```
In[1]:= startTime = AbsoluteTime[];
```

SLAM-code-2017.nb

Begun 5 August 2004; Revised 25 November 2017

Introduction

This version of the SLAM code is meant to project six planes to the ground surface and to map the intersection of those planes and the ground surface. The six planes include the mean fault-plane solution, as well as the planes that bound the uncertainty volume that reflects the uncertainty in the focal location and the uncertainty in the dip angle and dip direction.

Prepare the Earthquake Data File

The **eqDataIn** is an Excel (.xls) spreadsheet that contains the earthquake data necessary for the analysis. The first record (row) of the spreadsheet contains words that describe the contents of each column, and each other record (row) contains the data for one earthquake focal mechanism. The data types are as follows, starting with the left-most column (column A): year, month, day, hour, minute, seconds, decimal latitude (°), decimal longitude (°), depth (km), greatest horizontal error (EH1; km), least horizontal error (EH2; km), azimuth of EH1 axis (°), vertical error (km), dip direction of the fault plane (°), dip angle of the fault plane (°), rake of hanging-wall slip vector on the fault plane (°), strike uncertainty (°), dip uncertainty (°), rake uncertainty (°), and optional other columns containing ID or other information.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Year	Month	Day	Hour	Minute	Seconds	Lat	Long	Depth	EH1	EH2	EH1Az	EZ	Mag	NP1_DipDir	NP1_DipAng	NP1_Rake	Strike uncert	Dip uncert	Rake uncert	NCEDC ID#
2	1983	7	3	15	8	20	39.412	-120.206	11.05	0.3	0.3	45	0.8	4	300	80	20	8	13	30	1097977
3	1988	11	7	10	11	54.915	39.40575	-120.21545	10.414	0.027	0.019	24	0.037	3.1	240	20	-70	20	13	25	126552
4	1992	3	24	11	38	54.84	39.40673	-119.96681	8.829	0.433	0.027	66	0.135	3.5	110	75	0	18	45	60	259005

Input

■ Import the Earthquake Data File

To modify this code so that it can be used for your dataset, do the following.

1. Enter the following as an input line (under the black box, below):

```
eqDataIn = Import[];
```

2. Put your cursor between the square brackets in the **Import[]** statement and click to establish the

insertion point we will need in the next step.

3. Go to the **Insert** menu, select **File Path**, navigate to the correct input data file and choose it, and the correct file path will be inserted at the cursor. In this example, the Excel file is located on the desktop of Vince's computer, and so *Mathematica* will insert the path "**/Users/vince/Desktop/Input_EQ_datafile.xls**" between the brackets. The path will be different for every different file, and on every different computer.

```
eqDataIn = Import["/Users/vincecronin/Desktop/Strasser_Input_EQ_-
datafile.xls"];
```

You can make sure it is an input line by clicking on that line's bracket on the far right edge of this window, going to the **Format** menu, selecting the **Style** submenu, and then choosing **Input**

In[2]:=
```
eqDataIn =
  Import["/Users/vincecronin/Desktop/Strasser_Input_EQ_datafile.xls"];
```

This won' t work unless you follow directions and add the appropriate input line above this line. Your input line should look something like the example:

```
eqDataIn = Import["/Users/vincecronin/Desktop/Strasser_Input_EQ_-
datafile.xls"];
```

The imported dataset will have to be modified so that it has the correct dimensions for this analysis, which is accomplished with the following code that defines the list **eqData**:

In[3]:=
```
eqData = Flatten[eqDataIn, 1];
```

Modifying the digital elevation model (DEM) so that the resulting hillshade map has an esthetically pleasing range of tones or colors

The range of tones or colors on the hillshade map generated by this code varies with the difference (relief) between the smallest and largest elevation value in the DEM dataset. If there are exceptionally small "nodata values" in the elevation dataset, the hillshade map might have a very small range of tones or colors -- it might look flat. If there are no "nodata values", a relatively flat ground surface might look like a mountain range on the hillshade map. The trick is to intercede and re-define the nodata values (if any) as being much closer to the ground-surface elevations, or to define one of the ground-surface elevations as a value that is much lower than the rest but not as low as the original "nodata value."

In the fragment of the DEM dataset shown above, which has no "nodata" values in the elevation matrix, the very first value in the elevation data (row 7, column 1) was changed to 0. By trial and error, you can adjust this bogus value to produce the desired range of tones or colors in the hillshade map.

Import the digital elevation model (DEM)

The ground surface is represented by a digital elevation model. The DEM file is an ASCII text file (that uses the generic data tag ".dat") that has a 6 line 2 column header followed by a matrix of many rows and columns of elevation data. The geographic grid used is a UTM grid (meters). The horizontal position of any given elevation datum is provided by its position in the matrix, relative to the UTM coordinates of the lower left

corner and the cellsize -- the horizontal distance between adjacent matrix nodes. The gridded DEM datafile looks like the following:

```
ncols 938
nrows        1164
xllcorner    732209.0021288905
yllcorner    4.356019342050166e6
cellsize     28.1312033770362
NODATA_value   -9999
0.    1788.43    1795.223   1801.482   1804.554   1807.982   1810.846
```

In a DEM that has no elements that have a null datapoint equal to the "NODATA_value," we use the first position in the data grid below the header to condition the dataset so that it displays a hillshade image with sufficient contrast. The DEM sampled above has a zero in the first grid position.

In[4]:=
```
mydata = Import["/Users/vincecronin/Desktop/thinDVFZCrop.dat"];
```

Modifying the digital elevation model (DEM) so that the resulting hillshade map has an esthetically pleasing range of tones or colors

The range of tones or colors on the hillshade map generated by this code varies with the difference (relief) between the smallest and largest elevation value in the DEM dataset. If there are exceptionally small "nodata values" in the elevation dataset, the hillshade map might have a very small range of tones or colors -- it might look flat. If there are no "nodata values", a relatively flat ground surface might look like a mountain range on the hillshade map. The trick is to intercede and re-define the nodata values (if any) as being much closer to the ground-surface elevations, or to define one of the ground-surface elevations as a value that is much lower than the rest but not as low as the original "nodata value."

In the fragment of the DEM dataset shown above, which has no "nodata" values in the elevation matrix, the very first value in the elevation data (row 7, column 1) was changed to 0. By trial and error, you can adjust this bogus value to produce the desired range of tones or colors in the hillshade map.

Set various operating parameters for the code

The **zoneMeridian** is the longitude at the center of the UTM zone that contains the epicentral region. If the epicenter is near a zone boundary, base the UTM grid for this analysis in the zone that is to the west. The zone meridian for work in the Tahoe area is -123

In[5]:=
```
zoneMeridian = -123;
```

Input the geographic coordinates of the corners of the map area. Input the corners as a quadrangle.

In[6]:=
```
nwLat = 39.544; nwLong = -120.211;
swLat = 39.370; swLong = -120.211;
seLat = 39.370; seLong = -120.037;
neLat = 39.544;
neLong = -120.037;
```

The **epicenterWidth** is the half-width of the square that will be put over the top of the epicenter on the maps generated by this code, in meters

```
In[10]:= epicenterWidth = 200;
```

The **widthFactor** is a multiplier that allows for the adjustment of the half-width of the zone between parallel planes that generate the trace of the mean nodal plane on the maps generated by this code, in meters. A value of 1.5 for the **widthFactor** sets the distance between the planes at 1.5 times the cell size of the DEM.

```
In[11]:= widthFactor = 1.5;
```

The **topoBase** value forces one point of the DEM to have a particular value, which in turn allows the user to adjust the tonal range of the hillshade map.

```
In[12]:= topoBase = 0;
```

The constant **minusOne** is used as input for the pointEvaluator module, in the place for the null value.

```
In[13]:= minusOne = -1;
```

The **meanMultiplier** increases the width of the zone between parallel planes around the mean nodal plane, and is used to simplify interpretation of the seismo-lineament. Care must be taken so that the meanMultiplier value does not become so large that the ground-surface trace of the resulting planes extends beyond the boundaries of the seismo-lineament..

```
In[14]:= meanMultiplier = 1;
```

■ Input for the first earthquake in the set

The value that you input for **counter** determines which record (which row) of the earthquake dataset is evaluated.

```
In[15]:= counter = 7;
```

Remember to change the names of the output files

Evaluate the Notebook (Run the Program)

Select the Evaluation drop-down menu from the horizontal menu bar at the top of the window/screen, and choose Evaluate Notebook.

Some user-defined functions

```
In[16]:= makeVector[plunge_, trend_] := {Cos[plunge Degree] Sin[trend Degree],
           Cos[plunge Degree] Cos[trend Degree], -Sin[plunge Degree]};

In[17]:= vectorNorm[x_] := √(x.x);

In[18]:= unitVector[x_] :=
           {x[[1]] / vectorNorm[x], x[[2]] / vectorNorm[x], x[[3]] / vectorNorm[x]};
```

```
In[19]:= vectorAngle[a_, b_] := ArcCos[a.b / (vectorNorm[a] vectorNorm[b])];
```

The lat/long to UTM conversion is after Snyder (1982), and assumes use of the North American Datum of 1927 (NAD27). Another excellent resource for converting from lat/long to UTM and back, using a variety of datums (*e.g.*, NAD27, WGS84 and so on) is provided by Steven Dutch at http://www.uwgb.edu/dutchs/UsefulData/UTM-Formulas.htm, and by an online calculator at http://www.rcn.montana.edu/resources/tools/coordinates.aspx

```
In[20]:= convertToUTM[inLat_, inLong_, centMerid_] := Module[
           {c1, c2, c3, c4, c5, v1, v2, v3, v4, v5, v6, utmX, utmY}, c1 = 6 378 206.4;
           c2 = 0.00676866;
           c3 = 0;
           c4 = centMerid;
           c5 = 0.9996;
           v1 = c2 / (1 - c2);
           v2 = c1 / Sqrt[1 - (c2 * (Sin[inLat Degree]^2))];
           v3 = Tan[inLat Degree]^2;
           v4 = v1 * (Cos[inLat Degree]^2);
           v5 = (Cos[inLat Degree]) * ((inLong - c4) (π/180));
           v6 = (111 132.0894 * inLat) - (16 216.94 * Sin[2 * (inLat Degree)]) +
             (17.21 * Sin[4 * (inLat Degree)]) - (0.02 * Sin[6 * (inLat Degree)]);
           utmX = (c5 * v2 * (v5 + (((1 - v3 + v4) * v5^3) / 6) +
                 ((5 - (18 * v3) + (v3^2) + (72 * v4) - (58 * v1)) * v5^5) / 120)) + 500 000;
           utmY = c5 * (v6 - 0 + (v2 * Tan[inLat Degree] *
                 ((v5^2 / 2) + (((5 - v3 + (9 * v4) + (4 * (v4^2))) * v5^4) / 24) +
                   (((61 - (58 * v3) + (v3^2) + (600 * v4) - (330 * v1)) * v5^6) / 720))));
           {utmX, utmY}];
```

The following module differentiates between points that are within "width" meters from the fault plane and those that are further away. Given unit vector N that is normal to the fault plane that passes through the origin of the coordinate system, the distance from an arbitrary point (whose position vector is P) to that plane is given by |N·P|.

```
In[21]:= pointEvaluator[xCoord_, yCoord_, zCoord_, width_, fltUNrml_, nulData_] :=
          Module[{locVect, distToFlt, result}, locVect = {xCoord, yCoord, zCoord};
           distToFlt = Abs[Dot[fltUNrml, locVect]];
           result = If[(distToFlt ≤ width), 10., nulData];
           result];
```

The module **epicenterEval** defines the location of the epicenter on the hillshade map.

```
In[22]:= epicenterEval[xCoord_, yCoord_, zCoord_, width_, nulData_] := Module[
           {locVect, distToEpicenter, result}, result = If[(Abs[xCoord] < width),
             If[(Abs[yCoord] < width), zCoord, nulData], nulData];
           result];
```

The module **findHalfWidth** is used to determine the half-width of the uncertainty envelope given the lengths of the semi-axes of the triaxial uncertainty ellipsoid around the hypocenter, the azimuth of the greatest horizontal semi-axis, and the dip azimuth and dip angle of a nodal plane. The uncertainty envelope is bounded by parallel planes that are tangent to the uncertainty ellipsoid.

Input Parameters

- **dAz** is the trend or azimuth of the dip vector of the nodal plane, measured in degrees clockwise from north. All input azimuths are measured relative to the geographic coordinate system. Hence, an azimuth of 90° corresponds to due east.
- **dAng** is the plunge of the dip vector of the nodal plane, measured in degrees down from horizontal. Hence, a dip angle of 90° corresponds to a vertical vector pointing down.
- **hEr1** is the length of the **a** semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter, measured in kilometers. Hence, a value of 1 for **hEr1** means that the surface of the 95% CI uncertainty ellipse

is 1 km from the mean hypocenter measured horizontally in the direction **hEr1Az**.

• **hEr1Az** is the trend of the e1h uncertainty axis (i.e., of the **a** semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter), measured in degrees clockwise from north. All input azimuths are measured relative to the geographic coordinate system. Hence, an azimuth of 90° corresponds to due east.

• **hEr2** is the length of the **b** semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter, measured in kilometers. Hence, a value of 1 for **hEr2** means that the surface of the 95% CI uncertainty ellipse is 1 km from the mean hypocenter measured horizontally 90° from the direction **hEr1Az**.

• **vEz** is the length of the **c** semi-axis of the 95% CI uncertainty ellipsoid surrounding the mean hypocenter, measured in kilometers. Hence, a value of 1 for **vEz** means that the surface of the 95% CI uncertainty ellipse is 1 km from the mean hypocenter measured vertically.

Local Variables

• **var1** is the angle, in degrees, through which the geographic coordinate system (in which north is {0,1,0} and east is {1,0,0}) is transformed by rotation around the common **z** axis to the coordinate systen that is fixed to the axes of the triaxial uncertainty ellipsoid around the hypocenter (in which the **x** axis coincides with the **a** ellipsoidal axis and with **hEr1**, the **y** axis coincides with the **b** ellipsoidal axis and with **hEr2**, and the z axis is vertical and coincides with the **c** ellipsoidal axis and with **vEr**).

• **var2** is the unit vector parallel to the dip vector, in the geographic coordinate system in which north is {0,1,0} and east is {1,0,0}.

• **var3** is the unit vector parallel to the right - hand - rule strike, in the geographic coordinate system in which north is {0,1,0} and east is {1,0,0}.

• **var4** is the 3x3 rotation matrix in which a positive angle results in an anticlockwise rotation around the positive z axis.

• **var5** is the unit vector that is normal to the nodal plane in the geographic coordinate system in which north is {0,1,0} and east is {1,0,0}.

• **var6** is the unit vector that is normal to the nodal plane in the ellipsoid-centered coordinate system that is rotated **var1** degrees around the vertical (z) axis.

• **var7-9** are intermediate values that are ultimately used to compute the value of the Lagrangian multiplier lambda.

• **var10** is the Lagrangian multiplier lambda

• **var11** is the set of components of the location vector to one of the tangent points, expressed in the ellipsoid-centered coordinate system that is rotated **var1** degrees around the vertical (z) axis.

• **var12** is the distance, in km, between the a nodal plane through the hypocenter and a parallel nodal plane through one of the tangent points to the triaxial uncertainty ellipsoid around the hypocenter.

Output

• **result** is the distance, in km, between the a nodal plane through the hypocenter and a parallel nodal plane through one of the tangent points to the triaxial uncertainty ellipsoid around the hypocenter..

```
In[23]:= findHalfWidth[dipAz_, dipAng_, hEr1_, hEr1Az_, hEr2_, vEr_] :=
           Module[{var1, var2, var3, var4, var5, var6, var7, var8,
             var9, var10, var11, var12, result}, var1 = hEr1Az - 90;
            var2 = {Cos[dipAng Degree] Sin[dipAz Degree],
              Cos[dipAng Degree] Cos[dipAz Degree], -Sin[dipAng Degree]};
            var3 = {Sin[(dipAz - 90) Degree], Cos[(dipAz - 90) Degree], 0};
            var4 = {{Cos[var1 Degree], -Sin[var1 Degree], 0},
              {Sin[var1 Degree], Cos[var1 Degree], 0}, {0, 0, 1}};
            var5 = Cross[var2, var3];
            var6 = var4.var5;
            var7 = (1 / hEr1^2) (var6[[1]] (hEr1^2 / 2))^2;
            var8 = (1 / hEr2^2) (var6[[2]] (hEr2^2 / 2))^2;
            var9 = (1 / vEr^2) (var6[[3]] (vEr^2 / 2))^2;
            var10 = Sqrt[(1 / (var7 + var8 + var9))];
            var11 = {var10 * var6[[1]] * (hEr1^2 / 2),
              var10 * var6[[2]] * (hEr2^2 / 2), var10 * var6[[3]] * (vEr^2 / 2)};
            var12 = var6.var11;
            result = var12];
```

Computations specific to an individual hypocenter, for which there might be multiple focal mechanism solutions available

■ Read and interpret the DEM header information

```
In[24]:= headerData = Table[mydata[[i, j]], {i, 6}, {j, 2}];

In[25]:= ncols = headerData[[1, 2]];

In[26]:= nrows = headerData[[2, 2]];

In[27]:= xllcorner = headerData[[3, 2]];

In[28]:= yllcorner = headerData[[4, 2]];

In[29]:= cellsize = headerData[[5, 2]];

In[30]:= nodataValue = headerData[[6, 2]];
```

■ Read and interpret the hypocenter location data

```
In[31]:= record = counter;

In[32]:= focalLat = eqData[[record, 7]];

In[33]:= focalLong = eqData[[record, 8]];

In[34]:= focalDepthKm = eqData[[record, 9]];
```

The UTM coordinates that specify the horizontal position of the earthquake focus (i.e., the epicenter location) are given in meters. The variable focalDepth gives the depth of the earthquake focus, in meters.

```
In[35]:= focalDepth = focalDepthKm * (-1000);
```

The reported vertical and horizontal uncertainties in the location of the earthquake focus are given in kilometers. The variables horizError and vertError convert these to meters.

```
In[36]:= eh1 = eqData[[record, 10]] * 1000;

In[37]:= eh2 = eqData[[record, 11]] * 1000;

In[38]:= eh1Azimuth = eqData[[record, 12]];

In[39]:= eh1Az = If[((eh1Azimuth + gridNorthAdjustment) < 0),
             (360 + eh1Azimuth + gridNorthAdjustment),
             If[((eh1Azimuth + gridNorthAdjustment) > 360),
              ((eh1Azimuth + gridNorthAdjustment) - 360),
              (eh1Azimuth + gridNorthAdjustment)]];

In[40]:= ez = eqData[[record, 13]] * 1000;
```

Convert the input coordinates of the earthquake focus to the same coordinate system as the DEM data

```
In[41]:= utmCoordinates = convertToUTM[focalLat, focalLong, zoneMeridian];
```

The "focus" is the (x, y, z) coordinates in meters of the reported earthquake focus, in a coordinate system in which the origin is the lower-left (southwest) corner of the DEM at sea level.

```
In[42]:= focusMean = {(utmCoordinates[[1]] - xllcorner),
             (utmCoordinates[[2]] - yllcorner), focalDepth};

In[43]:= demImageFileM =
           Table[mydata[[i + 6, j]] - focusMean[[3]], {i, nrows}, {j, ncols}];

In[44]:= rawElevDataM =
           Table[mydata[[i + 6, j]] - focusMean[[3]], {i, nrows}, {j, ncols}];

In[45]:= demImageFileM = Table[rawElevDataM[[i, j]], {i, nrows}, {j, ncols}];
```

Make the hillshade image

■ Adjustment for input geographic coordinates based on true north to the relevant UTM grid north

```
In[46]:= midLat = ((nwLat - swLat) / 2) + swLat;

In[47]:= midLong = ((seLong - swLong) / 2) + swLong;
```

The **gridNorthAdjustment** is the angle between true north and grid north at/near the center of the map area. If grid north is found by a clockwise rotation from true north, the sign of the **gridNorthAdjustment** is negative; otherwise, it is positive. The **gridNorthAdjustment** is approximately equal to (zoneMeridian – (longitude at center of map area))*Sin[latitude at center of map area].

```
In[48]:= gridNorthAdjustment = (zoneMeridian - (midLong)) * Sin[midLat Degree]

Out[48]= -1.82769
```

Select and label the matrix of elevation data

```
In[49]:= rawElevDataInput = Table[mydata[[i + 6, j]] , {i, nrows}, {j, ncols}];

In[50]:= elev3 = Table[demImageFileM[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

Work with the orientation data for a given fault - plane solution

```
In[51]:= npDipTrend = eqData[[record, 15]];

In[52]:= npDipTr = If[((npDipTrend + gridNorthAdjustment) < 0),
             (360 + npDipTrend + gridNorthAdjustment),
             If[((npDipTrend + gridNorthAdjustment) > 360),
              ((npDipTrend + gridNorthAdjustment) - 360),
              (npDipTrend + gridNorthAdjustment)]];

In[53]:= npDipPlunge = eqData[[record, 16]];

In[54]:= npDipTrendUncert = eqData[[record, 18]];

In[55]:= npDipAngUncert = eqData[[record, 19]];

In[56]:= npDipTrRad = npDipTr * (π / 180);

In[57]:= lightDirection =
            If[npDipTrRad > (3 π / 2), (2 π) - npDipTrRad, (π / 2) - npDipTrRad];

In[58]:= hillshadeMap1 = ReliefPlot[elev3, ColorFunction -> "GrayTones",
             LightingAngle -> {lightDirection, Pi / 12}];

In[59]:= ClearAll[elev3];
```

Computations specific to the selected nodal plane

Swath 1 of 7: The middle of the road

Convert the fault dip vector from trend and plunge to a unit location vector

```
In[60]:= npDipVect = unitVector[makeVector[npDipPlunge, npDipTr]];
```

Find the strike vector defined using the right-hand rule

```
In[61]:= npStrike = If[(npDipTr < 90), (npDipTr + 270), (npDipTr - 90)];

In[62]:= npStrikeVect = {Sin[npStrike Degree], Cos[npStrike Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

```
In[63]:= npNormalVect = unitVector[Cross[npDipVect, npStrikeVect]];

In[64]:= zoneHalfWidthMThin = widthFactor * cellsize * Sin[npDipPlunge Degree];

In[65]:= N[%]

Out[65]= 40.759
```

```
In[66]:= uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
             ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
             zoneHalfWidthMThin, npNormalVect, minusOne], {i, nrows}, {j, ncols}];

In[67]:= elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];

In[68]:= traceImgFileMThin = ListContourPlot[elevData, ContourShading -> False,
            AspectRatio -> Automatic, Contours -> {topoBase}];

In[69]:= elevDataTemp1 = elevData;

In[70]:= ClearAll[elevData, uncertSwath];
```

■ Swath 2 of 7: Gentle dip, smaller strike azimuth (GS)

Subtract the dip-angle uncertainty from the mean dip angle (i.e., from the plunge of the mean dip vector)

```
In[71]:= gentleDipAng = If[(npDipPlunge - npDipAngUncert) < 0,
            Abs[(npDipPlunge - npDipAngUncert)], (npDipPlunge - npDipAngUncert)];
```

Subtract the dip-trend uncertainty from the mean dip azimuth (i.e., from the mean dip trend)

```
In[72]:= gentleDipAzS = If[(npDipPlunge - npDipAngUncert) < 0, If[
             ((npDipTr - npDipTrendUncert) > 180), ((npDipTr - npDipTrendUncert) - 180),
             ((npDipTr - npDipTrendUncert) + 180)],
            If[((npDipTr - npDipTrendUncert) < 0), 360 + (npDipTr - npDipTrendUncert),
             (npDipTr - npDipTrendUncert)]];
```

Convert the fault dip vector from trend and plunge to a unit location vector

```
In[73]:= dipVectGS = unitVector[makeVector[gentleDipAng, gentleDipAzS]];
```

Find the strike vector defined using the right-hand rule

```
In[74]:= strikeGS = If[(gentleDipAzS < 90), (gentleDipAzS + 270), (gentleDipAzS - 90)];

In[75]:= strikeVectGS = {Sin[strikeGS Degree], Cos[strikeGS Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

```
In[76]:= normalVectGS = unitVector[Cross[dipVectGS, strikeVectGS]];
```

Find the width of the nodal-plane uncertainty zone with the given dip angle, dip azimuth, and triaxial hypocenter uncertainty ellipsoid

```
In[77]:= zoneHalfWidthGS =
           findHalfWidth[gentleDipAzS, gentleDipAng, eh1, eh1Az, eh2, ez];

In[78]:= N[zoneHalfWidthGS]

Out[78]= 48.6285
```

Find the boundaries of the uncertainty swath as it intersects the ground surface

```
In[79]:= uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
             ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
             zoneHalfWidthGS, normalVectGS, minusOne], {i, nrows}, {j, ncols}];

In[80]:= elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

```
In[81]:= elevDataTemp2 = elevData + elevDataTemp1;

In[82]:= ClearAll[elevData, elevDataTemp1, uncertSwath];
```

Swath 3 of 7: Gentle dip, larger strike azimuth (GL)

Add the dip-trend uncertainty from the mean dip azimuth (i.e., from the mean dip trend)

```
In[83]:= gentleDipAzL = If[(npDipPlunge - npDipAngUncert) < 0, If[
             ((npDipTr + npDipTrendUncert) > 180), ((npDipTr + npDipTrendUncert) - 180),
             ((npDipTr + npDipTrendUncert) + 180)],
            If[((npDipTr + npDipTrendUncert) > 360),
             (npDipTr + npDipTrendUncert) - 360, (npDipTr + npDipTrendUncert)]];
```

Convert the fault dip vector from trend and plunge to a unit location vector

```
In[84]:= dipVectGL = unitVector[makeVector[gentleDipAng, gentleDipAzL]];
```

Find the strike vector defined using the right-hand rule

```
In[85]:= strikeGL = If[(gentleDipAzL < 90), (gentleDipAzL + 270), (gentleDipAzL - 90)];

In[86]:= strikeVectGL = {Sin[strikeGL Degree], Cos[strikeGL Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

```
In[87]:= normalVectGL = unitVector[Cross[dipVectGL, strikeVectGL]];
```

Find the width of the nodal-plane uncertainty zone with the given dip angle, dip azimuth, and triaxial hypocenter uncertainty ellipsoid

```
In[88]:= zoneHalfWidthGL =
           findHalfWidth[gentleDipAzL, gentleDipAng, eh1, eh1Az, eh2, ez];

In[89]:= N[zoneHalfWidthGL]

Out[89]= 48.2926
```

Find the boundaries of the uncertainty swath as it intersects the ground surface

```
In[90]:= uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
             ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
             zoneHalfWidthGL, normalVectGL, minusOne], {i, nrows}, {j, ncols}];

In[91]:= elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];

In[92]:= elevDataTemp1 = elevData + elevDataTemp2;

In[93]:= ClearAll[elevData, elevDataTemp2, uncertSwath];

In[94]:= Dimensions[elevDataTemp1]

Out[94]= {1164, 938}
```

Swath 4 of 7: Steep dip, smaller strike azimuth (SS)

Add the dip-angle uncertainty to the mean dip angle (i.e., from the plunge of the mean dip vector)

```
In[95]:= steepDipAng = If[(npDipPlunge + npDipAngUncert) > 90,
             (180 - (npDipPlunge + npDipAngUncert)), (npDipPlunge + npDipAngUncert)];
```

Subtract the dip-trend uncertainty from the mean dip azimuth (i.e., from the mean dip trend)

```
In[96]:= steepDipAzS = If[(npDipPlunge + npDipAngUncert) > 90, If[
              ((npDipTr - npDipTrendUncert) > 180), ((npDipTr - npDipTrendUncert) - 180),
              ((npDipTr - npDipTrendUncert) + 180)],
             If[((npDipTr - npDipTrendUncert) < 0), 360 + (npDipTr - npDipTrendUncert),
              (npDipTr - npDipTrendUncert)]];
```

Convert the fault dip vector from trend and plunge to a unit location vector

```
In[97]:= dipVectSS = unitVector[makeVector[steepDipAng, steepDipAzS]];
```

Find the strike vector defined using the right-hand rule

```
In[98]:= strikeSS = If[(steepDipAzS < 90), (steepDipAzS + 270), (steepDipAzS - 90)];
```

```
In[99]:= strikeVectSS = {Sin[strikeSS Degree], Cos[strikeSS Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

```
In[100]:=
        normalVectSS = unitVector[Cross[dipVectSS, strikeVectSS]];
```

Find the width of the nodal-plane uncertainty zone with the given dip angle, dip azimuth, and triaxial hypocenter uncertainty ellipsoid

```
In[101]:=
        zoneHalfWidthSS =
          findHalfWidth[steepDipAzS, steepDipAng, eh1, eh1Az, eh2, ez];
```

```
In[102]:=
        N[zoneHalfWidthSS]

Out[102]=
        32.268
```

Find the boundaries of the uncertainty swath as it intersects the ground surface

```
In[103]:=
        uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
             ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
             zoneHalfWidthSS, normalVectSS, minusOne], {i, nrows}, {j, ncols}];
```

```
In[104]:=
        elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

```
In[105]:=
        elevDataTemp2 = elevData + elevDataTemp1;
```

```
In[106]:=
        ClearAll[elevData, elevDataTemp1, uncertSwath];
```

Swath 5 of 7: Steep dip, larger strike azimuth (SL)

Add the dip-trend uncertainty from the mean dip azimuth (i.e., from the mean dip trend)

In[107]:=
```
steepDipAzL = If[(npDipPlunge + npDipAngUncert) > 90,
   If[((npDipTr + npDipTrendUncert) > 180),
    ((npDipTr + npDipTrendUncert) - 180), ((npDipTr + npDipTrendUncert) + 180)],
   If[((npDipTr + npDipTrendUncert) > 360),
    (npDipTr + npDipTrendUncert) - 360, (npDipTr + npDipTrendUncert)]];
```

Convert the fault dip vector from trend and plunge to a unit location vector

In[108]:=
```
dipVectSL = unitVector[makeVector[steepDipAng, steepDipAzL]];
```

Find the strike vector defined using the right-hand rule

In[109]:=
```
strikeSL = If[(steepDipAzL < 90), (steepDipAzL + 270), (steepDipAzL - 90)];
```

In[110]:=
```
strikeVectSL = {Sin[strikeSL Degree], Cos[strikeSL Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

In[111]:=
```
normalVectSL = unitVector[Cross[dipVectSL, strikeVectSL]];
```

Find the width of the nodal-plane uncertainty zone with the given dip angle, dip azimuth, and triaxial hypocenter uncertainty ellipsoid

In[112]:=
```
zoneHalfWidthSL =
  findHalfWidth[steepDipAzL, steepDipAng, eh1, eh1Az, eh2, ez];
```

In[113]:=
```
N[zoneHalfWidthSL]
```

Out[113]=
```
30.9832
```

Find the boundaries of the uncertainty swath as it intersects the ground surface

In[114]:=
```
uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
    ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
    zoneHalfWidthSL, normalVectSL, minusOne], {i, nrows}, {j, ncols}];
```

In[115]:=
```
elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

In[116]:=
```
elevDataTemp1 = elevData + elevDataTemp2;
```

In[117]:=
```
ClearAll[elevData, elevDataTemp2, uncertSwath];
```

Swath 6 of 7: Steep dip, mean strike azimuth (SM)

Add the dip-trend uncertainty from the mean dip azimuth (i.e., from the mean dip trend)

In[118]:=
```
steepDipAzM = If[(npDipPlunge + npDipAngUncert) > 90,
   If[(npDipTr > 180), (npDipTr - 180), (npDipTr + 180)], npDipTr];
```

Convert the fault dip vector from trend and plunge to a unit location vector

In[119]:=
```
dipVectSM = unitVector[makeVector[steepDipAng, steepDipAzM]];
```

Find the strike vector defined using the right-hand rule

In[120]:=
```
strikeSM = If[(steepDipAzM < 90), (steepDipAzM + 270), (steepDipAzM - 90)];
```

In[121]:=
```
strikeVectSM = {Sin[strikeSM Degree], Cos[strikeSM Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

In[122]:=
```
normalVectSM = unitVector[Cross[dipVectSM, strikeVectSM]];
```

Find the width of the nodal-plane uncertainty zone with the given dip angle, dip azimuth, and triaxial hypocenter uncertainty ellipsoid

In[123]:=
```
zoneHalfWidthSM =
  findHalfWidth[steepDipAzM, steepDipAng, eh1, eh1Az, eh2, ez];
```

In[124]:=
```
N[zoneHalfWidthSM]
```

Out[124]=
```
31.5164
```

Find the boundaries of the uncertainty swath as it intersects the ground surface

In[125]:=
```
uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
    ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
    zoneHalfWidthSM, normalVectSM, minusOne], {i, nrows}, {j, ncols}];
```

In[126]:=
```
elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

In[127]:=
```
elevDataTemp2 = elevData + elevDataTemp1;
```

In[128]:=
```
ClearAll[elevData, elevDataTemp1, uncertSwath];
```

Swath 7 of 7: The middle of the road

Convert the fault dip vector from trend and plunge to a unit location vector

In[129]:=
```
npDipVect = unitVector[makeVector[npDipPlunge, npDipTr]];
```

Find the strike vector defined using the right-hand rule

In[130]:=
```
npStrike = If[(npDipTr < 90), (npDipTr + 270), (npDipTr - 90)];
```

In[131]:=
```
npStrikeVect = {Sin[npStrike Degree], Cos[npStrike Degree], 0};
```

Find the unit vector that is normal to the fault plane and is directed upwards

In[132]:=
```
npNormalVect = unitVector[Cross[npDipVect, npStrikeVect]];
```

Find the width of the nodal-plane uncertainty zone with the given dip angle, dip azimuth, and triaxial hypocenter uncertainty ellipsoid

In[133]:=
```
zoneHalfWidthMean =
  findHalfWidth[npDipTr, npDipPlunge, eh1, eh1Az, eh2, ez];
```

In[134]:=
```
N[zoneHalfWidthMean]
```

Out[134]=
```
26.206
```

In[135]:=
```
uncertSwath = Table[pointEvaluator[((cellsize * (j - 1)) - focusMean[[1]]),
    ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
    zoneHalfWidthMean, npNormalVect, minusOne], {i, nrows}, {j, ncols}];
```

In[136]:=
```
elevData = Table[uncertSwath[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

In[137]:=
```
elevDataTemp1 = elevData + elevDataTemp2;
```

In[138]:=
```
summary = Table[
   If[elevDataTemp1[[i, j]] < 0, minusOne, 1], {i, nrows}, {j, ncols}];
```

In[139]:=
```
traceImgFileSum = ListContourPlot[summary, ContourShading → False,
   AspectRatio → Automatic, Contours → {topoBase}];
```

In[140]:=
```
ClearAll[elevData, elevDataTemp1, elevDataTemp2, uncertSwath];
```

■ Epicenter

In[141]:=
```
epicenterDot = Table[epicenterEval[((cellsize * (j - 1)) - focusMean[[1]]),
    ((cellsize * (nrows - i)) - focusMean[[2]]), demImageFileM[[i, j]],
    epicenterWidth, minusOne], {i, nrows}, {j, ncols}];
```

In[142]:=
```
elevData = Table[epicenterDot[[i, j]], {i, nrows, 1, -1}, {j, ncols}];
```

In[143]:=
```
traceImgFileE = ListContourPlot[elevData, ContourShading -> False,
   AspectRatio -> Automatic, Contours -> {topoBase}];
```

In[144]:=
```
ClearAll[elevData, epicenterDot];
```

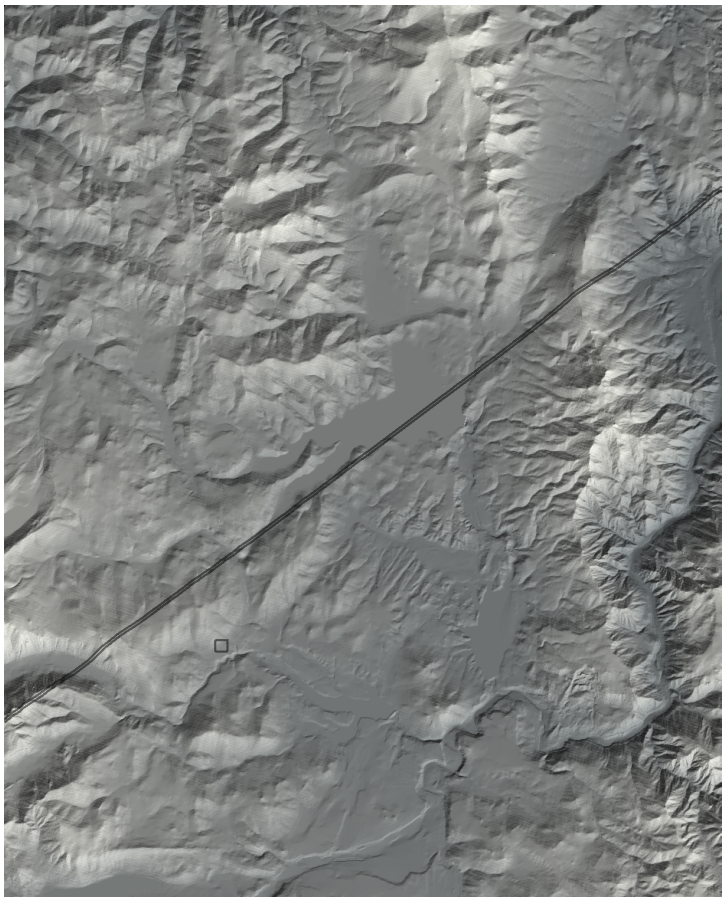
In[145]:=
```
ClearAll[npDipTr, npDipTrend, npDipPlunge, npDipVect, npStrike, npStrikeVect,
  npNormalVect, zoneHalfWidthMean, gentleDipAng, gentleDipAzS,
  dipVectGS, strikeGS, strikeVectGS, normalVectGS, zoneHalfWidthGS,
  gentleDipAzL, dipVectGL, strikeGL, strikeVectGL, normalVectGL,
  zoneHalfWidthGL, steepDipAng, steepDipAzS, dipVectSS, strikeSS,
  strikeVectSS, normalVectSS, zoneHalfWidthSS, steepDipAzL, dipVectSL,
  strikeSL, strikeVectSL, normalVectSL, zoneHalfWidthSL, steepDipAzM,
  dipVectSM, strikeSM, strikeVectSM, normalVectSM, zoneHalfWidthSM];
```

Output

In[146]:=
```
outFile1 = Show[hillshadeMap1, traceImgFileMThin, traceImgFileE];
```

In[147]:=
```
Show[%]
```

Out[147]=



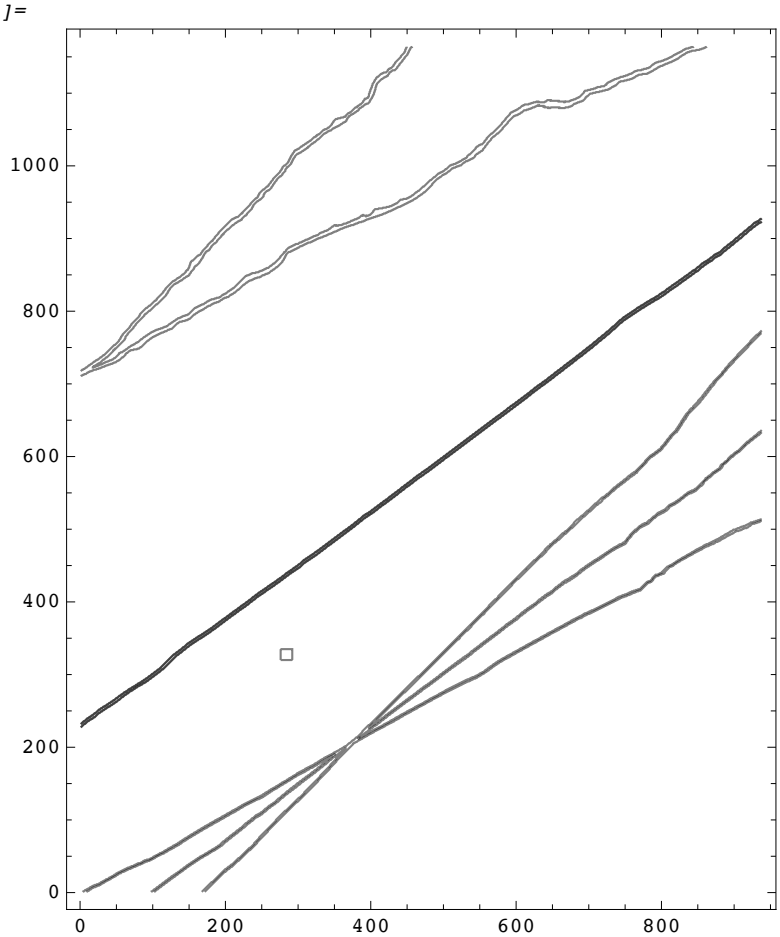
In[148]:=

```
outFile2 = Show[traceImgFileMThin, traceImgFileSum, traceImgFileE];
```

In[149]:=

```
Show[%]
```

Out[149]=

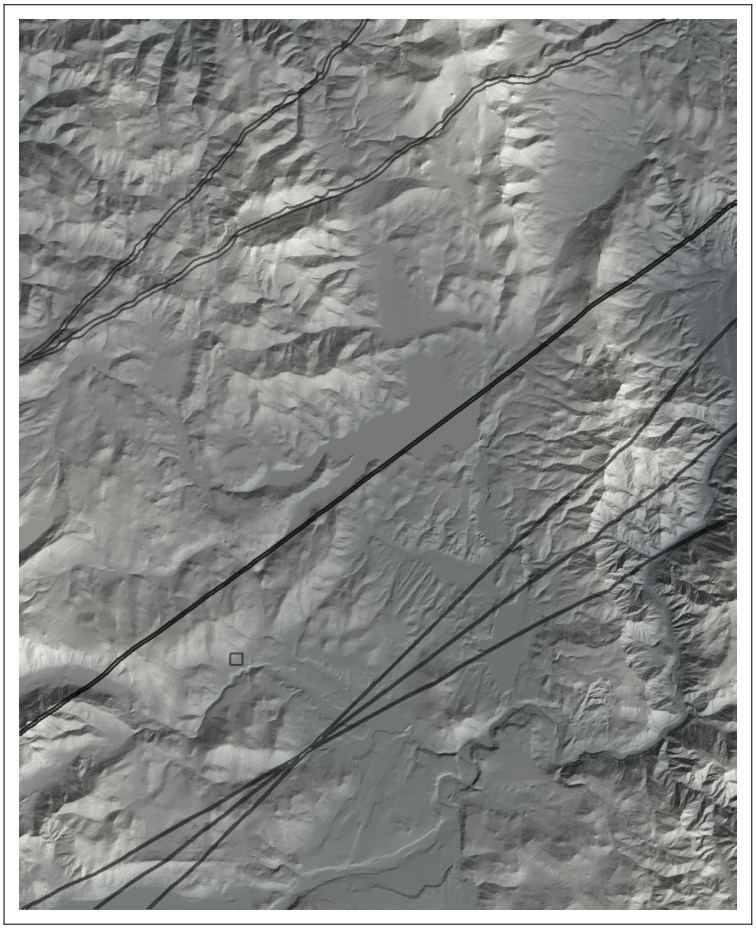


In[150]:=

```
outFile3 = Show[hillshadeMap1, outFile2];
```

In[151]:=
```
Show[%]
```

Out[151]=



The black square in each of the maps above marks the epicenter of the earthquake. The length of each side is approximately "**epicenterWidth**" in meters.

In[152]:=
```
ClearAll[traceImgFileMean, traceImgFileGS, traceImgFileGL,
  traceImgFileSS, traceImgFileSL, traceImgFileSM, traceImgFileMThin];
```

■ Export data and image files

The file created in the next line is JPEG image that contains the graphic showing the hillshade image developed from the DEM, curves marking the boundaries of the uncertainty regions, and a circle with 1 km diameter centered on the epicenter.

IMPORTANT NOTE: The user must supply a name for the output file that is different from existing file names, or else the existing files may be over-written.

In[153]:=
```
Export["/Users/vincecronin/Desktop/Event-S11-map1.jpg", outFile1];
```

In[154]:=
```
Export["/Users/vincecronin/Desktop/Event-S11-map2.jpg", outFile2];
```

In[155]:=
```
Export["/Users/vincecronin/Desktop/Event-S11-map3.jpg", outFile3];
```

In[156]:=
```
ClearAll[outFile1, outFile2, outFile3];
```

Closing

■ How long did this program take to run, in minutes?

In[157]:=
```
minutesForProcessing = (AbsoluteTime[] - startTime) / 60;
```

In[158]:=
```
N[%]
```

Out[158]=
```
3.19667
```

References

Colley, S.J., 2002, Vector calculus [2nd edition]: Upper Saddle River, New Jersey, Prentice-Hall, 558 p., (pp. 43-50), ISBN 0-13-041531-6.

Cronin, V.S., 2014, Seismo-Lineament Analysis Method (SLAM), using earthquake focal mechanisms to help recognize seismogenic faults, *in* Grutzner, C., Choi, J - H., Edwards, P., and Kim, Y - .S., [editors], Proceeding of the 5th International INQUA Meeting on Paleoseismology, Active Tectonics and Archeoseismology, 21-27 September 2014, p.28 - 31, ISBN 9791195344109 93450; available via http://CroninProjects.org/Vince/S-LAM/CroninINQUA_PATA14.pdf

Cronin, V.S., and Sverdrup, K.A., 1998, Preliminary assessment of the seismicity of the Malibu Coast Fault Zone, southern California, and related issues of philosophy and practice, *in* Welby, C.W., and Gowan, M.E. [editors], A Paradox of Power--Voices of Warning and Reason in the Geosciences: Geological Society of America, Reviews in Engineering Geology, p. 123-155.

Cronin, V.S., Byars, B.W., and Gammill, T., 2003, Developing techniques for regional structural interpretation using GIS and DEM-based terrain analysis: Geological Society of America, Abstracts with Programs, v. 34, no. 7, p. 261. http://gsa.confex.com/gsa/2003AM/finalprogram/abstract_61851.htm

Cronin, V.S., Millard, M.A., Seidman, L.E., and Bayliss, B.G., 2008, The Seismo-Lineament Analysis Method [SLAM] -- A reconnaissance tool to help find seismogenic faults: Environmental and Engineering Geoscience, v. 14, no. 3, p. 199-219.

Davis, H.F., and Snider, A.D., 1987, Introduction to vector analysis [5th edition]: Boston, Allyn and Bacon, 365 p. (pp. 34-37), ISBN 0-205-10263-8.

Gammill, T., Cronin, V.S., and Byars, B.W., 2004, Combining earthquake focal data and digital map analysis in reconnaissance for active faults, central Santa Monica Mountains and northern Santa Monica Bay, California:

Geological Society of America, Abstracts with Programs,v. 36, no. 5, http://gsa.confex.com/gsa/2004AM/finalprogram/abstract_80352.htm

Haneberg, W.C., 2004, Computational geosciences with *Mathematica*: Berlin, Springer-Verlag, 381 p., CD, ISBN 3-540-40245-4.

Snyder, J.P., 1982, Map projections used by the U.S. Geological Survey: U.S. Geological Survey Bulletin 1532, pp. 63-69 and 233-235.

Other references related to this work are available online at http://croninprojects.org/DVFZ/index.htm and http://croninprojects.org/Vince/SLAM/index.htm

---