0 [41]. The principal directives employed during compilation include: optimization for size, small memory model, 80286 instruction set and availability of 80287 floating-point co-processor. The navigation and alignment algorithms are implemented using conventional double precision arithmetic.

The software design methodology consists of a modular, functional programming approach whereby the navigation task is subdivided into several small segments. Functional decomposition is a general purpose design method that is well-suited for the type of problem addressed in this thesis, since the procedural steps are clearly evident. Modules are self-contained subsystems that execute a set of related instructions and which may contain their own local data. To minimize the complexity of individual modules, each module is generally associated with an independent, unique concept in the

navigation algorithms. Although utilizing a large number of modules enhances the clarity of the program and facilitates debugging of the piece of software, the disadvantage of having many modules resides in subroutine linkage overhead. Thus, the accepted point of view that a module should occupy between 7 to 40 lines of code [42] is adopted where possible.

In an attempt to minimize the overhead associated with frequently invoking the same module, macros have been used where appropriate. During the compilation process the macro name is replaced with direct in-line code.

It is beyond the scope of this thesis to detail the various language constructs available in C. However, one of the predominant elements which merits mention is the use of pointers. The pointer data type allows data objects to be referenced indirectly via their addresses. Thus, instead of passing entire arrays, passing the address of the head of the array is sufficient. Each element can be selected in terms of an offset from the first array element, referred to as the base. Furthermore, operations on the array elements, such as matrix multiplication, can be realized more quickly using pointer arithmetic. This feature is especially important since navigation and alignment applications must adhere to critical time constraints.

In their current states, the navigation and alignment programs are implementations of the algorithms derived in Chapters 5 and 6 of this thesis. Some windowing features and prompts have been added to interact with the user. However, an emphasis is not placed on user friendliness considerations since these have not been deemed essential in this phase of the software development.

The final implementation will use real-time interrupt-driven interfaces to the gyroscopes and accelerometers. However, to date, the hardware A/D converters have not been completed. For this reason, trial data sampled at a rate under 1 Hz, was recorded in files and used as input data in the current software implementation.

7.2 Coarse Alignment Software

The program structure flowchart in Figure 7.1 portrays the organization of the coarse alignment software as a series of hierarchical components. In summary, the sequence of events consists of initialization, file I/O operations, processing of gyroscope and accelerometer data until a decision condition is reached, data error compensation of the inertial sensor error terms, and finally computation of the $\mathbf{B}^T$ and $\mathbf{C}$ matrices. The CAV13.CPP, Coarse Align version 1.3 source code, is included in Appendix C.

In the initialization mode, gyroscope and accelerometer noise terms are specified. The noise terms are adjustable parameters that control the allowable standard deviation of the mean. The values of these depend on a combination of the inertial sensor accuracy, environmental and electronic noise under diverse operating conditions. For instance, in the event of significant base motion it is useful to have a means of adjusting the requisite coarse alignment accuracy to a smaller, more reasonable threshold. An investigation of the system noise levels under different environmental conditions would be worthwhile to compile a general set of noise guidelines for use in future versions of the HRU software.

Secondly, an initial estimate of the system's current latitude must also be entered

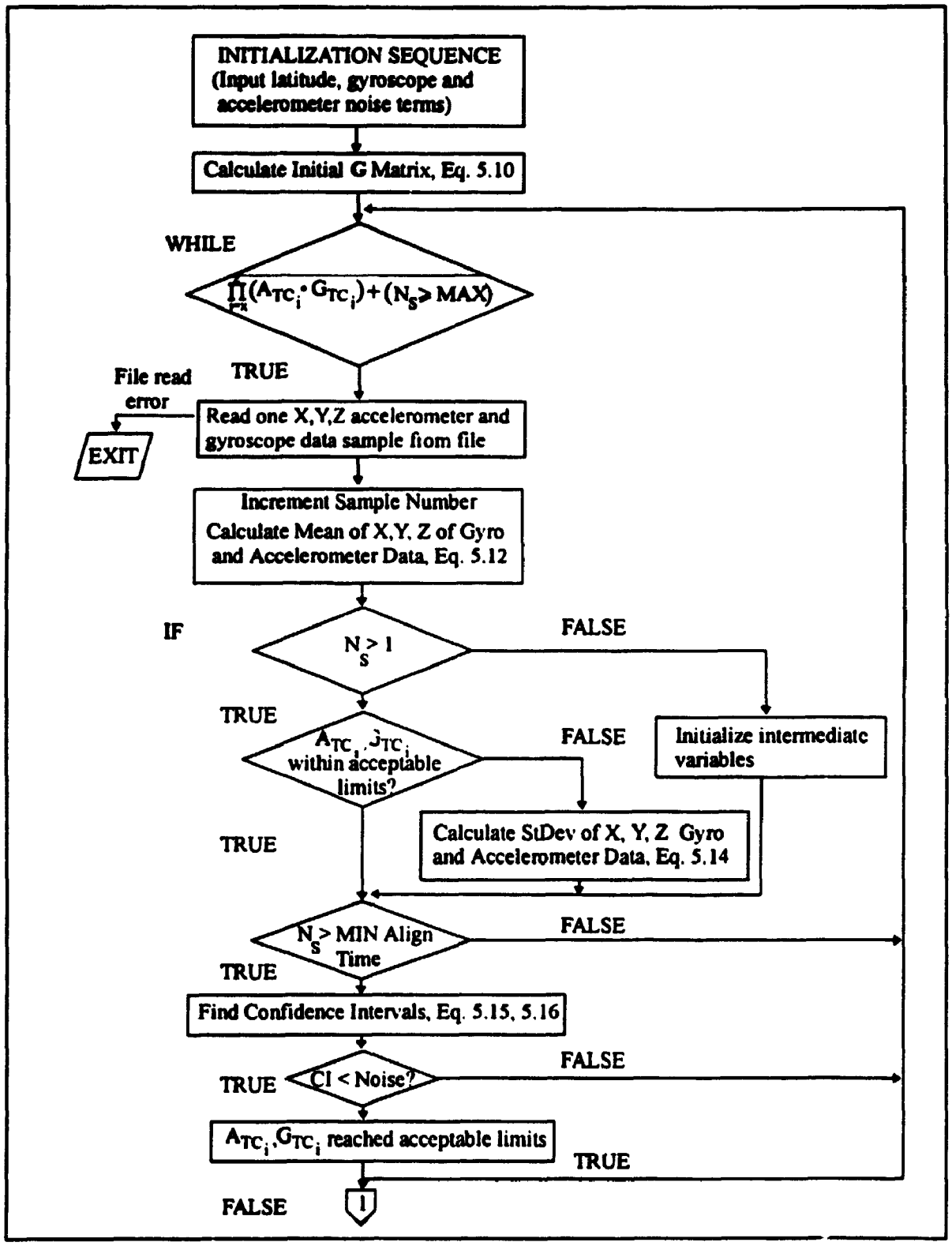
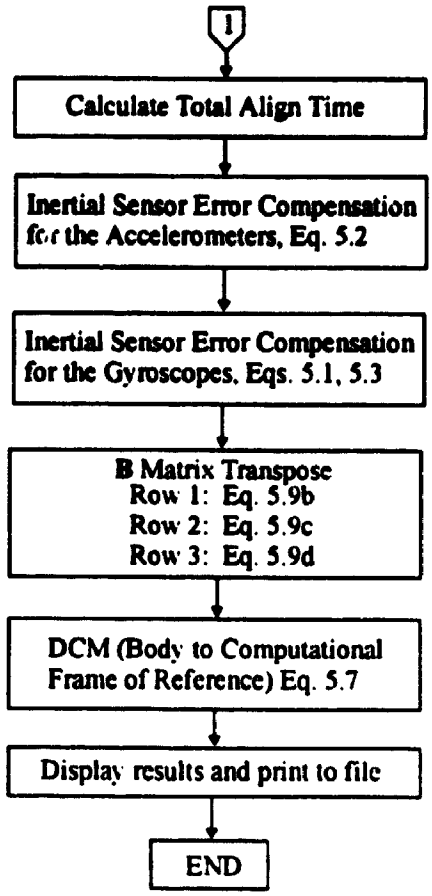


Figure 7.1: Coarse Alignment Program Flowchart



SYMBOLS:

$\cdot$	AND
$\overline{}$	NOT
$+$	OR
Π	PRODUCT OF SUMS
A_{TC_i}, G_{TC_i}	ACCELEROMETER AND GYROSCOPE t-DISTRIBUTION CONDITIONS i, SUBSCRIPT REFERENCE FOR AXIS
N_S	NUMBER OF SAMPLES
CI	CONFIDENCE INTERVAL

Figure 7.1: Coarse Alignment Program Flowchart

to start coarse alignment and enable calculation of the G matrix.

CAV13.CPP opens input gyroscope and accelerometer data files, generated by HRU CDU V1.5 for raw data collection. The header information of the input data file must conform to this standard file format to ensure it is compatible with CAV13.CPP.

Results are stored in a separate output data file. The Coarse Align output file consists of a header section, raw sensor outputs, intermediate and final results. The header contains initialization information such as system defaults and operator supplied parameters. It is followed by the averaged raw sensor data outputs. Intermediate results include the total time taken to align, error compensated sensor outputs, and their associated standard deviations and confidence intervals. Finally, the resultant $\mathbf{B}^T$ and the C direction cosine matrices are given. A sample CAV13.CPP output data file is included in Appendix B.1.

7.2.1 Experimental Static Data and Results

Static test data from the HRU in the NavLab is used to validate the coarse alignment algorithms and software. The HRU sensor orientations are changed by applying precise rotations about the Contraves Motion Table's tilt and azimuth axes. The appropriate DCM that transforms the body-referenced sensor data to computational coordinates is found regardless of the direction in which the instrument input axes are initially aligned and it is an indication of the systems' initial attitude and heading. For instance, any given gyroscope axis can be mounted to sense rates between 0 and 15.0417 deg/hr in a stationary environment.

For uniformity between test configurations, raw sensor data was collected for 20 minutes at each table position and used in its entirety for alignment purposes. In the home position, Position #2 in Tables 7.1, 7.2, the sensor package is oriented as shown below:

Home:	*Axis #1: 180.000 deg*	
	Axis #2: 180.000 deg	
Gyro S/N 19:	*X-Case: East*	*Y-Case: Down*
Gyro S/N 24:	*X-Case: North*	*Y-Case: Down*
Accelerometers:	*AX (S/N 232): East*	
	AY (S/N 228): South	
	AZ (S/N 234): Up	

Axes #1, and #2 refer to the CMT's azimuth and tilt axes, respectively. The static test setup is illustrated in Figure 7.2. The raw static accelerometer and gyroscope outputs corresponding to the home position are plotted in Figures 7.3, 7.4 and 7.5 The standard deviations for principal gyroscope axes data, not including the redundant axis, are on the order of 5.829, 6.315 and 8.226 deg/hr for the roll, pitch and yaw gyro data respectively. The standard deviations for the X, Y and Z accelerometer data are 0.194, 0.194 and 0.149 mg, respectively. These results are based on data sampled at less than 1 Hz and without any block averaging. Block averaging is not used because instantaneous data collection is a requirement of the navigation software. The use of higher sampling rates in the alignment algorithms would increase the sample size and should thereby reduce the standard deviations of the data

Alternatively, fine-tuning of the analog rebalance loop and research into the possible degradation of the CSG-2 gyros currently in the HRU should be investigated further to determine their effect on the noise characteristics of the sensor outputs.

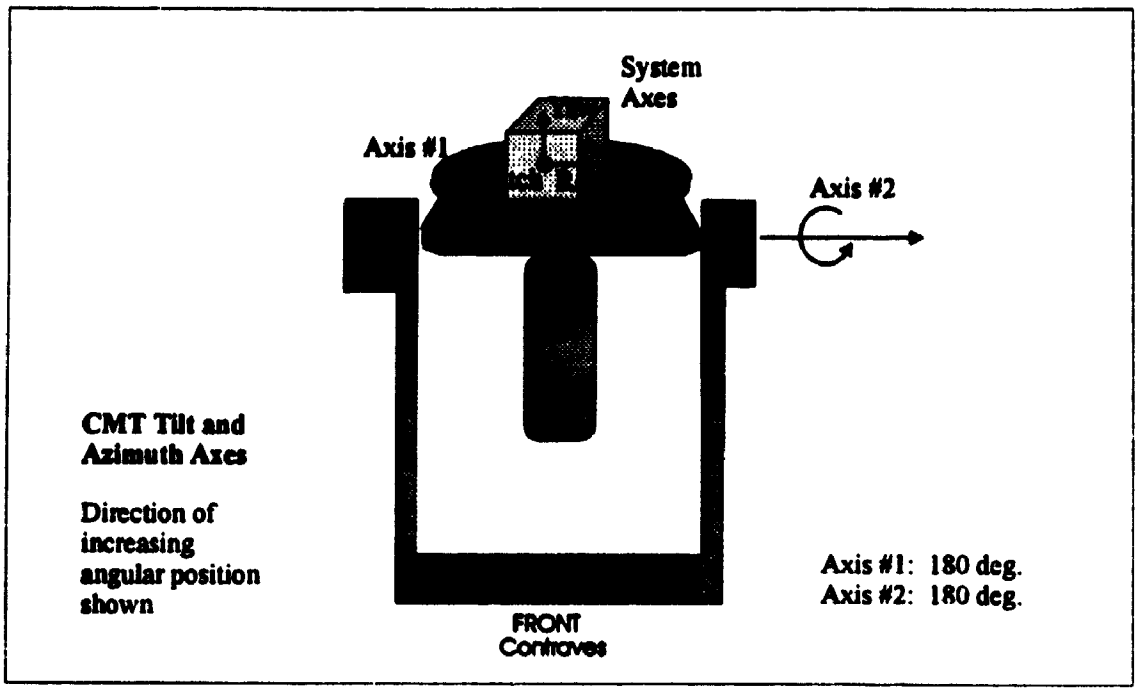


Figure 7.2: HRU Static Test Setup. The Inertial Sensor Assembly is shown mounted in the Home Position on the CMT.

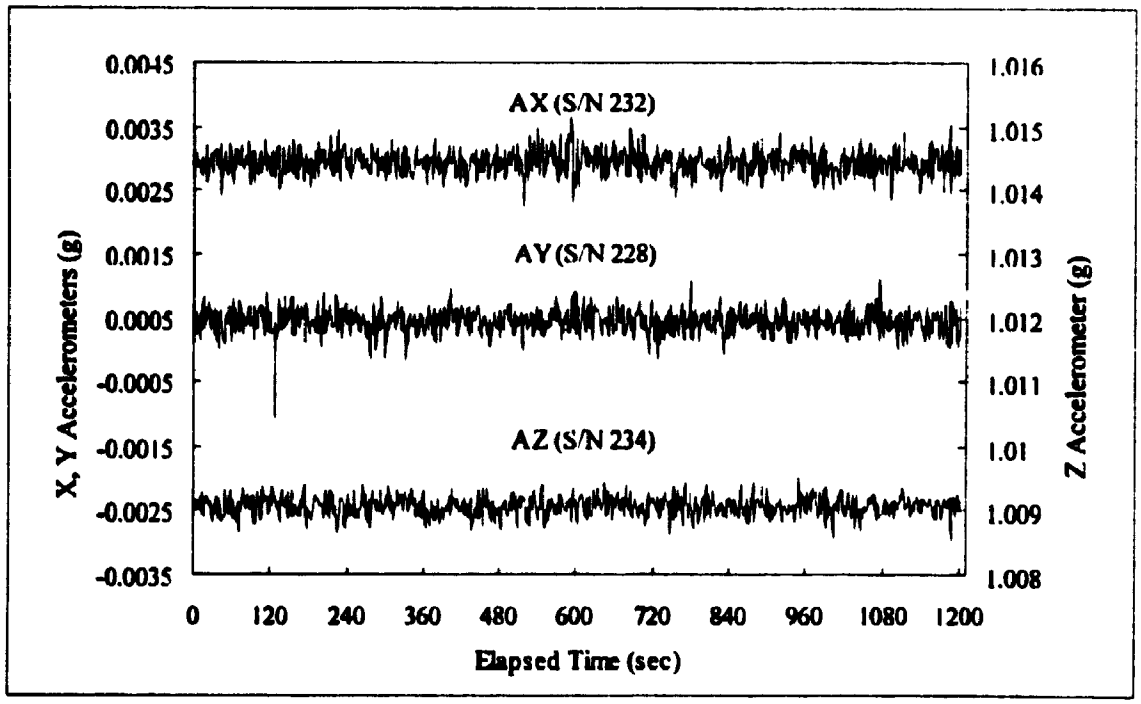


Figure 7.3: DREO HRU Accelerometer Static Data. The raw data outputs correspond to the configuration where the HRU is mounted in the Home Position. The accelerometer input axes are aligned: AX: East, AY: South and AZ: Up.

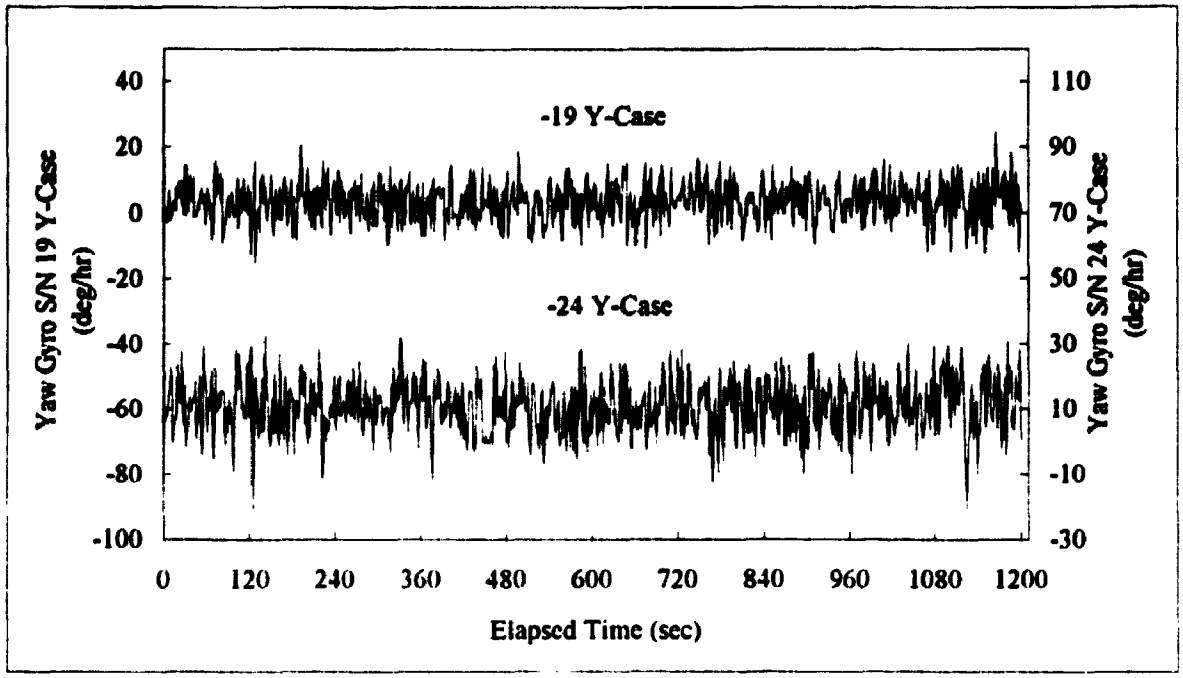


Figure 7.4: DREO HRU Yaw Gyroscope Axes Static Data. The HRU is mounted in the Home Position and Gyros S/N 19 and 24 Y-Case axes are pointing Down.

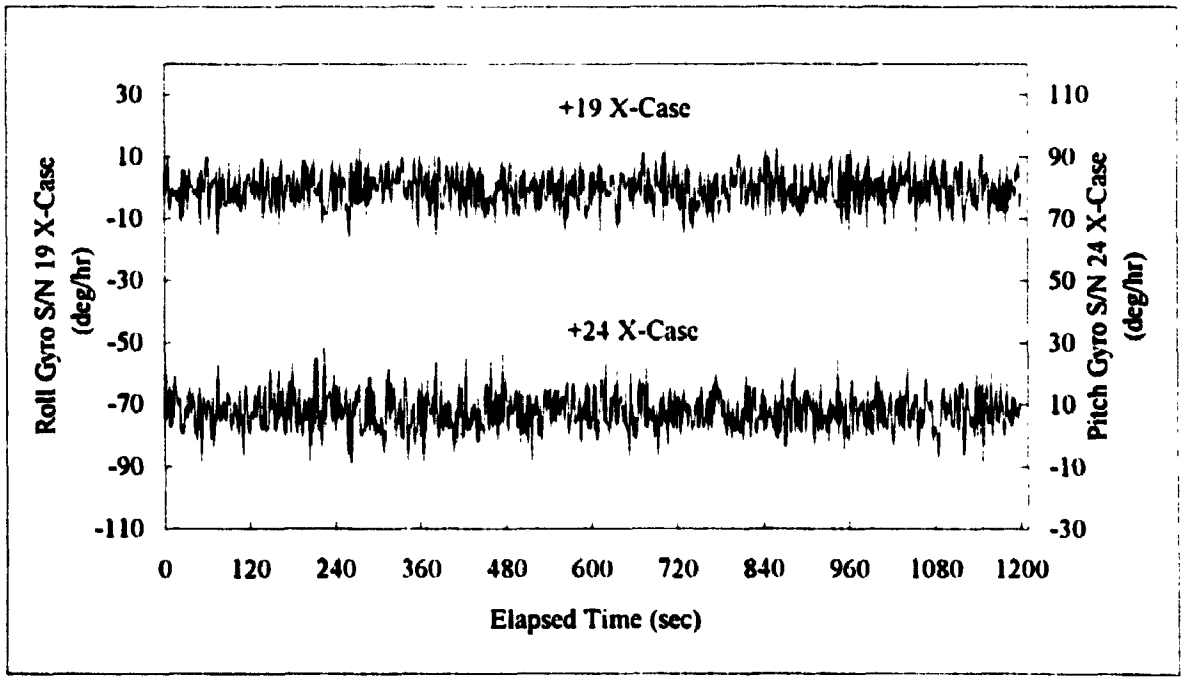


Figure 7.5: DREO HRU Roll, Pitch Gyroscope Axes Static Data. In the Home Position Gyro S/N 19 X-Case points East and Gyro S/N 24 X-Case points North.

Despite the high variances of the raw gyroscope outputs, and scale factor instabilities not being accounted for, the efficacy of the open loop alignment algorithms is summarized in Tables 7.1 and 7.2. The roll, pitch and yaw designations normally include misalignment and bias corrections, but Tables 7.1 and 7.2 are exceptions and these designations are only used to identify the axis along which the inertial sensor is oriented. The appropriate compensations appear in the transformed inertial sensor data only.

Also, it should be noted that since the bias drift, mass unbalance and quadrature terms have been determined using a multiposition test that did not exercise the CMT's tilt axis, this is an apparent source of error for position 7 where the rotation is applied about the tilt axis.

Table Position			Raw Gyro Data (Case Axes-deg/hr)			Transformed Gyro Data (deg/hr)		
Test Case	Axis #1 (deg)	Axis #2 (deg)	19 X -Roll X	24 X -Pitch Y	-24 Y +Yaw Z	East	North	Up
1	135.00	180.00	7.337	4.750	11.091	-5.39e-6	10.847	10.979
2	180.00	180.00	-0.300	7.809	10.633	1.79e-5	10.670	10.801
3	225.00	180.00	7.910	4.525	11.062	-5.70e-5	10.807	10.939
4	270.01	180.00	-10.931	-3.352	10.945	1.97e-4	10.609	10.739
5	15.00	179.99	2.884	-14.222	11.237	-6.97e-5	11.050	11.370
6	95.01	179.99	10.513	-5.290	11.363	6.98e-7	10.902	11.035
7	90.00	225.48	-1.745	-2.773	13.981	1.07e-4	9.515	9.631

Table 7.1: Coarse Alignment Results Verified with Gyroscope Data. For instance, the raw gyro data transformed to a geographic frame of reference yields angular rates of zero degrees per hour about the East axis and local earth rate components are sensed about the North and Up axes.

Test Case	Raw Accelerometer Data (g's)			Transformed Accelerometer Data (g's)		
	X -Roll	Y +Pitch	Z +Yaw	East	North	Up
1	2.9241e-3	4.292e-4	1.0090711	3.179e-6	-1.921e-2	0.9998601
2	2.9434e-3	4.339e-4	1.0090709	8.420e-8	-4.484e-2	0.9990304
3	2.9451e-3	4.182e-4	1.0090611	-1.718e-6	-1.831e-2	0.9998600
4	2.9552e-3	4.327e-4	1.0090558	-4.950e-8	-1.094e-2	0.9999697
5	2.9905e-3	4.381e-4	1.0090315	2.278e-6	-2.417e-2	0.9997204
6	2.9779e-3	4.365e-4	1.0090594	1.024e-6	5.992e-4	1.0000300
7	-0.674938	0.0105075	0.7074572	2.433e-6	8.635e-3	0.9733101

Table 7.2: Coarse Alignment Results Verified with Accelerometer Data. For instance, the raw accelerometer transformed to a geographic frame of reference, results in 1g being sensed along the Up axis and zero g along the level axes. Note: Refer to Table 7.1 for corresponding Axes #1 and #2.

Sensor outputs in computational coordinates are found by multiplying the compensated sensor outputs by the newly generated DCM. Thus, these computed results can be compared with precisely known quantities for the DREO NavLab CMT, namely, the fact that the local earth rate components along the East, North, and Up directions are 0, 10.57 and 10.70 deg/hr, respectively. Similarly, the most notable acceleration is the acceleration due to gravity and it acts entirely along the Up axis.

Considering only the six positions for which the bias drift and g-sensitive drift terms are best compensated, an estimate of the accuracy of the coarse alignment algorithm is obtained. By comparing the mean calculated East, North, Up components with their

actual known counterparts, the respective differences between measured and known quantities are: 0.05 arcsec/hr, -0.24 deg/hr, and 0.28 deg/hr. These numbers are a sufficiently accurate verification of the coarse alignment algorithm bearing in mind the standard deviations of the raw data.

Another source of error affecting the exactness of the calculated DCM is the magnitude of the gravitational vector. A nominal value of 9.806 m/s^2 is used, however, in the strictest sense this is not correct because this value fluctuates depending on the position of the package and its height above the reference ellipsoid. The insertion of an appropriate gravity model in the algorithms will remove this source of error.

Similarly, by considering the same six positions the accuracy of the coarse alignment algorithm can be obtained from accelerometer data. The mean calculated East, North, Up components of acceleration are compared with the true values. The respective differences between measured and known quantities are: 0.8 μg,-19 mg and -0.25 mg.

7.3 Navigation Software

The program structure flowchart of the navigation software is illustrated in Figure 7.6. There are two ways of initializing the navigation software. The first method consists of directly entering the DCM elements as found using the Coarse Alignment software. This process emulates the process that will occur when the coarse alignment and navigation software are combined to form a single program. Alternatively, the Euler angles can be entered if a secondary source of navigation information is available. Provisions should be made to use stored heading data from a previous session in later versions of the

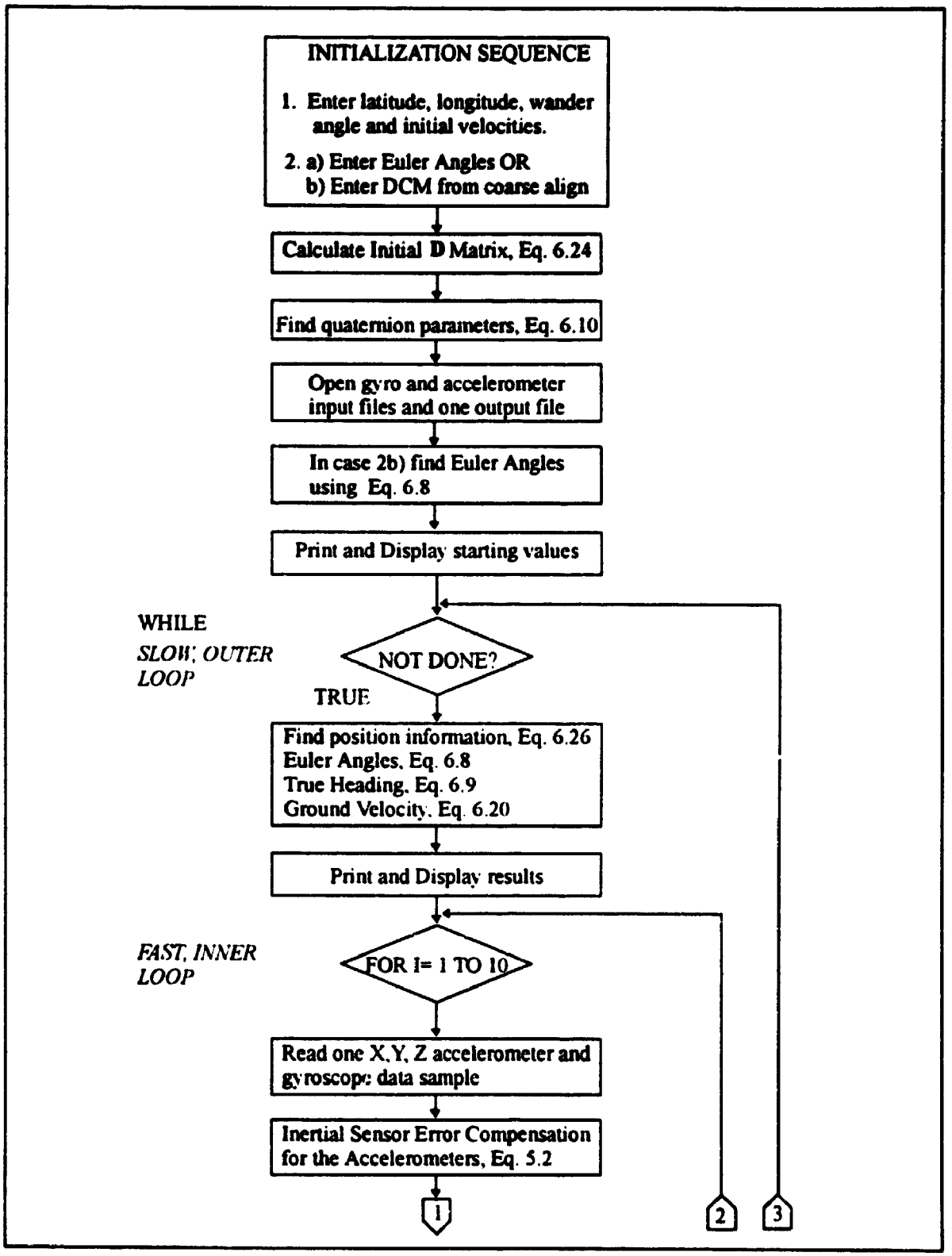


Figure 7.6: Strapdown Local-Level Wander-Azimuth Navigation Software Flowchart

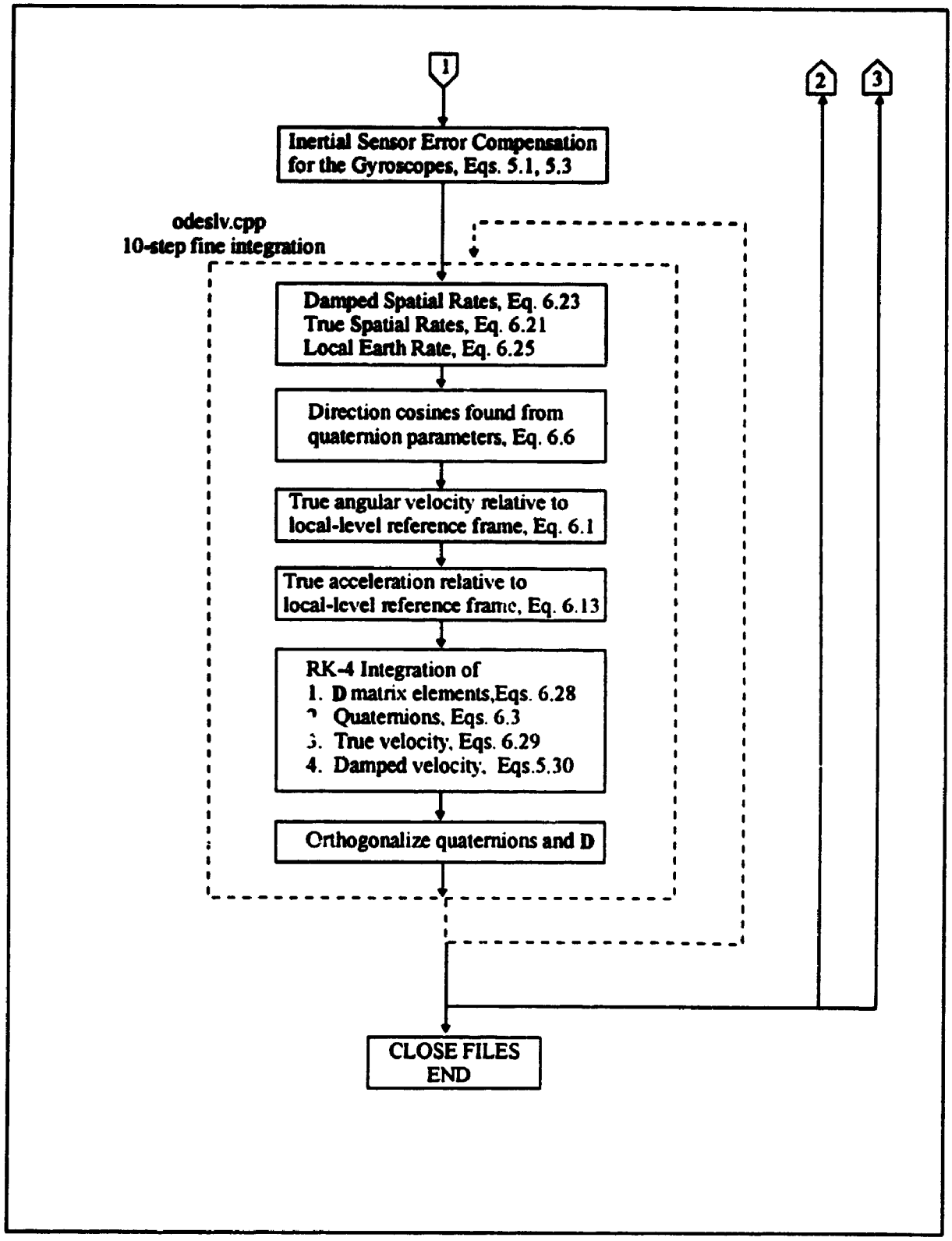


Figure 7.6: Strapdown Local-Level Wander-Azimuth Navigation Software Flowchart

software. Initial values for the **D** matrix are also needed before navigation can commence. The **D** matrix can be calculated from operator input latitude and longitude data. By default, the wander angle is zero. Similarly, by default, the initial velocities are also zero.

The navigation software post-processes raw HRU data stored in two separate data files containing gyroscope and accelerometer data. The files are created by the DREO raw data collection software HRU CDU V1.5 and contain GPS time-stamps at the start and end of the file.

The output data file generated by the strapdown local-level wander-azimuth navigation software, NAVIGAT.CPP consists of attitude, heading, position and velocity information at a rate equal to one-tenth the input data rate for long simulation files. However, since the present HRU sample data rate is under 1 Hz, the navigation outputs are supplied at a rate equal to the input sample rate when raw HRU data is processed. A partial sample NAVIGAT.CPP output data file is included in Appendix B.2.

7.3.1 Numerical Integration Techniques

The navigation solution involves solving 15 linear first order differential equations. The systems of four quaternion, 6 **D** parameter, 3 level velocity and 2 damped level velocity differential equations are initial value problems. The numerical methods considered to solve these differential equations include: Taylor's Series Method, Euler's Method, classical fourth order Runge-Kutta (RK), Milne's Method, Richardson Extrapolation and the Bulirsch-Stoer Methods [43].

The Taylor's Series Method requires higher-order derivatives and, for this reason, was considered too complex. Euler's Method is simple and does not require differentiation, however it is prone to large truncation errors. The classical fourth order Runge-Kutta has the advantage of a simple implementation. Furthermore, it does not require special procedures to start the routine and it has fewer formulas than a predictor-corrector algorithm. It also provides sufficient accuracy. A single-step approach with fixed step-size determined by the interval between data points is deemed to be a good solution.

A predictor-corrector algorithm, such as Milne's method is not the optimum approach because it requires more formulas than the RK method. Another disadvantage of the Milne's method is it is not a self-starting routine. However, it would be useful to keep a history of past values and use this method periodically to check the error.

The Richardson Extrapolation and the Bulirsch-Stoer are both high accuracy methods that require minimal computation effort. However, unlike the RK approach, these methods can fail and are very difficult to correct in the event that they do fail The Runge-Kutta-Fehlberg is an automatic step-size selection method. Its advantages are that it attempts to minimize both wasted computer time and large truncation errors. The first occurs when the step-size is too small and the latter occurs for step-sizes that are too large. However, this method requires too many equations and a fixed step-size is preferable for this particular application since the step-size can be dictated by the data rate.

Based on these considerations, the most viable approach for integrating the

navigation equations is a classical fourth order Runge-Kutta. Besides speed, accuracy and storage, propagation of error, as discussed in the next section, is an important concern when choosing a method for obtaining the navigation solution.

7.3.2 Propagation of Error

One principal source of error is roundoff errors which are a consequence of the computers' word length being finite. This implies that the final digit in the bit representation is only an approximation. Roundoff errors can be minimized by using a sufficiently large bit representation of the numbers, and scaling the data. The choice of optimum word length can be based on the maximum acceptable rms error for a specific number of computer iterations. The navigation algorithms are implemented using a 64 bit word length to minimize roundoff errors.

Another important source of error is due to truncation. These increase with larger step sizes However the disadvantage of using smaller step sizes is a resulting increase in the computational time. Alternatively, the truncation error can be reduced by using higher order integration schemes, hence the choice of a fourth-order integration method [22]

As discussed in Chapter 6 commutation errors are reduced by using the fastest rate of integration possible. Thus, commutation errors depend on the sensor data rate.

Quantization errors depend on how precisely an analog measurement is converted to a digital form. The overall system accuracy is limited by the bit representation available from the converters and in the worst case one bit of information is lost.

7.3.3 Modules

In this section the functions performed by the main program and all of its associated modules are briefly summarized:

NAVIGAT.CPP (main program):

This module prompts the user for initialization information, calculates necessary initial values, performs file I/O operations, and prints results to a display and to a file. It is organized into two main loops. In the outer loop the appropriate procedures are called for calculating position (Equation 6.26), ground velocity (Equation 6.20), Euler angle information (Equations 6.8 and 6.9) and the results are printed. The inner loop is executed at ten times the frequency of the outer loop. The input data is first read, then the data compensation routines (Equations 5.1, 5.2 and 5.3) are called, before performing the necessary integrations. The subroutines called by the main program are grouped into two separate files described below.

ODESLV.CPP:

This module file is called by the main program. The differential equations for obtaining the quaternions (Equation 6.3), **D** matrix elements (Equation 6.28), damped velocity and true velocity parameters (Equations 5.30 and 6.29) are integrated using a classical RK-4 method. For finer integration the step-size equals one-tenth the sampling rate and there are ten steps for each sample. The supplementary function calls required to evaluate all the variables in the differential equations include subroutines for solving the true and damped equations for the spatial rate (Equations 6.23 and 6.21), and the local earth rate

equations (Equation 6.25). Furthermore, a function call is made to evaluate the DCM based on the newly computed quaternion parameters (Equation 6.6). Finally, other functions for evaluating the intermediate equations include the true angular velocity of the vehicle relative to the local-level frame of reference (Equation 6.1) and the inertial acceleration outputs in the local-level frame of reference (Equation 6.13). At the end of the integration process the **D** matrix elements are normalized and forcing condition (Equation 6.4) is applied to the quaternion parameters.

MATRIX.CPP:

The principal functions called by the main program and the ordinary differential equation solver are located in this module. Furthermore, the constants such as gyro bias drift terms used by the data compensation modules, are compiled in this module.

The source code listing for the Strapdown Local-Level Navigation Software NAVIGAT.CPP V1.0, and its associated include files are provided in Appendix C.

7.3.4 Navigation Simulator Software

Slight modifications are made to the Navigation Software for use with simulated HRU data. For one, the data compensation algorithms are bypassed entirely since for testing purposes the input data is assumed to be ideal. Secondly, to simplify the generation of the input data file the integration step-size is adjusted for an input data rate of 1 Hz, as opposed to the larger sampling period of the Keithley meter, the temporary A/D converter in the existing HRU. The initialization sequence also differs according to the

type of simulation that is run, these are described below in §7.3.4.1 and §7.3.4.2.

The purpose of the simulation software is two-fold. First, the validity of the fine leveling and gyrocompassing theory developed in Chapter 5 is tested by injecting initial errors in the initial attitude matrix and verifying that they damp. Secondly, the navigation solution is verified using a simulated test track.

7.3.4.1 Initial Error Damping Test Cases

The error model developed in Chapter 5 describes a means of damping tilt and heading errors for gyro bias instabilities and manoeuvre accelerations by adjusting the precession rate applied to the mathematical platform. Thus, fine leveling and gyrocompassing is realized in the context of solving the navigation equations with applied precession rates defined in terms of three parameters, 'R', 'τ' and 'a'. This test procedure allows the effect of coupling between all three related error angle terms to be observed.

In this section, inaccurate initial coarse alignment is represented by a DCM that is erroneous with respect to the input data file. The inexact Euler angles used to compute the C DCM are hard-coded as part of the initialization process in the Navigation Simulator Software. The input data file is generated for the situation where the HRU is stationary and its X, Y, Z system axes are perfectly aligned along East, North, Up coordinates, respectively. Thus, the DCM $\mathbf{C}_b^n$ relating body coordinates to computational coordinates represents the misalignment between the computational and true navigational frames of reference since under ideal conditions $\mathbf{C}_b^n$ should be the identity matrix. In other words, the roll angle corresponds to the East error angle

between the true and estimated East axes, and similarly for the pitch and yaw designations.

The various case studies investigated are summarized below:

Case Study #1: This is the control case to which all the other test cases can be referenced. The DCM is the identity matrix which implies that there are no misalignment angles. In this case the roll, pitch and yaw angles are zero degrees initially and, as expected, they remained unchanged.

Case Study # 2: The Navigation Simulator software is initialized with a small level East initial misalignment error angle of 0.001 degrees. The values of the damping parameters are: τ=10800 sec, R=3000 km and a=0.43 sec/m. The coupling between the East and Azimuth channels is evident because both exhibit a decaying sinusoidal response. In comparison, the North channel error is negligible. The effect of having such a large time constant is very slow damping. The East, North and Up error angles are plotted in Figure 7.7.

Case Study #3: The parameter values remain unchanged from Case Study #2 except for the time constant τ which is set to the value chosen in §5.5.2.5, namely 1080 sec. This yields a better result than the previous case study since the damping is much more rapid. Refer to Figure 7.8. However, the disadvantage of reducing the time constant further is an increase in the steady-state error. Also, as indicated in Chapter 5, this simulation confirms that the loop time constant and the damping time constant of the error response differ by more than an order of magnitude.

Case Study #4: The purpose of this case study is to observe the effect of varying the 'a' parameter on the heading error response. The same parameter values as in Case Study #3 are chosen with the exception that 'a' is now reduced by a factor of 10. The results are plotted in Figure 7.9, and in comparison with the previous case study, it is evident that as 'a' decreases the period of oscillation of the error response increases. Also, the errors with a 24 hr period become more pronounced with smaller values of 'a'. Based on this observation, a value for 'a' of 0.43 sec/m is chosen.

Case Study #5: The error response is also observed for an initial heading error of 0.001 degrees and parameters, R=4000 km, τ=1080 sec and a=0.43 sec/m. Figure 7.10 demonstrates that these choices of applied precession rate parameters damp initial heading errors. Furthermore, the theory in §5.5.2.2 that a larger value of 'R' results in faster damping is confirmed when Figures 7.8 and 7.10 are compared. However, the steady-state error is larger for larger values of 'R', as shown in §5.5.2.5.

Case Study #6: Finally, the damping of an independent initial pitch error angle of 0.001 degrees is plotted in Figure 7.11. The parameter values are τ=1080 sec, R=4000 km and a=0.43 sec/m. The initial North error angle is coupled into the East channel, which in turn is coupled into the Azimuth channel. All three errors exhibit a decaying sinusoidal response.

The parameters of Case Study #3 are chosen for the fine leveling and gyrocompassing operations. Based on the above observations this parameter selection enables damping of the East, North and Up error angles.

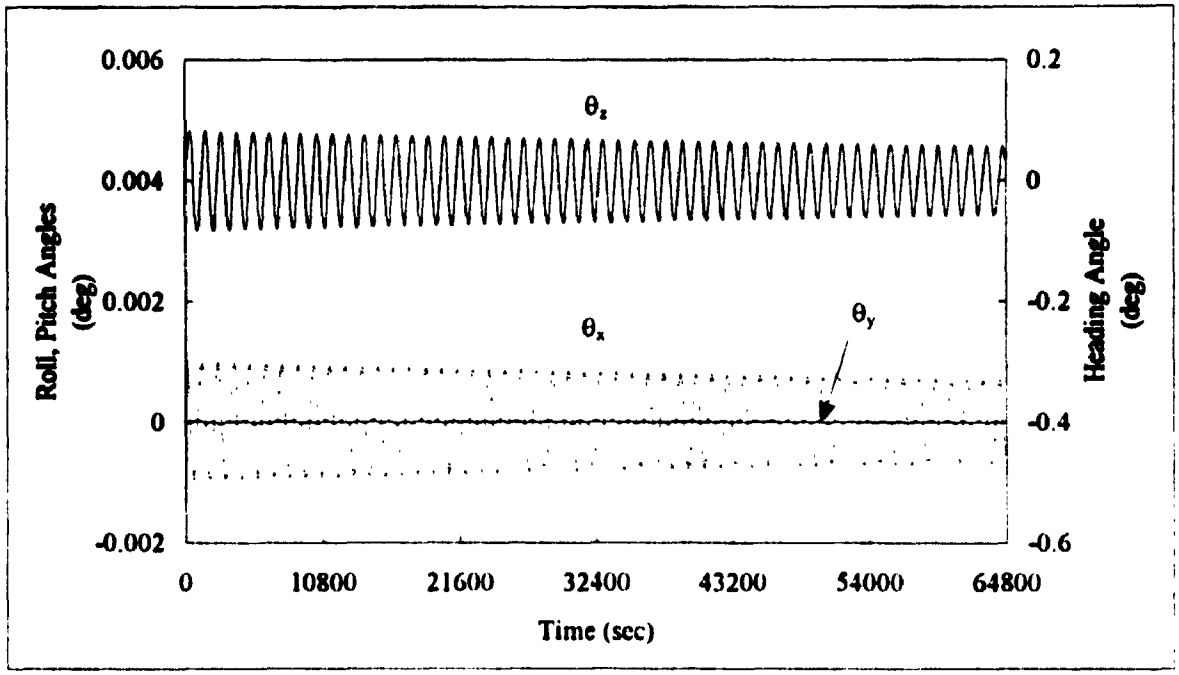


Figure 7.7: Initial Roll (θ_x) Error Angle of 0.001 deg (Case Study #2). Error angle response for parameter values: R=3000 km, τ=10800 sec, a=0.43 sec/m.

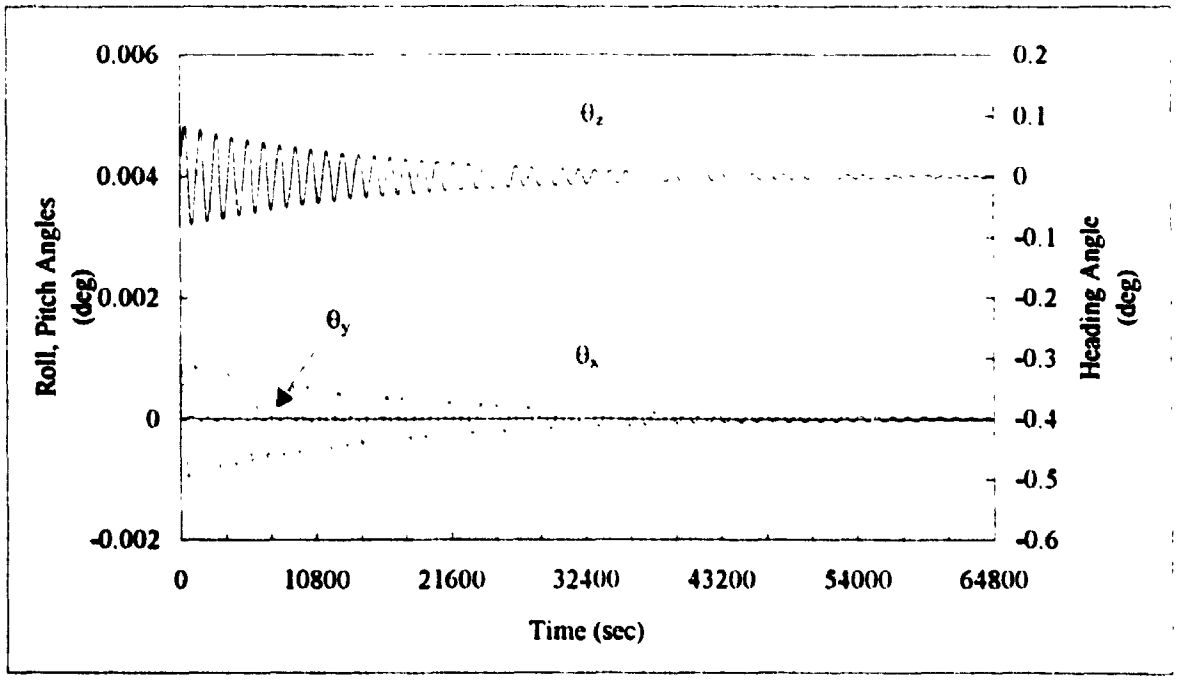


Figure 7.8: Initial Roll (θ_x) Error Angle of 0.001 deg (Case Study #3). Error angle response for parameter values: R=3000 km, τ=1080 sec, a=0.43 sec/m.

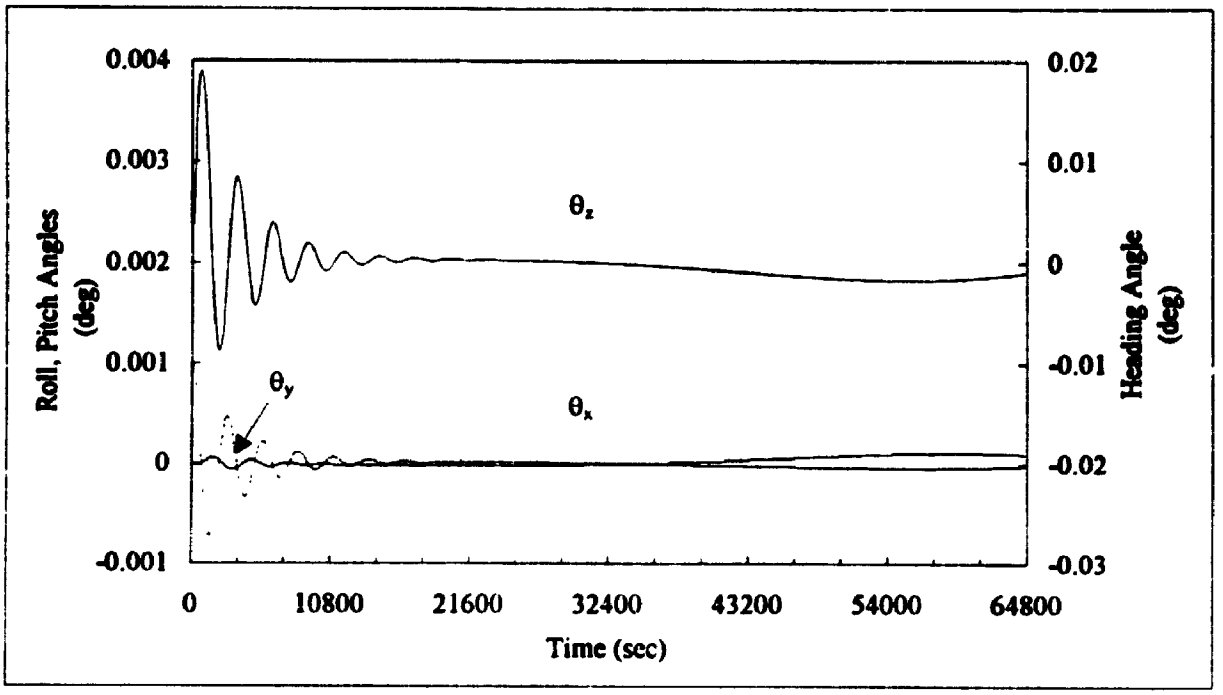


Figure 7.9: Initial Roll (θ_x) Error Angle of 0.001 deg (Case Study #4). Error angle response for parameter values: R=3000 km, τ=1080 sec, a=0.043 sec/m.

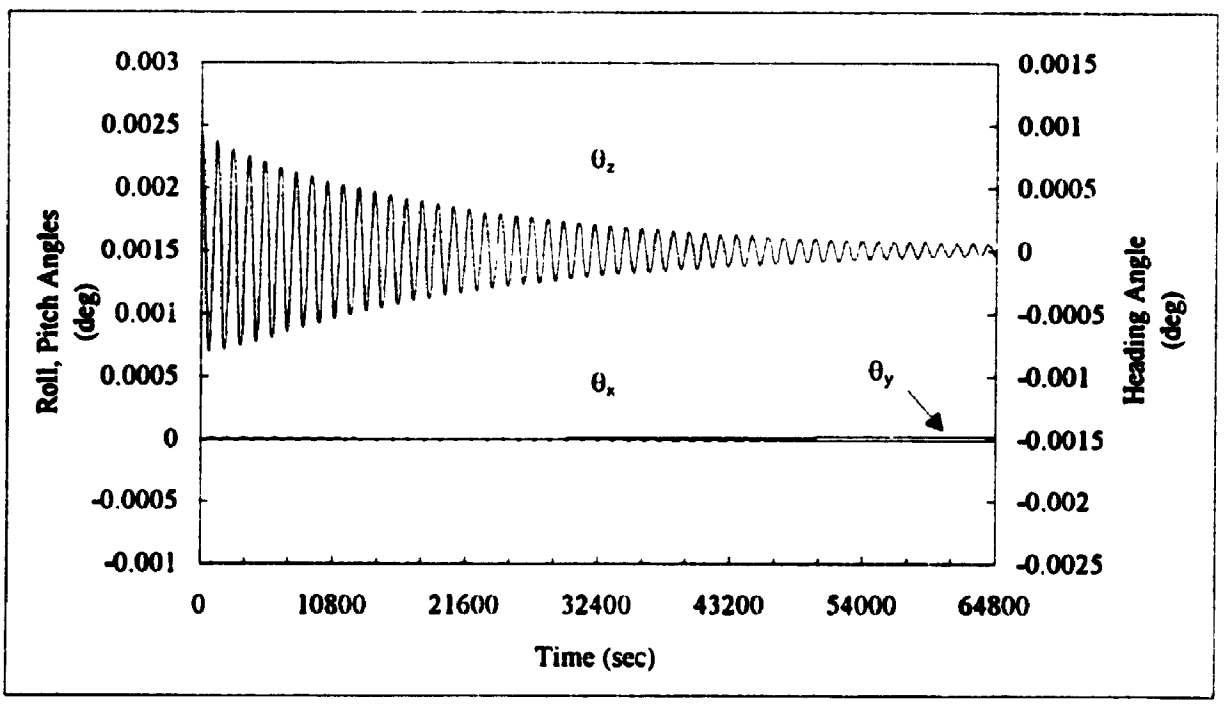


Figure 7.10: Initial Heading (θ_z) Error Angle of 0.001 deg (Case Study #5). Error angle response for parameter values: R=4000 km, τ=1080 sec, a=0.43 sec/m.

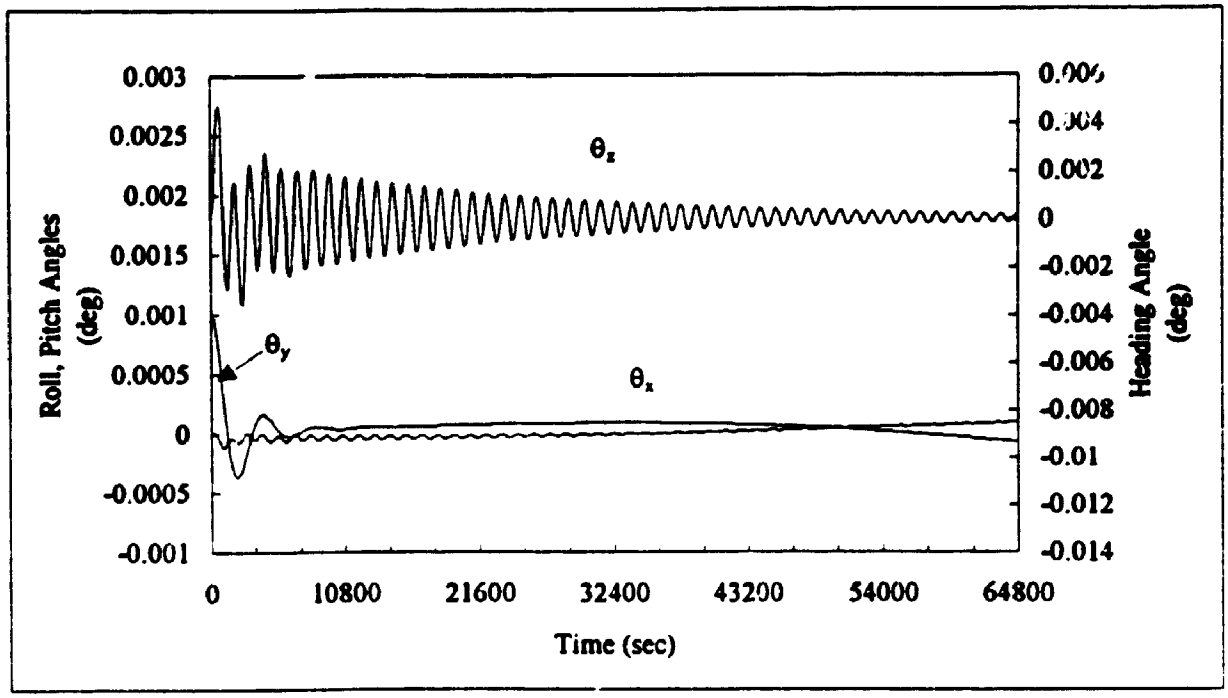


Figure 7.11: Initial Pitch (θ_y) Error Angle of 0.001 deg (Case Study #6). Error angle response for parameter values: R=4000 km, τ=1080 sec, a=0.43 sec/m.

7.3.4.2 Test Trajectory

Simulated gyroscope and accelerometer data for the test trajectory shown in Figure 7.12 is processed for the purpose of validating the navigation equations. There are five principal regions in the 2.5 km test track. These consist of three legs of constant velocity straight line motion linked by two 90 degree turns of opposite sense. There is no input acceleration. Initially, the system axes are assumed to be orient ' d ɔng the East, North and Up axes, respectively. The vehicle is travelling in straight line motion in a northerly direction at 20 kph. The vehicle then undertakes a negative 90 degree turn in 30 seconds and resumes travel at a constant velocity heading East. A second turn in the positive sense reorients the vehicle in a northerly direction. The relative distances travelled along each leg of the trajectory and the time taken are given in Figure 7.12.

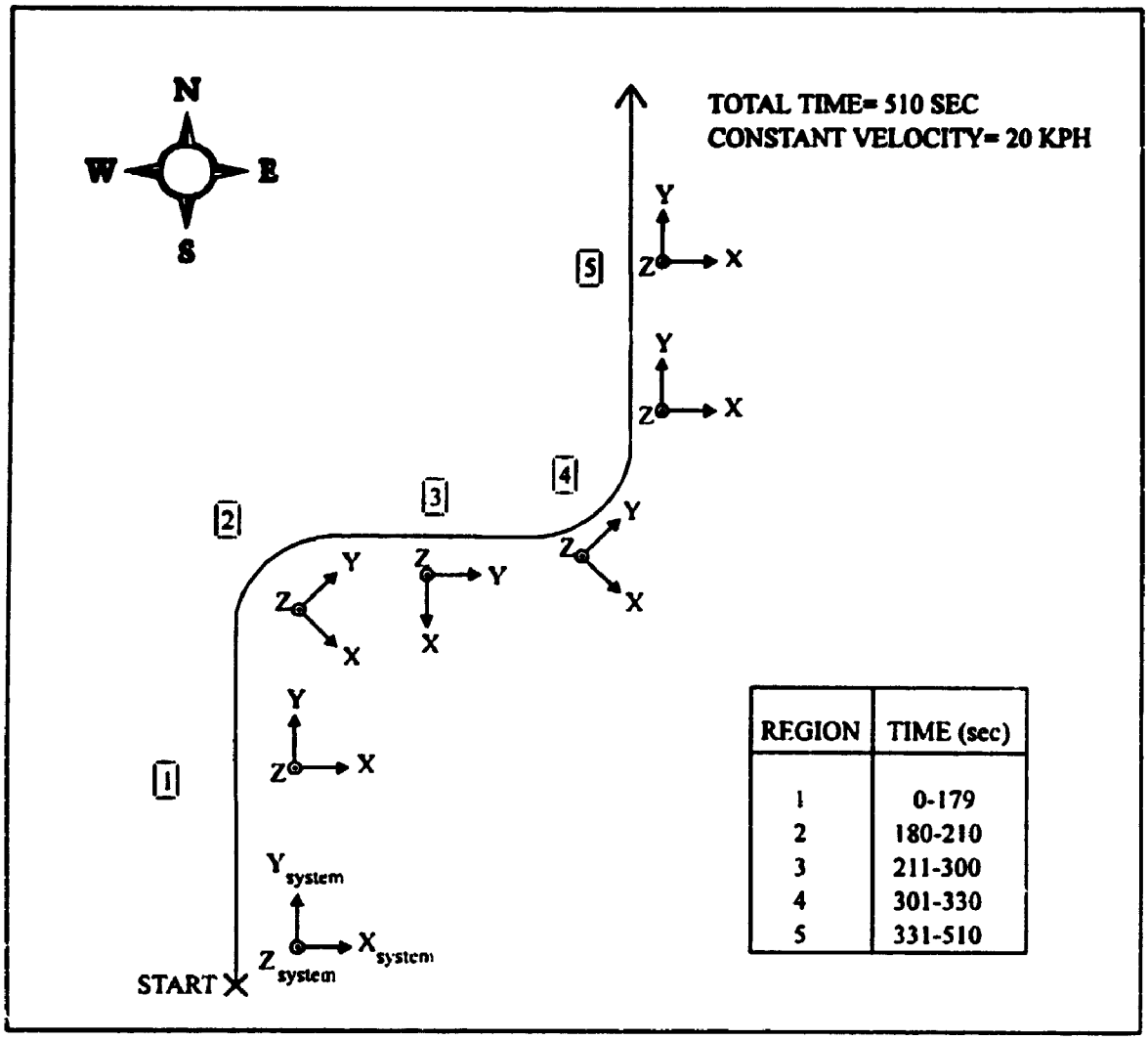


Figure 7.12: Simulated Test Track

The simulated sensor data is 1 Hz ideal data and it is void of all compensation errors and noise terms. The data format conforms to the outputs generated by the HRU CSG-2 gyroscopes and Q-Flex accelerometers. Rates sensed by the different gyroscope input axes include local earth rate, angular rate of the platform relative to the earth due to vehicle motion and angular rate of turn. The local earth rate components are a function of the latitude of the vehicle. Consequently, the resolution on the gyroscope output dictates the magnitude of the minimum detectable change in latitude. For these tests the

simulated data resolution matches the raw data collected by the HRU CDU software. The effect of the vehicle's orientation on the earth rate components sensed by the different gyroscope axes is also accounted for in the simulated data file.

Gravity and normal accelerations for curvilinear motion are the principal contributions to the accelerometer outputs. The initial alignment is achieved by setting $\mathbf{C}_b^n$ equal to the identity matrix. Hence, the navigation algorithms can be tested independently of the damping required for fine leveling and gyrocompassing. To determine the local earth rate components sensed by the different gyroscope input axes, a specific starting latitude is defined. This value for latitude is also used to initialize the **D** matrix.

The attitude and heading outputs from the Navigation Simulator program are plotted in Figure 7.13. In accordance with the test track, the roll and pitch angles are zero. The heading is initially zero and north-pointing. The heading then changes to positive 90 degrees, which implies that the body must rotate through 90 degrees in the positive counter-clockwise sense to point North The heading is then aligned with North again after a positive 90 degree turn.

The position data corresponding to the test track is plotted is Figure 7.14. By convention latitude is positive in the Northern hemisphere and longitude is positive East of the Greenwich meridian. Hence, the latitude is observed to increase over the regions where the vehicle is heading North. Also, longitude increases as the vehicle travels East. Furthermore, in the regions where the vehicle is moving North, a small increase in longitude is observed. The magnitudes of the latitude and longitude are in the right

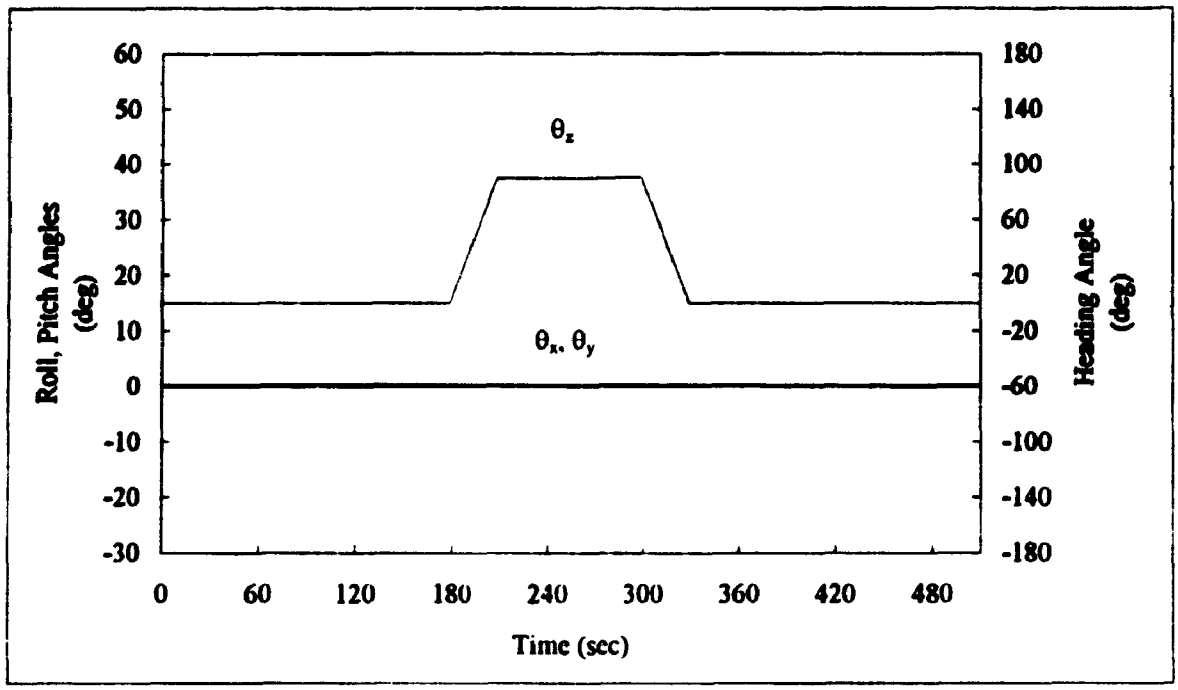


Figure 7.13: Attitude and Heading Results for Simulated Test Track. The vehicle travels in successive North, East, North directions.

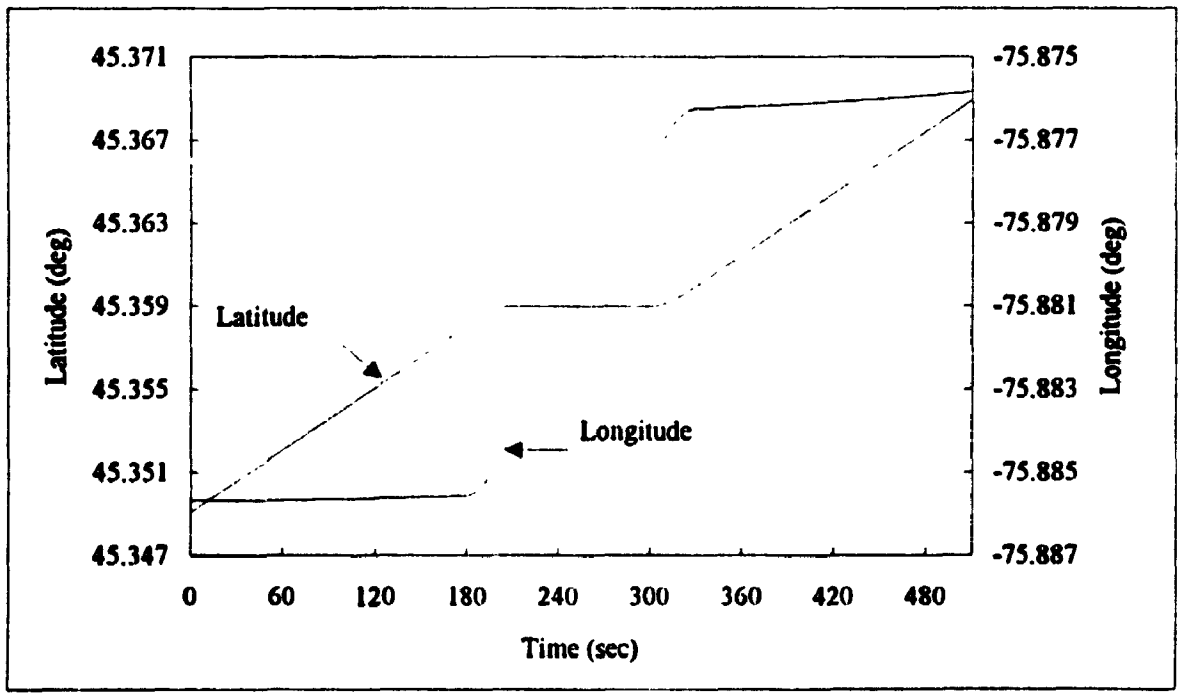


Figure 7.14: Position Results for Simulated Test Track. Increases in latitude and longitude correspond to vehicle travelling North and East, respectively.

range, also. At the equator one degree in latitude represents 60 nautical miles [44], where 1 nautical mile = 1.852 km. Therefore, 0.01 degrees of latitude at 45.349 degrees corresponds to approximately 0.78 km. Thus the position output is a reasonably good estimate of the distance travelled at a constant velocity of 20 kph and based on a 510 sec trajectory with two main regions effecting major changes in latitude.

7.3.5 Experimental Results from Dynamic Tests

HRU data collected during the DIINS (Dual Inertial Integrated Navigation System) Sea Trial, held in November 1994 on the Canadian West Coast, was post-processed using the Navigation Software, NAVIGAT.CPP. Various navigation systems were mounted aboard a ship called the CFAV Endeavour and subjected to numerous manoeuvres. The results, from a square box manoeuvre loop when the ship was travelling between 7 and 12 knots, are presented in this section. The box manoeuvre including the angles of approach and departure at either extremity is illustrated in Figure 7.15.

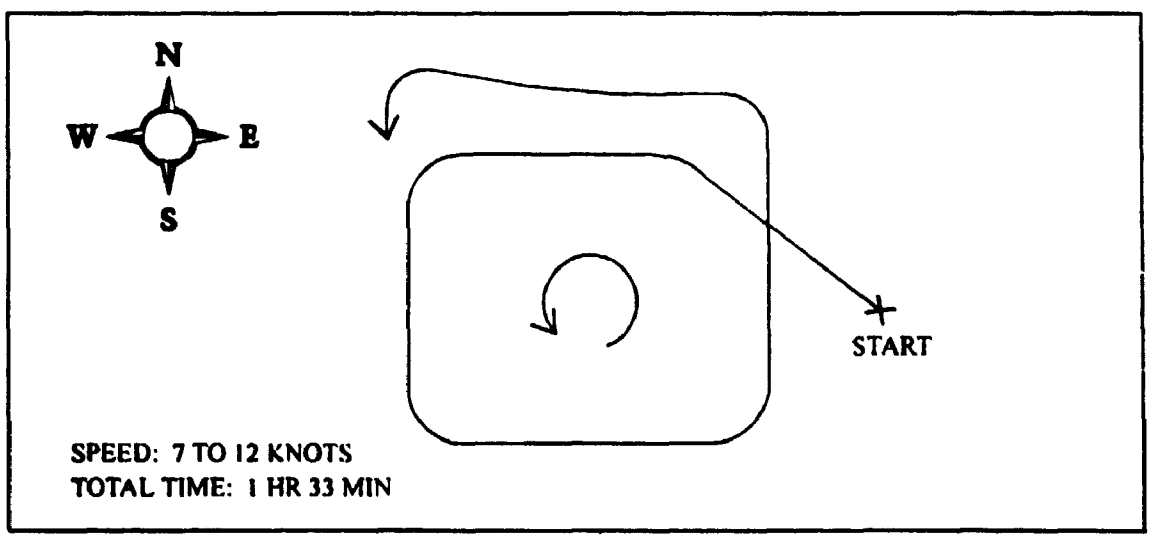


Figure 7.15: Box Manoeuvre executed during Sea Trial 94-01

The raw roll, pitch and yaw gyroscope outputs for the box manoeuvre are shown in Figures 7.16 and 7.17. The six noticeable abrupt changes in the yaw gyro rate indicate the ship is executing a clockwise turn. Furthermore, the turns are all in the same direction since they all cause positive increases in the gyro yaw rates. In comparison with the steady azimuth angular rate readings, the roll and pitch rates are significantly higher, roughly ± 2 deg/sec and ± 3 deg/sec, respectively. Consequently, a sampling rate of less than 1 Hz is insufficient for obtaining the navigation solution in the presence of such high roll and pitch dynamics. The tilt errors would only serve to incorrectly transform the accelerometer data.

The raw accelerometer data is presented in Figure 7.18. The excursions in the Y-Axis accelerometer data indicate that it senses the normal acceleration component when the ship is turning.

The Trimble Advanced Navigation Sensor (TANS) Vector GPS system provides the truth data necessary for assessing the results of the HRU navigation software. It is also used to initially align the HRU with attitude, heading and velocity information and to periodically correct the roll and pitch angles for the reasons given earlier. The GPS clock time stamp inserted during data logging is used to correlate the HRU and TANS Vector data files. Furthermore, the data files created by both systems are labelled using Julian day notation and Greenwich Mean Time (GMT).

The TANS Vector is a self-initializing, six-channel GPS receiver system that can calculate standard or differentially-corrected attitude and position information from satellite signals. For Sea Trial 94-01 it was operating in standard mode, hence using

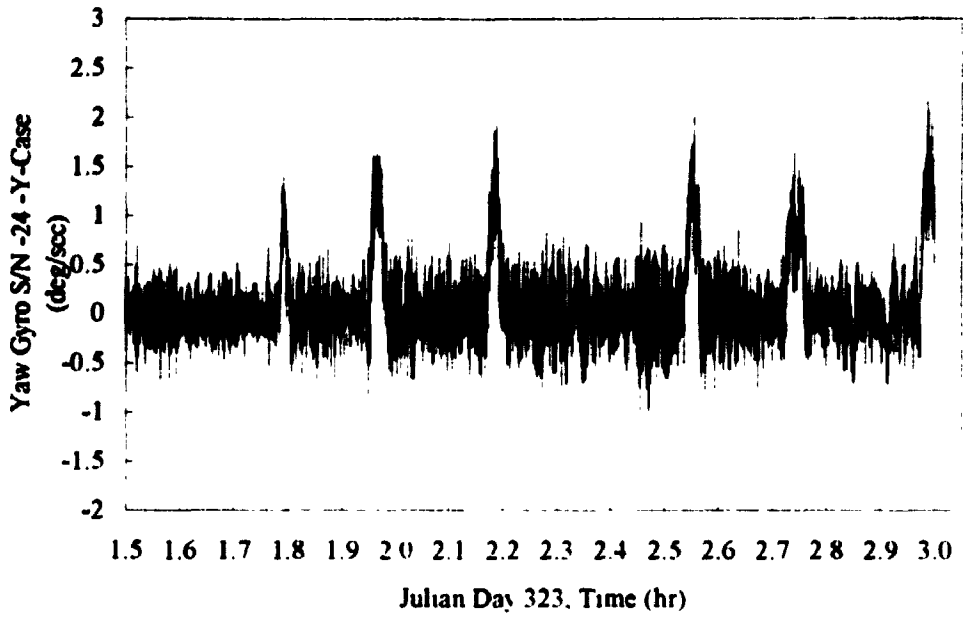


Figure 7.16: DIINS 94 Sea Trial, HRU Raw Yaw Gyro Data. Gyro S/N 24 Y-Case (Yaw) outputs for successive clockwise turns of CFAV Endeavour

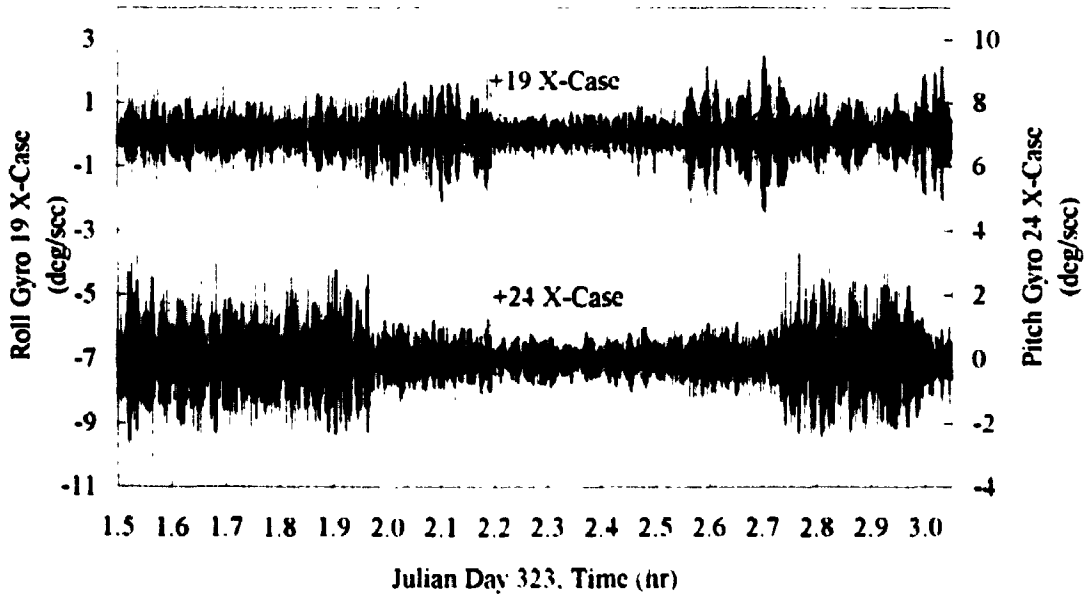


Figure 7.17: DIINS 94 Sea Trial, HRU Raw Attitude Gyro Data. Gyros S/N 19 X-Case (Roll), S/N 24 X-Case (Pitch) outputs for box manoeuvre.

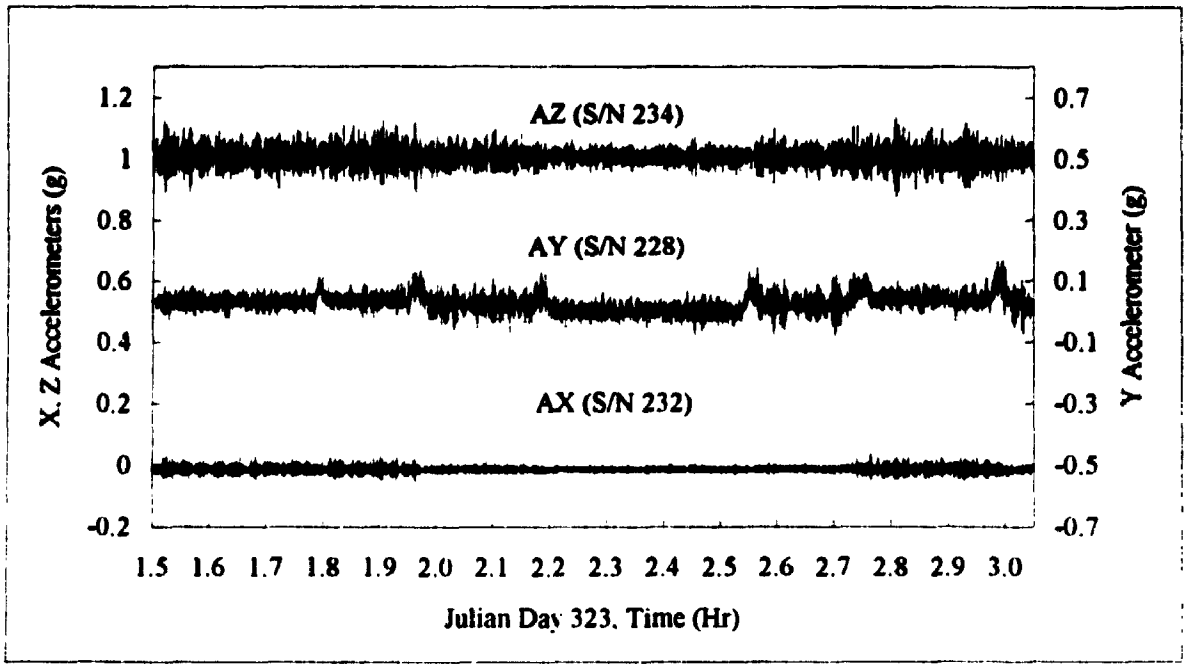


Figure 7.18: HRU DIINS 94 Sea Trial Raw Accelerometer Data. Data corresponds to a box manoeuvre executed by the CFAV Endeavour.

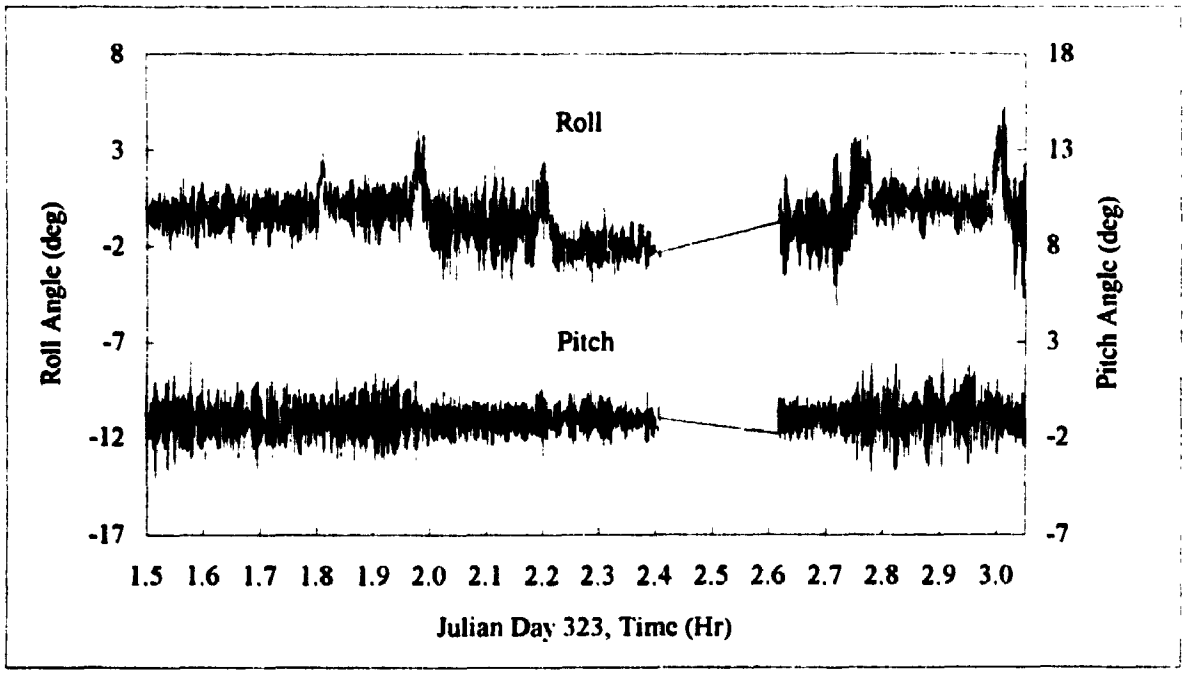


Figure 7.19: TANS Vector Data from DIINS 94 Sea Trial. Data was not available between 2.40 and 2.61 hours.

Ccarse/Acquisition C/A code on L1 frequency carrier for worldwide, all-weather operation. The L1 frequency carrier refers to the primary L-band signal radiated by each NAVSTAR satellite with centre frequency equal to 1575.42 MHz.

The data in the attitude file consists of time, speed, heading, roll, pitch and yaw angles, line biases and differential time. A separate file contains time-tagged, velocity, and X, Y, Z position information which can then be related to an ellipsoidal model of the earth, such as WGS-84. The time interval between successive data samples is approximately one second.

The TANS Vector system consists of one master and three slave antennas. The accuracy of the four-antenna system depends on the distance between the antennas. For Sea Trial 94-01 the antennas were installed to within 2 centimetres of 2 metre baselines. The baseline is defined as the largest distance between the master antenna and any of the slave antennas. The user specifications states an azimuth accuracy of 0.15 deg rms can be achieved with 2 metre antenna baselines [45]. For a 2 metre baseline the pitch/roll accuracy is 0.25 deg rms. The attitude solution improves as the distance between the antennas increases, however beyond a 2 metre solution the ability to resolve integers degrades. Figure 7.19 illustrates the roll and pitch data from the TANS Vector corresponding to the box manoeuvre. There is no data available for the 15 minute duration when the TANS Vector lost lock with the satellites.

It should be noted that the HRU and TANS Vector systems follow different conventions for defining heading. The TANS Vector defines azimuth angles between 0

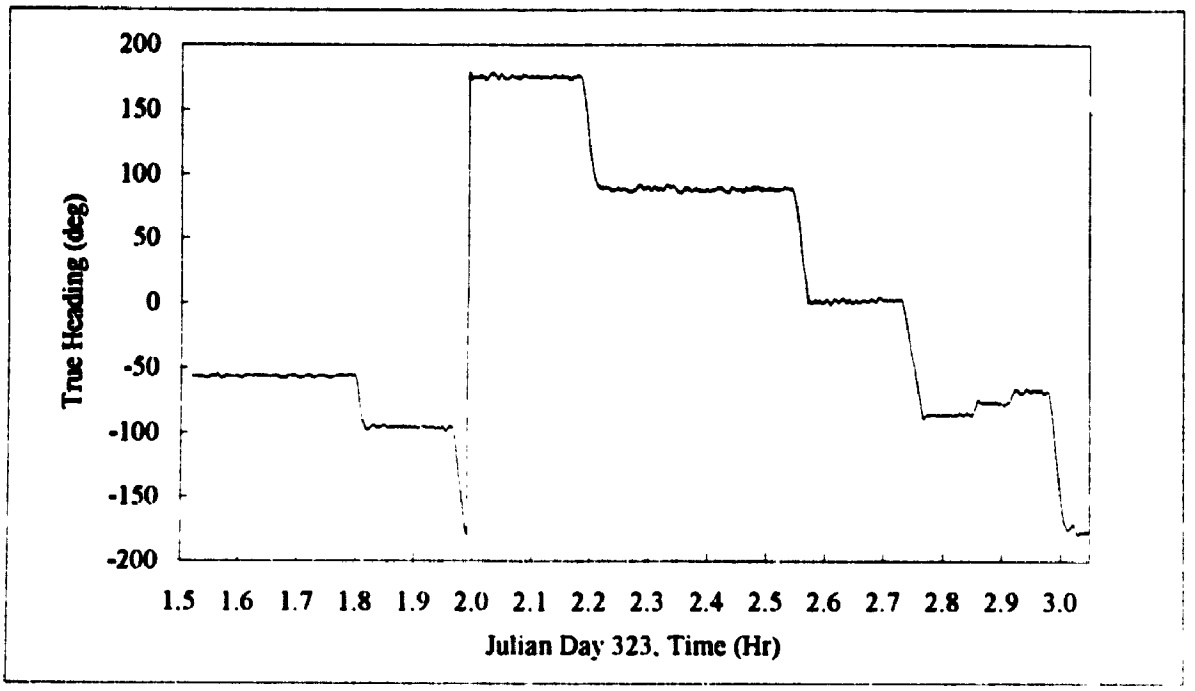


Figure 7.20: HRU Calculated True Heading. Navigation outputs computed from data collected during CFAV Endeavour's execution of a box manoeuvre.

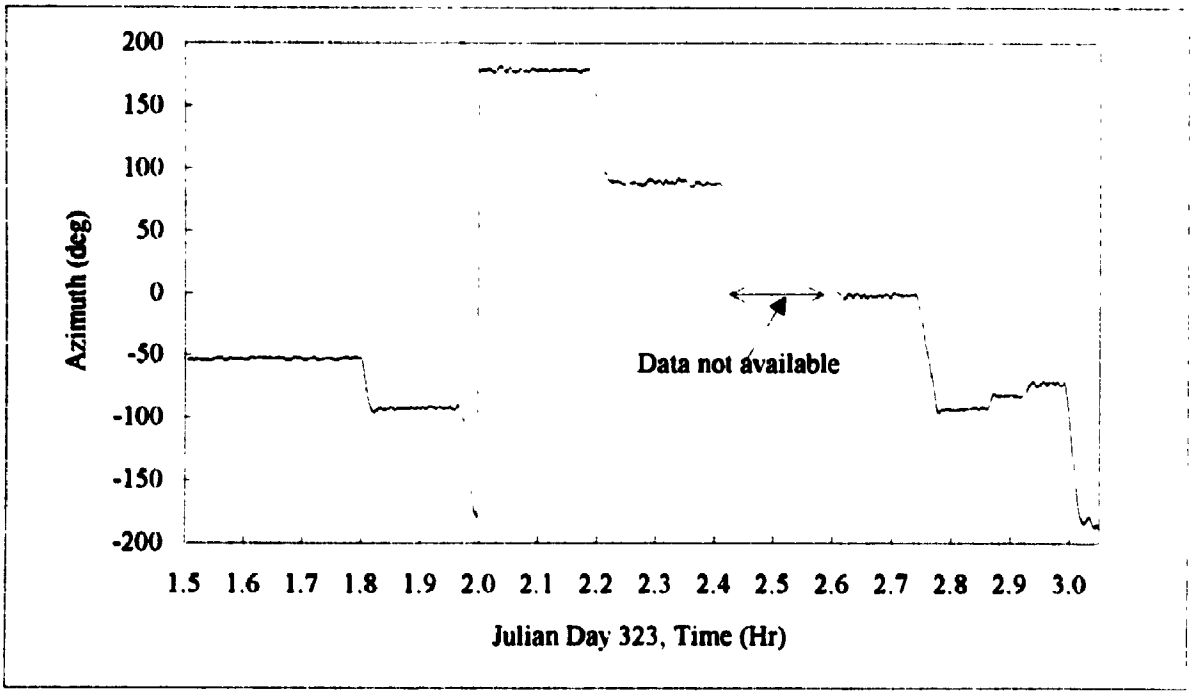


Figure 7.21: TANS Vector Azimuth Data. Data collected for box manoeuvre executed by CFAV Endeavour. Data was not available between 2.40 and 2.61 hrs.

and 360 degrees East (E) of North (N). In contrast, the HRU true heading output is between ±180 degrees. Positive angles refer to an orientation E of N whereas negative angles depict W of N. However, to facilitate comparison between the two systems, the TANS Vector azimuth output has been converted to the HRU convention of defining heading. The calculated HRU true heading and the raw TANS Vector azimuth outputs for the box manoeuvre executed during Sea Trial 94-01 can be compared in Figures 7.20 and 7.21. For a time span of 12 minutes the Vector lost lock with the satellites and consequently it did not supply azimuth information. However, as shown in Figure 7.20 the HRU heading outputs can be used to reconstruct the ship's heading information. Therefore, the benefit of an inertial navigation system such as the HRU becomes evident since this system is not subject to failures caused by insufficient satellite information, jamming in radio systems and the selective availability of GPS systems.

The segment during which the HRU heading angle changes from negative 90 to positive 180 degrees is examined in more detail. From the data files of computed headings for the HRU and azimuth outputs from the TANS Vector, the angular rates of turn in this region were computed. Based on these data files the TANS Vector and HRU angular rates of turn are found to be: 1.10 and 1.09 deg/sec respectively.

Thus, the post-processed HRU data and the reference TANS Vector data for a specific manoeuvre yield comparable, equivalent heading results.

Chapter 8

Concluding Remarks

8.1 Conclusions

The self-alignment and strapdown navigation algorithms have been designed and implemented for the DREO HRU. More specifically, the following tasks have been accomplished:

1. Alignment is the process of defining the orientation of the HRU system axes relative to geographic coordinates. A coarse alignment scheme that makes exclusive use of the inertial sensor outputs to achieve this purpose was derived as described in §5.4.1 Statistical data analysis techniques, presented in §5.4.2, are used to assess the success of the coarse alignment process for different operating environments. During alignment the system's initial position information is processed as described in §7.3.

2. A fine leveling and gyrocompassing stage was developed to refine the initial estimate of the attitude matrix obtained during coarse alignment. The error equations

governing the damped model for the analytic platform are derived in §5.5.1. The system has the distinct advantage that it can start navigating immediately after coarse alignment due to a fine alignment scheme that functions in the presence of vehicle motion.

- The parameters controlling error damping were obtained as the results of a tradeoff study (§5.5.2.5) for minimizing transient and steady-state level errors in the presence of bias instabilities (§5.5.2.2), constant velocities (§5.5.2.1), constant accelerations (§5.5.2.3) and manoeuvre accelerations (§5.5.2.4).
- The stability of the heading error in the presence of bias instabilities was investigated in §5.5.3.
- The fine alignment scheme postulated in Chapter 5 has been verified with the use of a series of initial error damping case studies as shown in §7.3.4.1.

3. The damped, local-level wander-azimuth, strapdown navigation algorithms were derived in Chapter 6. The attitude and heading information is obtained as explained in §6.3, ground velocity in §6.4 and position information in §6.5.

4. The outputs from the inertial sensors in the HRU package must be corrected for known bias, misalignment and scale factor terms. To this effect, a suitable data compensation algorithm was developed in §5.3.

5. The coarse alignment software implementation based on the algorithms formulated

in Chapter 5 is described in §7.2. A multiposition test of the HRU with the use of a two-axis Contraves motion table was undertaken for the purpose of testing the coarse alignment algorithms in §7.2.1.

6. The software implementation of the navigation algorithms is discussed in §7.3. A simulated test trajectory (§7.3.4.2) and experimental data obtained during Sea Trial 94-01 (§7.3.5) were used to validate the navigation algorithms.

8.2 Future Research

Suggestions for future research include:

1. Replace the crude gravity model used in §6.4 with a more accurate ellipsoidal model of the earth.

2. Inaccuracies in the gravity model have a direct impact on the altitude channel of a pure inertial navigation system. To instill stability in the calculation of the system height above the reference ellipsoid, a damped error model of the vertical channel should be developed.

3. The HRU alignment and navigation software can be implemented in a real-time system. This requirement hinges on the availability of simultaneous data collected at sampling rates of 400 Hz with the use of high-resolution ultra-precision digitizers.

4. For sufficiently high sampling rates the insertion of a pre-processor coning algorithm is feasible for improving the strapdown attitude realization (refer to §6.2).

5. Alignment and calibration of the inertial sensor error terms is needed to obtain an accurate measure of the misalignments and scale factors. Other error terms are the gyro bias drift and g-sensitive drift and accelerometer offset biases. These values are used in the data compensation algorithm (§5.3). Static multiposition testing is the method recommended for obtaining these error terms.

8.3 Summary of Contributions

1. Theoretical formulation and software implementation of the alignment and navigation algorithms were designed and developed specifically for the DREO HRU. In particular, the sensor inputs to the navigation algorithms are differential angular velocity inputs as opposed to incremental angular changes over a specific period of time.

2. Replacing the integrator in a normal Schuler loop with a low pass filter is an approach that was suggested previously by [31] to damp errors in a vertical control loop. This method is extended here to provide the analytic gyrocompassing and fine leveling of a strapdown AHRS. The gain in the azimuth applied precession rate term is adjusted to damp any initial heading errors.

3. The fine alignment algorithms are distinct since they function in the presence of vehicle motion, and do not depend on the execution of a specific manoeuvre to attain the required performance. Furthermore, they do not require external references.

4. A data compensation algorithm which corrects the raw sensor outputs for known error terms was derived in §5.3. The raw gyro and accelerometer data is transformed from body coordinates to angular rates and linear acceleration components along the system axes.

5. For a more detailed explanation of the above contributions refer to §4.4, Thesis Justification.

References

[1] Honeywell, "H-423 System Description", Honeywell Military, Avionics Division, Clearwater, Florida.

[2] Honeywell, "H-764 Small Inertial Navigation System for Multiple Applications", Honeywell Military, Avionics Division, St. Petersburg, Florida. 27 September 1991.

[3] Siouris, George M., Aerospace Avionics Systems, A Modern Synthesis, Academic Press Inc., San Diego, California, USA, 1993.

[4] Siouris, George M., "Investigation of a Strapdown Attitude and Heading Reference System Using Ring Laser Gyros", AGARD CP 298, Precision Positioning and Inertial Guidance Sensors Technology and Operational Aspects, France, 14-17 Oct., 1980.

[5] DREO (internal document), "HRU Requirements Specification", 19 March, 1990.

[6] Farrell, James L., Integrated Aircraft Navigation, Academic Press, Inc., New York, USA, 1976.

[7] Lawrence, Anthony., Modern Inertial Technology Navigation, Guidance, and Control, Springer-Verlag Inc., New York, USA, 1993.

[8] Savage, Paul G., "Laser Gyros in Strapdown Inertial Navigation Systems", Honeywell Government and Aeronautical Products Division, IEEE Position Location and Navigation Symposium, San Diego, California, USA, November 1-3, 1976.

[9] Richardson, K.I.T., The Gyroscope Applied, Hutchinson's Scientific and Technical Publications, London, Great Britain, 1954

[10] Vinnins, M.F., Gallop, L.D., Paquet., "Test Evaluation of the Litton (Canada) Canadian Strapdown Gyroscope (CSG-2)", DREO Report No. 914, Ottawa, Canada, September 1986.

[11] Vinnins, M.F., Gallop, L.D., "Analysis of Rate and Dynamic Tests on the Litton (Canada) Canadian Strapdown Gyroscope (CSG-2)", DREO Technical Note 88-20, Ottawa, Canada, April 1988.

[12] Vinnins, M.F., Gallop, L.D., "Test and Evaluation of the Litton (Canada) Canadian Strapdown Gyroscope (CSG-2) Performance Evaluation Unit (PEU)", DREO Report No. 1010, Ottawa, Canada, September 1988.

[13] Pearson, B., "Engineering Specification: Gyroscope, Strapdown (CSG-2B) SK10-6007", Proprietary Document, Litton Systems Canada, Limited., Rexdale, Ontario, Canada, June 1985.

[14] Leondes, C.T. (editor), Guidance and Control of Aerospace Vehicles, McGraw-Hill Book Company, Inc., New York, USA, 1963.

[15] Peters, Rex B., "Computer-Automated Characterization of a High Production Volume, Inertial Grade Accelerometer", Sundstrand Data Control, Inc., Washington, USA, 1982.

[16] Fejes, Thomas G., "HRU Preliminary Software Design", (internal document), deliverable 3.4.1, TINORU, Ottawa, Canada, Dec. 1994.

[17] Fejes, Thomas G., "HRU Preliminary Hardware Design", (internal document), deliverable 3.4.2, TINORU, Ottawa, Canada, Dec 1994.

[18] Den Hartog, J.P.,Mechanics, McGraw-Hill Book Company, Inc., New York, USA, 1948.

[19] Priel, Beni., "The Effect of Gyro Nonorthogonality Error on Gyrocompassing", IEEE Transactions on Aerospace and Electronic Systems, Vol. 28, No.3., July 1992.

[20] Hung, J.C., White, H.V., "Self-Alignment Techniques for Inertial Measurement Units", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-11, No.6., November 1975.

[21] Goshen-Meskin. Bar-Itzhack, I.Y., "Observability Analysis of Inertial Navigation Sytems During In-Flight Alignment", AIAA Guidance, Navigation and Control Conference, Minneapolis, Minnesota, August 15-17, 1988.

[22] Britting, Kenneth R., Inertial Navigation Systems Analysis, John Wiley & Sons, Inc., Toronto, Canada, 1971.

[23] Raytheon Company Equipment Division, Correspondence: "Section 4: Calibration Techniques.", Sudbury, Massachusetts, USA.

[24] Fay, Stanley., "Leveling of Analytic Platforms", IEEE Transactions on Aerospace and Electronics Systems, Vol. AES-5, No. 1, January 1969.

[25] Jurenka, Frank D., Leondes, Cornelius T., "Optimum Alignment of an Inertial Autonavigator", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-3, No. 6, November 1967.

[26] Huddle, James, R., "Inertial Navigation System Error Model Considerations in Kalman Filter Applications", in Advances in the Techniques and Technology of the Application of Nonlinear Filters and Kalman Filters, AGARD-AG-256, Chapter 13, California, USA, March 1982.

[27] Grammatikos, A., Schuler, A.R., Fegley, K.A., "Damping Gimballess Inertial Navigation Systems", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-3, No. 3., May 1967.

[28] Grubin, Carl., "Attitude Determination for a Strapdown Inertial System Using the Euler Axis/Angle and Quaternion Parameters", Paper No. 73-900, AIAA Guid. ce and Control Conference, Key Biscayne, Florida, August 20-22, 1973.

[29] Bar-Itzhack, Itzhack Y , " Navigation Computation in Terrestrial Strapdown Inertial Navigation Systems", IEEE Transactions on Aerospace and Electronic Systems", Vol. AES-13, No.6, November 1977.

[30] Goguen, J. P. Thomas, "DREO NavLab HP-AHRS, Hardware Definition", (internal document) DSS Contract Number W7714-9559/01-ST, CARO Engineering, Ottawa, Canada, April 30, 1991.

[31] Staley, D.A., "ANCON-R.829 Analysis of a Stable Platform Vertical Control Loop (Background Information)", ANCON Space Technology Corporation, Ontario, Canada, September 1984

[32] Bolduc, Carole R.M., "Alignment Calibration of an Attitude and Heading Reference System", DREO (internal document), Ottawa, Canada, Sept 1,1993.

[33] Staley, Douglas A., "HRU Alignment Calibration", ANCON-R.860, DSS Contract File No. 003ST.W7714-2-9631 (internal document), Ottawa, Canada, March 1993.

[34] Bowker, Albert H., Lieberman, Gerald J., Engineering Statistics, 2nd Edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.

[35] Ryan, Timothy J., "Alignment and Calibration of a Strapdown Inertial Measurement Unit", Measurement Systems Laboratory, MIT, Cambridge, Massachusetts, December, 1970.

[36] Franklin, Gene F., Powell, David J., Emami-Naeini, Abbas., Feedback Control of Dynamic Systems, Addison-Wesley Publishing Company, New York, USA, 1991.

[37] Bar-Itzhack, I.Y., Meyer, J., Fuhrmann, P.A., "Strapdown Matrix Orthogonalization: The Dual Iterative Algorithm", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-12, No. 1, January 1976.

[38] Jiang, Yeon Fuh., Lin, Yu Ping., "Improved Strapdown Coning Algorithms", IEEE Transactions on Aerospace and Electronic Systems, Vol. 28., No.2, April 1992.

[39] Lee, J.G., Yoon, Y.J., "An Extension of Strapdown Attitude Algorithm for High-Frequency Base Motion", AIAA Guidance, Navigation and Control Conference, Minneapolis, Minnesota,L USA., August 15-17, 1988.

[40] Meriam, J.L., Kraige, L.G., Engineering Mechanics Volume 2 Dynamics, John Wiley and Sons, Toronto, Canada, 1986.

[41] Borland International, Inc., Turbo C++ Version 3.0 User's Guide, Borland International, Inc., California, USA, 1992.

[42] Bell, Doug., Morrey, Ian., Pugh, John, Software Engineering A Programming Approach, Prentice Hall International (UK) Ltd., Hertdfordshire, UK, 1992.

[43] Press, William H., et al., Numerical Recipes The Art of Scientific Computing, Cambridge University Press, USA, 1986.

[44] Harding, Louis Allen, A Brief History of the Art of Navigation, The William-Frederick Press, New York, USA, 1952.

[45] Trimble Navigation Limited, "TANS VECTOR GPS Attitude Determination System Specification and User's Manual", Trimble Navigation Limited, Sunnyvale, California, USA, May 1994.

[46] Noble, Ben., Daniel, James W., Applied Linear Algebra, 3rd edition, Prentice-Hall, A Division of Simon & Schuster, Englewood Cliffs, New Jersey, USA, 1988.

[47] Boyce, William E., Diprima, Richard C., Elementary Differential Equations and Boundary Value Problems, John Wiley & Sons, Inc., New York, USA, 1986.

Appendix A

Mathematical Details

A.1 Matrices

1) *Three Dimensional Rotation Matrices*:

The change of coordinates by performing a rotation about an axis. [3]

Z-Axis rotation ψ Y-Axis rotation, θ X-Axis Rotation, ϕ

$$\begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix}$$

The elements of the transformation matrices represent the direction cosines from the transformed to the original axes. (Note: C and S are short forms for cos and sin, respectively.)

2) *Symmetric and skew-symmetric matrices* [46]:

$$\mathbf{A}^T = \mathbf{A} \qquad \text{symmetric}$$

$$\mathbf{A}^T = -\mathbf{A} \qquad \text{skew-symmetric}$$

3) *Time derivative of direction cosine matrix,* **C**, [6]:

$$\frac{d}{dt}(\mathbf{C}) = \mathbf{\Omega} \cdot \mathbf{C}$$

where $\mathbf{\Omega}$ is a skew-symmetric matrix representing the relative angular rate between the original and transformed frames of reference, in the coordinates of the original reference frame.

4) *Time derivative of direction cosine matrix for two frames of reference*, given angular velocities relative to an absolute frame of reference. [6]:

$$\dot{\mathbf{C}}_b^a = -\mathbf{C}_b^a \cdot \mathbf{\Omega}_b + \mathbf{\Omega}_a \cdot \mathbf{C}_b^a$$

where $\mathbf{\Omega}_b$ and $\mathbf{\Omega}_a$ are the angular velocities of coordinate frames 'b' and 'a' relative to inertial space, respectively.

5) *Similarity Tranformation*:

P and **A** are similar matrices if a non-singular matrix **T** exists such that [22]:

$$\mathbf{P} = \mathbf{T} \cdot \mathbf{A} \cdot \mathbf{T}^{-1}$$

Similar matrices have the same eigenvalues. Also, if **P** is similar to **A** then det **A** = det **B**, where the determinant is defined as:

$$\det \mathbf{A} = \sum (-1)^k a_{1j_1} \cdot a_{2j_2} \cdots a_{nj_n} \text{ [46]}$$

Similarity transformations can be used to express equations in another frame of reference, for example:

$$\mathbf{A} \cdot \vec{b}_b = \vec{c}_b$$

can be rewritten as: $$\mathbf{C}_b^i \cdot \mathbf{A} \cdot \mathbf{C}_i^b \cdot \vec{b}_i = \vec{c}_i$$

A.2 Numerical Methods

The classical fourth-order Runge-Kutta method is used to solve the 'Initial Value Problem' [47]:

$$\dot{y} = f(x, y) \qquad y(x_o) = y_o$$

The solution involves taking weighted average values of the function at different points in the interval, $x_n \le x \le x_{n+1}$. The variable 'h' is the step size.

$$y_{n+1} = y_n + \frac{h}{6} \cdot \left(k_{n1} + 2 \cdot k_{n2} + 2 \cdot k_{n3} + k_{n4}\right)$$

$$k_{n1} = f(x_n, y_n)$$
$$k_{n2} = f(x_n + \frac{h}{2}, y_n + \frac{h}{2} \cdot k_{n1})$$
$$k_{n3} = f(x_n + \frac{h}{2}, y_n + \frac{h}{2} \cdot k_{n2})$$
$$k_{n4} = f(x_n + h, y_n + h \cdot k_{n3})$$

Appendix B

Sample Output Files

B.1 Coarse Alignment

COARSE ALIGNMENT

Source filename: h2721415 (Position 1)
Latitude: 45.348333926
Maximum Gyro Noise Term (deg/hr): 0.4
Maximum Accelerometer Noise Term (g's): 0.00010

GYROSCOPE AND ACCELEROMETER RAW DATA (deg/sec and g's)

GXmean:	0.0020380	GYmean:	0.0013194	GZmean:	0.0030808
AXmean:	0.0029241	AYmean:	0.0004292	AZmean:	1.0090711

INTERMEDIATE RESULTS (deg/hr)

Align Time (min): 20.00

Mean, Standard Deviation, and Confidence Intervals (deg/hr and g's)

GYRO X :	-7.54176	GYRO Y:	-8.08375	GYRO Z:	10.76877
GSD X :	5.45640	GSD Y:	6.34687	GSD Z:	9.22497
GCI X :	0.39980	GCI Y:	0.39972	GCI Z:	0.52248

ACCEL. X:	0.00000	ACCEL. Y:	-0.00001	ACCEL. Z:	1.00004
ASD X :	0.00020	ASD Y :	0.00019	ASD Z :	0.00014
ACI X :	0.00003	ACI Y :	0.00003	ACI Z :	0.00002

FINAL RESULTS

BODY MATRIX TRANSPOSE

Row: 0 Col: 0 Element: -0.48866
Row: 0 Col: 1 Element: -0.52378
Row: 0 Col: 2 Element: 0.69776
Row: 1 Col: 0 Element: -0.73119
Row: 1 Col: 1 Element: 0.68218
Row: 1 Col: 2 Element: 0.00001
Row: 2 Col: 0 Element: -0.47600
Row: 2 Col: 1 Element: -0.51019
Row: 2 Col: 2 Element: -0.71634

DIRECTION COSINE MATRIX

Row: 0 Col: 0 Element: -0.73119
Row: 0 Col: 1 Element: 0.68218
Row: 0 Col: 2 Element: 0.00001
Row: 1 Col: 0 Element: -0.68205
Row: 1 Col. 1 Element: -0.73105
Row: 1 Col: 2 Element: -0.01922
Row: 2 Col: 0 Element: -0.01310
Row: 2 Col: 1 Element: -0.01406
Row: 2 Col: 2 Element: 0.99982

B.2 Navigation

SAMPLE NAVIGATION SOFTWARE OUTPUT

T	Roll	Pitch	Yaw	Lat	Long	Wander	GV
1	-0.540	-0.241	-56.479	48.00003	-123.00000	0.00000	11.030
2	-0.534	-0.261	-56.288	48.00007	-123.00001	0.00000	11.166
3	-0.533	-0.284	-56.305	48.00010	-123.00001	0.00001	10.894
4	-0.544	-0.316	-56.220	48.00013	-123.00001	0.00001	11.106
5	-0.545	-0.341	-56.244	48.00017	-123.00001	0.00001	11.951
6	-0.528	-0.354	-56.310	48.00020	-123.00002	0.00001	11.409
7	-0.520	-0.366	-56.028	48.00023	-123.00002	0.00001	9.980
8	-0.527	-0.389	-55.909	48.00027	-123.00002	0.00001	10.255
9	-0.531	-0.419	-56.036	48.00030	-123.00002	0.00001	11.304
10	-0.518	-0.442	-56.053	48.00034	-123.00002	0.00002	11.677
11	-0.542	-0.239	-56.205	48.00037	-123.00002	0.00002	10.633
12	-0.544	-0.267	-55.991	48.00040	-123.00003	0.00002	10.921
13	-0.547	-0.297	-56.109	48.00044	-123.00003	0.00002	11.943
14	-0.550	-0.323	-56.131	48.00047	-123.00003	0.00003	11.663
15	-0.535	-0.334	-56.141	48.00051	-123.00004	0.00003	11.704
16	-0.525	-0.346	-56.109	48.00054	-123.00004	0.00003	10.168
17	-0.531	-0.371	-56.142	48.00057	-123.00004	0.00003	10.767
18	-0.537	-0.398	-56.418	48.00060	-123.00004	0.00003	11.060
19	-0.539	-0.422	-56.454	48.00064	-123.00004	0.00003	10.991
20	-0.527	-0.440	-56.373	48.00067	-123.00004	0.00003	11.198
21	-0.540	-0.240	-56.215	48.00070	-123.00004	0.00003	10.379
22	-0.538	-0.268	-56.259	48.00074	-123.00005	0.00004	11.718
23	-0.542	-0.294	-56.550	48.00077	-123.00005	0.00004	11.420
24	-0.545	-0.317	-56.505	48.00081	-123.00005	0.00004	10.857
25	-0.533	-0.339	-56.551	48.00084	-123.00006	0.00004	11.833
26	-0.521	-0.357	-56.607	48.00087	-123.00006	0.00005	11.341
27	-0.533	-0.378	-56.722	48.00091	-123.00006	0.00005	10.872
28	-0.548	-0.399	-57.088	48.00094	-123.00006	0.00005	10.896
29	-0.534	-0.414	-56.755	48.00097	-123.00006	0.00005	10.606
30	-0.508	-0.435	-56.357	48.00101	-123.00006	0.00005	10.885
31	-0.543	-0.246	-56.386	48.00104	-123.00007	0.00005	10.830
32	-0.560	-0.273	-56.683	48.00107	-123.00007	0.00005	10.872
33	-0.571	-0.294	-57.007	48.00111	-123.00007	0.00005	11.335
34	-0.552	-0.314	-56.841	48.00114	-123.00007	0.00005	10.974
35	-0.535	-0.342	-56.527	48.00117	-123.00008	0.00006	11.148

Note: Roll, Pitch, Yaw, Lat, Long, Wander Angle (in degrees), Ground Velocity (kph)
T (refers to sample number in this case since this file corresponds to sea trial data collected every 1.3 to 1.5 seconds)

Appendix C

Software Source Code Listings

C.1 Coarse Alignment, CAV13.CPP

```
/*****************************************************************************
*                                                                            *
*      COARSE ALIGNMENT SOFTWARE FOR THE DREO HRU                            *
*                                                                            *
*      Filename:  CAV13.CPP                                                  *
*      Author.    Carole R.M. Bolduc                                         *
*      Date.      Nov 29, 1994                                               *
*      Version:   1.3                                                        *
*                                                                            *
*      Requires.  Two input files, containing raw HRU gyroscope and          *
*                 accelerometer data (HRU_ACC.out and HRU_GYR.out,respectively).*
*                 Files are originally generated by HRUCDU V1.50, raw data is*
*                 then time aligned to give a sample rate of exactly 1 Hz with*
*                 polynomial interpolator, CUBIC V1.0                        *
*                                                                            *
*      Description:    This module performs the following operations         *
*                                                                            *
*        1.    Prompts the operator for the initial latitude in degrees     *
*        2.    Computes the Geographic Align Matrix (3X3).                  *
*        3.    Reads in gyroscope and accelerometer data and                *
*              averages the results until the standard deviation of the     *
*              mean is below a specified tolerance                          *
*        4.    Compute row 1 of [B] transpose, normalize angular rates.     *
*        5.    Compute row 2 of [B] transpose, cross product of angular     *
*              rates and measured accelerations, normalize the result.      *
*        6.    Compute row 3 of [B] transpose, cross product of angular     *
*              rates with row 2 results, normalize the result.              *
*        7.    Matrix multiplication of two matrices: [G]*transpose[B].     *
```

```
*               Result is the 3X3 Direction cosine matrix (body to geographic).      *
*        8.     Prints results to file and display                                   *
*                                                                                    *
*      Outputs:     Raw data averages, angular rates and accelerations along        *
*                   East, North, Up coordinates, statistical results (standard      *
*                   deviations, confidence intervals), Body matrix transpose,       *
*                   Direction Cosine Matrix (body to geographic coordinates)        *
*                                                                                    *
*************************************************************************************/

#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <math.h>
#include <string.h>
#include <fcntl.h>
#include <io.h>
#include <dos.h>

// gyroscope bias drifts in deg/hr
#define     GB24x  -2.6698
#define     GB24y  0.84124
#define     GB19x  -7.9482
#define     GB19y  6.7845

// accelerometer biases in g
#define     AB232x 0.00292658083
#define     AB228y 0.0004389546
#define     AB234z 0.009026187

/*    Scale factors (raw data includes scale factor correction)     */
#define     K24x   1              // Gyroscopes
#define     K24y   1
#define     K19x   1
#define     K19y   1

#define     K232x  1              // Accelerometers
#define     K228y  1
#define     K234z  1

/*    Gyroscope and accelerometer matrix misalignment angles     */
#define     Gyx    0
#define     Gzx    0
#define     Gxy    0
#define     Gzy    0
#define     Gxz    0
#define     Gyz    0
```

```
#define     Ayx   0
#define     Azx   0
#define     Axy   0
#define     Azy   0
#define     Axz   0
#define     Ayz   0

/* Gyroscope Mass Unbalance and Quadrature Terms from Litton test data (deg/hr/g) */
#define     MU19x  1.191
#define     Q19x   7.743
#define     MU19y  1.233
#define     Q19y   7.680
#define     MU24x  -3.0981
#define     Q24x   -0.6642
#define     MU24y  -0.5190
#define     Q24y   -2.9654

double      GM[3][3]={1,Gyx,Gzx,Gxy,1,Gzy,Gxz,Gyz,1};
double      AM[3][3]={1,Ayx,Azx,Axy,1,Azy,Axz,Ayz,1};

double  lat;
char    lat_str[80],accelx_str[80],accely_str[80],accelz_str[80];
char    other_str[80],gyrox_str[80],gyroy_str[80],gyroz_str[80],gnoise_str[80],f_name[80];
double  ax,ay,az,wx,wy,wz,norm;                 // accelerometer and gyro data, norm
double  AXSD=0,AYSD=0,AZSD=0,GXSD=0,GYSD=0,GZSD=0;                    // SDEV vars
double  AXsum=0,AYsum=0,AZsum=0,AXsumprev,AYsumprev,AZsumprev;        // mean vars
double  AXmean=0,AYmean=0,AZmean=0,GXmean=0,GYmean=0,GZmean=0;
double  accelx=0,accely=0,accelz=0,deltax,deltay,deltaz,
double  paccelx,paccely,paccelz;                                      // previous value
double  sampleN=0,AXCI,AYCI,AZCI,GXCI,GYCI,GZCI;                      // Confidence Interval
double  gxrnd,gyrnd,gzrnd,                                            // Gyro noise term

int     i,j,k;                                                        // counter variables
int     handlea,handleb,

const   double  pi=3.1415926536;
const   N=3;                            // matrix dimension
const   MAX=1200;                       // t convergence must be reached before 20 minutes
const   headpos=350,                    // File position
const   double  anoise=0.00010,         // accelerometer noise factor

// [G] and [B] matrices 3 rows, 3 columns
double  G[3][3] ={0,1,0,0,0,0,0,0,0},   // initialized (row1,row2...)
double  BT[3][3];                       // transpose of [B]
double  C[3][3];                        // Direction Cosine Matrix
double  omega[3], *w;                   // Mean values
double  beta[3], *b;
double  DeltaX,DeltaY,DeltaZ;           // SDEV gyro variable
double  gyrox=0,gyroy=0,gyroz=0;        // data sample

float   align_time;
```

```
int     true=1, false=0,done=0;                    // boolean variables
                                                   // CI<Noise condition
int     AXtcond=0,AYtcond=0,AZtcond=0,GXtcond=0,GYtcond=0,GZtcond=0;
char    sequence[80];

FILE *in_accel;                                    // input accelerometer data file
FILE *in_gyro;                                     / input gyroscope data file
FILE *results;                                     // output file

void GyroComp(double *omegac,double *betac);
void AccelComp(double *betac);

void main(void)
{
        clrscr();
        textmode(C80);
        window(1,1,80,2);
        textcolor(YELLOW);
        cprintf ("                    COARSE ALIGNMENT");
        textcolor(CYAN);
        window(1,3,80,15);
        cprintf("Enter the output filename:");
        gets(f_name);
        results=fopen(f_name,"w");
        fprintf (results,"                    COARSE ALIGNMENT\n\n");
        cprintf("Enter the source filename:");
        gets(f_name);
        fprintf(results,"Source filename  :  %s\n",f_name);
        cprintf ("Enter the present latitude in degrees:");
        gets(lat_str);
        fprintf(results,"Latitude:  %s\n",lat_str);
        lat = atof(lat_str);                       // converts string to double format
        lat = lat*pi/180;                          // converts to radians
        cprintf("Enter the gyro noise term in degrees/hr:");
        gets(gnoise_str);
        fprintf(results,"Maximum Gyro Noise Term (deg/hr):  %s\n",gnoise_str);
        fprintf(results,"Maximum Accelerometer Noise Term (g's):  %8.5f",anoise);
        gxrnd=atof(gnoise_str)/3600;               // convert to deg/sec
        gyrnd=gxrnd;                               // gyro random noise term
        gzrnd=gxrnd;

        w = omega;        // w is set to the address of the first array element
                          // of omega, initialize the pointer
        b = beta;

        /* Initialize [G] matrix */
        G[1][0]=cos(lat);
        G[1][2]=sin(lat);
        G[2][0]=sin(lat);
        G[2][2]=-cos(lat);
```

```
/* Open files */
if (_dos_open("c:\\hru_acc.out",O_RDONLY|O_TEXT,&handlea)!=0){
        perror("Unable to open");
        }
in_accel=fdopen(handlea,"r");
if (in_accel==NULL)
        printf("fdopen accel failed \n");
if (_dos_open("c:\\hru_gyr.out",O_RDONLY|O_TEXT,&handleb)!=0){
        perror("Unable to open");
        }
in_gyro=fdopen(handleb,"r");
if (in_gyro== NULL)
        printf("fdopen gyro failed \n");

/* Read and discard the header */
do fgets(sequence,18,in_accel);
while (strcmp(sequence,"Sample    Elapsed"));
fseek(in_accel,headpos,SEEK_CUR); // locate first data point
do fgets(sequence,18,in_gyro);
while (strcmp(sequence,"Sample    Elapsed"));
fseek(in_gyro,460,SEEK_CUR);

while (!((AXtcond & AYtcond & AZtcond & GXtcond & GYtcond & GZtcond) |
        (sampleN>=MAX)))
{
 if (sampleN==0)          /* Beginning of file? */
                          /* Discard first five measurements due to inaccuracies in the cubic
                             spline interpolation at the start and end of file */
  {
   for (i=0;i<=4;i++)
    {
    fscanf(in_accel,"%8s %14s %14s %14s %14s %14s
        %14s",other_str,other_str,accelx_str,other_str,accely_str,other_str,accelz_str);
    fscanf(in_gyro,"%8s %14s %14s %14s %14s %14s %14s %14s
        %14s",other_str,other_str,other_str,other_str,gyrox_str,other_str,
        gyroz_str,other_str,gyroy_str);
    }
   /* Read first sample of accelerometer X, Y, Z data */
   if (fscanf(in_accel,"%8s %14s %14s %14s %14s %14s
        %14s",other_str,other_str,accelx_str,other_str,accely_str,other_str,accelz_str))
    {
        accelx=atof(accelx_str);
        accely=atof(accely_str);
        accelz=atof(accelz_str);
        }
   else
        cprintf("File Read Error\n");

   /* Read first sample of gyroscope X, Y, Z data */
   if (fscanf(in_gyro,"%8s %14s %14s %14s %14s %14s %14s %14s
        %14s",other_str,other_str,other_str,other_str,gyrox_str,other_str,
```

```
        gyroz_str,other_str,gyroy_str))
    {
        gyrox=atof(gyrox_str);
        gyroy=atof(gyroy_str);
        gyroz=atof(gyroz_str);
    }
  else
        cprintf("File Read Error\n");
    }

else //{if-else SampleN==0}
 {
  /* Read next sample of accelerometer X, Y, Z data */
  switch (fscanf(in_accel,"%8s %14s %14s %14s %14s %14s
        %14s",other_str,other_str,accelx_str,other_str,accely_str,other_str,accelz_str))
   {
   case '0': cprintf("File Read Error\n");break;
   case EOF:  cprintf("End of File Reached\n");    // Unable to align
                done=true;
                break;
   default:
        accelx=atof(accelx_str);
        accely=atof(accely_str);
        accelz=atof(accelz_str);
   }

  /* Read next sample of gyroscope X, Y, Z data */
  switch (fscanf(in_gyro,"%8s %14s %14s %14s %14s %14s %14s %14s
        %14s",other_str,other_str,other_str,other_str,gyrox_str,other_str,gyroz_str,
        other_str,gyroy_str))
   {
   case '0': cprintf("File Read Error\n");break;
   case EOF:  cprintf("End of File Reached\n");
                done=true;
                break;
   default:
        gyrox=atof(gyrox_str);
        gyroy=atof(gyroy_str);
        gyroz=atof(gyroz_str);
   }
 } //{end if-else SampleN==0}

 ++sampleN;                                  //Increment sample number

 /* Sequential estimation of the mean */
 AXmean=(sampleN-1)/sampleN * AXmean + accelx/sampleN;
 AYmean=(sampleN-1)/sampleN * AYmean + accely/sampleN;
 AZmean=(sampleN-1)/sampleN * AZmean + accelz/sampleN;
 GXmean=(sampleN-1)/sampleN * GXmean + gyrox/sampleN;
 GYmean=(sampleN-1)/sampleN * GYmean + gyroy/sampleN;
 GZmean=(sampleN-1)/sampleN * GZmean + gyroz/sampleN;
```

```
/* Sequential estimation of the standard deviation; Sample standard deviations are
   multiplied by sqrt(n-1). Thus, the equations below represent only the numerator
   of the std dev */
if (sampleN >1)
{
        if (AXtcond !=true)                    // CI<noise condition satified?
        {
         deltax+=-AXmean;                      // old mean-new mean
         AXSD=sqrt((sampleN-2)/(sampleN-1)*AXSD*AXSD+deltax*deltax+
                (accelx-AXmean)*(accelx-AXmean)/(sampleN-1));
         deltax=AXmean;                        // set to past value of mean
        }
        if (AYtcond !=true)
        {
         deltay+=-AYmean;
         AYSD=sqrt((sampleN-2)/(sampleN-1)*AYSD*AYSD+deltay*deltay+
                (accely-AYmean)*(accely-AYmean)/(sampleN-1));
         deltay=AYmean;
        }
        if (AZtcond !=true)
        {
         deltaz+=-AZmean;
         AZSD=sqrt((sampleN-2)/(sampleN-1)*AZSD*AZSD+deltaz*deltaz+
                (accelz-AZmean)*(accelz-AZmean)/(sampleN-1));
         deltaz=AZmean;
        }

        if (GXtcond !=true)
        {
         DeltaX+=-GXmean;
         GXSD=sqrt((sampleN-2)/(sampleN-1)*GXSD*GXSD+DeltaX*DeltaX+
                (gyrox-GXmean)*(gyrox-GXmean)/(sampleN-1));
         DeltaX=GXmean;
        }
        if (GYtcond !=true)
        {
         DeltaY+=-GYmean;
         GYSD=sqrt((sampleN-2)/(sampleN-1)*GYSD*GYSD+DeltaY*DeltaY+
                (gyroy-GYmean)*(gyroy-GYmean)/(sampleN-1));
         DeltaY=GYmean;
        }
        if (GZtcond !=true)
        {
         DeltaZ+=-GZmean;
         GZSD=sqrt((sampleN-2)/(sampleN-1)*GZSD*GZSD+DeltaZ*DeltaZ+
                (gyroz-GZmean)*(gyroz-GZmean)/(sampleN-1));
         DeltaZ=GZmean;
        }
}
else                                           // Initialize variables for first sample
{
```

```
        deltax=accelx;
        deltay=accely;
        deltaz=accelz;
        DeltaX=gyrox;
        DeltaY=gyroy;
        DeltaZ=gyroz;
 }

/* Minimum of 4 minutes of averaged data has been accumulated */
if (sampleN >= 240)
{
 /* Calculate confidence intervals when CI<noise condition has not already been reached*/
 /* Accelerometers: 99% probability */
 if (AXtcond !=true)
        AXCI = (2.576 + 4.92/(sampleN-1))*AXSD/sqrt(sampleN);
 if (AYtcond != true)
        AYCI = (2.576 + 4.92/(sampleN-1))*AYSD/sqrt(sampleN);
 if (AZtcond != true)
        AZCI = (2.576 + 4.92/(sampleN-1))*AZSD/sqrt(sampleN);

/* Gyroscopes 95 % probability */
 if (GXtcond !=true)
        GXCI = (1.960 + 2.4/(sampleN-1))*GXSD/sqrt(sampleN);
 if (GYtcond != true)
        GYCI = (1.960 + 2.4/(sampleN-1))*GYSD/sqrt(sampleN);
 if (GZtcond != true)
        GZCI = (1.960 + 2.4/(sampleN-1))*GZSD/sqrt(sampleN);

/* If Confidence Interval < Noise then condition is satisfied */
 if (AXCI < anoise) AXtcond=true;
 if (AYCI < anoise) AYtcond=true;
 if (AZCI < anoise) AZtcond=true;
 if (GXCI < gxrnd) GXtcond=true;
 if (GYCI < gyrnd) GYtcond=true;
 if (GZCI < gzrnd) GZtcond=true;
} // endif
} // end while

/*calculate alignment time, for 1 Hz data */
align_time=(sampleN)/60;

/* Averaged results found */
fprintf(results,"\n\n\n          GYROSCOPE AND ACCELEROMETER RAW DATA
        (deg/sec and g's)\n\n");
fprintf(results,"GXmean: %10.7f   GYmean: %10.7f   GZmean:
        %10.7f\n",GXmean,GYmean,GZmean);
fprintf(results,"AXmean: %10.7f   AYmean: %10.7f   AZmean:
        %10.7f\n",AXmean,AYmean,AZmean);

beta[0]=-AXmean;                // Accelerometer mean
beta[1]=AYmean;
```

```
beta[2]=AZmean;

AccelComp(b);                    // Error compensation of accelerometer data

omega[0]=-GXmean;                // Gyroscope mean
omega[1]=-GYmean;
omega[2]=GZmean;                 // no sign change since data referenced to torquers

GyroComp(w,b);                   // Error compensation of gyroscope data

window(1,19,80,30);
textcolor(YELLOW);
cprintf("\n                INTERMEDIATE RESULTS (deg/hr and g's)\n\r");
fprintf(results,"\n\n\n                    INTERMEDIATE RESULTS (deg/hr)\n\n");
textcolor(CYAN);

cprintf("Align Time (min): %3.2f\n\r",align_time);
cprintf("GYRO X : %8.5f    GYRO Y: %8.5f    GYRO Z:
        %8.5f\n\r",*omega,*(omega+1),*(omega+2));
cprintf("GSD  X : %8.5f    GSD  Y: %8.5f    GSD  Z:
        %8.5f\n\r",GXSD*3600,GYSD*3600,GZSD*3600);
fprintf(results,"Align Time (min): %3.2f\n\n",align_time),
fprintf(results,"Mean, Standard Deviation, and Confidence Intervals (deg/hr and g's)\n\n");
fprintf(results,"GYRO X : %8.5f    GYRO Y: %8.5f    GYRO Z:
        %8.5f\n",*omega,*(omega+1),*(omega+2));
fprintf(results,"GSD  X : %8.5f    GSD  Y: %8.5f    GSD  Z:
        %8.5f\n",GXSD*3600,GYSD*3600,GZSD*3600);
fprintf(results,"GCI  X : %8.5f    GCI  Y: %8.5f    GCI  Z:
        %8.5f\n\n",GXCI*3600,GYCI*3600,GZCI*3600);

cprintf("ACCEL. X: %8.5f    ACCEL. Y: %8.5f    ACCEL. Z:
        %8.5f\n\r",*beta,*(beta+1),*(beta+2));
cprintf("ASD  X : %8.5f    ASD  Y : %8.5f    ASD  Z : %8.5f\n\n\r",AXSD,AYSD,AZSD);
fprintf(results,"ACCEL. X: %8.5f    ACCEL. Y: %8.5f    ACCEL. Z:
        %8.5f\n",*beta,*(beta+1),*(beta+2));
fprintf(results,"ASD  X : %8.5f    ASD  Y : %8.5f    ASD  Z :
        %8.5f\n",AXSD,AYSD,AZSD),
fprintf(results,"ACI  X : %8.5f    ACI  Y : %8.5f    ACI  Z : %8.5f\n\n",AXCI,AYCI,AZCI);

window(1,14,80,16);
textcolor(CYAN);
textcolor(YELLOW);
cprintf("\n                FINAL RESULTS\r");
fprintf(results,"\n                    FINAL RESULTS\n"),
textcolor(CYAN);
window(1,16,40,25);

/* Determine [B] transpose matrix, using previously computed mean results */
// Row 0
norm=sqrt(omega[0]*omega[0] + omega[1]*omega[1] + omega[2]*omega[2]);
if (norm==0) cprintf("Norm Error division by zero"),
```

```
        else
        {
                BT[0][0]=*omega/norm;
                BT[0][1]=*(omega+1)/norm;
                BT[0][2]=*(omega+2)/norm;
        }

// Row 1
BT[1][0] = omega[1]*beta[2] - omega[2]*beta[1];
BT[1][1] = omega[2]*beta[0] - omega[0]*beta[2];
BT[1][2] = omega[0]*beta[1] - omega[1]*beta[0];
norm=sqrt(BT[1][0]*BT[1][0] + BT[1][1]*BT[1][1] + BT[1][2]*BT[1][2]);
if (norm==0.000001) cprintf("Norm Error division by zero");
        else
        {
                BT[1][0] = BT[1][0]/norm;
                BT[1][1] = BT[1][1]/norm;
                BT[1][2] = BT[1][2]/norm;
        }

// Row 2
BT[2][0] = omega[1]*BT[1][2] - omega[2]*BT[1][1];
BT[2][1] = omega[2]*BT[1][0] - omega[0]*BT[1][2];
BT[2][2] = omega[0]*BT[1][1] - omega[1]*BT[1][0];
norm=sqrt(BT[2][0]*BT[2][0] + BT[2][1]*BT[2][1] + BT[2][2]*BT[2][2]);
if (norm==0.000001) cprintf("Norm Error division by zero");
        else
        {
                BT[2][0] = BT[2][0]/norm;
                BT[2][1] = BT[2][1]/norm;
                BT[2][2] = BT[2][2]/norm;
        }

cprintf ("BODY MATRIX TRANSPOSE \r");
fprintf (results,"BODY MATRIX TRANSPOSE \n");
for (i=0;i<N;i++)
        for (j=0;j<N;j++){
                cprintf("\nRow: %d   Col: %d   Element: %8.5f\r",i,j,BT[i][j]);
                fprintf(results,"Row: %d   Col: %d   Element: %8.5f\n",i,j,BT[i][j]);
                }

/* MATRIX MULTIPLICATION to find DCM */
for (i=0; i<N;i++)
        for (j=0;j<N;j++)
                for (k=0, C[i][j]=0;k<N;k++)
                        C[i][j] += G[i][k] * BT[k][j];


window(41,16,80,25);
cprintf("DIRECTION COSINE MATRIX\r");
fprintf(results,"\n\nDIRECTION COSINE MATRIX\n");
```

```
        for (i=0;i<N;i++)
                for (j=0;j<N;j++){
                        cprintf("\nRow: %2d   Col: %2d   Element: %8.5f\r",i,j,C[i][j]);
                        fprintf(results,"Row: %2d   Col: %2d   Element: %8.5f\n",i,j,C[i][j]);
                        }
        fclose(in_accel);
        fclose(in_gyro);
        fclose(results);
}

/*****************************************************************************
*                                                                            *
*       void GyroComp(double *omegac, double *betac)                         *
*                                                                            *
*       Description: Gyroscope Data Compensation Algorithm                   *
*                                                                            *
*       Input:   raw gyroscope data, in deg/sec [omegac]                     *
*                corrected accelerometer data, in g [betac]                  *
*       Output:  corrected gyroscope data, along system axes,                *
*                in deg/hr [omegac]                                          *
*                                                                            *
*****************************************************************************/
void   GyroComp(double *omegac,double *betac)
{
        double  temp[3][1];
        const   N=3;
        const   M=0;
        int     i,k;

        /* in deg/hr, X, Y, Z System axes */
        temp[0][0]=3600 * K19x * (*omegac) + GB19x + Q19x*(*(betac+2)) + MU19x*(*betac);
        temp[1][0]=3600 * K24x * (*(omegac+1)) + GB24x + Q24x*(*(betac+2)) +
                        MU24x*(*(betac+1));
        temp[2][0]=3600 * K24y * (*(omegac+2)) - GB24y - Q24y*(*(betac+1)) -
                        MU24y*(*(betac+2)),

        /* Matrix multiplication by misalignment angles (3X3)*(3X1) */
        for (k=0;k<N;k++)
                *(omegac+k)=0;                  // re-initialize
        for (k=0;k<N;k++)
                        for (i=0, i<N ; i++)
                                *(omegac+k) += GM[k][i] * temp[i][M];
}

/*****************************************************************************
*                                                                            *
*       void AccelComp(double *betac)                                        *
*                                                                            *
*       Description: Accelerometer Data Compensation Algorithm               *
*                                                                            *
*       Input:   raw accelerometer data, in g [betac]                        *
```

```
*       Output: corrected accelerometer data, along system axes,                    *
*               in g [betac]                                                        *
*                                                                                   *
*************************************************************************************/
void   AccelComp(double *betac)
{
        double  temp[3][1];
        const   N=3;
        const   M=0;
        int     i,k;

        /* in units of g */
        temp[0][0]=K232x * (*betac) + AB232x;           // X system
        temp[1][0]=K228y * (*(betac+1)) - AB228y;       // Y system
        temp[2][0]=K234z * (*(betac+2)) - AB234z;       // Z system

        /* Matrix multiplication (3X3)*(3X1) */
        for (k=0;k<N;k++)
                *(betac+k)=0;                   // re-initialize
        for (k=0;k<N;k++)
                        for (i=0; i<N ; i++)
                                *(betac+k) += AM[k][i] * temp[i][M];
}
```

C.2 Navigation Main Program, NAVIGAT.CPP

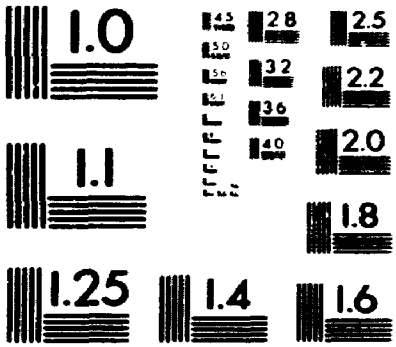
```
/*****************************************************************************
*                                                                            *
*    STRAPDOWN LOCAL-LEVEL WANDER-AZIMUTH NAVIGATION SOFTWARE                *
*    FOR THE DREO HRU  (MAIN PROGRAM)                                        *
*                                                                            *
*       Filename:    NAVIGAT.CPP                                            *
*       Author:      Carole R.M. Bolduc                                     *
*       Date:        Nov 28, 1994                                           *
*                                                                            *
*       Version 1.0                                                          *
*                                                                            *
*       Requires:    Two input files, containing raw HRU gyroscope and      *
*                    accelerometer data. File format: as generated by       *
*                    HRU CDU v1.5 for raw data collection, with GPS time stamps.
*                                                                            *
*       Description: MAIN program                                            *
*                 1. Initialization Sequence (User prompts)                  *
*                 2. Calculate required initial values (D elements,          *
*                    quaternions, Euler Angles, Position, Ground Velocity)   *
*                 3. Read data from file and compensate for known errors.    *
*                 4. Call differential equation solver to integrate the linear
*                    D E's and obtain navigation solution                    *
*                 5. Print results to file and display                       *
*                                                                            *
*       Outputs:     Time(sec), Roll(deg), Pitch(deg), Heading(deg),        *
*                    Wander Angle(deg), Latitude(deg), Longitude(deg)       *
*                    Ground Velocity(kph)                                   *
*                                                                            *
*****************************************************************************/

#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "matrix.h"
#include "odeslv.h"

double  Q[9];
double *Qmatrix;                    // Q matrix
double q[4] = {0,0,0,0};            // quaternion parameters
double *qprev;
double  veh_rate_e[3] = {0,0,0};    // vehicle rates relative to local-level frame of reference
double EA[3];                       // Euler Angles
double  *vehrates;
double  D[9];                       // D matrix
double *Dmatrix,*Eul;
double  ACC[3],V[3]={0,0,0},v[3]={0,0,0}; //Accelerometer and gyro data
```

```
double  *Acc_pt,*Vel_pt,*filterv_pt;
double  latitude,longitude,wander;          //Position information
double  *la,*lo,*wa;
double  gvel;                               // ground velocity
char    ax_str[80],ay_str[80],az_str[80],d_str[80],data_str[80];
char    gx_str[80],gy_str[80],gz_str[80],sequence[80];
char    gfilename[80],afilename[80],ofilename[80];
int     i,j,sampleN=0;                      // Counters
int     DONE=0;                             // initialize outer loop as not done

const   true=1;
const   double pi=3.1415926536;

FILE    *in_accel;
FILE    *in_gyro;
FILE    *results;

void main(void)
{
        clrscr();
        printf("NAVIGATION SOFTWARE\n"),
        Qmatrix = Q;                        // pointer initialization
        qprev = q;
        vehrates=veh_rate_e;
        Dmatrix = D;
        Eul = EA;
        Acc_pt=ACC;
        Vel_pt=V;
        filterv_pt=v;
        la=&latitude;
        lo=&longitude;
        wa=&wander;

        /* Initial Values */
        printf("\n\nRequest for alignment information:\n\n");

        /* Prompt user for initial DCM found during Coarse Alignment */
//      printf("Enter the C Direction Cosine Matrix Elements (a.k.a [Q])\n");
/*      for (i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        printf("C[%i][%i]?  ",i,j);
                        gets(data_str);
                        *(Q+3*i+j)=atof(data_str),
                }
        }
*/
        /* Alternatively, ask user for initial Euler Angles */
        printf("Enter the Euler Angles (R,P,Y (deg))\n");
                for(j=0;j<3;j++)
```

```
        {
                printf("EA[%i]? ",j);
                gets(data_str);
                *(Eul+j)=atof(data_str);
        }
EA[0]=EA[0]*pi/180;                     // Convert to radians
EA[1]=EA[1]*pi/180;
EA[2]=EA[2]*pi/180;
EA_quat(Eul,qprev);                     // Find quaternion parameters from Euler Angles

/* Prompt user for initial lat,long and wander angle */
printf("\nEnter position information (in degrees)\n");
printf("Latitude? ");
gets(data_str);
latitude=(pi/180)*atof(data_str);
printf("Longitude? ");
gets(data_str);
longitude=(pi/180)*atof(data_str);
printf("Wander Angle? (Enter 0 for default)");
gets(data_str);
wander=(pi/180)*atof(data_str);
printf("\n\nHit Enter to Continue");
gets(data_str);
clrscr();
printf("NAVIGATION SOFTWARE\n");
printf("\n\nEnter initial true and filtered velocities (in m/s) \n");
printf("True Velocity Vx? (Enter 0 for default) ");
gets(data_str);
V[0]=atof(data_str);
printf("True Velocity Vy? (Enter 0 for default) ");
gets(data_str);
V[1]=atof(data_str);
printf("True Velocity Vz? (Enter 0 for default) ");
gets(data_str);
V[2]=atof(data_str);
printf("Filtered Velocity vx? (Enter 0 for default) ");
gets(data_str);
v[0]=atof(data_str);
printf("Filtered Velocity vy? (Enter 0 for default) ");
gets(data_str);
v[1]=atof(data_str);
printf("Filtered Velocity vz? (Enter 0 for default) ");
gets(data_str);
v[2]=atof(data_str);

DDCM_init(Dmatrix,latitude,longitude,wander);   // Initialize 6 elements of D matrix

//      quat_params(Qmatrix,qprev);             // Find quaternion parameters from Q
                                                // for Coarse align initialization sequence
```

3 of/de 3

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT



PRECISION℠ RESOLUTION TARGETS

```
/* Source filenames */
printf("\nEnter filename (including path) for source accel data:  ");
gets(afilename);
if ((in_accel = fopen(afilename,"r"))==NULL) {
        printf ("cannot open accelerometer data file \n");
        return;
        }
printf("\nEnter filename (including path) for source gyro data:  ");
gets(gfilename);
if ((in_gyro = fopen(gfilename,"r"))==NULL) {
        printf ("cannot open gyro data file \n");
        return;
        }
printf("\nEnter results output filename  ");
gets(ofilename);
if ((results = fopen(ofilename,"w"))==NULL) {
        printf ("cannot open output file \n");
        return;
        }

/* Go to specific file position, remove GPS time stamp */
do fgets(sequence,14,in_accel);
while (strcmp(sequence,"********** GPS"));
fseek(in_accel,40,SEEK_CUR);

do fgets(sequence,14,in_gyro);
while (strcmp(sequence,"********** GPS"));
fseek(in_gyro,40,SEEK_CUR);

clrscr();

/* Select start position of file, for example, index can be adjusted to start processing data at
   point 500  Currently, only first data point is removed  */
for (i=0;i<1;i++)
{
 switch (fscanf(in_accel,"%8s %14s %14s %14s %14s %14s
        %14s",d_str,d_str,ax_str,d_str,ay_str,d_str,az_str))
          {
                case '0':  printf("File Read Error\n");
                                DONE=true;break;
                case EOF        printf("End of File Reached\n");
                                DONE=true;break;
                default         if (strcmp(ax_str,"GPS")==0)   DONE=true;
          }
          switch (fscanf(in_gyro,"%8s %14s %14s %14s %14s %14s %14s %14s
                %14s",d_str,d_str,d_str,d_str,gx_str,d_str,gz_str,d_str,gy_str))
          {
                case '0':       printf("File Read Error\n");
                                DONE=true;break;
                case EOF:       printf("End of File Reached\n");
                                DONE=true;break;
```

```
                default:        if (strcmp(gx_str,"TIME")==0)  DONE=true;
            }
        }

        printf(" Time  Roll   Pitch   Yaw     Lat     Long     Wander    GV\n\n");
        fprintf(results," Time  Roll   Pitch   Yaw     Lat     Long     Wander    GV\n\n");

//      Euler_Angles(qprev,Eul,wander)          // IF DCM used for initialization
        /* Note:  for 1 Hz data, sampleN = elapsed time */
        printf("%4u  %5.3f  %5.3f  %5.3f  %8.5f  %8.5f  %8.5f %
                5.3f\n",sampleN,EA[0]*180/pi,EA[1]*180/pi,EA[2]*180/pi,latitude*180/pi,
                longitude*180/pi,wander*180/pi,gvel*3.6);

                while (!DONE)                           // Outer, Slower loop
        {
                Position(la,lo,wa,Dmatrix);
                Euler_Angles(qprev,Eul,wander);
                gvel = sqrt(V[0]*V[0] + V[1]*V[1]);

                printf("%4u  %5.3f  %5.3f  %5.3f  %8.5f  %8.5f  %8.5f %
                        5.3f\n",sampleN,EA[0]*180/pi,EA[1]*180/pi,EA[2]*180/pi,latitude*180/pi,
                        longitude*180/pi,wander*180/pi,gvel*3.6);
                fprintf(results,"%4u  %5.3f  %5.3f  %5.3f  %8.5f  %8.5f  %8.5f %
                        5.3f\n",sampleN,EA[0]*180/pi,EA[1]*180/pi,EA[2]*180/pi,latitude*180/pi,
                        longitude*180/pi,wander*180/pi,gvel*3.6);

                for (i=0;i<10;i++)                      // Fast, inner loop
                {
                  /* Read in raw data; Read to EOF (GPS time stamp removed) */
                  switch (fscanf(in_accel,"%8s %14s %14s %14s %14s %14s
                        %14s",d_str,d_str,ax_str,d_str,ay_str,d_str,az_str))
                  {
                        case '0':  printf("File Read Error\n");
                                   DONE=true;break;
                        case EOF:  printf("End of File Reached\n");
                                   DONE=true;break;
                        default:   if (strcmp(ax_str,"GPS")!=0)
                                   {
                                        ACC[0]=atof(ax_str);
                                        ACC[1]=atof(ay_str);
                                        ACC[2]=atof(az_str);
                                   }
                                   else DONE=true;
                  }
                  switch (fscanf(in_gyro,"%8s %14s %14s %14s %14s %14s %14s %14s
                        %14s",d_str,d_str,d_str,d_str,gx_str,d_str,gz_str,d_str,gy_str))
                  {
                        case '0':  printf("File Read Error\n");
                                   DONE=true;break;
                        case EOF:  printf("End of File Reached\n");
                                   DONE=true;break;
```

```
                    default:    if (strcmp(gx_str,"TIME")!=0)
                                {
                                        veh_rate_e[0]=atof(gx_str);
                                        veh_rate_e[1]=atof(gy_str);
                                        veh_rate_e[2]=atof(gz_str);
                                }
                                else DONE=true;
                }

                if (DONE==true)         break;  //exits the for loop

                ++sampleN;

                ACC[0]=-ACC[0].                 // Data-sign correction for ENU system axes
                ACC[1]=ACC[1].                  // convention
                ACC[2]=ACC[2].
                veh_rate_e[0]=-veh_rate_e[0].
                veh_rate_e[1]=-veh_rate_e[1].
//              veh_rate_e[2]=-veh_rate_e[2].   Z-rate unchanged since data referenced to torquers

                /* Inertial Sensor Error Compensation */
                AccelComp(Acc_pt).              // Accelerometers
                GyroComp(vehrates.Acc_pt).      // Gyroscopes

                /* Integrate the differential equations */
                RK4_Routines(qprev. vehrates. Dmatrix. Acc_pt. Vel_pt. filterv_pt).

                Position(la.lo.wa.Dmatrix).
                Euler_Angles(qprev.Eul.wander).
                gvel = sqrt(V[0]*V[0] + V[1]*V[1]).

                /* Results not displayed for high sampling rates
                   The execution of these statements is optional */
                printf("%4u  %5.3f  %5.3f  %5.3f  %8.5f  %8.5f  %8.5f  %
                        5.3f\n".sampleN.EA[0]*180/pi.EA[1]*180/pi.EA[2]*180/pi.latitude*180/pi.
                        longitude*180/pi.wander*180/pi.gvel*3.6).
                fprintf(results."%4u  %5.3f  %5.3f  %5.3f  %8.5f  %8.5f  %8.5f  %
                        5.3f\n".sampleN.EA[0]*180/pi.EA[1]*180/pi.EA[2]*180/pi.latitude*180/pi.
                        longitude*180/pi.wander*180/pi.gvel*3.6).
            }  // END INNER LOOP
        } // END OUTER LOOP

        fclose(in_accel).
        fclose(in_gyro);
        fclose(results);
}
```

C.3 Navigation Software Functions, MATRIX.CPP

```
/*****************************************************************************
*                                                                            *
*       STRAPDOWN LOCAL-LEVEL WANDER-AZIMUTH NAVIGATION SOFTWARE             *
*       (INCLUDE FILE)                                                       *
*                                                                            *
*       Filename:       MATRIX.CPP                                           *
*       Author:         Carole R.M. Bolduc                                   *
*       Date:           Oct 27,1994                                          *
*                                                                            *
*       Description:    Collection of functions used by the NAVIGAT.CPP      *
*                       main program                                         *
*                                                                            *
*****************************************************************************/

#include "matrix.h"
#include <math.h>
#include <stdio.h>

//gyroscope biases in deg/hr (found from simplified multiposition test)
#define         GB24x   -2.6698
#define         GB24y   0.84124
#define         GB19x   -7.9482
#define         GB19y   6.7845

//accelerometer biases in g (found from simplified multiposition test)
#define         AB232x 0.00292658083
#define         AB228y 0.0004389546
#define         AB234z 0.009026187

// Gyro and accelerometer scale factors (Note: raw data already includes scale factor correction)
#define         K24x    1
#define         K24y    1
#define         K19x    1
#define         K19y    1

#define         K232x   1
#define         K228y   1
#define         K234z   1

// Misalignment angles  (not available)
#define         Gyx     0
#define         Gzx     0
#define         Gxy     0
#define         Gzy     0
#define         Gxz     0
#define         Gyz     0
```

```
#define        Ayx    0
#define        Azx    0
#define        Axy    0
#define        Azy    0
#define        Axz    0
#define        Ayz    0

// Gyroscope Mass Unbalance and Quadrature Terms (Litton test data) */
#define        MU19x  1.191
#define        Q19x   7.743
#define        MU19y  1.233
#define        Q19y   7.680
#define        MU24x  -3.0981
#define        Q24x   -0.6642
#define        MU24y  -0.5190
#define        Q24y   -2.9654

double  GM[3][3]={1,Gyx,Gzx,Gxy,1,Gzy,Gxz,Gyz,1},
double  AM[3][3]={1,Ayx,Azx,Axy,1,Azy,Axz,Ayz,1},


/*****************************************************************************
*                                                                            *
*       void EA_quat(double *EA, double *qfirst)                             *
*                                                                            *
*       Description   Calculates quaternion parameters                       *
*                                                                            *
*        Input   Roll, Pitch and Yaw Euler angles (in radians) [EA]          *
*        Output: quaternions [qfirst]                                        *
*                                                                            *
*****************************************************************************/
void EA_quat(double *EA, double *qfirst)
{
        double  tmp,norm,

        tmp= 1+(cos(EA[1])*cos(EA[2]))+(cos(EA[0])*cos(EA[2])+
                sin(EA[0])*sin(EA[1])*sin(EA[2])) + cos(EA[1])*cos(EA[0]),
        *(qfirst+3)=(sqrt(tmp))/2,
        *qfirst=(-sin(EA[0])*cos(EA[2])+cos(EA[0])*sin(EA[1])*sin(EA[2])-
                sin(EA[0])*cos(EA[1]))/(4*(*(qfirst+3))),
        *(qfirst+1)=(-sin(EA[1])-(sin(EA[0])*sin(EA[2])+
                cos(EA[0])*sin(EA[1])*cos(EA[2])))/(4*(*(qfirst+3))),
        *(qfirst+2)=(-cos(EA[1])*sin(EA[2])+(-cos(EA[0])*sin(EA[2])+
                sin(EA[0])*sin(EA[1])*cos(EA[2])))/(4*(*(qfirst+3))),

        /* Normalize the results:  Apply the forcing condition */
        norm = sqrt((*qfirst)*(*qfirst) + *(qfirst+1)*(*(qfirst+1)) + *(qfirst+2)*(*(qfirst+2)) +
                *(qfirst+3)*(*(qfirst+3))),
        *(qfirst)=*(qfirst)/norm;
        *(qfirst+1)=*(qfirst+1)/norm,
        *(qfirst+2)=*(qfirst+2)/norm,
```

```
        *(qfirst+3)=*(qfirst+3)/norm.
}

/*****************************************************************************
*                                                                            *
*       void quat_params(double *Qm, double *qfirst)                         *
*                                                                            *
*       Description:  Calculates quaternion parameters                       *
*                                                                            *
*       Input:   Q direction cosine matrix elements [Q]                      *
*       Output:  quaternions [qfirst]                                        *
*                                                                            *
*****************************************************************************/
void  quat_params(double *Qm, double *qfirst)
{
        double   norm.

        *(qfirst+3) = sqrt(1+ *Qm + *(Qm+4) + *(Qm+8))/2.
        if (*(qfirst+3) > fabs(0.0000000000000001))   // double (15 digit precision)
        {
                *qfirst= (*(Qm+5) - *(Qm+7))/(4 * (*(qfirst+3))).
                *(qfirst+1) = (*(Qm+6) - *(Qm+2))/(4*(*(qfirst+3))).
                *(qfirst+2) = (*(Qm+1) - *(Qm+3))/(4*(*(qfirst+3))).
        }
        else printf("Error. Quaternion parameter q4 is zero\n").

        /* Normalize the results:  Apply the forcing condition */
        norm = sqrt((*qfirst)*(*qfirst) + *(qfirst+1)*(*(qfirst+1)) + *(qfirst+2)*(*(qfirst+2)) +
                *(qfirst+3)*(*(qfirst+3))).
        *(qfirst)=*(qfirst)/norm;
        *(qfirst+1)=*(qfirst+1)/norm.
        *(qfirst+2)=*(qfirst+2)/norm;
        *(qfirst+3)=*(qfirst+3)/norm;
}

/*****************************************************************************
*                                                                            *
*       void Gyrocomp(double *omegac, double *betac)                         *
*                                                                            *
*       Description:    Gyroscope Data compensation algorithm                *
*                                                                            *
*       Input:    raw gyroscope data, in deg/sec [omegac]                    *
*                 corrected accelerometer data, in m/(sec*sec) [betac]       *
*       Output: corrected gyroscope data, along system axes, rad/sec [omegac] *
*                                                                            *
*****************************************************************************/
void  GyroComp(double *omegac, double *betac)
{
        double   temp[3];
        const    N=3;
```

```
        int     i,k;
        const   double pi=3.1415926536;

        /* X,Y,Z system axes, converted to deg/hr since bias drift, MU and Q are in degs/hr/g */
        temp[0]=3600 * K19x * (*omegac) + GB19x + Q19x * (*(betac+2))/9.806 +
                MU19x * (*betac)/9.806;
        temp[1]=3600 * K24x * (*(omegac+1)) + GB24x + Q24x * (*(betac+2))/9.806 +
                MU24x * (*(betac+1))/9.806;
        temp[2]=3600 * K24y * (*(omegac+2)) - GB24y - Q24y * (*(betac+1))/9.806 -
                MU24y * (*(betac+2))/9.806;

        /* Misalignment Matrix multiplication, conversion to rad/sec */
        for (k=0;k<N;k++)
                *(omegac+k)=0;
        for (k=0;k<N;k++)
                for (i=0;i<N;i++)
                        *(omegac+k) += GM[k][i] * (pi/648000)*temp[i];
}

/*****************************************************************************
*                                                                            *
*       void AccelComp(double *betac)                                        *
*                                                                            *
*       Description   Accelerometer Data compensation algorithm              *
*                                                                            *
*       Input   raw accelerometer data, in g  [betac]                        *
*       Output  corrected accelerometer data, along system axes m/(sec*sec) [betac]  *
*                                                                            *
*****************************************************************************/
void  AccelComp(double *betac)
{
        double  temp[3];
        const   N=3;
        int     i,k;

        /* Units of g */
        temp[0]=K232x * (*betac) + AB232x;              // X system
        temp[1]=K228y * (*(betac+1)) - AB228y;          // Y system
        temp[2]=K234z * (*(betac+2)) - AB234z;          // Z system

        /* Misalignment Matrix multiplication, conversion to metres per second squared */
        for (k=0;k<N;k++)
                *(betac+k)=0;
        for (k=0;k<N;k++)
                for (i=0;i<N;i++)
                        *(betac+k) += AM[k][i] * 9.806*temp[i];
}
```

```
/*****************************************************************************
*                                                                            *
*       void Euler_Angles(double *quat, double *Euler, double wan)           *
*                                                                            *
*       Description:  Calculates attitude and heading information            *
*                                                                            *
*       Input:   quaternion parameters [quat], wander angle (rad) [wan]      *
*       Output:  roll, pitch, true heading (in radians) [Euler]              *
*                                                                            *
*****************************************************************************/
void  Euler_Angles(double *quat, double *Euler, double wan)
{
        double   y1,x1,y3,x3,z2;

        y1=2*(quat[1]*quat[2]-quat[0]*quat[3]);
        x1=quat[3]*quat[3]-quat[0]*quat[0]-quat[1]*quat[1]+quat[2]*quat[2];
        y3=2*(quat[0]*quat[1]-quat[2]*quat[3]);
        x3=quat[3]*quat[3]+quat[0]*quat[0]-quat[1]*quat[1]-quat[2]*quat[2];
        z2=(-2)*(quat[1]*quat[3]+quat[0]*quat[2]);

        Euler[0]=atan2(y1,x1);          // Range for roll -pi to +pi
        Euler[1]=asin(z2);              // Range for pitch -pi/2 to pi/2
        Euler[2]=atan2(y3,x3)-wan;      // Range for yaw between -pi and +pi
}

/*****************************************************************************
*                                                                            *
*       void Quat_CDCM(double *quat, double *C)                              *
*                                                                            *
*       Description:  Calculates 'C' DCM used to transform vectors from      *
*                     body to local-level wander-az frame of reference.      *
*                                                                            *
*       Input:   quaternion parameters [quat]                                *
*       Output:  updated transformation matrix C [C]                         *
*                                                                            *
*****************************************************************************/
void  Quat_CDCM(double *quat, double *C)
{
        C[0]=quat[3]*quat[3]+quat[0]*quat[0]-quat[1]*quat[1]-quat[2]*quat[2];
        C[1]=2*(quat[0]*quat[1]-quat[2]*quat[3]);
        C[2]=2*(quat[0]*quat[2]+quat[1]*quat[3]);
        C[3]=2*(quat[0]*quat[1]+quat[2]*quat[3]);
        C[4]=quat[3]*quat[3]-quat[0]*quat[0]+quat[1]*quat[1]-quat[2]*quat[2];
        C[5]=2*(quat[1]*quat[2]-quat[0]*quat[3]);
        C[6]=2*(quat[0]*quat[2]-quat[1]*quat[3]);
        C[7]=2*(quat[1]*quat[2]+quat[0]*quat[3]);
        C[8]=quat[3]*quat[3]-quat[0]*quat[0]-quat[1]*quat[1]+quat[2]*quat[2];
}
```

```
/*****************************************************************************
*                                                                            *
*       DDCM_init(double *DM, double phi, double lambda, double alpha)       *
*                                                                            *
*       Description:  Calculates 6 elements of'D' DCM relating local-level   *
*                     frame of reference to earth-fixed coordinates.         *
*                                                                            *
*       Input:   latitude [phi], longitude[lambda], wander angle[alpha] (in radians)  *
*        Output: updated 6 D matrix elements [DM]                            *
*                                                                            *
*****************************************************************************/
void DDCM_init(double *DM,double phi, double lambda, double alpha)
{
        *DM=cos(alpha)*cos(lambda)-sin(alpha)*sin(phi)*sin(lambda);        // D11
        *(DM+1)=-sin(alpha)*cos(lambda)-sin(phi)*sin(lambda)*cos(alpha);   // D21
        *(DM+2)=cos(phi)*sin(lambda);                                      // D31
        *(DM+3)=sin(alpha)*cos(phi);                                       // D12
        *(DM+4)=cos(alpha)*cos(phi);                                       // D22
        *(DM+5)=sin(phi);                                                  // D32
}

/*****************************************************************************
*                                                                            *
*       void True_Accel(double *C, double *Acc, double *TAcc)                *
*                                                                            *
*       Description:  Transforms corrected accelerometer outputs in body coordinates to  *
*                     level frame of reference coords  This is the acceleration term used in *
*                      the velocity linear differential equations            *
*                                                                            *
*       Input    'C' transformation matrix, [C],Corrected accelerometer data  [Acc] (m/(s*s))  *
*       Output: accelerations in local-level frame of reference [TAcc] (m/(s*s)))  *
*                                                                            *
*****************************************************************************/
void True_Accel(double *C, double *Acc,double *TAcc)
{
        int     k,i;
        double  gmodel[3],
        const   N=3,

        for (k=0;k<N;k++)
                *(TAcc+k)=0,
        for (k=0;k<N;k++)                   // multiply by transformation matrix
                for (i=0;i<N;i++)
                        *(TAcc+k) +=C[i+3*k] * Acc[i],
}
```

```
/*****************************************************************************
*                                                                            *
*       void True_Ang_Vel(double *C, double *omegat, double *frame_rate,     *
*                         double *ER, double *Trate)                         *
*                                                                            *
*       Description:  Transforms corrected gyroscope outputs in body coordinates to *
*                     local-level frame of ref. coords. Removes earth rate and applies damped *
*                     precession rate control to compensate for angular rate of platform relative *
*                     to the earth.                                          *
*                                                                            *
*        Input:   'C' transformation matrix [C], Corrected gyroscope data (in rad/sec) [omegat] *
*                 damped spatial rate [frame_rate], earth rate [ER]          *
*        Output: system angular velocity relative to local-level frame  [Trate] *
*                                                                            *
*****************************************************************************/
void  True_Ang_Vel(double *C, double *omegat, double *frame_rate, double *ER, double *Trate)
{
        int     i,k;
        const   N=3;

        for (k=0;k<N;k++)
                *(Trate+k)=0;
        for (k=0;k<N;k++)
                for (i=0;i<N;i++){
                        *(Trate+k) += C[i+3*k] * (omegat[i]);
                                }
        for (i=0;i<3;i++)
                *(Trate+i)+=-*(frame_rate+i)-*(ER+i);           // units rad/sec
}

/*****************************************************************************
*                                                                            *
*       void Position(double *phi, double *lambda, double *alpha, double * Geog) *
*                                                                            *
*       Description:  Determines position information                        *
*                                                                            *
*       Input:    6 elements of 'D' matrix [Geog]                            *
*       Output:  latitude [phi], longitude [lambda], wander angle [alpha] (in radians) *
*                                                                            *
*****************************************************************************/
void  Position(double *phi, double *lambda, double *alpha, double *Geog)
{
        *phi=asin(Geog[5]);                                     // from -pi/2 to pi/2
        *alpha=atan2(Geog[3],Geog[4]);                          // from -pi to +pi
        *lambda=atan2(Geog[2],(Geog[0]*Geog[4]-Geog[3]*Geog[1]));   // from -pi to +pi
}
```

C.4 Navigation Differential Equation Solver, ODESLV.CPP

```
/*****************************************************************************
*                                                                            *
*   STRAPDOWN LOCAL-LEVEL WANDER-AZIMUTH NAVIGATION SOFTWARE                 *
*   FOR THE DREO HRU  (DIFFERENTIAL EQUATION SOLVER)                         *
*                                                                            *
*      Filename:      ODESLV.CPP                                             *
*      Author:        Carole R.M. Bolduc                                     *
*      Date:          Nov 2,1994                                             *
*                                                                            *
*      Version 1.0                                                           *
*                                                                            *
*      Inputs:  Quaternions, 6 'D' matrix elements, corrected accelerometer and gyroscope *
*      data, actual and filtered velocities                                  *
*                                                                            *
*      Outputs:       Updated quaternions,'D' matrix elements, 'C' matrix and velocities *
*                                                                            *
*      Description:   Called by main program  Finely Integrates the following differential *
*                     equations, using classical fourth-order Runge-Kutta method:  *
*                        1  Quaternions                                      *
*                        2. Low-Pass Filtered accelerations                  *
*                        3. Pure Accelerations                               *
*                        4. 'D' matrix elements                              *
*                                                                            *
*****************************************************************************/

#include <math.h>
#include "odeslv.h"
#include "matrix.h"

#define     h       0.13                    // step interval based on Keithley
                                            // meter sampling period
#define     Nsteps  10                      // Number of steps
#define     FER     0.000072924219472       // Full Earth Rate (rad/sec)

// Set of coupled Quaternion equations
#define     q1dot(q2,q3,q4)  ((h/2)*(newR[2]*(q2) - newR[1]*(q3) + newR[0]*(q4)))
#define     q2dot(q1,q3,q4)  ((h/2)*(-newR[2]*(q1) + newR[0]*(q3) + newR[1]*(q4)))
#define     q3dot(q1,q2,q4)  ((h/2)*(newR[1]*(q1) - newR[0]*(q2) + newR[2]*(q4)))
#define     q4dot(q1,q2,q3)  ((h/2)*(-newR[0]*(q1) - newR[1]*(q2) - newR[2]*(q3)))

// 6 D matrix element differential equations
#define     D11dot(D31)          (h*((-D31)*RHO[1]))
#define     D21dot(D31)          (h*((D31)*RHO[0]))
#define     D31dot(D11,D21)      (h*((D11)*RHO[1] - (D21)*RHO[0]))
#define     D12dot(D32)          (h*((-D32)*RHO[1]))
```

```
#define        D22dot(D32)          (h*((D32)*RHO[0]))
#define        D32dot(D12,D22)      (h*((D12)*RHO[1] - (D22)*RHO[0]))

// Pure accelerations
#define        Vxderiv(Vz,Vy)  (h*(newA[0]-(RHO[1]+2*LER[1])*(Vz) + (2*LER[2])*(Vy)))
#define        Vyderiv(Vx,Vz)  (h*(newA[1]-(2*LER[2])*(Vx) + (RHO[0]+2*LER[0])*(Vz)))
#define        Vzderiv(Vy,Vx)  (h*(newA[2]-(RHO[0]+2*LER[0])*(Vy) +
                               (RHO[1]+2*LER[1])*(Vx)-gzmodel))

void    RK4_Routines(double *qpar, double *rates, double *Delem, double *Accel, double
*Velocity,double *filter_v)
{
        double  kq1[4],kq2[4],kq3[4],kq4[4],qtmp[4],CM[9];     // RK parameters and C matrix
        double  *Cmatrix,*LER_pt;
        double  kD11[4],kD21[4],kD31[4],kD12[4],kD22[4],kD32[4],Dtmp[6];
        double  Vel[3];
        double  kVx[4],kVy[4],kVz[4],LER[3],newR[3],newA[3],
        double  kv1[4],kv2[4],vtmp[2];
        double  rho[3],RHO[2],norm;
        double  *Q,*w,*A;
        double  *rho_start,*RHO_start,*newRate,*newAcc,*qt;    // spatial rates, corrected sensor data
        int     i,j,ii,
        const   N=4;

        const  double Ry=6378140;          // Earth's radius
        const  double Rx=6378140,
        const  double R=3000000;           // 'R' fine alignment damping parameter
        const  double tau=1080;            // time constant
        const  double a=0.0;               // heading error damping
        const  double gzmodel=9.806;       // in metres per second squared

        Cmatrix=CM,
        qt=qtmp;

        /* Initialization phase*/
        rho_start=rho;
        LER_pt=LER;
        newRate=newR;
        newAcc=newA;

        for (i=0;i<N;i++)qtmp[i]=qpar[i];                  //RK  Initialization
        for (i=0;i<3;i++) Vel[i]=Velocity[i];
        for (i=0;i<2;i++) vtmp[i]=filter_v[i],
        for (i=0;i<6;i++) Dtmp[i]=Delem[i];

        for (i=0;i<Nsteps;i++)             /* loop iterations: 10 steps */
        {
                for (j=0;j<N;j++)          /* loop iterations: 4 */
                {
                        *rho=-vtmp[1]/R;                   // damped spatial rates
                        *(rho+1)=vtmp[0]/R;
```

```
*(rho+2)= a*Vyderiv(Vel[0],Vel[2]),
*(RHO)=-Vel[1]/Ry;                    // exact spatial rates
*(RHO+1)=Vel[0]/Rx;
*LER=FER*Dtmp[3];                     // local earth rate
*(LER+1)=FER*Dtmp[4];
*(LER+2)=FER*Dtmp[5];

Quat_CDCM(qt,Cmatrix);                // Find C from quaternions
True_Ang_Vel(Cmatrix,rates,rho_start,LER_pt,newRate);
True_Accel(Cmatrix,Accel,newAcc);

kD11[j]=D11dot(Dtmp[2]);
kD21[j]=D21dot(Dtmp[2]),
kD31[j]=D31dot(Dtmp[0],Dtmp[1]),
kD12[j]=D12dot(Dtmp[5]);
kD22[j]=D22dot(Dtmp[5]),
kD32[j]=D32dot(Dtmp[3],Dtmp[4]),

kq1[j]=q1dot(qtmp[1],qtmp[2],qtmp[3]),
kq2[j]=q2dot(qtmp[0],qtmp[2],qtmp[3]),
kq3[j]=q3dot(qtmp[0],qtmp[1],qtmp[3]),
kq4[j]=q4dot(qtmp[0],qtmp[1],qtmp[2]),

kVx[j]=Vxderiv(Vel[2],Vel[1]),
kVy[j]=Vyderiv(Vel[0],Vel[2]),
kVz[j]=Vzderiv(Vel[1],Vel[0]),

kv1[j]=kVx[j] - (h*vtmp[0]/tau),
kv2[j]=kVy[j] - (h*vtmp[1]/tau),

if (j<2)
{
        Dtmp[0]=Delem[0] + kD11[j]/2,
        Dtmp[1]=Delem[1] + kD21[j]/2,
        Dtmp[2]=Delem[2] + kD31[j]/2,
        Dtmp[3]=Delem[3] + kD12[j]/2,
        Dtmp[4]=Delem[4] + kD22[j]/2,
        Dtmp[5]=Delem[5] + kD32[j]/2,

        qtmp[0]=qpar[0] + kq1[j]/2;
        qtmp[1]=qpar[1] + kq2[j]/2,
        qtmp[2]=qpar[2] + kq3[j]/2;
        qtmp[3]=qpar[3] + kq4[j]/2;

        Vel[0]=Velocity[0]+kVx[j]/2,
        Vel[1]=Velocity[1]+kVy[j]/2,
        Vel[2]=Velocity[2]+kVz[j]/2,

        vtmp[0]=filter_v[0]+kv1[j]/2,        // Damped velocities
        vtmp[1]=filter_v[1]+kv2[j]/2,
}
```

```
else
{       if (j==2)
        {
                Dtmp[0]=Delem[0] + kD11[j]/2;
                Dtmp[1]=Delem[1] + kD21[j]/2;
                Dtmp[2]=Delem[2] + kD31[j]/2;
                Dtmp[3]=Delem[3] + kD12[j]/2;
                Dtmp[4]=Delem[4] + kD22[j]/2;
                Dtmp[5]=Delem[5] + kD32[j]/2;

                qtmp[0]=qpar[0]+kq1[j];
                qtmp[1]=qpar[1]+kq2[j];
                qtmp[2]=qpar[2]+kq3[j];
                qtmp[3]=qpar[3]+kq4[j];

                Vel[0]=Velocity[0]+kVx[j];
                Vel[1]=Velocity[1]+kVy[j];
                Vel[2]=Velocity[2]+kVz[j];

                vtmp[0]=filter_v[0]+kv1[j];
                vtmp[1]=filter_v[1]+kv2[j];
        }
}/* end if-else */
}/* end for */

/* 'D' updated */
Delem[0]+=(kD11[0] + 2*kD11[1] + 2*kD11[2] + kD11[3])/6;
Delem[1]+=(kD21[0] + 2*kD21[1] + 2*kD21[2] + kD21[3])/6;
Delem[2]+=(kD31[0] + 2*kD31[1] + 2*kD31[2] + kD31[3])/6;
Delem[3]+=(kD12[0] + 2*kD12[1] + 2*kD12[2] + kD12[3])/6;
Delem[4]+=(kD22[0] + 2*kD22[1] + 2*kD22[2] + kD22[3])/6;
Delem[5]+=(kD32[0] + 2*kD32[1] + 2*kD32[2] + kD32[3])/6;

/* Orthogonalization */
norm = sqrt((Delem[0]*Delem[0])+(Delem[1]*Delem[1])+
                (Delem[2]*Delem[2]));
Delem[0]=Delem[0]/norm;
Delem[1]=Delem[1]/norm;
Delem[2]=Delem[2]/norm.

norm = sqrt((Delem[3]*Delem[3])+(Delem[4]*Delem[4])+
                (Delem[5]*Delem[5]));
Delem[3]=Delem[3]/norm;
Delem[4]=Delem[4]/norm;
Delem[5]=Delem[5]/norm;

/* q's updated  */
qpar[0]+=(kq1[0] + 2*kq1[1] + 2*kq1[2] + kq1[3])/6;
qpar[1]+=(kq2[0] + 2*kq2[1] + 2*kq2[2] + kq2[3])/6;
qpar[2]+=(kq3[0] + 2*kq3[1] + 2*kq3[2] + kq3[3])/6;
qpar[3]+=(kq4[0] + 2*kq4[1] + 2*kq4[2] + kq4[3])/6;
```

```
                    /* Orthogonalization */
                    norm = sqrt((qpar[0]*qpar[0])+(qpar[1]*qpar[1])+(qpar[2]*qpar[2])+
                                (qpar[3]*qpar[3]));
                    qpar[0]=qpar[0]/norm;
                    qpar[1]=qpar[1]/norm;
                    qpar[2]=qpar[2]/norm;
                    qpar[3]=qpar[3]/norm;

                    /* Update true velocities */
                    Velocity[0]+=(kVx[0] + 2*kVx[1] + 2*kVx[2] + kVx[3])/6;
                    Velocity[1]+=(kVy[0] + 2*kVy[1] + 2*kVy[2] + kVy[3])/6;
                    Velocity[2]+=(kVz[0] + 2*kVz[1] + 2*kVz[2] + kVz[3])/6;

                    /* Update filtered velocities */
                    filter_v[0]+=(kv1[0] + 2*kv1[1] + 2*kv1[2] + kv1[3])/6;
                    filter_v[1]+=(kv2[0] + 2*kv2[1] + 2*kv2[2] + kv2[3])/6;

                    for (i1=0;i1<6;i1++) Dtmp[i1]=Delem[i1];
                    for (i1=0;i1<4;i1++) qtmp[i1]=qpar[i1];
                    for (i1=0;i1<3;i1++) Vel[i1]=Velocity[i1];
                    for (i1=0;i1<2;i1++) vtmp[i1]=filter_v[i1];
                }/* end for */
}/* end RK4_Routines */
```

END

16-11-95

FIN