Two-Dimensional Finite-Element Gridding on a Spherical Planet:

OrbWin MANUAL

Peter Bird, Professor Emeritus
Department of Earth, Planetary, and Space Sciences
University of California, Los Angeles
http://PeterBird.name
pbird@epss.ucla.edu
OrbWin version 1.2; manual version 14 March 2019

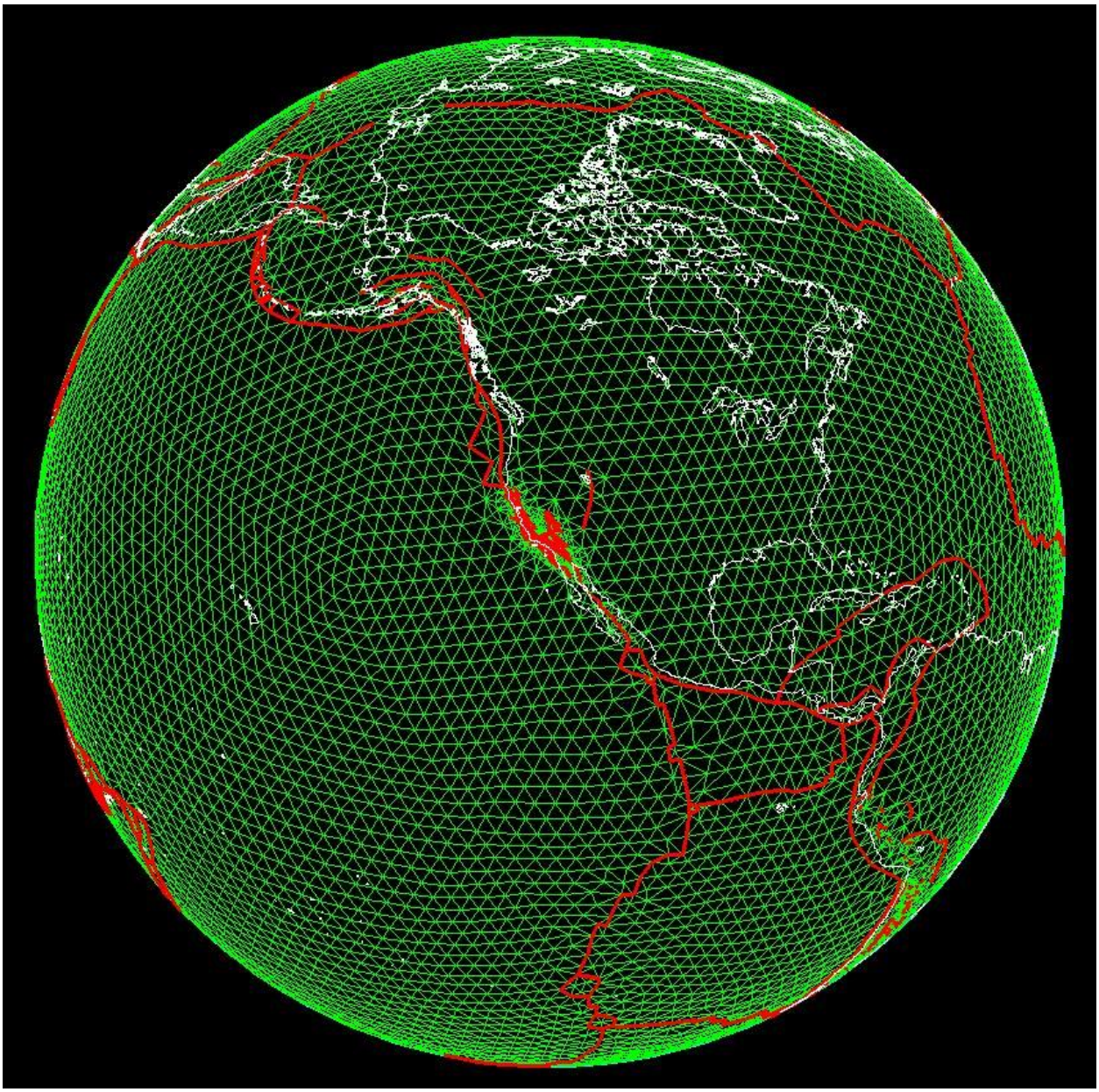


TABLE OF CONTENTS:

I. OVERVIEW

Over the last 41 years, I have developed 3 different finite-element programs (***Shells***, ***NeoKinema***, & ***Restore***) for simulating motion and deformation of plates of faulted lithosphere on the surface of a spherical planet. One common feature of these codes is that they all use two-dimensional grids of finite elements, allowing for rapid computations on ordinary desktop computers with limited memory. Another is that now they all use the spherical-triangle element invented by *Kong & Bird* [1995]. ***OrbWin*** is the free, open-source application for the Windows 8+ operating system that permits you to create and edit such finite-element grid files. All operations are user-directed from one top-level group of menus of commands. Most grid components can be added, selected, moved, or deleted with the mouse cursor, and the update of the display is almost immediate. Some global tests of grid topology are provided to catch common problems and guide you to the specific region that needs attention. This manual also gives brief introductions to my other utility programs that you may need to digitize fault traces, renumber nodes, and make publication-quality maps of grids, their nodal data, and F-E model output.

II. BACKGROUND CONCEPTS

The plate-tectonic conceptual revolution (paradigm shift) beginning in the 1960's has provided us with first-order descriptions of the relative motions of the largest plates of lithosphere since Late Cretaceous-Tertiary times. Some prominent models of present-day relative rotation rates of plates include RM2 of *Minster & Jordan* [1978] and PB2002 of *Bird* [2003]. These models make predictions of fault slip-rates along plate boundaries which are useful starting points for seismic hazard estimation. However, all geologists and seismologists know that there are many other active faults which concentrate ongoing deformation in the interiors of plates as well. There may even be widely-distributed permanent strain that is either not accommodated on faults at all, or else is accommodated on a fractal array of innumerable faults of very small offset. Aided by satellite geodesy (GPS = Global Positioning System, and InSAR = Interferometric Synthetic Aperture Radar), our science is now advancing to a more sophisticated understanding of such plate motions and deformations, in which no region can be assumed to be exactly rigid over geologic times.

Computer modeling is essential to investigate and understand these processes, because we cannot conduct plate-scale experiments in the real world (with control of boundary tractions, initial conditions, *etc*.). The finite-element (F-E) method is a natural formalism for computer modeling because it allows for the representation of complex fault traces and plate boundaries. Because my own career began at a time when most computers used 16-bit or 32-bit memory addressing and contained only limited amounts of fast RAM (*e.g.*, 1 MB to 2 GB), I have concentrated on developing efficient 2-D F-E models that can give approximate answers to geologic and geophysical questions with modest computing resources. (Doubtless, such methods will eventually be replaced by more precise models using 3-D F-E grids; however, this transition will be slow, and in some problems the new insights may be minor.)

II.A. Types of finite-element (F-E) modeling supported

My web site (http://peterbird.name) currently offers 3 major F-E programs for modeling the motion and deformation of 2-D plates of faulted lithosphere on the surface of a spherical planet. (Each of these programs is free and open-source, in Fortran. Executable versions are provided for the Windows O/S.)

Shells [*Kong & Bird*, 1995; *Bird*, 1999] is a dynamic (or "forward") modeling program which takes inputs including: lithospheric structure (crustal thicknesses, mantle-lithosphere thicknesses, and geotherms), fault traces and dips, permanent-strain rheologies of rocks (with separate frictional and dislocation-creep deformation mechanisms, and different constants for crust, mantle, and active faults), and lateral

(edge) boundary conditions on either long-term velocities or traction anomaly vectors. It solves for 2-D stress equilibrium (of vertically-integrated stress anomaly tensors) by a "weak" (Galerkin) method. The outputs from each calculation include: long-term velocities of nodes, long-term (permanent) strain-rates in elements, stress anomalies as a function of depth, and slip-rates of any modeled faults (together with their rakes). Note that geodetic and geologic data on relative motions, fault offset rates, and principal stress directions are NOT included among the inputs; however, these data are very useful for scoring and judging the relative realism of competing ***Shells*** models. *Bird et al.* [2008] used ***Shells*** to create the "Earth5" global model of present plate motions and assess the likely sources of "plate-driving forces". An image of their F-E grid is on the cover of this manual. The principal weaknesses of ***Shells*** models include: (a) an assumption of local isostasy, which prevents modeling plate-bending near subduction zones; (b) an assumption of steady-state geotherms, which prevents accurate modeling of tectonics of oceanic spreading ridges; and (c) the problem that computed velocity fields often agree very poorly with geodetic data, while computed fault slip-rates may be outside the ranges allowed by geologic data. Thus, it can be challenging to use ***Shells*** models for seismic hazard estimation.

NeoKinema is a kinematic (or "inverse") modeling program which takes inputs of: fault traces and dips, geologic long-term offset rates (with uncertainties), interseismic geodetic velocities (with uncertainties), and azimuths of most-compressive horizontal principal stress. It uses the weighted least-squares method to fit these data as well as it can, guided by 3 user-selected weighting factors. It predicts both long-term-average and interseismic velocities of nodes, both permanent and interseismic strain-rates in elements, long-term offset rates and rakes of faults, and principal stress directions (but not intensities). When the misfits of the best ***NeoKinema*** model are judged to be acceptably small, these model predictions can have direct applications in seismic hazard estimation. *Bird* [2009] presented NeoKinema models of neotectonics in the western United States, and an updated version of that model was incorporated into the Unified California Earthquake Rupture Forecast version 3 (UCERF3) [*Field et al.*, 2013] and also into the 2014 update of the National Seismic Hazard Map by the U.S. Geological Survey [*Petersen et al.*, 2014a, b]. The weaknesses of ***NeoKinema*** models include: (a) lack of continuum-mechanical physics, such as stress-equilibrium and rheology; (b) failure to predict the intensity of stress; and (c) lack of ability to compute simulations in the geologic past (when geodetic data were not available).

Restore is a kinematic F-E modeling program that allows for finite strains and large displacements over geologic times. Like *NeoKinema*, it reads in fault traces with their dips and offset histories (including uncertainties). It also accepts geologic data on azimuths of the most-compressive horizontal principal stresses through time [*e.g.*, *Bird*, 2002]. Unlike *NeoKinema*, ***Restore*** uses paleomagnetic inclination and declination anomalies (with uncertainties) to constrain North-South displacements and vertical-axis rotations through time. Published shortening/extension of balanced cross-sections during geologic history can provide additional constraints. Model output predictions include: motions of the F-E grid (and any basemap features attached to it) through time; finite-strains of elements through time, vertical-axis rotations through time, and cumulative fault slip through time. Natural applications of these models include: (1) educational simulations; (2) hypothesis-free investigation of the kinematic implications of tectonic and paleomagnetic data, and (3) automated production of rough palinspastic geologic maps. Retrospective predictions of past crustal thicknesses and elevations would be natural extensions. *Bird* [1998b] used this method to reconstruct the Laramide orogeny that formed the Rocky Mountains in the central United States, and also the subsequent Basin/Range extension in Utah and Colorado. One weakness of ***Restore*** models is that they do not (directly) reveal the stress sources or stress intensities that caused past deformation. (However, these may sometimes be inferred from deformation patterns.)

Each these 3 programs requires a ".FEG" (Finite-Element Grid) file among its inputs.

II.B. Finite-Element Grid (.FEG) file format

"**.FEG**" is the filename extension I have always used for **F**inite-**E**lement **G**rid files; all my programs know and expect this extension. (However, be aware that web browsers like Edge and Chrome do not "know" this filename extension. Thus, it may be necessary to add a second filename extension of ".txt" before these files can be passed across the Internet—expect browser errors and perhaps antivirus alerts if you omit this step!)

FEG files consist of plain ASCII letters and numbers, which can be read and edited in any elementary text-editor (*e.g.*, ***NotePad*** of Windows; or, I like the shareware ***EditPad Pro***). Each line is typically terminated by invisible CR and LF bytes (as in the DOS/Windows tradition), although lines terminated by only one invisible byte (as in the Unix tradition) may work as well.

FEG files are always read by FORMAT-free Fortran ("READ(1, *) …") statements, so the exact placement of the numbers within each row of the file is not important. You just need the correct ordering of items, and the correct item count. Items must be separated by one or more spaces. (Do not use TAB characters, as some versions of FORMAT-free READ do not handle these properly.)

The general structure of an FEG has 7 parts, each beginning on a fresh line:

1. Title line (up to 80 ASCII characters; anything you wish).

2. Number of nodes (INTEGER(4) :: numNod). Also, for historical reasons, this line contains 3 more integers and a logical switch (INTEGER(4) :: nRealN, nFakeN, n1000; LOGICAL :: brief). The best settings are: nRealN = numNod; nFakeN = 0; n1000 > numNod; brief = .FALSE. Thus, for example, the second line of sample file Earth5R.feg is:
   16008 16008 0 100000 T (numNod, nRealN, nFakeN, n1000, brief)

3. List of nodes. Each line has: node#, longitude, latitude. In units of degrees; E and N are positive.
   In FEG files intended for use with ***Shells***, there will be either 4 or 6 nodal data following: elevation, heat-flow, crustal thickness, mantle-lithosphere thickness[, density anomaly of chemical origin, non-steady curvature of the geotherm]. Use SI units: m, W/m^2, m, m[, kg/m^3, $°K/m^2$]. Zero values appear if these quantities have not yet been estimated.
   In FEG files intended for use with ***NeoKinema***, the latitude may (optionally) be followed by a single real-number datum: the nodal value of strain-rate compliance parameter μ (in s^{-1}).

4. Number of triangular continuum elements.

5. List of elements. Each line has: element#, cornerNode1#, cornerNode2#, cornerNode3#.
   All positive integers. Corner node indices must be listed in counterclockwise order.
   In FEG files produced by ***Restore3***, there will be 3 additional floating-point numbers following the definition of each element; these are measures of the modeled finite-strain in 2-D. In FEG files intended for use with ***Shells_v5.0+***, there may (optionally) be notations like "LR2" or "LR1970" following an element definition; these indicate that the element has a special Lithospheric Rheology (defined in another, extra input file).

6. Number of fault elements. (In FEG grids for ***NeoKinema*** or ***Restore***, this will be zero.)

7. List of fault elements (if any). Each line has: fault#, cornerNode1#, cornerNode2#, cornerNode3#, cornerNode4#, dip1, dip2, net-offset. {If net-offset is not known, it may be zero.} In FEG files

intended for use with ***Shells_v5.0+***, there may (optionally) be notations like "LR2" or "LR1970" following any fault element definition; these indicate that the fault has a special Lithospheric Rheology (defined in another, extra input file).

Constructing an FEG file "manually" in a text-editor is possible. (I used to do it!) But, it is very laborious and subject to a number of possible errors. To minimize labor and problems, I suggest that you create (and/or edit) your FEG files in my program ***OrbWin***.

II.C. Rules and violations of F-E grid topology

A finished F-E grid (which is ready to use in a scientific calculation with ***Shells***, ***NeoKinema***, or ***Restore***) must satisfy certain minimum criteria:

- Every node must be connected to at least one (triangular continuum, or linear fault) element.
- The nodes defining each triangular element must be specified in counterclockwise order.
- The area of each triangular element must be positive. (Zero and negative areas are forbidden.)
- The interior area of a triangular element may not overlap the area of another element.
- The perimeter of the F-E grid must be "simply-connected"; that is, there may not be any outlying islands or interior holes in the grid. (Even linear holes with zero area are forbidden.)

Naturally, an unfinished grid which is being edited with ***OrbWin*** will often violate some of these rules. However, such problems must eventually be detected, isolated, and fixed. For this reason, ***OrbWin*** includes two validation tools (**Perimeter/Area Test**, and **View Gaps/Overlaps**, described in section III.G) which you should run at the end of editing.

II.D. History of development of ***OrbWin***

DrawGrid was written by Peter Bird in 1990-2000, to produce FEG files for then-current flat-Earth dynamic F-E programs ***Faults*** (crust-only) and ***Plates*** (whole-lithosphere). In these codes, the triangular elements have 6 nodes instead of 3, allowing them to be bent for better representation of fault traces. Fault elements also have 6 nodes, and can be bent. ***DrawGrid*** was written in the Microsoft Basic 7 language for the DOS operating system. This program emulated some aspects of Windows, including simulated multithreading and a basic callback structure. However, it suffered from the limits of 16-bit memory addressing (not supported by Windows 10+), 8-byte filenames, *etc.*

TriGrid (Peter Bird, 1994-2000) is similar to ***DrawGrid*** except that the triangular elements are simple 3-node triangles with straight sides, and there are no fault elements. This program was intended to give undergraduates an entry-level experience in 2-D (vertical-plane) F-E modeling of the Darcy groundwater-flow equation, using my program ***Darcy***. It was also in Microsoft Basic 7, for DOS, so it also uses 16-bit memory addresses, and is not supported by Windows 10+.

OrbWeave[r] (Peter Bird, 1993-2001) is based on ***TriGrid***, but with the coordinate system switched from flat-Earth Cartesian *(x, y)* to round-Earth spherical *(longitude, latitude)*. Fault elements (4-nodes, arc-of-great-circle) are added. This was the first grid-editor for ***Shells***. It was also in Microsoft Basic 7, for DOS, so it also uses 16-bit memory addressing which is not supported by Windows 10+.

If there should ever be a reason to use any of these old codes, it might be possible to run them under modern versions of Windows by using the shareware virtual-computer O/S ***DosBox***.

OrbWin version 1.0 (Zhen Liu, 2004-2005) was a translation of ***OrbWeave[r]*** to the Fortran 90 language and to the Win32 Application Programming Interface (API). This removed the old 64-KB limit on working memory (in any single array or program unit). It also yielded a more professional look-and-feel. The

cost was a complex multi-file source code, plus additional resource files, that are difficult for most scientists to comprehend or modify. ***OrbWin*** version 1.1 (Peter Bird, 2010) fixed a few bugs, but not all of them. These versions were both compiled with Compaq/Microsoft/Digital Visual Fortran 6.6.a release of 2000 (now typically referred to as "Digital Fortran") and used fairly successfully on Windows versions from Windows NT through Windows 7. As 64-bit versions of Windows became widespread, I began to hear reports that this code would no longer run on some 64-bit ("x64") Windows computers. This seemed to depend on the brand of the CPU chip. Then, when Windows 10 was released, we found that ***OrbWin*** 1.1 would start, but would not work properly, even on high-end Intel CPU chips.

OrbWin version 1.2 (Peter Bird, 2016; 2018; 2019) is a debugged update compiled with Intel Visual Fortran (specifically, Intel Parallel Studio XE 2013 with Microsoft Visual Studio 2010 Shell). Because this compiler release dates from the time of the Windows 8 $\rightarrow$ 8.1 transition, I believe ***OrbWin*** 1.2 should work properly with either variant of this O/S. Also, this recompilation and testing was done on a 64-bit Dell computer running Windows 10, with no new difficulties. I would be interested to hear about your experiences (either good or bad) of using ***OrbWin*** 1.2 with versions of (Win32 API) Windows prior to Windows 8 (Windows 95, NT, 98, 2000, XP, Vista, or 7). The present release is provided as Fortran 90 source code (.f90 files) with necessary resource files (.h, .fd, .rc, .ico), and with executables (.exe) for both 32-bit Windows ("IA-32") computers (OrbWin-Win32seq.exe), and for 64-bit Windows ("x64") computers (OrbWin-Win64seq.exe). The characters "seq" in the name of the executable are a sign that the program is sequential, and does not do any parallel processing (although there is a tiny bit of multithreading). Therefore, no .DLL file for OpenMP is needed when running ***OrbWin*** 1.2.

There has never been any "installer" program for any version of my grid editor. Just download and place the executable (.exe) file wherever you like on your hard drive, and start it. Of course, you must have a mouse or equivalent pointing device already installed. This manual file also resides on my web page, at http://peterbird.name/oldFTP/OrbWin/Orbwin_manual.pdf.

II.E. DIGitized-lines (.DIG) file format

When working in ***OrbWin***, you typically want to have a basemap file loaded, in order to place grid edges, nodes, and faults in the proper positions. The only kind of basemap that ***OrbWin*** can load is one in my .DIG format.

Output from my (very old) program ***Digitize*** is always in .DIG format, as is output from my (updated) program ***Projector***. This format is read and used by all my graphical utility programs. It is an extremely simple and flexible format. If you prefer to use a more modern, commercial digitizing program (from a company like Golden Software or ESRI), please be sure to select an application that has the option to "Export / Save As Type: SHP ESRI Shapefile (*.SHP)". Then, you can use my new conversion utility program ***SHP_toFrom_xyDIG*** to convert your shapefile (and associated dBase .DBF file) into a file in my .DIG format.

What makes the .DIG format so simple is that it only represents lines: straight line segments defined by two end-points, and jagged multi-point lines, called polylines. It cannot represent smooth curves, areas, or text. Also, no graphical attributes (color, width, continuity, ...) are stored; there are only 1~4 (optional) text headers for each polyline. Therefore, a segment is just a list of *(x, y)* or *(longitude, latitude)* coordinate pairs followed by a standard end marker. There is no "hard" limit on the number of points in each polyline. It is legal (but not very useful) to have only one, or zero, points in a polyline.

There is no limit to the number of polylines in a file. It is extremely easy to merge multiple .DIG files; you simply concatenate them. (See V.A.6. ***COPY.***)

If you have used text labels to identify polylines, then you can use any simple ASCII text editor to edit the files and delete or replace polylines.

Scientific notation (exponential format) is used for the coordinates so that values never go off-scale. Six significant digits are used in order to give 10 m resolution on the Earth. (There would be no point in attempting higher resolution, because my programs treat planets as spheres and ignore their rotational ellipticity.)

If *(longitude, latitude)* format is used, note these conventions: All values are in degrees. North latitude (measured from the equator) is positive, and South latitude is negative. East longitude (measured from the Greenwich meridian) is positive. West longitudes are usually negative, but they can also be converted to positive East longitudes in the range +180 to +360.

After struggling with the general problem of how to distinguish a text label from a pair of numbers (and do it identically in 3 programming languages), I hit upon a simple, though inefficient rule: Text labels should start in column 1. Number pairs should begin in column 2 with a leading sign (+ or -). If you depart from these rules, some of my programs may be able to interpret your .DIG file, but others may fail.

If you code in Fortran, it is easiest to write the number pairs with:

```
FORMAT (1X, SP, 1P, E12.5, ',', E12.5)  ! in Fortran 77, or
FORMAT (1X, SP, ES12.5, ',', ES12.5)    ! in Fortran 90.
```

The following text-box displays the first 25 lines of a typical .DIG file. (This one consists of digitized fault traces in degree units.) Note the required "***" end-marker beginning in column 1, which I usually expand to “*** end of polyline ***” to make its significance more clear to users. Lines 1 and 22 are text labels to identify the segments that follow them. Such labels are optional.

```
F1302R
 -1.05289E+02,+3.46860E+01
 -1.05298E+02,+3.47768E+01
 -1.05288E+02,+3.48878E+01
 -1.05286E+02,+3.50316E+01
 -1.05286E+02,+3.51756E+01
 -1.05281E+02,+3.52886E+01
 -1.05275E+02,+3.53434E+01
 -1.05286E+02,+3.54419E+01
 -1.05260E+02,+3.55517E+01
 -1.05270E+02,+3.55918E+01
 -1.05289E+02,+3.56336E+01
 -1.05267E+02,+3.56759E+01
 -1.05245E+02,+3.57181E+01
 -1.05237E+02,+3.57895E+01
 -1.05261E+02,+3.58552E+01
 -1.05235E+02,+3.59543E+01
 -1.05206E+02,+3.60516E+01
 -1.05191E+02,+3.60975E+01
 -1.05201E+02,+3.61451E+01
*** end of polyline ***
F0917R
 -1.06924E+02,+3.42904E+01
 -1.06891E+02,+3.43457E+01
 -1.06822E+02,+3.44201E+01
  [...listing truncated....]
```

III. OrbWin COMMAND REFERENCE

III.A. Decision-point dialog

The first thing you will see, after ***OrbWin*** starts, is a dialog that asks you to choose the F-E program you will be editing an .FEG file for. Please consider this question carefully, and consult section II.A. of this manual if you want to read more about the choices available. Then, enter integer 1, 2, or 3 in the white box provided, and press **Enter** or **OK**. (This choice cannot be changed after starting ***OrbWin***.)

III.B. Specify-maximum-array-sizes dialog

The second thing you will see is a dialog that asks you to select the maximum numbers of: nodes, (continuum triangle) elements, and (linear) fault elements. Your choices are used to ALLOCATE Fortran arrays to hold the necessary data during the current editing session. (These choices cannot be changed after starting ***OrbWin***.) Note that suggested values are provided, which should be plenty large enough for most beginning users. With these suggested values, ***OrbWin*** will occupy about 90 MB of computer memory. If you should ever run into a program crash after entering *really large* numbers (because of the 2-MB user memory limit in 32-bit Windows), then you can always switch to a 64-bit Windows computer (with more memory) and run the 64-bit version of ***OrbWin***.

III.C. Introduction to the main window

After these 2 dialogs, you will see a black screen with a blue circle. This circle represents the planet. Nothing else is visible because initially no basemap is loaded, and no F-E grid is loaded. If you move the mouse cursor across the planet, you will see cursor *(longitude, latitude)* coordinates displayed in the 3rd compartment of the status-bar at the bottom of the ***OrbWin*** frame window.

The map projection of ***OrbWin*** is always orthographic. North is always toward the top of the screen. These two things cannot be changed. However, you will be able to select your desired window position (viewpoint above the planet) and your desired zoom factor with commands described below.

OrbWin draws its graphics into a "virtual window" which is the same size as your computer monitor. Because of the space taken by Windows overhead (window titles, command menus, status bars, & task bars), the visible graphical window is smaller. To see parts of the virtual window which are currently hidden, use the vertical and horizontal scroll bars. However, scrolling is inconvenient and tedious; thus, ***OrbWin*** should normally be run with its frame window maximized. This will also allow you to read the useful status messages that appear in its lower status-bar display.

There is a group of command menus near the upper left corner of the frame window. Please be warned that there is no such thing as an "Undo" command! (Just save your work from time to time.)

III.D. **File** commands (Alt+F)

You can open this menu by clicking on it, or with the accelerator key combination **Alt+F**. (If you change your mind, you can close it with the **Esc** key, or by clicking the mouse anywhere outside the menu.)

Once a menu is open, you can select any displayed command by either (1) left-clicking the mouse on it; or (2) moving the highlight down to it (with your keyboard arrow keys) and pressing **Enter**. Paradoxically, you cannot select a command using shortcut accelerators like "Ctrl+G" while the menu is open!?!?! These only work when *NO* command menu is open. (However, with experience you may begin to memorize these shortcuts, and use them to work faster.)

If you forget which command is currently selected, this is indicated in two places: (1) The command is checked in submenu of the top-level menu; and (2) the command name is written in the left-most compartment of the lower status bar.

III.D.1. Load**G**rid (Ctrl+G)

This command opens a standard Windows file-tree display, allowing you to select any existing .FEG file that you may wish to open. Only .FEG files (and folder names, and shortcuts) will be displayed. If you cannot find an .FEG file that you know to be on your computer, then perhaps it has a second filename extension, such as ".FEG.TXT"? (Always rename files to remove any 2nd filename extension of ".TXT" after downloading any .FEG, .DIG, or .GRD file from my web site.)

Once the .FEG file is chosen, it will be read into memory and displayed. The initial window position (view point) coordinates are automatically selected to be in the center of the grid. The current .FEG filename (and its path) are displayed in the top of the ***OrbWin*** frame window. Within the grid, each node will be shown as a small blue circle, and each (continuum triangle) element will be outlined with green lines, and any fault elements (in a ***Shells***-type .FEG) will be shown with wider red lines. Each element will also have a small green-triangle icon at its center. These graphical options can be changed; see III.F.6. **SetColors.**

III.D.2. Load**B**ase (Ctrl+B)

This command opens a standard Windows file-tree display, allowing you to select any .DIG file that you may wish to display as a basemap. Only .DIG files (and folder names, and shortcuts) will be displayed. Only .DIG files in spherical (*longitude, latitude*) coordinates can be used in OrbWin; convert any Cartesian (*x, y*) .DIG files with my utility ***Projector***. If you cannot find some .DIG file that you know to be on your computer, then perhaps it has a second filename extension, such as ".DIG.TXT"? (Always rename files to remove any 2nd filename extension of ".TXT" after downloading any .DIG, .FEG, or .GRD file from my web site.)

Once the .DIG file is chosen, it will be read into memory and displayed. If no .FEG file has been loaded (or created) yet, then the initial window position (view point) coordinates are automatically selected to be in the center of the basemap. However, if any .FEG file is loaded or in-progress, the viewpoint will *not* move; therefore, it is theoretically possible that your basemap *could* be on the other side of the planet, and not initially visible. (To move the viewpoint, see either III.F.4. **WindowPosition** or III.F.2. **ZoomInOut**.) The initial line-color for a basemap is white, but this graphical option can be changed; see III.F.6. **SetColors.**

III.D.3. Sa**v**eGrid (Ctrl+V)

This command will pop open a dialog showing you the contents of the first line of the .FEG file, and inviting you to update this header line (with a new date? perhaps other memos?).

SaveGrid then opens a standard Windows file-tree display, allowing you to save the current .FEG file anywhere on your computer. Only .FEG filenames (and folder names, and shortcuts) will be displayed. *If you choose an existing filename, that file will be overwritten!*

You can specify a new file name by typing it in the white "File name:" box. Windows will allow you to specify a file name that contains spaces. However, this is not a good idea. Many of my programs (*e.g.*, ***NeoKinema***) will not be able to open any .FEG file whose name contains spaces. Also, such files are quite difficult to transport across the WWW. Therefore, please choose a file

name without spaces in it. You can use "camelCaps" (*e.g.,* "myVeryBestGrid.feg") or you can use underscore characters (*e.g.*, "my_very_best_grid.feg"). Personally, I often like to include the date in a file name; if you use "20160917" to represent "17 September 2016" then your files will be automatically sorted by date in most file-tree displays.

III.D.4. ClearGrid

This command will remove the existing F-E grid from computer memory (and the display). If the current F-E grid has not been edited since it was loaded, you can do this with one click. However, if your F-E grid has been edited, you will be reminded that you may want to save it before clearing (see III.D.3. **SaveGrid** above).

ClearGrid may be a necessary first step if you have any F-E grid currently loaded, and you wish to either (1) load a different existing grid with III.D.1. **LoadGrid**; or, (2) create a new global grid with III.G.1. **GlobalGrid**.

III.D.5. ClearBase

This command will remove the basemap (.DIG file) from computer memory (and the display).

ClearBase may be a necessary first step if you have a .DIG basemap currently loaded, and you want to display a different basemap.

III.D.6. Print

This command will copy the current virtual graphical window to a printer, in bitmap format. Note that you will get the entire virtual window, not just the part currently visible. If the current window-height is 2.02 radii, this will print an entire hemisphere of the planet. No Windows overhead features (window titles, command menus, scroll bars, status bars, or task bars) will be included. If you use the Windows Print Setup dialog to select a virtual printer like "Microsoft Print to PDF", you can get this graphical output as a one-page PDF file, rather than as ink-on-paper. (If you *ARE* printing on paper, then consider using III.F.6. **SetColors** to change the background color from black to white, and the basemap color from white to black.) Frankly, much better maps are available through my other programs (see V.D. **Making maps with graphical post-processing programs**), but this command may occasionally be convenient for a quick-and-dirty "working copy."

III.D.7. E**x**it (Ctrl+X)

This command closes ***OrbWin***. If you have created a new F-E grid, or edited an existing one, then you will be reminded that you should consider saving it first (see III.D.3. **SaveGrid)**. There is no need to save your basemap .DIG file, because ***OrbWin*** never modifies these files.

III.E. **E**dit commands (Alt+E)

III.E.1. Add/Drop **N**ode (Ctrl+N)

Add new nodes by left-clicking with the mouse. The index number of the new node will be displayed in the right-most compartment of the lower status bar.

Nodes should *only* be added outside the existing grid. If a node is created inside an existing element, it will usually either: (a) end up as an unconnected "orphan"; or (b) cause overlap conflicts between triangular elements. Either of these would be a topological defect in the grid.

Drop (delete) existing nodes by right-clicking with the mouse. You will not be allowed to drop a node which is connected into a triangular element (or a fault element); ***OrbWin*** will merely "chirp" as a warning message. (You must delete any connected element(s) first, with III.E.3. **Add/Delete Element**.)

III.E.2. **A**djustNode (Ctrl+A)

This command allows you to adjust the position of any existing node by left-clicking on it, holding the left mouse button down, and dragging. Any connected triangular elements (and fault(s), if any) will be dragged along with it.

If the adjusted node is part of triangular element(s), and if you drag it too far, then the central icon of one or more triangular elements may change from a little triangle to a little "X". This is a warning that this element has been deformed too much, and that it now has a negative area. In such cases, either use a different/smaller adjustment to the node's position, or delete and then redefine any bad elements with the following command…

III.E.3. Add/Delete **E**lement (Ctrl+E)

This command allows you to add and/or delete triangular continuum elements.

Add an element by clicking on 3 corner nodes in counterclockwise order. Completion of each element is indicated when ***OrbWin*** draws its outline, and places an element icon (small green triangle) at its center. If you see a small "X" then the element has been defined in the incorrect (clockwise) order, and is defective. (Delete it, and then re-enter it correctly.)

Try to avoid defining any new element so that it overlaps existing elements. This would be a topological defect in the grid. (It can be detected and located with Tools listed below.)

You will NOT be allowed to use any node which is part of a linear fault element (red heavy line, in a ***Shells***-type F-E grid). ***OrbWin*** will merely chirp. This is because there are actually multiple nodes sharing the same position at each end of a fault element, and your selection would be ambiguous. The solution is to **Heal** the fault (with the following command III.E.4. **Cut/Heal Fault**), then form your new element(s), and then finally re-**Cut** the fault.

Delete any element by right-clicking the mouse on the icon at its center (either triangle or "X"). You will NOT be allowed to delete an element which is connected to a linear fault element (red heavy line, in a Shells-type F-E grid). ***OrbWin*** will merely chirp. The solution is to Heal the fault (with the following command III.E.4. **Cut/Heal Fault**), and then delete the unwanted element(s).

III.E.4. Cut/Heal **F**ault (Ctrl+F)

If you chose option "1" (for a ***Shells-type*** F-E grid) in the initial Decision-Point dialog, then you may cut (linear) fault elements into your grid, set their properties, and may even "heal" them again if you decide that they are unwanted.

Cut a new fault element by left-clicking at the midpoint of one side of an existing triangular element. (If ***OrbWin*** "chirps" you may have just missed the midpoint; try again.) Success is indicated when ***OrbWin*** paints the fault trace red with a heavy line.

Heal an existing fault element by right-clicking at its midpoint. (If ***OrbWin*** "chirps" you may have just missed the midpoint; try again.) Success is indicated when ***OrbWin*** re-paints the fault trace with a heavy green line. (This heavy green line is a "scar" to remind you where the fault used to be. It will vanish the next time you redraw the screen with any command like III.F.2. **ZoomInOut** or III.F.3. **Refresh**.)

An interesting feature of ***OrbWin*** is that it allows you to Cut Faults along the outer edges of your F-E grid. This is useful when you want to model one or more (contiguous) tectonic plates in some region, but not the whole planet. If you cut faults along the outer boundaries of your grid, then there will be extra nodes created along the "outsides" of these boundary faults, and when you use my ***OrbNumber*** program to get a list of boundary nodes, these extra external nodes will be the ones chosen and placed in the list (to be assigned boundary conditions, by you). Then, you should assign the velocities of neighboring plates as edge boundary conditions in your F-E simulation with ***Shells***, even though no actual area of those neighboring plates is included in your .FEG domain. See *Liu & Bird* [2002b] for an example of this method.

(However, if you later decide to expand your grid beyond these former boundary faults, then the rule stated above in III.E.3. **Add/Delete Element** will require you to temporarily Heal these faults before you will be allowed to add more triangular elements outside them. You can always re-Cut the same faults once your grid of triangular elements is complete.)

Another "oddity" to be aware of is that a fault consisting of a single linear element in the interior of an .FEG has no physical meaning, and will automatically be detected and deleted by ***OrbWin***. (This occurs whenever you leave command **Cut/Heal Fault** and move on to some other command.) Such "zero-DOF faults" have no degrees of freedom for slipping, so they add nothing (except confusion and frustration) to a ***Shells*** simulation. To prevent the auto-deletion of zero-DOF faults, you must either: (a) use at least 2 connected fault elements to represent the trace of any short fault which is isolated in the interior of our grid; or (b) connect your short fault to other faults in a fault junction at either end, or at both ends of its trace; or (c) connect at least one end of your short fault to the outer perimeter of your F-E grid.

Faults in a ***Shells***-type .FEG created with ***OrbWin*** may intersect in junctions of arbitrary complexity. ***OrbWin*** handles the bookkeeping for you. At a "tectonic knot" where *N* faults intersect, ***OrbWin*** will create *N* distinct nodes at the same location, and each one will be tied to a triangular continuum element in one of the *N* intervening microplates. (You can wave the mouse cursor over a fault junction, and the numbers of these multiple nodes will be displayed in the right-most compartment of the lower status bar; however, no more than 5 node numbers will fit into this display, so any node numbers beyond the 5th are not shown.)

III.E.5. Inclination(dip) of fault (Ctrl+I)

If you chose option "1" (for a ***Shells-type*** F-E grid) in the initial Decision-Point dialog, then you may cut (linear) fault elements into your grid and then set their properties.

This command allows you to set the dip of faults (already defined using the previous command). This is critical because ***Shells*** assumes that any vertical fault (one with 90° dip) has purely strike-

slip motion. In order to allow for dip-slip motion (or an oblique motion, with arbitrary rake), you must specify a fault dip significantly less than 90°.

When you first choose this command, a pop-up dialog will ask you to set (or check and confirm) the two "standard" dips to be used. By default, these are 20° for a "shallow" dip and 55° for a "steep" dip, but you may type in other values. (If you wish to re-enter new dip values at any time, just go to the top-level menu and restart this command.)

To apply one of these pre-defined dips to an existing fault element, left-click the midpoint of the fault. (If ***OrbWin*** "chirps" you may have just missed the midpoint; try again.) The fault will change color to indicate that it has been selected. Then, click either the left mouse button (for "shallow" dip) or the right mouse button (for "steep" but non-vertical) dip near each end of the fault element, on the hanging-wall side. The display will change to show the sense of dip.

It is not necessary to specify the dips of vertical strike-slip faults. Each fault element in ***OrbWin*** begins as vertical (90° dip), so you only need to apply this command to non-vertical faults. In fact, the easiest way to return a fault to vertical dip is to Heal and then re-Cut it (see above).

Note that dip1 and dip2 (at each end of the fault element) are recorded in the .FEG file, and a linear variation of dip is assumed in-between. This allows you to create a "corkscrew" fault whose dip changes along its trace, even possibly reversing. Because the effects of such faults in a ***Shells*** simulation have not been fully tested, I encourage you to limit your experiments to modest changes in dip (*e.g.*, from 20° to 45°) within any one fault element, but avoid *reversing* the sense of dip in one fault element.

III.E.6. **H**eading(azimuth) of fault (Ctrl+H)

If you chose option "1" (for a ***Shells-type*** F-E grid) in the initial Decision-Point dialog, then you may cut (linear) fault elements into your grid and then set their properties.

This command allows you to adjust the heading/azimuth angle (measured clockwise from local North, in degrees) of the trace of any pre-existing fault element. Simply left-click on the center of the fault element and then drag the mouse along the desired heading. ***OrbWin*** will show the tangent great-circle arc, and will display its azimuth (at its center) in the bottom status-bar. When the heading/azimuth is acceptable, release the mouse button. The positions of nodes at both ends of the fault will be adjusted to record the change.

This command is primarily useful for modeling short oceanic transform faults that consist of a single transform-fault element (connected to spreading ridges at each end). In the absence of a good basemap (.DIG) file that would show their actual fault traces, you can obtain azimuths in other ways. I used this command when creating the Earth5 grid of *Bird et al.* [2008], in order to be sure that the azimuths of short oceanic transform faults along mid-ocean spreading ridges were consistent with their real azimuths as given in Table 3 of *DeMets et al.* [1990]. Using this command on a long continental transform fault (like the San Andreas or North Anatolia fault) would not be satisfactory, because each heading change would disrupt the headings of the neighboring fault elements along-strike. In such cases, it is much simpler to import the real fault trace with III.D.2. **LoadBase**, and then to use III.E.2. **AdjustNode** to move all fault nodes exactly onto the digitized trace.

III.E.7. Elevation(*OR* **m**u)/Q/Crust/Mantle/Density-anomaly/Cooling-curvature

Nodal data fields were described in section II.B of this manual (above). This command creates a color-contour-map visualization of any nodal-datum field, and allows local manual adjustments. When the command is launched, a pop-up dialog requires you to choose which field will be displayed (and perhaps edited). The ***OrbWin*** display is augmented by coloring-in all triangular continuum elements to show the values of this field. A key to the color-scale is shown on the left side of the main display window; all values are in SI units, since that is what ***Shells*** (or ***NeoKinema***) expects in any nodal-data fields.

To examine the exact value at any node (and possibly change it), left-click on the node. A dialog box will appear. As soon as you close this dialog, the display will update.

If the node that you selected and changed was a (multiple) fault node, then the nodal data of all co-located nodes are changed together. This preserves lateral continuity of nodal data fields across faults. (One might wish to consider models with discontinuities in nodal data across faults. However, in such cases the discontinuities would have to be entered manually into your .FEG file(s) with a text-editor, or else created by your own utility program that modifies your .FEG file(s). In such cases, ***OrbWin*** can still be used to display the discontinuous nodal data, but cannot be used to adjust discontinuous values. Any attempt to make such an adjustment will immediately re-enforce lateral continuity of the selected nodal datum in that particular node-cluster.)

Once you have used the left-click dialog to set some preferred value, you can set other nodes to the same exact value using Ctrl+(Right-click). (Another option would be to use the next command, III.E.8. **Block Set mode**.)

To turn off this display/editing command, with its colored contour map, go back to the top-level command again and click on it again (to cancel the check-mark beside it).

In practice, I have never used this command to set all the nodal data for an entire F-E grid. It would be too laborious, and subject to error, and it would not guarantee internal-consistency between the different nodal data fields (*i.e.*, lithosphere thickness should vary inversely with reduced heat-flow; and crustal thickness should be correlated with elevation). See section V.C. **Programs to compute nodal data** below for some better options. This command is mainly useful to visualize the data and to make local small adjustments. (For example, a ***Shells*** model may fail because of a local "landslide" instability. In that case, the easiest fix is usually to manually reduce a few heat-flow values in the affected region with this command, and then to re-run ***OrbData*** or ***OrbData5***, as described below.)

III.E.8. Block Set mode for Elev.(OR mu)/Q/Crust/Mantle

To avoid repetition, I will assume that you have read the preceding manual section, III.E.7.

This command provides a quick and convenient way to set the nodal data of many adjacent nodes to the same value, in one operation. First, you must have just entered color-contour-map display mode by using command III.E.7. **Elevation(OR mu)/Q/Crust/Mantle/Density-anomaly/Cooling-curvature** (described above).

Second, you must outline a convex polygon, in the counterclockwise direction, with left-clicks of the mouse. Finish with one right-click of the mouse near your starting point. (For example, to

outline a rectangle, you would use 4 left-clicks and then 1 right-click.) The location of the right-click does not have to be precise. Once the polygon is complete, a pop-up dialog will ask you for the desired datum value. The display will update as soon as you press **Enter** or **OK**.

If you should accidentally define your polygon in the clockwise direction, nothing will happen.

If you should accidentally define a polygon that is not convex (*i.e.*, it has a "bite removed"), then results will be unpredictable, and probably not what you intended! (The easiest way to fill-in nodal data in a non-convex polygon is to break it into a group of convex polygons, and use this command several times.)

Because of such issues, it is best to use command III.D.3. **SaveGrid** before you start!

This command does not provide any way to reset all values of a global grid in one operation. All the nodes to be changed have to be visible in the display (and also inside your polygon). However, you can adjust all the nodes of a global grid fairly quickly by using these 5 viewpoints in sequence: (0°N, 0°E); (0°N, 120°E); (0°N, 240°E); (90°N, 0°E); (-90°N, 0°E). It might also be convenient to use III.F.2. **ZoomInOut** to set the window-height to 3.0 first, so that you will not have to use the scroll-bars to see the top and bottom of the planet.

III.E.9. Lithospheric Rheology (LR) index of each element

If you chose option "1" (for a ***Shells-type*** F-E grid) in the initial Decision-Point dialog, then you have an *OPTION* to specify different Lithospheric Rheologies indices for different elements. If you later run ***Shells_v5.0+***, these indices will be used in the solution; however, earlier versions of Shells (***Shells_v4.1-***) will just ignore these numbers.

A Lithospheric Rheology index number (or LR#, for short) is a non-negative integer that is a key to a set of different rheologic constants, briefly identifying a different "terrane" with a different geologic-column of rock-types in that part of the lithosphere. LR#s may be applied to either/both triangular continuum elements and linear fault elements. If you want an element to have the default rheology, you can either specify this with "LR0" [←zero, not "O"], or just omit any LR# specification.

The physical meaning of the different LR#s is entirely up to the user. The different rheologic constants needed to define LR1, LR2, LR3, … are *NOT* specified in ***OrbWin***. Instead, they will be read from an additional ASCII-table input file that you will provide when ***Shells_v5.0+*** is started.

This command creates a colored-triangular and colored-fault-trace visualization of LR#s, and allows local manual adjustments. The ***OrbWin*** display is augmented by coloring-in all triangular continuum elements to show their LR#s. A key to the color-scale is shown on the left side of the main display window; all displayed LR# values are actually integers (even though the displayed key to the color-scale uses floating-point numbers for programming convenience).

To examine the LR# of any triangular continuum element (and possibly change it), left-click on the center of the element. A dialog box will appear. As soon as you close this dialog, the display will update.

To see the LR# of any linear fault element, focus on the color of the center-stripe of the fault trace. (The colors of the outer parts of the fault trace may be different, as they are still being controlled by command III.F.6. **Set Colors**.) To check (and possibly change) the LR# of a fault,

click on its midpoint. A dialog box will appear. As soon as you close this dialog, the display will update

Once you have used the left-click dialog to set some preferred value of the LR#, you can set other elements (continuum or fault) to the same exact value using Ctrl+(Right-click). (Another option would be to use the next command, III.E.10. **Block Set mode for LR#s of elements**.)

To turn off this display/editing command, with its colored contour map, go back to the top-level command again and click on it again (to cancel the check-mark beside it).

III.E.10. Block Set mode for LR#s of elements

To avoid repetition, I will assume that you have read the preceding manual section, III.E.9.

This command provides a quick and convenient way to set the LR#s of many adjacent elements (both triangular continuum elements and linear fault elements) to the same integer value, in one operation. First, you must have just entered LR#-editing display mode by using command III.E.9. **Lithospheric Rheology (LR) index of each element** (described above).

Second, you must outline a convex polygon, in the counterclockwise direction, with left-clicks of the mouse. Finish with one right-click of the mouse near your starting point. (For example, to outline a rectangle, you would use 4 left-clicks and then 1 right-click.) The location of the right-click does not have to be precise. Once the polygon is complete, a pop-up dialog will ask you for the desired LR# (integer) value. The display will update as soon as you press **Enter** or **OK**.

If you should accidentally define your polygon in the clockwise direction, nothing will happen.

If you should accidentally define a polygon that is not convex (*i.e.*, it has a "bite removed"), then results will be unpredictable, and probably not what you intended! (The easiest way to set uniform LR#s in a non-convex polygon is to break it into a group of convex polygons, and use this command several times.)

Because of such issues, it is best to use command III.D.3. **SaveGrid** before you start!

This command does not provide any way to reset all LR#s of a global grid in one operation. All the nodes to be changed have to be visible in the display (and also inside your polygon). However, you can adjust all the nodes of a global grid fairly quickly by using these 5 viewpoints in sequence: (0°N, 0°E); (0°N, 120°E); (0°N, 240°E); (90°N, 0°E); (-90°N, 0°E). It might also be convenient to use III.F.2. **ZoomInOut** to set the window-height to 3.0 first, so that you will not have to use the scroll-bars to see the top and bottom of the planet.

To turn off this display/editing command, go back to the top-level command menu again and click on it again (to cancel the check-mark beside it).

III.F. **View** commands (Alt+V)

III.F.1. **D**rawGrid (Ctrl+D)

This command is automatically executed after **LoadGrid** or **ClearGrid** or **LoadBase** or **ClearBase** or **GlobalGrid**. It resets the **ZoomInOut** window-height factor to 2.02 planetary radii, and turns off any special element-coloring mode(s) like those described just above. The viewpoint also returns to the default position on the sphere. You can select this command any time you want to quickly "reset" the display.

III.F.2. **Z**oomInOut (Ctrl+Z)

This is the primary tool for changing your view point, and also the height (zoom factor) of your display window. Each time this command is entered, a pop-up dialog asks you to set (or confirm) the window-height, expressed in units of planetary radii. (For example, window-heights of 2.0 to 3.0 are good for seeing an entire hemisphere; small window-heights like 0.1 or 0.01 are good for seeing local details.) If you do not want to change the zoom-factor, just press **Enter** or click **OK**.

Now, left-click the mouse at any point on the planetary sphere, to make that point the center of the display. You can do this as many times as you like, so that the display "walks" around the globe.

Remember that the *(longitude, latitude)* coordinates of the mouse cursor are displayed in the lower status-bar. North is always "up" in the display; this cannot be changed.

If you want to make a large or precise change to the window position (without changing the zoom-factor) an alternative is to use one of these commands: III.F.4. **WindowPosition**, III.F.7. **ShowNode**, or III.F.8. **ShowElement**.

III.F.3. **R**efresh (Ctrl+R)

This command tells ***OrbWin*** to re-draw the display without changing the viewpoint, the zoom factor, or the color-contour-map mode (if active). Logically, its only effects are: (1) to turn off any element-coloring that was previously inserted with the **View Gaps/Overlaps** tool; (2) turn off any virtual second coordinate system established by III.F.5. **2ndOriginSet**; and (3) to clear any wide green "scab" lines left where faults have been **Cut** and then **Heal**ed. However, it may also be useful at any time the display becomes unclear for any reason. (For example: You may need to use **Refresh** after an especially energetic session with **AdjustNode**.)

If you need to turn off color-contour-map mode, use command III.F.1. **DrawGrid** instead.

III.F.4. WindowPosition

This command allows you to accurately set the *(latitude, longitude)* coordinates of the center of the display window. As always, North latitudes are positive and South latitudes are negative; East longitudes are positive and West longitudes are negative.

The window-height (zoom factor) set by the most recent use of III.F.2. **ZoomInOut** or III.F.1. **DrawGrid** will not be changed.

As in command III.F.2. **ZoomInOut**, you can use left-clicks of the mouse to select new window center positions (from within your current field of view) if you want to make small changes.

III.F.5. 2nd**O**riginSet (Ctrl+O)

This command allows you to measure distances and angles on the planetary sphere. It does this by establishing a "secondary virtual North pole" and an associated "secondary virtual prime meridian" of zero longitude. (However, the original nodal coordinates, with which your .FEG file will eventually be saved, are not affected.)

After starting the command, left-click once to set the virtual North pole, and right-click once (on some other point) to set the direction of the virtual prime meridian, as it leaves the North pole. After this, alternative secondary geographic coordinates of the mouse cursor will be displayed (along with conventional coordinates) in the lower status-bar.

For example, you can measure the distance between two nodes (or between two features on the basemap). After left-clicking on the first point, and right-clicking on the second point, note the virtual latitude of the second point. If it is 80.131°N, for example, and if the planetary radius is 6371 km, then the arc-of-great-circle distance is (90°N – 80.131°N) * (π / 180°) * (6371 km) = 1097 km, or 0.1722 radians, or 9.869°.

To measure angles, choose a reference line that will be your virtual prime meridian. Left-click on this reference line at the point where some other line crosses it. Then right-click at any other point on your reference line. You can now move the mouse (without clicking) and read "virtual longitudes" of the mouse cursor (in the lower status-bar) as angles from your reference line, about the pivot point that you selected first. Positive virtual longitudes represent angles that are counterclockwise from your reference line, and negative virtual longitudes represent angles that are clockwise from your reference line.

Note that the secondary-prime-meridian reference line (from polar left-click to right-click) remains marked in white. You can replace it by using another pair of mouse clicks: first left-click, and then right-click. Or, you can eliminate the second virtual coordinate system entirely by using command III.F.3. **Refresh**.

III.F.6. SetColors (Ctrl+K)

This command allows you to change the color of every part of the display, including the background color. Colors are specified using RGB triplets of additive "integer brightness" levels *(ibRed, ibGreen, ibBlue)*. Each integer can range from 0 up to 255. Therefore, (0, 0, 0) is black, (255, 255, 255) is white, (255, 255, 0) is yellow, and so on.

The only difference between the "**Apply**" button and the "**OK**" button is that the "**Apply**" button keeps the dialog box open, in case you might wish to make more changes. The "**OK**" button closes the dialog box.

In this dialog, you can choose to turn off the display of small-triangle (or small-X) icons at the centers of elements. This will produce a "cleaner-looking" display. However, it will make it almost impossible to work with commands III.E.2. **AdjustNode** and III.E.3. **Add/Delete Element**, and difficult to work with command III.G.4. **View Gaps/Overlaps**.

Likewise, you can choose to turn off the display of node circles. This will also "clean up" the display. However, it will make it almost impossible to work with commands III.E.1. **Add/Drop Node** and III.E.2. **AdjustNode** and III.E.3. **Add/Delete Element**.

Thus, the default colors are best for most purposes. If you accidentally create a color combination that makes it impossible to work, then just **SaveGrid** and then **Exit**. When you start ***OrbWin*** again, the default colors will return.

III.F.7. ShowNode

This command prompts you to enter an integer node number in a pop-up dialog window. After you press **Enter** or click "**OK**", the display will move to place this node in the center of the

window, and mark it with cross-hairs. (If you use **Refresh** or **ZoomInOut**, these cross-hairs will vanish.) The window-height (zoom factor) is not changed.

III.F.8. ShowElement

This command prompts you to enter an integer element number in a pop-up dialog window. After you press **Enter** or click "**OK**", the display will move to place this element in the center of the window, and color it in. (You can turn off the coloring with **Refresh**.) The window-height (zoom factor) is not changed.

III.G. **Tools** commands (Alt+T)

The first two commands below are for creating a uniform F-E grid when you do not already have one loaded (*e.g.*, at the start of a modeling project). The last two tools below are for checking the topology of a grid, before sending it on to applications like ***OrbNumber*** and perhaps ***OrbData*** or ***OrbData5***, and then ***Shells*** or ***NeoKinema*** or ***Restore***. Thus, these latter two commands are most often used near the end of F-E grid editing, just to be sure that no errors have crept in.

III.G.1. GlobalGrid

This command creates a uniform global F-E grid of triangular continuum elements (and necessary nodes), by iterative and recursive subdivision of the faces of an icosahedron [*Baumgardner*, 1983; *Glatzmaier & Schubert*, 1993]. Your only choice is the level of subdivision, which is always a non-negative integer (nSlice). The final grid will have $20\times(4^{nSlice})$ elements.

> Be aware that ***OrbWin*** can easily create grids which are too fine to solve (in reasonable amounts of computer time and memory) in my F-E applications ***Shells***, ***NeoKinema***, and ***Restore***. It would be unfortunate if you spent days creating a very fine, hand-edited grid—and then discovered that it is impractical to use. Therefore, I suggest that you limit your initial experiments with **GlobalGrid** to subdivisions of nSlice ≤ 5 (like model Earth5 of *Bird et al.* [2008]). You can always create locally-finer grid modifications in regions of special interest, like around faults.

During execution of **GlobalGrid**, it may appear that ***OrbWin*** has frozen, when actually it is just busy creating elements on the far side of the planet, where they are not presently visible. For this reason, a graphical progress-bar appears across the bottom of the ***OrbWin*** frame window.

III.G.2. TileRegionalGrid

Like the previous command **GlobalGrid**, this command creates a uniform F-E grid of triangular continuum elements (and necessary nodes), by iterative and recursive subdivision of the faces of an icosahedron. The difference is that creation of elements and nodes will be limited to the area of a convex polygon (within the nearer, visible hemisphere of the planet) which you will outline with mouse-clicks.

Your first choice is the level of subdivision, which is always a non-negative integer (nSlice). If the polygon you draw covers 10% of the area of the planet, then the final grid will have about 10% $\times 20\times(4^{nSlice}) = 2\times(4^{nSlice})$ elements.

Be aware that ***OrbWin*** can easily create grids which are too fine to solve (in reasonable amounts of computer time and memory) in my F-E applications ***Shells***, ***NeoKinema***, and ***Restore***. It would be unfortunate if you spent days creating a very fine, hand-edited grid—and then discovered that it is impractical to use. Therefore, I suggest that you limit your initial experiments with **TileRegionalGrid** to subdivisions of nSlice ≤ 7. You can always create locally-finer grid modifications in regions of special interest, like around faults.

Draw a convex polygonal region on the planet, in the counterclockwise direction, using left-clicks of the mouse. Indicate that the polygon is finished by entering one right-click near your starting position. (Exact location of the final right-click is not important.)

An interesting feature of **TileRegionalGrid** is that you may draw more than one polygon (if you do it immediately, while the command is still running), and the resulting grids will be knit together with no overlapping nodes or elements. This feature allows you to create a model domain that has a non-convex perimeter, if you wish. HOWEVER, if you return to command **TileRegionalGrid** and enter a different subdivision level (nSlice) from the one you used before, this automatic linking and merger of the polygons will not work. Instead, you will have to patiently "stitch" the finer and the coarser grids together with repeated use of command III.E.3. **Add/Delete Element**. Therefore, in such cases it is best to leave a narrow strip of ungridded surface area between the two polygons (of different nSlice) created by **TileRegionalGrid**.

If you make a mistake with this command, it is usually best to simply **ClearGrid** and try again.

III.G.3. Perimeter/Area Tests

Please review the introductory section: II.C. **Rules and violations of F-E grid topology**.

This command actually runs two consecutive test calculations when you click on "**Start**":

Perimeter test. First, each node is tested to see if it is attached to some element. (If not, a pop-up dialog appears, which allows you to request that all such unconnected nodes be deleted.) Each side of each triangular continuum element (and linear fault element, if any) is then considered to see if it has any "outside edges" (which are not shared with other connected elements). Nodes which lie on any perimeter (edge) of the F-E grid are identified in this way. Then, the calculation attempts to link these nodes by stepping through existing triangular (and/or fault elements, if any) into a continuous perimeter "necklace". If the grid is simply-connected, this perimeter chain should return to its starting point just as the supply of exterior nodes is used up. If there are surplus exterior nodes, then there must be some combination of exterior islands and/or interior holes. In case of failure, a special basemap file BADNODES.DIG is created, using different kinds of symbols to distinguish nodes which belong to the main perimeter necklace, versus the exterior "island" or interior "hole" necklace(s). You should **ClearBase** (if a basemap is currently loaded) and then **LoadBase** to see BADNODES.DIG. If the display is too crowded to understand, then you could: (1) **SaveGrid**, (2) temporarily **ClearGrid** to see only BADNODES.DIG, and then (3) **LoadGrid** again to resume fixing the problems. (Note that BADNODES.DIG will not automatically update as problems are fixed. It is a static file. Therefore, you should eventually **ClearBase** and then run **Perimeter/Area Test** again.)

The second test is run only if the first test is passed:

Area test. If all triangular elements have positive areas, and they all share edges (but do not overlap, and leave no gaps), then the sum of individual element areas should equal the total area inside the perimeter of the F-E grid. This numerical test is done using the formulas for areas of spherical triangles. All areas quoted are in steradians; to convert to areas in square-meters, multiply areas in steradians by the square of the radius of the planet. Due to numerical round-off errors, even a topologically-correct grid may show a tiny discrepancy (*e.g.*, 0.0004%; that is only 4 ppm). However, any error of 0.0100% or more should be investigated. The following command will be useful…

III.G.4. View Gaps/Overlaps

Please review the introductory section: II.C. **Rules and violations of F-E grid topology**.

If the **Area Test** (see above) identified some topological problem with your F-E grid, but the Perimeter Test was passed (and thus no BADNODES.DIG file was created), then this command may help you to see where the problem is. (Remember to look at both hemispheres if you have a global model.) This command should always be run at the end of a grid-editing session.

Click on the center icon of any triangular continuum element, and it will be colored-in. Also, the color will spread to any adjacent connected triangular element(s) through common shared nodes (but color will not spread across linear fault elements). If you click more than once, a different color will appear and spread.

One obvious use of this command is to search for holes in the grid (which will not be colored, and will remain background-colored). Another is to find overlaps between elements, which will appear as a contrasting color.

If you are designing a ***Shells***-type F-E grid with linear fault elements, you may have intended to model plates that are entirely surrounded by faults. In this case, the coloring should remain confined to one plate, and should not "leak" out. However, if you are modeling intraplate faults or a juvenile plate boundary, then imperfect connections between faults may be the actual situation, and "leakage" of color might be expected.

When you are finished using this command, either III.F.3. **Refresh** or III.F.1. **DrawGrid** will turn off the special coloring-mode.

III.H. **Help** commands (Alt+V)

III.H.1. Hints

This command displays a little pop-up text box with really basic suggestions: (1) Read the hints (about how to use the mouse) that are displayed in the lower status-bar. (2) Read the online manual (via the next command…).

III.H.2. Manual

This command starts a web browser (***Edge***, ***Internet Explorer***, or ***Chrome***) in a separate window, and loads this manual PDF. (If you wish to use a different browser, or a PDF-reader application like ***Acrobat Reader***, then you can download OrbWin_manual.PDF in advance from http://peterbird.name/oldFTP/OrbWin. It is not necessary to access this manual through the ***OrbWin*** menu.)

III.H.3. About

Really basic information about ***OrbWin***: version number, authors, and copyright year.

IV. SUGGESTIONS FOR USING ***OrbWin***

IV.A. Creating your basemap

In most cases, you will produce a better .FEG file if you start with a good digitized basemap (.DIG) file. The locations of certain features (plate boundaries and other fault traces; geodetic benchmarks) can be critical during grid design. Also, geographic features such as coastlines and state/national boundaries can provide context, and help you to relate your work to geologic information that is already in your personal memory.

There are several .DIG files already posted at http://peterbird.name which may be useful:

Content:	Filename:	In folder:
Plate boundaries	PB2002_boundaries.dig	/oldFTP/PB2002/
Plate outlines	PB2002_plates.dig	/oldFTP/PB2002/
Orogens	PB2002_orogens.dig	/oldFTP/PB2002/
Coastlines	worldMap.dig	oldFTP/neotec/Shells/
State lines	North_America_states.dig	oldFTP/Restore/

If you have GPS data on secular (interseismic) velocities of benchmarks, you should convert this data to my .GPS format (documented at: http://peterbird.name/guide/gps_format.htm). The next step is to convert benchmark locations to .DIG file format with my utility program ***GPS_2_DIG*** (discussed below in section V.A.2. ***GPS_2_DIG***).

If you will be modeling faults, then you should digitize their traces, in a two-or-three-step process: (1) Place your fault map on a digitizing tablet, and capture *(x, y)* coordinates of points along the fault trace with a digitizing program (*e.g.*, ***Didger5*** from Golden Software, or my ancient DOS program ***Digitize***); (1A) If you digitized with any program OTHER than ***Digitize***, you need to save your work as an ESRI Shapefile (.SHP) and associated dBase file (.DBF), and then convert these to .DIG format with my utility program ***SHP_toFrom_xyDIG***. (2) Convert from *(x, y)* to *(longitude, latitude)* coordinates with my utility program ***Projector***, which will require you to enter details of the map projection. ALL of these programs are discussed below in sections V.A.3. ***Digitize*** and V.A.4 S***HP_toFrom_xyDIG*** and V.A.5. ***Projector***.

Another option is to obtain digital fault traces from any reliable source, and create your own program (or spreadsheet) to re-write these traces into my .DIG format (documented above, in section II.E. **DIGitized-lines (.DIG) file format**). For example, fault traces in the USA might be obtained from the U.S. Geological Survey. Fault traces in California might be obtained from the Community Fault Model of the Southern California Earthquake Center.

You can combine all these components (and others …) into one .DIG file simply by concatenating them. You can do this in a simple ASCII text-editor program (*e.g.*, ***NotePad***), or use the old DOS ***COPY*** command (see section V.A.7. ***COPY*** below).

Even though you must combine all these components into a single .DIG file for use in ***OrbWin***, it is also good to retain your old copies in the form of separate .DIG files. Then, if you use one of my graphical post-processing programs (described in section V.D. **Making maps with graphical post-processing programs**) to plot them, these separate .DIG files will be transformed to separate graphical “groups”

within a single Adobe Illustrator .AI file, allowing you to easily select different line colors and line-weights for coastlines versus plate boundaries and faults, *etc.*

IV.B. Gridding plate-interior (continuum) areas

Start ***OrbWin*** and work through the two introductory dialog interactions (described above in sections III.A and III.B). Then, load your basemap (.DIG) file with III.D.2. **LoadBase**. Then, decide whether you will create a global model, or a regional model.

For a global model, begin with command III.G.1. **GlobalGrid**. I suggest choosing a subdivision level no higher than 5 (which is shown in the Earth5 sample grid on the cover of this manual). Remember, you will be able to insert finer grids locally where there are active faults and folds.

For a regional model, consider first where the outer edges will be. It is <u>not</u> good to have the outer edges of a regional model located within a complex "orogen" with ongoing distributed deformation [*Bird*, 2003], because then the relative velocities of boundary points, needed for boundary conditions, will usually not be known. It is usually better to have the outer edges located within large, strong lithospheric plates whose Euler poles are already known. (One possible reference is Table 1 of *Bird* [2003]; this table is also built into my F-E codes ***Shells*** and ***NeoKinema***, where it can be referenced easily using two-letter plate codes like "PA" and "NA".)

Start command III.G.2. **TileRegionalGrid**, and select a subdivision level no higher than 7 (for your first experiments). Then, outline the model region with left-clicks of your mouse in the counterclockwise direction; use a right-click when you return (approximately) to your starting point. As explained in section III.G.2, you can use multiple overlapping convex polygons to build up a model region which is *not* convex, if you wish.

IV.C. Local grid refinement

Perhaps there are some regions within your model that are of special interest, or have fine-scale structure (like many closely-spaced faults)? In that case, you will probably want to refine the F-E grid in those regions. (Do this <u>before</u> you try to insert individual fault traces.) There are 5 steps:

(1) Use III.E.3. **Add/Delete Element** to delete some existing large elements from this region of future refinement. Try to make its outline convex, if possible.

(2) Use III.E.1. **Add/Drop Node** to drop (delete) any "free-floating" unattached nodes left over.

(3) Use III.G.2. **TileRegionalGrid** and select a subdivision level that is *higher by one* than in your large basic grid. Outline a convex polynomial (or, more than one) with mouse clicks just barely inside the cleared area of your larger grid.

(4) Use III.F.2. **ZoomInOut** and then use III.E.3. **Add/Delete Element** to manually "sew together" the older coarse grid and the newer fine grid by adding new elements.

(5) Use command III.G.3. **Perimeter/Area Tests** and then III.G.4. **View Gaps/Overlaps** to check the topology of your grid. Fix any problems before proceeding further. Then, **SaveGrid**.

It is even possible to insert a refinement-within-a-refinement by repeating this method.

IV.D. Representing faults

I will assume that you already have fault traces included in your .DIG basemap file, and that this is loaded into ***OrbWin*** with command III.D.2. **LoadBase**. You should use command III.F.2. **ZoomInOut** frequently to get a zoomed-in, clear view of each region you are working on. Also, use **SaveGrid** often (remembering that ***OrbWin*** has no "Undo" command).

IV.D.1. Fault elements (for ***Shells***)

The plan is first to approximate each fault trace with a series of (green) sides of triangular continuum elements; then to "Cut" (red) faults along these sides; and finally to set their dips. The usual steps are:

(1) Look for "triple junctions" (or higher-order "tectonic knots") where 3 or more fault traces intersect. Use III.E.2. **AdjustNode** to move the closest existing node onto each of these points.

(2A) Where faults are widely spaced: Simply drag existing nodes onto the fault trace, so that the (white) line in the basemap file is (approximately) overlain by green element-side lines.

(2B) Where faults are closely spaced, there may not be enough existing nodes to use method (2A). In that case, use **Add/Delete Element** to delete existing elements and thus clear some space around a group of faults. Also delete any unconnected nodes. Then, use **Add/Drop Node** to create new nodes along the trace of each fault. (If a fault ends without connecting to another, then place a new node at this propagating tip.) Finally, "sew" together the new nodes and the existing nodes by using **Add/Delete Element** to create new elements.

(3) Use command III.E.4. **Cut/Heal Fault** and a series of left-clicks on the midpoints of element sides (that lie over fault traces in the basemap) to cut the faults.

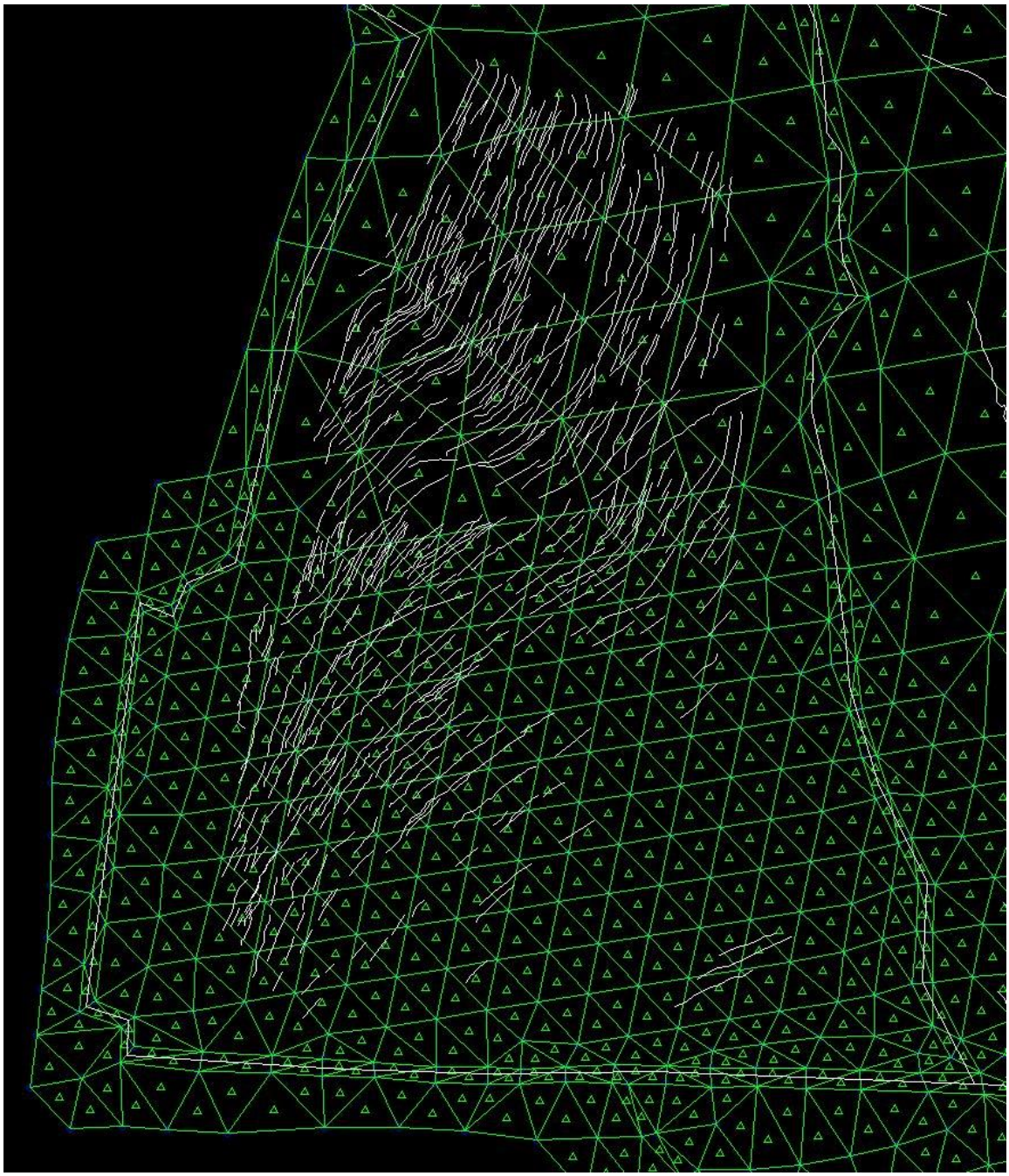
(4) Use command III.E.5. **Inclination(dip) of fault** to set the proper dip of each fault.

(5) Use the topological tests III.G.3. **Perimeter/Area Tests** and then III.G.4. **View Gaps/Overlaps** to detect any problems with your new grid. Fix these immediately, before moving on to a new area.

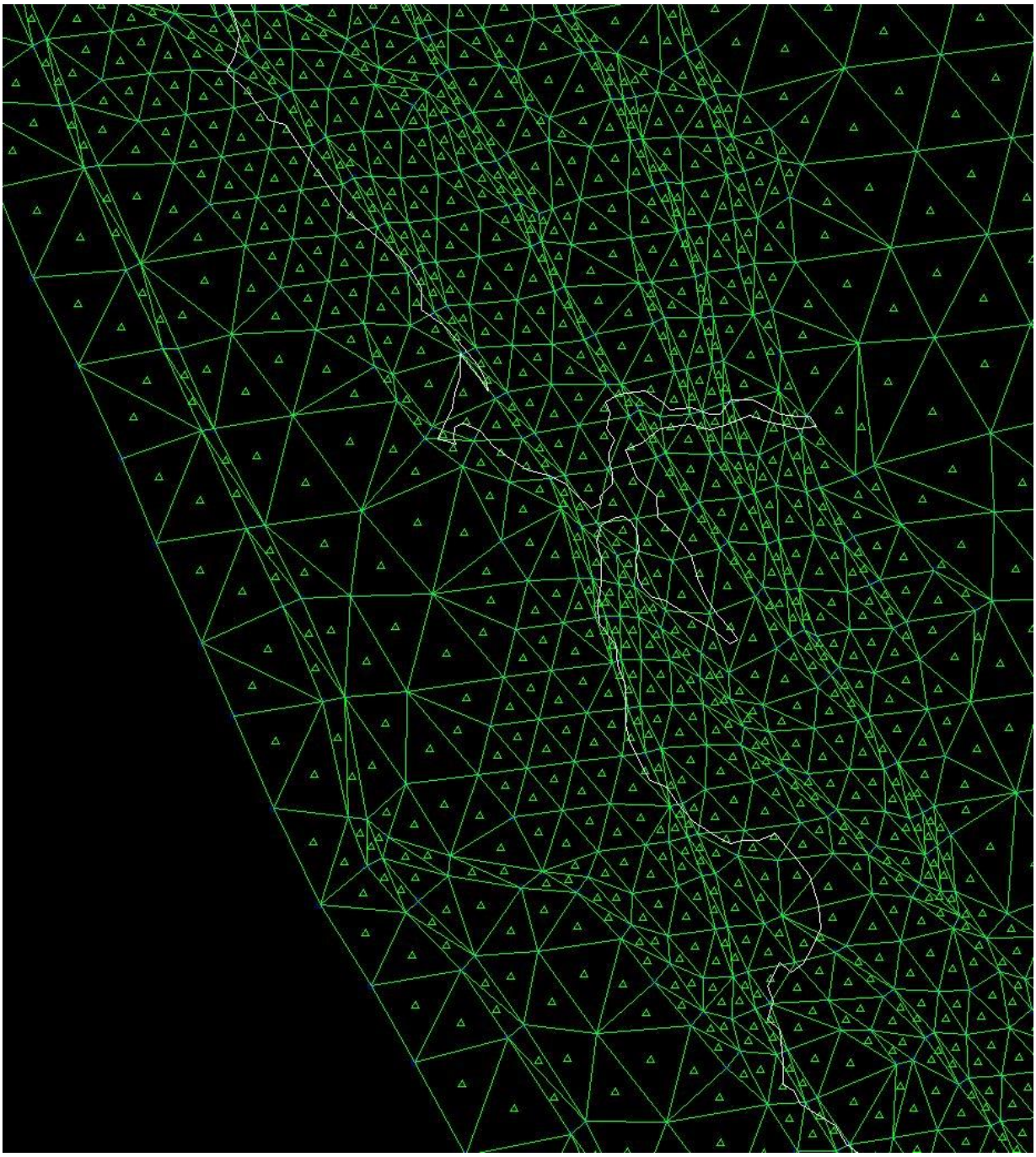
IV.D.2.Optional fault-corridors (for ***NeoKinema***, ***Restore***)

If you have been reading the documentation for ***NeoKinema***, then you may already know that: (a) no linear (red) fault elements are allowed in .FEG files used by ***NeoKinema***; and (b) that fault traces are supplied directly to ***NeoKinema*** in the f*_yourChosenName*.DIG file. This means that, in principle, you do not need to adjust your F-E grid to reflect the positions of faults.

In my 2009 NeoKinema model of the Gorda-California-Nevada orogen [*Bird*, 2009] I decided not to refine my F-E grid in the offshore region called the "Gorda plate" (although "Gorda orogen" would be a more appropriate name). *Chaytor et al.* [2004] had used high-resolution swath bathymetry and seismic reflection profiling to define 545 active or potentially-active faults in the region, which was an overwhelming number. Also, there seemed to be little point in trying to model them individually, because (a) only one of these faults had a geologic slip-rate constraint; (b) none of these faults had GPS benchmarks nearby; and (c) no one lives anywhere near those faults, so their seismic hazard does not imply any seismic risk. Therefore, I just left the auto-generated F-E grid from **TileRegionalGrid** in the area of these 545 faults, as shown below. The result (in the ***NeoKinema*** simulations) was to make the whole region about equally weak.



However, as the figure above also shows, I outlined the fast-moving plate-boundary faults (Mendocino transform fault, Gorda Rise spreading center, and Cascadia Trench) with “fault corridors” of narrow triangular continuum elements. Also, I used fault corridors for all the major (fast-slipping) faults on land in the western United States. For example, the next image shows a part of the grid from the San Francisco Bay area:



There are some major advantages to defining “fault corridors” in your .FEG (although it is somewhat laborious): (a) ***NeoKinema*** and ***Restore*** do not actually represent faults with velocity discontinuities; they represent them with zones of enhanced weakness and higher strain-rate. A narrow zone of enhanced strain-rate is a better approximation of a velocity discontinuity than a broad zone is. (b) The exact shapes and relationships between fault traces (especially strike-slip fault traces) create an anisotropy and inhomogeneity in the regional strength that is captured better if you use narrow fault corridors. (c) ***NeoKinema*** detects cases of GPS benchmarks within the same triangular finite-element as a fast-slipping faults, and requires you to remove these GPS benchmarks from the input data. If the fast-slipping faults are confined in narrow fault

corridors, then the number of GPS benchmarks removed can be minimized, and the number of GPS data kept in the simulation can be maximized.

Assuming that you decide to define fault corridors (along fast-slipping faults, >1 mm/a, on land), the procedure is fairly simple:

(1*) Use **Add/Delete Element** to delete existing elements and thus clear some space around a group of faults. Also delete any unconnected nodes.

(2*) Use **Add/Drop Node** to create pairs of new nodes, spaced periodically along the trace of each fault. Where a fault ends without connecting to another, place a single new node just ahead of this propagating tip. Around a triple-junction between 3 faults, place 3 new nodes (one on each microplate). Around a quadruple-junction, place 4 new nodes, *et cetera*.

> *As of January 2018, a new utility program called ***Fault_Corridors*** is available to automate steps #1 and #2 above. It reads an .FEG file without fault corridors, and a .DIG file with digitized fault traces, and uses these to create a new .FEG file with fault corridors. It will still be necessary to manually perform steps #3, #4, and #5 (described below) in ***OrbWin***.

(3) Finally, "sew" together the new nodes and the existing nodes by using **Add/Delete Element** to create new elements. Try to avoid very large (obtuse) angles at the vertices of these new elements.

(4) Check that the fault corridors actually enclose the fault traces, and **AdjustNode** if necessary.

(5) Use the topological tests III.G.3. **Perimeter/Area Tests** and then III.G.4. **View Gaps/Overlaps** to detect any problems with your new grid. Fix these immediately, before moving on to a new area. Then **SaveGrid**.

IV.E. Renumbering grid nodes

Whenever you feel that an ***OrbWin***-generated .FEG file is "done" it is time to renumber the nodes with my utility program ***OrbNumber***. This automatic process will not make any change in the appearance of the .FEG file inside ***OrbWin***—except that the node numbers appearing in the lower right status-bar, in the "Nearest nodes:" display, will be different. However, this renumbering will lead to HUGE reductions in both the computer-memory and execution-time requirements when these grids are used in ***Shells***, ***NeoKinema***, or ***Restore***! (Technically, we are minimizing the bandwidth of the banded coefficient matrix in the linear system produced and solved when implementing the F-E method.)

If you later decide to make some edits to your .FEG file that involve deleting triangular elements and/or adding triangular elements (or, adding and/or deleting fault elements in a ***Shells***-type grid), then it will be necessary to run **OrbNumber** again. (However, changes to nodal data, as described in the next section, do not count as changes to grid topology.)

An additional benefit of running ***OrbNumber*** is that it will produce an ordered list of boundary nodes, going counterclockwise about the perimeter of your F-E grid. You will need this list to assign boundary conditions for ***Shells***, ***NeoKinema***, or ***Restore***. So, save this new "perimeter" (.PER) file next to your renumbered (and re-named) .FEG. (The only time you will not get this second output file is when you process a global grid—it has no side boundaries!)

IV.F. Working with nodal data

Nodal data are additional variable values attached to each node, and listed in the .FEG file following the *(longitude, latitude)* coordinates of the node. See section II.B. **Finite-Element Grid (.FEG) file format**.

The visualization (and minor manual editing) of nodal data in ***OrbWin*** was discussed in manual sections III.E.7 and III.E.8, above. Once any nodal-data field is complete, it can be visualized with sophisticated mapping programs; see manual section V.D. **Making maps with graphical post-processing programs**.

IV.F.1. Lithosphere structure (for ***Shells***)

F-E grids intended for use in ***Shells*** require at least 4 [or, optionally, 6] nodal data for each node: elevation, heat-flow, crustal thickness, mantle-lithosphere thickness [, density anomaly of chemical origin, non-steady curvature of the geotherm]. All are expressed in SI units: m, W/m^2, m, m[, kg/m^3, $°K/m^2$]. Zero values appear if these quantities have not yet been estimated, as when new nodes have recently been created. When a single node becomes a set of multiple co-located nodes (due to the Cutting of faults), any nodal data are copied from the existing node to the new nodes.

Fortunately, I have written two utility programs to fill in these values for you: ***OrbData***, and ***OrbData5***. ***OrbData*** creates the simpler, 4-variable lithosphere structure that was used in early version of ***Shells*** (and is still available, by backward-compatibility). ***OrbData5*** creates the more complex (but more realistic) lithosphere structure that was used in the Earth5 model of *Bird et al.* [2008]. Both are described further in section V.C. **Programs to compute nodal data**.

To run ***OrbData***, you need: (1) a finished .FEG from ***OrbWin***; (2) a grid-file (.GRD) of topography from some DEM; (3) a grid-file (.GRD) of heat-flow; and (4) some physical parameters (*e.g.*, thermal conductivities, rock densities) in the same ASCII-file format that will be read by ***Shells***. There is an option to also use a grid-file (.GRD) of the ages of oceanic lithosphere.

To run ***OrbData5***, you need: (1) a finished .FEG from ***OrbWin***; (2) a grid-file (.GRD) of topography from some DEM; (3) a grid-file (.GRD) of heat-flow; (4) a grid-file (.GRD) of teleseismic travel-time anomalies of mantle origin; (5) a grid-file (.GRD) of crustal thicknesses; and (6) some physical parameters (*e.g.*, thermal conductivities, rock densities) in the same ASCII-file format that will be read by ***Shells***. There is an option to also use a grid-file (.GRD) of the ages of oceanic lithosphere.

My gridded-data (.GRD) file format is explained: http://peterbird.name/guide/grd_format.htm

There are several .GRD files already posted at http://peterbird.name which may be useful:

Content:	Filename:	In folder:
Seafloor ages [*Müller et al.*, 1997]	age_1p5.grd	/oldFTP/neotec/Shells/Earth5/Global_GRDs/
Low-resolution (20' = 1/3°) topographic DEM [NGDC}	ETOPO20.grd	/oldFTP/neotec/Shells/Earth5/Global_GRDs/
Medium-resolution (5' = 1/12°) topographic DEM [NGDC]	ETOPO5.zip	/oldFTP/neotec/Shells/Earth5/Global_GRDs/

Crustal thickness [*Bassin et al.*, 2000]	CRUST2.grd	/oldFTP/neotec/Shells/Earth5/Global_GRDs/
Mantle travel-time anomalies of vertically-traveling S-waves [*Ritsema & van Heijst*, 2000]	delta_ts.grd	/oldFTP/neotec/Shells/Earth5/Global_GRDs/
Global heat-flow (Earth**2** version)	mean_Q.grd	/oldFTP/neotec/Shells/Earth**2**/Global_GRDs/

The process of filling-in nodal data for a ***Shells***-type .FEG is also illustrated in Steps 12~15 of my online Guide, at: http://peterbird.name/guide/home.htm

The protocol for both ***OrbData*** and ***OrbData5*** follows these rules: (1) If elevation is exactly zero at any node, the correct elevation is taken from the topographic .GRD file. (2) If heat-flow is exactly zero at any node, the correct heat-flow is computed (by slightly different methods in these two programs). (3) Other nodal values, following the heat-flow, are computed and written; any existing values of these variables are ignored and overwritten.

This protocol means that you can manually reduce the heat-flow of a few nodes in ***OrbWin***, and then either ***OrbData*** or ***OrbData5*** will respect and keep your edits when it is re-run. This can be important for fixing non-physical "landslide" problems in steep, high-heat-flow portions of the model.

IV.F.2. Continuum flexibility parameter μ (for ***NeoKinema***, ***Restore***)

NeoKinema and ***Restor***e can each (optionally) read a single nodal datum, which is "mu" (μ). This is the prior (pre-modeling) estimate of permanent deformation rates in unfaulted areas [*Bird*, 2009]. This is expressed as the standard deviation of the strain-rate that is nominally zero. The units are "per second" or s^{-1}. In regions of flat-lying Paleozoic and Mesozoic sediments where any deformation would quickly become apparent, I suggest a low value such as 1.0E-16 /s. However, the complex and rapidly-deforming California region, *Bird* [2009] found the optimal (self-consistent) value to be about 5.0E-16 /s.

In early modeling experiments, you may wish to have μ be uniform across your .FEG. In this, case, you can leave all the values in the .FEG file left to the default (zero) value, and specify the uniform level of μ as a parameter read by ***NeoKinema***.

Thus, including μ values in your .FEG is only appropriate when you desire lateral variations. In such cases, use commands III.E.7. **Elevation(OR mu)** and III.E.8. **Block Set mode** to set them.

V. RELATED UTILITY PROGRAMS

The following little utility programs may save you a lot of time and trouble, as you work to produce good basemap (.DIG) files and good F-E grid (.FEG) files. Most of these simple programs do not have graphical user interfaces or a menu of commands; typically they just present white text on a black screen, and ask you to supply some filenames: names of existing files for input, and a new filename to be used for the output. These simple programs also do not have manuals. You can find out what they do by reading the Fortran (.f90, or .for) source code files, which often contains useful comments. (If you know some Fortran and have access to a compiler, you can also make changes to suit special circumstances.)

It will be much simpler to run these utility programs if their executable (.exe) files, their input file(s), and their output file(s) are all in the same folder (subdirectory). Then, paths do not need to be included when typing filenames. (You can always move the output file(s) after the utility program finishes.)

V.A. Programs for creating basemap .DIG files

V.A.1. ***Graticule*** to create a grid of parallels and meridians

Graticule produces a simple .DIG file containing only parallels (lines of constant latitude) and meridians (lines of constant longitude). Depending on the limits you choose, the graticule can be regional or global. You can choose to subdivide the steps, in order to better approximate the small-circle shapes of most parallels. A graticule-type basemap can be helpful when setting the limits for **TileRegionalGrid** in ***OrbWin***, if you want your grid edges to follow parallels and meridians. (You can also tidy-up the edges of any .FEG with **AdjustNode** in ***OrbWin***.)

V.A.2. ***GPS_2_DIG*** to plot geodetic benchmark locations

Assuming you have some GPS benchmark locations (and interseismic velocities, and uncertainties) in my .GPS format (http://peterbird.name/guide/gps_format.htm), utility ***GPS_2_DIG*** will create a .DIG file consisting of little cross-hairs ("+"), inside little triangles, to mark the benchmark locations. This will help you to design "fault corridors" (in a ***NeoKinema***-style .FEG) that avoid as many benchmark locations as possible. Or, if you are creating fault elements for a ***Shells***-type .FEG, this feature can help you make sure that each benchmark is on the correct side of each fault. (This is especially important if the faults are creeping.) Your only run-time option is the size of the symbols, expressed in kilometers.

V.A.3. ***Digitize*** to capture lines (*e.g.*, fault traces) from printed maps as Cartesian *(x, y)* data

This program acts as "secretary" for a digitizing tablet, by recording all the positions you click on and storing them as Cartesian *(x, y)* coordinate pairs in my .DIG file format. Typically, you would have a geologic map taped to the digitizing tablet, and you would be clicking points along the traces of faults (or, perhaps, along a coastline). The program performs one linear coordinate transformation from the native (iRow, jCol) integer coordinates of the digitizing tablet (in units of 0.001"?) to real-number, Earth-sized *(x, y)* coordinates, based on coordinate values for reference points that you both click-on and then type-in. The units of *(x, y)* may be English or metric, but they must express the actual sizes of things on the real planet; thus, *x* and *y* are typically big numbers. The next step, of converting from Cartesian *(x, y)* coordinates to *(longitude, latitude)* coordinates, can be handled by my program ***Projector***, described below.

Digitize was written in Microsoft Basic 7 for the DOS operating system. Thus, its memory-address model is 16-bit, and file sizes are limited to 64 KB. Also, filenames are limited to 8+3 capital-letter characters (no spaces!) as in DOS. It takes input from the digitizing tablet through a serial cable, plugged into a serial (RS-232, or COM) port in a desktop computer. These days, serial ports are getting rare, and you might have to buy a $25 expansion card to add one to your computer. (Also, you might need a desktop computer, as laptops typically have no room for expansion cards.) The serial cable will probably be more expensive; but perhaps your local IT technician has an old one in a closet somewhere? The digitizing tablet itself might cost up to $3,000! (Mine is a 36" x 24" Drawing Board VI from the GTCO CalComp Peripherals company.)

The good thing about ***Digitize*** is that it is totally "agnostic" about the brand of digitizer, the serial communication parameters such as baud-rate, parity, data-bits, and stop-bits, and the format of the numbers transmitted (as long as they come in via serial port). There are formatting and testing menus to establish and check the communications before you start to work.

The ***bad*** thing about ***Digitize*** is that the last computers that were able to run it as intended (as a stand-alone, full-screen interactive program) were Windows NT and Windows XP computers with 1024×768-pixel monitors. On later versions of Windows (7, 8, 8.1, 10) the program only runs inside the shareware DOS-emulator program (virtual computer sandbox) called ***DosBox***, and then the display is tiny. (Perhaps this issue can be fixed with ***DosBox*** settings? Anyone?)

Thus, an important alternative is to digitize with modern equipment and modern software, and just convert the output file format afterward. This alternative is described in the next section.

V.A.4. ***SHP_toFrom_xyDIG*** to convert digitized lines from ESRI Shapefile to my .DIG format

If your computer runs a modern version of Windows (e.g., ***Windows 10***), you may need to use modern software for digitizing. Both Golden Software and ESRI publish digitizing programs for modern Windows computers. Sometimes I use ***Didger5*** from Golden Software. This program requires a (pre-installed) device-driver program (such as ***TabletWorks***) which communicates using the industry-standard WinTab API (Application Programming Interface). For example, when I purchased my Calcomp Drawing Board VI digitizing tablet, it came with a USB port for output, and the free device-driver program ***TabletWorks.*** The digitizer sends data through the USB cable to ***TabletWorks*** inside the computer, and then ***TabletWorks*** communicates with ***Didger5*** through the WinTab API.

I have written a short "white-paper" on setting-up such a system, which is posted at URL of:
http://peterbird.name/oldFTP/Digitise/Digitizing_tips_2019.pdf

If you choose this route, then you will need to output your polylines (*e.g.*, fault traces and/or state-lines and/or coastlines) with command Export / Save As Type: ESRI Shapefile (*.SHP). With every output shapefile you will also obtain a dBase (.DBF) file with the names that you assigned to your polylines. Then, you should run my utility program ***SHP_toFrom_xyDIG*** to convert these two files into a single file in my .DIG format. The point coordinates will still be Cartesian (x, y) coordinates, probably in units of meters. Then, proceed to use ***Projector*** in the next step:

V.A.5. ***Projector*** to convert digitized lines from *(x, y)* to *(longitude, latitude)* data

Oddly, ***Projector*** does not have a graphical user interface. It runs with white text displayed on a black screen. The program asks a series of questions, whose answers will allow it to define the functional mapping from *(x, y)* to *(longitude, latitude)* coordinates:

-type of map projection (from a menu of 10 types);
-standard parallels (if appropriate);
-projection point (least-distorted point on the map; typically near its center);
-values of your (*x*, *y*) coordinates at this projection point;
-orientations (azimuths) of your ***x*** and ***y*** axes at this projection point; and
-radius of the planet.

Then it reads one (or more) of your .DIG files in *(x, y)* format, and produces one (or more) .DIG files in *(longitude, latitude)* format.

There is an option to produce a very basic map display in Adobe Illustrator .AI format. You can open and view this (assuming that you have purchased Adobe Illustrator, Academic Edition) to check for any errors in the parameters describing the conversion between coordinate systems. Step 5 of my on-line Guide to modeling with ***Shells*** (http://peterbird.name/guide/home.htm) shows an example.

I confess that ***Projector*** is not as accurate as the (proprietary) coordinate-conversion utilities found inside GIS-type codes like Golden Software and ESRI products. This is because ***Projector*** approximates the planet as a sphere, while actually the Earth is slightly elliptical due to its rotation. *Thus, you may prefer to digitize faults inside a proprietary GIS product if you have access to one that supports a digitizing tablet.*

V.A.6. ***ReverseDig*** to reverse the sense of progress along any digitized line

My convention (which is enforced inside ***NeoKinema*** and ***NeoKineMap*** and ***Restore*** and ***RetroMap***) is to digitize faults so that their dip (if not vertical) is to the left as one progresses along the fault trace. Some other software packages (and the USGS) use the opposite convention, that the dip (if not vertical) is always to the right. Thus, it may be necessary to reverse the order of points along certain lines in a .DIG file. Fortunately, ***ReverseDig*** can do this (selectively) to avoid any need for re-digitizing. It reads an existing .DIG file, summarizes the length and overall direction of each line, and then asks you whether it should be reversed.

V.A.7. ***COPY*** to combine any number of .DIG files into one basemap

OrbWin only allows a single basemap (.DIG) file to be shown at one time. If you want to combine several different .DIG files that are already in *(longitude, latitude)* format, the simplest way is to:

(1) Open a “Command Prompt” window. In Windows 10, choose StartMenu / All apps / Windows System / Command Prompt. This text-based user interface is similar to that of the old DOS, and also similar to many older versions of Unix. (The main difference is that Unix uses character “/” in paths, while DOS and Windows use “\”.)

(2) Navigate to the folder where you have your data files, with command **CD**. For example, I would enter "**CD** C:\Users\pbird\Desktop" and press **Enter**. Optionally, you can then type **DIR** and press **Enter** to get a list of files in the current directory.

(3) Combine your .DIG files like this:

COPY graticule.DIG+faults.DIG+coastlines.DIG+GPS.DIG everything.DIG

Here, "everything.DIG" is a new file to be produced by combining all the others.

(4) Type **EXIT** and press **Enter** to close the Command Prompt.

V.B. ***OrbNumber***, for re-numbering the nodes in any ***OrbWin*** .FEG file

When you create a new .FEG file with OrbWin, the nodes are assigned integer index numbers in the order that they are created. If this numbering is not replaced, then there may be tight connections (elements in common) between some of your lowest-numbered nodes and some of your highest-numbered nodes. Then, within one of my F-E programs (***Shells***, ***NeoKinema***, or ***Restore***), the coefficient matrix of the linear system will be a full matrix. This leads to very large demands on computer memory (to store the matrix) and computer execution time (to solve the linear system).

OrbNumber can automatically re-number the nodes so that nodes are listed in successive concentric circles about the optimal starting point. (To find the optimal starting point, it makes many renumbering attempts.) This reduces the "bandwidth" of the eventual coefficient matrix in the following F-E program, giving exactly the same solution with far fewer resources. Thus, ***OrbNumber*** should be run on any .FEG file produced by ***OrbWin***, whenever you feel it is "done" and ready to use.

OrbNumber does not offer the user any options or choices, except for the filename of the existing .FEG which is to be renumbered, and also names for the new files that it will create. The run time will typically be a few minutes to an hour.

An additional benefit of running ***OrbNumber*** is that it will produce an ordered list of boundary nodes, going counterclockwise about the perimeter of your F-E grid. I like to call this the "perimeter" (.PER) file. You will need this list to assign boundary conditions for ***Shells***, ***NeoKinema***, or ***Restore***. So, save this new file next to your renumbered (and re-named) .FEG. (The only time you will not get this second output file is when you process a global grid—it has no side boundaries!)

V.C. Programs to compute nodal data

Nodal data are additional variable values attached to each node, and listed in the .FEG file following the *(longitude, latitude)* coordinates of the node. See section II.B. **Finite-Element Grid (.FEG) file format**.

The visualization (and minor manual editing) of nodal data in ***OrbWin*** was discussed in manual sections III.E.7 and III.E.8, above. Once any nodal-data field is complete, it can be visualized with sophisticated mapping programs; see section V.D. **Making maps with graphical post-processing programs**, next.

V.C.1. ***OrbData*** to compute simple lithosphere structures (for ***Shells***)

F-E grids intended for use in ***Shells*** require at least these 4 nodal data for each node: elevation, heat-flow, crustal thickness, & mantle-lithosphere thickness (which does not include the crustal thickness). All are expressed in SI units: m, W/m^2, m, & m. Zero values appear if these quantities have not yet been estimated, as when new nodes have recently been created.

OrbData computes this simple, 4-variable lithosphere structure that was used in early version of ***Shells*** (and is still available, by backward-compatibility). The elevation of each node (if not already specified) is interpolated from a topographic DEM that is read in the form of a .GRD file. The heat-flow of each node (if not already specified) is interpolated from a second .GRD file. Then, two assumptions are used to compute crustal thickness and mantle-lithosphere thickness: (1) Local isostasy, or constant mass in each column above some reference depth in the asthenosphere; and (2) Steady-state heat conduction, in which the heat-flow into the base of the lithosphere, plus the extra heat-flow produced by internal radioactive heat production, is equal to the heat-flow coming out of the top.

This protocol means that you can manually reduce the heat-flow of a few nodes in ***OrbWin***, and then ***OrbData*** will respect and keep your edits when it is re-run. This can be important for fixing non-physical "landslide" problems (in topographically-steep, high-heat-flow portions of the model) by reducing the local heat-flow. However, if you make manual edits directly to crustal thickness and/or mantle-lithosphere thickness, this work will be overwritten the next time you run ***OrbData***.

The calculation is mildly nonlinear, so an iterative scheme is used. ***OrbData*** will write a warning to its log-file about any node where the computation fails (or hits some internal limit, such as minimum heat-flow or minimum or maximum crustal thickness). Regardless of any warning message, the result is usually "reasonable". Try visualizing these values in ***OrbWin***, or with ***FiniteMap***.

To run ***OrbData***, you need: (1) a finished .FEG from ***OrbWin***; (2) a grid-file (.GRD) of topography from some DEM; (3) a grid-file (.GRD) of heat-flow; and (4) some physical parameters (*e.g.*, thermal conductivities, rock densities) in the same ASCII-file format that will be read by ***Shells***. There is an option to also use a grid-file (.GRD) of the ages of oceanic lithosphere.

My gridded-data (.GRD) file format is explained: http://peterbird.name/guide/grd_format.htm

Several existing .GRD files are already posted at http://peterbird.name . Please see manual section IV.F.1 above for details.

The process of filling-in nodal data for a ***Shells***-type .FEG is also illustrated in Steps 12~15 of my online Guide, at: http://peterbird.name/guide/home.htm

V.C.2. ***OrbData5*** to compute complex lithosphere structures (for ***Shells***)

*To avoid repetition, I will assume that you have read the **OrbData** section, above.*

The difference between ***OrbData5*** and ***OrbData*** is that the newer and more complex program determines the crustal thickness and mantle-lithosphere thickness from two additional .GRD files that it reads as input. This allows the use of actual crustal thicknesses (*i.e.*, from seismic refraction or converted-wave studies) and more realistic model mantle-lithosphere thicknesses (in the oceans: from crustal age; in the continents: from teleseismic travel-time anomalies, after correction for crustal structure).

OrbData5 still assumes isostasy, so to accommodate a known crustal thickness, it invokes a new degree of freedom: a lithospheric density anomaly of compositional (not thermal) origin, which becomes a new nodal variable. In order to remain reasonable, this new variable is limited in amplitude to no more than ±50 kg/m^3. Wherever this limit is hit, local lithospheric isostasy is compromised. (However, this is often appropriate. Many subduction trenches are depressed by

the weight of attached slabs. Also, some hot-spots like Iceland are held up by rising plumes in the asthenosphere.)

OrbData does not assume steady-state geotherms. In fact, most oceanic lithosphere is not in equilibrium because it is cooling. Many cratonic regions may also be cooling, since their last orogeny. The steady-state geotherm is perturbed by one additional term, which has a quadratic form within the lithosphere, and zero values at both top and bottom. The amplitude of this non-steady-state quadratic term becomes the 6^{th} nodal variable. Its units are $°K/m^2$ (or $°C/m^2$, which amounts to the same thing).

The Earth5 model described by *Bird et al.* [2008] was computed with ***OrbData5***. All earlier Shells models (by myself or others) were computed with ***OrbData***. If you have the time and energy, I suggest computing alternative lithosphere structures with both of these programs, and then deciding which looks more reasonable in each case.

V.D. Making maps with graphical post-processing programs

Each of my major F-E programs (***Shells***, ***NeoKinema***, & ***Restore***) has its own dedicated graphical post-processing program, as described below. These "post-processors" can also be used to make advance plots of the input files for an F-E simulation, such as .FEG files. These 3 map-making programs have very similar structures and also similar user-interfaces, because they are all based on common Fortran 90 MODULEs (DAdobe_Illustrator, DMap_Projections, & DMap_Tools—also available separately on my web site).

Each of these programs has an interface of white text on a black background. There is no graphical display while these programs are running. Instead, graphics are written as ASCII text (in the mysterious Adobe Illustrator dialect of the old PostScript open-standard language) into an .AI file, which you can then open and edit with Adobe Illustrator. (A little file of boiler-plate .AI descriptions, named "AI7frame.AI", must be available to each of my map-making programs as it runs.)

Each program allows you to select the size and shape of the (virtual) paper it will plot on, the margins, and the use of color (if desired). Then, it allows you to select from among 10 map projections, and define the center of the desired map (the "projection point") and the map scale, direction of North, and so on. Finally, it asks the user to select map components from two menus: "mosaics" and "overlays".

In these 3 programs, a "mosaic" is any display that fills the map window with colors (or grayscales, or patterns), so that it is effectively opaque. (Although the user could choose to plot more than one mosaic, this is rarely successful, because only the last one, on the top of the stack, will be visible.) Many mosaics can be produced either as sets of colored polygons (with a limited palette of distinct colors), or alternatively as embedded rectangular TIFF-style bitmaps. The TIFF standard gives the programmer (me) unlimited access to the RGB color components of each pixel, opening the possibility of shaded-relief as well as continuously-variable color. The combination of continuous-color with shaded-relief can be quite impressive! (Note that the number and size of pixels in one of my .AI files has no relation to the number of pixels displayed on your computer monitor. The suggested defaults give one pixel per "point" (1/72 of an inch) on the (virtual) paper plot, but you are free to request more pixels for a higher-resolution display (at the cost of longer run-time and a larger map file).

An "overlay" is any display that consists only of lines and text (not areas), so that it is not opaque. Examples of overlays include coastlines and/or state-lines (in a .DIG file), neotectonic or geodetic velocity vectors, stress tensors, and earthquake epicenters or focal mechanisms. Many maps will have more than one overlay.

Each run of a map-making program produces only a single graphical page (one map). Each of these 3 graphics programs tries to remember your previous choices, and present them as suggested responses (shown in square-brackets, "[ ]"). If your previous choice is still appropriate, then just press **Enter**. Or, you can type a different response. Notice that some questions expect a text-string answer (like a filename); others expect a real number (like "6371.E3"); others expect an integer from a list, and others expect a True/False (or Yes/No) answer. If you have any trouble, it may be because you are supplying the wrong class of answer (such as an integer selection, when True/False is expected). The program will prompt you to try again.

The page- and map-projection choices made during any run of these programs is stored in a small ASCII file named "Map_Tools.INI" (which is human-readable, even editable). I like to save copies of .INI files that worked out particularly well, along with the map .AI files, so that a number of maps can be produced with exactly the same map projection and window area. (Such co-registered maps are more easily understood by an audience at a PowerPoint talk. Also, you can switch rapidly back-and-forth between two co-registered PowerPoint slides to show subtle differences between two similar models.)

Note that Adobe Illustrator gives you the option to save any finished map page in a variety of formats, including .EPS, .PDF, .JPG, and .GIF. Therefore, my codes do not offer those options, and only produce .AI format.

For historical reasons, each of my graphical programs allows the user to mix input files in *(longitude, latitude)* format with other input files in Cartesian *(x, y)* coordinates. (Each projection must be defined at the start of the run; only one Cartesian coordinate system is allowed per run.) This option can be confusing for beginners! If you are able, I would recommend converting any Cartesian *(x, y)* files to *(longitude, latitude)* before starting to make maps.

V.D.1. ***FiniteMap*** for plotting grids, lithosphere structure, & basemaps (for ***Shells***)

Historically, ***FiniteMap*** is my replacement for the older graphical post-processor ***OrbMapAI***. Two pages of my online Guide (http://peterbird.name/guide/graphics/introduction.htm and http://peterbird.name/guide/graphics/differences.htm) give a table contrasting these 2 programs for plotting input and output of ***Shells***. ***FiniteMap*** is more versatile, more modern, and more fully-supported. Another consideration is that only ***FiniteMap*** can properly display the more modern lithosphere structure models computed by ***OrbData5***.

The following can be easily displayed by ***FiniteMap***:

- any basemap (.DIG) file (overlay);
- locations of active volcanoes from Smithsonian data file (overlay);
- locations and velocities of geodetic benchmarks from .GPS file (overlay);
- earthquake epicenters and/or focal mechanisms from .EQC file (overlay);
- seafloor-spreading-rate data from file (overlay);
- plate boundaries, Euler poles, heave-rates, and velocities from the PB2002 model (overlays);
- SKS splitting azimuth and time data from file (overlay);
- any .GRD file (mosaic);
- F-E grid (.FEG) files from ***OrbWin*** and ***OrbNumber***;
- any of 6 nodal data fields from ***OrbWin*** and/or ***OrbData[5]*** (mosaic);

- some derived products of nodal data, like Moho temperature, total lithosphere thickness, or pressure anomaly below the lithosphere (mosaic);
- assumed velocities of the deeper mantle under the model (mosaic and/or overlay);
- shear tractions on the base of the lithosphere from ***Shells*** (mosaic and/or overlay);
- velocities predicted by ***Shells*** (mosaic and/or overlay);
- fault slip-rates or heave-rates from ***Shells*** (overlay);
- strain-rates predicted by ***Shells*** (mosaic and/or overlay);
- vertical-axis rotation rates predicted by ***Shells*** (mosaic);
- vertically-integrated stress anomalies from ***Shells*** (mosaic and/or overlay);
- principal stress azimuths, both from the ***Shells*** model & from data (overlays);
- logarithm of the effective viscosity from ***Shells*** (mosaic);
- equivalent-nodal reaction forces to imposed boundary conditions (overlay).

V.D.2. ***NeoKineMap*** for plotting grids and basemaps (for ***NeoKinema***)

NeoKineMap is similar to ***FiniteMap*** (see above), but is customized to read the input and output file formats of my ***NeoKinema*** F-E program. In particular, it reads the parameter-input text file "p_*{modelNameToken}*.NKI" that was used in the ***NeoKinema*** run (or will be used, shortly) and from this it reads (or infers) the names of other related files. This saves a lot of tedious typing of filenames, and also reduces frustration due to errors.

Because of the limits any "kinematic" or "inverse" F-E code like ***NeoKinema*** (see Section II.A), no plots of quantitative stresses, asthenospheric tractions, or boundary nodal reaction forces are available. However, ***NeoKinema*** models include two distinct predictions of geodetic velocities and associated nodal velocities and element strain-rates: short-term(interseismic; comparable to most GPS data), and long-term-average (comparable to plate velocities).

The following can be easily displayed by ***NeoKineMap***:

- any basemap (.DIG) file (overlay);
- fault traces from f_*{datasetNameToken}*.DIG (overlay);
- fault heave rates from geologic data file f_*{datasetNameToken}*.NKI (overlay);
- locations of active volcanoes from Smithsonian data file (overlay);
- locations and velocities of geodetic benchmarks from .GPS file (overlay);
- earthquake epicenters and/or focal mechanisms from .EQC file (overlay);
- plate boundaries, Euler poles, heave-rates, and velocities from the PB2002 model (overlays);
- any .GRD file (mosaic);
- F-E grid (.FEG) files from ***OrbWin*** and ***OrbNumber***;
- nodal data field (μ) from ***OrbWin*** (mosaic);
- long-term or short-term velocities predicted by ***NeoKinema*** (mosaic and/or overlay);
- long-term-average fault slip-rates or heave-rates from ***NeoKinema*** or from data (overlay);
- long-term or short-term strain-rates predicted by ***NeoKinema*** (mosaic and/or overlay);
- long-term vertical-axis rotation rates predicted by ***NeoKinema*** (mosaic);
- principal stress azimuths, both from the ***NeoKinema*** model & from data (overlays);
- long-term seismicity-rate predictions from ***Long_Term_Seismicity*** (overlay).

V.D.3. ***RetroMap*** for plotting deforming grids & basemaps through time (for ***Restore***)

RetroMap2 and ***RetroMap3*** (for corresponding versions of ***Restore***) are similar to ***NeoKineMap*** (see above), but are customized to read the input and output file formats of my ***Restore*** F-E programs. In particular, they read the parameter-input text file that was used in the ***Restore*** run (or will be used, shortly) and from this they read (or infer) the names of other related files. This saves a lot of tedious typing of filenames, and also reduces frustration due to errors.

Because of the lack of geodetic data in the geologic past, display of geodetic benchmarks and their velocities is not included in ***RetroMap***. However, their place is filled by displays of paleomagnetic inclination and declination anomalies, and the positions and lengths of restored cross-sections.

RetroMap can display the positions of model features at any geologic time covered by the ***Restore*** computation. With a very large number of RetroMap plots, one could conceivably create a retro-deformation (or forward-deformation) movie. While I have never gone that far, you can see some "film-strip" sequences of model variables at 5-m.y. intervals on the web page for *Bird* [199b]: http://peterbird.name/publications/1998_Laramide/1998_Laramide.htm

The following can be easily displayed by ***RetroMap3***:

- any basemap (.DIG) file (mosaic or overlay);
- fault traces in present or past positions (overlay);
- paleomagnetic inclination & declination anomalies (overlay);
- paleo-stress-directions, from data, or interpolated by ***Restore*** (overlay);
- balanced cross-section locations, with restored lengths (overlay);
- any .GRD file (mosaic);
- F-E grid (.FEG) files from ***OrbWin*** and ***OrbNumber***;
- nodal data field (μ) from ***OrbWin*** (mosaic);
- paleo-velocities predicted by ***Restore*** (mosaic and/or overlay);
- net displacement predicted by ***Restore*** (mosaic and/or overlay);
- paleo-fault slip-rates or paleo-heave-rates from ***Restore*** or from data (overlay);
- net heave on faults from ***Restore*** (overlay);
- strain-rates predicted by ***Restore*** (overlay);
- net vertical-axis rotation predicted by ***Restore*** (mosaic);
- net strain (natural dilatation) predicted by ***Restore*** (mosaic);

VI. REFERENCES CITED

Bassin, C., G. Laske, and G. Masters [2000] The current limits of resolution for surface wave tomography in North America (abstract), *Eos Trans. AGU, 81*, F897.

Baumgardner, J. R. [1983] A three-dimensional finite element model for mantle convection, Ph.D. dissertation, University of California Los Angeles, 271 pages.

Bird, P. [1998] Kinematic history of the Laramide orogeny in latitudes 35 to 49 degrees North, western United States, *Tectonics, 17*(5), 780-801.

Bird, P. [1999] Thin-plate and thin-shell finite element modeling programs for forward dynamic modeling of plate deformation and faulting, *Comput. Geosci., 25*(4), 383-394.

Bird, P. [2002] Stress-direction history of the western United States and Mexico since 85 Ma, *Tectonics, 21*(3), 10.1029/2001TC001319.

Bird, P. [2003] An updated digital model of plate boundaries, *Geochem. Geophys. Geosyst., 4*(3), 1027, doi:10.1029/2001GC000252.

Bird, P. [2009] Long-term fault slip rates, distributed deformation rates, and forecast of seismicity in the western United States from fitting of community geologic, geodetic, and stress direction datasets, *J. Geophys. Res., 114*(B11403), doi: 10.1029/2009JB006317.

Bird, P., Z. Liu, and W. K. Rucker [2008] Stresses that drive the plates from below: Definitions, computational path, model optimization, and error analysis, *J. Geophys. Res., 113*(B11), B11406, doi:10.1029/2007JB005460, plus digital appendices.

Chaytor, J. D., C. Goldfinger, R. D. Dziak, and C. G. Fox [2004] Active deformation of the Gorda plate: Constraining deformation models with new geophysical data, *Geology, 32*(4), 353-356.

DeMets, C., R. G. Gordon, D. F. Argus, and S. Stein [1990] Current plate motions, *Geophys. J. Int., 101*, 425-478.

Field, E. H., G. P. Biasi, P. Bird, T. E. Dawson, K. R. Felzer, D. D. Jackson, K. M. Johnson, T. H. Jordan, C. Madden, A. J. Michael, K. R. Milner, M. T. Page, T. Parsons, P. M. Powers, B. E. Shaw, W. R. Thatcher, R. J. Weldon, II, and Y. Zeng [2013] Unified California Earthquake Rupture Forecast, version 3 (UCERF3)-The time-independent model, *U.S. Geol. Surv. Open-File Rep., 2013-1165* (*Cal. Geol. Surv. Spec. Rep. 228*, and *Southern California Earthquake Center Pub. 1792*), 97 pages; http://pubs.usgs.gov/of/2013/1165/.

Glatzmaier, G. A., and G. Schubert [1993] Three-dimensional spherical models of layered and wholemantle convection, *J. Geophys. Res., 98*(B12), 21,969-21,976.

Kong, X., and P. Bird [1995] SHELLS: A thin-plate program for modeling neotectonics of regional or global lithosphere with faults, *J. Geophys. Res., 100*(B11), 22,129-22,131.

Liu, Z., and P. Bird [2002] North America plate is driven westward by lower mantle flow, *Geophys. Res. Lett., 29*(24), 2164, doi:10.1029/2002GL016002.

Minster, J. B., and T. H. Jordan [1978] Present-day plate motions, *J. Geophys. Res., 83*(B11), 5331-5354.

Müller, D., W. R. Roest, J.-Y. Royer, L. M. Gahagan, and J. G. Sclater [1997] Digital isochrons of the world's ocean floor, *J. Geophys. Res., 102*(B2), 3211-3214.

Petersen, M. D., Y. Zeng, K. M. Haller, R. McCaffrey, W. C. Hammond, P. Bird, M. Moschetti, Z. Shen, J. Bormann, and W. Thatcher [2014a] Geodesy- and geology-based slip-rate models for the Western United States (excluding California) national seismic hazard maps, *U.S. Geol. Surv. Open-File Rep., 2013-1293*, 80 pages; http://dx.doi.org/10.3133/ofr20131293.

Petersen, M. D., M. P. Moschetti, P. M. Powers, C. S. Mueller, K. M. Haller, A. D. Frankel, Y. Zeng, S. Rezaeian, S. C. Harmsen, O. S. Boyd, N. Field, R. Chen, K. S. Rukstales, N. Luco, R. L. Wheeler, R. A. Williams, and A. H. Olsen [2014b] Documentation for the 2014 update of the United States National Seismic Hazard Maps, *U.S. Geol. Surv. Prof. Pap., 2014-1091*, 255 pages.

Ritsema, J., and H. J. van Heijst [2000] Seismic imaging of structural heterogeneity in Earth's mantle: Evidence for large-scale mantle flow, *Science Progress, 83*, 243-259.