Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Tools for analyzing intersecting tracks: The x2sys package

Paul Wessel*

Department of Geology and Geophysics, School of Ocean and Earth Science and Technology, University of Hawaii at Manoa, 1680 East–West Road, Honolulu, HI 96822, USA

ARTICLE INFO

Article history:
Received 23 February 2009
Received in revised form
28 May 2009
Accepted 29 May 2009

Keywords:
Track intersection
Crossover errors
Leveling
Data quality improvement

ABSTRACT

I present a new set of tools for detection of intersections among tracks in 2-D Cartesian or geographic coordinates. These tools allow for evaluation of crossover errors at intersections, analysis of such crossover errors to determine appropriate linear models of systematic corrections for each track, and application of these corrections and further adjustments to data that completely eliminates crossover discrepancies from final 2-D data compilations. Unlike my older x_system tools, the new x2sys tools implement modern algorithms for detecting track intersections and are capable of reading a wide range of data file formats, including data files following the netCDF COARDS convention. The x2sys package contains several programs that address the various tasks needed to undertake a comprehensive crossover analysis and is distributed as a supplement to the Generic Mapping Tools, making them available for all computer platforms and architectures.

1. Introduction

Measuring one or more observables as a function of position or time along-tracks is a standard procedure in many scientific or engineering endeavors. Examples include underway geophysical observations collected by oceanographic vessels, remotely-sensed data collected by orbiting satellites, airborne collection campaigns, or terrestrial measurements of various types obtained along crisscrossing traverses. In many applications, the users will compile all data obtained by one or more such expeditions and use interpolation methods to resample the measured data from their arbitrary locations along track lines onto an equidistant lattice required for further analysis. It has long been known that complications are associated with this seemingly innocuous task, giving rise to what is known as crossover errors (COE) (e.g., Wessel and Watts, 1988; Smith, 1993). These are defined to be the difference between the two repeat measurements at track intersections. Such COE can result from a variety of sources. For instance, the coordinates of the track may have uncertainties and therefore the location of the track intersection may not really correspond to the correct repeat measurement point along the two tracks. Unless the phenomenon being measured is constant, one will find nonzero COE simply due to this mis-registration (e.g., Fox et al., 1992; Neumann et al., 2001). Interactive tools, such as MB-system's mbnavedit (Caress and Chayes, 1996) may be used to correct such problems. Other sources of difficulty derive from variability of the phenomena itself or how the observations are obtained, such as the introduction of aliasing. Some phenomena, such as the Earth's magnetic field, change over time and unless proper corrections are made, one may end up with COE between tracks from different epochs. Other COE may derive from improper or missing calibration of instruments, and in some cases different tracks may have used different units for the measurements. There are thus numerous examples of studies in which COE or leveling corrections have been applied (e.g., Foster et al., 1970; Swan and Young, 1978; Yarger et al., 1978; Kellogg, 1979; Prince and Forsyth, 1984; Green, 1987; Nishimura and Forsyth, 1988; Tai, 1988; Wessel and Watts, 1988; Ghosh and Hall, 1992; Minty, 1991; Klokocnik et al., 1998; Huang and Fraser, 1999; Neumann et al., 2001; Mauring and Kihle, 2006; Huang, 2008) and some software solutions have been published (Wessel, 1989; Hsu, 1995).

The approach presented here divides the many tasks that comprise a comprehensive crossover analysis into smaller steps, and each step has been encoded as a stand-alone program. The step-by-step assessment makes it easy to test intermediate results and make sure all procedures are well described and documented. While most of the tools deal with crossover calculation and extraction, some are dedicated to maintaining a database of all tracks and their content, which is useful when tracks contain more than one type of measurement and when one needs to access just the tracks that contain a particular measurement of interest. In the next sections, I present an overview of the various tools, and then demonstrate their use with a synthetic case study. In the interest of brevity and to preempt confusion arising from possible future syntax changes, I will not describe the particular program syntax here since these are best left to the documentation that comes with the package

* Tel.: +1 808 956 4778; fax: +1 808 956 5154.
E-mail address: pwessel@hawaii.edu

0098-3004/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2009.05.009

because it will be updated, as changes require. The x2sys package is distributed as an official supplement to the Generic Mapping Tools (Wessel and Smith, 1995) and benefits from the general processing capabilities of GMT. In turn, GMT uses the x2sys algorithms for hidden tasks such as determining intersections between map contours and annotation lines.

2. The x2sys family of tools

There are nine tools in the x2sys suite, each dedicated to a particular aspect of the analysis. Not all are necessarily required for a particular analysis; I will indicate their typical use as I discuss their general purposes.

2.1. Initializing with x2sys_init

The tool x2sys_init is the starting point for anyone wishing to use x2sys; it initializes a set of databases that are particular to one kind of track data. These data, their associated databases, and key parameters are given a shorthand notation called an x2sys tag. The tag keeps track of common settings such as data format, whether the data are geographic or Cartesian, what units to use for speed and distance, the extent of the data domain, how to recognize data gaps along-tracks (based on exceeding time or distance criteria) and the optional binning resolution for track indices (more on this below). Running x2sys_init is a prerequisite to running any of the other x2sys programs. A key input to x2sys_init is a description of the data file format. This information is encoded in an ASCII table called a definition file that must have the extension ".def". The definition file has two sections: Header information and column information. All header information starts with the character # in the first column, immediately followed by an upper-case directive. Spaces or tabs separate directives from any optional argument. Five header directives are recognized: ASCII indicates that the data files are in ASCII format, BINARY states that the data files are native binary files, and NETCDF states that the data files are COARDS-compliant 1-D netCDF files (see http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html). SKIP takes an integer argument that is either the number of lines to skip (for ASCII files) or the number of bytes to skip (for native binary files); it is not used with netCDF files. GEO indicates that these files are geographic data sets, with periodicities in the x-coordinate (longitudes). MULTISEG means each track consists of multiple segments separated by a GMT multi-segment header; again, this directive is not used with netCDF files.

The column information consists of one line per column in the order that the columns appear in the data file. For each column you must provide the seven attributes *name type NaN NaN-proxy scale offset oformat*. Here, *name* is the name of the column variable. It is required that you use the special names lon (or x if Cartesian) and lat (or y) for the two required coordinate columns, and time when optional time data are present; other column names have no restrictions. The *type* is always **a** for ASCII representations of numbers, whereas for binary files you may choose among **c** for signed 1-byte character (−127,+128), **u** for unsigned 1-byte (0–255), **h** for signed 2-byte integers (−32768, +32767), **i** for signed 4-byte integers (−2,147,483,648, +2,147,483,647), **f** for 4-byte floating points, and **d** for 8-byte double precision floating points. For netCDF files, simply use **d** because netCDF will automatically handle type-conversions during reading. NaN should be **Y** if certain values (e.g., −9999) are to be replaced by NaN (i.e., IEEE "Not-a-Number"), and **N** otherwise; *NaN-proxy* should hold the designated replacement value. Next, *scale* is used to multiply the data after reading, while

Table 1
Example of a file format definition file (here, geoz.def).

```
# Definition file for X2SYS processing of ASCII lon,lat,z files
#
# This file applies to plain 3-column ASCII files
#
#
#ASCII              # The input file is ASCII
#SKIP 1             # The number of header records to skip
#
#name   intype   NaN-proxy?   NaN-proxy   scale   offset   oformat
lon     a        N            0           1       0        %10.5 f
lat     a        N            0           1       0        %9.5 f
z       a        N            0           1       0        %6.1 f
```

offset is used to add to the scaled data. Finally, *oformat* is a C-style format string used to print values from this column. If you specify – as the *oformat*, then GMT's formatting machinery will be used (i.e., you can any format you like by manipulating the settings for D_FORMAT, PLOT_DEGREE_FORMAT, PLOT_DATE_FORMAT, and PLOT_CLOCK_FORMAT).

Some file formats already have definition files that come with the x2sys distribution. These include *mgd77* for plain ASCII MGD77 data files (Hittelman et al., 1977), *mgd77+* for enhanced MGD77+ netCDF files (Wessel and Chandler, 2007), *gmt* for old mgg supplement binary files (Wessel and Smith, 1991), *xy* for ASCII Cartesian (x, y) tables, *xyz* extends *xy* with one z-column, *geo* for plain ASCII (longitude, latitude) files, and *geoz* which extends *geo* with one z-column. The latter data file format will be used for the examples in this paper and is reproduced in Table 1. For any other formats you must craft your own definition file; the x2sys_init documentation provides further examples.

2.2. Maintaining a track database

If your track data have more than one observed data type or you anticipate getting additional tracks later (and at that time will need to recompute cross-overs involving these new tracks and the older tracks they cross), I recommend you maintain and use a track database. During the initialization above you can specify a binning interval, which lets the database know the geographic extent of each track at that resolution. E.g., if you choose 1° then the database will maintain indices at the $1^\circ \times 1^\circ$ resolution and thus know all such bins that a track passes through. There are three x2sys programs that maintain and use the track database: (1) x2sys_binlist will read all new data tracks and create a bin index file based on the chosen bin interval, (2) x2sys_put will read the new bin index file and append the information to the binary database created by x2sys_init in the previous section, and (3) x2sys_get is used to query the database about tracks that satisfy specific criteria (e.g., geographic region, type of observables), or optionally to provide a list of track pairs that might cross based on their indices and thus should be subject to further crossover assessment.

2.3. Determining crossover errors

The program x2sys_cross is used to determine all intersections between two tracks ("external cross-overs") or within a single track ("internal cross-overs"), and will report the time, position, distance along-track, heading and speed along each track segment, and the crossover error (COE) and mean values for all observables in the track files. You may specify a list of tracks (in which case the program will examine all possible pairs for