Zonal-Meridional-Velocity-Map.nb

 Written November 26, 2018, by Vince Cronin

A code to compute the meridional and zonal components of the instantaneous velocity of points along a plate boundary and to map them within a specified lat-long box

Introduction

The background of this analysis is explained in Cronin (1994).

This version contains data that is useful to investigate the North American-Nubian plate system.

Input

Angular velocities of the 2 - plate system

The following data describe the instantaneous motion of individual plates in an NNR reference frame (e.g., Argus, D.F., Gordon, R.G., and DeMets, C., 2011). A code to compute the meridional and zonal components of the instantaneous velocity of a point along a mid-o

NA refers to the North American plate, and NB refers to the Nubian plate.

```
latNA = -4.85; longNA = -80.64; angvelNA = 0.209;
```

```
latNB = 47.68; longNB = -68.44; angvelNB = 0.292;
```

The following data describe the instantaneous motion of one plate as observed from another plate (e.g., DeMets, C., Gordon, R.G., and Argus, D.F., 2010).

Angular velocities describe counter - clockwise (positive) rotation of the first listed plate relative to the plate listed second

```
latNBNA = 79.2; longNBNA = 40.2; angvelNBNA = 0.233;
```

Locations of the corner points of the map

```
latNWcorner = 38; longNWcorner = -60;
```

```
latSEcorner = 21; longSEcorner = -22;
```

User - defined functions

```
convert2cart[lat_, long_] := {Cos[lat Degree] Cos[long Degree],
   Cos[lat Degree] Sin[long Degree], Sin[lat Degree]};

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}];

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

tangentialVelocityComputer[latPole_, longPole_, latPoint_, longPoint_, angVel_] :=
  Module[{a, b, c, d, e, answer}, a = {Cos[latPole Degree] Cos[longPole Degree],
     Cos[latPole Degree] Sin[longPole Degree], Sin[latPole Degree]};
   b = {Cos[latPoint Degree] Cos[longPoint Degree],
     Cos[latPoint Degree] Sin[longPoint Degree], Sin[latPoint Degree]};
   c = VectorAngle[a, b] / Degree;
   d = (angVel * 6371.01 * 2 * π * Sin[c Degree]) / 360;
   e = unitVect3D[Cross[a, b]] * d;
   answer = e;
   answer];

findTangentialVector[locVectPole_, locVectPoint_, angVel_] :=
  Module[{a, b, c, answer}, a = VectorAngle[locVectPole, locVectPoint] / Degree;
   b = (angVel * 6371.01 * 2 * π * Sin[a Degree]) / 360;
   c = unitVect3D[Cross[locVectPole, locVectPoint]] * b;
   answer = c;
   answer];
```

We now know the length of each side of a plane triangle (two plane triangles, in fact -- one for each end of the transform fault) and need to know the interior angles. Given a triangle whose sides have lengths a, b, and c (where the angle between sides a and b is C, etc.), the law of cosines holds that

$$c^2 = a^2 + b^2 - 2ab\,\text{Cos}[C]$$

Re-arranging to isolate the unknown quantity, the angle *C*,

$$C = \text{ArcCos}\left[\frac{a^2+b^2-c^2}{2\,ab}\right]$$

```
interiorAngles[a_, b_, c_] :=
  Module[{α, β, χ, result}, α = ArcCos[(b^2 + c^2 - a^2) / (2 * b * c)] / Degree;
   β = ArcCos[(a^2 + c^2 - b^2) / (2 * a * c)] / Degree;
   χ = ArcCos[(a^2 + b^2 - c^2) / (2 * a * b)] / Degree;
   result = {a, b, c, α, β, χ};
   result];

findTangentialVelocities[refPt_, poleNA_, angVelVectorNA_,
   poleNB_, angVelVectorNB_, poleNBNA_, angVelVectorNBNA_] :=
  Module[{tangVelVectNA, tangSpeedNA, tangVelVectNB, tangSpeedNB,
    tangRelVelVectNBNA, tangSpeedNBNA, firstStep, meridionalVel, zonalVel, answer},
   tangVelVectNA = findTangentialVector[poleNA, refPt, angVelVectorNA[[4]]];
   tangSpeedNA = Norm[tangVelVectNA];
   tangVelVectNB = findTangentialVector[poleNB, refPt, angVelVectorNB[[4]]];
   tangSpeedNB = Norm[tangVelVectNB];
   tangRelVelVectNBNA =
    findTangentialVector[poleNBNA, refPt, angVelVectorNBNA[[4]]];
   tangSpeedNBNA = Norm[tangRelVelVectNBNA];
   firstStep = interiorAngles[tangSpeedNA, tangSpeedNB, tangSpeedNBNA];
   meridionalVel = Abs[tangSpeedNA Sin[firstStep[[5]] Degree]];
   zonalVel = (tangSpeedNA Cos[firstStep[[5]] Degree]) - tangSpeedNBNA/2;
   answer = {Abs[meridionalVel], Abs[zonalVel]};
   answer];
```

Computation

Convert the input data from geographic coordinates (lat, long) to cartesian coordinates

```
poleNA = convert2cart[latNA, longNA];
```

```
poleNB = convert2cart[latNB, longNB];
```

```
poleNBNA = convert2cart[latNBNA, longNBNA];
```

Combine unit location vector coordinates with the corresponding angular speed

```
angVelVectorNA = {poleNA[[1]], poleNA[[2]], poleNA[[3]], angvelNA};
```

```
angVelVectorNB = {poleNB[[1]], poleNB[[2]], poleNB[[3]], angvelNB};
```

```
angVelVectorNBNA = {poleNBNA[[1]], poleNBNA[[2]], poleNBNA[[3]], angvelNBNA};
```

Prepare the cartesian coordinates of all nodal points within the map area

```
minLat = latSEcorner; maxLat = latNWcorner;

minLong = longNWcorner; maxLong = longSEcorner;

latRange = (latNWcorner - latSEcorner);

longRange = (longSEcorner - longNWcorner);

mapPointsLats = Table[minLat + i, {i, 0, latRange}];

latFileLength = Dimensions[mapPointsLats];

mapPointsLongs = Table[minLong + i, {i, 0, longRange}];

longFileLength = Dimensions[mapPointsLongs];

mapPointsGeogCoords = Flatten[Table[{mapPointsLats[[i]], mapPointsLongs[[j]]},
    {i, latFileLength[[1]]}, {j, longFileLength[[1]]}], 1];

temp = Dimensions[%]; fileLength = temp[[1]];

mapPtsCart =
  Table[N[convert2cart[mapPointsGeogCoords[[i, 1]], mapPointsGeogCoords[[i, 2]]]],
   {i, fileLength}];
```

Find the velocities

```
velocities = Table[findTangentialVelocities[mapPtsCart[[i]], poleNA, angVelVectorNA,
    poleNB, angVelVectorNB, poleNBNA, angVelVectorNBNA], {i, fileLength}];

finalMapSetMerid = Table[{mapPointsGeogCoords[[i, 2]],
    mapPointsGeogCoords[[i, 1]], velocities[[i, 1]]}, {i, fileLength}];
```

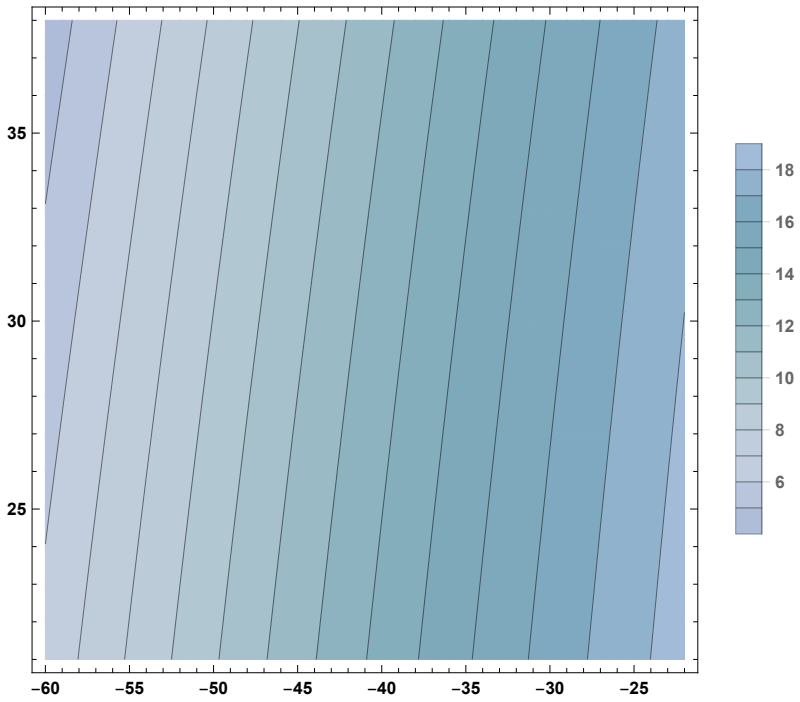
Output

The map presented below shows instantaneous tangential speed contours reflecting the velocity field for points along a plate boundary moving toward/away from the poles of relative motion between the two plates.

```
meridMap1 = ListDensityPlot[%, PlotLegends → Automatic];

meridMap1 = ListContourPlot[finalMapSetMerid,
   Contours → {4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20},
   PlotLegends → Automatic, ColorFunction → (ColorData["Aquamarine"])];
```

```
Show[%]
```

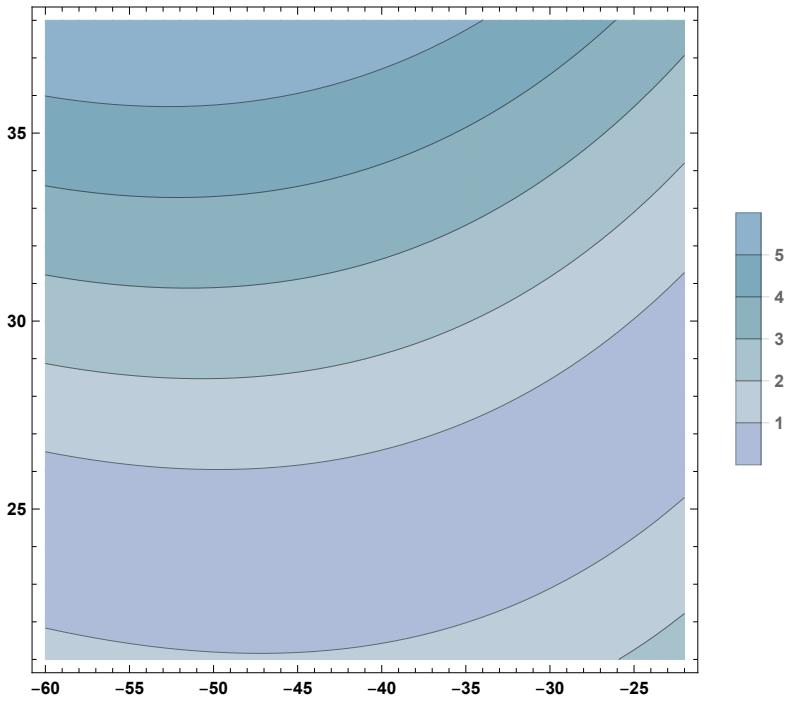


The map presented below shows instantaneous tangential speed contours reflecting the velocity field for points along a plate boundary moving around the poles of relative motion between the two plates.

```
finalMapSetZonal = Table[{mapPointsGeogCoords[[i, 2]],
    mapPointsGeogCoords[[i, 1]], velocities[[i, 2]]}, {i, fileLength}];

zonalMap1 = ListContourPlot[finalMapSetZonal, PlotLegends → Automatic,
   Contours → {1, 2, 3, 4, 5, 6, 7}, ColorFunction → (ColorData["Aquamarine"])];
```

Show [%]



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(11), Q1101, doi:10.1029/2011GC003751.

Cronin, V.S., 1994, Instantaneous velocity of mid-ocean ridges: Tectonophysics, v. 230, p. 151-159.

DeMets, C., Gordon, R.G., and Argus, D.F., 2010, Geologically current plate motions: Geophysical Journal International, v. 181, p. 1-80, doi:10.1111/j.1365-246X.2010.04491.x.