Modeling the position of marine magnetic anomaly isochrons

Vince Cronin, version of 26 April 2012

Algorithm for finding an isochron on opposite sides of a mid - ocean ridge

Assumptions

1. Earth is a unit sphere

2. New crust is added symmetrically to the trailing edge of both plates

3. Strain is partitioned equally to both plates at the center of a ridge-ridge transform fault

4. The angular velocity of both plates relative to the external reference frame is constant over the modeled time interval, and so the angular velocity of one plate relative to the other is also constant. Hence, the angular distance betweeen the three corresponding poles is also constant.

Input data

We will need the following data to model the isochrons.

1. The angular velocity of each plate relative to an external reference frame.

2. The location of two representative points along a single ridge axis.

3. The approximate location of the midpoint of each of the transform faults that mark the two ends of the ridge axis.

Reference frames

The motion of the ridge-transform-ridge system will be tracked in a reference frame that is fixed to the external reference frame, in which the three rotational poles are fixed. The final isochron location on a given plate is found by rotation around the plate's pole of rotation by the angle ωt, where t is the isochron age and ω is the angular speed of the plate in the external reference frame.

■ Actions/computations

Over a small time interval (perhaps 0.01 Myr), move each of the 4 reference points around the pole to plate A from its initial position to mark temporary point A, then around the pole to plate B from its initial position to mark temporary point B. Then find the location vector to the midpoint between point A and point B for each of the 4 reference points. The result is the new location of the 4 points at the end of the small time interval. This becomes the starting location for the next small time interval. Repeat this until the elapsed model time is the same as the isochron age that we want to model. At the isochron age, output the location vector of each of the 4 reference points to a file with the isochron age. The ends of the ridge axis will be found along the great circle defined by the two points along the ridge axis, with an angular distance from the pole of relative motion that is equal to that of the two transform-fault midpoints.

User - defined functions

```
convert2Cart[lat_, long_] := {Cos[lat Degree] Cos[long Degree],
   Cos[lat Degree] Sin[long Degree], Sin[lat Degree]};

unitVect3D[vect_] := {(vect[[1]] / Norm[vect]),
   (vect[[2]] / Norm[vect]), (vect[[3]] / Norm[vect])};

findGeogCoord[vect_] := Module[{lat, long, a, b, c, d, e, f},
   a = ArcSin[vect[[3]]]; b = {vect[[1]], vect[[2]], 0}; c = If[
     ((Abs[vect[[1]]] < (1 × 10^-14)) && (Abs[vect[[2]]] < (1 × 10^-14))),
     {1, 1, 0}, {vect[[1]] / Norm[b], vect[[2]] / Norm[b], 0}];
   d = {1, 0, 0}; e = VectorAngle[c, d];
   f = If[(vect[[2]] < 0), (-e), (e)]; lat = a (180 / π); long = If[
     ((Abs[vect[[1]]] < (1 × 10^-14)) && (Abs[vect[[2]]] < (1 × 10^-14))),
     0, (f (180 / π))]; {lat, long}];

makeGreatCircle[normal_] :=
 Module[{a, x1, y1, z1, x2, y2, z2, j1, j2, j3, θ, b},
  a = Table[{Cos[i Degree], Sin[i Degree], 0}, {i, 0, 360, 5}];
  x1 = {1, 0, 0}; y1 = {0, 1, 0}; z1 = {0, 0, 1}; z2 = normal;
  x2 = unitVect3D[Cross[z2, z1]]; y2 = unitVect3D[Cross[z2, x2]];
  j1 = {{x1.x2, y1.x2, z1.x2}, {x1.y2, y1.y2, z1.y2},
    {x1.z2, y1.z2, z1.z2}}; θ = VectorAngle[z1, z2];
  j2 = {{1, 0, 0}, {0, Cos[θ], Sin[θ]}, {0, -Sin[θ], Cos[θ]}};
  j3 = Inverse[j1]; b = Table[j3.j2.j1.a[[i]], {i, 1, Length[a]}]; b]
```

```
circMotion[x_, angVelVect_, dT_] :=
  Module[{north, rotPole, w, xPole, yPole, m1, m2, m3, answer},
   north = {0, 0, 1}; rotPole = unitVect3D[angVelVect];
   w = Norm[angVelVect]; xPole = unitVect3D[rotPole × north];
   yPole = unitVect3D[rotPole × xPole]; m1 =
    {{xPole[[1]], xPole[[2]], xPole[[3]]}, {yPole[[1]], yPole[[2]],
      yPole[[3]]}, {rotPole[[1]], rotPole[[2]], rotPole[[3]]}};
   m2 = {{Cos[(w dT) Degree], -Sin[(w dT) Degree], 0},
     {Sin[(w dT) Degree], Cos[(w dT) Degree], 0}, {0, 0, 1}};
   m3 = Inverse[m1]; answer = m3.m2.m1.x; answer];

zRotByAngle[x_, rotPole_, angleRad_] :=
  Module[{north, xPole, yPole, m1, m2, m3, answer},
   north = {0, 0, 1}; xPole = unitVect3D[rotPole × north];
   yPole = unitVect3D[rotPole × xPole]; m1 =
    {{xPole[[1]], xPole[[2]], xPole[[3]]}, {yPole[[1]], yPole[[2]],
      yPole[[3]]}, {rotPole[[1]], rotPole[[2]], rotPole[[3]]}};
   m2 = {{Cos[(angleRad) Degree], -Sin[(angleRad) Degree], 0},
     {Sin[(angleRad) Degree], Cos[(angleRad) Degree], 0},
     {0, 0, 1}}; m3 = Inverse[m1]; answer = m3.m2.m1.x; answer];

midPt[v_, w1_, w2_, t_] := Module[{a, b, midpoint},
   a = circMotion[v, w1, t]; b = circMotion[v, w2, t];
   midpoint = unitVect3D[(a + b) / 2]; midpoint];

nextIsochron[m_, w1_, w2_, youngerChronT_, olderChronT_,
   intervalT_] := Module[{newT, oldT, nextCart}, nextCart = m;
   oldT = 0; endT = Abs[olderChronT] - Abs[youngerChronT];
   Catch[Do[newT = oldT + Abs[intervalT];
     If[newT > endT, Throw[nextCart]];
     nextCart = Table[midPt[nextCart[[i]], w1, w2, intervalT],
       {i, 1, Length[nextCart]}]; oldT = newT, {1000}]]];
```

```
findRidgeEnds[m_, relVelVect_] :=
  Module[{angle1, angle2, a1, a2, a3, a4, a5, a6, a7, b1, b2,
    b3, b4, b5, c1, c2, c3, c4, c5, c6, c7, d1, d2, d3, d4, d5},
   angle1 = VectorAngle[m[[1]], relVelVect];
   angle2 = VectorAngle[m[[4]], relVelVect];
   a1 = VectorAngle[m[[2]], relVelVect];
   a2 = unitVect3D[m[[3]] × m[[2]]];
   a3 = zRotByAngle[m[[2]], a2, 0.001];
   a4 = VectorAngle[a3, relVelVect]; a5 = Abs[a4 - angle1];
   a6 = Catch[a6 = a3; a7 = a5; Do[b1 = zRotByAngle[a6, a2, i];
      b2 = VectorAngle[b1, relVelVect]; b3 = Abs[b2 - angle1];
      If[b3 > a7, (Return[b5]; Break[])]; a7 = b3;
      a6 = b1; b5 = b4; b4 = b1, {i, 0, 0.6, 0.000001}]];
   c1 = VectorAngle[m[[3]], relVelVect];
   c2 = unitVect3D[m[[2]] × m[[3]]];
   c3 = zRotByAngle[m[[3]], c2, 0.001];
   c4 = VectorAngle[c3, relVelVect]; c5 = Abs[c4 - angle2];
   c6 = Catch[c6 = c3; c7 = c5; Do[d1 = zRotByAngle[c6, c2, j];
      d2 = VectorAngle[d1, relVelVect]; d3 = Abs[d2 - angle2];
      If[d3 > c7, (Return[d5]; Break[])]; c7 = d3; c6 = d1;
      d5 = d4; d4 = d1, {j, 0, 0.6, 0.000001}]]; {a6, c6}];

buildIsochrons[startLoc_, w1_, w2_, t_, intT_] :=
  Module[{a, b, c, builtFile}, a = startLoc; builtFile = {a};
   Do[b = nextIsochron[a, w1, w2, t[[i]], t[[i + 1]], intT];
    c = Join[builtFile, {b}]; builtFile = c;
    a = b, {i, 1, Length[t] - 1}]; builtFile];

makeIsochronFile[m_, w1_, w2_, mts_] :=
  Module[{a, b, c, answer}, a = {m[[1]]};
   b = Table[circMotion[m[[i, j]], w1, mts[[i]]], {i, 2, Length[m]},
     {j, 1, 2}]; c = Table[circMotion[m[[i, j]], w2, mts[[i]]],
     {i, 2, Length[m]}, {j, 1, 2}]; answer = Join[a, b, c]; answer];
```

Input data

The input data are in a list in which the first element is the latitude (north is +ve, south is -ve) of the pole around which the individual plate rotates anti-clockwise when viewed from above relative to an external reference frame, the second element is the longitude (east is +ve, west is -ve), and the third element is the angular speed (degrees per Myr). The data below are from the NNR-MORVEL56 model of Argus and others (2011).

```
nb = {47.68, -68.44, 0.292}; sa = {-22.62, -112.83, 0.109};
```

The previous data are converted into angular velocity vectors, and the angular velocity vector of one plate as observed from the other plate is computed.

```
eΩnb = nb[[3]] * convert2Cart[nb[[1]], nb[[2]]];
eΩsa = sa[[3]] * convert2Cart[sa[[1]], sa[[2]]]; nbΩsa = eΩsa - eΩnb;
```

The ridge axis to be modeled is in the south-central Atlantic Ocean Basin, east-northeast of Ascension Island. Geographic locations were obtained from GeoMapApp (www.geomapapp.org). The first column below consists of latitudes, and the second is a set of longitudes. The first and fourth records are the midpoints of transform faults on either end of the ridge axis, and the second and third records are points along the ridge axis.

$$\textbf{initialGeog} = \begin{pmatrix} -6.892 & -12.158 \\ -7.131 & -13.019 \\ -7.317 & -13.017 \\ -7.360 & -13.201 \end{pmatrix};$$

The previous data are converted to unit Cartesian location vectors below.

```
t1o = convert2Cart[initialGeog[[1, 1]], initialGeog[[1, 2]]];
r1o = convert2Cart[initialGeog[[2, 1]], initialGeog[[2, 2]]];
r2o = convert2Cart[initialGeog[[3, 1]], initialGeog[[3, 2]]];
t2o = convert2Cart[initialGeog[[4, 1]], initialGeog[[4, 2]]];
```

The previous set of location vectors are collected in a single matrix (**initialCart**), in which the first and fourth records correspond to the midpoints of the transform faults, and the second and third records correspond to the two points along the ridge axis.

```
initialCart = {t1o, r1o, r2o, t2o};
```

Finally, we input a list that contains some of the youngest geomagnetic reversal ages, from the geomagnetic time scale of Cande and Kent (1995).

```
gts = {0.000, 0.780, 0.990, 1.070, 1.770,
   1.950, 2.140, 2.150, 2.581, 3.040, 3.110, 3.220,
   3.330, 3.580, 4.180, 4.620, 4.800, 4.890, 4.980};
```

Find a midpoint after a small rotation

The first argument in the user - defined function **buildIsochrons** is the matrix of initial location vectors (**initialCart**), the second and third are the angular velocity vectors for both plates relative to an external reference frame (**eΩnb** and **eΩsa**), the fourth is the name of the geomagnetic time scale file (**gts**), the fifth is the number of isochrons to be computed (2), and the sixth (-0.01) is the duration of each iteration step used to find the isochrons.

```
isochronFile = buildIsochrons[initialCart, eΩnb, eΩsa, gts, -0.01];
```

Finding the endpoints of the ridge axis

What we want to retain as a final result are the location vectors for the endpoints of each ridge axis.

Now the task is to find points along the great circle defined by the ridge axis that are, respectively, the same angular distance away from the pole of relative motion between the plates as the two transform-fault midpoints. We employ the user-defined function findRidgeEnds to perform this task.

```
isochronEnds = Table[findRidgeEnds[isochronFile[[i]], nbΩsa],
   {i, 1, Length[isochronFile]}];
```

```
transformA = {initialCart[[1]], isochronEnds[[1, 1]]};
```

```
transformB = {initialCart[[4]], isochronEnds[[1, 2]]};
```

Finding the position of the isochron on each plate

```
isochrons = makeIsochronFile[isochronEnds, eΩsa, eΩnb, gts];
```

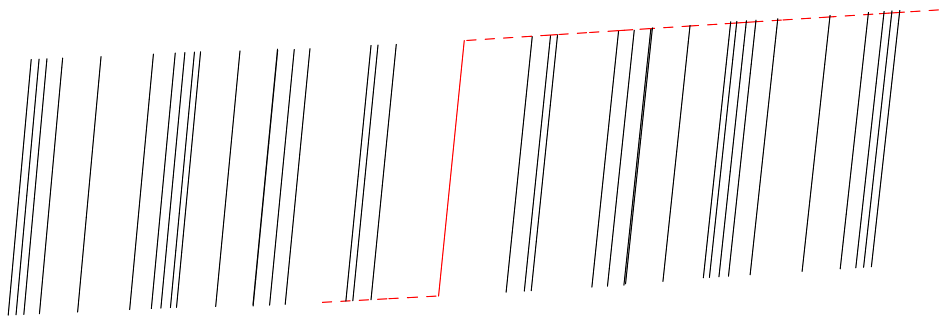
```
Dimensions[%]
```

```
{37, 2, 3}
```

Look at the results

```
prepGraphOut = Graphics3D[
   {Black, Table[Line[isochrons[[i]]], {i, 2, Length[isochrons]}],
    Red, Line[isochrons[[1]]], Dashed, Line[transformA],
    Line[transformB]}, AspectRatio → 1, Boxed → False];
```

```
Show[prepGraphOut]
```



The solid red line is the current ridge axis. The dashed red lines represent the current positions of the transform faults that bound the current ridge axis. The black lines are the model isochrons defined by geomagnetic reversals.

References

Argus, D.F., Gordon, R.G., and DeMets, C., 2011, Geologically current motion of 56 plates relative to the no-net-rotation reference frame: Geochemistry, Geophysics, Geosystems v. 12, no. 11, Q11001, doi:10.1029/2011GC003751.

Cande, S.C., and D.V. Kent, 1995, Revised calibration of the geomagnetic polarity timescale for the late Cretaceous and Cenozoic: Journal of Geophysical Research, v. 100, p. 6093-6095.