AfarTJ-22MyrTo10Myr.nb

Several points along the (mostly) non-extended edges of the Arabian, Nubian, and Somalian plates at the Afar triple junction have been selected and grouped with the rift that they are adjacent to: Red Sea rift, East African rift, Aden rift.
These points are moved back to where they were 22 million years ago, or about the time the East Africa rift started spreading.
We assume that the current angular velocity vectors for the Arabian, Nubian, and Somalian plates have been constant for the past 22 Myr and will continue to be constant in the next few million years, during the time we are interested in modeling.

Input

In[1]:=
```
poleLatSM = 54.6131;
poleLongSM = -71.5876;
angSpeedSM = 0.225471;
```

In[4]:=
```
poleLatNB = 50.3991;
poleLongNB = -36.5919;
angSpeedNB = 0.189645;
```

In[7]:=
```
poleLatAR = 42.155;
poleLongAR = -0.270986;
angSpeedAR = 0.456179;
```

The following matrices contain the geographic coordinates of several points along the (mostly) non-extended edges of the plate referenced in the matrix name. Each record corresponds to a different boundary point. The first column contains decimal latitudes (°) and the second column contains decimal longitudes (°). The data were collected by visual interpretation of a Mercator map projection of a global DEM presented by GeoMapApp (www.geomapapp.org).

In[10]:=
$$\text{nubiaPtsToday} = \begin{pmatrix} 12.662 & 39.507 \\ 11.613 & 39.590 \\ 10.367 & 39.814 \\ 9.674 & 39.744 \\ 8.69657 & 39.4531 \\ 7.0467 & 37.8161 \end{pmatrix};$$

In[11]:=
```
noRecsNB = 6;
```

In[12]:=
$$\text{somaliaPtsToday} = \begin{pmatrix} 7.2350 & 38.8180 \\ 8.8330 & 40.5044 \\ 9.440 & 41.493 \\ 9.518 & 42.047 \\ 9.748 & 43.128 \end{pmatrix};$$

In[13]:=
```
noRecsSM = 5;
```

In[14]:=
```
arabiaPtsToday = {{17.095, 43.124}, {15.441, 43.370}, {14.179, 43.576}, {13.693, 45.035}, {13.885, 45.756}, {14.218, 46.986}};
```

In[15]:=
```
noRecsAR = 6;
```

User-Defined Functions

The function **unitVect3D** operates on a 3D vector (*vect_*) and outputs the Cartesian coordinates of the corresponding unit vector.

In[16]:=
```
unitVect3D[vect_] :=
  {(vect[[1]] / Norm[vect]), (vect[[2]] / Norm[vect]), (vect[[3]] / Norm[vect])};
```

In[17]:=
```
zRotation[w_, dT_] := {{Cos[(w dT) Degree], -Sin[(w dT) Degree], 0},
  {Sin[(w dT) Degree], Cos[(w dT) Degree], 0}, {0, 0, 1}};
```

The function **geog2Cart** operates on two geographic coordinates (*lat_* and *long_*) and outputs the Cartesian coordinates of the unit location vector to that point.

In[18]:=
```
geog2Cart[lat_, long_] := {Cos[lat Degree] Cos[long Degree],
  Cos[lat Degree] Sin[long Degree], Sin[lat Degree]};
```

The function **cart2Geog** operates on a 3D unit location vector (*vect_*) and outputs the geographic coordinates of that point in latitude and longitude.

In[19]:=
```
cart2Geog[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}];
```

The function **distAzFromVectors** operates on a 3D unit location vector to a reference point (*refPtVect_*) and a 3D unit location vector to another point (*otherPtVect_*), and outputs the distance in km and azimuth in degrees from the reference point to the other point.

In[20]:=
```
distAzFromVectors[refPtVect_, otherPtVect_] :=
  Module[{radEarth, circEarth, northPole, θ1, θ2, θ3, θ4, normVect1,
    normVect2, normVect3, distKm, bearing, answer}, radEarth = 6371.02;
   circEarth = 2 * π * radEarth;
   northPole = {0, 0, 1};
   normVect1 = Cross[northPole, refPtVect];
   normVect2 = Cross[otherPtVect, refPtVect];
   θ1 = VectorAngle[refPtVect, otherPtVect];
   distKm = (θ1 / (2 * π)) * circEarth;
   θ2 = (180 / π) * (VectorAngle[normVect1, normVect2]);
   θ3 = (180 / π) * (VectorAngle[normVect1, otherPtVect]);
   normVect3 = Cross[refPtVect, normVect1];
   θ4 = (180 / π) * (VectorAngle[normVect3, otherPtVect]);
   bearing =
    If[((θ3 == 90) ∨ (θ3 == 180)), If[(θ4 <= 90), 0, 180], If[(θ3 > 90), (360 - θ2), θ2]];
   answer = {distKm, bearing};
   answer];
```

The function **circMotion** operates on a 3D unit location vector (*x_*), a coordinate transformation matrix (*m1_*), an angular speed (*w_*) and a time interval (*dT_*) in Myr, and outputs the unit location vector to that point after it rotates around its axis.

In[21]:=
```
circMotion[x_, m1_, w_, dT_] := Module[{m2, m3, answer}, m2 = zRotation[w, dT];
   m3 = Inverse[m1];
   answer = m3.m2.m1.x;
   answer];
```

The function **spherePlot1** operates on a set of 3D unit location vectors of computed data (*inData_*) and two sets of 3D unit location vectors of colored markers (*markers1_* and *markers2_*) to help establish the coordinate system(s), and outputs a graphic of a unit sphere on which the input points are plotted.

In[22]:=
```
spherePlot1[inData_, markers1_, markers2_] :=
  Module[{g1, g2, g3, g4, answer}, g1 = Graphics3D[Sphere[{0, 0, 0}, 1],
     AspectRatio → 1, BoxRatios → {1, 1, 1}, PlotRange → All,
     PlotRangePadding → 0.1, ColorOutput → GrayLevel, Lighting → "Neutral"];
   g2 = ListPointPlot3D[inData, AspectRatio → 1, BoxRatios → {1, 1, 1},
     PlotStyle → Blue, PlotRange → All, PlotRangePadding → 0.1];
   g3 = ListPointPlot3D[markers1, AspectRatio → 1, BoxRatios → {1, 1, 1},
     PlotStyle → Red, PlotRange → All, PlotRangePadding → 0.1];
   g4 = ListPointPlot3D[markers2, AspectRatio → 1, BoxRatios → {1, 1, 1},
     PlotStyle → Green, PlotRange → All, PlotRangePadding → 0.1];
   answer = Show[g1, g2, g3, g4];
   answer];
```

In[23]:=
```
cycloid[obsΩ_, movΩ_, refPt_, t_] := Module[
    {a, b, c, d, e, f, g, h, i, j, k, l, m1, m2, m3, m4, m5, n, p, answer}, a = {1, 0, 0};
    b = {0, 1, 0};
    c = {0, 0, 1};
    d = unitVect3D[obsΩ];
    e = Norm[obsΩ];
    f = unitVect3D[movΩ];
    g = Norm[movΩ];
    h = unitVect3D[d × f];
    i = h;
    j = unitVect3D[h × d];
    k = unitVect3D[i × f];
    m1 =
     {{k[[1]], k[[2]], k[[3]]}, {i[[1]], i[[2]], i[[3]]}, {f[[1]], f[[2]], f[[3]]}};
    m2 = {{Cos[(-g Degree) t], Sin[(-g Degree) t], 0},
      {-Sin[(-g Degree) t], Cos[(-g Degree) t], 0}, {0, 0, 1}};
    l = VectorAngle[d, k];
    If[l < (π / 2), (n = 1), (n = (-1))];
    p = (n) (VectorAngle[f, d]);
    m3 = {{Cos[p], 0, -Sin[p]}, {0, 1, 0}, {Sin[p], 0, Cos[p]}};
    m4 = {{Cos[(e Degree) t], Sin[(e Degree) t], 0},
      {-Sin[(e Degree) t], Cos[(e Degree) t], 0}, {0, 0, 1}};
    m5 = {{j[[1]], h[[1]], d[[1]]}, {j[[2]], h[[2]], d[[2]]},
      {j[[3]], h[[3]], d[[3]]}};
    answer = m5.m4.m3.m2.m1.refPt;
    answer];
```

The user-defined function `northArrow` takes the average of all the unit location vectors of points along the plate boundary (*viewPoint*) and a user-defined scale (*nArrowScale*), generally ranging from about 0.05 to 0.1, and finds a suitable endpoint for a vector pointing toward north in the output graphics.

In[24]:=
```
northArrow[viewPoint_, nArrowScale_] :=
  Module[{north, center, normal, northward, answer}, north = {0, 0, 1};
    center = {0, 0, 0};
    normal = Cross[north, viewPoint];
    northward = Cross[viewPoint, normal] * nArrowScale;
    answer = {center, northward};
    answer];
```

In[25]:=
```
midpoint[cartVect1_, cartVect2_] := Module[{a, answer}, a = (cartVect1 + cartVect2) / 2;
    answer = {(a[[1]] / Norm[a]), (a[[2]] / Norm[a]), (a[[3]] / Norm[a])};
    answer];
```

In[26]:=
```
intersection[vect1_, vect2_, vect3_, vect4_] :=
  Module[{a, b, c, d, answer}, a = Cross[vect1, vect2];
   b = Cross[vect3, vect4];
   c = Cross[b, a];
   d = {(c[[1]] / Norm[c]), (c[[2]] / Norm[c]), (c[[3]] / Norm[c])};
   answer = If[(VectorAngle[d, vect2] > (π / 2)), -d, d];
   answer];
```

In[27]:=
```
average3Vectors[vect1_, vect2_, vect3_] :=
  Module[{a, b, answer}, a = (vect1 + vect2 + vect3) / 3;
   answer = {(a[[1]] / Norm[a]), (a[[2]] / Norm[a]), (a[[3]] / Norm[a])};
   answer];
```

Computation

The model time is 22 Myr ago (-22).

In[28]:=
```
modelTime = -22;
```

Move each set of points back to where they would have been at the selected time with a circular motion around the respective plate pole.

Define the axes of the geographic coordinate system

In[29]:=
```
xGeog = {1, 0, 0};
```

In[30]:=
```
yGeog = {0, 1, 0};
```

In[31]:=
```
zGeog = {0, 0, 1};
```

Nubian Plate

Convert the input pole location data for the pole to the Nubian plate to Cartesian coordinates

In[32]:=
```
poleEulerNB = geog2Cart[poleLatNB, poleLongNB];
```

Define the angular speed "ω" for the Nubian plate

In[33]:=
```
ωNB = angSpeedNB;
```

Define the axes of the Euler pole coordinate system

In[34]:=
```
zPoleNB = poleEulerNB;
```

In[35]:=
```
xPoleNB = Cross[zPoleNB, zGeog];
```

In[36]:=
```
yPoleNB = Cross[zPoleNB, xPoleNB];
```

Find the Cartesian coordinates of the reference points on the Somalian plate

In[37]:=
```
cartNBtoday =
  Table[geog2Cart[nubiaPtsToday[[i, 1]], nubiaPtsToday[[i, 2]]], {i, 1, noRecsNB}];
```

Define the transformation matrix between the geographic coordinate system and the Euler pole coordinate system

In[38]:=
```
jNB = ( Dot[xGeog, xPoleNB]  Dot[yGeog, xPoleNB]  Dot[zGeog, xPoleNB]
        Dot[xGeog, yPoleNB]  Dot[yGeog, yPoleNB]  Dot[zGeog, yPoleNB]
        Dot[xGeog, zPoleNB]  Dot[yGeog, zPoleNB]  Dot[zGeog, zPoleNB] );
```

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[39]:=
```
rotPtsNB22mybp =
  Table[circMotion[cartNBtoday[[i]], jNB, ωNB, modelTime], {i, 1, noRecsNB}];
```

Somalian Plate

Convert the input pole location data for the pole to the Somalian plate to Cartesian coordinates

In[40]:=
```
poleEulerSM = geog2Cart[poleLatSM, poleLongSM];
```

Define the angular speed "ω" for the Somalian plate

In[41]:=
```
ωSM = angSpeedSM;
```

Define the axes of the Euler pole coordinate system

In[42]:=
```
zPoleSM = poleEulerSM;
```

In[43]:=
```
xPoleSM = Cross[zPoleSM, zGeog];
```

In[44]:=
```
yPoleSM = Cross[zPoleSM, xPoleSM];
```

Find the Cartesian coordinates of the reference points on the Somalian plate

In[45]:=
```
cartSMtoday = Table[
    geog2Cart[somaliaPtsToday[[i, 1]], somaliaPtsToday[[i, 2]]], {i, 1, noRecsSM}];
```

Define the transformation matrix between the geographic coordinate system and the Euler pole coordinate system

In[46]:=
```
jSM = ( Dot[xGeog, xPoleSM]  Dot[yGeog, xPoleSM]  Dot[zGeog, xPoleSM]
        Dot[xGeog, yPoleSM]  Dot[yGeog, yPoleSM]  Dot[zGeog, yPoleSM]
        Dot[xGeog, zPoleSM]  Dot[yGeog, zPoleSM]  Dot[zGeog, zPoleSM] );
```

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[47]:=
```
rotPtsSM22mybp =
  Table[circMotion[cartSMtoday[[i]], jSM, ωSM, modelTime], {i, 1, noRecsSM}];
```

Arabian Plate

Convert the input pole location data for the pole to the Arabian plate to Cartesian coordinates

In[48]:=
```
poleEulerAR = geog2Cart[poleLatAR, poleLongAR];
```

Define the angular speed "ω" for the Arabian plate

In[49]:=
```
ωAR = angSpeedAR;
```

Define the axes of the Euler pole coordinate system

In[50]:=
```
zPoleAR = poleEulerAR;
```

In[51]:=
```
xPoleAR = Cross[zPoleAR, zGeog];
```

In[52]:=
```
yPoleAR = Cross[zPoleAR, xPoleAR];
```

Find the Cartesian coordinates of the reference points on the Arabian plate

In[53]:=
```
cartARtoday = Table[
   geog2Cart[arabiaPtsToday[[i, 1]], arabiaPtsToday[[i, 2]]], {i, 1, noRecsAR}];
```

Define the transformation matrix between the geographic coordinate system and the Euler pole coordinate system

In[54]:=
```
jAR = ( Dot[xGeog, xPoleAR]  Dot[yGeog, xPoleAR]  Dot[zGeog, xPoleAR]
        Dot[xGeog, yPoleAR]  Dot[yGeog, yPoleAR]  Dot[zGeog, yPoleAR]
        Dot[xGeog, zPoleAR]  Dot[yGeog, zPoleAR]  Dot[zGeog, zPoleAR] );
```

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[55]:=
```
rotPtsAR22mybp =
  Table[circMotion[cartARtoday[[i]], jAR, ωAR, modelTime], {i, 1, noRecsAR}];
```

Finding the Initial Triple Junction

Red Sea Rift

In[56]:=
```
midpt1at22mybp = midpoint[rotPtsNB22mybp[[1]], rotPtsAR22mybp[[1]]];
```

In[57]:=
```
midpt2at22mybp = midpoint[rotPtsNB22mybp[[3]], rotPtsAR22mybp[[3]]];
```

East Africa Rift

In[58]:=
```
midpt3at22mybp = midpoint[rotPtsNB22mybp[[5]], rotPtsSM22mybp[[2]]];
```

In[59]:=
```
midpt4at22mybp = midpoint[rotPtsNB22mybp[[6]], rotPtsSM22mybp[[1]]];
```

Aden Rift

In[60]:=
```
midpt5at22mybp = midpoint[rotPtsSM22mybp[[5]], rotPtsAR22mybp[[6]]];
```

In[61]:=
```
midpt6at22mybp = midpoint[rotPtsSM22mybp[[3]], rotPtsAR22mybp[[4]]];
```

Intersections and mean triple junction

The method of finding the triple junction use here is to find the mean of the three riftline intersections. Consequently, no microplate is modeled to grow in this triple junction over time.

```
In[62]:= tj1 = intersection[midpt4at22mybp, midpt3at22mybp, midpt2at22mybp, midpt1at22mybp];

In[63]:= tj2 = intersection[midpt5at22mybp, midpt6at22mybp, midpt3at22mybp, midpt4at22mybp];

In[64]:= tj3 = intersection[midpt1at22mybp, midpt2at22mybp, midpt6at22mybp, midpt5at22mybp];

In[65]:= meanTJcart22mybp = average3Vectors[tj1, tj2, tj3];

In[66]:= riftTJpoints22mybp =
           {midpt1at22mybp, midpt2at22mybp, meanTJcart22mybp, midpt3at22mybp, midpt4at22mybp,
            midpt3at22mybp, meanTJcart22mybp, midpt6at22mybp, midpt5at22mybp};

In[67]:= nubiaPtsCart22mybp = {midpt1at22mybp,
            midpt2at22mybp, meanTJcart22mybp, midpt3at22mybp, midpt4at22mybp};

In[68]:= nubiaPtsGeog22mybp =
           Table[cart2Geog[nubiaPtsCart22mybp[[i]]], {i, 1, Length[nubiaPtsCart22mybp]}];

In[69]:= somaliaPtsCart22mybp = {midpt4at22mybp,
            midpt3at22mybp, meanTJcart22mybp, midpt6at22mybp, midpt5at22mybp};

In[70]:= somaliaPtsGeog22mybp = Table[
            cart2Geog[somaliaPtsCart22mybp[[i]]], {i, 1, Length[somaliaPtsCart22mybp]}];

In[71]:= arabiaPtsCart22mybp = {midpt1at22mybp,
            midpt2at22mybp, meanTJcart22mybp, midpt6at22mybp, midpt5at22mybp};

In[72]:= arabiaPtsGeog22mybp =
           Table[cart2Geog[arabiaPtsCart22mybp[[i]]], {i, 1, Length[arabiaPtsCart22mybp]}];
```

Prepare the graphic output

Parameters that affect the graphic output

Set the size of the north arrow on the graphic.

```
In[73]:= northArrowScale = 0.1;
```

Choose the distance from Earth' s center (expressed in Earth radii) from which the boundary will be viewed in the graphic. The value assigned to this variable must be greater than 1.

```
In[74]:= viewDistance = 1.28;

In[75]:= centerOfMap = geog2Cart[11.5, 42.0];
```

Variable `averageVV` is the point from which the boundary is viewed looking toward the center of Earth, as plotted using the built-in *Mathematica* function `Graphics3D`

In[76]:=
```
averageVV = viewDistance * centerOfMap;
```

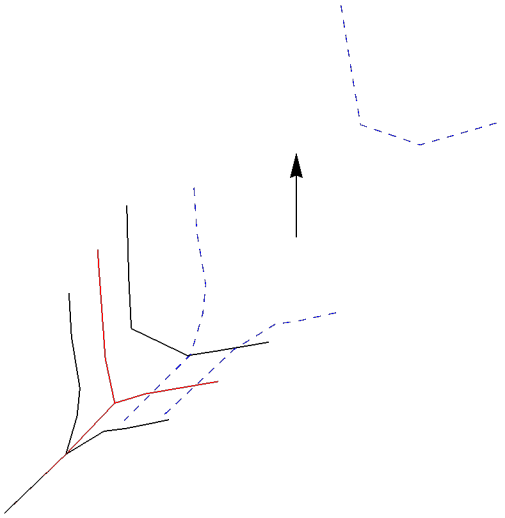
Code that creates the initial graphic output

Note : Some useful (easily visible) built - in colors for use in the graphics include the following : Red, Green, Blue, Black, Gray, Cyan, Magenta, Brown, Orange, Pink, and Purple.

In[77]:=
```
boundaryOut22mybp = Graphics3D[{Arrow[northArrow[averageVV, northArrowScale]],
     Black, Line[Table[rotPtsAR22mybp[[i]], {i, 1, Length[rotPtsAR22mybp]}]],
     Line[Table[rotPtsSM22mybp[[i]], {i, 1, Length[rotPtsSM22mybp]}]],
     Line[Table[rotPtsNB22mybp[[i]], {i, 1, Length[rotPtsNB22mybp]}]], Red,
     Line[Table[riftTJpoints22mybp[[i]], {i, 1, Length[riftTJpoints22mybp]}]],
     Blue, Dashed, Line[Table[cartARtoday[[i]], {i, 1, Length[cartARtoday]}]],
     Line[Table[cartSMtoday[[i]], {i, 1, Length[cartSMtoday]}]],
     Line[Table[cartNBtoday[[i]], {i, 1, Length[cartNBtoday]}]]},
   AspectRatio → Automatic, Boxed → False, ViewVector → {averageVV, {0, 0, 0}}];
```

In[78]:=
```
boundaryOut22mybp
```

Out[78]=



The graphic shown above shows the Afar triple junction (red) at -22 MYBP.

• The original triple junction at 22 MYBP is shown by the solid red lines in the lower left, and the solid black lines on either side are the edges of the current non-extended continental crust as rotated back over 22 Myr.

• The blue dashed lines are the edges of the current non-extended continental crust today.

In[79]:=
```
ClearAll[tj1, tj2, tj3];
```

Growing the triple junction from 22 MYBP to today

In[80]:=
```
modelTime0 = 22;
```

Nubian Plate

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Nubian plate 22 million years ago. The third record is the modeled location of the triple junction at that time.

In[81]:=
```
MatrixForm[nubiaPtsGeog22mybp]
```

Out[81]//MatrixForm=

11.1682	37.142
8.5461	37.3523
7.46535	37.58
6.13429	36.3305
4.50736	34.6802

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[82]:=
```
rotPtsNB0 = Table[circMotion[nubiaPtsCart22mybp[[i]], jNB, ωNB, modelTime0],
    {i, 1, Length[nubiaPtsCart22mybp]}];
```

Somalian Plate

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Somalian plate 22 million years ago. The third record is the modeled location of the triple junction at that time.

In[83]:=
```
MatrixForm[somaliaPtsGeog22mybp]
```

Out[83]//MatrixForm=

4.50736	34.6802
6.13429	36.3305
7.46535	37.58
7.71436	38.3363
8.05449	40.1239

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[84]:=
```
rotPtsSM0 = Table[circMotion[somaliaPtsCart22mybp[[i]], jSM, ωSM, modelTime0],
    {i, 1, Length[somaliaPtsCart22mybp]}];
```

Arabian Plate

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Arabian plate 22 million years ago. The third record is the modeled location of the triple junction at that time.

In[85]:=
```
MatrixForm[arabiaPtsGeog22mybp]
```

Out[85]//MatrixForm=

11.1682	37.142
8.5461	37.3523
7.46535	37.58
7.71436	38.3363
8.05449	40.1239

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[86]:=
```
rotPtsAR0 = Table[circMotion[arabiaPtsCart22mybp[[i]], jAR, ωAR, modelTime0],
    {i, 1, Length[arabiaPtsCart22mybp]}];
```

Red Sea Rift

In[87]:=
```
midpt1Today = midpoint[rotPtsNB0[[1]], rotPtsAR0[[1]]];
```

In[88]:=
```
midpt2Today = midpoint[rotPtsNB0[[2]], rotPtsAR0[[2]]];
```

East Africa Rift

In[89]:=
```
midpt3Today = midpoint[rotPtsNB0[[4]], rotPtsSM0[[2]]];
```

In[90]:=
```
midpt4Today = midpoint[rotPtsNB0[[5]], rotPtsSM0[[1]]];
```

Aden Rift

In[91]:=
```
midpt5Today = midpoint[rotPtsSM0[[5]], rotPtsAR0[[5]]];
```

In[92]:=
```
midpt6Today = midpoint[rotPtsSM0[[4]], rotPtsAR0[[4]]];
```

Intersections and mean triple junction

The method of finding the triple junction use here is to find the mean of the three riftline intersections. Consequently, no microplate is modeled to grow in this triple junction over time.

In[93]:=
```
tj1 = intersection[midpt4Today, midpt3Today, midpt2Today, midpt1Today];
```

In[94]:=
```
tj2 = intersection[midpt5Today, midpt6Today, midpt3Today, midpt4Today];
```

In[95]:=
```
tj3 = intersection[midpt1Today, midpt2Today, midpt6Today, midpt5Today];
```

In[96]:=
```
meanTJcartToday = average3Vectors[tj1, tj2, tj3];
```

In[97]:=
```
riftTJpoints0 = {midpt1Today, midpt2Today, meanTJcartToday, midpt3Today,
    midpt4Today, midpt3Today, meanTJcartToday, midpt6Today, midpt5Today};
```

In[98]:=
```
nubiaPtsCartToday =
  {midpt1Today, midpt2Today, meanTJcartToday, midpt3Today, midpt4Today};
```

In[99]:=
```
nubiaPtsGeogToday =
  Table[cart2Geog[nubiaPtsCartToday[[i]]], {i, 1, Length[nubiaPtsCartToday]}];
```

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Nubian plate today. The third record is the modeled location of the triple junction today.

In[100]:=
```
MatrixForm[nubiaPtsGeogToday]
```

Out[100]//MatrixForm=

14.8537	41.3544
12.2503	41.7206
11.0651	42.166
8.76515	39.9787
7.14112	38.3169

In[101]:=
```
somaliaPtsCartToday =
  {midpt4Today, midpt3Today, meanTJcartToday, midpt6Today, midpt5Today};
```

In[102]:=
```
somaliaPtsGeogToday =
  Table[cart2Geog[somaliaPtsCartToday[[i]]], {i, 1, Length[somaliaPtsCartToday]}];
```

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Somalian plate today. The third record is the modeled location of the triple junction today.

In[103]:=
```
MatrixForm[somaliaPtsGeogToday]
```

Out[103]//MatrixForm=

7.14112	38.3169
8.76515	39.9787
11.0651	42.166
11.511	43.3131
11.9189	45.1035

In[104]:=
```
arabiaPtsCartToday =
  {midpt1Today, midpt2Today, meanTJcartToday, midpt6Today, midpt5Today};
```

In[105]:=
```
arabiaPtsGeogToday =
  Table[cart2Geog[arabiaPtsCartToday[[i]]], {i, 1, Length[arabiaPtsCartToday]}];
```

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Arabian plate today. The third record is the modeled location of the triple junction today.

In[106]:=
```
MatrixForm[arabiaPtsGeogToday]
```

Out[106]//MatrixForm=

14.8537	41.3544
12.2503	41.7206
11.0651	42.166
11.511	43.3131
11.9189	45.1035

In[107]:=

Prepare the graphic output

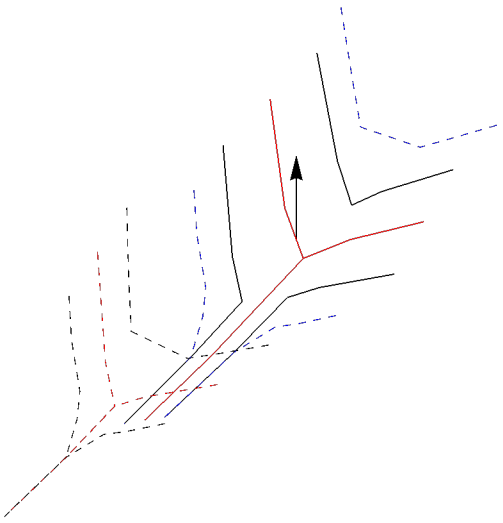
Code that creates the graphic output

Note : Some useful (easily visible) built - in colors for use in the graphics include the following : Red, Green, Blue, Black, Gray, Cyan, Magenta, Brown, Orange, Pink, and Purple.

In[108]:=
```
boundaryOut22mybp = Graphics3D[{Arrow[northArrow[averageVV, northArrowScale]],
    Black, Line[Table[rotPtsAR0[[i]], {i, 1, Length[rotPtsAR0]}]],
    Line[Table[rotPtsSM0[[i]], {i, 1, Length[rotPtsSM0]}]],
    Line[Table[rotPtsNB0[[i]], {i, 1, Length[rotPtsNB0]}]], Red,
    Line[Table[riftTJpoints0[[i]], {i, 1, Length[riftTJpoints0]}]], Blue,
    Dashed, Line[Table[cartARtoday[[i]], {i, 1, Length[cartARtoday]}]],
    Line[Table[cartSMtoday[[i]], {i, 1, Length[cartSMtoday]}]],
    Line[Table[cartNBtoday[[i]], {i, 1, Length[cartNBtoday]}]], Black,
    Dashed, Line[Table[rotPtsAR22mybp[[i]], {i, 1, Length[rotPtsAR22mybp]}]],
    Line[Table[rotPtsSM22mybp[[i]], {i, 1, Length[rotPtsSM22mybp]}]],
    Line[Table[rotPtsNB22mybp[[i]], {i, 1, Length[rotPtsNB22mybp]}]], Red, Dashed,
    Line[Table[riftTJpoints22mybp[[i]], {i, 1, Length[riftTJpoints22mybp]}]]},
  AspectRatio → Automatic, Boxed → False, ViewVector → {averageVV, {0, 0, 0}}];
```

In[109]:=
```
boundaryOut22mybp
```

Out[109]=



The graphic shown above shows the Afar triple junction (red) at -22 MYBP and today.

• The original triple junction at 22 MYBP is the dashed red line in the lower left, and the dashed black lines on either side are the edges of the current non-extended continental crust as rotated back over 22 Myr.

• The solid red lines in the middle of the diagram, with the black north arrow sticking out of them, is the current (model) triple junction.They are flanked by solid black lines, which are the displaced plate boundary from 22 MYBP (the red dashed lines). The blue dashed lines are the edges of the current non -

extended continental crust today. The area between the solid black lines here is the new oceanic crust + extended continental crust of the last 22 Myr.

In[110]:=
```
ClearAll[tj1, tj2, tj3];
```

Growing the triple junction from Today to a Future Time

In[111]:=
```
modelTimeF = 10;
```

Nubian Plate

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[112]:=
```
rotPtsNBF = Table[circMotion[nubiaPtsCartToday[[i]], jNB, ωNB, modelTimeF],
    {i, 1, Length[nubiaPtsCartToday]}];
```

Somalian Plate

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[113]:=
```
rotPtsSMF = Table[circMotion[somaliaPtsCartToday[[i]], jSM, ωSM, modelTimeF],
    {i, 1, Length[somaliaPtsCartToday]}];
```

Arabian Plate

Rotate the reference point around the Euler pole by an angle equal to the angular speed multiplied by the time interval.

In[114]:=
```
rotPtsARF = Table[circMotion[arabiaPtsCartToday[[i]], jAR, ωAR, modelTimeF],
    {i, 1, Length[arabiaPtsCartToday]}];
```

Red Sea Rift

In[115]:=
```
midpt1F = midpoint[rotPtsNBF[[1]], rotPtsARF[[1]]];
```

In[116]:=
```
midpt2F = midpoint[rotPtsNBF[[2]], rotPtsARF[[2]]];
```

East Africa Rift

In[117]:=
```
midpt3F = midpoint[rotPtsNBF[[4]], rotPtsSMF[[2]]];
```

In[118]:=
```
midpt4F = midpoint[rotPtsNBF[[5]], rotPtsSMF[[1]]];
```

Aden Rift

In[119]:=
```
midpt5F = midpoint[rotPtsSMF[[5]], rotPtsARF[[5]]];
```

In[120]:=
```
midpt6F = midpoint[rotPtsSMF[[4]], rotPtsARF[[4]]];
```

Intersections and mean triple junction

The method of finding the triple junction use here is to find the mean of the three riftline intersections. Consequently, no microplate is modeled to grow in this triple junction over time.

In[121]:=
```
tj1 = intersection[midpt4F, midpt3F, midpt2F, midpt1F];
```

In[122]:=
```
tj2 = intersection[midpt5F, midpt6F, midpt3F, midpt4F];
```

In[123]:=
```
tj3 = intersection[midpt1F, midpt2F, midpt6F, midpt5F];
```

In[124]:=
```
meanTJcartF = average3Vectors[tj1, tj2, tj3];
```

In[125]:=
```
riftTJpointsF = {midpt1F, midpt2F, meanTJcartF,
    midpt3F, midpt4F, midpt3F, meanTJcartF, midpt6F, midpt5F};
```

In[126]:=
```
nubiaPtsCartF = {midpt1F, midpt2F, meanTJcartF, midpt3F, midpt4F};
```

In[127]:=
```
nubiaPtsGeogF = Table[cart2Geog[nubiaPtsCartF[[i]]], {i, 1, Length[nubiaPtsCartF]}];
```

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Nubian plate after "modelTimeF" million years in the future. The third record is the modeled location of the triple junction at that time.

In[128]:=
```
MatrixForm[%]
```

Out[128]//MatrixForm=

16.5958	43.2261
14.0024	43.6612
12.7305	44.2788
9.95784	41.6477
8.33672	39.9779

In[129]:=
```
somaliaPtsCartF = {midpt4F, midpt3F, meanTJcartF, midpt6F, midpt5F};
```

In[130]:=
```
somaliaPtsGeogF =
  Table[cart2Geog[somaliaPtsCartF[[i]]], {i, 1, Length[somaliaPtsCartF]}];
```

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Somalian plate after "modelTimeF" million years in the future. The third record is the modeled location of the triple junction at that time.

In[131]:=
```
MatrixForm[%]
```

Out[131]//MatrixForm=

8.33672	39.9779
9.95784	41.6477
12.7305	44.2788
13.291	45.5535
13.7265	47.349

In[132]:=
```
arabiaPtsCartF = {midpt1F, midpt2F, meanTJcartF, midpt6F, midpt5F};
```

In[133]:=
```
arabiaPtsGeogF =
  Table[cart2Geog[arabiaPtsCartF[[i]]], {i, 1, Length[arabiaPtsCartF]}];
```

The following table contains the latitude (first column) and longitude (second column) for the 5 model points on the Arabian plate after "modelTimeF" million years in the future. The third record is the modeled location of the triple junction at that time.

In[134]:=
```
MatrixForm[%]
```

Out[134]//MatrixForm=

16.5958	43.2261
14.0024	43.6612
12.7305	44.2788
13.291	45.5535
13.7265	47.349

In[135]:=

Prepare the graphic output

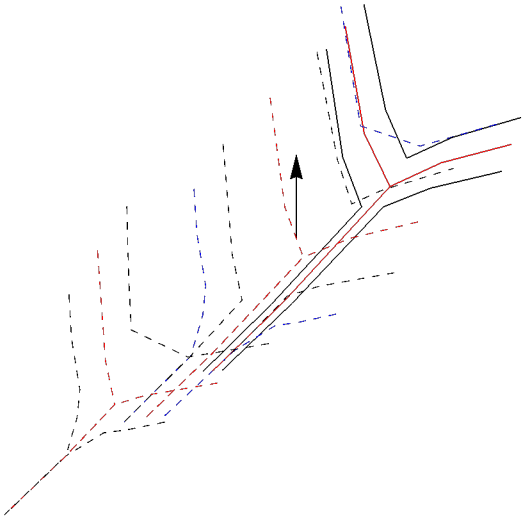
Code that creates the graphic output

Note : Some useful (easily visible) built - in colors for use in the graphics include the following : Red, Green, Blue, Black, Gray, Cyan, Magenta, Brown, Orange, Pink, and Purple.

In[136]:=
```
boundaryOutF = Graphics3D[{Arrow[northArrow[averageVV, northArrowScale]],
     Black, Line[Table[rotPtsARF[[i]], {i, 1, Length[rotPtsARF]}]],
     Line[Table[rotPtsSMF[[i]], {i, 1, Length[rotPtsSMF]}]],
     Line[Table[rotPtsNBF[[i]], {i, 1, Length[rotPtsNBF]}]], Red,
     Line[Table[riftTJpointsF[[i]], {i, 1, Length[riftTJpointsF]}]], Blue,
     Dashed, Line[Table[cartARtoday[[i]], {i, 1, Length[cartARtoday]}]],
     Line[Table[cartSMtoday[[i]], {i, 1, Length[cartSMtoday]}]],
     Line[Table[cartNBtoday[[i]], {i, 1, Length[cartNBtoday]}]], Black,
     Dashed, Line[Table[rotPtsAR0[[i]], {i, 1, Length[rotPtsAR0]}]],
     Line[Table[rotPtsSM0[[i]], {i, 1, Length[rotPtsSM0]}]],
     Line[Table[rotPtsNB0[[i]], {i, 1, Length[rotPtsNB0]}]],
     Line[Table[rotPtsAR22mybp[[i]], {i, 1, Length[rotPtsAR22mybp]}]],
     Line[Table[rotPtsSM22mybp[[i]], {i, 1, Length[rotPtsSM22mybp]}]],
     Line[Table[rotPtsNB22mybp[[i]], {i, 1, Length[rotPtsNB22mybp]}]], Red,
     Dashed, Line[Table[riftTJpoints0[[i]], {i, 1, Length[riftTJpoints0]}]],
     Line[Table[riftTJpoints22mybp[[i]], {i, 1, Length[riftTJpoints22mybp]}]]},
    AspectRatio → Automatic, Boxed → False, ViewVector → {averageVV, {0, 0, 0}}];
```

In[137]:=
```
boundaryOutF
```

Out[137]=



The graphic shown above shows the Afar triple junction (red) at -22 MYBP, today, and 10 Myr in the future.

• The original triple junction at 22 MYBP is the dashed red line in the lower left, and the dashed black lines on either side are the edges of the current non-extended continental crust as rotated back over 22 Myr.

• The red dashed lines in the middle of the diagram, with the black north arrow sticking out of them, is the current (model) triple junction.They are flanked by black dashed lines, which are the displaced plate boundary from 22 MYBP. The blue dashed lines are the edges of the current non - extended continental crust today. The area between the dashed black lines here is the new oceanic crust + extended continental crust of the last 22 Myr.

• The solid red lines to the upper right is the ridge - triple - junction system 10 Myr in the future.The solid black lines are the displaced ridge - triple - junction system of today as displaced on each of the three plates over 10 Myr of finite displacement.

In[138]:=
```
ClearAll[tj1, tj2, tj3];
```