Geological Survey of Norway
NO-7441 Trondheim, Norway
Tel.: 47 73 90 40 00
Telefax 47 73 92 16 20

REPORT

Report no.: 2005.057 | ISSN 0800-3416 | Grading: Open

Title:
A software tool to rotate spatial data on the surface of the sphere (Earth) for Geosoft's Oasis Montaj

Authors:
Smethurst, M.A.

Client:
Statoil

County:

Commune:

Map-sheet name (M=1:250.000)

Map-sheet no. and -name (M=1:50.000)

Deposit name and grid-reference:

Number of pages: 18 Price (NOK): 83,-
Map enclosures:

Fieldwork carried out:	Date of report:	Project no.:	Person responsible:
	22/08/2005	307700	

Summary:

Movement of the Earth's plates through geological time can be described in terms of a series of rotations on the sphere (Earth). If we can determine what these rotations have been, we can rotate ancient geological features with their associated earth science data sets back to their former positions to produce paleogeographic maps.

We present an open source computer program that performs such rotations on the sphere. The program is written for *Microsoft Windows* and runs inside the widely used *Oasis Montaj* mapping software produced by Geosoft Inc. (www.geosoft.com) for processing, visualizing and analysing large earth science datasets (geophysical, geological and geochemical). The rotation function takes the form of a "GX" that is activated via a menu in the *Oasis Montaj* program and can operate on any geo-referenced data stored in a Geosoft database file.

The rotation program is distributed free of charge together with this report, but a licensed version of Geosoft's *Oasis Montaj* is required to run it. Anyone installing and using our rotation software does so at his or her own risk. We do not accept responsibility for any loss or corruption of data that might occur through use of the software.

Keywords: Euler rotation	Oasis Montaj	Geosoft
Paleogeography	Visual C++	
Rotation	GX	

CONTENTS

1. INTRODUCTION 4

2. INSTALLATION OF THE ROTATION SOFTWARE 5

3. OPERATION OF THE ROTATION SOFTWARE 6

4. SOFTWARE DISTRIBUTION 10

5. REFERENCES 11

APPENDIX SOURCE CODE 12

1. INTRODUCTION

Earth's plates are in motion. Over millions of years, plates move over large potions of the Earth to produce continuously evolving paleogeographies. The translation of a plate from one place to another can be described by a rotation on the sphere about an Euler pole (Cox & Hart 1986). An Euler pole is simply the pivot point about which the rotation takes place. It has a geographic position expressed in terms of its latitude and longitude, and a rotation angle associated with it in degrees counter-clockwise.

Construction of a palaeographic map for any given time in Earth history (e.g. Figure 1) requires knowledge of the present positions of the geographic/geologic entities to be placed on the map, and the Euler poles that translate the entities back to their former positions on the globe.

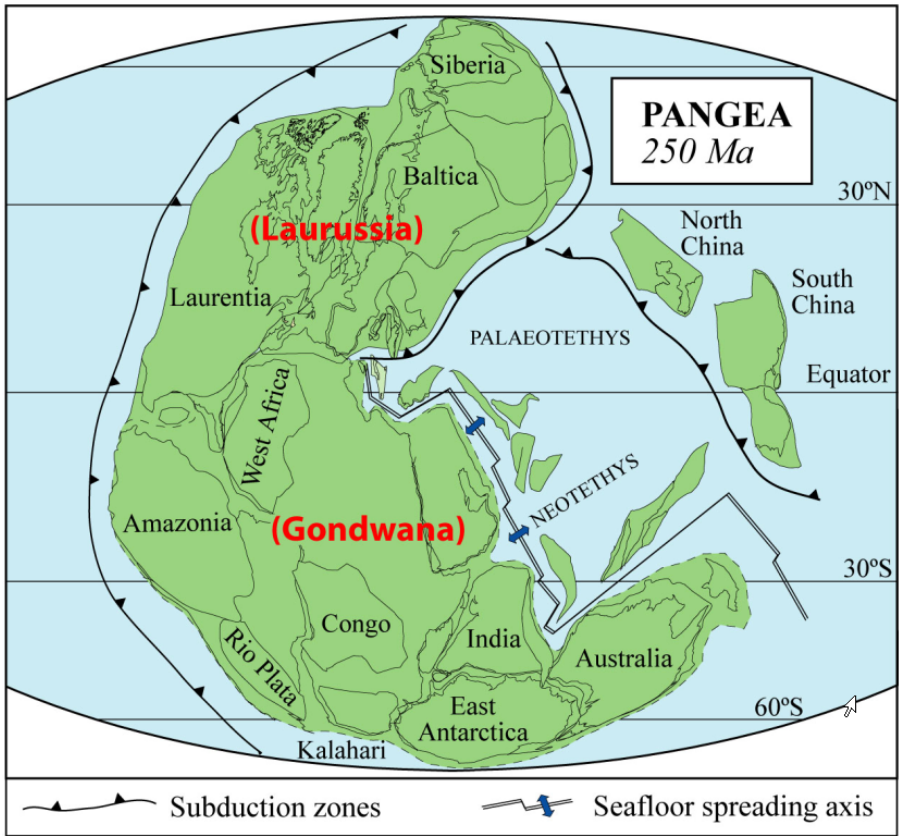


Figure 1. Paleogeographic map showing the former supercontinents Pangea, 250 million years ago (Torsvik, 2003). The present coastlines of the continents that made up Pangea are indicated by the thin solid lines.

The software presented here is designed to rotate map features and geo-referenced earth science data sets from one place to another on the surface of the globe according to user-defined Euler poles. The rotation algorithm is implemented within the commercial map-making and data processing program *Oasis Montaj* developed and distributed by Geosoft Inc. (www.geosoft.com). *Oasis Montaj* is a widely used and sophisticated tool for manipulating spatial data, including point data from geochemical surveys, line data from geophysical surveys, and data arranged in grid (raster) form. Earth science data of all kinds are easily imported into Geosoft databases (file type .GDB) and manipulated there.

Functioning as a *Geosoft Executable* (GX), our rotation algorithm can be run from an *Oasis Montaj* menu. The function reads spatial co-ordinates from any given Geosoft database and writes rotated co-ordinates back to the same database. The data in the database can then be displayed on maps in their restored paleo-positions. Complete paleogeographic maps like Figure 1 may be produced by rotating the contents of multiple Geosoft databases, in this case containing coastlines, about multiple Euler poles.

2. INSTALLATION OF THE ROTATION SOFTWARE

The rotation software is written in Microsoft Visual C++ .NET for the Microsoft Windows operating system and can only be run in its present form as a GX from within Geosoft's map-making and data processing program *Oasis Montaj*.

1. Geosoft's *Oasis Montaj* software package (v. 6.0 or higher) must be installed on the computer

2. Our rotation algorithm and supporting files reside in 3 files on a CD accompanying this report. Copy them into the Geosoft *Oasis Montaj* folders specified below:

File	**Geosoft folder**
Euler.GX	e.g. *C:\Program Files\Geosoft\Oasis montaj\gx*
MarkDev_.DLL	e.g. *C:\Program Files\Geosoft\Oasis montaj\bin*
Euler_rotation.omn	e.g. *C:\Program Files\Geosoft\Oasis montaj\omn*

3. The rotation function EulerR.GX is now available for use with Oasis Montaj.

   Run the GX in *Oasis Montaj* by starting Oasis Montaj and either

   (a) Clicking on the "GX" button
   (b) Selecting menu item *GX / Run GX…*
   (c) Installing the custom menu *Euler_rotation.omn* and selecting menu item *Euler_rotation / Rotate Lat/Long pairs.*

The software is now installed.

3. OPERATION OF THE ROTATION SOFTWARE

Requirements

1. EuleR.GX must be installed (see section 2)

2. The spatial data to be rotated must reside in a Geosoft database file (.GDB). If not, import them using one of the many techniques available in *Oasis Montaj* menu *Data / Import.*

3. If grid (raster) data are to be rotated, transfer the data from whatever grid file format they may be in to a Geosoft database file using *Oasis Montaj* menu item *Grid / Grid Utilities / Save grid to database…*

4. The database must include channels containing geographic latitude and longitude. If not, generate geographic latitude and longitude channels from the spatial co-ordinates that are present in the database using *Oasis Montaj* menu item *Coordinates / New projected coord…*

Performing a rotation

1. Open a project in *Oasis Montaj*

2. Open the Geosoft database that is to be rotated. The database should conform with the requirements above

3. Identify the database channels that contain latitude and longitude

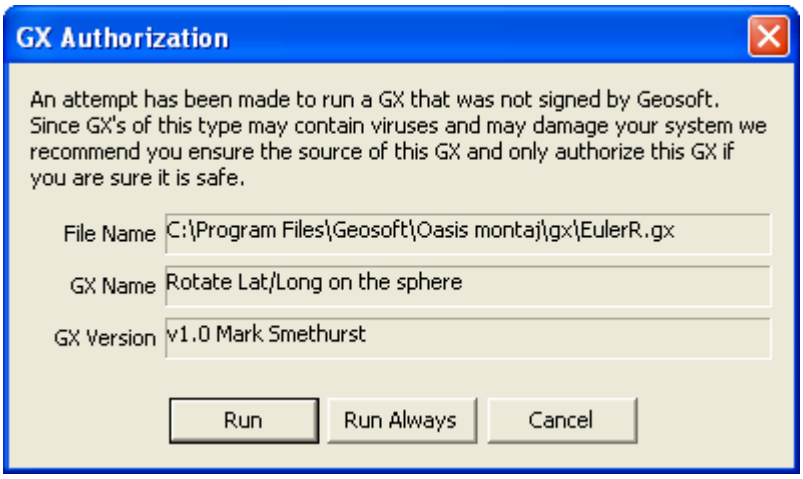
   (a) **Note that the rotation algorithm reads from the latitude and longitude channels and puts the rotated values back into the same channels, overwriting original latitude and longitude values. If original latitude and longitude values are to be preserved, make copies of the channels before performing the rotation**

   (b) If the latitude and longitude channels are defined as the database's current spatial reference (the channel names have "y" and "x" next to them), the rotation algorithm automatically copies the contents of the two channels to new ones with names starting "_" before performing the rotation

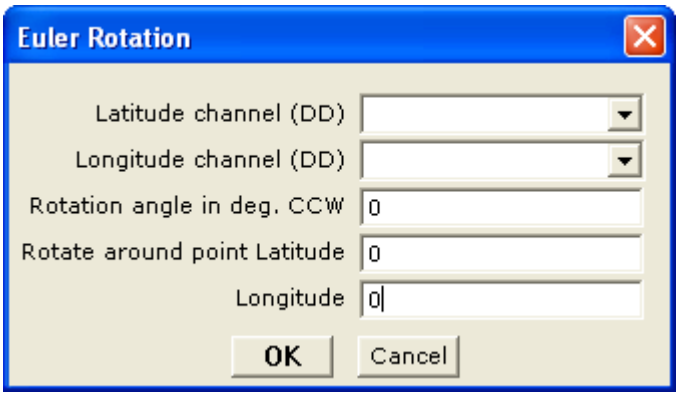
   (c) If the latitude and longitude channels are "protected" (read only), remove this protection before rotating

4. The rotation algorithm will only operate on those database "Lines" that are selected (by default all lines are selected)

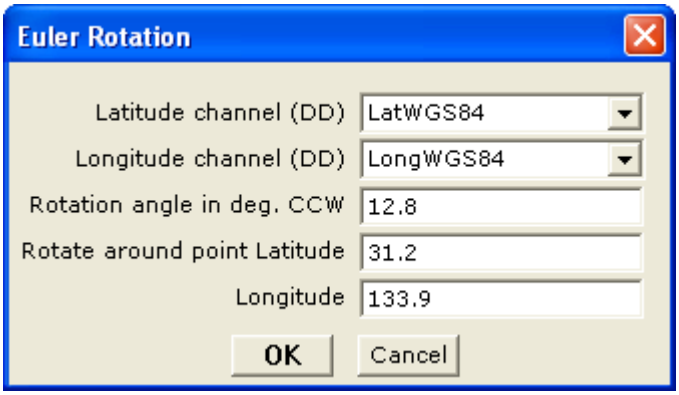
   Activate the GX "EulerR.GX" (see, section 2 step 3). The authorisation dialog box below will only appear the first time the GX is run. Click "Run Always".



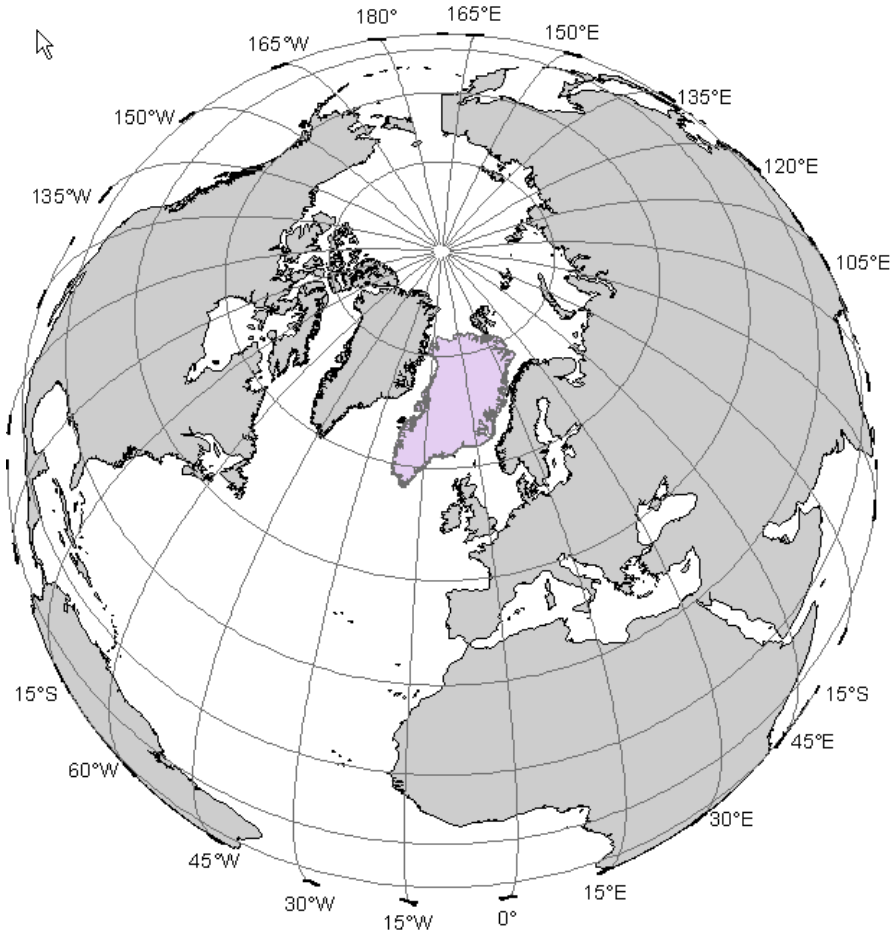
Then the rotation GX's own dialog box will be displayed:



Fill in the dialog box by (a) selecting the names of the latitude and longitude database channels from drop-down lists, (b) specifying the rotation to be applied to these channels in terms of rotation angle in degrees counter-clockwise and position of the Euler (rotation) pole e.g.:



5. Press "OK" to carry out the specified rotation. The example (above) is the rotation that translates Greenland back to the position it had with respect to Europe 250 million years ago (see Mosar et. al. 2002). If the Geosoft database contained the coastline of Greenland, Greenland would be rotated into its former position against the Atlantic coast of Norway as shown below:



Special considerations for rotating grid data

Oasis Montaj can store and manipulate grid (raster) data in a wide variety of grid formats (e.g. USGS, ESRI, World Geoscience, ER-Mapper and Surfer file formats). The simplest way to rotate data in grid form is to:

1. Transfer the grid data to a Geosoft database using *Oasis Montaj* menu item *Grid / Grid Utilities / Save grid to database…*

2. The X/Y co-ordinate system for the database will be the same as the grid. If this is not geographic (lat/long), use *Oasis Montaj* menu item *Coordinates / New projected coord…* to generate database channels containing latitude and longitude

3. Rotate the co-ordinates in the latitude and longitude channels as indicated in "***Performing a rotation***"

and to put the rotated data into a new grid file …

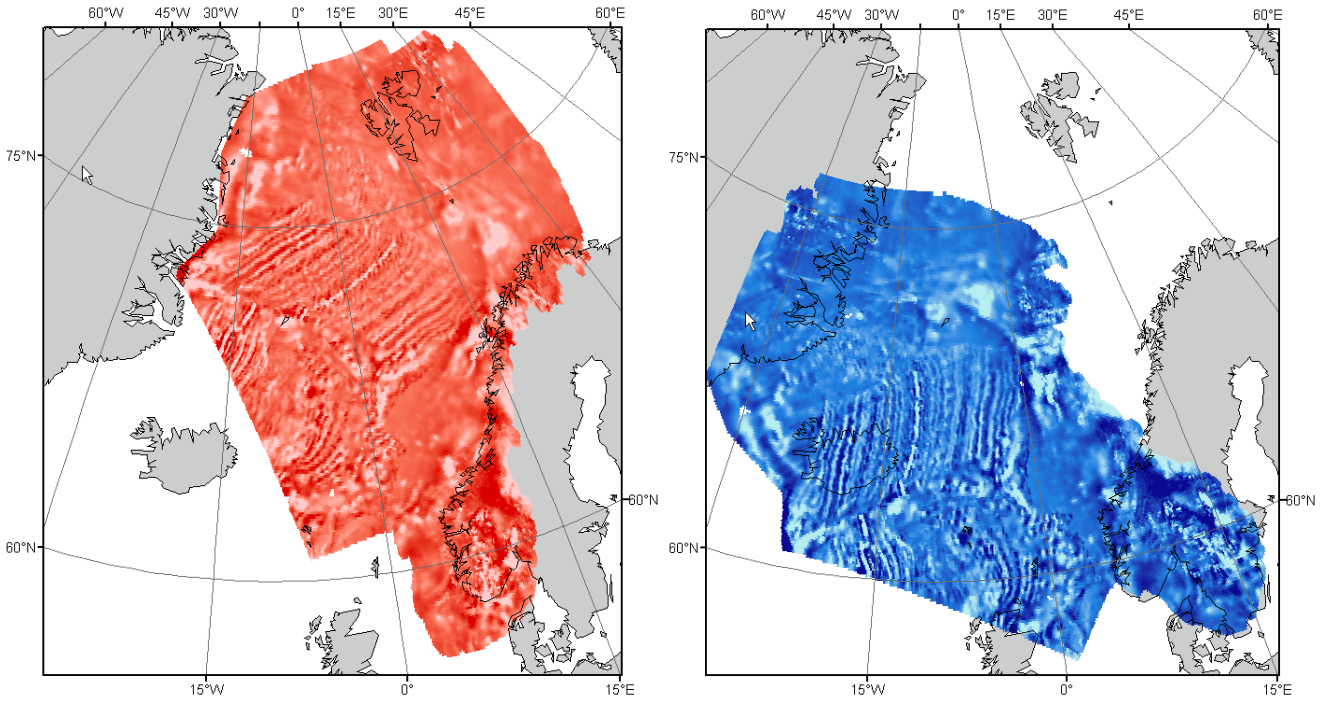
4. Decide upon an appropriate X/Y co-ordinate system for the new grid file

5. Use *Oasis Montaj* menu item *Coordinates / New projected coord…* to generate new X and Y channels in the chosen co-ordinate system from the rotated latitude and longitude channels

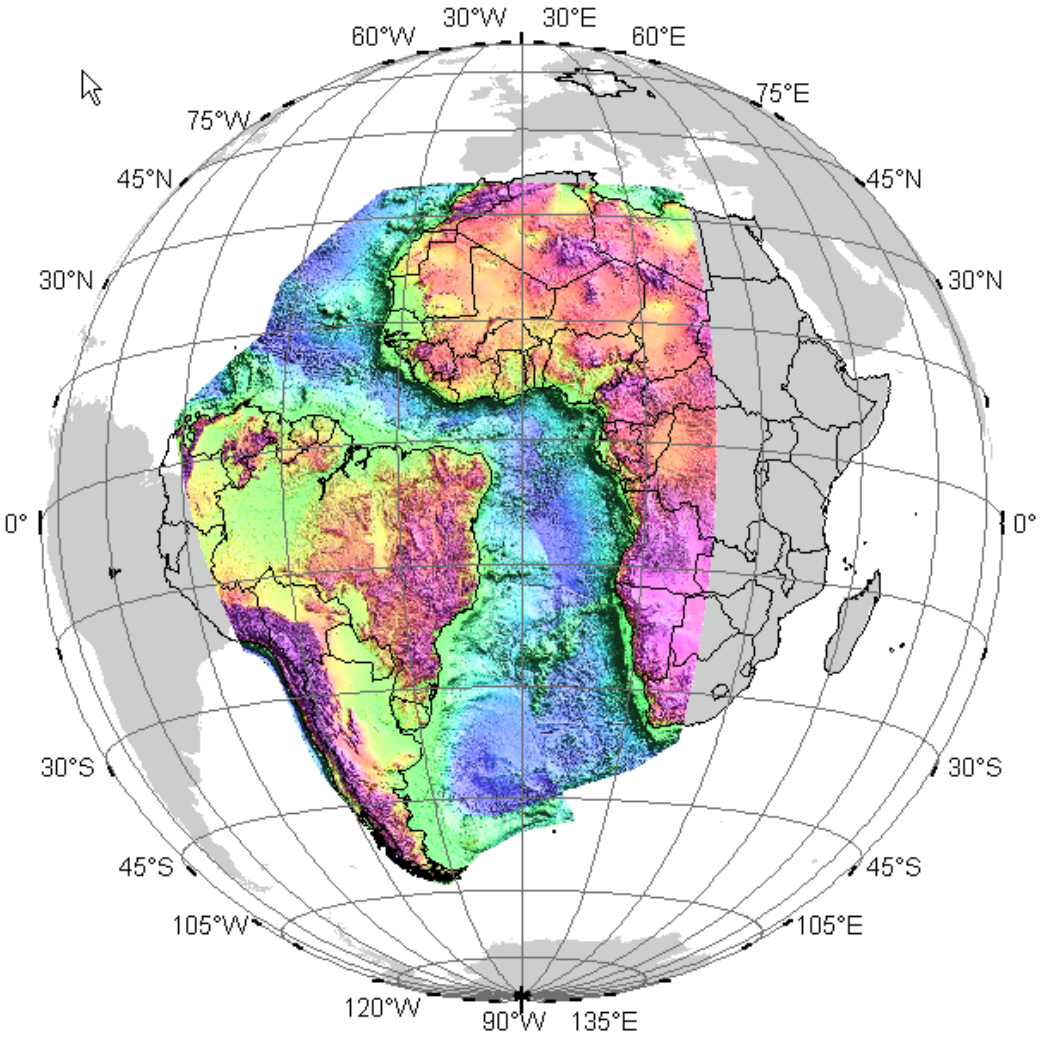
6. Use *Oasis Montaj* menu item *Coordinates / Change X,Y coordinates…*to set the new X and Y channels to be the database's current spatial reference ("x" and "y" will appear next to the channel names in the database)

7. Generate a new grid data file based on the rotated data set by running one of the gridding routines in *Oasis Montaj* menu *Grid / Gridding*

The example below shows a grid of magnetic anomaly values for the North Atlantic (red colour scale) and the same grid rotated through 45 degrees counter-clockwise about a pivot point at 63 degrees north, 10 degrees east.



The next example (below) shows South America rotated into its position relative to Africa 83.5 million years ago. The diagram was produced by inserting the country outlines for South America (line data) into a Geosoft database and rotating them 33.52 degrees counter-clockwise about 61.64 degrees north / 326.26 degrees east (Müller et al. 1997). The topography/bathymetry grid for South America and adjacent Atlantic Ocean floor out to magnetic anomaly 34 (83.5 My) was inserted into another database and rotated by the same amount.



4. SOFTWARE DISTRIBUTION

This software is distributed electronically as compiled files ready to install and use, and as C++ and GXC source code files that can be inspected, modified, and compiled using Microsoft Visual C++ .NET and Geosoft GX Developer. Source code is also listed in the Appendix at the end of this document. A licence for Geosoft's *Oasis Montaj* v. 6.0 or later is required to run the software.

> Anyone installing and using our rotation software does so at his or her own risk. We do not accept responsibility for any loss or corruption of data that might occur through use of the software.

5. REFERENCES

Cox, A. & Hart, R.B. 1986: Plate Tectonics: How It Works. Blackwell Science Ltd., Oxford, UK. 392p.

Mosar, J., Torsvik, T.H. & the BAT team 2002: Opening of the Norwegian and Greenland Seas: Plate tectonics in Mid-Norway since the Late Permian. *In:* Eide E.A.(coord.) BATLAS - Mid Norway Plate Reconstruction Atlas with Global and Atlantic Perspectives. Geological Survey of Norway, Trondheim, pp. 48-59.

Müller, R.D., Roest, W.R., Royer, J-Y, Gahagan, L.M. and Sclater, J.G. 1997: Digital isochrons of the world's ocean floor. J. Geophys. Res. 102, 3211-3214.

Torsvik, T.H. 2003: The Rodinia jigsaw puzzle, Science, 300, 1379-1381.

APPENDIX SOURCE CODE

EulerR.grc

```
//-------------------------------------------------------
// EulerR.grc
//
// Specifications for the dialog box. This file is
// compiled using Geosoft GX developer command GRC
// to produce files:
//
//      EulerR.gr   RESOURCE file for EulerR.GXC
//      EulerR.grh  INCLUDE file for EulerR.GXC
//-------------------------------------------------------

RESOURCE,FORM,EULERR,"Euler Rotation",-1
LEDIT,,,24,"Latitude channel (DD)",R,FORCE,Y,CHAN
LEDIT,,,24,"Longitude channel (DD)",R,FORCE,X,CHAN
EDIT ,,,24,"Rotation angle in deg. CCW",R,real,0.0
EDIT ,,,24,"Rotate around point Latitude",R,real,0.0
EDIT ,,,24,"Longitude",R,real,0.0

EBUT,&OK,0
EBUT,&Cancel,-1,CANCEL

// HBUT,&Help,help   no help provided on this dialog

RESOURCE,LIST,CHAN

RESOURCE,HELP,help,nogx.hlp
```

EulerR.gxc

```
//-------------------------------------------------------------------------
// EulerR.gxc
//
// C-like source code for EulerR.gx. This file is compiled to produce
// EulerR.gx using Geosoft GX developer command GRC
//
// Successful compilation of this file depends on the existence of the
// following files:
//
//   RESOURCE file EulerR.gr (generated by running command GRC on EulerR.grc)
//   INCLUDE file EulerR.grh (generated by running command GRC on EulerR.grc)
//   INCLUDE file MarkDev_.gxh (GX prototype header for calling RotateXY_ in
//                              MarkDev_.dll)
//-------------------------------------------------------------------------
//
//
//-------------------------------------------------------------------------
NAME          = "Rotate Lat/Long on the sphere"
VERSION       = "v1.0 Mark Smethurst"
DESCRIPTION   = "

      EULERR.LatCHAN   - Latitude channel
            .LongCHAN  - Longitude channel
            .EulerLat  - Lat,Long point about which to rotate
            .EulerLong
            .Angle     - rotation angle in degrees CCW
"
// Revision History:
//
//-------------------------------------------------------------------------
//                                RESOURCES
//-------------------------------------------------------------------------
RESOURCE = "EulerR.gr"
#include "EulerR.grh"

//-------------------------------------------------------------------------
//                                 INCLUDE
//-------------------------------------------------------------------------
#include <all.gxh>     // system
```

```
#include "MarkDev_.gxh"

//------------------------------------------------------------------------------
//                                 VARIABLES
//------------------------------------------------------------------------------
string(STR_DB_SYMBOL) sXCh,sYCh,sXBackCh,sYBackCh,sXNewCh,sYNewCh;

EDB         EData;            // Database handle
DB          Data;             // Database handle
DGW         Diag;             // Dialogue handle
LST         List;             // List handle
int         i;
real        rAngle,rXo,rYo;

//------------------------------------------------------------------------------
//                                   CODE
//------------------------------------------------------------------------------

{
   // --- Get database ---
   EData = Current_EDB();
   Data = Lock_EDB(EData);

   // --- Are we running interactively ? ---
   if (iInteractive_SYS())
   {

      // --- Create the Dialogue ---
      Diag = Create_DGW("EULERR");

      // --- Set up lists ---
      List = GetList_DGW(Diag,_EULERR_0);
      SymbLST_DB(Data,List,DB_SYMB_CHAN);
      Sort_LST(List,0,0);

      List = GetList_DGW(Diag,_EULERR_1);
      SymbLST_DB(Data,List,DB_SYMB_CHAN);
      Sort_LST(List,0,0);

      // --- Set any Defaults ---
      SetInfoSYS_DGW(Diag,_EULERR_0,DGW_TEXT,"EULERR","YCHAN");  // Latitude
      SetInfoSYS_DGW(Diag,_EULERR_1,DGW_TEXT,"EULERR","XCHAN");  // Longitude
      SetInfoSYS_DGW(Diag,_EULERR_2,DGW_TEXT,"EULERR","ANGLE");
      SetInfoSYS_DGW(Diag,_EULERR_3,DGW_TEXT,"EULERR","Y0");     // Latitude
      SetInfoSYS_DGW(Diag,_EULERR_4,DGW_TEXT,"EULERR","X0");     // Longitude

      // --- Run the Dialogue ---
      i = iRunDialogue_DGW(Diag);
      if (i == -1) Cancel_SYS();

      GetInfoSYS_DGW(Diag,_EULERR_0,DGW_TEXT,"EULERR","YCHAN");  // Latitude
      GetInfoSYS_DGW(Diag,_EULERR_1,DGW_TEXT,"EULERR","XCHAN");  // Longitude
      GetInfoSYS_DGW(Diag,_EULERR_2,DGW_TEXT,"EULERR","ANGLE");
      GetInfoSYS_DGW(Diag,_EULERR_3,DGW_TEXT,"EULERR","Y0");     // Latitude
      GetInfoSYS_DGW(Diag,_EULERR_4,DGW_TEXT,"EULERR","X0");     // Longitude

      // --- Destroy the Dialogue ---
      Destroy_DGW(Diag);

   }

   // --- Get Parameters ---
   GetString_SYS("EULERR","XCHAN",sXNewCh);
   GetString_SYS("EULERR","YCHAN",sYNewCh);
   rAngle = GetReal_SYS("EULERR","ANGLE");
   rXo    = GetReal_SYS("EULERR","X0");
   rYo    = GetReal_SYS("EULERR","Y0");

   // --- Do the Channels Exist ? ---
   if (!iExistChan_DB(Data,sXNewCh) ||
       !iExistChan_DB(Data,sYNewCh))
      Abort_SYS("X or Y channel does not exist.");

   // --- Get current X, Y channels ---
   GetXYZChan_DB(Data, DB_CHAN_X, sXCh);
   GetXYZChan_DB(Data, DB_CHAN_Y, sYCh);

   // --- Get names of backups to current X, Y channels ---
```

```
   Strcpy_STR(sXBackCh, "__");
   Strcat_STR(sXBackCh, sXCh);
   Strcpy_STR(sYBackCh, "__");
   Strcat_STR(sYBackCh, sYCh);

   // --- if output is current X and Y and no backups exist, make them ---
   if ( (iStrcmp_STR(sXNewCh,sXCh,STR_CASE_TOLERANT) == 0) &&
        (iStrcmp_STR(sYNewCh,sYCh,STR_CASE_TOLERANT) == 0) &&
        (!iExistChan_DB(Data,sXBackCh)) && (!iExistChan_DB(Data,sYBackCh)) )
   {
      UnLock_EDB(EData);
      iRunGX_SYS("xysave");
      Data = Lock_EDB(EData);
   }

   // --- Perform the rotation. Call the routine in MarkDev_.dll ---

   RotateXY_(Data,sXNewCh,sYNewCh,rXo,rYo,rAngle);

   UnLock_EDB(EData);

   // --- Done ---

}
```

EulerR.gxh

```
//-------------------------------------------------------------
// MarkDev_.gxh
//
// INCLUDE file for EulerR.gxc that specifies how to call
// the RotateXY_ subroutine in MarkDev_.dll (C++ source code
// MarkDev_.c)
//-------------------------------------------------------------

// these lines prevent this file from being included twice

#ifndef MARKDEV__DEFINED
#define MARKDEV__DEFINED

// RotateXY_   Add a base value to a named channel.

[MarkDev_]           // name of the DLL
void RotateXY_(
            DB,      // database
            string,  // X channel name (must exist, Long)
            string,  // Y channel name (must exist, Lat)
            real,    // Rotation origin Long
            real,    // Rotation origin Lat
            real);   // Rotation angle CCW
#endif
```

MarkDev_.C

```
//-------------------------------------------------------------------------
// MarkDev_.c
//
// C++ source code for MarkDev_.dll written for Microsoft Visual C++ .NET
//
//    RotateXY_      Rotate Lat (Y) and Long (X) about an Euler pole
//
// See MarkDev_.gxh for the GX prototype header that describe the calling
// parameters for this function
//-------------------------------------------------------------------------

// --- STANDARD INCLUDES ---
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// --- Required Includes from GX developer
// --- Check the paths to these files on your system
```

```
#define C_MICROSOFT
#include "..\include\gx_lib.h"
#include "..\include\gx_define.h"
#include "..\include\gx_extern.h"

// --- Added because I can't find Geosoft's version
#define rGeoDUMMY -1.0E32

// --- RotateXY_ ---

__declspec(dllexport)
void __cdecl RotateXY_(          // rotate Long/Lat co-ordinates
         GX_OBJECT_PTR pGeo,     // geosoft handle
         const long *phDB,       // database handle
         const char *pcXCh,      // channel name, X (Longitude)
         const char *pcYCh,      // channel name, Y (Latitude)
         const double *pdXo,     // Rotation origin X (Longitude)
         const double *pdYo,     // Rotation origin Y  (Latitude)
         const double *pdAngle)  // Rotation angle CCW

{
   GX_HANDLE hVVx = 0;   // uninitialized handles set to 0
   GX_HANDLE hVMx = 0;

   GX_HANDLE hVVy = 0;   // uninitialized handles set to 0
   GX_HANDLE hVMy = 0;

   long  hChanx,hChany,hLine;   // database symbols
   long  lVV;                   // number of VV elements
   short sGeoError = 0;         // error flag, 0 - no error
   double *pdx;                 // double pointer to X
   double *pdy;                 // double pointer to Y
   // rotation parameters in radians
   double rXo,rYo,rAngle, rRadians;
   // rotation matrix
   double rR11,rR12,rR13,rR21,rR22,rR23,rR31,rR32,rR33;
   // rotatiion pole in cartesian coordinates
   double rEx,rEy,rEz;
   // vector to rotate in cartesian coordinates
   double rAx,rAy,rAz;
   // rotated vector in cartesian coordinates
   double rAxr,rAyr,rAzr;

   long  l;                     // counter
   char  szErr[64];             // my error message

   // --- initially no error ---
   *szErr = '\0';

   // --- Get the X channel Channel handle --
   hChanx = FindSymb_DB(pGeo,phDB,pcXCh,_l(DB_SYMB_CHAN));
   GEO_ERROR;
   if (hChanx == -1) {

      // --- channel does not exist ---
      IStrcpy_STR(pGeo,szErr,pcXCh,_l(sizeof(szErr)));
      IStrcat_STR(pGeo,szErr," < channel does not exist.",_l(sizeof(szErr)));
      sGeoError = 1;
      goto GEO_EXIT;
   }

      // --- Get the Y channel Channel handle --
   hChany = FindSymb_DB(pGeo,phDB,pcYCh,_l(DB_SYMB_CHAN));
   GEO_ERROR;
   if (hChany == -1) {

      // --- channel does not exist ---
      IStrcpy_STR(pGeo,szErr,pcYCh,_l(sizeof(szErr)));
      IStrcat_STR(pGeo,szErr," < channel does not exist.",_l(sizeof(szErr)));
      sGeoError = 1;
      goto GEO_EXIT;
   }

   // --- Lock X Channel for read-write ---
   LockSymb_DB(pGeo,phDB,&hChanx,_l(DB_LOCK_READWRITE),_l(DB_WAIT_NONE));
   GEO_ERROR;

   // --- Lock Y Channel for read-write ---
```

```
LockSymb_DB(pGeo,phDB,&hChany,_l(DB_LOCK_READWRITE),_l(DB_WAIT_NONE));
GEO_ERROR;

// --- Create VVs to hold XY data arrays ---
hVVx = CreateExt_VV(pGeo,_l(GS_DOUBLE),_l(0));
GEO_ERROR;
hVVy = CreateExt_VV(pGeo,_l(GS_DOUBLE),_l(0));
GEO_ERROR;

// --- get real VMs to to hold the data in memory ---
hVMx = Create_VM(pGeo,_l(GS_REAL),_l(0));
GEO_ERROR;
hVMy = Create_VM(pGeo,_l(GS_REAL),_l(0));
GEO_ERROR;

   //------------------------------------------------------------------------
   //------------------------------------------------------------------------
   //------------------------------------------------------------------------

   // Convert rotation parameters to radians
   rRadians = 3.141592654/180.0;
   rXo = *pdXo * rRadians;
   rYo = *pdYo * rRadians;
   rAngle = *pdAngle * rRadians;

   // Convert to cartesian coordinates
   rEx = cos(rYo) * cos(rXo);
   rEy = cos(rYo) * sin(rXo);
   rEz = sin(rYo);

   // Build rotation matrix
   rR11 = rEx * rEx * (1 - cos(rAngle)) +       cos(rAngle);
   rR12 = rEx * rEy * (1 - cos(rAngle)) - rEz * sin(rAngle);
   rR13 = rEx * rEz * (1 - cos(rAngle)) + rEy * sin(rAngle);

   rR21 = rEy * rEx * (1 - cos(rAngle)) + rEz * sin(rAngle);
   rR22 = rEy * rEy * (1 - cos(rAngle)) +       cos(rAngle);
   rR23 = rEy * rEz * (1 - cos(rAngle)) - rEx * sin(rAngle);

   rR31 = rEz * rEx * (1 - cos(rAngle)) - rEy * sin(rAngle);
   rR32 = rEz * rEy * (1 - cos(rAngle)) + rEx * sin(rAngle);
   rR33 = rEz * rEz * (1 - cos(rAngle)) +       cos(rAngle);

   //------------------------------------------------------------------------
   //------------------------------------------------------------------------
   //------------------------------------------------------------------------

// --- Go through all selected Lines ---
hLine = FirstSelLine_DB(pGeo,phDB);
GEO_ERROR;

do {

   // --- break if line is not a valid line ---
   if (!iIsLineValid_DB(pGeo,phDB,&hLine)) break;
   GEO_ERROR;

   // --- get VV data ---
      GetChanVV_DB(pGeo,phDB,&hLine,&hChanx,&hVVx);
      GEO_ERROR;
      GetChanVV_DB(pGeo,phDB,&hLine,&hChany,&hVVy);
      GEO_ERROR;

   // ---
   // Get data VM from the VV.  This will re-size the VM to
   // hold all the data in the VV.
   // ---
   CopyVVtoVM_VV(pGeo,&hVMx,&hVVx);
   GEO_ERROR;
   CopyVVtoVM_VV(pGeo,&hVMy,&hVVy);
   GEO_ERROR;

   // --- now get a pointer to the data ---
   pdx = (double*)GetPtrVM_GEO(pGeo,&hVMx);
   GEO_ERROR;
   pdy = (double*)GetPtrVM_GEO(pGeo,&hVMy);
   GEO_ERROR;
```

```
   //-------------------------------------------------------------------------
   //                              MAIN MATH CODE
   //-------------------------------------------------------------------------
   lVV = iLength_VV(pGeo,&hVVx);
   GEO_ERROR;

   //-------------------------------------------------------------------------
   //-------------------------------------------------------------------------
   //-------------------------------------------------------------------------

   // Loop through the elements

   for(l=0;l<lVV;l++,pdx++,pdy++) {

      if(*pdx != rGeoDUMMY && *pdy != rGeoDUMMY) {
        // To radians
        *pdx = *pdx * rRadians;
        *pdy = *pdy * rRadians;
         // To cartesian
         rAx = cos(*pdy) * cos(*pdx);
        rAy = cos(*pdy) * sin(*pdx);
        rAz = sin(*pdy);
        // Rotate
        rAxr = rR11 * rAx + rR12 * rAy + rR13 * rAz;
        rAyr = rR21 * rAx + rR22 * rAy + rR23 * rAz;
        rAzr = rR31 * rAx + rR32 * rAy + rR33 * rAz;
         // Back to Lat/Long degrees - code that makes long. between + and - 180
        if(rAxr == 0.0)
           if (rAyr < 0.0) *pdx = -90.0;
           else
              *pdx = 90.0;
           else
              if (rAxr < 0.0)
                 if (rAyr < 0.0) *pdx = -180.0 + (atan(rAyr/rAxr) / rRadians);
                     else
                        *pdx = (atan(rAyr/rAxr) / rRadians) + 180;
              else
                 *pdx = atan(rAyr/rAxr) / rRadians;
           *pdy = asin(rAzr) / rRadians;
      }    // if no dummies
   } // for loop

   //-------------------------------------------------------------------------
   //-------------------------------------------------------------------------
   //-------------------------------------------------------------------------

   // --- Put the VM data back in the VV ---
   CopyVMtoVV_VV(pGeo,&hVVx,&hVMx);
   GEO_ERROR;
   CopyVMtoVV_VV(pGeo,&hVVy,&hVMy);
   GEO_ERROR;

   // --- Write data back to the database ---
   PutChanVV_DB(pGeo,phDB,&hLine,&hChanx,&hVVx);
   GEO_ERROR;
   PutChanVV_DB(pGeo,phDB,&hLine,&hChany,&hVVy);
   GEO_ERROR;

   // --- Advance to Next Line ---
   hLine = NextSelLine_DB(pGeo,phDB,_l(hLine));
   GEO_ERROR;

} while (1);

// --- Cleanup ---
UnLockSymb_DB(pGeo,phDB,&hChanx);
GEO_ERROR;
UnLockSymb_DB(pGeo,phDB,&hChany);
GEO_ERROR;

// ---
// Destroy any handles that were created.  This is not really necessary since
// all handles will be destroyed when a GX ends.
// ---

if (hVMx) Destroy_VM(pGeo,&hVMx);
GEO_ERROR;
if (hVMy) Destroy_VM(pGeo,&hVMy);
```

```
   GEO_ERROR;

   GEO_ERROR;


   // --- Done ---
GEO_EXIT:

   // --- terminate the GX if an error occured ---
   if (sGeoError != 0) {
      if (*szErr)
         _Abort_SYS(pGeo,szErr);
      else
         _Abort_SYS(pGeo,"An error occured in RotateXY_.");
   }

   return;

}
```