IMPROVING THE QUALITY CONTROL OF MARINE GEOPHYSICAL TRACKLINE DATA

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAIʻI IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

GEOLOGY AND GEOPHYSICS

MAY 2006

By
Michael T. Chandler

Thesis Committee:

Paul Wessel, Chairperson
L. Neil Frazer
Fernando Martinez

We certify that we have read this thesis and that, in our opinion, it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Geology and Geophysics.

THESIS COMMITTEE

_____

Chairperson

_____

_____

Abstract

I have examined each of the National Geophysical Data Center's 4,917 trackline geophysics cruises using new error checking methods developed in this research. Each cruise was checked for unreasonable observations, excessive gradients, and for agreement with satellite altimetry-derived gravity and predicted bathymetry grids. Thresholds for error checking were determined through inspection of histograms for all geophysical values, gradients, and grid offsets. Robust regression was used to further scrutinize ship data in comparing bathymetry and free-air anomalies to global grids. I found recurring error types in the NGDC archive including poor navigation, inappropriately scaled data, DC shifted gravity, excessive gradients, and extended depth and gravity offsets from global grids. To enable the removal of gross errors without over-writing original cruise data, I developed an errata system that concisely reports all errors encountered in a cruise. With these errata files, scientists may share cruise corrections, thereby preventing redundant processing. I have implemented these quality control methods into an open-source command-line program called the "mgd77sniffer" which is currently being distributed in the new mgd77 supplement to the popular Generic Mapping Tools software suite.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

The National Geophysical Data Center (NGDC) in Boulder, Colorado, preserves and disseminates publicly funded marine geophysical information collected around the world. Among the many databases maintained at NGDC, the Marine Trackline Geophysics archive is of particular importance. Archived cruises contain differing combinations of bathymetry, gravity and magnetic data, yielding a current total of 64 million data records from around the globe (see Figure 1.1). As of May, 2006, this archive contains 4,917 cruises collected by 53 large academic and government institutions from the United States and abroad (Figure 1.2).

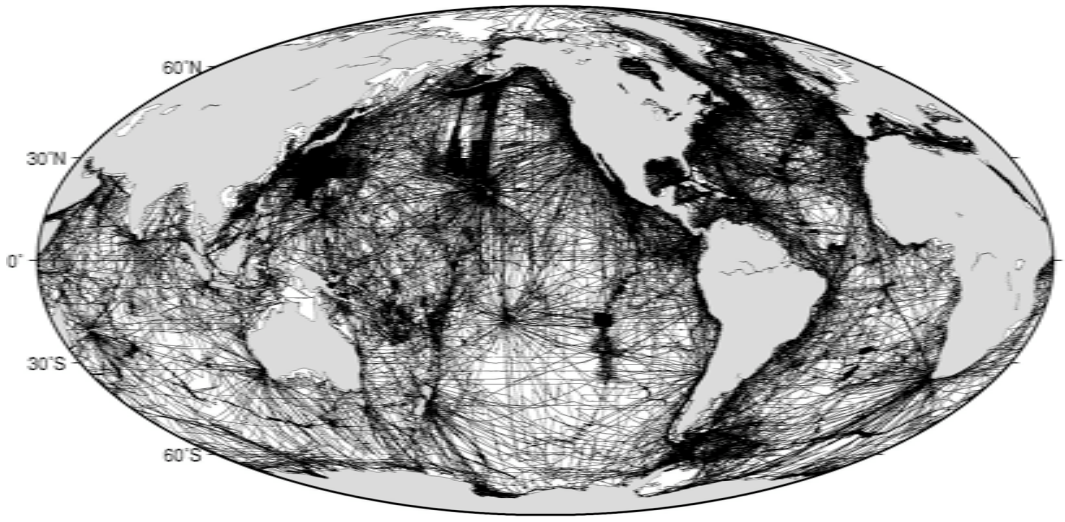


Figure 1.1: Distribution of depth soundings currently archived at NGDC showing roughly 43 million records on a Mollweide Equal-Area projection.

Found within the trackline archive are decades of seagoing research which have brought seminal contributions to Earth science. For example, geomagnetic data gathered by Scripps Institution of Oceanography beginning in 1953 unexpectedly revealed magnetic anomalies arranged in large-scale striped patterns in the northeast Pacific ocean [*Raff and Mason*, 1961]. A subsequent survey by the University of Cambridge

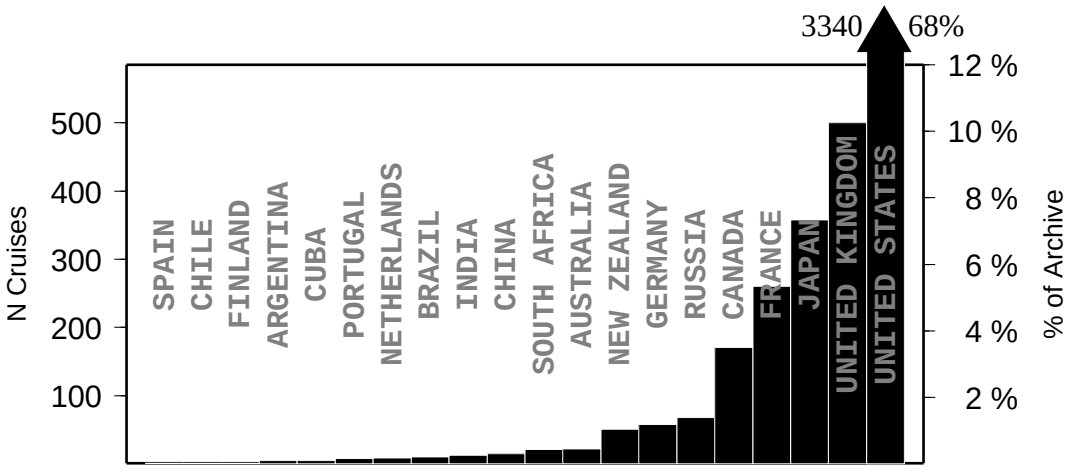


Figure 1.2: Histogram of submitted cruises arranged by nationality, with roughly one-third of contributions originating from foreign sources. 0.6% of cruises originate from Argentina, Chile, Cuba, Finland, Netherlands, Portugal, and Spain combined, with each nation contributing fewer than ten cruises.

in 1961 prompted *Vine and Matthews* [1963] to theorize that seafloor spreading and Earth's periodically reversing magnetic field combine to produce magnetic striping. This radical idea gained credence in 1965 when the U.S.S. ELTANIN collected a series of magnetic profiles (Figure 1.3) across the Pacific-Antarctic Ridge showing clear symmetry of magnetic anomalies about the ridge axis [*Pitman and Heirtzler*, 1966]. The detailed mapping of magnetic field reversals enabled scientists to estimate the age of oceanic lithosphere.

Early bathymetric surveys by Lamont-Doherty Earth Observatory starting in 1955 aboard the R/V VEMA mapped submarine mountain ranges, trenches, and fracture zones providing additional evidence for the then controversial idea that the seafloor is dynamically transforming through time. Maps based on ship soundings (e.g. Figure 1.4) conveyed exciting discoveries to the broader community [*Heezen and Tharp*, 1977]. Gravimeters adapted for use at sea [*Lacoste*, 1959; *Graf and Schultze*, 1961] further explained the structure of oceanic crust by contributing crustal density and isostatic information. These early expeditions, in concert with paleomagnetic observations [*Runcorn*, 1956] and the improved global coverage of seismicity [*Sykes*, 1963],

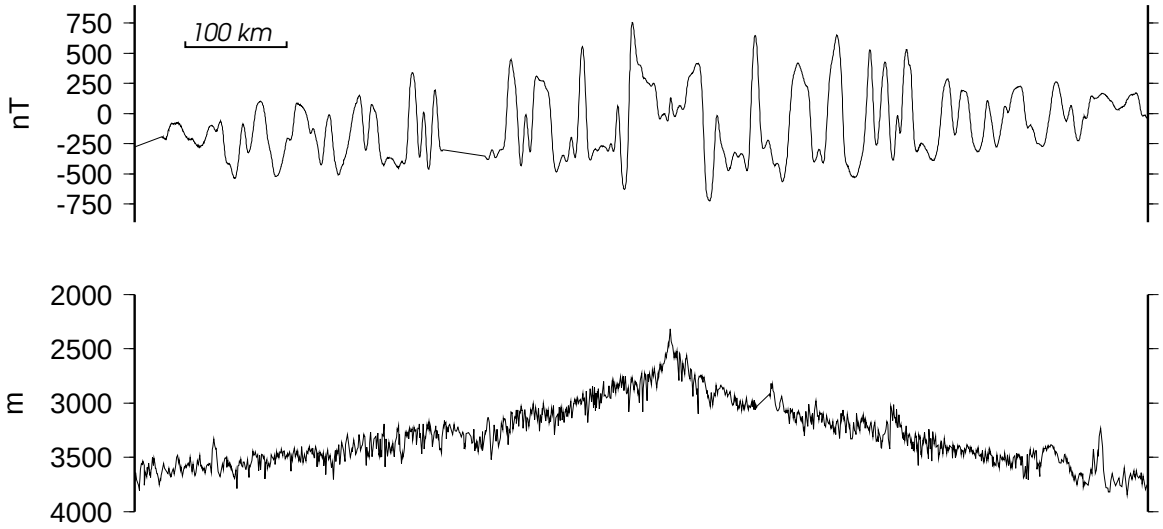


Figure 1.3: The famous ELTANIN 19 magnetic anomalies (top) and bathymetry (bottom) collected in the South Pacific in 1965 (NGDC id 01020019). Profiles run chronologically along the ship track (reproduced from *Pitman and Heirtzler* [1966]).

transformed geology and provided the key information that allowed the earlier continental drift hypothesis to evolve into the modern plate tectonic theory [*Oreskes*, 2002].

This improved understanding of the Earth has come at a cost to taxpayers that is difficult to assess accurately. I derive daily survey costs from ship operation cost projections for 2006 [*Prince*, 2005] and a crude science party cost estimate. Ship operating costs for 2006 are shown in Table 1.1 and are preliminary. Costs for scientific support, including salary paid for survey planning, execution, and post-processing, vary widely, but a rule of thumb (B. Taylor and J. M. Prince, pers. comm. 2005) suggests that science party costs tend to mimic ship operating costs. Based on the 2006 UNOLS report, 41% of sea days are scheduled for large vessels, while 25%, 19%, and 15% are allotted to intermediate, regional, and small vessels respectively. Extrapolating these proportions to the 171,809 sea days on record yields my estimate of 6.3 billion dollars for the accumulated taxpayer investment in marine geophysical research, reported in 2006 USD.

Whereas shipboard geophysics has contributed significantly to marine geology,

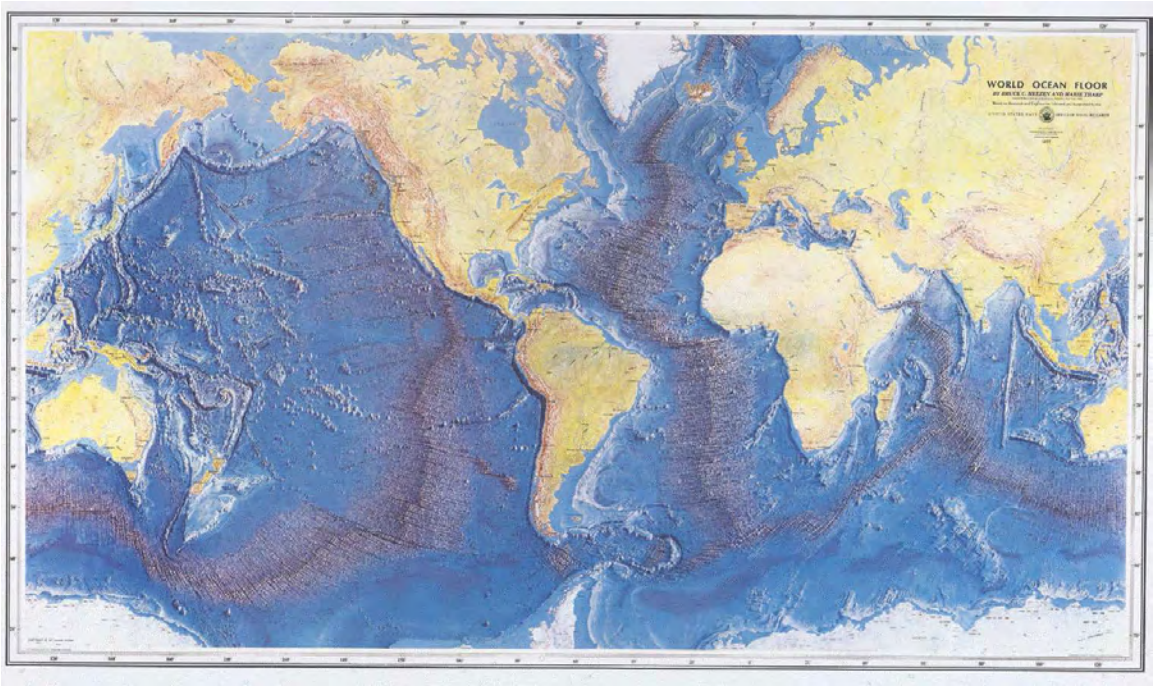


Figure 1.4: The World Ocean Floor map showcased 30 years of seagoing research [*Heezen and Tharp*, 1977].

Table 1.1: Median ship length, cost per day and scheduled sea days for 2006.

Vessel Class	Length (ft)	Cost per Day	Survey Days	Proportion %
Large/Global	274	$27,852	268	41%
Intermediate/Ocean	184	$16,078	162	25%
Regional	123	$11,522	122	19%
Small/Local	92	$4,800	95	15%

some may question their continued relevance given the availability of global grids derived from satellite altimetry missions such as Seasat (e.g. *Haxby et al.* [1983]) and more recently Geosat/ERS-1 (e.g. *Sandwell and Smith* [1997]), yet many geological phenomena remain resolvable only at the finer resolutions afforded by seagoing research. Features with length scales less than 15 km are not well expressed in altimetric data, making the characterization of abyssal hill fabric [*Goff and Smith*, 2004], sharp tectonic scars such as pseudo-faults and propagating rifts [*Hey*, 1977], and small seamounts from altimetry problematic [*Wessel*, 2001]. Bathymetry grids, in particular, rely upon high-density trackline data for accurate bathymetric prediction [*Smith and Sandwell*, 1994, 1997]. Finally, there is no satellite-based technique capable of resolving the very short-wavelength magnetic anomalies that reflect the reversals in the Earth's magnetic field frozen into the oceanic crust.

Vogt et al. [2000] proposed that deep oceans (deeper than 500 m) be mapped to the same spatial resolution (100 m) as other terrestrial planets have been mapped. This ambitious project, if undertaken, would require approximately 20 to 30 years to achieve the necessary coverage at an estimated cost of $10 to $20 billion. These estimates, however, ensure the continued use of current ship data and satellite grids for decades to come. Therefore, the archived data remain very valuable and we should make the utmost effort to optimize the use of these data.

1.1 Background

Trackline geophysical data are commonly used for mapping and interpreting seafloor structure from a plate tectonics perspective. Further undocumented use is widespread as researchers use existing track data to obtain background for survey planning, enabling both the identification of areas of sparse exploration as well as permitting geophysical background and reconnaissance. Development of satellite-derived global

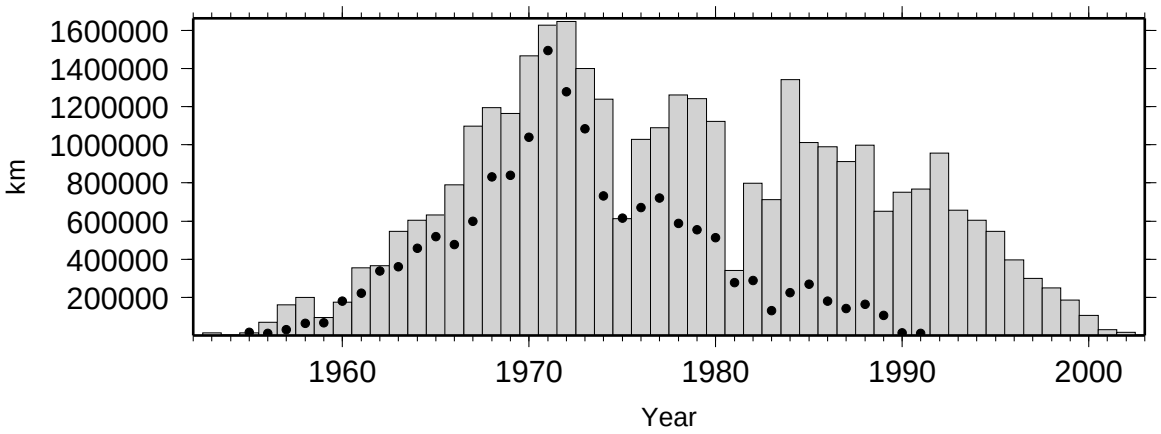


Figure 1.5: Survey distance by year (34,537,200 km total) shown by gray bins with black points showing trackline distance by year (15,771,094 km total) reported by *Smith* [1993]. Both tallies show a similar pattern of decline for more recent years, indicating submittal back-log.

gravity and bathymetry grids furthered the use of archived data on a global scale. For example, shipboard gravity is an essential reference by which satellite-derived gravity anomalies are checked. Perhaps the most exhaustive use of trackline data to date occurred in the compilation of predictive bathymetry grids, in which shipboard bathymetry served to constrain the predicted bathymetry model obtained by transforming satellite-derived gravity anomalies [*Smith and Sandwell*, 1994; *Calmant et al.*, 2002].

Researchers utilizing trackline geophysics have, unfortunately, happened upon gross errors in the archive. Data quality problems are so pervasive that experienced researchers examine each trackline cruise prior to rigorous use. Prior investigations into the accuracy of shipboard gravity and bathymetry focused on discrepancies of measurements at locations where ship-tracks intersect. Utilizing the errors detected at ship crossing points, called crossover errors (COE), these studies identified error sources in trackline data. To date, no global analysis of magnetic crossover errors has been completed.

Wessel and Watts [1988] identified primary error sources for marine gravity measurements including gravimeter drift and DC offset, incomplete Eötvös corrections,

as well as poor navigation. By correcting for drift and DC shift, Wessel and Watts were able to reduce the rms of gravity COEs by 40%.

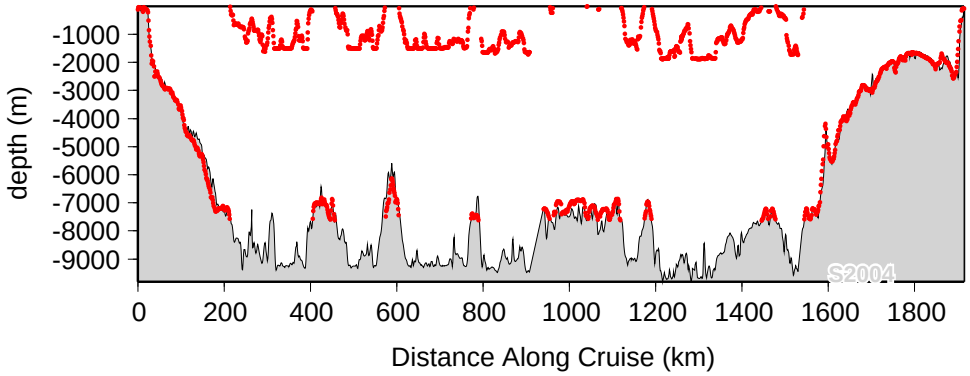


Figure 1.6: Bathymetry profile showing data collected by the Scripps Institution of Oceanography's R/V ARGO off Tokyo, Japan in 1966. Cruise data in red show extreme disparity from Smith and Sandwell predicted bathymetry at depths greater than 7,500 meters. Incorrect operation of the onboard Precision Depth Recorder system and inadequate post-processing by the source institution allowed the submittal of cruise ZTES4BAR (NGDC id 15010076).

Smith [1993] checked bathymetry measurements at trackline intersections both internally, when cruises cross their own track, and externally, when intersecting other cruise tracks. Through internal COE analysis, Smith found that errors in digitizing two-way travel time affected 5% of cruises. Figure 1.6 exhibits a case in which operators of the onboard Precision Depth Recorder failed to adjust the instrument to account for depths beyond 7,500 meters, allowing the instrument to report depths offset by multiples of 1,500 meters from true values. Smith's external COE analysis found 89 cruises having incorrectly scaled bathymetry due to incorrect units and travel-time correctors. Figure 1.7 illustrates a case in which ship bathymetry was reported in fathoms rather than meters as required by NGDC.

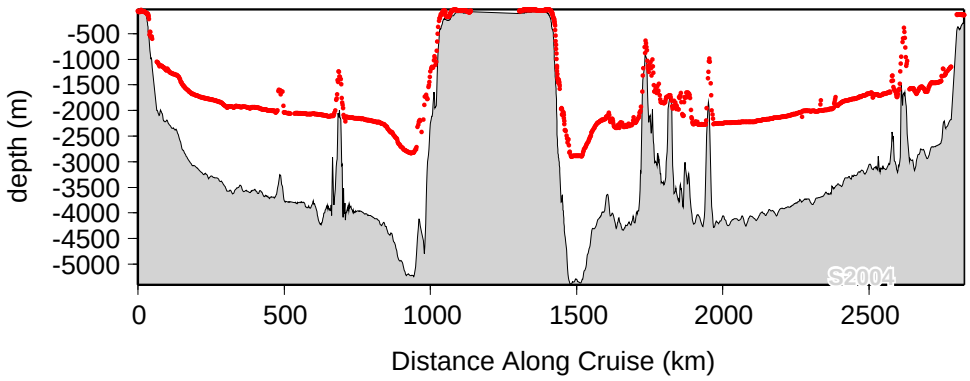


Figure 1.7: Bathymetry profile showing NOAA's PATHFINDER cruise POL7106/OPR 394 (NGDC id 03050023) surveyed off Kodiak Island, USA in 1971. Cruise data (red) are systematically offset by a factor of 0.54 from S2004 gridded topography (grey) suggesting that cruise 03050023 falsely reports depth in fathoms.

1.2 Problem to be addressed

The presence of raw, unprocessed data archived at the National Geophysical Data Center has long been apparent. While scientists certainly process shipboard data in support of individual research projects, there is no stringent quality control in place to ensure that processed records are actually submitted for archival. Many source institutions lack funds, personnel, and original meta-data necessary to clean and re-submit historic cruise data. Anecdotal evidence suggests that many users are weary of the quality of the NGDC data archive. For example, Mr. Graeme Potter of the United Kingdom Hydrographic Office provides this assessment of archived NGDC trackline data:

"There are quite a few problems with some data at NGDC; some of it is completely raw; some whole datasets we have found to be completely unusable. I would not recommend anyone to use the data without first individually inspecting each and every cruise. ... Although there are problems with using data from NGDC, that does not take anything away from what is a databank of immense value. In remote parts of the world the UKHO relies heavily on such sources to populate navigational charts. We

have many charts that would have large blank areas without such data."

The inspection of "each and every cruise" mentioned above is a process repeated by other scientists utilizing NGDC trackline data. There is currently no public mechanism for sharing data corrections, causing duplication of data processing. Even basic quality control measures at NGDC fail to detect crude errors, allowing a wide range of problems to persist in the archive. Previous studies focusing on data quality issues at NGDC did not propagate their findings to NGDC. This is, in part, due to NGDC's role as a data library, which protects original data by limiting data revision to source institutions. Inexperienced or naive users are less likely to undertake their own rigorous quality assessment and may consequently derive incorrect statistical results or prepare flawed graphical depictions of tectonic features.

1.3 Proposed solution

Three avenues exist for quality control of marine geophysical trackline data including self-consistency checks, comparisons with gridded data, and crossover tests. As cited above, previous studies focused on crossover error analyses for shipboard gravity and bathymetry. Due to time constraints, I omit crossover tests noting previous work in this area and instead check each cruise for self-consistency and compare ship gravity and bathymetry with global grids. Checking individual cruises for self-consistency enables direct detection of data problems and therefore should precede further processing steps such as an exhaustive, global crossover analysis.

In this thesis, I develop self-consistency and global grid comparison methods for the improved detection of problematic cruise data which I then apply to the current 4,917 cruise NGDC archive. I obtain, as a result, distributions of all geophysical fields currently archived at NGDC, as well as error detection limits for both grid comparison and internal error checking. I compile these methods into a computer

program (named "mgd77sniffer") usable by NGDC and other survey institutions for quick detection of problematic cruise data.

Finally, I initiate an errata system by which flagged errors may be stored and corrections applied to raw cruise data to generate corrected trackline data. This errata system enables data correction without modification of original cruise data. In this manner, correction terms, as well as information on which data values are erroneous and should be disregarded, may be shared amongst scientists to limit processing duplication.

Chapter 2

TESTING FOR INTERNAL CONSISTENCY

The National Geophysical Data Center archives trackline geophysics data in the Marine Geophysical Data Exchange (MGD77) format [*Hittleman et al.*, 1977]. This format specifies 24 header rows containing cruise descriptions and an unrestricted number of data records each having 24 numeric and three text fields. Each data record may include time, navigation, bathymetry, magnetics, gravity, and seismic line information. Different levels of scrutiny are available to each MGD77 field (see Table 2.1). Text fields store cruise, seismic line, and shot-point identification labels; these are not examined. All numeric fields are subject to basic "sanity" checking while geophysical measurements undergo more rigorous tests.

Trackline geophysics data are available through the National Geophysical Data Center's Geophysical Data System (GEODAS) either via online data download or by ordering Marine Trackline CDs or DVDs. For this study, data are downloaded from a GEODAS version 4.1.18 DVD. Additional data are downloaded through NGDC's online GEODAS website periodically as NGDC adds new cruises to the archive. During data download from NGDC, I specified that corrected depths be computed by applying sound velocity correctors using Carter Tables. Bathymetry for cruises lacking two-way travel time were left as is.

A lack of suitable public MGD77 format decoding tools required first the development of basic MGD77 input and output utilities. This endeavor built largely upon the "MGG" supplemental Generic Mapping Tools package developed by *Wessel and Smith* [1998]. Our updated MGD77 decoding tools allow examination of all MGD77 data fields and are publicly available as part of the new "MGD77" supplemental GMT

package, described elsewhere [*Wessel and Chandler*, 2006].

Satellite altimetry-derived gravity grids (e.g. [*Haxby et al.*, 1983] and [*Sandwell and Smith*, 1997]) and their predictive bathymetry counterparts ([*Smith and Sandwell*, 1994, 1997]) provide global coverage at a resolution adequate for comparison with cruise data. The satellite-derived marine gravity anomaly, in particular, serves as an independent reference for comparison with ship free-air gravity. However, the large number of trackline cruises used as constraints for compiling predictive bathymetry grids reduces the grid's independence as a reference for shipboard bathymetry comparison. Nevertheless, comparison of trackline bathymetry with predictive bathymetry grids remains useful for the many omitted erroneous cruises and recent cruises unavailable at the time of grid compilation.

A recent evaluation of global bathymetry grids by *Marks and Smith* [2006] aided in the selection of the 1 minute-resolution Smith and Sandwell blend of GEBCO and predicted topography (here called S2004) as a reference for trackline bathymetry comparison. Shipboard gravity measurements are compared against the 1 minute resolution Sandwell and Smith version 15.1 marine gravity anomaly grid [*Sandwell and Smith*, 1997]. As mentioned, no global magnetic grid suitable for comparison with shipboard measurements exists.

All error checking performed here fall into two categories: along-track analysis of each data record, and systematic testing of whole cruises. Along-track tests include basic sanity testing (i.e., checking the range of data) on a row-by-row basis, calculation of gradients along-track, and along-track comparison of ship gravity and bathymetry with global grids. Analysis of entire cruises involves checking ship data against global grids using regression analysis as well as checking for correct magnetic and gravity reference fields and two-way travel time correction codes.

2.1 Along-track Tests

Scanning marine geophysical data along-track reveals a multitude of errors and is useful for locating problematic areas within a cruise. Software was written for examining millions of data records one record at a time, checking data for sanity, excessive slopes along-track, and looking for excessive gravity and bathymetry offsets from global grids.

2.1.1 Sanity

As shown in Table 2.1, all numerical data fields are checked for basic sanity. Focusing on one record at a time, data fields are flagged as erroneous if they exceed reasonable lower and upper limits. Time and position have *a priori* boundaries defined by NGDC. For example, time is calculated from the six MGD77 time components, checked to be increasing, and tested to range between 1939 and the present. Likewise, latitude and longitude are confined to within $\pm$ 90° and $\pm$ 180°, respectively. Similarly, all MGD77 components of type "int" (such as processing flags and codes) are confined to specific values or to be given within specified boundaries by NGDC.

Such elementary testing is imperative given that there are 24 numeric fields for each of the current 64 million archived records. The geophysical fields of type float do not have a priori boundaries and therefore must be experimentally determined. Due to the massive investment by researchers and taxpayers to develop this archive, effort is directed toward deriving reasonable boundaries from the data themselves. This entails compiling values from each geophysical field for all NGDC archived cruises whereby a reasonable set of boundaries is determined through simple statistics and visual interpretation of histograms.

Table 2.1: MGD77 data components and tests

MGD77 Field	Abbrev.	Type	Units	Sanity	Gradient	Grid
Data Record Type	drt	int	-	X	-	-
Time-zone Correction	tz	int	h	X	-	-
Year	-	int	a	X	-	-
Month	-	int	mo	X	-	-
Day	-	int	d	X	-	-
Hour	-	int	h	X	-	-
Minutes	-	int	min	X	-	-
Latitude	lat	float	degree	X	X	-
Longitude	lon	float	degree	X	X	-
Position Type Code	ptc	int	-	X	-	-
Two-way Travel Time	twt	float	sec	X	X	-
Corrected Depth	depth	float	m	X	X	X
Bathymetric Correction Code	bcc	int	-	X	-	-
Bathymetric Type Code	btc	int	-	X	-	-
Magnetics Total Field 1	mtf1	float	nT	X	X	-
Magnetics Total Field 2	mtf2	float	nT	X	X	-
Magnetics Residual Field	mag	float	nT	X	X	-
Sensor for Residual Field	msens	int	-	X	-	-
Magnetics Diurnal Correction	diur	float	nT	X	X	-
Magnetic Sensor Depth	msd	float	m	X	X	-
Observed Gravity	gobs	float	mGal	X	X	-
Eötvös Correction	eot	float	mGal	X	X	-
Free Air Anomaly	faa	float	mGal	X	X	X
Navigation Quality Code	nqc	int	-	X	-	-
Survey Identifier	id	text	-	-	-	-
Seismic Line Number	ssln	text	-	-	-	-
Seismic Shot-point Number	sspn	text	-	-	-	-

2.1.2 Gradient

I further analyze geophysical observations along-track by checking for excessive changes in geophysical values. Assuming a predominant continuity in the geopotential fields and the approximate continuity of bathymetry (e.g., fracture zones are not mathematical step functions but reflect rapid change in depth over a small but finite distance), I expect a natural ceiling for valid gradients.

To test this assumption, I calculate spatial gradients with respect to distance (dz/ds). For each gradient calculation, the current geophysical field is compared with the same field from the previous record. Often I look further than the previous record, as data-contributing institutions tend to report some fields intermittently based upon sampling rates of different instruments. I then compute the distance between the two field measurements in order to calculate dz/ds.

Gradients are calculated for all NGDC trackline geophysical fields to obtain an improved understanding of the nature of along-track gradients as well as for establishing well-founded gradient limits. Ship velocity is also computed and checked for sanity. Error sources for ship velocity include bad position coordinates, bad time increments, or both.

2.1.3 Grid offset

Bathymetry and free-air gravity are further scrutinized through detection of localized offsets from global grids along-track. For both bathymetry and gravity, the relevant grid is sampled at each cruise position, resulting in an ordered series of ship and grid z-value pairs. Differences are calculated between each ship/grid pair by subtracting grid values from each ship value. Integrating these differences with respect to distance along-track approximates offset area between ship and grid data. Distinct offsets from grids are characterized by constant offset sign; positive while ship values are greater than grid values and negative for the opposite case (see Figure 2.1). The

longer and more extreme the offset the greater the integrated offset area becomes. By running this algorithm on the entire set of bathymetry and gravity cruises, an appreciation for significant offset area is obtained through visual interpretation of offset area histograms.

Monitoring data offsets from grids along-track helps detect intermediate length errors (i.e., longer than a few records but not errors affecting an entire cruise). These types of problems are fairly common in geophysics. For instance, single beam echo sounders often lose bottom tracking for several minutes or more.

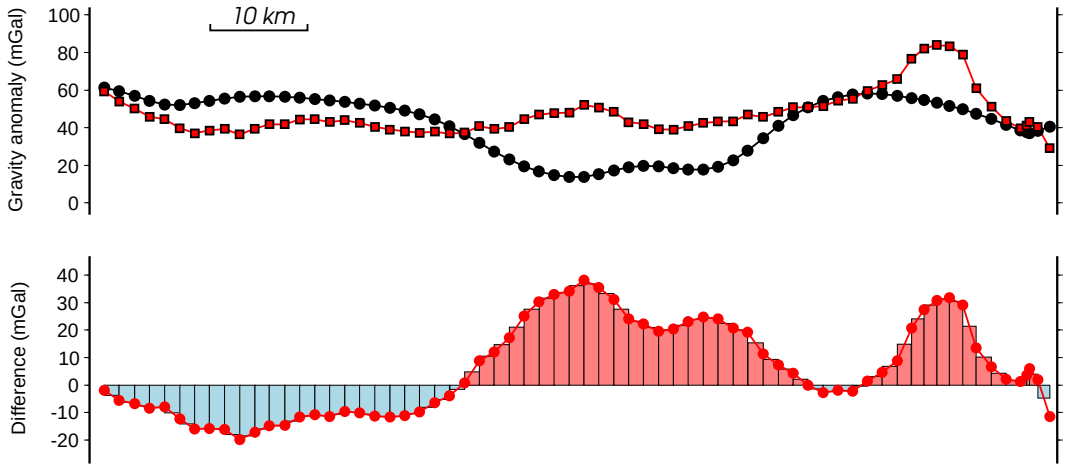


Figure 2.1: (top) Gravity profile showing free-air anomalies from Geological Survey of Japan's GH7902 cruise (NGDC id J2010013) (red squares) plotted with grid values extracted at each ship location (black circles). (bottom) The area between the ship and grid curves is estimated by summing discrete rectangles (i.e., blue or red rectangles). Offset area accumulates until the ship and grid profiles intersect signaling the start of a new offset.

2.2 Systematic Tests

The detection of problems affecting entire cruises requires additional testing. Such problems include instrument drift, incorrect data scaling or units, invalid travel-time correctors, incorrect magnetic or gravity reference fields, and systematically offset gravity data. I locate these errors through examination of the MGD77 header and

through regression analysis of ship and grid data.

2.2.1 MGD77 header analysis

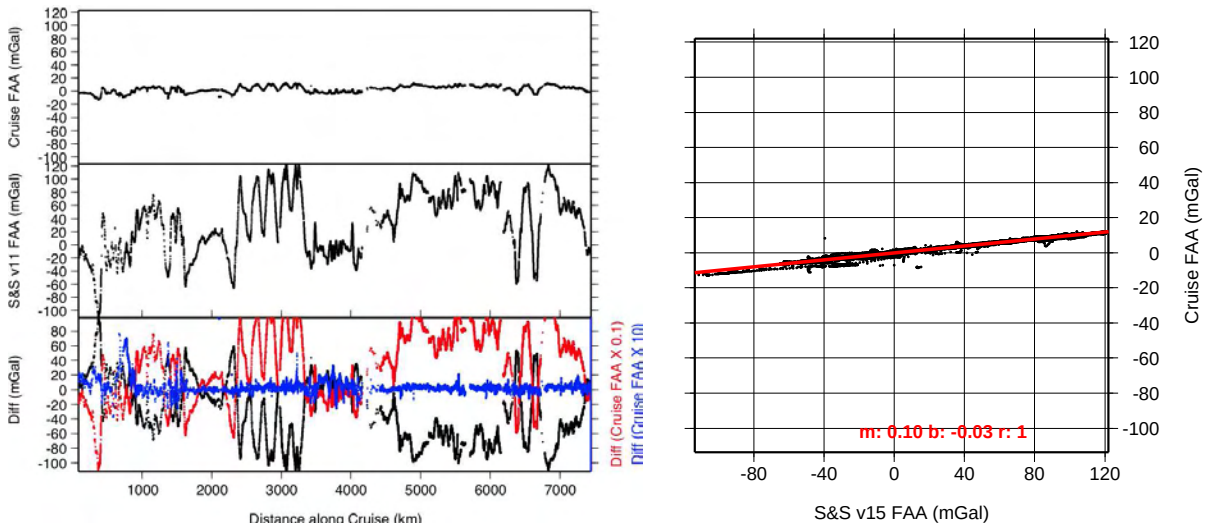
The MGD77 header consists of 24 header sequences and 72 header fields containing information applicable to an entire cruise. These fields are reviewed closely to verify the presence of essential information and to enforce MGD77 format specifications. Further scrutiny is undertaken to ensure that the appropriate magnetic and gravity reference field models are used for calculating anomalies. This involves simply checking that a survey falls within the valid time period of the specified model.

2.2.2 Comparison with gridded data

I utilize robust regression techniques to test how well ship bathymetry and gravity correlate with global grids. Least Median of Squares (LMS) regression determines the model that best relates ship and grid data by identifying outliers and assigning them zero weight for subsequent Re-Weighted Least Squares regression [*Rousseuw and Leroy*, 1986]. Once outliers are assigned zero weight, ordinary least squares regression is used to calculate regression slope, intercept, standard deviation, and correlation [*Draper and Smith*, 1998]. This process improves regression by eliminating the problems associated with least squares regression on raw data which typically contains outliers. Figure 2.2 illustrates the presence and detection of incorrectly scaled ship data.

2.3 Error Handling

The above error detection methods quickly identify extreme data problems so that data analysts at source institutions and at NGDC may better verify data integrity prior to final archival into NGDC's marine trackline database. These methods are



(a) (b)

Figure 2.2: (a) Intuitive, albeit incorrect, reporting of Free Air gravity in mGal (top) by USGS cruise 06050010 (MGD77 format requires tenths of mGal). Manual comparison of differences between ship and grid data show improvement when ship data are scaled by 10 (blue profile). (b) Scale factor is determined much quicker with RLS regression.

not designed to scrutinize questionable data, but rather to flag only extreme, obvious errors. For the many cruises already present in the NGDC archive that cannot be reprocessed and resubmitted by source institutions, attempts should be made to remove gross outliers. To aid in this endeavor, I have extended the above error detection methods to output error flags into structured errata tables.

This errata format, named "E77", contains a variable length header followed by error records each summarizing errors encountered in one MGD77 record. Information pertaining to an entire cruise, such as NGDC and survey institution identification codes, cruise examination time, two-way travel time corrector information, data precision warnings, as well as systematic scales, and DC shifts from global grid comparisons are reported as E77 header information. Individual error records have a strict format. Included is a time or distance column followed by record number, a formatted error code string, and finally a verbose description of errors detected in the record. Four error classes are encoded into the error code string with different alphabetic characters

representing unique error types. Error records use the following syntax:

```
<time/distance>  <record  number>  <error code string> <verbose description>

Sample:

1971-04-28T18:27:30 345  0-0-OQ-0   gradient: mtf1,  mag  excessive
1971-05-14T18:01:30 5413 0-0-OQ-0    gradient:  mtf1, mag excessive
1971-05-15T03:41:30 5634 0-Q-0-0    value: mag invalid
```

The error code string is comprised of four error classes including navigation errors (NAV), invalid data values (VALUE) (either out of range or flat-lined), excessive data gradients (GRAD), and excessive offsets from grids (GRID). These error classes have alphabetic codes describing unique error types. The NAV error class has unique cases while VALUE, SLOPE, and GRID classes are described by alphabetic characters for each of the 24 numeric fields in MGD77 format order (Table 2.2).

Table 2.2: E77 error classes

```
NAV: 0 - fine                          VALUE:  0 - fine
     A - time out of range                     K - twt invalid
     B - time decreasing                       L - depth invalid
     C - excessive speed                       O - mtf1 invalid
     D - above sea level                       etc.
     E - lat undefined
     F - lon undefined

GRAD:0 - fine                          GRID:   0 - fine
     K - d[twt] excessive                      K - twt offset
     L - d[depth] excessive                    L - depth offset
     O - d[mtf1] excessive                     O - mtf1 offset
     etc.                                      etc.
```

This errata system provides an interface for removing or correcting internally inconsistent trackline data. Future distribution of these tables amongst scientists will aid in the reduction of redundant data processing by trackline data users. Software designed to parse and apply E77 correctors is already available as part of a new mgd77 supplement to the Generic Mapping Tools software suite and is described

elsewhere [*Wessel and Chandler*, 2006]. The generic E77 design, coupled with its public availability and support by other applications may qualify its use by other quality control researchers.

Chapter 3

ANALYSIS OF TRACKLINE GEOPHYSICS

Data processing methods developed in this research are compiled into the "mgd77sniffer" command-line program, written in the POSIX C programming language [*Kernighan and Ritchie*, 1988]. Source code for the mgd77sniffer program is listed in Appendix A.1, with mgd77sniffer documentation listed in Appendix A.2. The mgd77sniffer program is calibrated to locate errors exceeding thresholds also determined in this research. I calibrate the error checking first by determining the variability of geophysical observations within the MGD77 archive.

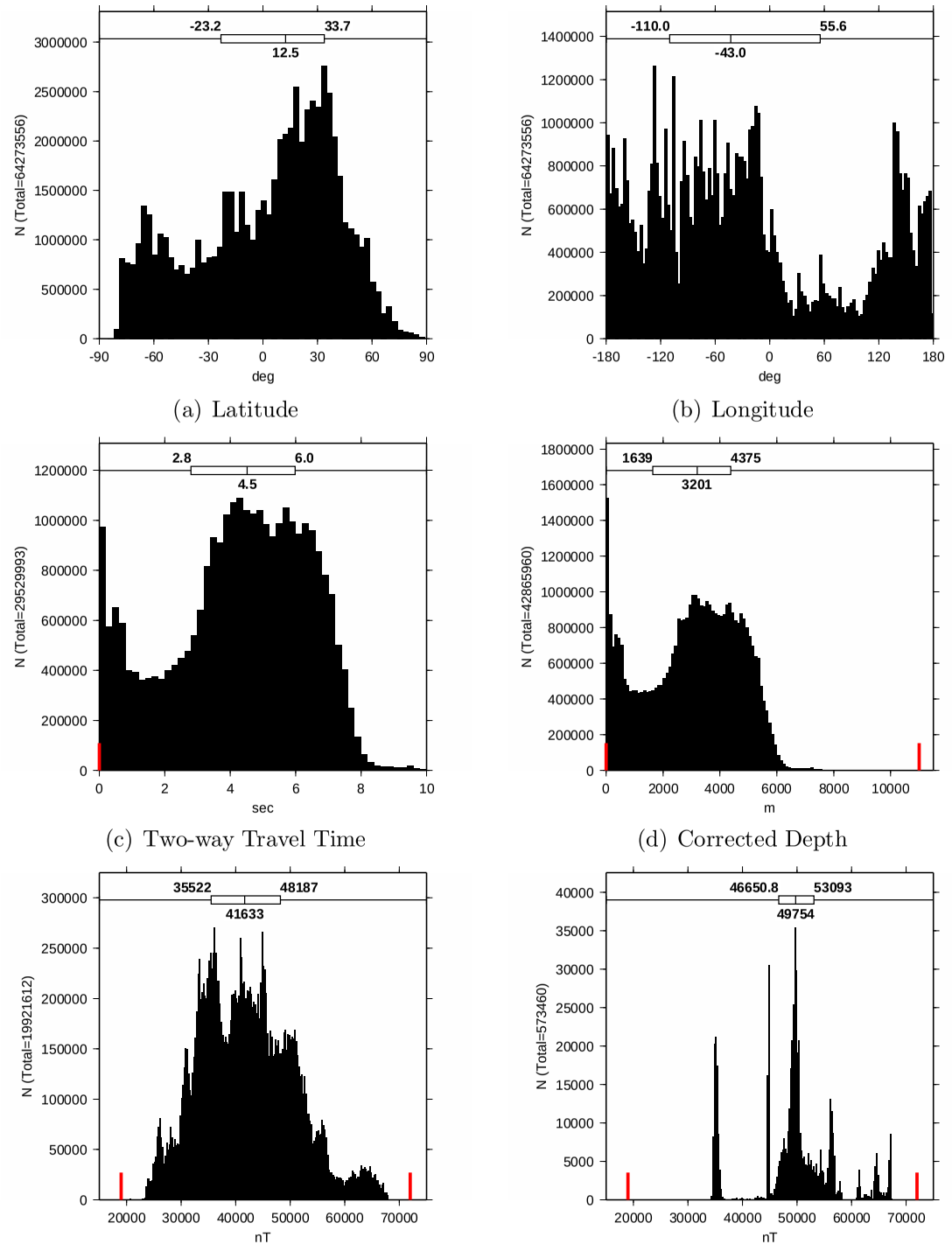
By running the mgd77sniffer program on all archived cruises, geophysical values are decoded from the MGD77 format, records are examined along-track, and cruises are systematically compared to global bathymetry and gravity grids. I analyze this output by generating histograms and x versus y scatter-plots which enable visual exploration of data properties and processing methods.

3.1 Along-track Analysis

3.1.1 Distribution of values

Histograms containing all archived geophysical measurements are generated for the purpose of establishing reasonable limits for shipboard geophysical observations. Figures 3.1 – 3.2 show values for all archived geophysical fields.

Figures 3.1(a) and 3.1(b) indicate an acquisition bias toward the Northern and Western hemispheres (also depicted in Figure 1.1). The African and Eurasian land



(a) Latitude

(b) Longitude

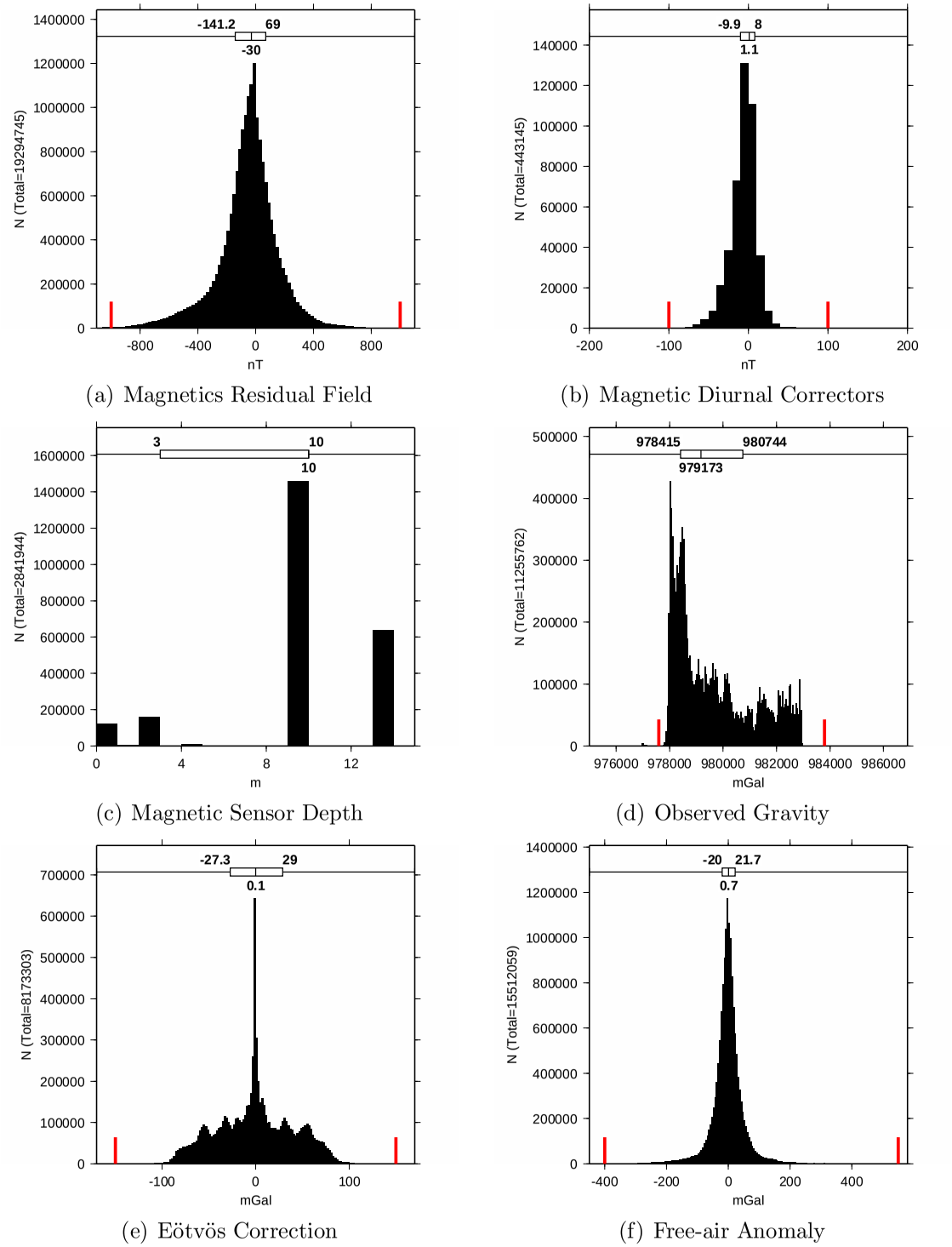
(c) Two-way Travel Time

(d) Corrected Depth

(e) Magnetics Total Field 1

(f) Magnetics Total Field 2

Figure 3.1: Histograms of all archived geophysical observations with 25, 50, and 75 percentiles indicated by box-and-whisker diagrams. Red hash marks indicate thresholds determined in this study.



(a) Magnetics Residual Field

(b) Magnetic Diurnal Correctors

(c) Magnetic Sensor Depth

(d) Observed Gravity

(e) Eötvös Correction

(f) Free-air Anomaly

Figure 3.2: Histograms of all archived geophysical observations (continued) with 25, 50, and 75 percentiles indicated by box-and-whisker diagrams. Red hash marks indicate thresholds determined in this study.

masses account for much of the data deficiency from 0° to 120° longitude. The data gap due to the presence of North and South America is masked by intense acquisition in Central and Western American waters. Figures 3.1(e) and 3.1(f) indicate that magnetic total field intensities largely range from 20,000 to 70,000 nT. Due to sparse total field *2* measurements (i.e. the archive contains 35 times more total field *1* measurements than total field *2*), this study focuses on the first magnetic sensor for total field interpretations. As shown in Figure 3.1(d), bathymetry values range from near zero to a maximum depth of 11,159 meters. This false maximum depth, collected by the *R/V Moana Wave* in 1974, occurs in Hawaiian waters (156.71°W, 19.68°N) where regional water depths do not exceed 7 km. For comparison, the deepest ocean depth yet measured, located at the Challenger Deep in the Mariana Trench, has been measured at 10,924 m by the Hydrographic Department, Maritime Safety Agency of Japan [*Fujioka et al.*, 2002]. Apart from these erroneous values, the remaining data portray the familiar hypsometric curve for bathymetry, with a peak at mean seafloor depth around 4 km. This is also mimicked by two-way travel time in Figure 3.1(c). The observed gravity distribution shown in Figure 3.2(d) indicates a predominance of gravity measurements from tropic or near-tropic latitudes where normal gravity is lower due to Earth's pole-ward ellipsoidal flattening. Figure 3.3 more clearly illustrates this low latitude concentration of gravity measurements. Figure 3.2(e) shows a strong peak at zero for Eötvös corrections indicating considerable data collected while stationary as well as a tendency to survey in North/South directions. Figure 3.4 shows that scientists also commonly collect gravity data heading due East and West and at several prevalent acquisition speeds. Magnetic and free-air gravity anomalies do show unrealistic magnitudes. Free-air anomalies (Figure 3.2(f)) largely vary between ± 250 mGal but contain outliers up to ± 1,000 mGal. By comparison, Sandwell and Smith's version 15.1 gravity anomaly grid varies from -364 to 547 mGal. Similarly, shipboard magnetic residual anomalies (Figure 3.2(a)) generally vary between ± 800

nT, but contain outliers up to ± 10,000 nT.

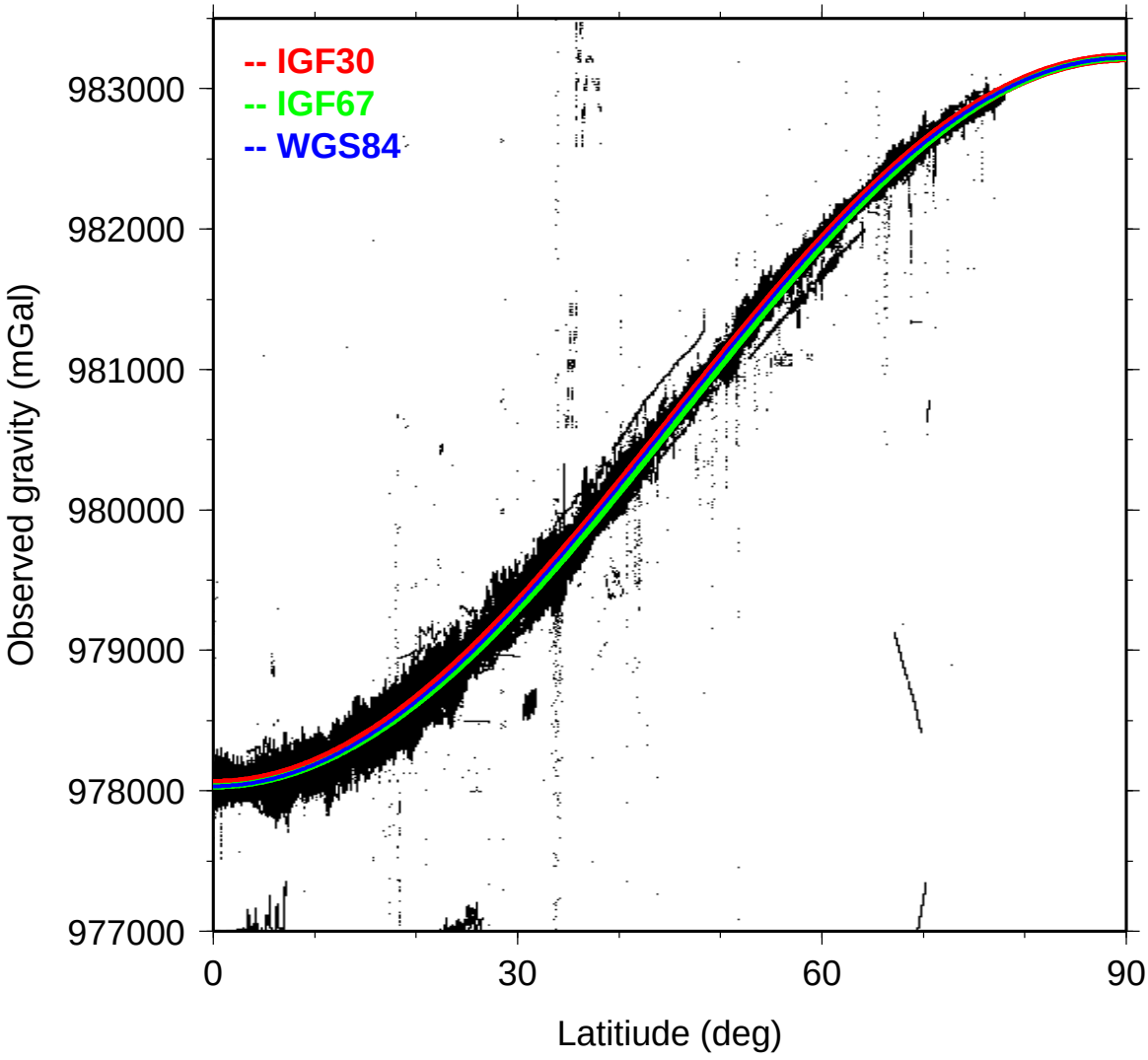


Figure 3.3: Shipboard gravity measurements (black dots) are concentrated in lower latitudes and show first-order agreement with theoretical gravity, though considerable error is apparent.

This marks a recurring trend where total field measurements, such as observed gravity and magnetic total field, have reasonable limits, while derived fields, such as free-air anomaly and magnetic residual anomaly, have outliers that lead to unreasonable limits.

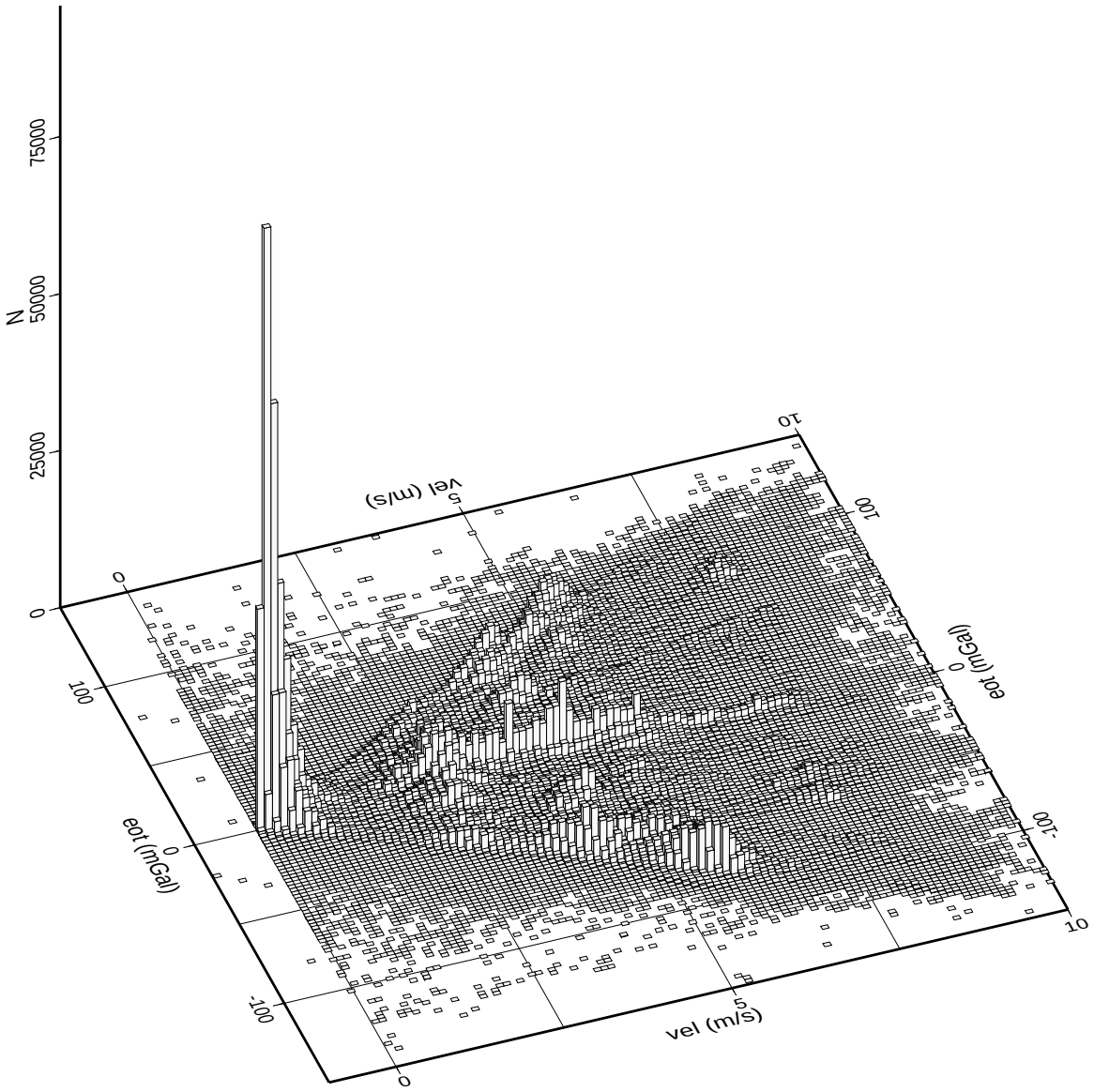


Figure 3.4: Three dimensional histogram showing concentrations of Eötvös corrections at various ship speeds. The strong spike at (0,0) indicates a tendency to collect gravity data while on-station. Prevalent non-zero speeds include ∼2.5 m/s, ∼4.7 m/s, and ∼6 m/s. Prevalent headings include due East (ridge of positive Eötvös corrections), due West (ridge of negative Eötvös corrections), and North/South (Eötvös corrections near zero).

A word about gaps

It became evident in the course of this research that large gaps in time and distance occur within individual cruise files. Time gaps often result from extended sampling intervals, the occasional typo, and from pauses between cruise legs. An extreme artificial time gap occurred in the 1971 NOAA cruise NOSIDOE (NGDC id 03020006) where a single record falsely reported the year as 1961. Distance gaps often result from loss of satellite navigation or from the presence of multiple cruise segments within one submitted cruise. In past studies, scientists used a maximum time gap of 15 minutes to avoid comparing data which may not be continuous (i.e. *Smith* [1993] and *Wessel and Watts* [1988]). Because 14% of cruises lack time information, I use a maximum distance gap between continuous records of 5 km. A gap greater than 5 km is assumed to be a data discontinuity and is thus carefully avoided in calculations.

Time and distance intervals

In addition to time and distance gaps, other troublesome time and distance intervals are encountered. Nearly all records contain positively increasing time, yet 0.2% of records contain duplicate time while an additional 0.24% of records contain decreasing time. Figure 3.5 illustrates the presence of non-increasing and decreasing time increments in the trackline archive. Duplicate and decreasing time records contributed by Minerals Management Service make up 94% of all time errors encountered in this research. Kenneth Piper (*personal comm., 2005*), of the Minerals Management Service, offers the following explanation.

"We suspect that most of your "decreasing time" errors may not be errors. Ship tracklines are defined in advance of the actual survey. Adjacent parallel lines sometimes have shotpoints defined in the same direction and sometimes they are reversed. When they are defined as increasing in the same direction (e.g., all west to east), then usually the shot time data for every other line will look as if it is reversed if the data is

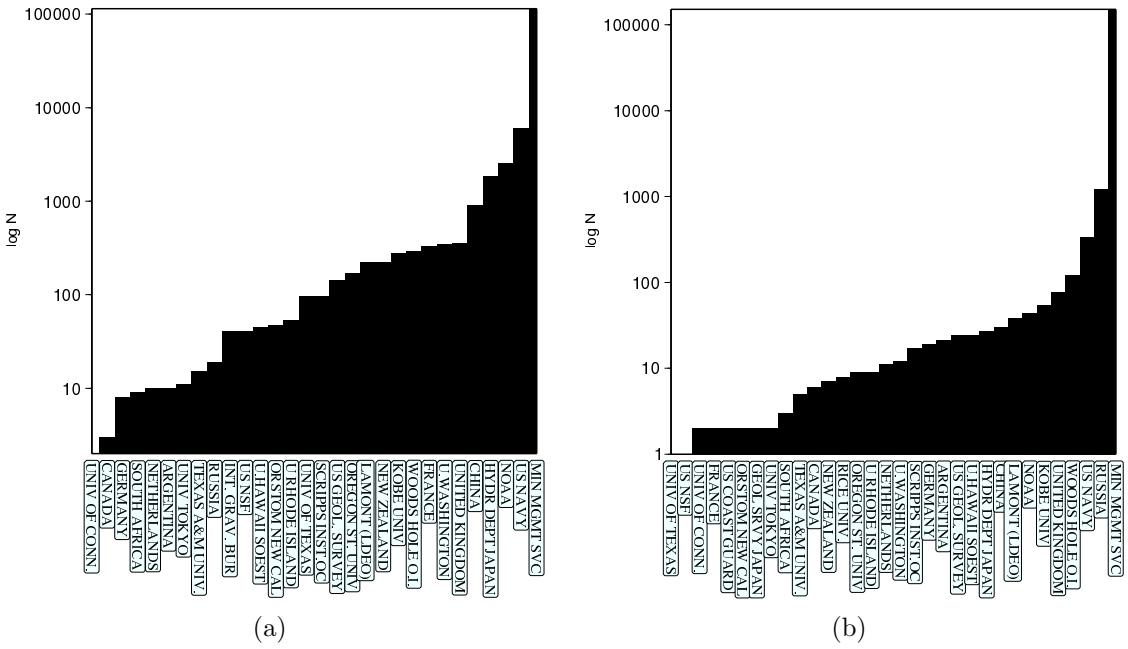


Figure 3.5: (a) Number of non-increasing time increments ($\Delta t = 0$) archived in NGDCs Marine Trackline Geophysics Database arranged by survey institution. Minerals Management Services submitted 114,448 of 128,623 duplicate time records. (b) Number of records having negative time increments by survey institution. Minerals Management Services submitted 151,665 of the total 153,798 negative time records contained in the archive.

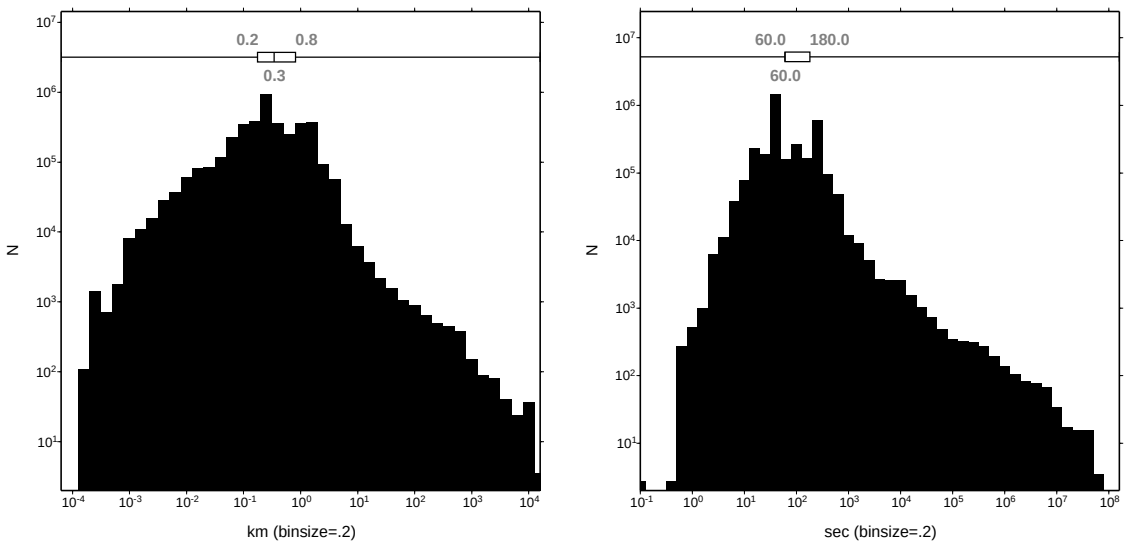
sorted by shotpoint number. That is because the ships turn around and shoot the next adjacent line in the opposite direction. On the original field tape that is recorded in real time, the shotpoints will be reversed. However the processed tape may have been reorganized so that shotpoint numbers will be in increasing order. So then, typically every other line will appear to have decreasing time values. We suspect that is what you are seeing."

While the affected cruises do appear to be sorted according to shot point number rather than by time, the contention that this does not constitute an error is false. The MGD77 format specifies that data records be "sequentially and chronologically organized until the end of the file" [*Hittleman et al.*, 1977]. These erroneous data may be easily improved by re-sorting with respect to time rather than shot point number.

Figure 3.6 shows analyses of all time (Δt) and distance (Δs) increments. The median Δs and Δt describe the average spatial resolution and sample interval for trackline bathymetry. 25% of bathymetry records have sampling lengths less than 200 m, with median Δs of 300m. Excessive Δs reflect large data gaps in the archive. The majority of records report Δt in multiples of one minute. The median Δs and Δt suggest a median ship speed of 5 m/s, very close to the median speed presented later in this thesis.

Timeless cruises

Currently, 695 archived cruises contain no time information other than month and year reported in the MGD77 header (Figure 3.7). The possibly classified nature of these data likely requires the absence of time information, therefore lack of time is not considered an error. This observation prompted the switch from 15 minute time-based gap checking to 5 km distance-based gap checking.



(a) Δs (b) Δt

Figure 3.6: (a) Histogram of distance increments (Δs) for all archived bathymetry with an overlaid box-and-whisker diagram showing 25, 50, and 75 Δs percentiles. (b) Histogram of positive time increments (Δt) for all archived bathymetry showing prevalent sampling intervals of 60 and 180 seconds.

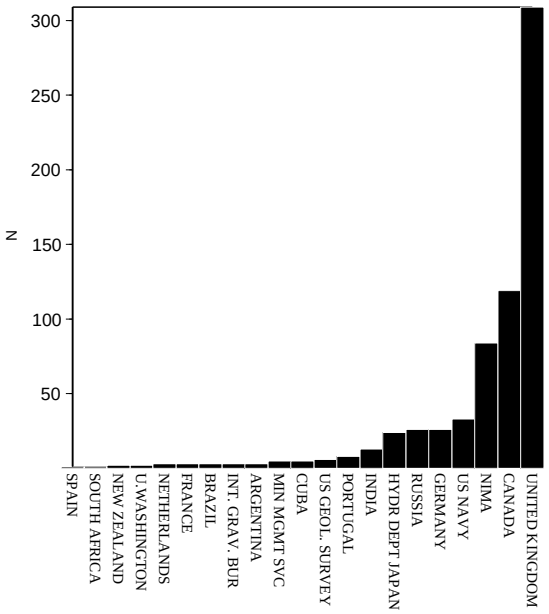
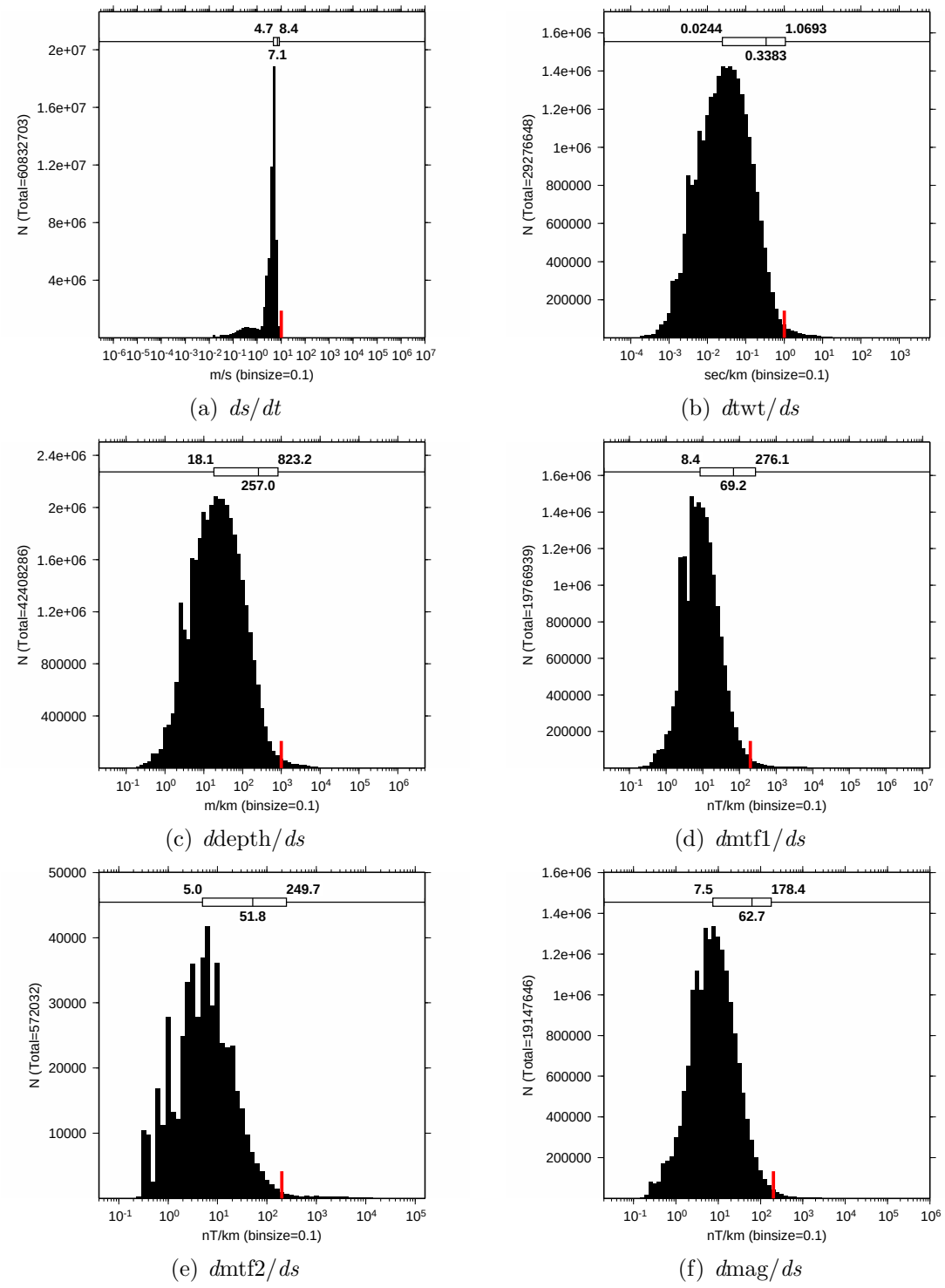


Figure 3.7: Military and government mapping agencies submit a majority of cruises lacking time, shown here arranged by survey institution.

3.1.2 Distribution of gradients

I calculate gradients with respect to distance along-track, and evaluate scale-independent changes in geophysical values along-track for all geophysical cruise data. Due to symmetry about zero for spatial gradients, I analyze gradient absolute values. Due to long-tailed gradient distributions, I plot histograms of geophysical gradients using logarithmic scales. Lognormal histograms for all geophysical gradients, as well as vessel speed, are shown in Figures 3.8 – 3.9.

Figure 3.8(a) shows median ship speed of 4.7 m/s (9.1 knots) with 99% of ship speeds less than 8.4 m/s (16.3 knots). It is notable that extreme ship speeds are observed out to the maximum $1.2e10^7$ m/s ($2.3e10^7$ knots), reported by Texas A&M University's ODP189JR cruise (NGDC id 23060095) surveyed in 2000. Extreme ship speeds are generally the result of large position changes over a small change in time, often caused by loss of satellite navigation. A median depth gradient of 18.1 m/km (Figure 3.8(c)) implies that, on average, water depth changes by 18.1 m per 1 km horizontal, which translates to an angle of 1.0°. 99% of bathymetry gradients are less than 823.2 m/km, or 39.5°. Residual magnetic gradients primarily vary from 0.2 to 200 nT/km, while most free-air anomaly gradients range from 0.1 to 100 mGal/km.

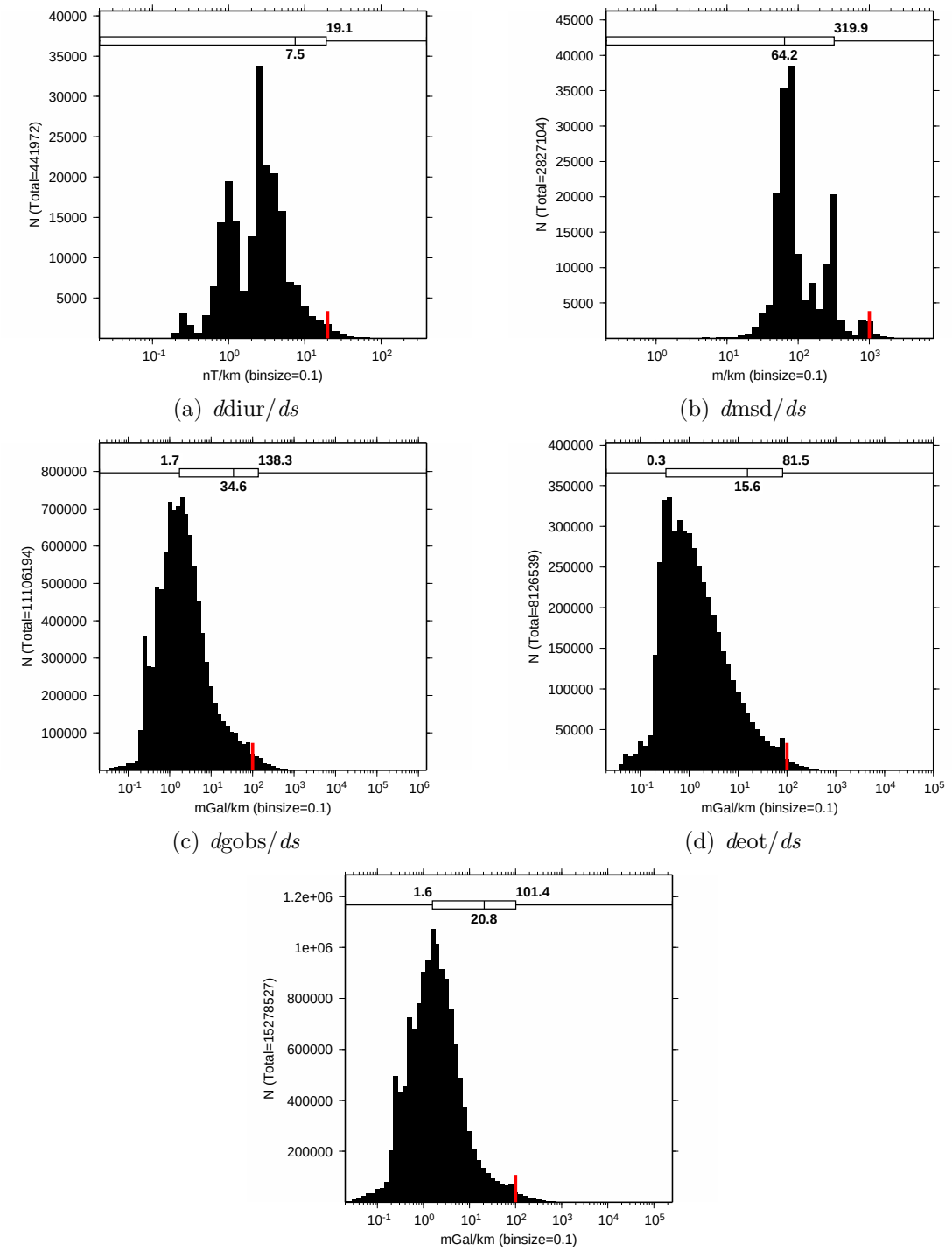


(a) ds/dt (b) $d\text{twt}/ds$

(c) $d\text{depth}/ds$ (d) $d\text{mtf1}/ds$

(e) $d\text{mtf2}/ds$ (f) $d\text{mag}/ds$

Figure 3.8: Spatial gradient histograms for all geophysical fields with 50, 95, and 99 gradient percentiles indicated by box-and-whisker diagrams. Red hash marks indicate gradient thresholds determined in this study. See Table 2.1 for definitions of field acronyms.



(a) $d\text{diur}/ds$

(b) $d\text{msd}/ds$

(c) $d\text{gobs}/ds$

(d) $d\text{eot}/ds$

(e) $d\text{faa}/ds$

Figure 3.9: Spatial gradient histograms (continued) for all geophysical fields with 50, 95, and 99 gradient percentiles indicated by box-and-whisker diagrams. Red hash marks indicate gradient thresholds determined in this study. See Table 2.1 for definitions of field acronyms.

3.1.3 Distribution of grid offsets

Monitoring offsets from global gravity and bathymetry grids along-track is an effective tool for locating potentially bad data. Figure 3.10– 3.11 show all offsets detected in this study. Scatter plots showing offset length versus offset height help one appreciate the extreme magnitude of some offsets. Most (99%) of bathymetry offsets are less than 4,742 m·km, while 99% of free-air gravity offsets are less than 2,509 mGal·km. Offsets within these ranges have lengths and heights that plot very close to the axes in Figures 3.10(left) and 3.11(left).

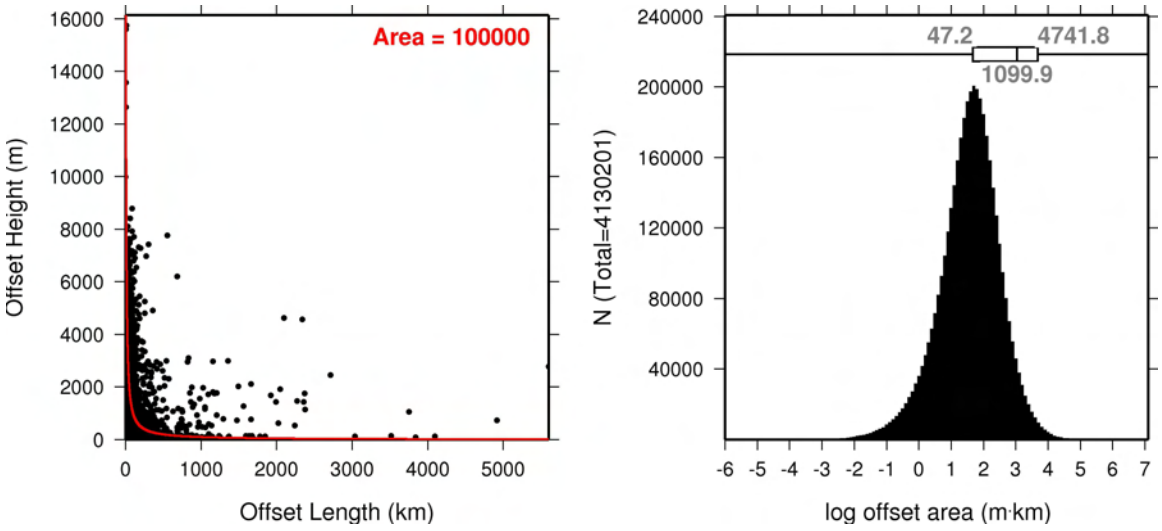


Figure 3.10: Offsets between ship and gridded data calculated along-track reveal large errors in shipboard bathymetry. (left) Offset areas exceeding 100,000 m·km (red curve) are flagged as excessive. Note that many offset lengths exceed 1,000 km along-track, with some offset heights exceeding 10,000 m for poorly navigated cruises crossing land. (right) Lognormal offset area histogram with overlaid box-and-whisker diagram indicating 50, 95, and 99 area percentiles. 99% of offsets are smaller than 4,742 m·km (∼70 m depth offset for 70 km along-track).

3.1.4 Navigation on land

Obvious ship-track navigation errors reflect the unprocessed nature of many submitted cruises, as illustrated in Figure 3.12. Mis-located geophysical measurements are rarely, if ever, salvageable. I detected that 40% of cruises surveyed from 1995 to 2000

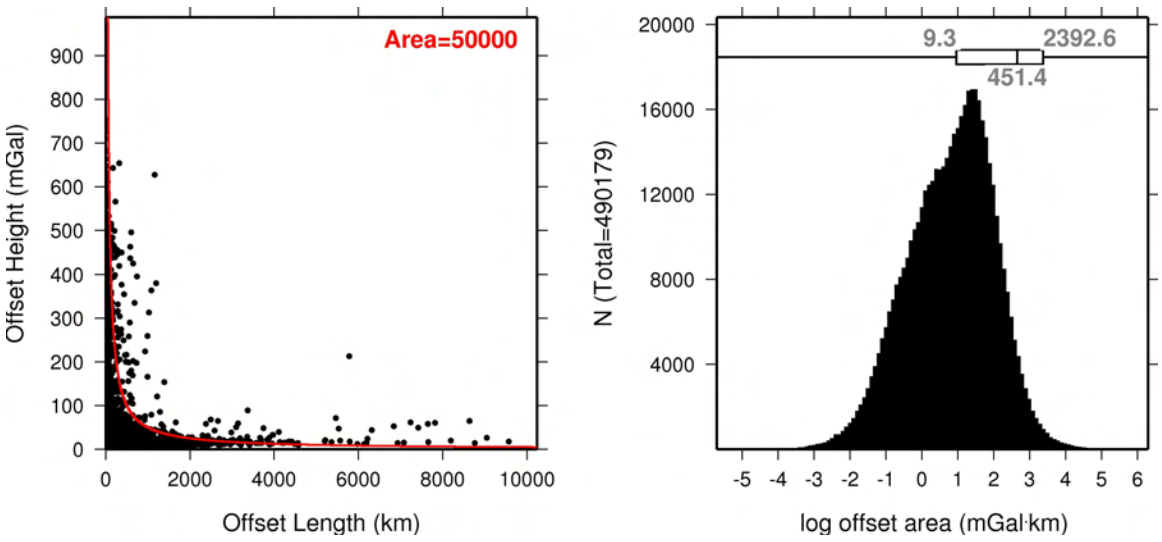


Figure 3.11: Along-track offsets between ship and gridded gravity reveal errors. (left) Offset areas exceeding 50,000 mGal·km are flagged as excessive. Many gravity offset lengths exceed 4,000 km along-track, with offset heights exceeding 900 mGal for cruises with uncalibrated gravimeters. (right) Lognormal offset area histogram with overlaid box-and-whisker diagram indicating 50, 95, and 99 area percentiles. 99% of offsets are smaller than 2,509 m·km (∼50 mGal offset for 50 km along-track).

contained records passing over land, indicating that this problem is not confined to the early days of satellite navigation. As there are not enough GPS satellites, navigation quality degrades as satellites enter or leave the visible horizon. These errors should be removed by the data processor. Being a common occurence in marine geophysics, loss of satellite navigation ought to be better treated in processing.

3.2 Systematic Analysis

3.2.1 Grid analysis

Free-air gravity and bathymetry data collected at sea are compared against global grids using Re-weighted Least Squares (RLS) regression analysis to detect errors affecting entire cruises. If both cruise and the more or less independent grid both recorded the phenomena correctly, then they should agree at all locations, and I expect a 1:1 slope and perfect correlation. I suspect that systematic problems, such

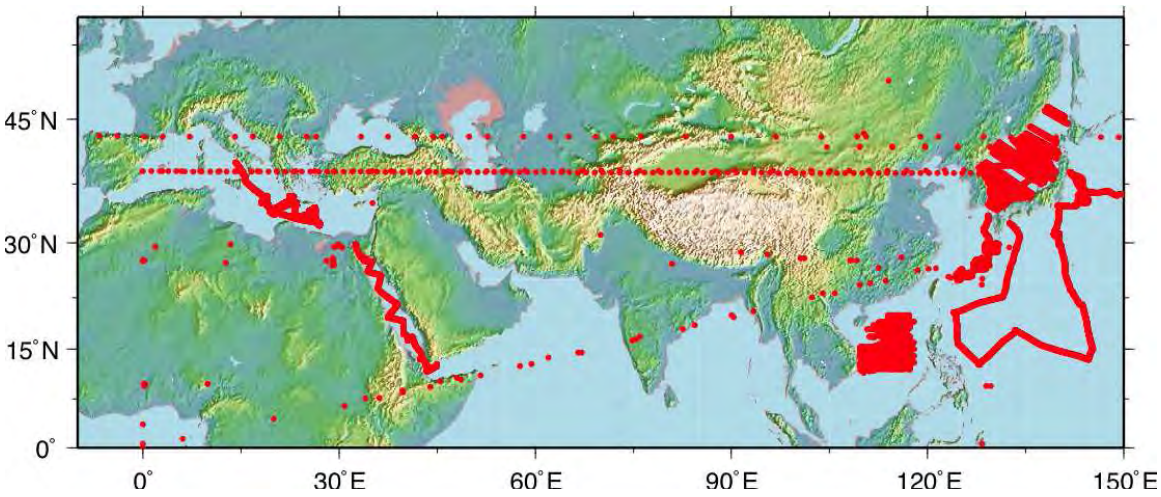


Figure 3.12: Marine trackline records falsely reported over land masses illustrate the lack of processing of some cruises prior to submittal to NGDC. These false data propagated in the archive create more work for scientists and are easily misconstrued. NGDC cruises mapped above, submitted by Lamont, WHOI, NOAA, Scripps, Russia, Hydro Dept. of Japan, Geological Survey of Japan, and Kobe University, include 01010029, 02020011, 06050008, 15050029, 16060001, 16990001, 17080012, 17120001, 29990001, J1020001, J1020024, J2010014, J2010015, J5010001, and J5010004.

as recording data in the wrong unit, or erroneous instrument calibration, will give slopes and intercepts that can be interpreted in terms of such effects. However, it became evident in the course of this research that RLS regression consistently found unexpected gravity regression slopes due to spectral differences between ship and grid data and due to high data density of anomalies near zero. Because gravity anomalies tend to center near zero, insufficient weight is given to the remaining range of data. A further hindrance to RLS regression for both gravity and bathymetry is due to the failure of Least Median of Squares regression to determine the correct regression slope and intercept when the number of outliers exceeds the number of quality data points.

3.2.2 Decimated Re-weighted Least Squares regression

To improve RLS regression for both gravity and bathymetry comparison with global grids, I modify the RLS method by decimating ship and grid data into bins of 5 mGal

for gravity and 50 m for bathymetry. The range of acceptable data values (described in detail in Chapter 4) is divided into bins for both ship and grid data. Each bin is then populated by the number of ship and grid values found. Ship values exceeding thresholds determined in this research are skipped to eliminate biasing by outliers. The non-empty bins are then used in the regression analysis. In cases where the decimated RLS correlation is insignificant or insufficient bins are encountered, RLS regression is performed on the full data set. In effect, this decimation modification reduces the number of regression points near zero for gravity and reduces weight given to large clusters of outliers (see Figure 3.13).

Distribution of RLS regression coefficients

I generate histograms of regression coefficients obtained from all bathymetry and gravity cruises including regression slope, intercept (gravity only), and correlation. Regression slope describes overall agreement between ship and grid values, thus slopes close to one are expected. RLS regression intercept is a measure of average offset, or DC shift, of ship data compared to satellite-derived grids. Regression intercepts are forced to remain at zero for bathymetric regression because bathymetry grids are constrained by trackline depths and therefore should not be DC shifted. Correlation (r) ranges from $\pm$ 1 and describes the extent to which ship and grid data are linearly related ($\pm$ 1 implies a strong linear relationship whereas 0 implies no linear relationship).

Figure 3.14 shows regression coefficients obtained from 1,653 out of 1,660 total archived gravity cruises. Regression slope (Figure 3.14(a)) shows overall agreement with Sandwell and Smith's version 15.1 gravity grid, with noticeable exceptions to the rule near 0.1 and 10. These clusters correspond to cruises reporting free-air gravity with incorrectly scaled units. Figure 3.14(b) shows regression intercepts (DC shift) for all gravity cruises with values centering on zero, as expected. DC shifts out to 50

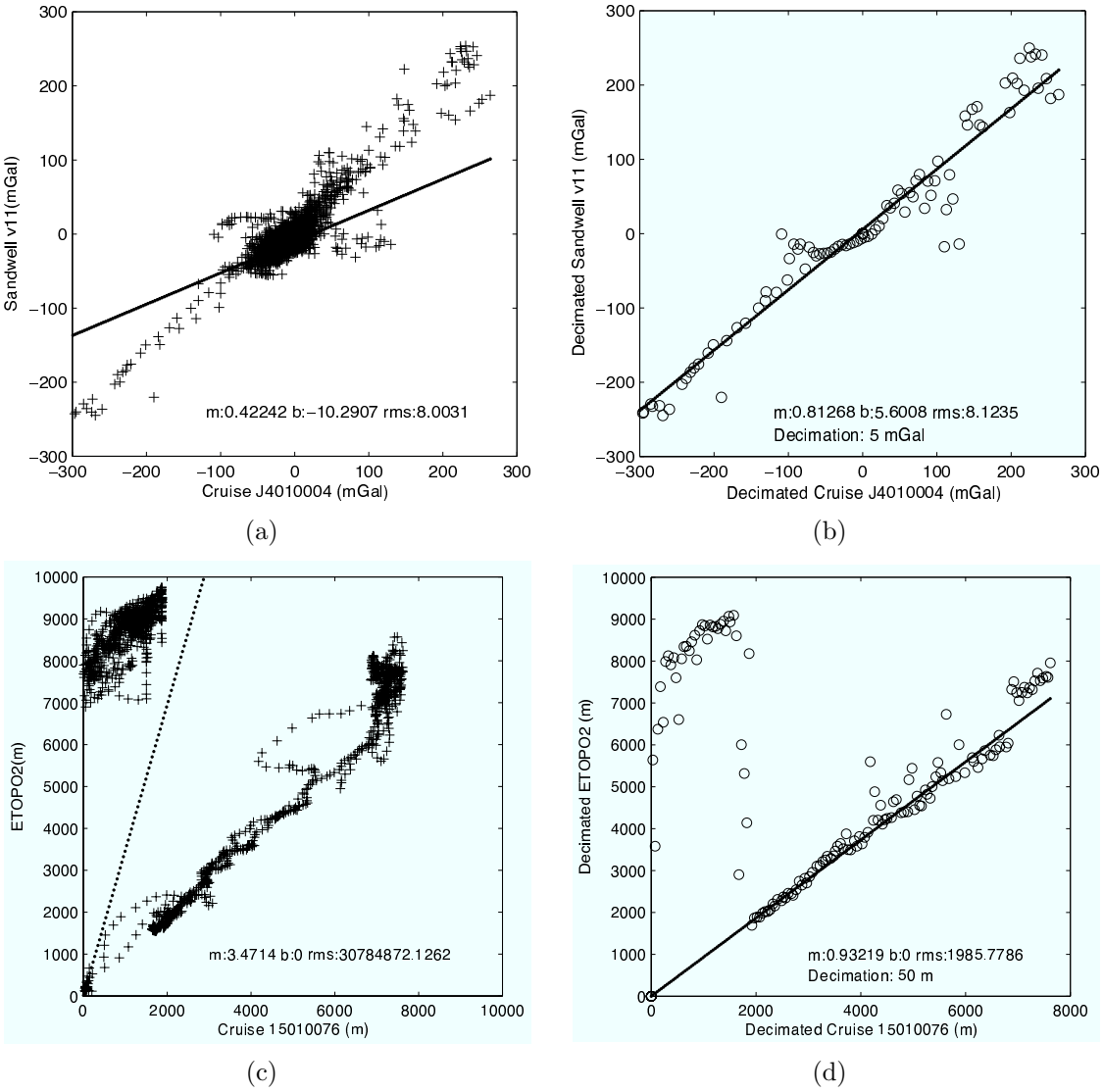


Figure 3.13: (a) Free-air gravity anomalies concentrated near zero complicate regression analysis. (b) Decimation of ship-grid pairs at 5 mGal intervals improves regression. (c) LMS regression breaks down when 50% of data are outliers. (d) Decimation of ship-grid bathymetry pairs reduces weight given to outlying data points.

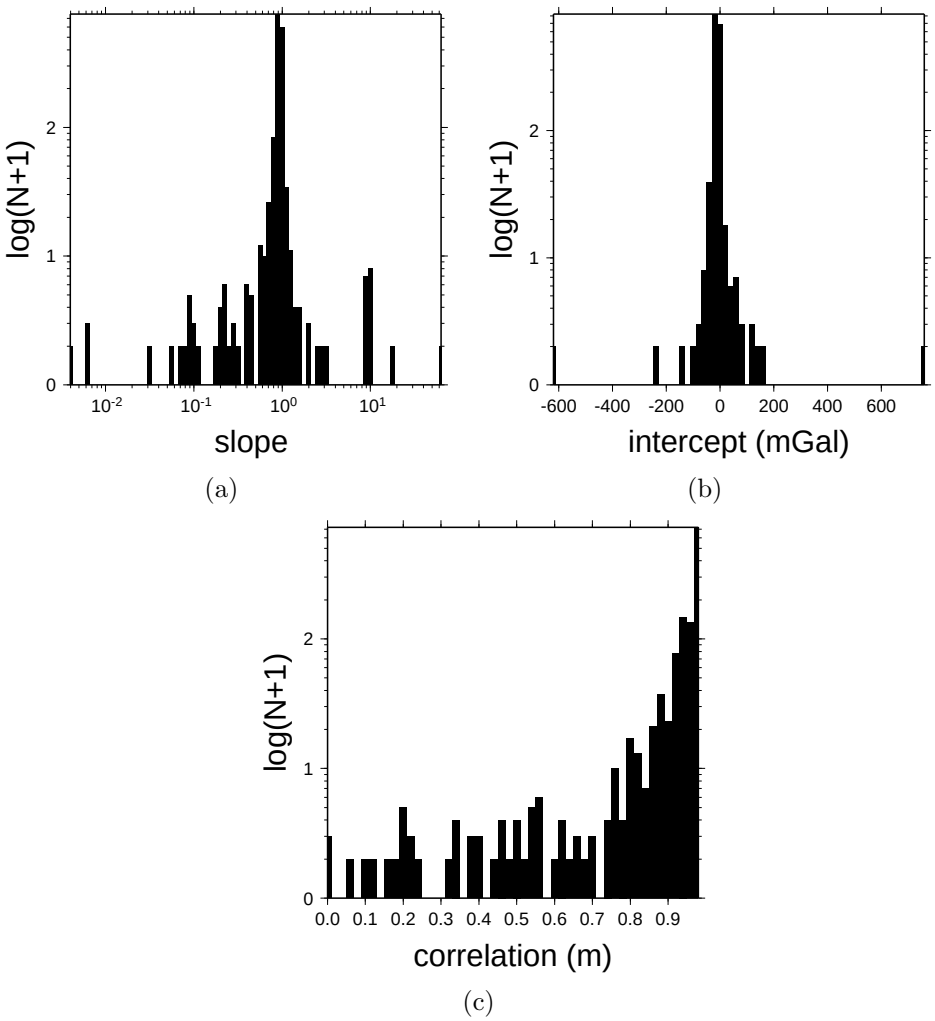
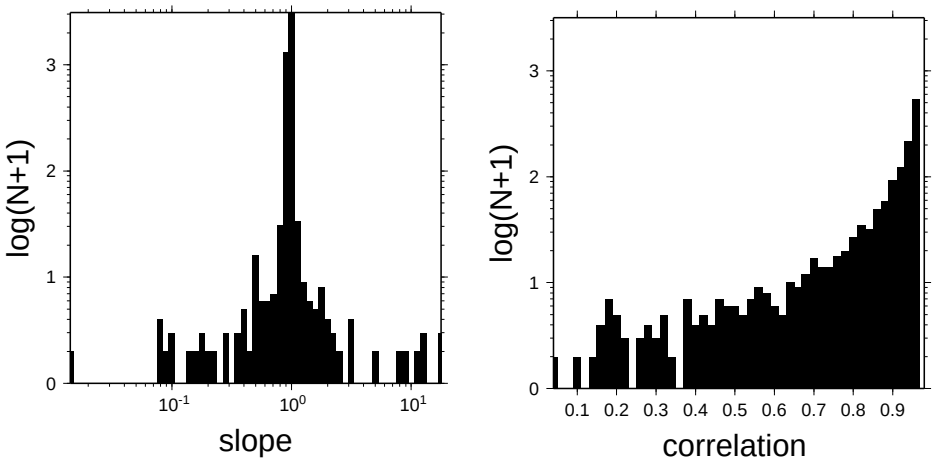


Figure 3.14: Decimated RLS regression statistics for trackline and grid free-air gravity anomalies from 1,653 gravity cruises showing: (a) regression slopes clustering near expected 1 with notable outliers near 0.1 and 10; (b) DC shifts near zero with an extreme offset of -990 mGal; (c) correlation coefficients for all gravity RLS comparisons. Note logarithmic scales.

- 100 mGal appear fairly common, with extreme offsets from -990.8 to 365.3 mGal. Correlation coefficients cluster near one (Figure 3.14(c)) but poor correlation in many cruises is evident. Of the seven cruises not accounted for, three cruises are located outside the ± 72° latitude boundaries of Sandwell and Smith's global gravity grid, and four cruises have insufficient gravity values or total gravity variation less than 5 mGal.



(a) (b)

Figure 3.15: Decimated RLS regression statistics for trackline and predicted bathymetry grid values from 4,572 bathymetry cruises showing: (a) regression slopes clustering near 1; (b) correlation coefficients for all bathymetry RLS comparisons. Note logarithmic scales.

Figure 3.15 shows regression statistics for 4,572 out of 4,621 archived bathymetry cruises. Regression slope (Figure 3.15(a)) again shows overall agreement with global grids (S2004). Bathymetry incorrectly scaled by 0.1 and 10 occurs to a lesser extent than with gravity data. Other scale factors such as 0.547 (fathoms), and 0.75 (ms two-way travel time rather than meters) are barely discernible, if at all, amongst the slew of other randomly distributed slopes, but are known to be present in the archive (e.g., Figure 1.7). Correlation coefficients again cluster near one (Figure 3.15(b)) but poor correlation in many cruises is still evident. 45 of the excluded bathymetry cruises have less than 50 m total depth range while the rest of the excluded cruises have undefined regression due to non-varying bathymetry or grid values. Many of

these excluded bathymetry cruises contain data from shallow water surveys, many of which report depth soundings in integer meters which is problematic for surveys in areas with little bathymetric relief.

3.2.3 Low precision

The MGD77 format specifies the stored precision of the data (i.e., the number of implied decimals). For bathymetry, free-air gravity, and magnetic anomalies these are 0.1m, 0.1 mGal, and 0.1 nT, respectively. However, it is clear that submitted data often may have lower precision. For instance, histograms of differences between successive free-air observations reveal that integer values occur much more frequently than multiples of 0.1 mGal (Figure 3.16). This is possibly because the source institution internally processed their data to the nearest mGal, or they used integers for an intermediate format.

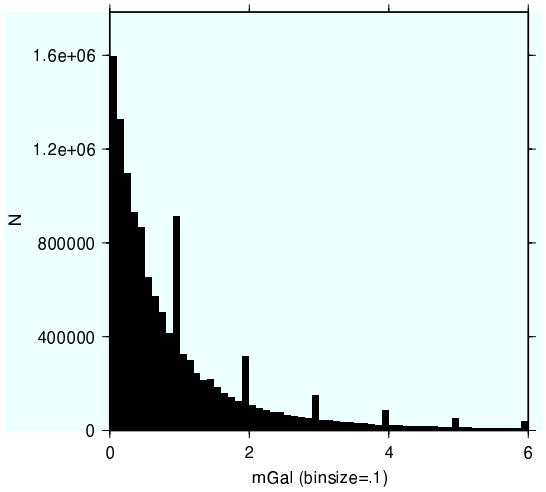


Figure 3.16: Along-track differences in free-air anomalies show a clear tendency to store integer values rather than tenths of mGal as specified by MGD77 format. This also occurs in bathymetric and magnetic observations.

I find 3,994 bathymetry cruises storing depth as integer meters rather than tenths of meters as specified in the MGD77 format. Another 79 cruises store depth as integer multiples of 5 m. 1,516 magnetic cruises store residual magnetics as integers

with another 213 cruises storing residual magnetics as integer multiples of 5 nT, while 243 gravity cruises store free-air anomalies with integer precision and 12 cruises store free-air anomalies with integer multiples of 5 mGal. In shallow areas and areas with low relief, truncated ship observations are indeed problematic, as illustrated in Figure 3.17.

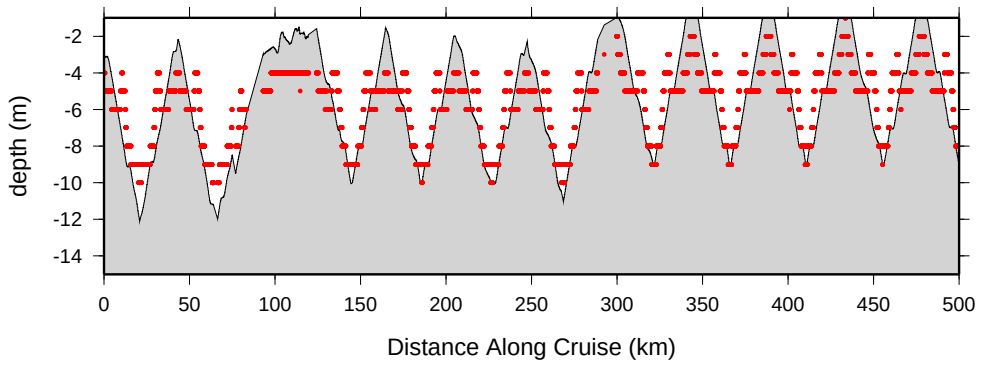


Figure 3.17: Truncation of ship data is particularly detrimental in areas of low relief as shown by this bathymetry profile from the 1986 USGS cruise "1986N5" (NGDC id 06990023) in red plotted with S2004 predicted bathymetry grid in gray.

Chapter 4

GEOPHYSICAL BOUNDARIES

4.1 Summary of Along-track Findings

4.1.1 Value statistics

The complete ranges of all trackline geophysical values and the chosen upper and lower limits for such data are given in Table 4.1; refer to Table 2.1 for MGD77 field abbreviations. Absolute field observations such as bathymetry, total field magnetics, and observed gravity vary within reasonable limits so I choose conservative geophysical bounds for these fields by slightly extending the observed ranges. On the other hand, derived fields including residual magnetics, Eötvös correction, and free-air anomaly deviate unreasonably, necessitating further limitation on what constitutes acceptable lower and upper limits for MGD77 sanity checking. The selected lower and upper boundaries for geophysical values are presented for all geophysical fields including position, bathymetry, magnetics, and gravity. The number of latitude and longitude records indicates the current number of data records archived at NGDC. Magnetic diurnal correctors, with just 443,145 records, is the least well constrained of the geophysical fields. By virtue of being a floating-point value in each MGD77 data record, magnetic sensor depth/altitude (msd) is listed with 2.8 million records. MGD77 format provides for positive and negative msd values but all archived msd values range between 0 and 30 m. The upper limit for Eötvös correctors is set in accordance with the upper velocity limit determined in this research, of 10 m/s. Maximum Eötvös corrections arise when a vessel is situated on the equator, heading due east. A vessel cruising at the upper velocity limit of 10 m/s (19.4 kts) should encounter Eötvös correctors no larger than 150 mGal [*Dehlinger*, 1978].

Table 4.1: Statistics for geophysical values

Field	Units	N	Min	Max	**Lower**	**Upper**
lat	degree	64,273,556	-89.9997	90	**-90**	**90**
lon	degree	64,273,556	-180	180	**-180**	**180**
twt	s	29,529,993	0.0002	15	**0**	**15**
depth	m	42,865,960	0.1	11,159	**0**	**11,000**
mtf1	nT	19,921,612	19,231.5	71,679	**19,000**	**72,000**
mtf2	nT	573,460	33,567.1	67,436	**19,000**	**72,000**
mag	nT	19,294,745	-9,864	9,998	**-1,000**	**1,000**
diur	nT	443,145	-688	517	**-100**	**100**
msd	m	2,841,944	0	30	**-1,000**	**1,100**
gobs	mGal	11,255,762	975,972	985,691	**977,600**	**983,800**
eot	mGal	8,173,303	-940.3	8,002.5	**-150**	**150**
faa	mGal	15,512,059	-998.9	998.4	**-400**	**550**

4.1.2 Gradient statistics

Results from gradient calculations are presented in Table 4.2. As the sign of the gradients is arbitrary and ignored, all gradients have zero minimum, with most having 5% levels also close to zero. Therefore, these statistics are omitted from the table. Vessel speed ($\Delta s/\Delta t$) is unlike other gradients due to its dependence on time, rather than distance. The median speed of 4.7 (9.1 kts) appears reasonable. When computing gradients, some records are omitted due to gap checking. This loss is fairly small, however. For example, 1.1% of depth records, 0.8% of magnetic records, and 1.5% of free-air anomalies are excluded due to gap skipping. Excessive maximum gradients such as $1.93e10^7$nT/km for mtf1 likely occur when a vessel is on station, causing the denominator to approach zero which yields unrealistic gradients. The millions of gradients calculated form long-tailed distributions making interpretation problematic, necessitating the logarithmic-scale gradient histograms presented in Chapter 3. The right-most column lists gradient boundaries determined in this study. These boundaries are determined visually by first finding the 99% gradient value. I then locate where changes in counts per bin begins to decrease slowly (i.e. forms a tail). I apply this process consistently to the different gradients. As this method is subjective, I

round limits to the nearest multiple of 100.

Table 4.2: Statistics for geophysical gradients

Field	Units	N	Median	Max	**Bounds**
ds/dt	m/s	60,832,703	4.73408	1.18745e+07	**0-10**
$d$twt/ds	s/km	29,276,648	0.0243557	7517.58	± **1**
$d$depth/ds	m/km	42,408,286	18.0977	5.65428e+06	± **1000**
$d$mtf1/ds	nT/km	19,766,939	8.39238	1.93376e+07	± **200**
$d$mtf2/ds	nT/km	572,032	4.96145	165569	± **200**
$d$mag/ds	nT/km	19,147,646	7.49064	1.20142e+06	± **200**
$d$diur/ds	nT/km	441,972	0	434.578	± **20**
$d$msd/ds	m/km	2,827,104	0	8999.36	± **100**
$d$gobs/ds	mGal/km	11,106,194	1.72204	1.83468e+06	± **100**
$d$eot/ds	mGal/km	8,126,539	0.332347	108407	± **100**
$d$faa/ds	mGal/km	15,278,527	1.5552	299679	± **100**

4.1.3 Offset area statistics

I present offset area statistics in Table 4.3. I combine positive offset areas with the absolute value of negative offsets so that all offsets are analyzed. The number of gravity records is about one-third that of bathymety records, yet the number of gravity offset areas found is almost 10 times fewer that the number of bathymetry offsets found. This is likely due to spectral differences between gravity and topography. I choose conservative offset area boundaries of 50,000 mGal·km for gravity and 100,000 m·km for bathymetry.

Table 4.3: Offset area statistics

Field	Units	N	Median	75%	95%	Max	**Limit**
faa	mGal·km	446,587	8.5995	47.6739	417.709	2.01475e+06	**50000**
depth	m·km	4,130,201	47.22	165.613	1,099.93	1.55244e+07	**100000**

4.2 Summary of Systematic Findings

Regression statistics

Tables 4.4 and 4.5 show RLS regression statistics for gravity and bathymetry. Regression slopes for both fields center near 1, as expected. Correlation is higher for shipboard bathymetry comparisons with gridded data than for gravity comparisons, especially at the 5% correlation level. This observation was anticipated due to the reliance of global predicted bathymetry grids on shipboard bathymetry.

Table 4.4: Gravity regression statistics

Coefficient	Min	5%	Median	95%	Max
slope	-2.95756	0.713762	0.996156	1.07237	10.6817
intercept (mGal)	-990.768	-18.1737	-0.718918	13.8575	365.347
correlation	0.039004	0.741218	0.977029	0.996982	1

It is also notable that gravity regression slopes are skewed toward negative slopes (5%=0.71) when compared to bathymetry regression slopes (5%=0.97). This is likely due to resolution differences between shipboard and satellite altimetry data. *Neumann et al.* [1993] compared shipboard gravity anomalies with satellite-derived gravity anomalies, noting larger residuals between ship and grid data (7 mGal standard deviation) than those found by internal crossover analysis of ship data (1.8 mGal standard deviation). The averaging effect of lower resolution gridded data dampens the amplitude of the gravity signal and may affect regression analysis.

Table 4.5: Bathymetry regression statistics

Coefficien	Min	5%	Median	95%	Max
slope	-1.97111	0.965014	1	1.03279	19.6656
correlation	0.041023	0.821866	0.991951	0.999011	1

Chapter 5

DATA IMPROVEMENT APPLICATIONS

Geophysical limits determined in this research serve to calibrate the mgd77sniffer program for error detection. Without modifying raw MGD77 files, I use the mgd77sniffer tool to generate E77 errata tables which I apply to problematic data using the new publicly available tools mgd77manage and mgd77list ([*Wessel and Chandler*, 2006]). For these purposes, it is sufficient to know that mgd77manage is able to read E77 errata tables and to set correction flags that can be used by mgd77list to generate corrected data. I now illustrate the improvement of several historic cruises containing archetypical, albeit extreme, errors.

5.1 Re-scaling Cruise Data

Figure 5.1 illustrates scale-correction of shipboard gravity data submitted to NGDC in mGal rather than tenths of mGal. Scientists submitting incorrectly scaled data to NGDC risk truncated data or loss of precision. For example, submitting mGal gravity data rather than tenths of mGal automatically reduces data precision to integer mGal. Data submitted in tens of mGal will likely have all anomalies exceeding $\pm$ 100 truncated. It is even possible in this scenario that only the largest base digit will be lost, leaving the remaining digits completely invalid.

To improve this cruise, I compare cruise free-air gravity with Sandwell and Smith version 15 gravity using the mgd77sniffer program. See Appendix A.2 for mgd77sniffer documentation.

```
mgd77sniffer 06050010 -gfaa,grav.15.1.img,0.1,0 -De
```

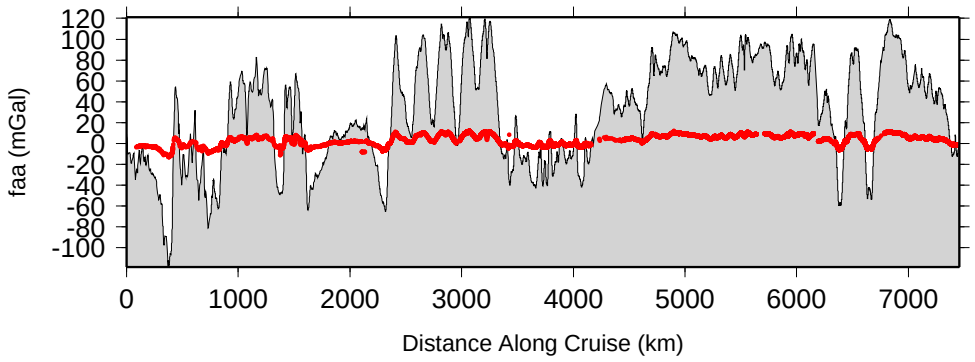


Figure 5.1: Raw free-air gravity data from the 1976 USGS cruise L47WG (NGDC id 06050010) (in red) plotted with Sandwell and Smith version 15 gridded gravity anomaly in gray. Data were submitted in whole mGal units rather than tenths of mGal as specified by NGDC.

This generates the E77 errata table 06050010.e77 listed below:

```
# Cruise 06050010 ID L476WG MGD77 FILE VERSION: 19870415 N_RECS: 27268
# Examined: Mon Apr  3 13:15:04 2006
# Examiner: mtchandl
# Arguments:  -gfaa,grav.15.1.img,0.1,0 -De
# Errata: Header
N-faa-E-01: Regression scale 0.098 different from 1. Recommended: [10]
N-faa-E-02: Regression offset -0.03 different from 0. Recommended: [0]
I-twt-W-03: More recent bathymetry correction table available
I-depth-W-04: Integer precision in depth
# Errata: Data
1976-06-26T08:15:00 2 C-0-0-0  NAV: excessive speed
...
```

These flags indicate that cruise data are incorrectly scaled by 0.1. To correct, apply the recommended scale factor as follows:

```
mgd77sniffer 06050010 -gfaa,grav.15.1.img,0.1,0 -De -Afaa,10,0
```

Resulting in this updated errata table:

```
# Cruise 06050010 ID L476WG MGD77 FILE VERSION: 19870415 N_RECS: 27268
# Examined: Mon Apr  3 13:23:25 2006
# Examiner: mtchandl
# Arguments:  -gfaa,grav.15.1.img,0.1,0 -De -Afaa,10,0
# Errata: Header
Y-faa-E-01: Regression scale 0.965 different from 1. Recommended: [10]
N-faa-E-02: Regression offset 0.34 different from 0. Recommended: [0]
I-twt-W-03: More recent bathymetry correction table available
I-depth-W-04: Integer precision in depth
I-faa-W-04: Integer precision in faa
```

```
# Errata: Data
1976-06-26T08:15:00 2 C-0-0-0  NAV: excessive speed
```

These flags are set using mgd77manage and corrected data are extracted using mgd77list. Neither the original MGD77 file nor the E77 file are altered. Instead, the data and flags are reformatted into a more flexible netCDF-based format file which can accommodate additional meta-data. This results in the corrected profile shown in Figure 5.2. As in this example, scientists should always examine data closely before applying such substantial changes.

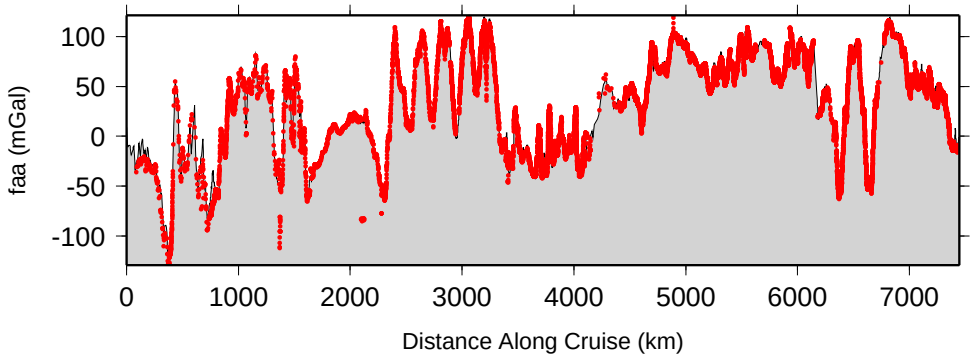


Figure 5.2: Cruise L476WG (06050010) free-air gravity is scale-corrected without modifying the original MGD77 data, revealing secondary outliers.

5.2 Removing Navigation Outliers

Since the early 1990s, satellite navigation has improved to provide constant, global coverage. Much of the trackline archive was surveyed prior to accurate GPS satellite positioning. Figure 5.3 shows the 1965 Woods Hole cruise A2015L06 (NGDC id 02020011) which lost position while, heading east, crossed the International Dateline. Linear interpolation between navigation fixes falsely reported the ship travelling up to 1,800 m/s as its position reached -180° by going all around the world, crossing Greenwich in the process.

To improve this cruise, I analyze navigation and compare bathymetry to S2004 gridded data using the mgd77sniffer program.

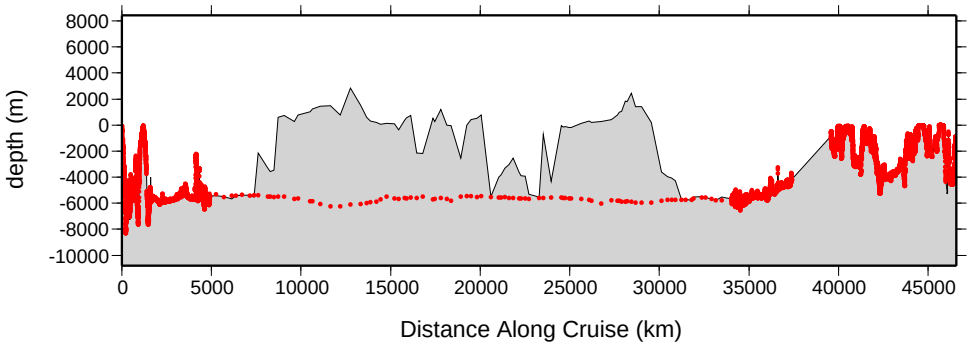


Figure 5.3: Unprocessed bathymetry from WHOI's 1965 cruise A2015L06 (NGDC id 02020011) plotted in red with S2004 gridded depth in gray. Many records report coordinates passing over land.

```
mgd77sniffer 02020011 -Gdepth,S2004_hdr.i2 -De
```

This generates the E77 errata table 19150034.e77 which contains the following:

```
# Cruise 02020011 ID A2015L06 MGD77 FILE VERSION: 19820427 N_RECS: 13911
# Examined: Mon Apr  3 15:33:26 2006
# Examiner: mtchandl
# Arguments:  -Gdepth,S2004_hdr.i2 -De
# Errata: Header
N-H-02020011-04-03-E: Invalid Survey Departure Day: (29) [30]
N-H-02020011-04-06-E: Invalid Survey Arrival Day: (15) [10]
N-H-02020011-11-05-E: Invalid Leftmost Longitude: (+139) [-180]
N-H-02020011-11-06-E: Invalid Rightmost Longitude: (-079) [+180]
N-H-02020011-16-01-E: Invalid Number of Ten Degree Identifiers: (20) [49]
N-H-02020011-16-06-E: Ten Degree Identifier 7400 not marked in header but block was crossed
...
N-H-02020011-16-06-E: Ten Degree Identifier 1415 not marked in header but block was crossed
I-depth-W-04: Integer precision in depth
# Errata: Data
1965-09-30T07:26:00 175 0-0-K-0  GRAD: twt excessive
1965-10-02T21:45:00 1096 0-0-KL-0  GRAD: twt, depth excessive
1965-10-02T21:50:00 1097 0-0-KL-0  GRAD: twt, depth excessive
1965-10-10T15:17:00 3218 0-0-K-0  GRAD: twt excessive
1965-10-11T14:03:00 3703 C-0-0-0  NAV: excessive speed
1965-10-11T14:07:00 3704 C-0-0-0  NAV: excessive speed
```

The resultant E77 file lists a multitude of navigation warnings indicating that this cruise passed through regions not specified in the MGD77 header. I apply the recommended corrections using mgd77manage and mgd77list. This results in the improved profile shown in Figure 5.4. Notice that all records above land are removed, as are records with speeds greater than 10 m/s.

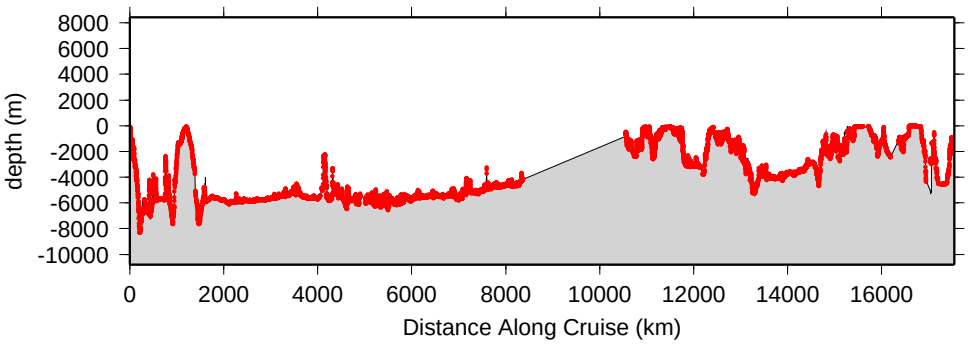


Figure 5.4: Cruise A2015L06 (02020011) is improved by removal of records having excessive speed and navigation over land.

5.3 Correcting for Systematic Offset

Insufficient gravity tie-ins to land gravity benchmarks and sudden gravimeter tares may significantly offset gravity data from expected values. Perhaps the most obvious example of this is illustrated in Figure 5.5. Robust regression provides an effective means for determining this offset. In this case a considerable amount of data was truncated due to the MGD77 format's ±999.8 limit for free-air anomaly data.

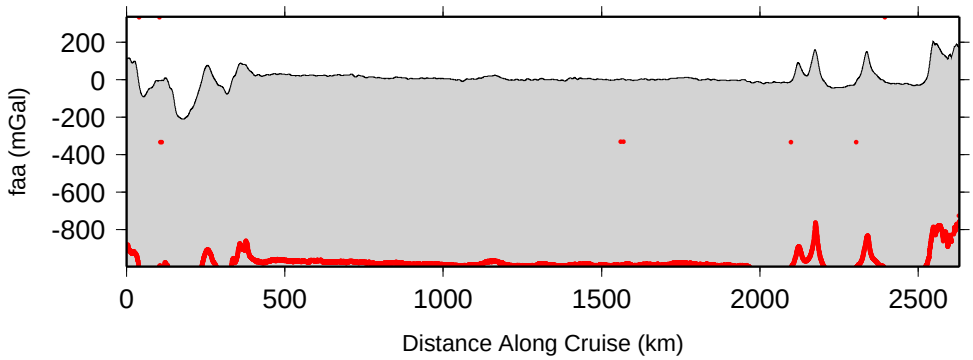


Figure 5.5: 1984 USGS cruise L884SP (NGDC id 06050075) reports free-air gravity data, in red, offset by -990 mGal from Sandwell and Smith gridded data in gray.

To improve this cruise, I first note that free-air gravity in this cruise is entirely outside the acceptable range. To handle this case, I need to modify the default lower limit for free-air gravity from -400 to -1000 mGal.

```
mgd77sniffer -Dl > limits_06050075.d
```

Then modify the contents of limits_06050075.d to the following:

```
#abbrev min max maxSlope maxArea
faa -1000 550 100 50000
```

Now I override the mgd77sniffer defaults using the new custom limits file and follow similar steps by comparing cruise free-air anomalies to Sandwell and Smith version 15 gravity using the mgd77sniffer.

```
mgd77sniffer 06050075 -gdepth,grav.15.1.img,0.1,0 -De -Llimits_06050075.d
```

This generates the following E77 errata table:

```
# Cruise 06050075 ID L884SP MGD77 FILE VERSION: 19870805 N_RECS: 8767
# Examined: Mon Apr  3 16:55:15 2006
# Examiner: mtchandl
# Arguments:  -gfaa,grav.15.1.img,0.1,0 -De -Llimits_06050075.d
# Errata: Header
N-faa-E-01: Regression scale 0.904 different from 1. Recommended: [1]
N-faa-E-02: Regression offset -991.25 different from 0. Recommended: [991.2]
I-depth-W-04: Integer precision in depth
# Errata: Data
1984-07-19T23:24:00 1 0-0-0-W  GRID: faa offset
1984-07-19T23:25:00 2 0-0-0-W  GRID: faa offset
...
```

These flags indicate that free-air anomalies are DC shifted by -991.2 mGal. Also notice that free-air gravity records are flagged as offset from the gravity grid along-track (i.e. the E77 error records above indicate faa grid offsets). By using the E77 table in its current form, the analyst would incorrectly flag all free-air records as questionable. To correct, add the recommended DC correction and generate a new E77 table as follows:

```
mgd77sniffer 06050075 -gfaa,grav.15.1.img,0.1,0 -De -Llimits_06050075.d
-Afaa,1,991.2
```

Resulting in this updated E77 errata table 06050075.e77:

```
# Cruise 06050075 ID L884SP MGD77 FILE VERSION: 19870805 N_RECS: 8767
# Examined: Mon Apr  3 17:17:57 2006
# Examiner: mtchandl
# Arguments:  -gfaa,grav.15.1.img,0.1,0 -De -Llimits_06050075.d -Afaa,1,991.2
```

```
# Errata: Header
N-faa-E-01: Regression scale 0.914 different from 1. Recommended: [1]
Y-faa-E-02: Regression offset -0.01 different from 0. Recommended: [991.2]
I-depth-W-04: Integer precision in depth
# Errata: Data
1984-07-20T02:46:00 203 0-0-KL-0  GRAD: twt, depth excessive
1984-07-20T02:48:00 205 0-0-KL-0  GRAD: twt, depth excessive
...
```

Notice that by correcting the free-air data for DC offset in this example, the mgd77sniffer no longer flags free-air records as offset along-track from the gravity grid. Applying these flags as usual yields the improved profile shown in Figure 5.6.

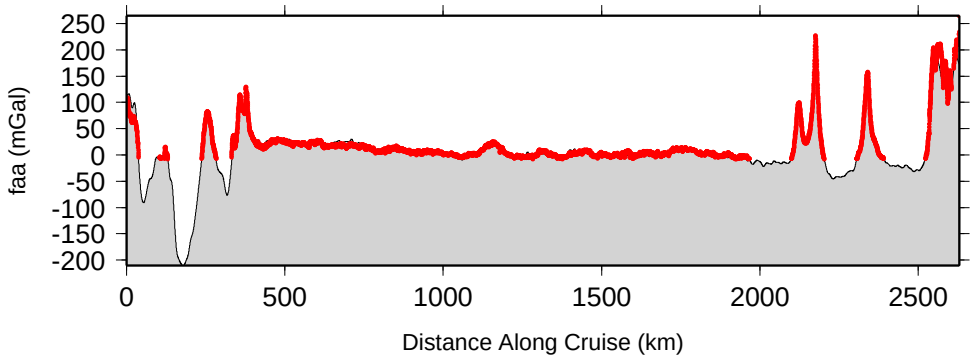


Figure 5.6: Considerable improvement is gained by applying offset correctors. Due to the extreme original offset, all negative anomalies are truncated, rendering even the corrected data anomalous.

5.4 Removing Erroneous Subsets

Portions of cruises may contain invalid data due to various problems encountered during data acquisition such as temporary instrument malfunction, operator error, and poor bottom tracking. These problems are often difficult to detect without sophisticated methods such as crossover analysis. For bathymetry and free-air gravity, however, offsets from grids may be located, as depicted in Figure 5.7

Note that the loss of bottom tracking in the 1994 British cruise was not detected by the United Kingdom Hydrographic Office data processors or NGDC prior to data

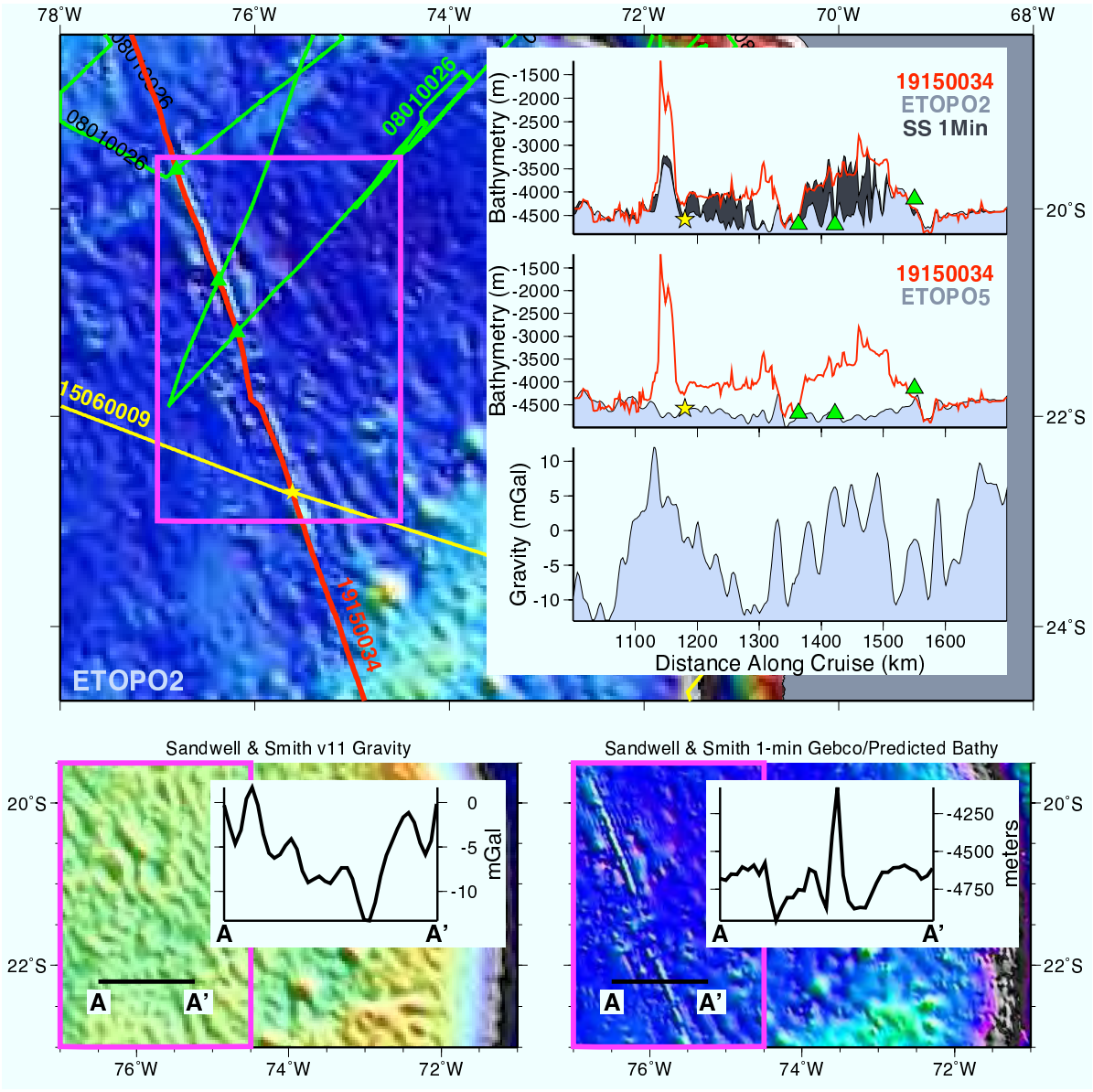


Figure 5.7: Predicted bathymetry grids such as those compiled by Smith and Sandwell are adversely affected when poor quality ship-track data are used to constrain topographic predictions. An example, shown here in red, is the ship-track from cruise 19150034 (19940045), collected west of Chile in 1994 by the British Navys HMS ENDURANCE. An artifact not visible in Sandwell and Smith gravity (lower left) incorrectly appears as a linear topographic ridge in ETOPO2 (top) and S2004 bathymetry (lower right).

archival, attesting to the difficulty in locating these types of errors. Had analysts compared ship and ETOPO5 bathymetry this erroneous data might have been removed. As this cruise has been included in predicted bathymetry grids since archival, a false bathymetric ridge has been introduced west of Chile in several popular grids. It is therefore desirable to perform comparisons when ship and grid data are independent.

To improve this cruise, I compare cruise and ETOPO5 bathymetry.

```
mgd77sniffer 19150034 -Gdepth,etopo5_hdr.i2 -De
```

The following E77 errata table is generated:

```
# Cruise 19150034 ID 19940045 MGD77 FILE VERSION: 19941220 N_RECS: 2987
# Examined: Mon Apr  3 12:10:40 2006
# Examiner: mtchandl
# Arguments:  -Gdepth,etopo5_hdr.i2 -De
# Errata: Header
N-H-19150034-12-01-W: Invalid Bathymetry Digitizing Rate: (999) [   ]
I-depth-W-04: Integer precision in depth
# Errata: Data
1107.4 362 0-0-0-L  GRID: depth offset
...
1561.45 542 0-0-0-L  GRID: depth offset
```

Applying this errata table results in the cleaned profile shown in Figure 5.8.

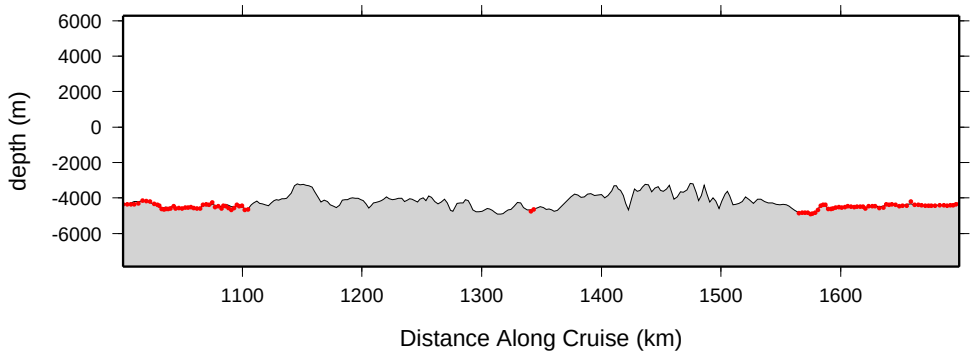
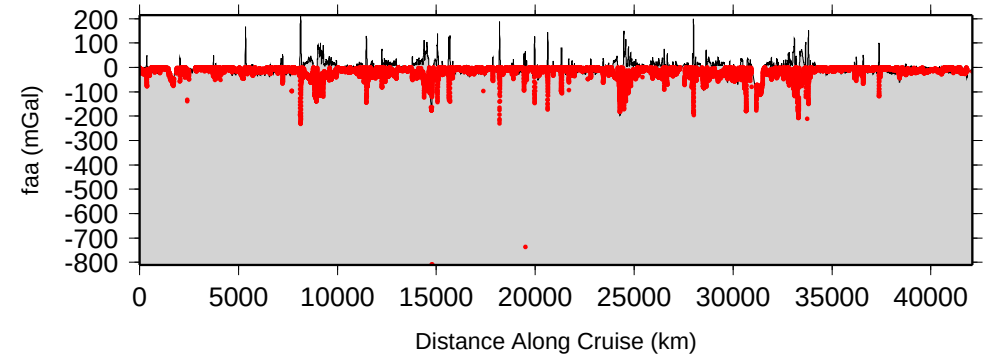


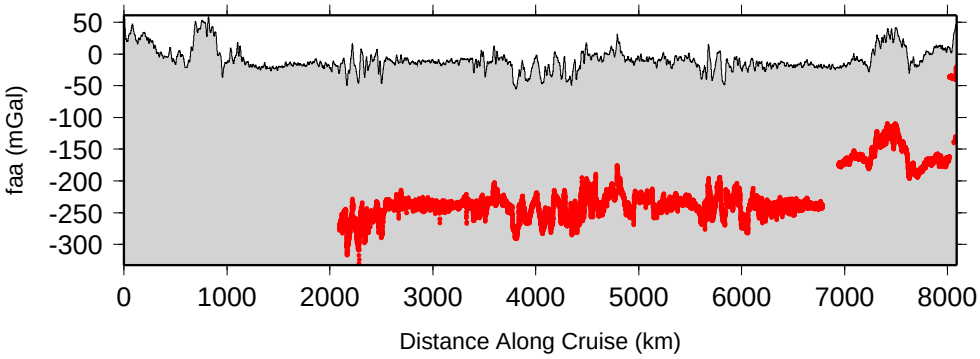
Figure 5.8: Corrected British cruise 19150034 shown in red with ETOPO5 gridded bathymetry in gray. ETOPO5 is chosen here due to the inclusion of erroneous bathymetry in recent grids.

5.5 Caveat

Some errors remain unresolvable using methods developed in this research. For these cases, which generally contain unique or infrequent problems, manual inspection and correction is required. Figures 5.9 and 5.10 illustrate data that can be improved through manual editing as well as bogus data that should be removed entirely.

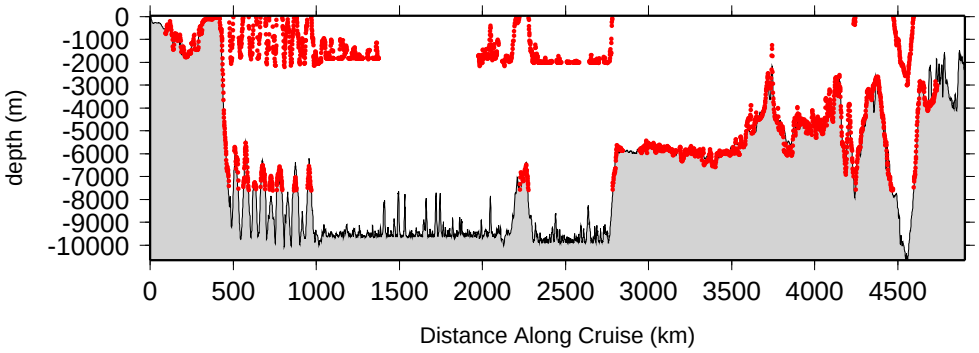


(a)

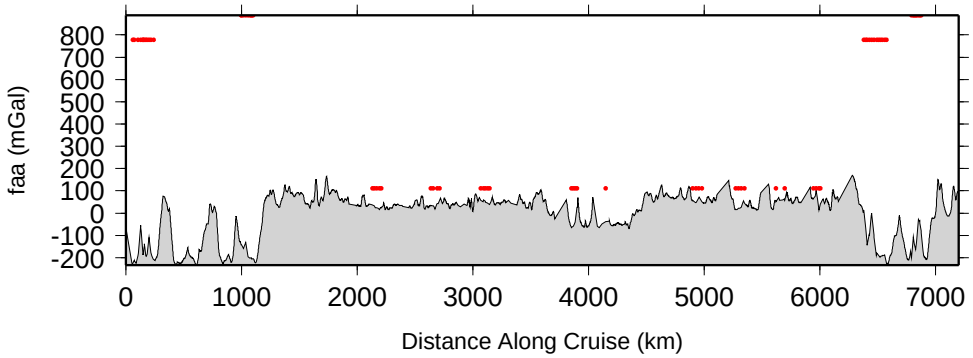


(b)

Figure 5.9: Examples of data issues not compatible with these error detection methods: (a) 1972 NOAA cruise POL7201 (NGDC id 03040057) reports only negative magnitude free-air gravity - positive anomalies are falsely reported as negative; (b) 1996 NSF cruise NBP9604 (88010008) appears to have non-constant DC shifted gravity requiring reprocessing and close manual inspection.



(a)



(b)

Figure 5.10: More trackline errors requiring manual processing: (a) 1975 Scripps cruise ERDC07WT (15040015) showing PDR wrap-around effects identical to those reported in Figure 1.6; (b) free-air gravity from the 1986 Orstom cruise V1300 (NGDC id 84010015) should be removed entirely as its acquisition was apparently not required or monitored.

Chapter 6

DISCUSSION

6.1 Assessment of trackline data quality

As found in previous studies (e.g. *Wessel and Watts* [1988] and *Smith* [1993]), along-track analysis and grid comparison of ship data performed in this study indicate the presence of raw, unprocessed data in NGDC's marine geophysics trackline archive. Excessive gradients, grid offsets, DC shifts and improperly scaled data attest to a lack of rigorous processing prior to data submittal to NGDC. Furthermore, the presence of crude errors, such as navigation on land and non-increasing time, suggests that NGDC's basic quality control procedures are clearly inadequate.

While scientists requisitely ensure data quality prior to publication of work dependent upon marine geophysical data, this processed version may not always be submitted to NGDC for archival. Processing procedures vary for each institution, with differing interactions between shipboard technicians, scientists, and archivers. Furthermore, the two year deadline for data submittal to NGDC which applies to U.S. NSF funded expeditions may inadvertently promote early submittal of raw or marginally processed data.

6.2 What to do about detected errors

These new methods for error detection quickly identify data problems and are therefore useful to both survey institutions and data archivers for preventing further archival of unprocessed data. Repairing historic data on the other hand, is a more complex task. The optimal solution is re-submittal of fully processed data by source institutions. This is not always feasible, however, due to lack of funds, original data

logs, and original personnel. Recent cruises can be improved and re-submitted if institutions so desire.

The E77 errata system developed in this research provides an interface for automated correction of poor quality ship data without modifying original data. Scientists must interpret the presented error types to determine actions to take such as removal, correction, or no action. Close oversight of data changes is required. By sharing these log files, duplication of processing by the scientific community can be minimized.

Scientists involved in MGD77 data archival and submittal may find the methods and software developed in this research a useful addition to their own processing toolkit. Prior to data transmittal to NGDC, institutions will be able to scan MGD77 files to ensure that cruise data will be accepted. The new mgd77 library of programs by *Wessel and Chandler* [2006] provides many new tools for data archivists and for scientists in general who are interested in extracting ship information.

6.3 Conclusion and Recommendations

In general, systematic errors, such as DC shifts and scale factors obtained through regression analysis, require global correction of a particular field throughout a cruise, while along-track errors, such as values and gradients out of range, require removal of that field within one record. Entire records need to be removed when gross errors such as extreme speed, navigation on land, or decreasing time, occur.

Although a great deal of effort is directed at obtaining reasonable error limits to calibrate the error checking methods developed in this research, these limits can be inconvenient when working in certain areas where average data values may be near the default geophysical limits. For this reason, the mgd77sniffer program has options for overriding default limits, adjusting maximum acceptable speed, deactivating or adjusting gap checking, and turning on or off different data warning messages. See

the mgd77sniffer manpage (see Appendix A.2) for more information.

Methods presented in this research locate errors in marine geophysical trackline data through along-track analysis and comparison of ship data with satellite-derived gravity and predicted bathymetry grids. All archived geophysical values, gradients, as well as grid comparison results are examined to understand data tendencies and to derive reasonable error detection limits. Substantial evidence is found suggesting the presence of raw, unprocessed data within the National Geophysical Data Center's marine geophysical trackline archive. These new methods may be useful both for NGDC and survey institutions to ensure quality of ship-track data prior to submittal and archival. The presented errata scheme further enables subsequent correction of ship-track data by scientists.

The methods discussed in this thesis are part of a larger effort to improve marine geophysical data quality. By implementing these methods at NGDC, problematic data sets will be more quickly identified. It is probably in the interests of taxpayers, funding agencies for marine research, and marine science in general, that publicly funded scientists be dissuaded from submitting raw, poorly processed data, or from withholding data for longer than two years. Once NGDC is able to identify data problems quickly, simple procedures can help stimulate processing and submittal. For instance, NSF may wish to delay funding proposals for PI's who have not submitted prior processed data to NGDC in a timely manner. At the same time, NGDC should not accept unprocessed data from publicly funded scientists, but instead return cruises that cannot pass the mgd77sniffer tests to the source institution for further processing. These two pressures should provide some impetus for timely data processing and submittal. It may also prove effective to require Principal Investigators to review final data prior to submittal to NGDC. After thoughtful review, the PI could require further processing or could sign a form stating that data are satisfactory and meet funding requirements. In cases where scientists gather additional

geophysical information for the benefit of science but are unable to closely monitor or post-process the extra information, scientists should indicate the processing state in the meta-data, or should refrain from calculating and submitting anomalies for the un-processed data. It may also be advantageous to encourage NGDC to undertake a "follow-up" correspondence with PIs or the source institution a few years after submittal to ascertain if a more fully processed version of a cruise might be available for re-submittal.

6.4 Future work

This thesis has addressed one of several key undertakings planned for improving the NGDC archive. Future work includes (a) making sure that these procedures are implemeted at NGDC, (b) undertaking a global COE analysis of all data and determining systematic errors not easily found by the along-track analysis, and (c) developing a MGD77 website, ultimately hosted by NGDC, where COE and E77 metadata as well as other information about MGD77 data, including available tools, will be collected.

Bibliography

Calmant, S., M. Berge-Nguyen, and A. Cazenave (2002), Global seafloor topography from a least-squares inversion of altimetry-based high-resolution mean sea surface and sparse shipboard soundings, *Geophys. J. Int.*, *151*, 795–808.

Dehlinger, P. (1978), *Marine Gravity*, 22, Elsevier Scientific Publishing Company.

Draper, N. R., and H. Smith (1998), *Applied Regression Analysis*, third ed., John Wiley & Sons, Inc.

Fujioka, K., K. Okino, T. Kanamatsu, and Y. Ohara (2002), Morphology and origin of the Challenger Deep in the Southern Mariana Trench, *Geophys. Res. Lett.*, *19*, 1–4.

Goff, J. A., and W. H. F. Smith (2004), The contributions of abyssal hill morphology and noise to altimetric gravity fabric, *Oceanography*, *17*(1), 24–37.

Graf, A., and R. Schultze (1961), Improvements on the sea gravimeter GSS-22, *J. Geophys. Res.*, *66*, 1813–1821.

Haxby, W. F., G. D. Karner, J. L. LaBrecque, and J. K. Weissel (1983), Digital images of combined oceanic and continental data sets and their use in tectonic studies, *EOS Transactions, American Geophyical Union*, *64*(52), 995–1004.

Heezen, B., and M. Tharp (1977), *World Ocean Floor*, U.S. Navy, Off. Naval Res., USA.

Hey, R. (1977), A new class of "pseudofaults" and their bearing on plate tectonics: a propagating rift model, *Earth Planet. Sci. Lett.*, *37*, 321–325.

Hittleman, A. M., R. C. Groman, T. L. Haworth, T. L. Holcombe, G. McHendrie, and S. M. Smith (1977), The Marine Geophysical Data Exchange Format MGD77 - key to geophysical records documentation 10, *Tech. rep.*, National Geophysical Data Center, Boulder, Colorado, USA.

Kernighan, B. W., and D. M. Ritchie (1988), *The C Programming Language*, 2 ed., Prentice Hall.

Lacoste, L. J. B. (1959), Surface ship gravity measurements on the Texas A&M College ship Hidalgo, *Geophysics*, *24*, 309–322.

Marks, K. M., and W. H. F. Smith (2006), An evaluation of publicly available global bathymetry grids, in press.

Neumann, G. A., D. W. Forsyth, and D. T. Sandwell (1993), Comparison of marine gravity from shipboard and high-density satellite altimetry along the Mid-Atlantic Ridge, 30.5°-35.5°S, *Geophysical Research Letters*, *20*(15), 1639–1642.

Oreskes, N. (2002), *Plate Tectonics: An Insider's History of the Modern Theory of the Earth*, Westview Press.

Pitman, W. C., and J. R. Heirtzler (1966), Magnetic anomalies over the Pacific-Antarctic Ridge, *Science*, *154*(3753), 1164–1171.

Prince, M. (2005), UNOLS fleet cost projection for 2006, unpublished.

Raff, A., and R. Mason (1961), A magnetic survey off the west coast of North America, 40°N to 52.5°N, *Bulletin of the Geological Society of America*, *72*(8), 1267 – 1270.

Rousseuw, P. J., and A. M. Leroy (1986), *Robust Regression and Outlier Detection*, Wiley Series in Probability and Statistics, John Wiley & Sons, Inc.

Runcorn, S. K. (1956), Paleomagnetic comparisons between Europe and North America, *Proc. Geol. Assoc. Can.*, *8*, 77–85.

Sandwell, D. T., and W. H. F. Smith (1997), Marine gravity anomaly from Geosat and ERS-1 satellite marine gravity field, *Journal of Geophysical Research*, *102*, 10,039–10,054.

Smith, W. H. F. (1993), On the accuracy of digital bathymetric data, *Journal of Geophysical Research*, *98*(B6), 9591–9603.

Smith, W. H. F., and D. T. Sandwell (1994), Bathymetric prediction from dense satellite altimetry and sparse shipboard bathymetry, *Journal of Geophysical Research*, *99*(B11), 21,803–21,824.

Smith, W. H. F., and D. T. Sandwell (1997), Global sea floor topography from satellite altimetry and ship depth soundings, *Science*, *277*(5334), 1893–2088.

Sykes, L. R. (1963), Seismicity of the South Pacific Ocean, *Journal of Geophysical Research*, *68*(21), 5999–6006.

Vine, F. J., and D. H. Matthews (1963), Magnetic anomalies over oceanic ridges, *Nature*, *199*(4897), 947–949.

Vogt, P., W.-Y. Jung, and D. Nagel (2000), GOMaP: A matchless resolution to start the new millenium, *EOS Transactions, American Geophysical Union*, *81*(23), 254 – 258.

Wessel, P. (2001), Global distribution of seamounts inferred from gridded Geosat/ERS-1 altimetry, *Journal of Geophysical Research*, *106*(B9), 19,431–19,441.

Wessel, P., and M. T. Chandler (2006), The mgd77 supplement to the Generic Mapping Tools, in review.

Wessel, P., and W. H. F. Smith (1998), New, improved version of Generic Mapping Tools released, *EOS Transactions, American Geophyical Union*, *79*(47), 579.

Wessel, P., and A. B. Watts (1988), On the accuracy of marine gravity measurements, *Journal of Geophysical Research*, *93*(B1), 393–413.

Appendix A
Sniffer Program

A.1 Source Code

```
/* ------------------------------------------------------------------
 * $Id: mgd77sniffer.c,v 1.51 2006/05/02 05:10:20 mtchandl Exp $
 *
 * File: mgd77sniffer.c
 *
 * mgd77sniffer scans MGD77 files for errors in three ways: one, point-
 * by-point scanning to determine if values are within reasonable limits;
 * two, along-track scanning to find excessive changes in values; three,
 * comparing ship gathered gravity and topography data with global
 * reference grids for errors. Errors are reported to standard output
 * by default with optional output of structured "E77" errata tables.
 *
 * Authors:
 *    Michael Chandler and Paul Wessel
 *    School of Ocean and Earth Science and Technology
 *    University of Hawaii
 *
 * Date: March 2006
 *
 * ------------------------------------------------------------------*/

#include "mgd77.h"
#include "mgd77sniffer.h"

/*
#define HISTOGRAM_MODE 0
#define FIX 0
#define OUTPUT_TEST 0
*/

#if OUTPUT_TEST == 1
#define OR_TRUE || 1
#define AND_FALSE && 0
#else
#define OR_TRUE
#define AND_FALSE
#endif

int main (int argc, char **argv) {

    /* THE FOLLOWING VARIABLES DO NOT VARY FOR EACH CRUISE */
    int argno, n_alloc = GMT_CHUNK, n_cruises = 0, pos = 0, n_grids = 0, n_out_columns;
    unsigned int MGD77_this_bit[32], n_types[N_ERROR_CLASSES], n_bad_sections = 0;

    double time_factor = 1.0, distance_factor = 1.0, maxTime, west=0.0, east=0.0, north=0.0, south
        =0.0, adjustDC[32];
    double test_slope[5] = {0.1, 10.0, 0.5337, MGD77_METERS_PER_FATHOM, MGD77_FATHOMS_PER_METER},
        adjustScale[32];
    double max_speed, MGD77_NaN, maxSlope[MGD77_N_NUMBER_FIELDS], maxGap, threshold = 1.0;
    float *f[8];
    time_t clock;

    char c, tmp_min[16], tmp_max[16], tmp_maxSlope[16], tmp_area[16], *derivative;
    char *custom_limit_file = NULL, custom_limit_line[BUFSIZ], arguments[BUFSIZ];
    char field_abbrev[8], *speed_units = "m/s", *distance_units = "km";
    char file[BUFSIZ], *display = NULL;

    BOOLEAN error = FALSE, nautical = FALSE, custom_max_speed = FALSE, simulate = FALSE, bilinear =
        FALSE, bad_sections = FALSE;
    BOOLEAN custom_warn = FALSE, warn[MGD77_N_WARN_TYPES], custom_maxGap = FALSE, decimate = TRUE,
        adjustData = FALSE;

    FILE *custom_fp, *fpout = NULL;

    struct MGD77_SNIFFER_DEFAULTS mgd77snifferdefs[MGD77_N_DATA_FIELDS] = {
#include "mgd77snifferdefaults.h"
    };

    struct BAD_SECTION BadSection[MAX_BAD_SECTIONS];

    /* THESE VARIABLES VARY FOR EACH CRUISE AND REQUIRE EXTRA CARE (RESET FOR EACH CRUISE) */
    int i, j, k, curr, nwords, nout, nvalues, distanceErrorCount, duplicates[MGD77_N_NUMBER_FIELDS
        ], *iMaxDiff = NULL, n_nan;
    int noTimeCount, noTimeStart, timeErrorCount, timeErrorStart, distanceErrorStart, overLandStart
        , overLandCount, last_day;
    int **bin2d, ship_bin, grid_bin, n, npts, *offsetStart, rec, type, field, bccCode, col;
    unsigned int lowPrecision, lowPrecision5;

    double gradient, dvalue, dt, ds, **out, thisArea, speed, **G = NULL, min, *distance;
```

```
    double *offsetArea, stat[8], stat2[8], *ship_val, *grid_val, max, tcrit, se, range;
    double thisLon, thisLat, lastLon, lastLat, *MaxDiff = NULL, **diff = NULL, *decimated_ship;
    double *offsetLength, *decimated_grid, S_xx, recommended_scale;

    char timeStr[32], placeStr[64], errorStr[128], outfile[32], abbrev[8];

    BOOLEAN gotTime, landcruise, *offsetSign, newScale;
    BOOLEAN *prevOffsetSign, prevFlag, prevType, decimated;
#ifdef FIX
    BOOLEAN deleteRecord = FALSE;
#endif

    /* INITIALIZE MEMORY FOR MGD77 DATA STRUCTURES */
    struct MGD77_DATA_RECORD *D;
    struct MGD77_HEADER H;
    struct MGD77_CONTROL M, Out;
    struct MGD77_GRID_INFO this_grid[8];
    struct MGD77_ERROR *E = NULL;
    struct tm *systemTime;
    struct MGD77_CARTER C;

    /* INITIALIZE GMT & MGD77 MACHINERY */
    argc = GMT_begin (argc, argv);
    gmtdefs.time_system = 4;
    clock = time (NULL);
    maxTime = GMT_dt_from_usert ((double) clock);
    systemTime = gmtime (&clock);
    MGD77_Init (&M, TRUE);
    MGD77_Init (&Out, TRUE);
    MGD77_carter_init (&C);
    Out.fp = GMT_stdout;
    GMT_make_dnan (MGD77_NaN);

    /* INITIALIZE E77 */
    n_types[E77_NAV] = N_NAV_TYPES;
    n_types[E77_VALUE] = N_DEFAULT_TYPES;
    n_types[E77_SLOPE] = N_DEFAULT_TYPES;
    n_types[E77_GRID] = N_DEFAULT_TYPES;

    /* TURN ON MGD77SNIFFER ERROR MESSAGES */
    for (i = 0; i<MGD77_N_WARN_TYPES; i++) warn[i] = TRUE;

    /* SET PROGRAM DEFAULTS */
    arguments[0] = 0;
    for (i = 0; i < MGD77_N_DATA_FIELDS; i++) MGD77_this_bit[i] = 1 << i;
    maxGap = (double) MGD77_MAX_DS;  /* 5 km by default */
    n_out_columns = 0;           /* No formatted output */
    M.verbose_level = 3; /* 1 = warnings, 2 = errors, 3 = both */
    M.verbose_dest = 1;      /* 1 = stdout, 2 = stderr */
    max_speed = MGD77_MAX_SPEED;
    derivative = "SPACE";
    for (i = 0; i<MGD77_N_NUMBER_FIELDS; i++) {
       maxSlope[i] = mgd77snifferdefs[i].maxSpaceGrad; /* Use spatial gradients by default */
       adjustScale[i] = MGD77_NaN;
       adjustDC[i] = MGD77_NaN;
    }
    memset ((void *)this_grid, 0, 8 * sizeof (struct MGD77_GRID_INFO));

    /* READ COMMAND LINE ARGUMENTS */
    for (i = 1; i < argc; i++) {
       if (argv[i][0] == '-') {
          sprintf(arguments,"%s %s",arguments,argv[i]);
          switch (argv[i][1]) {
             case 'b':
             case 'V':
             case 'R':
             case '\0':
                error += GMT_parse_common_options (argv[i], &west, &east, &south, &north);
                break;
             case 'A':   /* adjust slope and intercept */
                if (!error && sscanf (&argv[i][2], "%[^,]", abbrev) != 1) {
                   fprintf (stderr, "%s: SYNTAX ERROR -A option: Give field abbreviation, slope and
                         intercept\n", GMT_program);
                   error = TRUE;
                }
                /* Find what column number this field corresponds to (i.e. depth == 11) */
                col = 0;
                while (strcmp (abbrev, mgd77defs[col].abbrev) && col < MGD77_N_NUMBER_FIELDS)
                   col++;
                if (col == MGD77_N_NUMBER_FIELDS) {
                   fprintf (stderr, "%s: SYNTAX ERROR -A option: invalid field abbreviation\n",
                         GMT_program);
                   error = TRUE;
                }
                if (!error && sscanf (&argv[i][2], "%[^,],%lf,%lf", abbrev, &adjustScale[col], &
                      adjustDC[col]) != 3) {
                   fprintf (stderr, "%s: SYNTAX ERROR -A option: Give field abbreviation,slope,
                         intercept\n", GMT_program);
                   error = TRUE;
                }
                adjustData = TRUE;
                break;
```

```
case 'C':   /* set max speed */
   max_speed = atof (&argv[i][2]);
   custom_max_speed = TRUE;
   break;
case 'D':
   if (argv[i][2] == 'd') { /* cruise - grid differences */
      display = "DIFFS";
      n_out_columns = 6;
   }
   else if (argv[i][2] == 'e') { /* E77 error output */
      display = "E77";
      n_out_columns = 6;
   }
   else if (argv[i][2] == 'f') { /* deltaZ and deltaS for each field */
      display = "DFDS";
      n_out_columns = 20;
   }
   else if (argv[i][2] == 'l') { /* Sniffer limits */
      display = "LIMITS";
      n_out_columns = 4;
   }
   else if (argv[i][2] == 'm') { /* MGD77 output */
      display = "MGD77";
      n_out_columns = 27;
   }
   else if (argv[i][2] == 's') { /* gradients */
      display = "SLOPES";
      n_out_columns = 11;
   }
   else if (argv[i][2] == 'v') { /* values */
      display = "VALS";
      n_out_columns = 12;
   }
   else {
      fprintf (stderr, "%s: SYNTAX ERROR:  Unrecognized option -%c%c\n", GMT_program,\
      argv[i][1], argv[i][2]);
      error = TRUE;
   }
   /* Silence all warning messages for data dumps */
   for (j = 0; j<MGD77_N_WARN_TYPES; j++) warn[j] = FALSE;
   M.verbose_dest = 2;      /* 1 = stdout, 2 = stderr */
   break;
case 'F':   /* fake mode (specify field and constant z value in -G - no grid reading
    */
   simulate = TRUE;
   break;
case 'g':   /* Get grid filename and geophysical field name to compare with grid */
   this_grid[n_grids].format = 1;   /* Mercator grid */
   if (sscanf (&argv[i][2], "%[^,],%[^,],%lf,%d,%lf", this_grid[n_grids].abbrev,
         this_grid[n_grids].fname, &this_grid[n_grids].scale, &this_grid[n_grids].mode,
          &this_grid[n_grids].max_lat) < 4) {
      fprintf (stderr, "%s: SYNTAX ERROR -g option: Give field abbreviation, grdfile,
            scale, mode [, and optionally max lat]\n", GMT_program);
      error = TRUE;
   }
case 'G':   /* Get grid filename and geophysical field name to compare with grid */
   if (!error && this_grid[n_grids].format == 0 && sscanf (&argv[i][2], "%[^,],%s",
         this_grid[n_grids].abbrev, this_grid[n_grids].fname) != 2) {
      fprintf (stderr, "%s: SYNTAX ERROR -G option: Give field abbreviation and
            grdfile\n", GMT_program);
      error = TRUE;
   }
   else {
      /* Find what column number this field corresponds to (i.e. depth == 11) */
      this_grid[n_grids].col = 0;
      while (strcmp (this_grid[n_grids].abbrev, mgd77defs[this_grid[n_grids].col].
            abbrev) &&\
      this_grid[n_grids].col < MGD77_N_NUMBER_FIELDS)
         this_grid[n_grids].col++;
      if (this_grid[n_grids].col == MGD77_N_NUMBER_FIELDS) {
         fprintf (stderr, "%s: SYNTAX ERROR -G option: invalid field abbreviation\n",
               GMT_program);
         error = TRUE;
      }
      if (!strcmp (this_grid[n_grids].abbrev,"depth")) this_grid[n_grids].sign = -1;
      else this_grid[n_grids].sign = 1;
      n_grids++;
   }
   break;
case 'H':   /* Avoid decimation during grid comparison */
   decimate = FALSE;
   break;
case 'I':   /* Pass ranges of data records to ignore for output to E77 */
   if (!error && sscanf (&argv[i][2], "%[^,],%d,%d", BadSection[n_bad_sections].abbrev
         , &BadSection[n_bad_sections].start, &BadSection[n_bad_sections].stop) != 3) {
      fprintf (stderr, "%s: SYNTAX ERROR -A option: Give field abbreviation,slope,
            intercept\n", GMT_program);
      error = TRUE;
   }
   /* Find what column number this field corresponds to (i.e. depth == 11) */
   col = 0;
```

```
            while (strcmp (BadSection[n_bad_sections].abbrev, mgd77defs[col].abbrev) && col <
                MGD77_N_NUMBER_FIELDS)
               col++;
            if (col == MGD77_N_NUMBER_FIELDS) {
               fprintf (stderr, "%s: SYNTAX ERROR -I option: invalid field abbreviation\n",
                     GMT_program);
               error = TRUE;
            }
            bad_sections = TRUE;
            BadSection[n_bad_sections].col = col;
            n_bad_sections++;
            if (n_bad_sections == MAX_BAD_SECTIONS) {
               fprintf (stderr, "%s: SYNTAX ERROR -I option: Max number of sections (%d)
                     reached\n", GMT_program, MAX_BAD_SECTIONS);
               error = TRUE;
            }
            break;
         case 'L':   /* Overwrite default sniffer limits */
            custom_limit_file = &argv[i][2];
            break;
         case 'N':   /* Change to nautical units instead of metric */
            nautical = TRUE;
            speed_units = "knots";
            distance_units = "nm";
            break;
         case 'Q':   /* bilinear interpolation parameters */
            bilinear = TRUE;
            threshold = (argv[i][2]) ? atof (&argv[i][2]) : 1.0;
            break;
         case 'S':   /* Specify spatial, time, or simple difference gradients */
            if (argv[i][2] == 'd') {
               derivative = "DIFF";
               for (j = 0; j<MGD77_N_NUMBER_FIELDS; j++) maxSlope[j] = mgd77snifferdefs[j].
                     maxTimeGrad;
            }
            else if (argv[i][2] == 's') {
               derivative = "SPACE";
               for (j = 0; j<MGD77_N_NUMBER_FIELDS; j++) maxSlope[j] = mgd77snifferdefs[j].
                     maxSpaceGrad;
            }
            else if (argv[i][2] == 't') {
               derivative = "GMT_TIME";
               for (j = 0; j<MGD77_N_NUMBER_FIELDS; j++) maxSlope[j] = mgd77snifferdefs[j].
                     maxTimeGrad;
            }
            else {
               fprintf (stderr, "%s: SYNTAX ERROR:  Unrecognized option -%c%c\n", GMT_program,
                     argv[i][1],\
               argv[i][2]);
               error = TRUE;
            }
            break;
         case 'T':   /* Specify maximum gap between records */
            custom_maxGap = TRUE;
            maxGap = atof (&argv[i][2]);
            if (maxGap < 0) {
               fprintf (stderr, "%s: SYNTAX ERROR -M option: max gap cannot be negative\n",
                     GMT_program);
               error = TRUE;
            }
            break;
         case 'W':   /* Choose which warning types to go to stdout (default - all) */
            for (j = 0; j<MGD77_N_WARN_TYPES; j++) warn[j] = FALSE;
            while (GMT_strtok (&argv[i][2], ",", &pos, &c)) {
               if (c == 'v')
                  warn[VALUE_WARN] = TRUE;
               else if (c == 'g')
                  warn[SLOPE_WARN] = TRUE;
               else if (c == 'o')
                  warn[GRID_WARN] = TRUE;
               else if (c == 't')
                  warn[TIME_WARN] = TRUE;
               else if (c == 's')
                  warn[SPEED_WARN] = TRUE;
               else if (c == 'c')
                  warn[TYPE_WARN] = TRUE;
               else if (c == 'x')
                  warn[SUMMARY_WARN] = TRUE;
               else {
                  fprintf (stderr, "%s: SYNTAX ERROR:  Unrecognized option -%c%c\n",
                        GMT_program,\
                     argv[i][1], argv[i][2]);
                  error = TRUE;
               }
            }
            custom_warn = TRUE;
            break;
         default:
            fprintf (stderr, "%s: SYNTAX ERROR:  Unrecognized option -%c\n", GMT_program, argv[
                  i][1]);
            error = TRUE;
            break;
```

```
            }
325         }
            else
               n_cruises++;
        }

        /* DISPLAY ERROR PAGE */
        if (GMT_give_synopsis_and_exit || argc == 1) {  /* Display usage */
            fprintf(stderr,"mgd77sniffer %s - Along-track quality control of MGD77 cruises\n\n",
                MGD77_VERSION);
            fprintf (stderr, "usage:  mgd77sniffer <cruises> [-Afieldabbrev,scale,offset] [-Cmaxspd] [-
                Dd|e|f|l|m|s|v]\n");
            fprintf (stderr, "\t[-gfieldabbrev,imggrid,scale,mode[,latmax]] [-Gfieldabbrev,grid] [-H] [-
                Ifieldabbrev,rec1,recN] [-Lcustom_limits_file]\n");
            fprintf (stderr, "\t[-N] [-Q[<value>]] [%s] [-Sd|s|t] [-Tgap] [-Wc|g|o|s|t|v|x] [-V] [%s]\n\
                n", GMT_Rgeo_OPT, GMT_bo_OPT);
            if (GMT_give_synopsis_and_exit) exit (EXIT_FAILURE);
            fprintf (stderr, "\tScan MGD77 files for errors using point-by-point sanity checking,\n");
            fprintf (stderr, "\t\talong-track detection of excessive slopes and comparison of cruise\n")
                ;
            fprintf (stderr, "\t\tdata with global bathymetry and gravity grids.");
            fprintf (stderr, "\twhere <cruises> is one or more MGD77 legnames, e.g. 08010001 etc.\n");
            fprintf (stderr, "\n\tOPTIONS:\n");
            fprintf (stderr, "\t-A Apply scale factor and DC adjustment to specified data field. Allows
                adjustment of\n");
            fprintf (stderr, "\t   cruise data prior to along-track analysis. CAUTION: data must be
                thoroughly examined\n");
            fprintf (stderr, "\t   before applying these global data adjustments. May not be used for
                multiple cruises.\n");
            fprintf (stderr, "\t-C Set maximum ship speed (10 m/s by default, use -N to indicate knots)\
                n");
            fprintf (stderr, "\t-D Dump cruise data such as sniffer limits, values, gradients and mgd77
                records.\n");
            fprintf (stderr, "\t  -Dd print out cruise-grid differences (requires -G option)\n");
            fprintf (stderr, "\t  -De output formatted error summary for each record. See E77 ERROR
                FORMAT below.\n");
            fprintf (stderr, "\t  -Df for each field, output value change and distance since last
                observation.\n");
350         fprintf (stderr, "\t  -Dl print out mgd77sniffer default limits (requires no additional
                arguments)\n");
            fprintf (stderr, "\t  -Dm print out MGD77 format\n\t  -Ds print out gradients\n\t  -Dv print
                 out values\n");
            fprintf (stderr, "\t-g Compare cruise data to the specified Sandwell/Smith Mercator grid.
                Requires valid MGD77\n");
            fprintf (stderr, "\t   field abbreviation followed by a comma, the path (if not in current
                directory)\n");
            fprintf (stderr, "\t   and grid filename, scale (0.1 or 1), and mode (see mgd77manage for
                details)\n");
            fprintf (stderr, "\t   Optionally, append max latitude in the IMG file [72.0059773539]\n");
            fprintf (stderr, "\t-G Compare cruise data to the specified GMT geographic grid. Requires
                valid MGD77 field abbreviation\n");
            fprintf (stderr, "\t   followed by a comma, then the path (if not in current directory) and
                grid filename.\n");
            fprintf (stderr, "\t   Excessive offsets are flagged according to maxArea threshold (use -L
                option to\n");
            fprintf (stderr, "\t   adjust maxArea). Useful for comparing faa or depth to global grids
                though any MGD77\n");
            fprintf (stderr, "\t   field can be compared to any GMT or IMG compatible grid. Multiple
                grid comparison is\n");
            fprintf (stderr, "\t   supported by  using  separate -G or -g calls for each grid. See GRID
                FILE INFO below.\n");
            fprintf (stderr, "\t-H (with -G|g only) disable data decimation during RLS regression..\n");
            fprintf (stderr, "\t-I Give one or more times to specify ranges of data record that should
                be flagged as bad\n");
            fprintf (stderr, "\t   prior to along-track analysis.  The flag information will be echoed
                out to E77 files.\n");
            fprintf (stderr, "\t   May not be used for multiple cruises.\n");
            fprintf (stderr, "\t-L Override mgd77sniffer default error detection limits. Supply path and
                 filename of\n");
            fprintf (stderr, "\t   the custom limits file. Rows not beginning with a valid MGD77 field
                abbreviation are\n");
            fprintf (stderr, "\t   ignored. Field abbreviations are listed below in exact form under
                MGD77 FIELD INFO.\n");
            fprintf (stderr, "\t   Multiple field limits may be modified using one default file, one
                field per line.\n");
            fprintf (stderr, "\t   Field min, max, maxGradient and maxArea may be changed for each field
                . maxGradient\n");
            fprintf (stderr, "\t   pertains to the gradient type selected using the -S option. maxArea
                is used by the\n");
            fprintf (stderr, "\t   -G option as the threshold for flagging excessive offsets. Dump
                defaults (-Dd) to\n");
            fprintf (stderr, "\t   view syntax or to quickly create an editable custom limits file.\n");
            fprintf (stderr, "\t   Example custom default file contents (see below for field units):\n")
                ;
375         fprintf (stderr, "\t\tdepth    0   11000 1000   4500\n");
            fprintf (stderr, "\t\tmag   -800  800    -   -\n");
            fprintf (stderr, "\t\tfaa   -250  250    100    2500\n");
            fprintf (stderr, "\t   Use a dash '-' to retain a default limit.\n");
            fprintf (stderr, "\t   Hint: to test your custom limits, try: mgd77sniffer -Dl -L<
                yourlimitsfile>\n");
            fprintf (stderr, "\t-N Use nautical units.\n");
```

```
fprintf (stderr, "\t-Q Quick mode, use bilinear rather than bicubic interpolation (requires -g or -G).\n");
fprintf (stderr, "\t   Optionally, append <value> in the 0 <= value <= 1 range.\n");
fprintf (stderr, "\t   [Default = 1 requires all 4 nearest nodes to be non-NaN.], <value> = 0.5\n");
fprintf (stderr, "\t   will interpolate about 1/2 way from a non-NaN to a NaN node, while\n");
fprintf (stderr, "\t   0.1 will go about 90%% of the way, etc.\n");
fprintf (stderr, "\t   -Q0 means only consider the nearest node (no interpolation).\n");
fprintf (stderr, "\t-S Specify gradient type for along-track excessive slope  checking.\n");
fprintf (stderr, "\t  -Sd Calculate change in z values along track (dz)\n");
fprintf (stderr, "\t  -Ss Calculate spatial gradients (dz/ds) [default]\n");
fprintf (stderr, "\t  -St Calculate time gradients (dz/dt)\n");
fprintf (stderr, "\t-T Set maximum acceptable distance gap between records (km) [5].\n");
fprintf (stderr, "\t   Set to zero to deactivate gap checking .\n");
fprintf (stderr, "\t-W Print out only certain warning types. Comma delimit any combination of c|g|o|s|t|v|x:\n");
fprintf (stderr, "\t   where (c) type code warnings, (g)radient out of range, (o)ffsets from grid (requires -G),\n");
fprintf (stderr, "\t   (s)peed out of range, (t)ime warnings, (v)alue out of range, (x) warning summaries.\n");
fprintf (stderr, "\t   By default ALL warning messages are printed. Not allowed with -D option.\n");
fprintf (stderr, "\t-V runs in verbose mode.\n\n");
fprintf (stderr, "\t-b output binary data for -D option.  Append d for double and s for single precision [double].\n\n");
fprintf (stderr, "\tMGD77 FIELD INFO:\n");
fprintf (stderr, "\tField\t\t\tAbbreviation\t\tUnits\n");
fprintf (stderr, "\tTwo-way Travel Time\ttwt\t\t\tsec\n");
fprintf (stderr, "\tCorrected Depth \tdepth\t\t\tm\n");
fprintf (stderr, "\tMag Total Field1\tmtf1\t\t\tnT\n");
fprintf (stderr, "\tMag Total Field2\tmtf2\t\t\tnT\n");
fprintf (stderr, "\tResidual Magnetic\tmag\t\t\tnT\n");
fprintf (stderr, "\tDiurnal Correction\tdiur\t\t\tnT\n");
fprintf (stderr, "\tMag Sensor Depth/Alt\tmsd\t\t\tm\n");
fprintf (stderr, "\tObserved Gravity\tgobs\t\t\tmGal\n");
fprintf (stderr, "\tEotvos Correction\teot\t\t\tmGal\n");
fprintf (stderr, "\tfree-air Anomaly\tfaa\t\t\tmGal\n\n");
fprintf (stderr, "\tGRID FILE INFO:\n");
fprintf (stderr, "\t-g: Img files must be of Sandwell/Smith signed two-byte integer (i2) type with no header.\n");
fprintf (stderr, "\t-G: Grid files can be any type of GMT grid file (native or netCDF) with header\n");
fprintf (stderr, "\tA correctly formatted grid file can be generated as follows:\n");
fprintf (stderr, "\t   e.g. gmtset GRIDFILE_SHORTHAND TRUE\n");
fprintf (stderr, "\t\tCreate/edit .gmt_io file to include the following rows:\n");
fprintf (stderr, "\t\t\t# GMT I/O shorthand file\n");
fprintf (stderr, "\t\t\t# suffix   format_id scale offset       NaN\n");
fprintf (stderr, "\t\t\tgrd             0       -       -       -\n");
fprintf (stderr, "\t\t\ti2              2       -       -       32767\n");
fprintf (stderr, "\t\tgrdraster 1 -R0/359:55/-90/90 -Getopo5_hdr.i2\n\n");
fprintf (stderr, "E77 ERROR OUTPUT\n");
fprintf (stderr, "\tError output is divided into (1) a header containing information globally\n");
fprintf (stderr, "\tapplicable to the cruise and (2) individual error records summarizing all\n");
fprintf (stderr, "\tall  errors  encountered in each cruise record. One error record is output\n");
fprintf (stderr, "\tfor each  readable  cruise  record.\n");
fprintf (stderr, "\tError Record Format: <time/distance> <record  number> <error code string> <description>\n");
fprintf (stderr, "Example:\n\t# Cruise 06050010 ID L476WG MGD77 FILE VERSION: 19870415 N_RECS: 27268\n");
fprintf (stderr, "\t# Examined: Tue Feb 21 16:33:34 2006\n");
fprintf (stderr, "\t# Examiner: mtchandl\n");
fprintf (stderr, "\t# Arguments:  -De -Gfaa,/data/GRIDS/sandwell_smith11_hdr.i2\n");
fprintf (stderr, "\t# Errata: Header\n");
fprintf (stderr, "\tN-faa-E-01: Regression scale 0.108643 different from 1. Recommended: [10]\n");
fprintf (stderr, "\tN-faa-E-02: Regression offset -1.319411 different from 0\n");
fprintf (stderr, "\tI-twt-W-03: More recent bathymetry correction table available\n");
fprintf (stderr, "\tI-depth-W-04: Integer precision in depth\n");
fprintf (stderr, "\t# Errata: Data\n");
fprintf (stderr, "\t1976-06-26T08:15:00   2  C-0-0-0   nav: excessive speed\n");
fprintf (stderr, "\t1976-06-26T08:16:00   3  C-0-0-0   nav: excessive speed\n");
fprintf (stderr, "\n\tError Class Descriptions\n");
fprintf (stderr, "\tNAV (navigation):\t0 --> fine\n");
fprintf (stderr, "\t\tA --> time out of range\n");
fprintf (stderr, "\t\tB --> time decreasing\n");
fprintf (stderr, "\t\tC --> excessive speed\n");
fprintf (stderr, "\t\tD --> above sea level\n");
fprintf (stderr, "\t\tE --> lat undefined\n");
fprintf (stderr, "\t\tF --> lon undefined\n\n");
fprintf (stderr, "\tVAL (value)\t0 --> fine\n");
fprintf (stderr, "\t\tK --> twt invalid\n");
fprintf (stderr, "\t\tL --> depth invalid\n");
fprintf (stderr, "\t\tO --> mtf1 invalid\n");
fprintf (stderr, "\t\tetc.\n\n");
fprintf (stderr, "\tGRAD (gradient):\t0 --> fine\n");
fprintf (stderr, "\t\tK --> d[twt] excessive\n");
fprintf (stderr, "\t\tL --> d[depth] excessive\n");
fprintf (stderr, "\t\tO --> d[mtf1] excessive\n");
```

```
            fprintf (stderr, "\t\tetc.\n\n");
            fprintf (stderr, "\tGRID (grid comparison):\t0 --> fine\n");
            fprintf (stderr, "\t\tL --> depth offset\n");
            fprintf (stderr, "\t\tW --> faa offset\n");
            fprintf (stderr, "\t\tetc.\n");
            fprintf (stderr, "\nEXAMPLES:\n\tAlong-track excessive value and gradient checking:\n\t\
                  tmgd77sniffer 08010001\n");
            fprintf (stderr, "\tDump cruise gradients:\n\t\tmgd77sniffer 08010001 -Ds\n");
            fprintf (stderr, "\tTo compare cruise depth with ETOPO5 bathymetry and gravity with Sandwell
                  /Smith 2 min gravity version 11, try\n");
            fprintf (stderr, "\t\tmgd77sniffer 08010001 -Gdepth,/data/GRIDS/etopo5_hdr.i2 -gfaa,/data/
                  GRIDS/grav.11.2.img,0.1,1\n\n");
            exit (EXIT_FAILURE);
         }
         /* ENSURE VALID USE OF OPTIONS */
         if (n_cruises == 0 && display != "LIMITS") {
            fprintf(stderr, "%s: ERROR: No cruises given\n", GMT_program);
            exit (EXIT_FAILURE);
         }
         else if (GMT_io.binary[GMT_OUT] && !display) {
            fprintf(stderr, "%s: ERROR: -b option requires -D.\n", GMT_program);
475         exit (EXIT_FAILURE);
         }
         else if (custom_warn && display) {
            fprintf(stderr, "%s: ERROR: Incompatible options -D and -W.\n", GMT_program);
            exit (EXIT_FAILURE);
         }
         else if (display == "DIFFS" && n_grids == 0) {
            fprintf(stderr, "%s: ERROR: -Dd option requires -G.\n", GMT_program);
            exit (EXIT_FAILURE);
         }
         if (east < west || south > north) {
            fprintf(stderr, "%s: ERROR: Region set incorrectly\n", GMT_program);
            exit (EXIT_FAILURE);
         }
         if (adjustData && n_cruises > 1) {
            fprintf(stderr, "%s: ERROR: -A adjustments valid for only one cruise.\n", GMT_program);
            exit (EXIT_FAILURE);
         }

         /* NAUTICAL CONVERSION FACTORS */
         if (nautical) {
            distance_factor = 1.0 / MGD77_METERS_PER_NM; /* meters to nm */
            time_factor = 1.0 / (3600); /* seconds to hours */
            /* Adjust max speed for unit change and user specified max speed */
            if (!custom_max_speed) max_speed = MGD77_MAX_SPEED * distance_factor / time_factor;
500      }

         /* READ AND APPLY CUSTOM LIMITS FILE */
         mgd77snifferdefs[MGD77_YEAR].maxValue = (double) systemTime->tm_year + 1900;
         if (custom_limit_file) {
            if ((custom_fp = fopen (custom_limit_file, "r")) == NULL) {
               fprintf (stderr, "%s: Could not open custom limit file %s\n", GMT_program,
                     custom_limit_file);
               exit (EXIT_FAILURE);
            }
            else {
               while (fgets (custom_limit_line, BUFSIZ, custom_fp)) {
                  if (sscanf (custom_limit_line,"%s %s %s %s %s", field_abbrev, tmp_min, tmp_max,
                        tmp_maxSlope, tmp_area) == 5) {
                     i = 0;
                     while (strcmp (mgd77snifferdefs[i].abbrev, field_abbrev) && i <=
                           MGD77_N_NUMBER_FIELDS) i++;
                     if (i <= MGD77_N_NUMBER_FIELDS) {
                        if (strcmp(tmp_min, "-"))  mgd77snifferdefs[i].minValue = atof (tmp_min);
                        if (strcmp(tmp_max, "-"))  mgd77snifferdefs[i].maxValue = atof (tmp_max);
                        if (strcmp(tmp_maxSlope, "-")) maxSlope[i] = atof (tmp_maxSlope);
                        if (strcmp(tmp_area, "-"))  mgd77snifferdefs[i].maxArea = atof (tmp_area);
                     }
                  }
                  else {
                     fprintf (stderr, "%s: Error in custom limits file [%s]\n", GMT_program,
                           custom_limit_line);
                     exit (EXIT_FAILURE);
                  }
525            }
            }
         }

         /* Use GMT time formatting */
         GMT_io.out_col_type[MGD77_TIME] = GMT_IS_ABSTIME;

         /* Adjust data dump for number of grids */
         if (display == "DIFFS")
            n_out_columns = 3+3*n_grids;

         /* PREPARE DUMP OUTPUT FORMATTING */
         if (display == "VALS" || display == "DIFFS") {
            for (i = MGD77_LATITUDE; i < (n_out_columns + MGD77_LATITUDE); i++) {
               /* Special lat formatting */
               if (i == MGD77_LATITUDE)
                  GMT_io.out_col_type[i-MGD77_LATITUDE] = GMT_IS_LAT;
```

```
            /* Special lon formatting */
            else if (i == MGD77_LONGITUDE)
               GMT_io.out_col_type[i-MGD77_LATITUDE] = GMT_IS_LON;

            /* Everything else is float */
            else
               GMT_io.out_col_type[i-MGD77_LATITUDE] = GMT_IS_FLOAT;
         }
      }
      else if (display) {
         for (i = 0; i < n_out_columns; i++) {
            /* All columns are floats */
            GMT_io.out_col_type[i] = GMT_IS_FLOAT;
         }
      }

      /* PRINT OUT DEFAULT GEOPHYSICAL LIMITS */
      if (display == "LIMITS") {
         fprintf (GMT_stdout, "#abbrev\tmin\tmax\tmaxSlope\tmaxArea\n");
         for (i = 0; i < MGD77_N_NUMBER_FIELDS; i++) {
            if ((1 << i) & (MGD77_GEOPHYSICAL_BITS + MGD77_CORRECTION_BITS)) {
               fprintf (GMT_stdout, "%s%s",mgd77defs[i].abbrev,gmtdefs.field_delimiter);
               GMT_ascii_output_one (GMT_stdout, mgd77snifferdefs[i].minValue, 0);
               fprintf (GMT_stdout, "%s",gmtdefs.field_delimiter);
               GMT_ascii_output_one (GMT_stdout, mgd77snifferdefs[i].maxValue, 1);
               fprintf (GMT_stdout, "%s",gmtdefs.field_delimiter);
               GMT_ascii_output_one (GMT_stdout, maxSlope[i], 2);
               fprintf (GMT_stdout, "%s",gmtdefs.field_delimiter);
               GMT_ascii_output_one (GMT_stdout, mgd77snifferdefs[i].maxArea, 3);
               fprintf (GMT_stdout, "\n");
            }
         }
         exit(EXIT_FAILURE);
      }

      /* PRINT HEADER FOR SNIFFER DATA DUMPS */
      if (display && !GMT_io.binary[GMT_OUT] && GMT_io.io_header[0]) {
         if (display == "SLOPES") {
            fprintf (GMT_stdout, "#speed(%s)%s",speed_units,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[twt]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[depth]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[mtf1]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[mtf2]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[mag]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[diur]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[msd]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[gobs]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[eot]%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[faa]\n");
         }
         else if (display == "DFDS") {
            fprintf (GMT_stdout, "#d[twt]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[depth]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[mtf1]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[mtf2]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[mag]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[diur]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[msd]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[gobs]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[eot]%sds%s",gmtdefs.field_delimiter,gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "d[faa]%sds\n",gmtdefs.field_delimiter);
         }
         else if (display == "VALS") {
            fprintf (GMT_stdout, "#lat%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "lon%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "twt%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "depth%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "mtf1%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "mtf2%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "mag%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "diur%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "msd%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "gobs%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "eot%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "faa\n");
         }
         else if (display == "DIFFS" && n_grids > 0) {
            fprintf (GMT_stdout, "#lat%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "lon%s",gmtdefs.field_delimiter);
            fprintf (GMT_stdout, "dist%s",gmtdefs.field_delimiter);
            for (i = 0; i < n_grids; i++) {
               fprintf (GMT_stdout, "crs_%s%s",this_grid[i].abbrev,gmtdefs.field_delimiter);
               fprintf (GMT_stdout, "grd_%s%s",this_grid[i].abbrev,gmtdefs.field_delimiter);
               fprintf (GMT_stdout, "diff%s",gmtdefs.field_delimiter);
            }
            fprintf (GMT_stdout, "\n");
         }
      }

      /* Open grid files */
      for (i = 0; i < n_grids; i++) {
```

```
            if (!simulate)
               /* Open and store grid file */
               read_grid (&this_grid[i], &f[i], west, east, south, north, bilinear, threshold);
         }

         if (n_grids) {
            MaxDiff = (double *) GMT_memory (VNULL, (size_t)n_grids, sizeof (double), "mgd77sniffer");
            iMaxDiff = (int *) GMT_memory (VNULL, (size_t)n_grids, sizeof (int), "mgd77sniffer");
         }

         MGD77_Ignore_Format (MGD77_FORMAT_ANY);    /* Reset to all formats OK, then ... */
         MGD77_Ignore_Format (MGD77_FORMAT_CDF);    /* disallow netCDF MGD77+ files */
         MGD77_Ignore_Format (MGD77_FORMAT_TBL);    /* and plain ASCI tables */

         /* PROCESS CRUISES */
         for (argno = 1; argno < argc; argno++) {

            if (argv[argno][0] == '-')
               continue;

            /* Verify valid cruise file */
            if (MGD77_Get_Path (file, argv[argno], &M)) {
                  fprintf (stderr, "%s : Cannot find leg %s\n", GMT_program, argv[argno]);
               continue;
            }

            /* Open cruise file */
            if (MGD77_Open_File (argv[argno], &M, 0))
               continue;

            if (gmtdefs.verbose)
               fprintf (stderr, "%s: Processing file %s\n", GMT_program, argv[argno]);

            if (display == "E77") {
               sprintf (outfile,"%s.e77",M.NGDC_id);
               if ((fpout = fopen (outfile, "w")) == NULL) {
                  fprintf (stderr, "%s: Could not open E77 output file %s\n", GMT_program, outfile);
                  exit (EXIT_FAILURE);
               }
            }

            /* Read MGD77 header */
            if (MGD77_Read_Header_Record_asc (argv[argno], &M, &H))
               fprintf (stderr, "%s: Cruise %s has no header.\n", GMT_program, argv[argno]);

            /* Allocate memory for data records */
            D = (struct MGD77_DATA_RECORD *) GMT_memory (VNULL, (size_t)n_alloc, \
            sizeof (struct MGD77_DATA_RECORD), "mgd77sniffer");

            /* READ DATA RECORDS */
            n_alloc = GMT_CHUNK;
            gotTime = FALSE;
            nvalues = M.bit_pattern[0] = 0;
            lowPrecision = lowPrecision5 = 0;
            while (!MGD77_Read_Data_Record_m77 (&M, &D[nvalues])) {
               /* Increase memory allocation if necessary */
               if (nvalues == n_alloc - 1) {
                  n_alloc += GMT_CHUNK;
                  D = (struct MGD77_DATA_RECORD *) GMT_memory ((void *)D, (size_t)n_alloc, \
                  sizeof (struct MGD77_DATA_RECORD), "mgd77sniffer");
               }
               if (!gotTime && !GMT_is_dnan(D[nvalues].time))
                  gotTime = TRUE;
               M.bit_pattern[0] |= D[nvalues].bit_pattern;
               nvalues++;
            }

            /* Scale and DC adjust if selected */
            if (adjustData) {
               for (i=0;i<MGD77_N_NUMBER_FIELDS;i++) {
                  if (!GMT_is_dnan(adjustScale[i]) || !GMT_is_dnan(adjustDC[i])) {
                     for (j=0; j<nvalues;j++)
                        D[j].number[i] = D[j].number[i] * adjustScale[i] + adjustDC[i];
                     fprintf (stderr,"%s (%s) Scaled by %f and %f added\n", GMT_program, mgd77defs[i].
                           abbrev,\
                     adjustScale[i],adjustDC[i]);
                  }
               }
            }

            /* Set user-specified flagged observations to NaN before analysis */
            if (bad_sections) {
               for (j=0;j<n_bad_sections;j++) { /* For each bad section */
                  for (i=BadSection[j].start-1;i<BadSection[j].stop;i++) { /* Loop over the flagged
                        records (adjust -1 for C index) */
                     D[i].number[BadSection[j].col] = MGD77_NaN;  /* and set them to NaN */
                  }
               }
            }

            /* Output beginning of E77 header */
            if (display == "E77") {
```

```
            fprintf (fpout, "# Cruise %s ID %s MGD77 FILE VERSION: %.4d%2.2d%2.2d N_RECS: %d\n",argv[
                argno],D[0].word[0],\
725         atoi(H.mgd77->File_Creation_Year),atoi(H.mgd77->File_Creation_Month),atoi(H.mgd77->
                File_Creation_Day),nvalues);
            fprintf (fpout,"# Examined: %s",ctime(&clock));
            fprintf (fpout,"# Examiner: %s\n",M.user);
            fprintf (fpout,"# Arguments: %s\n",arguments);
            fprintf (fpout,"# Errata: Header\n");
        }

        /* Scan MGD77 header */
        if (display == "E77" || warn[SUMMARY_WARN]) {
            MGD77_Verify_Prep_m77 (&M, &H.meta, D, nvalues);    /* First get key meta-data derived
                form data records */
            MGD77_Verify_Header (&M, &H, fpout);              /* Then verify the header information */
        }

        /* Re-set variables for this cruise */
        landcruise = FALSE;
        overLandCount = overLandStart = 0;
        timeErrorStart = noTimeStart = distanceErrorStart = -1;
        noTimeCount = timeErrorCount = distanceErrorCount = n_nan = bccCode = 0;
        offsetArea = (double *) GMT_memory (VNULL, (size_t)n_grids, sizeof (double), "mgd77sniffer")
            ;
        offsetStart = (int *)GMT_memory (VNULL, (size_t)n_grids, sizeof (int), "mgd77sniffer");
        offsetLength = (double *) GMT_memory (VNULL, (size_t)n_grids, sizeof (double), "mgd77sniffer
            ");
        offsetSign = (BOOLEAN *)GMT_memory (VNULL, (size_t)n_grids, sizeof (BOOLEAN), "mgd77sniffer"
            );
        prevOffsetSign = (BOOLEAN *)GMT_memory (VNULL, (size_t)n_grids, sizeof (BOOLEAN), "
            mgd77sniffer");
        range = S_xx = 0.0;
        prevFlag = FALSE;
750     for (i = 0; i<n_grids; i++) {
            offsetArea[i] = 0.0;
            offsetStart[i] = 0;
            offsetSign[i] = prevOffsetSign[i] = FALSE;
        }
        for (i = MGD77_LATITUDE; i<MGD77_N_NUMBER_FIELDS; i++) {
            /* Turn on low precision bits (will be turned off later if ok) */
            if ((M.bit_pattern[0] & (1 << i) & MGD77_FLOAT_BITS) && i != MGD77_MSD) lowPrecision |=
                (1 << i);
            if (M.bit_pattern[0] & (1 << i) & MGD77_FLOAT_BITS) lowPrecision5 |= (1 << i);
            duplicates[i] = 0;
        }
        /* Adjust along-track gradient type for time */
        if (derivative == "GMT_TIME" && !gotTime) {
            /*derivative = "SPACE";*/
            if (warn[TIME_WARN])
                fprintf (GMT_stdout, "%s Warning: cruise contains no time - time gradients invalid.\n"
                    ,GMT_program);
        }

        /* Allocate memory for distance array */
        distance = (double *) GMT_memory (VNULL, (size_t)nvalues, \
        sizeof (double), "mgd77sniffer");
        distance[0] = 0;

        /* Setup output array */
        out = (double **)GMT_memory (VNULL, nvalues, sizeof (double *), GMT_program);
775
        /* Allocate memory for error array */
        E = (struct MGD77_ERROR *) GMT_memory ((void *)E, (size_t)nvalues, \
        sizeof (struct MGD77_ERROR), "mgd77sniffer");

        /* PROCESS GRID FILES */
        if (n_grids > 0) {

            /* Allocate memory for 2D arrays */
            G = (double **)GMT_memory (VNULL, n_grids, sizeof (double *), GMT_program);    /* grid z
                values */
            diff = (double **)GMT_memory (VNULL, n_grids, sizeof (double *), GMT_program);    /*
                cruise-grid differences */

            for (i = 0; i < n_grids; i++) {

                /* Initialize variables */
                for (k=0; k<6; k++) stat[k] = stat2[k] = 0.0;
                tcrit = se = 0;
                newScale = FALSE;
                MaxDiff[i] = 0.0;

                this_grid[i].g_pts = 0;
                /* Skip if cruise lacks data field */
                if (!(M.bit_pattern[0] & (1 << this_grid[i].col))) {
                    fprintf (stderr, "%s: Warning: %s field not present in MGD77 file\n", GMT_program,
                        this_grid[i].abbrev);
                    if (this_grid[i].col != MGD77_DEPTH)
800                     continue;
                }

                /* Allocate memory for current array */
```

```
G[i] = (double *) GMT_memory (VNULL, (size_t)nvalues, sizeof (double), "mgd77sniffer")
    ;
diff[i] = (double *) GMT_memory (VNULL, (size_t)nvalues, sizeof (double), "
    mgd77sniffer");

/* Sample grid at each ship location */
if (simulate) {
   this_grid[i].g_pts = nvalues;
   for (j = 0; j < nvalues; j++)
      G[i][j] = 0.0;
}
else
   this_grid[i].g_pts = sample_grid (&this_grid[i], D, G, f[i], i, nvalues);

if (this_grid[i].g_pts < 2) {
      fprintf (stderr, "%s: Insufficient grid samples for %s comparison\n",
            GMT_program, this_grid[i].abbrev);
      continue;
}

/* Reverse grid sign if depth */
if (this_grid[i].sign == -1) {
   for (j = 0; j < nvalues; j++)
      G[i][j] *= this_grid[i].sign;
}

for (j = 0; j < nvalues; j++) {
   /* Compute cruise - grid differences */
   diff[i][j] = D[j].number[this_grid[i].col] - G[i][j];

   /* Count NaNs */
   if (GMT_is_dnan(D[j].number[this_grid[i].col]) || GMT_is_dnan(G[i][j]))
      n_nan++;
}

/* SHIP VS GRID RLS REGRESSION */
/* Allocate memory for NaN-free arrays */
ship_val = (double *) GMT_memory (VNULL, (size_t)nvalues-n_nan, sizeof (double), "
    mgd77sniffer");
grid_val = (double *) GMT_memory (VNULL, (size_t)nvalues-n_nan, sizeof (double), "
    mgd77sniffer");

/* Store grid/cruise pairs in NaN-free arrays */
for (j = k = 0; j < nvalues-n_nan; j++) {
   if (!GMT_is_dnan(D[j].number[this_grid[i].col]) && !GMT_is_dnan(G[i][j])) {
      ship_val[k] = D[j].number[this_grid[i].col];
      grid_val[k] = G[i][j];
      k++;
   }
}

/* Get max and min z values for this field */
min = mgd77snifferdefs[this_grid[i].col].minValue;
max = mgd77snifferdefs[this_grid[i].col].maxValue;
if (this_grid[i].col == MGD77_FAA) {
   min *= 2;
   max *= 2;
}

/* Create a 2-D bin table */
n = irint ((max - min)/mgd77snifferdefs[this_grid[i].col].delta) + 1;
bin2d = (int **)GMT_memory (VNULL, n, sizeof (int *), GMT_program);
for (j = 0; j < n; j++)
   bin2d[j] = (int *)GMT_memory (VNULL, n,  sizeof (int), GMT_program);

/* Then loop over all the ship,cruise pairs */
for (j = 0; j < nvalues-n_nan; j++) {
   /* Need to skip ship values that are outside of acceptable range */
   if (ship_val[j] >= min && ship_val[j] <= max) {
      ship_bin = irint ((ship_val[j] - min)/mgd77snifferdefs[this_grid[i].col].delta);
      grid_bin = irint ((grid_val[j] - min)/mgd77snifferdefs[this_grid[i].col].delta);
      bin2d[ship_bin][grid_bin]++;    /* Add up # of pairs in this bin */
   }
}

/* Then find how many binned pairs we got */
for (ship_bin = npts = 0; ship_bin < n; ship_bin++) {
    for (grid_bin = 0; grid_bin < n; grid_bin++) {
         if (bin2d[ship_bin][grid_bin] > 0)
         npts++;
   }
}

/* When all ship data are outside grid range skip regression */
if (npts > 2) {

   /* Then create arrays for passing to RLS */
   decimated_ship = (double *)GMT_memory (VNULL, npts, sizeof (double), GMT_program);
   decimated_grid = (double *)GMT_memory (VNULL, npts, sizeof (double), GMT_program);

   for (ship_bin  = k = 0; ship_bin < n; ship_bin ++) {
      for (grid_bin = 0; grid_bin < n; grid_bin ++) {
```

```
                        if (bin2d[ship_bin][grid_bin]) {
                            decimated_ship[k] = min + ship_bin * mgd77snifferdefs[this_grid[i].col].
                                delta;
                            decimated_grid[k] = min + grid_bin * mgd77snifferdefs[this_grid[i].col].
                                delta;
                            k++;  /* Count number of non-empty bins */
                        }
                    }
                }

                /* Decimation benefits marine gravity due to amplitude differences */
                /* between ship and satellite data and also broadens confidence  */
                /* intervals for any ship grid comparisons by reducing excessive */
                /* number of degrees of freedom */
                if (gmtdefs.verbose)
                    fprintf (stderr, "%s: Begin RLS Regression\n", GMT_program);
                if (!decimate) {
                    regress_rls (grid_val, ship_val, nvalues-n_nan, stat, this_grid[i].col, S_xx);
                    decimated = FALSE;
                    tcrit = GMT_tcrit (0.975, (double)nvalues-n_nan - 2.0);
                    npts=nvalues-n_nan;
                }
                else {
                    regress_rls (decimated_grid, decimated_ship, npts, stat, this_grid[i].col, S_xx)
                        ;
                    decimated = TRUE;
                    regress_rls (grid_val, ship_val, nvalues-n_nan, stat2, this_grid[i].col, S_xx);
                    if (stat[4] < stat2[4] && stat2[5] == 1.0) {
                        if (gmtdefs.verbose)
                            fprintf (stderr, "%s: Regression on undecimated data due to better
                                correlation\n",GMT_program);
                        for (k=0; k<6; k++) stat[k] = stat2[k];
                        npts=nvalues-n_nan;
                        decimated = FALSE;
                    }
                    tcrit = GMT_tcrit (0.975, (double)npts - 2.0);
                    if (stat[5] != 1.0) {
                        if (gmtdefs.verbose)
                            fprintf (stderr, "%s: RLS regression insignificant\n",GMT_program);
                    }
                }

                /* Analyze regression if significant */
                /* stat[0] is rls slope, stat[1] is rls icept, stat[2] is standard deviation, stat
                    [3] */
                /* is sum of squares, stat[4] is r, and stat[5] == 1.0 if this is a significant
                    correlation. */
                if (stat[5] == 1.0) {
                    /* Check if ship data are incorrectly scaled */
                    range = (tcrit * stat[2]) / sqrt(stat[3]);   /* Draper 1.4.8 */
                    if (1.0 <= (stat[0]-range) || 1.0 >= (stat[0]+range)) {
                        for (j = 0; j < 5; j++) {
                            if (test_slope[j] >= (stat[0]-range) &&\
                                test_slope[j] <= (stat[0]+range)) {
                                if (display == "E77") {
                                    if (!GMT_is_dnan(adjustScale[this_grid[i].col]) && fabs(adjustScale[
                                        this_grid[i].col]-1.0)>0.0)
                                        fprintf (fpout, "Y-%s-E-%.02d: Regression scale %.3f"\
                                        " different from 1. Recommended: [%g]\n",this_grid[i].abbrev,\
                                        E77_HDR_SCALE,stat[0],adjustScale[this_grid[i].col]);
                                    else {
                                        fprintf (fpout, "N-%s-E-%.02d: Regression scale %.3f"\
                                        " different from 1. Recommended: [%g]\n",this_grid[i].abbrev,\
                                        E77_HDR_SCALE,stat[0],1.0/test_slope[j]);
                                    }
                                }
                                else if (warn[SUMMARY_WARN])
                                    fprintf (GMT_stdout, "%s (%s) Slope %.3f different from 1."\
                                    " Recommended: [%g]\n",argv[argno],this_grid[i].abbrev,stat[0],1/
                                        test_slope[j]);
#ifdef FIX
                                /* Apply RLS slope to current field if new scale. */
                                for (k = 0; k < nvalues; k++) {
                                    D[k].number[this_grid[i].col] /= test_slope[j];
                                    diff[i][k] = D[k].number[this_grid[i].col] - G[i][k]; /* Re-compute
                                        cruise - grid differences */
                                }
                                if (gmtdefs.verbose)
                                    fprintf (stderr, "%s (%s) Warning: Scaled by %g for internal along-
                                        track analysis\n",\
                                    argv[argno],this_grid[i].abbrev, test_slope[j]);
#endif
                                newScale = TRUE;
                                /* If not depth comparison skip fathom check */
                                if (j == 1 && this_grid[i].col != MGD77_DEPTH) break;
                            }
                        }
                        if (!newScale) {
                            if (display == "E77") {
                                if (!GMT_is_dnan(adjustScale[this_grid[i].col]) && fabs(adjustScale[
                                    this_grid[i].col]-1.0)>0.0)
                                    fprintf (fpout, "Y-%s-E-%.02d: Regression scale %.3f"\
```

```
                                    " different from 1. Recommended: [%g]\n",this_grid[i].abbrev,\
                                    E77_HDR_SCALE,stat[0],adjustScale[this_grid[i].col]);
                                else {
975                                 recommended_scale = (!strcmp (this_grid[i].abbrev, "faa")) ? pow
                                        (10.0, rint (log10 (1.0/stat[0]))) : ((stat[0] > 1.5) ?
                                        MGD77_METERS_PER_FATHOM : 1.0);
                                    fprintf (fpout, "N-%s-E-%.02d: Regression scale %.3f"\
                                    " different from 1. Recommended: [%g]\n",this_grid[i].abbrev,
                                        E77_HDR_SCALE,stat[0],recommended_scale);
                                }
                            }
                            else if (warn[SUMMARY_WARN])
                                fprintf (GMT_stdout, "%s (%s) Slope %.3f is statistically different
                                    from 1\n",\
                                argv[argno],this_grid[i].abbrev,stat[0]);
                        }
                    }

                    /* Check for significant DC shift */
                    range = tcrit * stat[2] * sqrt(S_xx/(npts*stat[3]));  /* Draper 1.4.11 */
                    if (this_grid[i].col != MGD77_DEPTH && (0.0 <= (stat[1]-range) || 0.0 >= (stat
                        [1]+range))) {
                        if (display == "E77") {
                            if (!GMT_is_dnan(adjustDC[this_grid[i].col])) {
                                if (adjustDC[this_grid[i].col] == 0)
                                    fprintf (fpout, "N");
                                else
                                    fprintf (fpout, "Y");
                                fprintf (fpout, "-%s-E-%.02d: Regression offset %.2f different from 0.
                                    Recommended: [%g]\n",\
                                this_grid[i].abbrev,E77_HDR_OFFSET,stat[1],adjustDC[this_grid[i].col]);
                            }
                            else
                                fprintf (fpout, "N-%s-E-%.02d: Regression offset %.2f"\
1000                            " different from 0. Recommended: [%g]\n",this_grid[i].abbrev,
                                    E77_HDR_OFFSET,stat[1],\
                                -0.1*rint (10.0*stat[1]));
                        }
                        else if (warn[GRID_WARN])
                            fprintf (GMT_stdout, "%s (%s) Offset different than 0 (%.2f)\n",argv[argno
                                ],this_grid[i].abbrev,stat[1]);
#ifdef FIX
                        /* Apply DC shift to current field */
                        for (k = 0; k < nvalues; k++) {
                            D[k].number[this_grid[i].col] -= stat[1];
                            diff[i][k] = D[k].number[this_grid[i].col] - G[i][k]; /* Re-compute cruise
                                - grid differences */
                        }
                        if (gmtdefs.verbose)
                            fprintf (stderr, "%s (%s) Warning: Offset corrected by %g for internal
                                along-track analysis\n",\
                            argv[argno],this_grid[i].abbrev, stat[1]);
#endif
                    }
                }
                if (warn[SUMMARY_WARN])
                    fprintf (GMT_stdout, "%s (%s) RLS scale: %.3f offset: %.2f r: %.2f significant:
                        %d decimation: %d\n",
                    argv[argno],this_grid[i].abbrev,stat[0],stat[1],stat[4],(int)stat[5],(int)
                        decimated);

                GMT_free ((void *)decimated_ship);
                GMT_free ((void *)decimated_grid);
            }
            else {
1025            /* Turn off this empty field */
                M.bit_pattern[0] = M.bit_pattern[0] & (0 << this_grid[i].col);

                fprintf (stderr, "%s (%s) Warning: Cruise contains insufficient observations (%d
                    found)\n",\
                argv[argno],this_grid[i].abbrev, npts);
            }
            /* Free up regression array memory */
            GMT_free ((void *)ship_val);
            GMT_free ((void *)grid_val);
        }
    }

    if (display == "E77") { /* Echo out the ranges to be flagged as bad */
        for (i = 0; i < n_bad_sections; i++) {
            fprintf (fpout, "Y-%s-E-%.02d: Record range with invalid data: [%d-%d]\n", BadSection[
                i].abbrev,E77_HDR_FLAGRANGE,BadSection[i].start,BadSection[i].stop);
        }
    }

    /* CHECK CORRECT faa AND mag MODELS */

    /* CHECK SANITY ALONG-TRACK */
    lastLat = D[0].number[MGD77_LATITUDE];
    lastLon = D[0].number[MGD77_LONGITUDE];
    for (curr = 0; curr < nvalues; curr++) {
```

```
1050        thisLat = D[curr].number[MGD77_LATITUDE];
            thisLon = D[curr].number[MGD77_LONGITUDE];

            /* Compute distance (keep units in km) */
            /* Use GMT great circle distance function to calculate arc length between */
            /* current and previous record (in degrees) then convert to km and accumulate */
            if (curr > 0) {
               lastLat = D[curr-1].number[MGD77_LATITUDE];
               lastLon = D[curr-1].number[MGD77_LONGITUDE];
            }
            ds = GMT_great_circle_dist (lastLon, lastLat, thisLon, thisLat) * MGD77_DEG_TO_KM;
            distance[curr] = ds + distance[curr-1];

            /* Create the current time string formatted according to gmtdefaults */
            if (gotTime)
               GMT_ascii_format_one (timeStr, D[curr].time, GMT_io.out_col_type[MGD77_TIME]);
            else
               GMT_ascii_format_one (timeStr, distance[curr], GMT_io.out_col_type[GMT_IS_FLOAT]);

            /* Create the location portion of the verbose data warning string (not for E77) */
            sprintf (placeStr,"%s %s %d",argv[argno],timeStr,curr+1);

            /* Check for time out of range */
            if (D[curr].time > maxTime || D[curr].time < \
            GMT_rdc2dt(GMT_rd_from_gymd(irint(mgd77snifferdefs[MGD77_YEAR].minValue), 1, 1),0.0)) {
1075           E[curr].flags[E77_NAV] |= NAV_TIME_OOR;
               if (warn[TIME_WARN])
                  fprintf (GMT_stdout, "%s - Time out of range\n",placeStr);
            }

            nwords = nout = 0;
            /* Initialize output array for this record */
            out[curr] = (double *)GMT_memory (VNULL, n_out_columns, sizeof (double), GMT_program);
            for (i = 0; i < n_out_columns; i++)
               out[curr][i] = MGD77_NaN;


            /* Store latitude and longitude in the output array */
            if (display == "VALS") {
               out[curr][nout] = D[curr].number[MGD77_LATITUDE];
               nout++;
               out[curr][nout] = D[curr].number[MGD77_LONGITUDE];
               nout++;
            }

            /* SCAN FOR VALUES OUT OF RANGE (POINT-BY-POINT Error Checking) */
            /* Check the 24 numeric fields (start with latitude)*/
               for (i = MGD77_RECTYPE; i < MGD77_N_NUMBER_FIELDS; i++) {

               /* Store cruise values in the output array */
1100           if ((MGD77_this_bit[i] & (MGD77_GEOPHYSICAL_BITS | MGD77_CORRECTION_BITS)) && display
                  == "VALS") {
                  out[curr][nout] = D[curr].number[i];
                  nout++;
               }

               /* Only scan fields present in this cruise */
               if (M.bit_pattern[0] & (1 << i)) {
                  switch (i) {
                     case (MGD77_RECTYPE):
                        if ((int) D[curr].number[i] != 3 && (int) D[curr].number[i] != 5) {
                           E[curr].flags[E77_VALUE] |= (1 << i);
                           if (warn[TYPE_WARN])
                              fprintf (GMT_stdout, "%s - Invalid code %s [%d]\n",\
                              placeStr,mgd77defs[i].abbrev, (int) D[curr].number[i]);
                        }
                        break;
                     case (MGD77_PTC):
                     case (MGD77_BTC):
                        if (((int) D[curr].number[i] != 1 && (int) D[curr].number[i] != 3 &&\
                        (int) D[curr].number[i] != 9)) {
                           E[curr].flags[E77_VALUE] |= (1 << i);
                           if (warn[TYPE_WARN])
                              fprintf (GMT_stdout, "%s - Invalid code %s [%d]\n",\
                              placeStr,mgd77defs[i].abbrev, (int) D[curr].number[i]);
                        }
1125                    break;
                     case (MGD77_MSENS):
                        if ((int)D[curr].number[i] != 1 && (int)D[curr].number[i] != 2 &&\
                        (int) D[curr].number[i] != 9) {
                           E[curr].flags[E77_VALUE] |= (1 << i);
                           if (warn[TYPE_WARN])
                              fprintf (GMT_stdout, "%s - Invalid code %s [%d]\n", \
                              placeStr,mgd77defs[i].abbrev, (int) D[curr].number[i]);
                        }
                        break;
                     case (MGD77_NQC):
                        if ((int) D[curr].number[i] != 5 && (int) D[curr].number[i] != 6 &&\
                        (int) D[curr].number[i] != 9) {
                           E[curr].flags[E77_VALUE] |= (1 << i);
                           if (warn[TYPE_WARN])
                              fprintf (GMT_stdout, "%s - Invalid code %s [%d]\n",placeStr,\
```

```
                        mgd77defs[i].abbrev, (int) D[curr].number[i]);
                }
                break;
            case (MGD77_BCC):
                bccCode = irint (D[curr].number[i]);
                if (M.bit_pattern[0] & MGD77_TWT_BIT || M.bit_pattern[0] & MGD77_DEPTH_BIT) {
                    if (bccCode < 1 || bccCode > 55) {
                        switch ((int) bccCode) {
                            case 59: /* Matthews', no zone */
1150                        case 60: /* S. Kuwahara Formula */
                            case 61: /* Wilson Formula */
                            case 62: /* Del Grosso Formula */
                            case 63: /* Carter's Tables */
                            case 88: /* Other */
                            case 98: /* Unknown */
                            case 99: /* Unspecified */
                                break;
                            default:
                                E[curr].flags[E77_VALUE] |= (1 << i);
                                if (warn[TYPE_WARN])
                                    fprintf (GMT_stdout, "%s - Invalid code %s [%d]\n",placeStr,\
                                    mgd77defs[i].abbrev, (int) bccCode);
                                break;
                        }
                    }
                }
                break;
            case (MGD77_DAY): /* Separate case since # of days in a month varies */
                if (!GMT_is_dnan (D[curr].number[i])) {
                    if ((E[curr].flags[E77_VALUE] & (1 << MGD77_YEAR)) || (E[curr].flags[
                        E77_VALUE] & (1 << MGD77_MONTH))
                        last_day = mgd77snifferdefs[i].maxValue;  /* Year or month has error so
                                we use 31 as last day in this month */
                    else
                        last_day = GMT_gmonth_length (irint(D[curr].number[MGD77_YEAR]), irint(
                            D[curr].number[MGD77_MONTH]));        /* Number of day in the
                            specified month */

1175                if (GMT_is_dnan (D[curr].number[i]) && (D[curr].number[i] <
                        mgd77snifferdefs[i].minValue || D[curr].number[i] > last_day)) {
                    E[curr].flags[E77_VALUE] |= (1 << i);
                    if (warn[VALUE_WARN])
                        fprintf (GMT_stdout, "%s - %s out of range [%f]\n", placeStr,
                            mgd77defs[i].abbrev, D[curr].number[i]);
                }
                }
                break;
            case (MGD77_LONGITUDE): /* Handle longitudes in 180-360 range */
                if (!GMT_is_dnan (D[curr].number[i]) && D[curr].number[i] > mgd77snifferdefs[
                    i].maxValue && D[curr].number[i] <= 360.0) {
                    if (warn[VALUE_WARN])
                        fprintf (GMT_stdout, "%s - %s adjusted %f to +/- 180\n", placeStr,
                            mgd77defs[i].abbrev, D[curr].number[i]);
                    D[curr].number[i] -= 360.0;
                }
            default:
                /* Verify that measurements are within range */
                if (!GMT_is_dnan (D[curr].number[i]) && (D[curr].number[i] < mgd77snifferdefs
                    [i].minValue ||\
                D[curr].number[i] > mgd77snifferdefs[i].maxValue)) {
                    E[curr].flags[E77_VALUE] |= (1 << i);
                    if (warn[VALUE_WARN])
                        fprintf (GMT_stdout, "%s - %s out of range [%f]\n", placeStr, mgd77defs
                            [i].abbrev, D[curr].number[i]);
                }
                if ((i == MGD77_LATITUDE || i == MGD77_LONGITUDE) && GMT_is_dnan(D[curr].
                    number[i])) {
                    E[curr].flags[E77_NAV] |= (1 << i);
                    if (warn[VALUE_WARN])
                        fprintf (GMT_stdout, "%s - %s cannot be nine-filled\n", placeStr,
                            mgd77defs[i].abbrev);
1200            }
                break;
        }
    }
}

/* Along-Track Excessive Slope Error Checking */
if (curr > 0) {

    /* Check dt */
    if (!GMT_is_dnan(D[curr].time) && !GMT_is_dnan(D[curr-1].time))
        dt = D[curr].time - D[curr-1].time;
    else {    /* Time not specified */
        dt = MGD77_NaN;
        if (noTimeStart == -1)
            noTimeStart = curr;
        noTimeCount++;
    }
    if (!GMT_is_dnan(dt) && dt <= 0) { /* Non-increasing time */
        if (dt == 0) {
            if (warn[TIME_WARN])
```

```
                        fprintf (GMT_stdout, "%s - Time not monotonically increasing (%f sec.)\n",
                              placeStr, dt);
                     dt = MGD77_NaN;
                  } else {
                     E[curr].flags[E77_NAV] |= NAV_TIME_DECR;
1225                 if (warn[TIME_WARN])
                        fprintf (GMT_stdout, "%s - Time decreasing (%f sec.)\n",placeStr, dt);
                  }
                  if (timeErrorStart == -1)
                     timeErrorStart = curr;
                  timeErrorCount++;
               }

               /* Calculate speed */
               if (dt != 0)
                  speed = (ds*1000*distance_factor)/(dt*time_factor);
               else
                  speed = MGD77_NaN;

#ifdef HISTOGRAM_MODE
               /* Use this to print excessive slopes to stderr for
               tracking winning cruises when running histogram scripts */
               if (!GMT_is_dnan(speed) && speed > max_speed) {
                  E[curr].flags[E77_NAV] |= NAV_HISPD;
                  fprintf (stderr, "%s - Excessive speed %f %s\n",placeStr, speed, speed_units);
               }
#endif

               /* Check that ship speed is reasonable */
               if (!GMT_is_dnan(speed)) {
1250              if (speed > max_speed) {
                     E[curr].flags[E77_NAV] |= NAV_HISPD;
                     if (warn[SPEED_WARN])
                        fprintf (GMT_stdout, "%s - Excessive speed %f %s\n",placeStr, speed,
                              speed_units);
                  }
               }

               /* Store speed in the output array */
               nout = 0;
               if (display == "SLOPES") {
                  if (!GMT_is_dnan(speed))
                     out[curr][nout] = speed;
                  else
                     out[curr][nout] = MGD77_NaN;
                  nout++;
               }

               /* Check slope values for non-time geophysical measurements */
               for (i = MGD77_LATITUDE; i < MGD77_N_NUMBER_FIELDS; i++) {

                  /* Only scan floating point fields present in this cruise */
                  if (M.bit_pattern[0] & (1 << i) & MGD77_FLOAT_BITS) {

                     /* Compute the difference between current and previous value */
                     if (GMT_is_dnan(D[curr].number[i])) {
1275                    gradient = MGD77_NaN;
                        dvalue = MGD77_NaN;
                     }
                     else {
                        /* Search backward to find a non-empty record for the same field. */
                        /* Hope it doesn't have to search too far */
                        for (j = 1; GMT_is_dnan(D[curr-j].number[i]) && curr-j > 0; j++)
                           if (j > MGD77_MAX_SEARCH) break;
                        dvalue = D[curr].number[i]-D[curr-j].number[i];
                        lastLat = D[curr-j].number[MGD77_LATITUDE];
                        lastLon = D[curr-j].number[MGD77_LONGITUDE];
                        /* Calculate ds & dt between current and last valid record */
                        /* Note: ds may be different for each field */
                        ds = GMT_great_circle_dist (lastLon, lastLat, thisLon, thisLat) *
                              MGD77_DEG_TO_KM;
                        dt = D[curr].time - D[curr-j].time;

                        /* Set to nan if a gap is detected (unless gap skipping is turned off) */
                        if (ds > maxGap && maxGap != 0)
                           dvalue = MGD77_NaN;

                        /* Compute gradient */
                        if (derivative == "SPACE") {
                           if (ds >= MGD77_NAV_PRECISION_KM)
                              gradient = dvalue / ds;
                           else
1300                          gradient = MGD77_NaN;
                        }
                        else if (derivative == "DIFF")
                           gradient = dvalue;
                        else { /* Gradient with respect to time */
                           if (dt > 0)
                              gradient = dvalue / dt;
                           else
                              gradient = MGD77_NaN;
                        }
```

```
                        /* First Derivative Sanity Check */
                        if (fabs(gradient) > maxSlope[i]) {
                           E[curr].flags[E77_SLOPE] |= (1 << i);
                           if (warn[SLOPE_WARN])
                              fprintf (GMT_stdout, "%s - excessive %s gradient %f\n", placeStr,
                                    mgd77defs[i].abbrev,\
                              gradient);
                        }


#ifdef HISTOGRAM_MODE
                        /* Use this to print excessive slopes to stderr for tracking winning
                           cruises when running histogram scripts */
                        if (fabs(gradient) > maxSlope[i]) {
                           E[curr].flags[E77_SLOPE] |= (1 << i);
1325                       fprintf (stderr, "%s - excessive %s gradient %f\n", placeStr, mgd77defs[i
                                ].abbrev, gradient);
                        }
#endif
                     }
                  }
                  else
                     gradient = dvalue = ds = MGD77_NaN;

                  if ((1 << i) & (MGD77_GEOPHYSICAL_BITS | MGD77_CORRECTION_BITS)) {
                     if (display == "SLOPES")
                        /* Store slopes in output array */
                        out[curr][nout++] = gradient;
                     if (display == "DFDS") {
                        out[curr][nout++] = fabs(dvalue);
                        if (GMT_is_dnan(dvalue))
                           out[curr][nout++] = MGD77_NaN;
                        else
                           out[curr][nout++] = ds;
                     }
                  }
               }
            }

            /* CRUISE - GRID COMPARISON */
            if (n_grids > 0) {
1350           for (i = 0; i < n_grids; i++) {

                  if (this_grid[i].g_pts < 2)
                     continue;

                  /* Fill output array with cruise/grid differences */
                  if (display == "DIFFS") {
                     out[curr][0] = D[curr].number[MGD77_LATITUDE];
                     out[curr][1] = D[curr].number[MGD77_LONGITUDE];
                     out[curr][2] = distance[curr]*distance_factor;
                     out[curr][3+i*3] = D[curr].number[this_grid[i].col];
                     out[curr][4+i*3] = G[i][curr];
                     out[curr][5+i*3] = D[curr].number[this_grid[i].col]-G[i][curr];
                  }

                  /* Check if the cruise went over land */
                  if (!strcmp(this_grid[i].abbrev,"depth") && G[i][curr]*this_grid[i].sign > 0) {
                     E[curr].flags[E77_NAV] = NAV_ON_LAND;
                     if (!landcruise)
                        overLandStart = curr;
                     landcruise = TRUE;
                     overLandCount++;
                  }

                  if (M.bit_pattern[0] & (1 << this_grid[i].col)) {
1375                 /* Track min/max absolute difference between grid and cruise */
                     if (fabs(diff[i][curr]) > fabs(MaxDiff[i])) {
                        MaxDiff[i] = diff[i][curr];
                        iMaxDiff[i] = curr;
                     }

                     /* Compare cruise and grid data to find offsets */
                     if (curr > 0) {

                        /* Areas are estimated for each offset by summing discrete rectangles while
                              the offset sign
                           stays the same */
                        /* Search backward to find a non-empty record for the same field. */
                        /* Hope it doesn't have to search too far */
                        for (j = 1; GMT_is_dnan(D[curr-j].number[this_grid[i].col]) && curr-j > 0; j
                              ++)
                           if (j > MGD77_MAX_SEARCH) break;
                        lastLat = D[curr-j].number[MGD77_LATITUDE];
                        lastLon = D[curr-j].number[MGD77_LONGITUDE];
                        /* Calculate ds & dt between current and last valid record */
                        /* Note: ds may be different for each field */
                        ds = GMT_great_circle_dist (lastLon, lastLat, thisLon, thisLat) *
                              MGD77_DEG_TO_KM;
                        dt = D[curr].time - D[curr-j].time;
```

```
                        /* Calculate area of this offset */
                        if (!GMT_is_dnan(diff[i][curr])) {
                            thisArea = 0.5 * (diff[i][curr] + diff[i][curr-j]) * ds;
1400                        prevOffsetSign[i] = offsetSign[i];
                            offsetSign[i] = thisArea > 0;
                        }
                        else
                            thisArea = MGD77_NaN;

                        /* Accumulate area of total offset (Attempt to skip big gaps in data) */
                        /* Think about different types of gaps to avoid */
                        /* 1. Lose GPS - time stays consistent but ds gets huge - max ds test */
                        /* 2. Ship stops logging and moves to another region then logs again - max dt
                              test */
                        /* 3. No recorded time - use max ds test */
                        if (!GMT_is_dnan(thisArea) && offsetSign[i] == prevOffsetSign[i] && \
                            (ds < maxGap || maxGap == 0)) {
                            offsetArea[i] += thisArea;
                            offsetLength[i] += ds;
                        }

                        /* Analyze offset after it ends (sign switch) */
                        if (offsetSign[i] != prevOffsetSign[i] ||  curr == nvalues - 1) {
                            /* Flag the offset if significant */
                            if (fabs(offsetArea[i]) > mgd77snifferdefs[this_grid[i].col].maxArea) {
                                if (warn[GRID_WARN])
                                    fprintf (GMT_stdout, "%s - excessive offset from grid: Area/Length/
                                          Height"\
                                    " %f\t%f\t%f\t%d\n", placeStr, fabs(offsetArea[i]),offsetLength[i],\
                                    fabs(offsetArea[i]/offsetLength[i]),curr);
1425                            /* Go back and flag all records in this offset */
                                for (k = offsetStart[i]; k <= curr; k++)
                                    E[k].flags[E77_GRID] |= (1 << this_grid[i].col);
                            }
                            offsetArea[i] = 0;
                            offsetLength[i] = 0;
                            offsetStart[i] = curr;
                        }
                    }
                }
            }

            /* Check for lower precision values and duplicate records */
            for (i = MGD77_LATITUDE; i < MGD77_N_NUMBER_FIELDS; i++) {

                /* Only check floating point fields present in this cruise */
                if (M.bit_pattern[0] & (1 << i) & MGD77_FLOAT_BITS) {

                    /* Compute the difference between current and previous value */
                    if (!GMT_is_dnan(D[curr].number[i])) {

                        /* Check for lower precision */
                        for (j = 1; GMT_is_dnan(D[curr-j].number[i]) && curr-j > 0; j++)
                            if (j > MGD77_MAX_SEARCH) break;
1450                    dvalue = D[curr].number[i]-D[curr-j].number[i];

                        if (!GMT_is_dnan(dvalue) && dvalue != 0) {
                            if (fmod(dvalue,1.0) != 0.0 && (lowPrecision & (1 << i)))
                                lowPrecision -= (1 << i);
                            if (fmod(dvalue,5.0) != 0.0 && (lowPrecision5 & (1 << i)))
                                lowPrecision5 -= (1 << i);
                        }

                        /* Check for duplicate records */
                        if (dvalue == 0)
                            duplicates[i]++;
                        else {
                            if (duplicates[i] > MGD77_MAX_DUPLICATES) {
                                if (warn[VALUE_WARN])
                                    fprintf (GMT_stdout, "%s - %d duplicate %s records\n",placeStr,\
                                    duplicates[i], mgd77defs[i].abbrev);
                                for (k = curr-1; k >= curr-duplicates[i]; k--)
                                    E[k].flags[E77_VALUE] |= (1 << i);
                            }
                            duplicates[i] = 0;
                        }
                    }
                }
            }
1475 #ifndef FIX
            /* Dump data row-by-row if requested */
            if (display != "E77" && display != NULL) {
                if (display == "MGD77")
                    MGD77_Write_Data_Record_m77 (&Out, &D[curr]);
                else {
                    if (curr > 0 || display == "VALS") {
                        if (GMT_io.binary[GMT_OUT])
                            /* Use GMT output machinery which can handle binary output */
                            GMT_output (GMT_stdout, n_out_columns, out);
                        else {
                            for (i = 0; i < (n_out_columns); i++) {
```

```
                        GMT_ascii_output_one (GMT_stdout, out[curr][i], i);
                        if ((i+1) < n_out_columns) fprintf (GMT_stdout, "%s", gmtdefs.
                            field_delimiter);
                    }
                    fprintf (GMT_stdout, "\n");
                }
            }
        }
    }
#endif
    }

    if (warn[TYPE_WARN]) {
        if (bccCode != 63 && bccCode != 88 && (M.bit_pattern[0] & MGD77_TWT_BIT))
            fprintf (GMT_stdout, "%s - new bathymetry correction table available\n",placeStr);
        else if (bccCode > 0 && bccCode < 56 && !(M.bit_pattern[0] & MGD77_TWT_BIT))
            fprintf (GMT_stdout, "%s - twt may be extracted from depth for Carter correction\n",
                placeStr);
    }

#ifdef FIX
    /* Turn off fields if errors found */
    for (rec = 0; rec < curr; rec++) {
        deleteRecord = FALSE;
        for (type = 0; type < N_ERROR_CLASSES; type++) {
            if (E[rec].flags[type]) { /*Error in this category */
                thisLon = D[rec].number[MGD77_LONGITUDE];
                thisLat = D[rec].number[MGD77_LATITUDE];
                switch (type) {
                    case E77_NAV:
                        /* 9-fill records with nav errors */
                        for (i=MGD77_PTC; i<MGD77_N_NUMBER_FIELDS; i++) D[rec].number[i] = MGD77_NaN;
                        deleteRecord = TRUE;
                        break;
                    default:
                        for (field = MGD77_PTC; field < n_types[type]; field++) {
                            if (E[rec].flags[type] & (1 << field))
                                D[rec].number[field] = MGD77_NaN;
                        }
                        break;
                }
                D[rec].number[MGD77_LONGITUDE] = thisLon;
                D[rec].number[MGD77_LATITUDE] = thisLat;
            }
        }

        /* Dump "fixed" data row-by-row if requested */
        if (display != "E77" && display != NULL && !deleteRecord) {
            if (display == "MGD77")
                MGD77_Write_Data_Record_m77 (&Out, &D[rec]);
            else {
                if (rec > 0 || display == "VALS") {
                    if (GMT_io.binary[GMT_OUT])
                        /* Use GMT output machinery which can handle binary output */
                        GMT_output (GMT_stdout, n_out_columns, out);
                    else {
                        for (i = 0; i < (n_out_columns); i++) {
                            GMT_ascii_output_one (GMT_stdout, out[rec][i], i);
                            if ((i+1) < n_out_columns) fprintf (GMT_stdout, "%s", gmtdefs.
                                field_delimiter);
                        }
                        fprintf (GMT_stdout, "\n");
                    }
                }
            }
        }
    }
    }
#endif

    /* OUTPUT E77 ERROR FORMAT */
    if (display == "E77") {
        /* E77 HEADER MESSAGES */
        if ((bccCode != 63 && bccCode != 88 && (M.bit_pattern[0] & MGD77_TWT_BIT)) OR_TRUE)
            fprintf (fpout, "I-twt-W-%.2d: More recent bathymetry correction table available\n",
                E77_HDR_BCC);
        if ((bccCode > 0 && bccCode < 56 && !(M.bit_pattern[0] & MGD77_TWT_BIT)) OR_TRUE)
            fprintf (fpout, "I-twt-W-%.2d: twt may be extracted from depth for Carter correction\n
                ",E77_HDR_BCC);

        /* Output data precision warnings */
        if (lowPrecision || lowPrecision5 OR_TRUE) {
            for (k = MGD77_LATITUDE; k < MGD77_N_NUMBER_FIELDS; k++) {
                if (lowPrecision5 & (1 << k) OR_TRUE)
                    fprintf(fpout, "I-%s-W-%.02d: Integer multiple of 5 precision in %s\n",mgd77defs
                        [k].abbrev,\
                    E77_HDR_PRECISION,mgd77defs[k].abbrev);
                if (lowPrecision & (1 << k) OR_TRUE)
                    fprintf(fpout, "I-%s-W-%.02d: Integer precision in %s\n",mgd77defs[k].abbrev,
                        E77_HDR_PRECISION,
                    mgd77defs[k].abbrev);
            }
        }
```

```
            /* E77 ERROR RECORDS */
            fprintf (fpout,"# Errata: Data\n");
1575        for (rec = 0; rec < curr; rec++) {
               if (gotTime)
                  GMT_ascii_format_one (timeStr, D[rec].time, GMT_io.out_col_type[MGD77_TIME]);
               else
                  GMT_ascii_format_one (timeStr, distance[rec], GMT_io.out_col_type[GMT_IS_FLOAT]);
               sprintf (placeStr, "%s%s%d%s",timeStr,gmtdefs.field_delimiter,rec+1,gmtdefs.
                     field_delimiter);
               errorStr[0]='\0';
               for (type = 0; type < N_ERROR_CLASSES; type++) {
                  if (E[rec].flags[type] OR_TRUE) { /*Error in this category */
                     for (field = 0; field < (int)n_types[type]; field++) {
                        if (E[rec].flags[type] & (1 << field) OR_TRUE)
                           sprintf (errorStr, "%s%c",errorStr,'A'+field);
                     }
                  }
                  else
                     sprintf (errorStr, "%s0",errorStr);
                  if (type < N_ERROR_CLASSES-1)
                     sprintf (errorStr, "%s-",errorStr);
               }
               if (!strcmp(errorStr,"0-0-0-0")) continue;
               fprintf (fpout, "%s%s%s",placeStr,errorStr,gmtdefs.field_delimiter);
               prevType = FALSE;
               for (type = 0; type < N_ERROR_CLASSES; type++) {
                  if (E[rec].flags[type] OR_TRUE) { /*Error in this category */
                     fprintf (fpout, " ");
1600                 if (prevType && (E[rec].flags[type] OR_TRUE))
                        fprintf(fpout, "- ");
                     if (type == E77_NAV)
                        fprintf (fpout, "NAV: ");
                     if (type == E77_VALUE)
                        fprintf (fpout, "VAL: ");
                     if (type == E77_SLOPE)
                        fprintf (fpout, "GRAD: ");
                     if (type == E77_GRID)
                        fprintf (fpout, "GRID: ");
                     for (field=0; field < (int)n_types[type]; field++) {
                        if (E[rec].flags[type] & (1 << field) OR_TRUE) {
                           if (prevFlag)
                              fprintf (fpout, ", ");
                           switch (type) {
                              case E77_NAV:
                                 switch (1 << field) {
                                    case NAV_TIME_OOR:
                                       fprintf (fpout, "time out of range");
                                       break;
                                    case NAV_TIME_DECR:
                                       fprintf (fpout, "decreasing time");
                                       break;
                                    case NAV_HISPD:
                                       fprintf (fpout, "excessive speed");
1625                                   break;
                                    case NAV_ON_LAND:
                                       fprintf (fpout, "on land");
                                       break;
                                    case NAV_LAT_UNDEF:
                                       fprintf (fpout, "latitude undefined");
                                       break;
                                    case NAV_LON_UNDEF:
                                       fprintf (fpout, "longitude undefined");
                                       break;
                                    default:
                                       fprintf (fpout, "undefined nav error!");
                                       break;
                                 }
                                 break;
                              default:
                                 fprintf (fpout, "%s",mgd77defs[field].abbrev);
                                 break;
                           }
                           prevFlag = TRUE;
                        }
                     }
                     prevFlag = FALSE;
                     switch (type) {
                        case E77_VALUE:
1650                       fprintf (fpout, " invalid");
                           break;
                        case E77_SLOPE:
                           fprintf (fpout, " excessive");
                           break;
                        case E77_GRID:
                           fprintf (fpout, " offset");
                           break;
                        default:
                           break;
                     }
                     prevType = TRUE;
                  }
```

```
                }
                fprintf (fpout, "\n");
            }
        }

        /* Print Error Summary */
        if (warn[SUMMARY_WARN]) {
            if (noTimeCount == curr - 1)          /* no valid time in data */
                fprintf (GMT_stdout, "%s (time) Cruise contains no time record\n", argv[argno]);

            if (noTimeCount >0 && noTimeCount < curr) /* print the first and number of nine-filled
                time records */
                fprintf (GMT_stdout, "%s (time) %d records contain no time starting at record #%d\n",
                    argv[argno],\
1675            noTimeCount, noTimeStart);

            if (timeErrorCount > 0 && timeErrorCount < curr)/* print the first and number of time
                errors */
                fprintf (GMT_stdout, "%s (time) %d records had time errors starting at record #%d\n",
                    argv[argno],\
                timeErrorCount, timeErrorStart);

            if (distanceErrorCount > 0) /* print the first and number of distance errors */
                fprintf (GMT_stdout, "%s (dist) %d records had distance errors starting at record #%d\
                    n",argv[argno],\
                distanceErrorCount,distanceErrorStart);

            for (i = MGD77_LATITUDE; i < MGD77_N_NUMBER_FIELDS; i++) {
                if ((lowPrecision & (1 << i)) && !(lowPrecision5 & (1 << i)))
                    fprintf (GMT_stdout, "%s (%s) Data Precision Warning: only integer values found\n",
                        argv[argno],mgd77defs[i].abbrev);
                if (lowPrecision5 & (1 << i)) /* low precision data */
                    fprintf (GMT_stdout, "%s (%s) Data Precision Warning: only integer multiples of 5
                        found\n",argv[argno],\
                    mgd77defs[i].abbrev);
            }

            if (n_grids) {
                for (i = 0; i < n_grids; i++)
                    fprintf (GMT_stdout, "%s (%s) Max ship-grid difference [%.1f] at record %d\n",argv[
                        argno],\
                    mgd77defs[this_grid[i].col].abbrev,MaxDiff[i],iMaxDiff[i]);
            }

            if (landcruise)   /* vessel went over land (GPS error or other gross navigation) */
1700            fprintf (GMT_stdout, "%s (nav) Navigation Warning: %d records went over land starting
                    at record %d\n",\
                argv[argno], overLandCount, overLandStart);
        }
        /* Clean-up after finishing this cruise */
        MGD77_Close_File (&M);
        GMT_free ((void *)D);
        GMT_free ((void *)distance);
        GMT_free ((void *)H.mgd77);
        if (n_grids > 0) {
            for (i = 0; i < n_grids; i++) {
                GMT_free ((void *)G[i]);
                GMT_free ((void *)diff[i]);
            }
            GMT_free ((void *)G);
            GMT_free ((void *)diff);
        }
        if (display == "E77")
            fclose (fpout);
    }
    /* De-allocate grid memory */
    if (n_grids > 0) {
        GMT_free ((void *)MaxDiff);
        GMT_free ((void *)iMaxDiff);
        for (i = 0; i < n_grids; i++)
            GMT_free ((void *)f[i]);
1725    }
    exit (EXIT_SUCCESS);
}

void regress_rls (double *x, double *y, int nvalues, double *stat, int col, double S_xx)
{
    int i, n;
    double y_hat, threshold, s_0, res, *xx, *yy;

    regress_lms (x, y, nvalues, stat, col);
    /* Get LMS scale and use 2.5 of it to detect regression outliers */
    s_0 = 1.4826 * (1.0 + 5.0 / nvalues) * sqrt (stat[2]);
    threshold = 2.5 * s_0;

    xx = (double *)GMT_memory (VNULL, nvalues, sizeof (double), "mgd77sniffer");
    yy = (double *)GMT_memory (VNULL, nvalues, sizeof (double), "mgd77sniffer");
    for (i = n = 0; i < nvalues; i++) {
        y_hat = stat[0] * x[i] + stat[1];
        res = y[i] - y_hat;
        if (fabs (res) > threshold) continue;  /* Skip outliers */
        xx[n] = x[i];
```

```
            yy[n] = y[i];
            n++;
        }
        /* Now do LS regression of the 'good' points */
1750    regress_ls (xx, yy, n, stat, col, S_xx);
        /*stat[4] = GMT_corrcoeff (xx, yy, n, 0);*/
        if (n > 2) {    /* Determine if correlation is significant at 95% */
            double t, tcrit;
            t = stat[4] * sqrt (n - 2.0) / sqrt (1.0 - stat[4] * stat[4]);
            tcrit = GMT_tcrit (0.95, (double)n - 2.0);
            stat[5] = (double)(t > tcrit);    /* 1.0 if significant, 0.0 otherwise */
        }
        else
            stat[5] = GMT_d_NaN;

        GMT_free ((void *)xx);
        GMT_free ((void *)yy);
    }

    void regress_ls (double *x, double *y, int n, double *stat, int col, double S_xx)
    {
        int i;
        double S, sum_x, sum_y, sum_x2, sum_y2, sum_xy;
        double mean_x, mean_y, S_xy, S_yy, y_discrepancy = 0.0;

        sum_x = sum_y = sum_x2 = sum_y2 = sum_xy = 0.0;
        mean_x = mean_y = S_xx = S_xy = S_yy = 0.0;
        S = (double)n;

1775    for (i = 0; i < n; i++) {
            sum_x += x[i];
            sum_y += y[i];
            sum_x2 += x[i] * x[i];
            sum_y2 += y[i] * y[i];
            sum_xy += x[i] * y[i];
        }

        mean_x = sum_x / S;
        mean_y = sum_y / S;

        S_xy = sum_xy - S * mean_x * mean_y;
        S_xx = sum_x2 - S * mean_x * mean_x;
        S_yy = sum_y2 - S * mean_y * mean_y;
        if (col != MGD77_DEPTH) { /* Use LMS m & b for depth (since offset forced to 0) */
            stat[0] = S_xy / S_xx;                  /* Slope */
            stat[1] = mean_y - stat[0] * mean_x;    /* Intercept */
        }

        for (i = 0; i < n; i++)
            y_discrepancy += pow (y[i] - stat[0] * x[i] - stat[1], 2);
        stat[2] = sqrt (y_discrepancy / (S-1));          /* Standard deviation */
        stat[3] = S_xx;                                  /* Sum of squares */
        stat[4] = sqrt(S_xy * S_xy / (S_xx * S_yy)); /* Correlation (r) */
    }
1800
    void regress_lms (double *x, double *y, int nvalues, double *stat, int col)
    {

        double d_angle, limit, a, old_error, d_error, angle_0, angle_1;
        int n_angle;
        void regresslms_sub (double *x, double *y, double angle0, double angle1, int nvalues, int
              n_angle, double *stat, int col);

        d_angle = 1.0;
        limit = 0.1;
        n_angle = irint ((180.0 - 2 * d_angle) / d_angle) + 1;
        regresslms_sub (x, y, -90.0 + d_angle, 90.0 - d_angle, nvalues, n_angle, stat, col);
        old_error = stat[2];
        d_error = stat[2];

        while (d_error > limit) {
            d_angle = 0.1 * d_angle;
            a = atan (stat[0]) * 180 / M_PI;
            angle_0 = floor (a / d_angle) * d_angle - d_angle;
            angle_1 = angle_0 + 2.0 * d_angle;
            regresslms_sub (x, y, angle_0, angle_1, nvalues, 21, stat, col);
            d_error = fabs (stat[2] - old_error);
            old_error = stat[2];
        }
        if (gmtdefs.verbose)
1825        fprintf (stderr, "\n");
    }

    void regresslms_sub (double *x, double *y, double angle0, double angle1, int nvalues, int n_angle,
          double *stat, int col)
    {
        double da, *slp, *icept, *z, *sq_misfit, *angle, *e, emin = DBL_MAX;
        int i, j = 0;

        slp = (double *) GMT_memory (VNULL, (size_t) n_angle, sizeof (double), "mgd77sniffer");
        icept = (double *) GMT_memory (VNULL, (size_t) n_angle, sizeof (double), "mgd77sniffer");
        angle = (double *) GMT_memory (VNULL, (size_t) n_angle, sizeof (double), "mgd77sniffer");
```

```
        e = (double *) GMT_memory (VNULL, (size_t) n_angle, sizeof (double), "mgd77sniffer");
        z = (double *) GMT_memory (VNULL, (size_t) nvalues, sizeof (double), "mgd77sniffer");
        sq_misfit = (double *) GMT_memory (VNULL, (size_t) nvalues, sizeof (double), "mgd77sniffer");

        for (i=0; i < 4; i++)
           stat[i] = 0;
        for (i=0; i < n_angle; i++)
           slp[i] = icept[i] = angle[i] = e[i] = 0;
        da = (angle1 - angle0) / (n_angle - 1);

        for (i = 0; i < n_angle; i++) {
           angle[i] = angle0 + i * da;
           if (gmtdefs.verbose)
              fprintf (stderr, "%s: Trying angle...%+03.0f\r", GMT_program, angle[i]);
1850       slp[i] = tan (angle[i] * M_PI / 180.0);
           for (j = 0; j < nvalues; j++)
              z[j] = y[j] - slp[i] * x[j];
           if (col == MGD77_DEPTH)
              icept[i] = 0.0;
           else
              icept[i] = lms (z, nvalues);
           for (j = 0; j < nvalues; j++)
              sq_misfit[j] = pow(z[j]-icept[i], 2);
           e[i] = median (sq_misfit, nvalues);
        }
        for (i = 0; i < n_angle; i++) {
           if (e[i] < emin || i == 0) {
              emin = e[i];
              j = i;
           }
        }
        stat[0] = slp[j];
        stat[1] = icept[j];
        stat[2] = e[j];

        GMT_free ((void *)slp);
        GMT_free ((void *)icept);
        GMT_free ((void *)angle);
        GMT_free ((void *)e);
1875    GMT_free ((void *)z);
        GMT_free ((void *)sq_misfit);
     }

     double lms (double *x, int n)
     {
        double mode;
        int GMT_n_multiples = 0;

        GMT_mode (x, n, n/2, 1, 0, &GMT_n_multiples, &mode);
        return mode;
     }

     double median (double *x, int n)
     {
        double *sorted, med;
        int i;

        sorted = (double *) GMT_memory (VNULL, (size_t) n, sizeof (double), "mgd77sniffer");
        for (i=0; i < n; i++)
           sorted[i] = x[i];
        qsort ((void *) sorted, n, sizeof(double), GMT_comp_double_asc);
        med = (i%2) ? sorted[i/2] : 0.5*(sorted[(i-1)/2]+sorted[i/2]);
        GMT_free ((void *)sorted);
        return med;
1900 }

     /* Read Grid Header (from Smith & Wessel grdtrack.c) */
     void read_grid (struct MGD77_GRID_INFO *info, float **grid, double w, double e, double s, double n
           , BOOLEAN bilinear, double threshold) {

        if (strlen(info->fname) == 0) return;  /* No name */

        GMT_pad[0] = GMT_pad[1] = GMT_pad[2] = GMT_pad[3] = 2;  /* GMT_pad is declared automatically in
              gmt.h */

        if (info->format == 0) {   /* GMT geographic grid with header */
           if (GMT_read_grd_info (info->fname, &info->grdhdr)) {
                 fprintf (stderr, "%s: Error opening file %s\n", GMT_program, info->fname);
                 exit (EXIT_FAILURE);
           }

           /* Get grid dimensions */
           info->one_or_zero = (info->grdhdr.node_offset) ? 0 : 1;
           info->nx = irint ( (info->grdhdr.x_max - info->grdhdr.x_min) / info->grdhdr.x_inc) + info->
                 one_or_zero;
           info->ny = irint ( (info->grdhdr.y_max - info->grdhdr.y_min) / info->grdhdr.y_inc) + info->
                 one_or_zero;

           /* Allocate grid memory */
           *grid = (float *) GMT_memory (VNULL, (size_t)((info->nx + 4) * (info->ny + 4)), sizeof (
                 float), GMT_program);
```

```
            /* Read into memory with 2 rows/cols as padding */
            if (GMT_read_grd (info->fname, &info->grdhdr, *grid, w, e, s, n, GMT_pad, FALSE)) {
1925            fprintf (stderr, "%s: Error reading file %s\n", GMT_program, info->fname);
                exit (EXIT_FAILURE);
            }
        }
        else {   /* Read a Mercator grid Sandwell/Smith style */
            GMT_read_img (info->fname, &info->grdhdr, grid, w, e, s, n, info->scale, info->mode, info->
                max_lat, TRUE);
            if (fabs (info->grdhdr.x_max - info->grdhdr.x_min - 360.0) < GMT_CONV_LIMIT)
                GMT_boundcond_parse (&info->edgeinfo, "g");
        }

        info->mx = info->grdhdr.nx + 4;
        info->interpolate = (threshold > 0.0);
        if (fabs (info->grdhdr.x_max - info->grdhdr.x_min - 360.0) < GMT_CONV_LIMIT)
            GMT_boundcond_parse (&info->edgeinfo, "g");

        GMT_boundcond_param_prep (&info->grdhdr, &info->edgeinfo);

        /* Initialize bcr structure with 2 row/col boundaries:  */

        GMT_bcr_init (&info->grdhdr, GMT_pad, bilinear, threshold, &info->bcr);

        /* Set boundary conditions  */
        GMT_boundcond_set (&info->grdhdr, &info->edgeinfo, GMT_pad, *grid);
    }

    /* Sample Grid at Cruise Locations (from Smith & Wessel grdtrack.c) */
    int sample_grid (struct MGD77_GRID_INFO *info, struct MGD77_DATA_RECORD *D, double **g, float *
        grid, int n_grid, int n) {
1950
        int rec, pts = 0, ii, jj;
        double MGD77_NaN, x, y;
        GMT_make_dnan (MGD77_NaN);

        /* Get grid values at cruise locations */
        for (rec = 0; rec < n; rec++) {

            if (info->format == 1)  /* Mercator IMG grid - get Mercator coordinates x,y */
                GMT_geo_to_xy (D[rec].number[MGD77_LONGITUDE], D[rec].number[MGD77_LATITUDE], &x, &y);
            else {      /* Regular geographic grd, just copy lon,lat to x,y */
                x = D[rec].number[MGD77_LONGITUDE];
                y = D[rec].number[MGD77_LATITUDE];
                /* Adjust cruise longitude if necessary; We know cruise is between +/-180 */
                if (info->grdhdr.x_min >= 0.0 && D[rec].number[MGD77_LONGITUDE] < 0.0)
                    GMT_lon_range_adjust(0, &x);  /* Adjust to 0-360 range */
            }
            g[n_grid][rec] = MGD77_NaN;   /* Default value if we are outside the grid domain */
            if (y < info->grdhdr.y_min || y > info->grdhdr.y_max) continue;   /* Outside latitude range
                */

            /* If point is outside grd area, shift it using periodicity or skip if not periodic. */
            while ( (x < info->grdhdr.x_min) && (info->edgeinfo.nxp > 0))
                x += (info->grdhdr.x_inc * info->edgeinfo.nxp);
            if (x < info->grdhdr.x_min)
                continue;  /* West of our area */
1975
            while ((x > info->grdhdr.x_max) && (info->edgeinfo.nxp > 0))
                x -= (info->grdhdr.x_inc * info->edgeinfo.nxp);
            if (x > info->grdhdr.x_max)
                continue;  /* East of our area */

            /* Get the value from the grid - it could be NaN */
            if (info->interpolate) {   /* IMG has been corrected, and GRD is good to go */
                g[n_grid][rec] = GMT_get_bcr_z (&info->grdhdr, x, y, grid, &info->edgeinfo, &info->bcr);
            }
            else {   /* Take IMG nearest node and do special stuff (values already set during read) */
                ii = GMT_x_to_i (x, info->grdhdr.x_min, info->grdhdr.x_inc, info->grdhdr.xy_off, info->
                    grdhdr.nx);
                jj = GMT_y_to_j (y, info->grdhdr.y_min, info->grdhdr.y_inc, info->grdhdr.xy_off, info->
                    grdhdr.ny);
                g[n_grid][rec] = grid[(jj+GMT_pad[3])*info->mx+ii+GMT_pad[0]];
            }
            pts++;
        }
        return pts;
    }
```

```
/* ----------------------------------------------------------------------
 * $Id: mgd77sniffer.h,v 1.10 2006/04/04 04:42:38 mtchandl Exp $
 *
 * File: mgd77sniffer.h
 *
 * Include file for mgd77sniffer
 *
 * Authors:
 *      Michael Chandler and Paul Wessel
 *      School of Ocean and Earth Science and Technology
 *      University of Hawaii
 *
 * Date: 23-Feb-2004
 *
 * ----------------------------------------------------------------------*/

#include "gmt.h"
#include "mgd77.h"

/* Constants */
#define MGD77_N_DATA_FIELDS          27
#define MGD77_NM_PER_DEGREE          60
#define MGD77_METERS_PER_NM          1852
#define MGD77_DEG_TO_KM              (6371.0087714 * D2R)
#define MGD77_NAV_PRECISION_KM       0.00111                 /* km */
#define MGD77_MAX_SPEED              10                      /* m/s */
#define MGD77_MEDIAN_SPEED           4.7                     /* m/s */
#define MGD77_MAX_DT                 900                     /* sec */
#define MGD77_MAX_DS                 5                       /* km */
#define MGD77_MAX_DUPLICATES         50
#define MGD77_METERS_PER_FATHOM      (6 * 12 * 2.54 * 0.01)
#define MGD77_FATHOMS_PER_METER      (1.0 / MGD77_METERS_PER_FATHOM)
#define MGD77_MIN_RLS_BINS           20
#define MGD77_MIN_RLS_PTS            100
#define MGD77_N_MAG_RF               13
#define MGD77_MAX_SEARCH             50

/* LMS Limits - Obtained from 4616 bathy and 1657 gravity cruises */
#define MGD77_MIN_DEPTH_SLOPE        0.900404                /* Q2.5 */
#define MGD77_MAX_DEPTH_SLOPE        1.1149                  /* Q97.5 */
#define MGD77_MAX_DEPTH_INTERCEPT    0
#define MGD77_MAX_DEPTH_MISFIT       140.676                 /* Q95 */
#define MGD77_MIN_FAA_SLOPE          0.868674      /*Q25(Q2.5=-0.17)*/
#define MGD77_MAX_FAA_SLOPE          1.06862                 /* Q97.5 */
#define MGD77_MAX_FAA_INTERCEPT      19.1557                 /* Q95 */
#define MGD77_MAX_FAA_MISFIT         7.98608                 /* Q95 */

/* Verbose Error Display Codes */
#define TYPE_WARN               0
#define TIME_WARN               1
#define DISTANCE_WARN           2
#define SPEED_WARN              3
#define VALUE_WARN              4
#define SLOPE_WARN              5
#define GRID_WARN               6
#define SUMMARY_WARN            7
#define MGD77_N_WARN_TYPES      8

/* E77 Error classes */
#define E77_NAV                   0
#define E77_VALUE                 1
#define E77_SLOPE                 2
#define E77_GRID                  3
#define N_ERROR_CLASSES           4
#define N_DEFAULT_TYPES       MGD77_N_NUMBER_FIELDS

/* E77 Nav Error Types */
#define NAV_TIME_OOR            1           /* A */
#define NAV_TIME_DECR           2           /* B */
#define NAV_HISPD               4           /* C */
#define NAV_ON_LAND             8           /* D */
#define NAV_LAT_UNDEF          16           /* E */
#define NAV_LON_UNDEF          32           /* F */
#define N_NAV_TYPES             6

/* E77 Header Errata Codes */
#define E77_HDR_SCALE           1
#define E77_HDR_OFFSET          2
#define E77_HDR_BCC             3
#define E77_HDR_PRECISION       4
#define E77_HDR_FLAGRANGE       5

struct BAD_SECTION { /* To flag a range of records as bad for given field */
   char abbrev[8];   /* Field name */
   int col; /* Column number */
   int start;  /* First record to flag */
   int stop;   /* Last record to flag */
};
#define MAX_BAD_SECTIONS    100

struct MGD77_GRID_INFO {
   struct GRD_HEADER grdhdr;
```

```
    struct GMT_EDGEINFO edgeinfo;
    struct GMT_BCR bcr;
    int one_or_zero, nx, ny, col, sign, g_pts, format, mode, mx, interpolate;
    double scale, max_lat;
    char abbrev[8];
    char fname[32];
};

struct MGD77_SNIFFER_DEFAULTS {
    char abbrev[8];       /* MGD77 field name abbreviations  */
    double minValue;      /* minimum accepted field value    */
    double maxValue;      /* maximum accepted field value    */
    double delta;         /* RLS decimation binsize interval */
    double maxTimeGrad;   /* maximum +/- time derivative     */
    double maxSpaceGrad;  /* maximum +/- space derivative    */
    double maxDiff;       /* maximum +/- value change        */
    double maxArea;       /* maximum grid offset area        */
};

struct MGD77_ERROR {
    unsigned int flags[6];
};

struct MGD77_MAG_RF {
    char *model;          /* Reference field model name */
    int code;             /* Reference field code       */
    int start;            /* Model start year           */
    int end;              /* Model end year             */
};

/* Local functions */
void read_grid (struct MGD77_GRID_INFO *info, float **grid, double w, double e, double s, double n
      , BOOLEAN bilinear, double threshold);
int sample_grid (struct MGD77_GRID_INFO *info, struct MGD77_DATA_RECORD *D, double **g, float *
      grid, int n_grid, int n);
void regress_ls (double *x, double *y, int n, double *stat, int col, double S_xx);
void regress_rls (double *x, double *y, int nvalues, double *stat, int col, double S_xx);
void regress_lms (double *x, double *y, int nvalues, double *stat, int gridField);
void regresslms_sub (double *x, double *y, double angle0, double angle1, int nvalues, int n_angle,
      double *stat, int gridField);
double lms (double *x, int n);
double median (double *x, int n);
```

```
/* ----------------------------------------------------------------------
 * $Id: mgd77snifferdefaults.h,v 1.7 2006/04/30 07:16:37 mtchandl Exp $
 *
 * File: mgd77snifferdefaults.h
 *
 * Include file for mgd77sniffer
 *
 * Authors:
 *      Michael Chandler and Paul Wessel
 *      School of Ocean and Earth Science and Technology
 *      University of Hawaii
 *
 * Date: 23-Feb-2004
 *
 * ----------------------------------------------------------------------
 * Abbrev,      min,      max,   delta,   maxdt,   maxds,   maxdz,   maxArea*/
{    "drt",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{     "tz",      -13,       12,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{   "year",     1939,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{  "month",        1,       12,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "day",        1,       31,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{   "hour",        0,       24,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "min",        0,       60,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "lat",      -90,       90,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "lon",     -180,      180,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "ptc",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "twt",        0,       15,      .1,   .0013,       1,     .15,     FALSE},
{  "depth",        0,    11000,      50,       1,    1000,     104,    100000},
{    "bcc",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "btc",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{   "mtf1",    19000,    72000,     200,      .32,     200,      40,     FALSE},
{   "mtf2",    19000,    72000,     200,      .32,     200,      40,     FALSE},
{    "mag",    -1000,     1000,      10,      .29,     200,      38,     FALSE},
{  "msens",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{   "diur",     -100,      100,       2,      .05,      20,       3,     FALSE},
{    "msd",    -1000,    11000,      10,      .33,    1000,      10,     FALSE},
{   "gobs",   977600,   983800,       5,     .043,     100,     3.6,     FALSE},
{    "eot",     -150,      150,       2,     .033,     100,     2.4,     FALSE},
{    "faa",     -400,      550,       5,     .045,     100,     4.8,     50000},
{    "nqc",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{     "id",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{    "sln",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE},
{   "sspn",    FALSE,    FALSE,   FALSE,   FALSE,   FALSE,   FALSE,     FALSE}
```

```
/* ----------------------------------------------------------------------
 * $Id: mgd77magref.h,v 1.4 2006/05/02 05:10:20 mtchandl Exp $
 *
 * File: mgd77magref.h
 *
 * MGD77 Magnetic Reference Fields for mgd77sniffer
 *
 * Authors:
 *    Michael Chandler and Paul Wessel
 *    School of Ocean and Earth Science and Technology
 *    University of Hawaii
 *
 * Date: 1-Aug-2005
 *
 *  References:
 *    Langel, R. A., 1987. The Main Field. In Geomagnetism, Volume I,
 *      (Ed. J. A. Jacobs). London: Academic Press, pp249-512.
 *
 * ----------------------------------------------------------------------*/
/*       Field,      Code, Start,       End */
        {"AWC 70",       1,     1965,     1970},
        {"AWC 75",       2,     1967,     1974},
        {"IGRF-65",      3,     1965,     1970},
        {"IGRF-75",      4,     1975,     1980},
        {"GSFC-1266",    5,     1900,     1966},
        {"GSFC-0674",    6,     9999,     9999},
        {"UK 75",        7,     1970,     1975},
        {"POGO 0368",    8,     1965,     1968},
        {"POGO 1068",    9,     1965,     1968},
        {"POGO 0869",   10,        1965,     1968},
        {"IGRF-80",     11,        1980,     1985},
        {"IGRF-85",     12,        1985,     1990},
        {"IGRF-90",     13,        1990,     1995}
```

A.2 Documentation

NAME

mgd77sniffer - Scan for errors in MGD77 data

SYNOPSIS

mgd77sniffer *NGDC-ids* [**-A***fieldabbrev,scale,offset*] [**-C***maxspd*] [**-D***d|e|f|l|m|s|v*]
[**-g***fieldabbrev,imggrid,scale,mode*] [**-G***fieldabbrev,grid*] [**-H**] [**-I***fieldabbrev,rec1,recN*] [**-N**] [**-O***custom limits file*] [**-Q**[*value*] [**-S***d|s|t*] [**-T***gap*] [**-W***c|g|o|s|t|v|x*] [**-V**] [**-b**[**s**|**d**]]
[**-R**<*west*>/<*east*>/<*south*>/<*north*>]

DESCRIPTION

mgd77sniffer scans old (pre-Y2K) and new format MGD77 files for errors using point-by-point sanity checking, along-track detection of excessive slopes, and optional comparison of cruise data with global gravity and predicted bathymetry grids. Detected data problems are output by default as verbose descriptions of each detected error, often resulting in multiple messages per scanned record. Data problems are optionally output (-De option) using a computer-parseable format (see E77 ERROR FORMAT description below). Default error thresholds are derived from histograms of all MGD77 geophysical data collected between 1952 and January, 2006. Thresholds are adjustable with the -O option.

NGDC-ids
Can be one or more of five kinds of specifiers:
1) 8-character NGDC IDs, e.g., 01010083, JA010010etc.,etc.
2) 2-character <agency> codes which will return all cruises from each agency.
3) 4-character <agency><vessel> codes, which will return all cruises from those vessels.
4) =<list>, where <list> is a table with NGDC IDs, one per line.
5) If nothing is specified we return all cruises in the database.
(See mgd77info **-F** for agency and vessel codes). The ¨.mgd77¨ or ¨.nc¨ extensions will automatically be appended, if needed (use **-I** to ignore certain file types). Cruise files will be looked for first in the current directory and second in all directories listed in **$MGD77_HOME**/mgd77_paths.txt [If **$MGD77_HOME** is not set it will default to **$GMTHOME**/share/mgd77].
NOTE: Under Windows, only (1) and (4) are currently supported.

REQUIREMENTS

The mgd77sniffer links with Generic Mapping Tools 4.0 or later along with the supplemental GMT packages x2sys and mgd77. See http://gmt.soest.hawaii.edu for GMT details. Grids for comparison with cruise data may be downloaded via the web. Among the many useful grids are NOAA's ETOPO2 global topography grid available from the National Geophysical Data Center's website as well as Sandwell and Smith's Raw Gravity grid available from Scripps Institution of Oceanography's website.

OPTIONS

-A apply scale factor and DC adjustment to specified data field. Allows adjustment of cruise data prior to along-track analysis. CAUTION: data must be thoroughly examined before applying these global data adjustments. May not be used for multiple cruises.

-C set maximum ship speed in m/s, or knots with **-N** option. Ship speeds exceeding 10 m/s (˜20 knots) are flagged as excessive by default.

-D suppress default warning output and only dump cruise data row-by-row such as values, gradients, grid-cruise differences, E77 error summaries for each record, re-created MGD77 records or sniffer limits.

-Dd output differences between cruise and grid data. Requires **-G** option. Output columns include:

lat lon dist cruiseZ gridZ diff [*cruiseZ2 gridZ2 diff2 ...*]

Note: grid values are subtracted from cruise data so a positive difference implies cruise > grid. For multiple grid comparison, *cruiseZ gridZ diff* are repeated for each grid comparison in command line order.

-De output E77 error classification format. Error output is divided into (1) a header containing information globally applicable to the cruise and (2) individual error records summarizing all errors encountered in each cruise record. mgd77sniffer writes E77 directly to <ngdc_id.e77> file handle. See **E77 ERROR FORMAT** below for additional details.

-Df output delta Z (change in geophysical field) column and delta S (change in distance) for each geophysical field. Distance between observations often differ for different fields depending on instrument sampling rate, so ds is included for each geophysical observation. Output columns include:

d[*twt*] *ds d*[*depth*] *ds d*[*mtf1*] *ds d*[*mtf2*] *ds d*[*mag*] *ds d*[*diur*] *ds d*[*msd*] *ds d*[*gobs*] *ds d*[*eot*] *ds d*[*faa*] *ds*

-Dl display mgd77sniffer limits. Customize this output to create a custom limits file for the **-O** option. No additional arguments are required. Output columns include:

fieldabbrev min max maxSlope maxArea

-Dm output MGD77 format records in Y2K-compliant MGD77 format

-Ds output calculated gradients for speed and geophysical fields. Gradients correspond to the gradient type selected in the **-S** option (spatial derivatives by default). Output columns include:

speed d[*twt*] *d*[*depth*] *d*[*mtf1*] *d*[*mtf2*] *d*[*mag*] *d*[*diur*] *d*[*msd*] *d*[*gobs*] *d*[*eot*] *d*[*faa*]

See **MGD77 FIELD INFO** below for field and abbreviations descriptions.

-Dv display values for the twelve position and geophysical fields for each MGD77 data record (in this order):

lat lon twt depth mtf1 mtf2 mag diur msens gobs eot faa

See below for **MGD77 FIELD INFO**.

-g compare cruise data to the specified grid in Sandwell/Smith Mercator format. Requires a valid MGD77 field abbreviation (see **MGD77 FIELD INFO** below) followed by a comma, the path (if not in current directory) and grid filename, a scale to multiply the data (1 or 0.1), and mode which stand for the following: (0) Img files with no constraint code, returns data at all points, (1) Img file with constraints coded, return data at all points, (2) Img file with constraints coded, return data only at constrained points and NaN elsewhere, and (3) Img file with constraints coded, return 1 at constraints and 0 elsewhere.

-G compare cruise data to the specified grid. Requires a valid MGD77 field abbreviation (see **MGD77 FIELD INFO** below) followed by a comma, then the path (if not in current directory)

and grid filename. Multiple grid comparison is supported by using separate **-g** or **-G** calls for each grid. See **GRID FILE INFO** below.

Grid comparison activates several additional error checks. (1) Re-weighted Least Squares Regression of ship versus grid data determines slope and DC shift, which when differing from expected 1 and 0, respectively, may indicate incorrectly scaled ship data, including incorrect units or instrument drift as well as erroneous gravity tie-in. (2) Accumulated ship grid offsets are computed along-track and excessive offsets are flagged according to *maxArea* threshold (use **-O** option to adjust *maxArea*). Warning: predicted bathymetry grids are constrained by cruise data so grids and cruise data are not always independent. Comparison of cruise bathymetry with predicted bathymetry grids also activates a ¨navigation crossing over land¨ check.

-H (with -G|g only) disable data decimation during RLS regression.

-I append a field abbreviation and the first and last record in a range of records that should be flagged as bad (and set to NaN prior to the analysis). Repeat as many times as needed. May not be used for multiple cruises.

-N use nautical units (SI by default).

-O override mgd77sniffer default error detection limits. Supply path and filename to the custom limits file. Rows not beginning with a valid MGD77 field abbreviation are ignored. Field abbreviations are listed below in exact form under MGD77 FIELD INFO. Multiple field limits may be modified using one default file, one field per line. Field min, max, max slope and max area may be changed for each field. Max slope pertains to the gradient type selected using the **-S** option. Max area is used by the **-G** option as the threshold for flagging excessive offsets from the specified grid. Dump defaults (**-Dl**) to view syntax or to quickly create an editable custom limits file.

Example custom default file contents (see below for units):
abbrev min max maxSlope maxArea
twt 0 15 1 0
depth 0 11000 500 5000
mag -800 800 - -
faa -300 300 100 2500

Use a dash '-' to retain a default limit. Hint: to test your custom limits, try:
mgd77sniffer -Dl -O<yourlimitsfile>

-Q quick mode, use bilinear rather than bicubic interpolation [Default]. Optionally, append *value* in the 0 <= *value* <= 1 range. This parameter controls how close to nodes with NaN values the interpolation will go. E.g., a *value* of 0.5 will interpolate about 1/2-way from a non-NaN to a NaN node, whereas 0.1 will go about 90% of the way, etc. [Default is 1, which means none of the four nearby nodes may be NaN]. A *value* of 0 will just return the value of the nearest node instead of interpolating.

-R *west, east, south,* and *north* specify the Region of interest, and you may specify them in decimal degrees or in [+-]dd:mm[:ss.xxx][W|E|S|N] format. Append **r** if lower left and upper right map coordinates are given instead of wesn. The two shorthands **-Rg -Rd** stand for global domain (0/360 or -180/+180 in longitude respectively, with -90/+90 in latitude).

-S specify gradient type for along-track excessive slope checking.
-Sd Calculate change in z values along track (dz). Output is given in geophysical units, e.g. mGal.
-Ss Calculate spatial gradients (dz/ds). Output is given in geophysical units per km along the survey track, e.g. mGal/km.

-St Calculate time gradients (dz/dt) [default]. Output is given in geophysical units per second along the survey track, e.g. mGal/sec.

-T adjusts mgd77sniffer gap handling. By default, data gaps greater than 5 km are skipped. Set to zero to de-activate gap skipping.

-W print out only certain warning types for verbose error messages. Comma delimit any combination of **c**|**g**|**o**|**s**|**t**|**v**|**x**: where (**c**) type code warnings, (**g**)radient out of range, (**o**)ffsets from grid (requires **-G**|**g**), (**s**)peed out of range, (**t**)ime warnings, (**v**)alue out of range, (**x**) warning summaries. By default ALL warning messages are printed.Not compatible with any **-D** options.

-V runs in verbose mode.

-b output binary data for **-D**d|f|s|v option. Append **s** for single and **d for double precision** [**Default is double**].

MGD77 FIELD INFO

Field Abbreviation Units
Two-way Travel Time twt sec
Corrected Depth depth m
Mag Total Field1 mtf1 nT
Mag Total Field2 mtf2 nT
Residual Magnetic mag nT
Diurnal Correction diur nT
Mag Sensor Depth/Alt msens m
Observed Gravity gobs mGal
Eotvos Correction eot mGal
Free Air Anomaly faa mGal

GRID FILE INFO

For **-g** the grids must be in the format used by Sandwell & Smith, which is a spherical Mercator 2-byte grid with no header. For **-G** the grid files can be of any grid type supported by GMT and therefore must contain a GMT header. A correctly formatted *.i2 grid file can be generated using grdraster as shown below.

gmtset GRIDFILE_SHORTHAND TRUE

Create/edit .gmt_io file to include the following rows:
GMT I/O shorthand file
suffix format_id scale offset NaN
grd 0 - - -
i2 2 - - 32767

grdraster 1 -R0/359:55/-90/90 -Getopo5_hdr.i2

The new grid, etopo5_hdr.i2 in this example, contains a GMT header and can be used in the **-G** option to compare cruise depth with grid values.

E77 ERROR FORMAT

Header

Information pertaining to an entire cruise, such as NGDC and survey institution identification codes, cruise examination time, two-way travel time corrector information, data precision warnings, as well as systematic scales, DC shifts and correlation coefficients from global grid comparisons are reported as E77 header information.

Sample
Cruise 06050010 ID L476WG MGD77 FILE VERSION: 19870415 N_RECS: 27268
Examined: Tue Feb 21 16:33:34 2006
Examiner: mtchandl
Arguments: -De -Gfaa,/data/GRIDS/sandwell_smith11_hdr.i2
Errata: Header
N-faa-E-01: Regression against grid gave scale 0.108643 which is statistically different from 1.0. Recommended scale: (10.0)
N-faa-E-02: Regression against grid gave DC offset -1.319411 which is statistically different from 0.0
I-twt-W-03: More recent bathymetry correction table available
I-depth-W-04: Integer precision in depth

Error Record

Individual error records have strict format. Included is a time or distance column followed by record number, a formatted error code string, and finally a verbose description of errors detected in the record. Four error classes are encoded into the error code string with different alphabetic characters representing unique error types. See below for error code format description.

Format
<time/distance> <record number> <error code string> <description>

Sample
Errata: Data
1971-04-28T18:27:30 345 0-0-OQ-0 gradient: mtf1, mag excessive
1971-05-14T18:01:30 5413 0-0-OQ-0 gradient: mtf1, mag excessive
1971-05-15T03:41:30 5634 0-Q-0-0 value: mag invalid

Error Code Description

Each of the four error classes is separated by a dash '-' and described by a combination of alphabetic characters or 0 signifying no detected problems.

Error classes: NAV-VAL-GRAD-GRID

Error Class Descriptions
NAV (navigation):
0 - fine
A - time out of range
B - time decreasing
C - excessive speed
D - above sea level
E - lat undefined
F - lon undefined

VAL (value):
0 - fine
K - twt invalid
L - depth invalid

O - mtf1 invalid
etc.

GRAD (gradient):
0 - fine
K - d[twt] excessive
L - d[depth] excessive
O - d[mtf1] excessive
etc.

GRID (grid comparison):
0 - fine
L - depth offset
W - faa offset
etc.

The NAV error class has unique cases while VAL, GRAD and GRID classes are described by alphabetic characters for each of the 24 numeric fields in MGD77 format order.

```
MGD77 bit-pattern w/ E77 alpha characters
|-----------------------------------------------|----------|
| X W V U T S R Q P O N M L K J I H G F E D C B A | E77 Code |
| - - - - - - - - - - - - - - - - - - - - - - - - | - - - - -|
| n f e g m d m m m m b b d t p l l m h d m y t d | F  I     |
| q a o o s i s a t t t c e w t o a i o a o e z r | i  D     |
| c a t b d u e g f f c c p t c n t n u y n a   t | e        |
|       s   r n   2 1     t           r   t r     | l        |
|             s           h               h       | d        |
| - - - - - - - - - - - - - - - - - - - - - - - - | - - - - -|
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | Bit place|
| - G C G C C - G G G - - G G - - - T T T T T - - | Bit type |
|-----------------------------------------------|----------|
  Bit types: (G)eophysical, (C)orrection, (T)ime
```

EXAMPLES

To scan for excessive values or gradients, try

mgd77sniffer 08010001

To dump cruise gradients, try

mgd77sniffer 08010001 -Ds

To compare cruise depth with ETOPO5 bathymetry and gravity with Sandwell/Smith 2 min gravity version 11, try

mgd77sniffer 08010001 -Gdepth,/data/GRIDS/etopo5_hdr.i2 -gfaa,/data/GRIDS/grav.11.2.img,0.1,1